Available Online: 3 September 2025 DOI: 10.54254/2977-3903/2025.26530

Exploring the application of AIGC in high school information technology programming instruction—A case study of "Python Programming: Conditional Structures"

Biling Hu[1*], Qingyan Luo[2], Qiongqiong Wang[3]

[1]Mingli High School, Shenzhen Experimental School, Shenzhen, China

[2]Chongwen High School, Shenzhen Experimental School, Shenzhen, China

[3]School Division, The Affiliated High School of Renmin University of China (RDFZ) Shenzhen School, Shenzhen, China

*Corresponding Author. Email: 1963643992@qq.com

Abstract. With the rapid development of artificial intelligence, the application of Generative AI (AIGC) in education has gradually become a research focus. This study explores the potential of AIGC in programming instruction within high school information technology classes. Based on an in-depth analysis of AIGC's advantages and potential limitations in education, we propose an innovative programming-teaching approach that integrates AIGC and pilot its use in real classroom practice. Data analysis indicates that the AIGC-integrated approach can, to some extent, enhance students' classroom engagement. The findings offer new ideas and methods for high school information technology teaching and provide a reference for future innovation and development of programming-instruction models.

Keywords: generative artificial intelligence, high school information technology, programming instruction, teaching model

1. Introduction

Against the backdrop of the digital era, the rapid development of information technology has not only reshaped the social and economic landscape but also profoundly influenced transformations in education. High school information technology, as a key foundation for cultivating students' information literacy and innovation capability, is becoming increasingly important. However, traditional programming instruction often places emphasis on knowledge transmission and skills training while neglecting students' agency and individualized needs during learning, making it difficult to fully stimulate their interest and potential. As a significant branch of artificial intelligence, Generative AI (AIGC) possesses strong capabilities for content creation and innovation, offering boundless possibilities for educational innovation. Introducing AIGC into programming instruction can not only increase the interest and interactivity of teaching content, but also provide intelligent feedback tailored to students' individual needs—benefiting their in-class programming learning. Therefore, this study aims to explore the application potential of AIGC in programming instruction in high school information technology classrooms. Through an in-depth analysis of AIGC's advantages and potential limitations in education, we design an innovative AIGC-integrated teaching approach and attempt to apply it in actual programming-teaching practice.

2. Research background

2.1. Current situation of programming instruction

With the rapid development of information technology, programming instruction has become an essential component of the information technology curriculum. The goal of programming instruction is not only to cultivate students' computational thinking, logical reasoning, and problem-solving skills, but also to establish the core competencies required in the information age. While current programming instruction can efficiently cover a large number of students and deliver foundational knowledge, its inherent characteristics of standardization and uniformity limit the attention given to individual differences among learners.

Programming requires students to possess certain logical thinking and creative abilities, and the cultivation of these abilities often calls for differentiated instruction tailored to students' interests, abilities, and learning pace. However, in today's high school information technology classrooms, teachers typically have to teach to the average level of the class, making it difficult to meet each student's individual learning needs. As a result, teaching effectiveness and students' potential development are often constrained. Meanwhile, the highly practical nature of programming, combined with the rapid pace of knowledge updates, has placed higher demands on programming-instruction models. Although many educators have attempted to introduce advanced instructional approaches—such as project-based learning, problem-based learning, and flipped classrooms—into programming education to create more practice opportunities and improve learning outcomes, classroom-based collective instruction still leaves teachers stretched too thin to provide in-depth, individualized guidance. Consequently, innovating teaching methods and approaches to meet the practical needs of programming instruction in information technology classrooms has become an urgent challenge.

