JDCS
디지털콘텐츠학회논문지
Journal of Digital Contents Society
Vol. 25, No. 9, pp. 2517-2524, Sep. 2024

ChatGPT 활용이 비SW전공생 프로그래밍 학습에 미치는 효과성에 관한 연구

고 광 일*
우송대학교 테크노미디어융합학부 미디어디자인·영상전공 교수

Effectiveness of ChatGPT in Programming Education for Non-Computer Science Majors

Kwangil Ko*
Professor, School of Techno-Media Convergence, Woosong University, Daejeon 34606, Korea

[요 약]

인공지능 기술의 급속한 발전은 교육 분야에 획기적인 변화를 일으키고 있다. 특히 ChatGPT와 같은 생성형 인공지능 모델은 자연스러운 대화 능력을 바탕으로 자기주도적인 프로그래밍 학습에서 큰 가능성을 보여주고 있다. 본 연구는 비SW전공자를 대상으로 한 스크래치 수업에서 ChatGPT의 활용이 학습 성과에 미치는 영향을 조사하였다. ChatGPT를 사용하는 그룹과 그렇지 않은 그룹으로 나누어 동일한 평가를 실시한 후, ChatGPT 활용 그룹에 대한 설문조사를 진행한 결과, ChatGPT를 활용한 학습이 학습 효과를 유의미하게 증진시키는 것으로 나타났다. 특히 ChatGPT는 스크래치의 고급 기능 이해 및 알고리즘 학습과 같은 난이도 높은 학습 영역에서 뛰어난 효과를 보였다. 본 연구는 ChatGPT와 같은 생성형 인공지능이 프로그래밍 교육에서 효과적인 학습 도구로서의 가능성을 실증적으로 확인한 점에서 그 의미가 있다.

[Abstract]

The rapid advancement of artificial intelligence (AI) technology offers groundbreaking changes in education. Generative AI models such as ChatGPT, with their natural conversational abilities, show significant potential for self-directed programming learning. This study examined the impact of using ChatGPT on learning outcomes in a Scratch course designed for non-computer science majors. The participants were divided into two groups: one using ChatGPT and the other not using it, and both groups underwent the same assessments. After conducting a survey with the group using ChatGPT, the results indicated that using ChatGPT significantly enhanced learning effectiveness. Specifically, ChatGPT proved highly effective in more complex learning areas, such as understanding advanced Scratch functions and algorithm learning. This study is significant as it empirically demonstrates the potential of generative AI like ChatGPT as an effective tool in programming education.

색인어 : ChatGPT, 생성형 인공지능, 비SW전공, 프로그래밍 교육, 스크래치
Keyword : ChatGPT, Generative AI, Non-SW Major, Programming Education, Scratch

http://dx.doi.org/10.9728/dcs.2024.25.9.2517

Received 13 August 2024; **Revised** 04 September 2024
Accepted 23 September 2024

***Corresponding Author; Kwangil Ko**
Tel: +82-42-630-9343
E-mail: kwangil.ko@gmail.com

Ⅰ. Introduction

1-1 Research Background

The groundbreaking advancements in artificial intelligence technology are bringing innovative changes to the field of education. In particular, generative Artificial Intelligence (AI) models enable natural conversations with humans, presenting new approaches in educational curricula. The development of AI technology is also creating significant turning points in the field of programming education, where the potential of AI to help beginners understand and apply programming concepts is receiving considerable attention[1].

Among generative AI models, the most notable is ChatGPT, a conversational AI model developed by OpenAI based on the Generative Pre-trained Transformer (GPT) architecture[2]. ChatGPT has the capability for natural conversations with humans, allowing it to perform various tasks in education through interactions with learners, such as explaining concepts and assisting in problem-solving. As GPT has been upgraded, its multimodal conversation abilities have also advanced. GPT-3, released in 2022, supported text-centric conversation methods, but GPT-4, released in 2023, included image interpretation capabilities. Recently, GPT-4o, released in May 2024, also supports real-time voice conversations in various languages. These advancements in multimodal conversation abilities further enhance the applicability of ChatGPT in learner-centered, self-directed education.

