Yang, J. C. et al. (Eds.) (2018). Proceedings of the 26th International Conference on Computers in Education. Philippines: Asia-Pacific Society for Computers in Education

The Effect of Cognitive Styles and Guidance Strategies on Children's Performance in Learning Programming

Chia-Yen FENG[a*] & Ming-Puu CHEN[b]
Graduate Institute of Information and Computer Education, National Taiwan Normal University, Taiwan
*yeni0412@gmail.com

Abstract: This study aims to explore the impact of cognitive styles and guidance strategies on the learning effectiveness and attitude of primary school students learning programming design. With six-grade students as its subjects, this study employs the quasi-experimental research method of 2 x 2 factorial designs to study 106 valid samples. The independent variables are cognitive styles (FD vs. FI) and guidance strategies (question-guidance vs. completion problem-guidance), while dependent variables are learning effectiveness and attitude. This study finds that (1) in terms of the learning effectiveness of programming design, the FI cognitive style group is more effective than the FD group in applying the program acquired during the project assignment, whereas the completion problem-guidance group scored higher than the question-guidance group; (2) in terms of program learning attitude, the completion problem-guidance group holds a more positive attitude towards the benefit of learning programming design than the question-guidance group. It is suggested that future studies target the analysis of programming design strategy and problem-solving behavior, to further explore the thinking process of learners.

Keywords: *cognitive styles, guidance strategies, experienced learning*

1. Introduction

Programming design is regarded as one of the key abilities to solve real world problems (Grover & Pea, 2013). Using computational thinking to solve problems has become an essential skill for modern people. Computational thinking is a way of thinking via computers to solve problems systematically and logically. In particular, programming design is the best approach to developing computational thinking (Chen, Shen, Barth-Cohen, Jiang, Huang & Eltoukhy, 2017).

The process of programming design involves complex cognitive processes, since learners need to understand programming grammar and instructions as well as have problem-solving skills during the learning process. In particular, Govender and Grayson (2006) point out that for learners, problem-solving skills have the most influence on programming ability. However, programming design is not only to combine a number of instructions, but also represents a series of steps to solve problems (Chen, 2007). Beginners would encounter many challenges in learning programming language (Feng & Chen, 2014; McCall, 2016). For instance, in respect of problem solving, these learners often cannot find a clue to start with and apply the knowledge learnt, thus failing to enter the passage of solving the problem (Lye & Koh, 2014) and resulting in low interest in learning (West & Ross, 2002).

Hence, Soloway and Spohrer (2013) point out that the environment for beginners to learn programming language should be characteristics of simplicity and visualization. This study employs Scratch, a visual programming learning application developed by the Media Lab of MIT, as the tool for experiment. Scratch helps concretize abstract concepts through its visualized programming design interface and uses modes of representation to reduce the grammar that restricts students during programming design. Besides, its puzzle-style design allows easy operation, and enables

learners to avoid getting lost in dealing with wrong meaning or grammar. Instead, learners can focus on logical reasoning for problem solving during their programming design. Moreover, the dynamic presentation boosts students' learning interest and fits into the characteristics of aboriginals, such as processing diverse information, image preference, active exploration and interactive learning (Prensky, 2001).

Apart from that, cognitive style has a crucial impact on the learning of programming design (Mancy& Reid, 2004). Preferential differences in cognitive style may lead to distinctive structural knowledge among learners and directly affect the manner of information acquisition and problem handling. This, in turn, causes the difference in imitation, disassembly and modification. Under the information interaction among different individuals, the sharing and feedback mechanism may lead to different programming design strategies and problem-solving behaviors, which affect learning effectiveness. Therefore, teaching models can be predicted by understanding the individual differences in cognitive style among learners and analyzing possible operational process and learning effectiveness (Riding & Cheema, 1991).

Hence, this study is designed to explore cognitive style and prompt strategy as key support to guide learners to complete game design assignments. The following questions are studied:

(1) What impact does the combination of different cognitive styles and guidance strategies have on the learning effectiveness of programming design among learners?