2.2. Application of AIGC in programming instruction

AIGC (Artificial Intelligence Generated Content) refers to generative artificial intelligence, a technology that automatically produces content such as text, images, audio, and video by learning from large-scale datasets [1]. Its powerful capability for generating personalized content offers new possibilities for programming instruction in information technology classrooms. Current research on AIGC applications in education primarily focuses on teaching, learning, assessment, and tutoring [2]. Accordingly, the integration of AIGC into programming instruction in IT classrooms can also be explored and practiced along these four dimensions. In the teaching dimension, language models trained on large-scale data enable AIGC to generate diverse instructional designs and teaching resources tailored to specific teacher requirements. For instance, in programming instruction, AIGC can provide exercises and examples of varying difficulty levels and formats according to students' learning progress and comprehension [3]. In the learning dimension, AIGC acts as an intelligent learning companion, constructing problem scenarios and offering personalized guidance and support. Students can interact with AIGC to quickly access learning content and resolve doubts. For example, in programming learning, students may consult AIGC about coding challenges, syntax rules, or algorithmic strategies, and AIGC can provide immediate feedback and detailed explanations. This instant feedback mechanism helps students correct mistakes promptly and enhances learning efficiency [4]. In the assessment dimension, AIGC can compile and execute student-submitted code, detect syntax and logical errors, and provide specific improvement suggestions. Such automated assessment not only reduces teachers' grading workload but also enhances the objectivity and fairness of evaluation [5]. Finally, in the tutoring dimension, AIGC leverages its extensive knowledge base and reasoning capabilities to offer comprehensive programming support. Both during and outside of class, students can interact with AIGC to receive timely assistance and guidance in their learning process [6].

3. Research metho

3.1. Problem-Based Learning (PBL)

Problem-Based Learning (PBL) is a widely used student-centered instructional model in programming education. Its core principle is to place learners in complex and meaningful problem scenarios, where they engage in autonomous inquiry and collaboration to solve problems. Through this process, learners acquire the scientific knowledge embedded within the problem, while simultaneously developing problem-solving skills and the ability for self-directed learning [7]. In programming education, the PBL approach typically consists of four stages: problem presentation and comprehension — group discussion and inquiry — coding and problem-solving — project presentation and evaluation. Teachers are responsible for designing practical programming problems that are closely related to students' interests and disciplinary backgrounds. Students, through teamwork, analyze the problem, design solutions, write code, test, and debug programs, and ultimately complete the project. Teachers act as facilitators and guides, offering necessary support while encouraging students to explore, experiment, and learn actively. However, in today's context of information explosion and rapidly evolving knowledge, the traditional PBL model faces challenges such as outdated content and insufficient personalized guidance.

Artificial Intelligence Generated Content (AIGC), particularly generative AI models such as ChatGPT, offers unprecedented support for PBL with its powerful natural language processing and knowledge generation capabilities. On the one hand, AIGC can enrich PBL with more diverse, authentic, and contextually relevant problem scenarios and resources, making the learning process more engaging and efficient. On the other hand, it can provide real-time feedback and supplementary resources during problem-solving, thereby deepening students' understanding and broadening their perspectives. Based on these affordances, this study draws on the PBL teaching method and innovatively integrates AIGC technology to propose a new AIGC-integrated classroom programming teaching approach.

3.2. Aigc-integrated classroom programming teaching approach

Building upon the strengths of AIGC and the core principles of PBL, this study designs an AIGC-integrated classroom programming teaching method that divides the instructional process into four closely interconnected phases: Perception and Engagement — Problem and Exploration — Sharing and Collaboration — Reflection and Evaluation (see Figure 1).

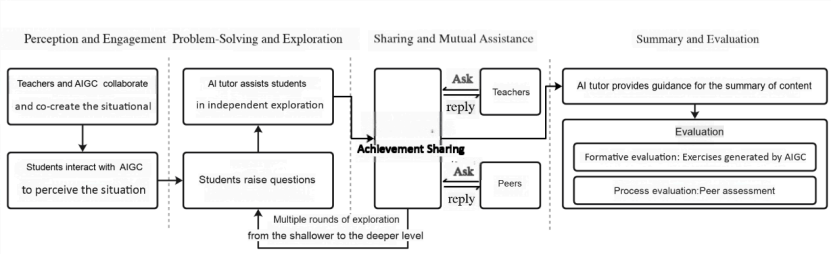

Figure 1. AIGC-integrated classroom programming teaching approach

3.2.1. Perception and engagement: stimulating interest and curiosity

The purpose of this stage is to foster emotional connection between students and the learning content, inspiring intrinsic motivation to solve programming problems. Teachers and AIGC collaborate to co-create authentic learning contexts. Leveraging AIGC's content generation capacity, teachers can access rich and inspiring references for constructing realistic scenarios. Meanwhile, students engage in multi-turn interactive dialogues with AIGC to deepen their empathetic understanding of the scenario. This interaction not only enhances students' clarity regarding the learning objectives but also lays a solid foundation for subsequent inquiry and deeper learning.