Scratch, developed by Massachusetts Institute of Technology (MIT) Media Lab, is a visual programming language where programs are created by assembling blocks. It provides an environment where beginner learners can easily acquire computer science concepts and programming skills[3]. As of 2022, over 90 million people worldwide use Scratch, and more than 100 million projects have been developed and shared[4].

1-2 Related Research

Various studies have been conducted on the use of generative AI in programming education. Soohwang Lee et al. explored the trends in domestic research on the educational utilization of generative AI, suggesting the necessity and implications of comprehensive studies on its educational applications[5]. Seulki Kim analyzed the potential of using ChatGPT in programming education and developed prompts applying teaching-learning strategies and prompt engineering techniques for code generation in programming education[6],[7]. Jungoh Park examined the changes in learners' experiences and perceptions regarding the use of AI chatbots in programming classes[8]. Byeongchan Kong et al. evaluated the potential and practicality of code generated by ChatGPT compared to human-written code[9]. Suzy Choi et al. conducted a comparative analysis of problem-solving abilities in coding tasks between human programmers and ChatGPT-3.5, ChatGPT-4.0, and proposed an evaluation framework for ChatGPT coding assignments[10].

1-3 Research Subject

This study was conducted in relation to basic software education for non-Software (SW) majors, which is an important area of the Software-Centered University Project. In the 21st century, software competency has become a key skill, regardless of one’s major. Against this backdrop, this study explores the effectiveness of ChatGPT on programming learning for non-SW majors. Specifically, it divides a Scratch class for non-SW majors into two groups: one utilizing ChatGPT and the other following traditional teaching methods. The learning outcomes of both groups were statistically analyzed using identical assessments. Additionally, a survey was conducted with the ChatGPT-utilzing group to evaluate the utility of ChatGPT and its helpfulness across different Scratch learning domains.

The structure of the subsequent chapters of the paper is as follows: The chapter 2 analyzes the applicability of ChatGPT in various Scratch learning domains to determine the scope of Scratch learning where ChatGPT can be effectively used. The chapter 3 introduces the experimental method of this study and analyzes the initial learning capabilities of both groups before the experiment. The chapter 4 statistically analyzes the differences in learning outcomes between the two groups based on the experiment results and evaluates the utility of ChatGPT through the survey. Finally, the chapter 5 summarizes the study, discusses its significance, and suggests future research directions, concluding the paper.

Ⅱ. Utilization of ChatGPT in Scratch Learning

To operate a ChatGPT-utilizing class for Scratch learning, we investigated the applicability of ChatGPT across different Scratch learning domains. This helped us determine which specific areas of Scratch learning would benefit from the use of ChatGPT. Notably, this study employed ChatGPT-3.5, which is freely accessible to learners, for the experiments.

Table 1. Utilization of ChatGPT in Scratch learning areas

Learning Area	Utilization of ChatGPT
Block Usage	• Explanation of each Scratch block's function • Explanation of assembling, disassembling, deleting, and copying blocks • Providing helpful tips for block assembly
Variable Usage	• Explanation of how to create and delete basic and structured variables • Explanation and usage of global and local variables • Providing programming examples using variables
Conditions	• Explanation of how to write relational and logical expressions • Generating conditions necessary for given problems • Providing programming examples utilizing conditions
Control Statements	• Explanation of selection/iteration control blocks • Providing programming examples utilizing selection/iteration control blocks
Procedure Usage	• Explanation of the concept of procedures • Explanation of how to create procedure blocks • Providing programming examples utilizing procedures
Sensing/Events	• Explanation of sensing/event blocks • Providing programming examples utilizing sensing/event blocks • Providing programming examples utilizing event broadcasting functionality
Cloning Functions	• Explanation of the concept of cloning and cloning blocks • Providing programming examples utilizing cloning blocks • Providing helpful tips for using cloning (such as preventing clone collisions and ensuring independent behavior)
Algorithm Understanding and Implementation	• Explanation of a specific algorithm (operation method, time complexity) • Providing a scenario demonstrating the operation of a specific algorithm • Providing programming examples of algorithm implementation