(2) What impact does the combination of different cognitive styles and guidance strategies have on the learning attitude of programming design among learners?

2. Review of related studies

2.1 Experienced learning

Kolb (1984) emphasizes that learning is a process of constantly converting experience and creating knowledge, while knowledge is the outcome of comprehension and experience conversion. In this study by Abdulwahed and Nagy (2011), an experiential learning cycle was integrated into the teaching design, to allow learners to study chemistry concepts by doing experiments in a virtual laboratory. The result suggests that learners' reflection can be facilitated and achieve a better learning effectiveness under a feedback mechanism, which is composed of doing experiments and the virtual laboratory.

As shown in *Figure 1*, Kolb (1984), having considered that experiential learning is a complete cyclic process, divides learning into a four-stage cycle, which includes concrete experience (CE), reflective observation (RO), abstract conceptualization (AC), and active experience (AE). Specifically, (1) CE is to learn through perception, by integrating the operational experience from the real world and daily life into the learning situation, thus enabling learners to construct knowledge independently; (2) RO is to leverage different information and content to stimulate learners to observe, start reflection by comparing previous knowledge and experience, and try to relate to their experience to find a solution to the problem; (3) AC is to summarize and synthesize thoughts and experience during the thinking process, to form concepts as the most ideal solution to the problem; and (4) AE is for learners to consolidate the acquisition concept, leverage self-checks to determine whether the concept is correct, and apply such knowledge and reasoning to various other situations.

2.2 Cognitive Styles

Cognitive style is a personal trait. Witkin, Oltman, Raskin & Karp (1971) believes that cognitive style is the personal way of organizing and searching information, through which one can understand how a person applies his/her intelligence to learn. Cognitive style also represents the individual habit and preference of searching and describing information (Chen, 2002). The categorization of cognitive styles varies among different scholars, depending on their perspectives and analytical aspects. Nonetheless, out of numerous categorization of cognitive styles, the most

explored and studied one is the division into field independence (FI) and field dependence (FD) (Witkin, Moore, Goodenough & Cox, 1977). In 1971, Witkin put forward the Grouped Embedded Figure Test (GEFT) as a test instrument, which comprised 25 test items. In particular, subjects are required to identify the simple geometric figures that are embedded in complex ones within limited time; both the accuracy of identifying such simple figures and the number of identified ones are used to distinguish FI and FD (Witkin et al., 1977).In general, FI learners excel in constructing and organizing information in the hypermedia learning environment, as they are more capable of working out the content on their assignment according to its requirements and widely selecting usable clues for application. By comparison, FD learners perform poorly in information analysis and organization. While they are inclined to opt for a more passive learning approach, their FI counterparts are inclined to choose a more active learning method (Lin & Gayle, 1996).

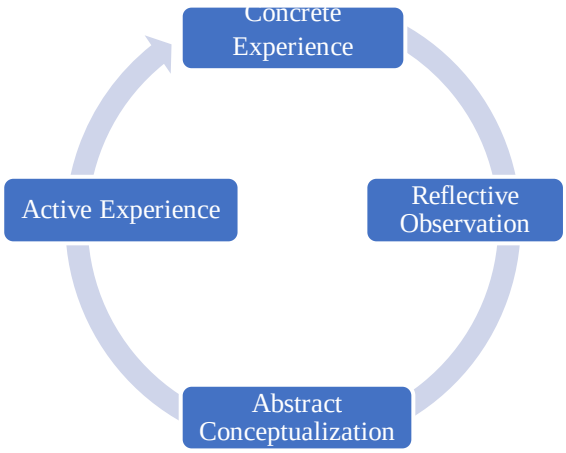


Figure 1. The experimental learning cycle helps cultivate diverse learning experience and improve learning effectiveness through four learning stages.