3.2.2. Problem and exploration stage: emphasizing the promotion of deep learning

The problem and exploration stage is a process in which students, after understanding the contextual scenario, use prompts to engage in autonomous inquiry with the help of AIGC. Prompts serve as the medium of interaction between users and AI models; they can be text inputs that guide the AI to generate expected outputs. At this stage, teachers need to carefully design a series of layered and progressively deepening question chains, closely aligned with the contextual problem and instructional objectives. These question chains not only provide students with clear directions for inquiry but also serve as examples of prompts they can use when seeking assistance from AIGC. During exploration, AIGC functions as an intelligent tutor that guides students toward deep learning. It offers targeted inquiry support, introduces new knowledge, and addresses students' questions encountered along the way, all tailored to individual learning needs. For instance, for the contextual problem of "How to write a Python program to calculate BMI," which involves concepts such as input commands, data types, and output commands, teachers can break down the knowledge points into a question chain, as shown in Table 1. Students can then use AIGC to explore these questions, deepen their understanding of each concept, and engage in programming practice. If students encounter compilation or runtime errors during practice, they can also seek help from the intelligent tutor. AIGC provides timely feedback to correct mistakes, enabling students to grasp knowledge more efficiently and strengthen their problem-solving skills.

Table 1. Prompt examples for "how to write a python program to calculate BMI"

Programming Knowledge Point	Prompt Example
Output command	Q1: How can output be implemented in Python?
Input command	Q2: What is the input command in Python? Q3: How can this command be used to input height and weight?
Data types	Q4: Can the value obtained by input() be directly used for numerical operations? Q5: If not, how should the code be modified?
Special operators	Q6: What is the exponentiation operator in Python?
Comprehensive application	Q7: Can you draw a flowchart for the BMI calculation program? Q8: Please write a Python program to calculate BMI based on the flowchart.

3.2.3. Sharing and collaboration stage: fostering critical thinking

In the sharing and collaboration stage, students present and share the knowledge, experiences, and programming outcomes they have accumulated during the problem and exploration stage. As presenters, students internalize their learning and practice, clearly articulating their programming ideas, problem-solving strategies, and the methods used to overcome challenges. Based on students' presentations, teachers can guide peers to ask constructive questions or provide useful feedback, thereby fostering a more active and in-depth classroom discussion. Importantly, the problem-and-exploration stage and the sharing-and-collaboration stage can alternate. Each round of sharing and feedback further deepens students' understanding of the learning content, while also stimulating them to consider higher-order questions, which in turn leads to more advanced inquiry. Through this process, students not only externalize and showcase their learning outcomes in a timely manner but also strengthen their understanding of knowledge through peer and teacher interaction. Moreover, this stage contributes to the cultivation of students' critical thinking skills.

3.2.4. Summarization and evaluation stage: enhancing learning outcomes and self-awareness

In the summarization and evaluation stage, students and teachers jointly review and reflect on the learning process to comprehensively assess learning effectiveness and provide targeted guidance for future learning. Here, AIGC continues to play the role of an intelligent tutor, assisting students in summarizing their learning while also generating evaluation content and offering evaluation support. For assessment, AIGC generates personalized classroom exercises based on students' learning performance. These exercises serve to test students' understanding and application of key knowledge points and act as important indicators of formative assessment. After students complete the exercises, AIGC can instantly provide feedback, pointing out mistakes and blind spots in their problem-solving process, along with corresponding correction suggestions and recommended learning resources. This enables students to continuously adjust and optimize their learning process, thereby improving effectiveness. Meanwhile, teachers can organize peer evaluation activities. Students, following predetermined evaluation criteria, assess their peers' programming projects and learning performance through performance-based evaluation. This process encourages reflective learning and mutual growth.