Table 1 summarizes how ChatGPT can be utilized in various Scratch learning areas based on the learning content provided by the instructor: In terms of block usage, ChatGPT not only explains how to assemble blocks but also provides useful tips, such as how to copy scripts between sprites, which are the objects in Scratch that are the targets of programming. For variable usage, ChatGPT explains how to create basic and structured variables, describes local and global variables and how to create them, and provides various examples of using variables. Regarding conditions, ChatGPT explains how to create conditions using relational and logical expressions, automatically generates conditions needed for problem-solving when given a problem, and provides programming examples. For control statements, ChatGPT explains the control blocks in Scratch and provides programming examples using these blocks. In terms of procedure usage, ChatGPT explains the concept of procedures, how to create procedure blocks, and offers programming examples that use procedures. For sensing/events functionality, ChatGPT explains how to use sensing/event blocks and provides programming examples, including complex examples for event broadcast functions to aid learner comprehension. When it comes to the advanced topic of cloning functions, ChatGPT explains the concept of cloning, how to use cloning-related blocks, and provides tips and programming examples to prevent clone collisions and ensure independent behavior of each clone. Finally, for understanding and implementing algorithms, ChatGPT explains the workings of specific algorithms (e.g., greedy algorithm, binary search algorithm, Euclidean algorithm for finding the greatest common divisor), details the time complexity needed for understanding algorithm performance, and provides specific algorithm scenarios to aid learners understanding. ChatGPT also offers Scratch programming examples that implement these algorithms.

Based on this analysis, this study concluded that learner-centered, self-directed learning utilizing ChatGPT is feasible across all Scratch learning areas outlined in Table 1.

Ⅲ. Experimental Method and Preparations

3-1 Experimental Method by Group

The study used a foundational software education course at a provincial private university, which is designated as a software-centric institution, to explore how ChatGPT impacts programming learning among non-SW majors. The chosen course emphasizes developing computational thinking and improving essential programming skills required for non-SW majors using Scratch, an educational programming language. It covers various learning areas, including designing conditions using relational and logical expressions for problem-solving, procedural design based on hierarchical decomposition and procedural thinking, and understanding and applying algorithms through selection and iteration control.

The research divided the software education course into two groups for experimentation. Group A received instructions during lab sessions solely on program outlines and expected outcomes, relying on ChatGPT thereafter to independently find solutions and complete the programs. Group B followed a traditional approach where the instructor led practical sessions, demonstrating programming processes and aiding student understanding through explanations. Both groups engaged in question-and-answer sessions with the instructor following the practical exercises. Table 2 summarizes the approaches to conducting practical exercise and the roles of the instructors between the two groups.

Table 2. Practice performance method of group A and group B

Group	Practical Exercise Performance Method	Role of Instructor
Group A	• (Professor → Student) Explain program practice content and results. • (Student) Self-directed practice using ChatGPT. • (Professor ↔ Student) Q&A between professor and students.	• Provide program practice • Manage and supervise practice • Answer student queries
Group B	• (Professor → Student) Explain program practice content and results • (Professor → Student) Instructor-led program creation and explanation • (Professor ↔ Student) Q&A between professor and students	• Provide program practice • Lead practice process • Answer student queries

3-2 Preparation for Experiment

This study employed different teaching methods across groups, and prior to the classes, an orientation session was conducted to inform and obtain consent regarding the practical exercise performance approach. To assess the programming experience of students in both groups, a survey was conducted, yielding results as shown in Table 3. In Group A, out of a total of 34 students, 2 had prior programming education, while in Group B, out of 31 students, 1 had received programming education before the course. Students with previous programming education had no experience with programming languages other than Scratch.

This research aimed to experimentally assess the effectiveness of using ChatGPT in introductory programming education. To achieve this, three students who had received prior Scratch training (2 from Group A and 1 from Group B) were excluded from the experimental group. The programming competencies between the two groups were compared using the remaining students. During the orientation session, the students were educated on Scratch's programing environment, basic block usage, the mathematical meaning of relational and logical expressions, and concepts of selection/repetition control in statements. A pre-assessment was conducted, consisting of 12 questions that assessed understanding of Scratch's runtime environment, block usage skills, comprehension of logical expressions, and abilities in selective/repetitive statement control.