2.3 Guidance strategies

In teaching, guidance refers to providing learners with precise and complete information and relevant narrative knowledge, thereby helping learners to complete a task, address the problem and attain the objective during the problem-solving process (Clark, 2009). During knowledge transfer, learners need guidance during their practice and feedback during their performance of a task, so that learners are corrected immediately and apply such experience to other situational tasks (Hill & Hannafin, 2001; Lazonder & Harmsen, 2016; Matlen & Klahr, 2013). Textbook design can help to guide learners in their study, reduce their cognitive load during the cognitive process. Proper guidance and prompts can help the construction of abstract concepts and connect them to specific knowledge. Some researchers believe that an approach featuring a high level of guidance to help learners study knowledge and concepts step by step, can allow knowledge to be quickly and temporarily stored in short-term memory. Nevertheless, the knowledge cannot be transferred to long-term memory and applied to new situations; a low level of guidance can prompt learners to learn actively, reflect on problems and construct knowledge, thus achieving meaningful learning (Clark, 2009; Hill & Hannafin, 2001).

Hill and Hannafin (2001) raised four approaches of learning guidance, namely, (1) concept guidance: conceptual information relating to the learning content is provided to connect learners to the theme of learning; (2) question guidance: questions are raised to guide learners to think, examine actively, test and modify a given concept; (3) procedure guidance: learners are guided to think step by step, complete tasks, construct knowledge and attain a deeper level of cognitive development; and (4) strategy guidance: professional strategies, methods or suggestions are offered to guide learners to complete tasks. Based on various levels of prompts, this study focuses on two guidance strategies that guide learners in learning programming design, namely, question guidance (low-level

guidance) and completion-problem guidance (medium-level guidance) which are explored as follows.

2.3.1 Question guidance

Question guidance allows learners to properly reflect and explore the problem during the problem-solving process, seek the solution from the situation of the problem concerned, sort it out, construct knowledge and store it in long-term memory. Such systematic construction cultivates the problem-solving ability (Dean & Kuhn, 2007; Lazonder & Harmsen, 2016).

2.3.2 Completion problem- guidance

Van Merrie¨nboer and Sweller (2005) holds that beginners are not suitable to be taught to do direct programming and create new programs. Hence, he put forward the completion strategy, which emphasizes that beginners are offered "well-structured" programs for reading, modification and expansion. His research findings show that students who employed the completion strategy created better program templates and attained greater effectiveness in comprehending meanings thanks to the support of robust learning examples. Hence, completion problem refers to that textbooks present given conditions, target conditions and some steps to follow, and that learners should complete the problem and work out the answer by themselves based on the clues given to them. This serves as a bridge between worked examples and ordinary problem solving (Sweller, Van Merrienboer, & Paas, 1998).With the significance and application of the above guidance strategies in mind, one may conclude that it is helpful to offer learning guidance to learners. In textbook design, proper integration of guidance strategies helps learners in knowledge exploration and construction, reflection, and connection of abstract concepts with concrete knowledge during the cognitive process. This study has question guidance and completion-problem guidance as the strategies for guiding programming design, puts forward respective conceptual questions to help learners reflect and provide them with half-finished programs as prompts, and explores whether both guidance strategies can effectively improve learning effectiveness and attitude.

3. Methods

3.1 Participants

This study recruited 106 sixth-grade students (51 males and 55 females aged between 11 and 12 years) from six classes at an elementary school in northern Taiwan. They participated in activities to learn game design for 40 minutes per week over eight weeks. The domain knowledge included computer programming using Scratch programming language. All participants were novices to programming; however, they had spent four weeks observing the scripts for games made using Scratch before their actual hands-on experience.

3.2 Framework of research design

A quasi-experimental, 2 x 2 (cognitive styles x guidance strategies) factorial design was employed to investigate the impact of cognitive styles and guidance strategies on learners' performance and attitude to game programming. The independent variables were cognitive styles (field-independence vs. field-dependence) and guidance strategies (question-guidance vs. completion problem-guidance), structured in the experiential learning cycle and integrated into the four stages of this cycle—concrete experience, reflective observation, abstract conceptualization and active experience.