3.3. Case study of a classroom programming teaching design integrating AIGC

Taking the high school information technology lesson "Python Programming: Conditional Structures" as an example, this study designs a teaching case based on the AIGC-integrated instructional approach. The selected teaching tool is "Code Raccoon", an AIGC-based programming assistant developed by SenseTime. This tool supports continuous dialogue, provides programming knowledge and technical guidance, and offers functions such as program translation, code generation, code refactoring, and debugging. These features are sufficient to meet students' classroom programming needs. The target learners are Grade 10 students, who have already studied the basics of Python programming, including input, output, data types, and variables. The main instructional objective is to enable students, with AIGC assistance, to independently explore the syntax rules and usage of conditional structures, and to apply them to solve real-world problems by writing simple programs. Through this process, students develop logical thinking and problem-solving skills. The specific instructional process is described below.

3.3.1. Perception and engagement

In the perception and engagement stage, teachers distribute a learning guide sheet (see Table 2) and present the learning context through multimedia resources such as videos and images. To introduce the concept of conditional structures, and in alignment with the instructional objectives, teachers—assisted by AIGC—design a contextual scenario called "Space Adventure" and pose a guiding problem that runs throughout the lesson: "How can we write a program to help the spaceship achieve basic automatic obstacle avoidance?" With this guiding problem in mind, students consult the AIGC intelligent tutor using the example prompts provided in the guide sheet. The first two questions in the guide sheet are designed to help students understand the context and stimulate intrinsic motivation to solve the problem.

Table 2. Learning guide sheet for space adventure

Learning Guide Sheet for Space Adventure
Scenario: After waking up, the astronaut finds themselves on a spaceship drifting aimlessly in space. The control lever is broken, and the only way to steer the spaceship is through programming commands. As the spaceship may encounter obstacles at any time, how can we write a program to help it achieve basic automatic obstacle avoidance?
Guiding Question: How can we write a program to help the spaceship achieve basic automatic obstacle avoidance?
Consult Code Raccoon using the following prompts to address the contextual problem.

		Prompt Examples
Understanding the context (emotional perception)		Q1: Why is an automatic obstacle detection system critical to the spaceship's safety? Q2: What types of obstacles might the spaceship encounter while traveling through space?
Clarifying objectives (problem formulation)		Q3: When the spaceship identifies different obstacles, how should the program respond differently (e.g., slowing down vs. changing direction)?
Independent inquiry	Basic concept of conditional structures	Q4: In Python, how can we execute different code blocks based on different conditions?
	Single-branch conditional structure	Q5: What is a single-branch conditional structure? Q6: If we only need to determine whether there is an obstacle, how can we write the program?
	Double-branch conditional structure	Q7: What is a double-branch conditional structure? Q8: Suppose the spaceship's object-recognition sensor can identify two objects—asteroids and space debris—and stores the recognition result in the variable obstacle (obstacle=1 for asteroid, obstacle=2 for space debris). When encountering an asteroid, the spaceship must change direction; when encountering space debris, it must slow down. How can we write such a program?
	Multi-branch conditional structure	Q9: What if there are more than two obstacles? For example, in addition to asteroids and space debris, suppose the spaceship may also encounter cosmic rays (obstacle=3). In this case, the spaceship must both avoid the area and activate its protective shield. How can we write such a program?
Learning summary		Q10: What new programming knowledge have you learned in this lesson? Can you summarize it?

3.3.2. Problem exploration

In the problem exploration stage, students are required to conduct in-depth analysis and deconstruction of the complex contextual problem, abstracting the real-world situation into clear programming requirements. Guided by the example prompts (Questions 4–9 in Table 2), students independently explore using the AIGC tool, step by step acquiring knowledge and practice related to conditional structures. Throughout this process, AIGC acts as an intelligent tutor, providing support for learning programming concepts, guiding coding practice, and assisting with debugging.