Table 4 summarizes the results of the t-test based

Table 3. Preliminary survey on programming experience

Category	Group A (34 Students)	Group B (31 Students)
Scratch Programming Experience	• Yes: 2 students • No: 32 students	• Yes: 1 student • No: 30 students
Programming Experience Outside of Scratch	• Yes: None • No: 34 students	• Yes: None • No: 31 students

Table 4. Programming competencies t-test analysis

Group	Students #	Average	Variance	t-Statistic	p-Value
Group A	32	9.19	1.05	-1.65	0.1035
Group B	30	9.56	0.42		

on the pre-evaluation outcomes, indicating that there was no statistically significant difference in programming competence between the two groups (significance level α = 0.05, p-value = 0.1035).

Ⅳ. Experimental Results and Analysis

4-1 Evaluation Method

The assessments for Group A and Group B commonly consisted of a midterm exam, a final exam, and programming assignments. The midterm exam included questions that evaluated understanding of Scratch syntax and the ability to create basic programs, while the final exam comprised questions that assessed the utilization of advanced Scratch features and comprehension of algorithms. Both the midterm and final exams had the same questions for both groups. Since the programming assignments were performed outside of class, they were not considered objective evaluation data for this study. Therefore, this study compared the learning effects based solely on the midterm and final exam scores of the two groups.

Table 5 summarizes the types of questions used in the midterm and final exams, the content of each type, and their point allocations.

Table 5. Composition of midterm and final exam questions

Problem Types		Problem Description	Points
Midterm Exam	Scratch Grammar Understanding	• Variable creation function • Arithmetic / relational / logical operator functions • Conditional statement syntax • Loop statement syntax	30
	Basic Code Writing Ability	• Writing conditions for relational and logical expressions • Using lists and procedures • Writing conditional statements • Writing loop statements	70
Final Exam	Advanced Scratch Feature Usage	• Using sensing features • Using event features • Using pen and instrument play features • Using cloning features	50
	Algorithm Understanding Ability	• Finding Euclidean greatest common divisor • Giving minimum coin change • Binary search algorithm • Procedure recursion	50

4-2 Assessment Results and Analysis

Table 6 summarizes the average scores of Group A and Group B by question type for the midterm and final exams. In the question type assessing understanding of Scratch syntax and basic code-writing ability, there was no significant difference in average scores between the two groups. This is attributed to the subject being relatively easy, allowing students to effectively practice through lectures and textbooks alone. However, for the question type assessing the use of advanced Scratch features, Group A's average score was 4.36 points higher than Group B's.

Table 6. Average scores of each group by problem type

Problem Types	Points	Group A	Group B
Scratch Grammar Understanding	30	26.72	26.67
Basic Code Writing Ability	70	56.41	56.83
Advanced Scratch Feature Usage	50	37.19	32.83
Algorithm Understanding Ability	50	34.38	27.67

This suggests that as the difficulty of practical tasks increased, Group A developed a better understanding of advanced Scratch features through self-directed problem-solving with ChatGPT. For the question type assessing algorithm comprehension, Group A scored 6.71 points higher than Group B. Upon analyzing this significant score difference, it was found that Group A, with the support of ChatGPT, engaged in learning not only Scratch coding methods but also understanding algorithms themselves. As a result of this learning, which enhanced their understanding of algorithms, Group A achieved higher scores in this type of question compared to Group B.

Table 7 presents the t-test results of the performance of Group A and Group B by question type (significance level α = 0.05).

The results showed no statistically significant difference between the two groups in the problem types of understanding Scratch syntax and basic code-writing ability (Scratch syntax understanding: p-value = 0.946116, basic code-writing ability: p-value = 0.802534). However, Group A scored significantly higher than Group B in the use of advanced Scratch features and algorithm comprehension (use of advanced Scratch features:

Table 7. t-test analysis on scores by problem type

Problem Types	Group	Average	Variance	t-Statistic	p-Value
Scratch Grammar Understanding	Group A	26.72	9.05	0.068	0.946116
	Group B	26.67	9.20		
Basic Code Writing Ability	Group A	56.41	52.00	−0.251	0.802534
	Group B	56.83	37.04		
Advanced Scratch Feature Usage	Group A	37.19	30.54	3.020	0.003708
	Group B	32.83	33.94		
Algorithm Understanding Ability	Group A	34.38	54.44	3.688	0.000489
	Group B	27.67	47.82		

p-value = 0.003708, algorithm comprehension: p-value = 0.000489).