This study has two dependent variables, namely, "learning performance of programming design" and "learning attitude of programming design". The former refers to the outcome of a learner's performance of game programming design after experiment-based teaching, which comprises of (1) programming comprehension: the understanding of programming knowledge contained in programming blocks; and (2) programming application: the ability to achieve objectives through a proper combination of different programming blocks. "Learning attitude of

programming design" refers to a learner's view of the learning attitude towards programming design after experiment-based teaching, which comprises of (a) learning motivation: the learner's interest in, willingness and the extent of preference towards the course; (b) learning benefit: the extent of benefit from learning approaches and instruments perceived by the learner; and (c) learning satisfaction: the extent of satisfaction with learning approaches and instruments on the part of learners.

3.3 Procedures

First, participants underwent the Grouped Embedded Figure Test (GEFT) and a test on the preparatory knowledge of programming concepts. Then, they took an example-based practical course to analyze the elements of a game and construct such concepts as repeated execution, sequential and parallel programming, condition execution and variables. Afterwards, there were a total of four sessions for the special game design "Monkey Catching Bananas" (Figure 2), in which participants followed guidance strategies to complete their works step by step (Table 1). In the last session, participants took a test on programming concepts to show their project performance, and filled in a questionnaire composed of a learning attitude scale, which is designed to understand their learning motive, benefit and satisfaction.

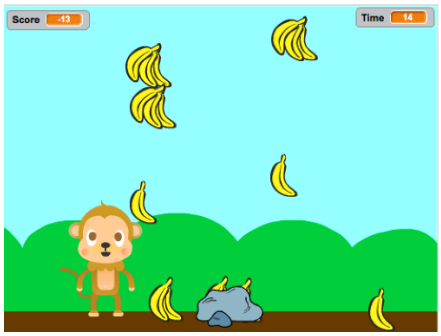
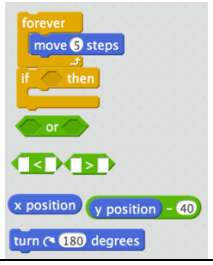


Figure 2. Game Design Sample, "Monkey Catching Bananas"

Table 1
Examples of guidance strategies

Question-guidance	Completion problem- guidance
Let the banana move to the right at the beginning. When you encounter the left and right edges (x position), let the banana fall in the opposite direction and move it one line down. Think about it: 1. How to move the banana to the right? Which direction is the direction? 2. Let the left and right edges of the banana fly in the opposite direction, and move down one line when touching the edge	Let the banana move to the right at the beginning. When you encounter the left and right edges (x position), let the banana fall in the opposite direction and move it one line down. 

3.4 Instructional design

In this study, learning activities are structured in an experiential learning cycle to enable learners to develop correct concepts on programming design through four steps, including "concrete experience", "reflective observation", "abstract conceptualization" and "active experience", and facilitate positive learning motivation and effectiveness. The special game design course "Monkey Catching Bananas" stimulated learners' interest. The course comprises of three tasks, with the objectives of each task and course content structure illustrated in Table 2.

Table 2
A Teaching Model Integrating "Monkey Catching Bananas" into the Experiential Learning Cycle

Step	Significance of Each Stage in the Cycle	Planned Tasks
Concrete experience	Learning through perception and by integrating the operational experience from the real world and daily life into learning situations, so that learners can construct knowledge independently.	Task 1: To enable the monkey to move From the Scratch example, students experience the abstract concepts of programming language (sequential and parallel programming, repeated execution, global variable, sequential structure, and conditional structure), set tasks and objectives, and identify the problem.
Reflective observation	In the course of the activity experience, you can think of ways to understand the existence or observation of doubts in your own problems, and try to find ways to solve problems by linking past experience.	Task 2: Monkey Catching Bananas 1. Through guidance, students undergo reflection and observation from constant trial and error during the programming design process, and acquire the correct game approach, understand the programming language and record it in their worksheets.
Abstract conceptualization	Generalizing and sorting out the thoughts and experience from the thinking process, and form concepts to serve as the ideal solution to the problem.	2. Students utilize toy blocks and task prompts to experience the abstract concepts of programming language in the game process. The teacher introduces programming concepts to guide students to develop their own combination of toy blocks, acquire correct abstract concepts and achieve the task objectives.
Active experience	Learners should be able to consolidate the acquisition concepts, do self-checks to determine whether a concept is correct, and apply such knowledge to problems in different situations.	Task 3: Monkey's Adventure in the New Amusement Park Discussion and sharing take place to modify programs and verify whether the structure and concept of programming language are correct.