3.3.3. Sharing and peer collaboration

During the exploration process, the teacher closely monitors students' progress and strategically introduces opportunities for presentation and sharing at appropriate points within the problem chain. For example, once most students have completed Questions 4–6, the teacher may invite them to share what new knowledge they have acquired, what challenges they encountered, how they addressed these challenges, and the interim programming outcomes they produced. Based on these presentations, the teacher and peers engage in discussion and exchange. During this process, the teacher encourages students to reflect more deeply by posing questions such as: "If we not only need to determine whether there is an obstacle, but also what type of obstacle it is, can a single-branch structure still meet the requirement?" This naturally leads students into the next phase of inquiry on double-branch conditional structures. Similarly, after completing Questions 7–8 and Question 9, further sharing sessions can be introduced. Such externalized demonstrations help students deepen their understanding of the content through reflection and collaborative dialogue.

3.3.4. Summary and evaluation

At the conclusion of independent inquiry, the teacher guides students in reviewing and reflecting on the learning process. With the assistance of AIGC, students complete summaries of the programming knowledge they have learned. Following this, the teacher organizes a peer evaluation activity (see Table 3), encouraging students to appreciate and learn from each other. Through evaluating peers' work and performance, students not only enhance their evaluative abilities but also cultivate critical thinking.

Table 3. Peer evaluation criteria

No.	Evaluation Standard	1 2 3 4 5
1	Actively interacts with the intelligent tutor to obtain useful information and guidance	
2	Thinks critically about the feedback provided by the intelligent tutor rather than copying it directly	
3	Modifies the program based on the intelligent tutor's feedback	
4	Proactively shares learning insights and interim outcomes	
5	Listens attentively to peers' presentations and engages in active discussion	
6	Completes a thoughtful summary of the learning process	

4. Teaching implementation and data analysis

The improvement and refinement of the AIGC-integrated classroom programming teaching method is grounded in actual teaching practice, and its effectiveness must be demonstrated through empirical data. Taking the senior high school information technology course Python Programming: Conditional Structures as an example, this study conducted an experimental teaching intervention in a Grade 10 cohort at a high school in Shenzhen, Guangdong Province. Participants were drawn from two Grade 10 classes, which were designated as the experimental group and the control group. The experimental group adopted the AIGC-integrated programming teaching method, where students utilized AIGC as an auxiliary tool for in-class learning. The control group followed the conventional information technology teaching approach without using any AIGC-related tools. Both groups were taught by the same experienced programming teacher, ensuring consistency in the teaching context, scenario design, and knowledge points covered.

4.1. Experimental measurement instrument

This study employed the Math and Science Engagement Scale (MSES) as the instrument to assess students' classroom engagement [8]. The MSES is designed to evaluate students' engagement in mathematics and science courses across three dimensions: behavioral engagement, emotional engagement, and cognitive engagement. Behavioral engagement involves indicators such as participation, effort, and attention. Emotional engagement reflects students' positive or negative responses toward teachers, peers, academic tasks, or school in general. Cognitive engagement concerns students' use of learning strategies, deep thinking, and psychological investment during the learning process.

4.2. Data analysis

The engagement scale consisted of 14 items distributed across the three dimensions. Collected data were analyzed using SPSS, and independent samples t-tests were conducted to determine whether the AIGC-integrated programming teaching method significantly influenced students' classroom engagement. The results are shown in Table 4. At the 0.05 significance level, there were statistically significant differences between the experimental and control groups in terms of classroom engagement. As illustrated in Figure 2, the confidence intervals of the total engagement scores for the two groups did not overlap, suggesting substantial differences.

Table 4. Independent samples t-test results

Dimension	Independent Samples t-test Results		
	p-value	t-value	Sig. (two-tailed)
Behavioral Engagement	0.013	8.781	0.000
Emotional Engagement	0.000	7.676	0.000
Cognitive Engagement	0.057	8.620	0.000
Total Score	0.101	11.13	0.000

Figures 3–5 further illustrate the score distributions for each engagement dimension. The results clearly indicate that, compared to the control group, the experimental group achieved higher levels of behavioral, emotional, and cognitive engagement.