4-3 Survey on the Use of ChatGPT

A survey on the use of ChatGPT was conducted with a total of 34 students, including 32 students from Group A and the 2 students who were excluded from Group A due to their prior Scratch learning experience. The survey aimed to determine the extent to which ChatGPT helped in the Scratch course and its usefulness in various areas of Scratch learning, as outlined in Table 8.

Fig. 1 represents the survey results of the respondents in graphical form. Among the respondents, 30 students (88.2%) answered "Yes" or "Very Much" when asked if the use of ChatGPT improved their understanding of Scratch course content, indicating a positive perception. Additionally, 28 students (82.4%) responded "Yes" or "Very Much" when asked if ChatGPT increased their interest, again reflecting a positive perception. The remaining students answered "Neutral," and no students responded negatively.

The survey results on how helpful ChatGPT was in various learning areas of Scratch, listed in descending order of scores, are as follows: algorithm understanding/ implementation (average score 4.79), condition creation (average score 4.62), selection/repetition statements control (average score 4.47), cloning features usage (average score 4.38), event features usage (average score 4.15), sensing

Table 8. Survey questions on ChatGPT usage

Survey Questions		Remarks
Did ChatGPT help with learning?	Enhancement of Understanding of Learning Content	① Not at all ② No ③ Neutral ④ Yes ⑤ Very much
	Enhancement of Interest in Learning	
How much did ChatGPT help in learning areas?	Block usage	
	Condition creation	
	Selection/repetition statement control	
	Sensing feature usage	
	Event feature usage	
	Cloning feature usage	
	Algorithm understanding/implementation	

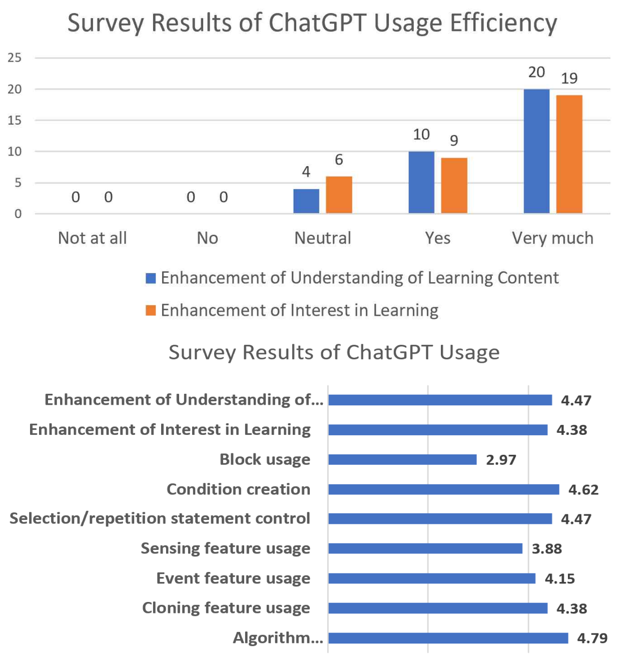

Fig. 1. Survey results on the use of ChatGPT

features usage (average score 3.88), and block usage (average score 2.97). This indicates, in conclusion, that students found significant help from ChatGPT in understanding the conditions and algorithms required for their exercises and implementing algorithms through control of selection/repetition statements. When observing the activities of students during the class time, it was noted that they used ChatGPT to learn how to implement the conditions and algorithms necessary to solve exercises. Additionally, students utilized ChatGPT to explore various examples beyond the provided exercises, especially when learning challenging topics such as cloning and event broadcasting features. Lastly, students evaluated that ChatGPT did not provide significant assistance with block assembly methods.

Ⅴ. Conclusion

As generative AI technology advances, research on its application in programming education is actively ongoing. This study specifically analyzed the effectiveness of integrating ChatGPT into Scratch programming education for non-SW major students. The Scratch classes for non-SW major students were divided into two groups: one utilized ChatGPT and the other did not. The study statistically analyzed the difference in learning outcomes based on identical assessment results. Additionally, a survey was conducted to evaluate the usefulness of ChatGPT.