3.5 Instruments

3.5.1 Test on the Learning Effectiveness of Programming Design

The test on the learning effectiveness of programming design mainly aims to assess the learner's comprehension and application of programming design concepts after learning programming design through guidance strategies. The test, with knowledge comprehension and application as its dimensions, comprises of 15 multiple-choice items, including 6 items on knowledge comprehension and 9 on knowledge application. The test content is identical to that on preparatory knowledge of programming design concepts, but the items and the order of choices are rearranged randomly. The Cronbach α coefficient was .847.

3.5.2 Project grading rubrics

Project grading rubrics were used to evaluate programming skills in the application of Scratch programming to design the game. The project grading rubrics comprised five aspects: correct programming, the completeness of programming, content creativity and the user's interaction with various styles of sprites (e.g., color and animation effects), appealing interface, and creative performance. When the project had been completed, each criterion in was evaluated on a scale of ten.

Three experts evaluated the projects developed by the participants. The grading correlation coefficient between the experts was 0.751 (Kendall's ω).

3.5.3 Learning attitude scale

The scale was divided into three dimensions: learning motivation, learning benefit, and learning satisfaction. Learning motivation measured the learners' interest and how much they enjoyed the game design courses; learning benefit measured the degree to which the learners found the course helpful; learning satisfaction measured students' satisfaction with the game design programming course. A Likert-type scale was adopted, ranging from "strongly agree" to "strongly disagree". The reliability coefficient was 0.78 (Cronbach's alpha).

4. Results

4.1 Influence of cognitive styles and guidance strategies on programming performance

As shown in Table 3, the two cognitive styles with completion problem-guidance obtained higher scores in programming comprehension, and the field-independence with completion problem-guidance obtained higher scores in programming application. Whether the differences between the mean of the two groups was statistically significant was further examined.

Table 3
Group means of game design performance

		Guidance strategies								
		Question-guidance			Completion problem-guidance			Total		
		M	SD	n	M	SD	n	M	SD	n
programming comprehension										
Cognitive styles	FI	59.81	21.79	28	70.54	17.45	26	**67.02**	19.44	54
	FD	63.65	21.43	26	70.38	16.70	26	65.37	20.22	52
	Total	67.02	19.43	54	70.46	17.05	52	66.18	19.76	106
Programming application										
Cognitive styles	FI	72.68	18.38	28	81.73	12.57	26	78.94	13.59	54
	FD	76.15	14.23	26	67.69	17.85	26	70.28	18.13	52
	Total	74.35	16.46	54	**74.71**	16.85	52	74.54	16.57	106

Table 4 reflects a prominent difference in the knowledge application effectiveness in relation to cognitive style ($F_{(1,102)}$ =7.93, p<. 05), as post hoc comparison finds that the mean of the FI group (M=67.02) is higher than that of the FD group (M=65.37); and there is also a prominent difference in the knowledge comprehension effectiveness in relation to guidance strategies ($F_{(1,102)}$ = 5.31, p<. 05), as post hoc comparison finds that the mean of the "completion problem-guidance" group (M=74.71) is higher than that of the "question-guidance" group (M=74.35).As to the reason for the above research results, learners who prefer field independence (FI) is most effective in programming comprehension, mainly because they are more active in programming study and able to dissemble and reassemble the existing programs when they observe the program example; during programming design, they understand the connection among programming codes, which benefits the ability to address similar problems in the future and the understanding of programming concepts. On the other hand, learners who took "completion problem-guidance" are more effective in programming application, which is consistent with Sweller et al. (1998). For such academic fields as programming design, it is more effective to teach beginners with examples than starting with problem-solving. Example-based learning is combined with "completion problems" designed in different ways, which can help beginners to gain from basic models when learning the examples and

can help students to better understand abstract concepts. Hence, they can score better performance in project assignments.