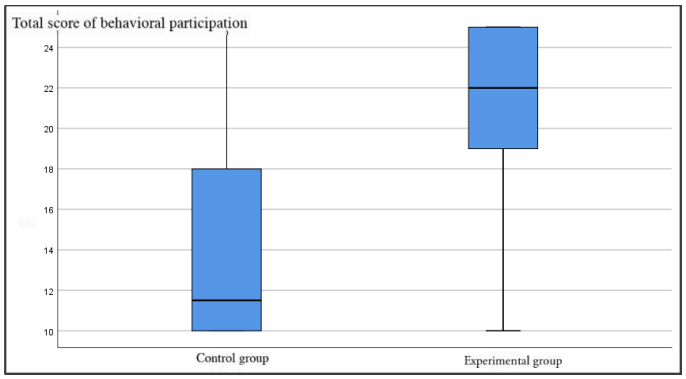

Figure 2. Distribution of total classroom engagement scores

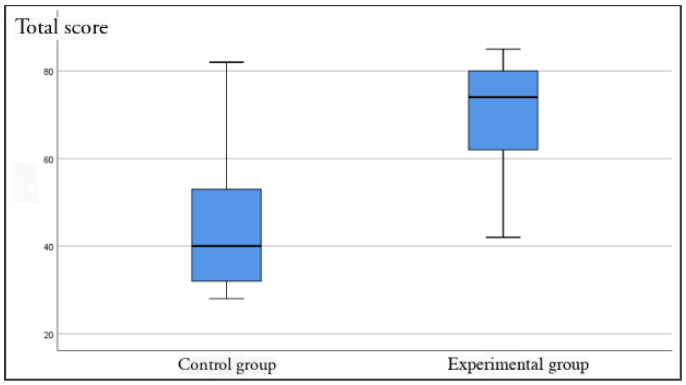

Figure 3. Distribution of behavioral engagement scores

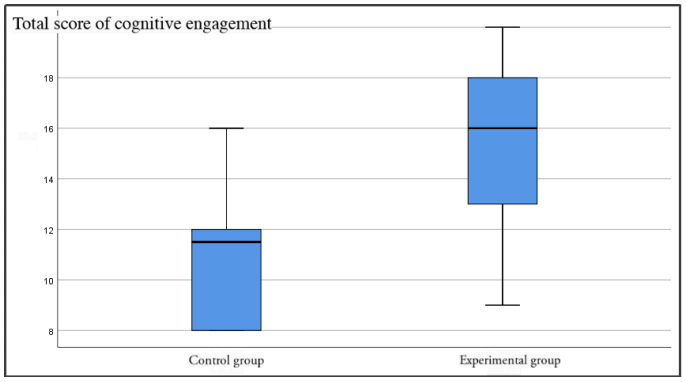

Figure 4. Distribution of emotional engagement scores

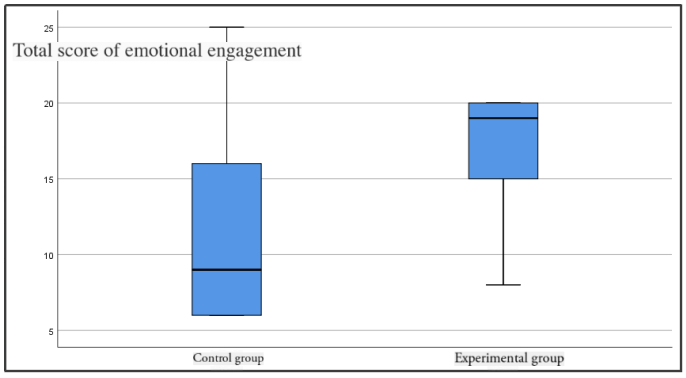

Figure 5. Distribution of cognitive engagement scores

5. Results and discussion

By comparing the classroom engagement of the experimental group (which adopted the AIGC-integrated programming teaching method) with that of the control group (which followed the conventional information technology teaching method), this study found that the AIGC-integrated approach significantly enhanced students' engagement in programming classes. Students in the experimental group scored noticeably higher in behavioral engagement, emotional engagement, and cognitive engagement compared with their peers in the control group. These results may be attributed to several factors. First, AIGC provides instant feedback, personalized guidance, and interactive learning experiences, which dynamically adjust instructional content according to students' progress and comprehension levels. This adaptability enhances students' interest and motivation in learning programming. Second, AIGC as an auxiliary tool helps students quickly immerse themselves in the designed learning scenarios, facilitating comprehension of contextualized problems. Furthermore, the sharing and peer-collaboration activities promoted meaningful interactions among students and between students and teachers, which may explain the significant improvement in emotional engagement. Finally, in terms of cognitive engagement, AIGC offered students diverse learning resources and intellectual challenges, stimulating curiosity and inquiry. This encouraged deeper reflection on programming problems and improved their cognitive skills and problem-solving abilities.

Therefore, the AIGC-integrated classroom programming method developed in this study demonstrates remarkable advantages in improving student engagement, providing new perspectives and methods for high school information technology instruction. Nevertheless, some limitations should be noted. The sample size was relatively small, which may affect the generalizability of the findings. Additionally, the study focused only on classroom engagement as the evaluation indicator, without incorporating other important outcomes such as learning achievement