The study results showed that the use of ChatGPT had a statistically significant impact on improving the learning effectiveness of Scratch classes. Specifically, while there was no statistically significant difference between the group that used ChatGPT and the group that did not in terms of understanding Scratch syntax and basic code writing, the group that used ChatGPT scored significantly higher on tests involving the use of advanced Scratch features and algorithm comprehension. Additionally, in a survey conducted with students from the ChatGPT-using group regarding the educational usefulness of ChatGPT, over 80% responded positively, indicating that ChatGPT helped their learning, with many stating that it was particularly helpful for condition generation and algorithm understanding. These results suggest that the more challenging areas of Scratch learning, in particular, benefited from self-directed learning with the assistance of ChatGPT. This implies that ChatGPT was especially helpful for students when tackling more difficult concepts. This study is significant in empirically exploring the potential of ChatGPT in programming education through quantitative analysis of programming learning effects.

This study explored the learning effectiveness of using ChatGPT in Scratch education for non-SW major students. However, considering ChatGPT's capability to support learning across various advanced programming languages, there is a need to expand this research to target advanced programming language courses for software major students.

Acknowledgment

This research is based on the support of 『2024 Woosong University Academic Research Funding』

References

[1] H. Jin, B. Yoon, and S. Shin, A Study on Policy Directions for Fostering Software Talent in Response to Generative AI, Software Policy & Research Institute, Seongnam, Reserach Report RE-158, December 2023.

[2] OpenAI. Official Website [Internet]. Available: https://openai.com/.

[3] Massachusetts Institute of Technology. Scratch Homepage, [Internet]. Available: https://scratch.mit.edu/.

[4] Brightchamps. Scratch Programming User and Growth Statistics 2022 [Internet]. Available: https://brightchamps.com/blog/scratch-stats/.

[5] S. Lee and K. Song, “Exploration of Domestic Research Trends on Educational Utilization of Generative Artificial Intelligence,” *The Journal of Korean Association of Computer Education*, Vol. 26, No. 6, pp. 15-27, November 2023. https://doi.org/10.32431/kace.2023.26.6.002

[6] S. Kim, “Exploring the Possibility of Using Generative Artificial Intelligence for Programming Education: Focusing on ChatGPT,” in *Proceedings of 2023 Summer Academic Conference of the Korean Association of Computer Education (KCEC 2023)*, Seoul, pp. 151-154, August 2023.

[7] S. Kim, “Developing Code Generation Prompts for Programming Education with Generative AI,” *The Journal of Korean Association of Computer Education*, Vol. 26, No. 5, pp. 107-117, September 2023. https://doi.org/10.32431/kace.2023.26.5.009

[8] J.-O. Park, “A Study on the Experience and Utilization of Generative AI-Based Classes - Focusing on Programming Classes,” *Journal of Practical Engineering Education*, Vol. 16, No. 1, pp. 33-39, February 2024. https://doi.org/10.14702/JPEE.2024.033

[9] B. Gong, S. Jo, J. Chang, S. Park, and G. Kwon, “Comparative Analysis of Code Generated by ChatGPT and Human Programmers,” in *Proceedings of the 2023 KIIT Summer Conference*, Jeju, pp. 789-792, June 2023.

[10] S. Choi and H.-W. Byun, “Human Programmers versus ChatGPT 3.5 & 4.0: A Comparison of Coding in Korean,” *Journal of Digital Contents Society*, Vol. 24, No. 12, pp. 3167-3178, December 2023. https://doi.org/10.9728/dcs.2023.24.12.3167

고광일(Kwangil Ko)

1995년 : 포항공과대학교 전자계산학과 (학사, 석사)
1999년 : 포항공과대학교 컴퓨터공학과 (공학박사)

1999년~2010년 8월: (주)알티캐스트 사업품질관리본부 본부장 및 서비스개발사업팀 팀장
2010년 9월~현 재: 우송대학교 테크노미디어융합학부 미디어디자인·영상전공 교수
※관심분야 : 디지털방송 소프트웨어, 스마트TV방송UI/UX, 소프트웨어공학, 요구분석공학, N-스크린 서비스, 소프트웨어 교육