Table 4
MANOVA summary of the performance of cognitive styles *and* guidance strategies *on dependent measures*

Source	Dependent Variable	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
Cognitive styles	Programming comprehension	90.36	1	90.36	.24	.63	.002
	Programming application	2030.00	1	2030.00	7.93*	.016	.072
Guidance strategies	Programming comprehension	2017.28	1	2017.28	5.31*	.02	.049
	Programming application	2.31	1	2.31	.01	.93	.000
Cognitive styles × Guidance strategies	Programming comprehension	105.75	1	105.756	.28	.60	.003
	Programming application	738.46	1	738.46	2.89	.09	.028
Error	Programming comprehension	38765.04	102	380.059			
	Programming application	26098.15	102	255.86			

*p<.05

4.2 *Influence of cognitive styles and guidance strategies on learning attitude*

Table 5 shows that the average score of "question-guidance" learners is higher than their "completion problem-guidance" counterparts in terms of the learning motivation and satisfaction under programming learning attitude. However, a variance analysis is needed to further examine whether there is any difference.

Table 6 reflects a prominent difference in learning benefits in relation to guidance strategies ($F(1,102) = 4.82$, $p < .05$), as post hoc comparison finds that the benefit of learning programming design for the "completion problem-guidance" group (M=3.75) is significantly higher than that of the "question-guidance" group (M=3.26). This is presumably because the former observes the operation of programming examples and uses the assistance of worksheets to select the programming block that most approximates the programming example. By doing so, students can observe programming operation, acquire basic programming concepts, integrate the acquired concepts into actual programming blocks and convert them into concrete programming instructions. Therefore, perception improves the comprehension of programming concepts and benefits project work.

5. Conclusions

This study summarizes research results and concludes that (1) in respect of the learning effectiveness of programming design, the FI group is more effective than the FD group in applying the acquired program during project work, and the completion problem-guidance group is better than the question-guidance group; and (2) in respect of the learning attitude of programming design,

Table 5
Group means of game design learning attitude

	Question-guidance			Completion problem-guidance			Total		
	M	SD	n	M	SD	n	M	SD	n
Learning motivation									
FI	3.47	1.02	27	3.41	.87	38	3.43	.929	65
FD	**3.65**	.69	27	3.40	.59	14	3.56	.66	41
Total	3.56	.87	54	3.40	.80	52	3.49	.83	106
Leaning benefit									
FI	3.26	1.02	27	**3.75**	.98	38	3.54	1.02	65
FD	3.68	.936	27	3.28	.93	14	3.54	.94	41
Total	3.47	.99	54	3.62	.98	52	3.54	.98	106
Learning satisfaction									
FI	3.38	1.01	27	3.33	1.02	38	3.35	1.00	65
FD	**3.64**	.89	27	3.24	.79	14	3.50	.87	41
Total	3.51	.95	54	3.31	.99	52	3.41	.956	106

Table 6
MANOVA summary of the learning attitude for the cognitive styles *and* guidance strategies *on dependent measures*

Source	Dependent Variable	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
Cognitive styles	Learning motivation	.18	1	.18	.260	.61	.003
	Learning benefit	.02	1	.02	.022	.88	.000
	Learning satisfaction	.18	1	.18	.195	.66	.002
Guidance strategies	Learning motivation	.56	1	.563	.797	.37	.008
	Learning benefit	.04	1	.04	**4.82***	.03	.000
	Learning satisfaction	1.15	1	1.15	1.252	.27	.012
Cognitive styles × Guidance strategies	Learning motivation	.22	1	.22	.308	.58	.003
	Learning benefit	4.57	1	4.57	.042	.84	.045
	Learning satisfaction	.720	1	.72	.782	.38	.008
Error	Learning motivation	72.09	102	.71			
	Learning benefit	96.64	102	.95			
	Learning satisfaction	93.86	102	.92			

*p<.05

learners from the completion problem-guidance group hold a more positive attitude towards the learning benefit of programming design than the question-guidance group.