or programming competence. Future research could address these limitations by expanding the sample size, extending the experimental duration, and adopting multiple evaluation metrics for a more comprehensive assessment of the effectiveness of AIGC-integrated teaching. Moreover, further exploration across different age groups and subject domains would provide valuable insights into the broader applicability of AIGC-supported instructional methods in educational practice.

Funding project

This research was supported by the Guangdong Province Primary and Secondary School Teachers' Information Technology Application Ability Enhancement Project 2.0 (Project Approval No. TSGCKT2023072).

References

[1] Wu, D., Li, H., & Chen, X. (2023). An analysis of the educational application impacts of general-purpose AI large models. *Open Education Research*, 29(2), 19–25, 45. https: //doi.org/10.13966/j.cnki.kfjyyj.2023.02.003

[2] Lu, Y., Yu, J., Chen, P., & Li, M. (2023). Educational applications and prospects of generative AI: A case study of the ChatGPT system. *China Distance Education*, 43(4), 24–31, 51. https: //doi.org/10.13541/j.cnki.chinade.20230301.001

[3] Wu, Y. (2023). A preliminary discussion on programming course instructional design in vocational colleges based on AIGC: Taking software technology programming training courses as an example. *Frontiers of Educational Research: Chinese and English Edition*, 13(3), 7–11.

[4] Zhang, X., Gao, Y., Cao, S., Tao, J., & He, X. (2024). Exploring the application of AIGC in biochemistry and molecular biology laboratory teaching. *Basic Medical Education*, 26(5), 406–411. https: //doi.org/10.13754/j.issn2095-1450.2024.05.12

[5] Li, Y., & Zheng, Y. (2023). Educational applications of generative AI. *People's Forum*, (23), 69–72.

[6] Shang, Z., & Yan, Y. (2023). ChatGPT educational applications and the resulting transformations and ethical challenges. *Journal of Northeast Normal University (Philosophy and Social Sciences Edition*, (5), 44–54. https: //doi.org/10.16164/j.cnki.22-1062/c.2023.05.006

[7] Xie, Y., Chen, F., & Zhong, Z. (2015). Research on micro-course design patterns based on problem-solving. *China Distance Education*, (5), 48–54, 80. https: //doi.org/10.13541/j.cnki.chinade.2015.05.008

[8] Fredricks, J. A., Wang, M.-T., Linn, J. S., Hofkens, T. L., Sung, H., Parr, A., & Allerton, J. (2016). Using qualitative methods to develop a survey measure of math and science engagement. *Learning and Instruction*, 43, 5–15.