6. Limitations and future research

This study employs quantitative analysis to gauge the impact of learners' cognitive style and learning guidance strategy on the learning effectiveness and attitude of programming. It is suggested that future research can analyze the strategy and problem-solving behavior of programming design,

further explore learners' thinking process, and study the connection between the information conversion and learning effectiveness of learners during their disassembly process.

Acknowledgement

This study is supported by the Ministry of Science and Technology, Taiwan, R.O.C. under Grant MOST 106-2511-S-003 -049 –MY3.

References

Abdulwahed, M., & Nagy, Z. K. (2011). The TriLab, a novel ICT based triple access mode laboratory education model. *Computers & Education, 56*(1), 262–274.

Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers & Education, 109*, 162–175.

Clark, R. E. (2009). How much and what type of guidance is optimal for learning from instruction? *Constructivist Instruction: Success or Failure*, 158–183.

Dean Jr, D. & Kuhn, D. (2007). Direct instruction vs. discovery: The long view. *Science Education, 91*(3), 384–397.

Feng, C.Y., & Chen, M. P. (2014). The effects of goal specificity and scaffolding on programming performance and self-regulation in game design. *British Journal of Educational Technology, 45*(2), 285−302. (SSCI, DOI: 10.1111/bjet.12022).

Govender, I. & Grayson, D. (2006). Learning to program and learning to teach programming: Acloserlook. *Paper presented at the 2006 World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED-MEDIA2006)*, Orlando, Florida.

Grover, S., & Pea, R. (2013). Computational thinking in K-12: a review of the state of the field. *Educational Researcher*, *42*(1), 38−43.

Hill, J. R. & Hannafin, M. J. (2001). Teaching and learning in digital environments: The resurgence of resource-based learning. *Educational Technology Research and Development*, *49*(3), 37–52.

Kolb, D.A. (1984). Experiential learning: experience as the source of learning and development. Englewood Cliffs, NJ: Prentice Hall. Retrieved from https://academic.regis.edu/ed205/kolb.pdf.

Lin, H. & Gayle, V.D. (1996). Effects of linking structure and cognitive style on students' performance and attitude in a computer-based hypertext. Environment. *Journal of Educational Computing Research*, 15, 17–329.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*, 51–61.

Mancy, R., & Reid, N. (2004). Aspects of cognitive style and programming. In E. Dunican & T. Green (Eds.), *Proceedings of the Sixteenth Annual Workshop of the Psychology of Programming Interest Group* (PPIG '04; pp. 1–9). Carlow, Ireland: Institute of Technology.

McCall, D. (2016). *Novice Programmer Errors-Analysis and Diagnostics.* Doctor of Philosophy (PhD) thesis, University of Kent.

Prensky, M. (2001). Digital Natives, Digital Immigrants. On the Horizon, 9(5), 1–6.

Riding, R. J. & Cheema, I. (1991). Cognitive styles-an overview and integration. Educational Psychology, 11.3–4: 193–215.

Soloway, E., & Spohrer, J. C. (2013). *Studying the novice programmer*. Psychology Press.

Sweller, J., van Merriënboer, J.G., & Paas, F.G.W.C. (1998). Cognitive architecture and instructional design. *Educational Psychology Review, 10* (3), 251–297.

Van Merrie¨nboer, J. J. G., & Sweller, J. (2005). Cognitive load theory and complex learning: Recent developments and future directions. *Educational Psychology Review,*17, 147–177.

West, M., & Ross, S. (2002). Retaining females in computer science: A new look at a persistent problem. *Journal of Computing Sciences in Colleges*, *17*(5), 1–7.

Witkin, H. A., Moore, C. A., Goodenough, D. R., & Cox, P. W. (1977). Field-dependent and field-independent cognitive styles and their educational implications. *Review of Educational Research, 47*(1), 1–64.

Witkin, H. A., Oltman, P. K., Raskin, E., & Karp, S. A. (1971). *A manual for the embedded figures tests*. Palo Alto, CA: Consulting Psychologists Press.