International Journal of Computer Science and Information Technology for Education
Vol.6, No.1 (2021), pp.53-64
http://dx.doi.org/10.21742/ijcsite.2021.6.1.05

Comparative Analysis of Programming Education Tools Based on Computational Thinking

Javier Teresa

University of Alicante, Spain
Javier.teresa@ua.es

Abstract

With the rapid development of intelligent technology, the era of "artificial intelligence education" has come, and programming education has received increasing attention. This article collects typical programming education tools from abroad, establishes the evaluation dimensions, and compares and analyzes each tool. Then, various programming education tools were tested, and based on the comparative analysis framework, the comparative analysis was mainly conducted from three aspects: computational thinking, education, and gameplay, and their characteristics were summarized. In combination with the actual situation, suggestions for the development and use of programming education tools are put forward. With the development of the times, the application of programming education tools will become more and more extensive. Countries need to strengthen their development and research, so that programming education tools can be used flexibly in the classroom, and continue to promote the development of programming education in primary and secondary schools.

Keywords: *Programming education, Computational thinking, Analysis framework, Programming tools*

1 Introduction

With the rapid development of intelligent technology, countries around the world have begun to deploy strategic plans, trying to seize the high ground of research and practice in the field of artificial intelligence [1]. To increase the speed of national development, it is necessary to implement a national intelligence education project, set up artificial intelligence-related courses in primary and secondary schools, and gradually promote programming education. Programming education tools are undoubtedly an important auxiliary tool for programming education, which can cultivate learners' computational thinking and problem-solving skills. Therefore, if you want to vigorously develop programming education, you need to develop and use programming education tools reasonably.

Throughout the development of programming education in various countries, most of them follow a top-down development model that is promoted by the linkage of government, enterprises, society, and schools [2]. Since 2012, the United States, the United Kingdom, Australia, Singapore, Japan, South Korea, etc. have begun to attach importance to programming education, and successively promulgated policies to include programming in the compulsory courses of primary and secondary schools [3]. In addition, the "One Hour of Programming" campaign initiated by the American non-profit organization Code.org has received support

Article history:
Received (January 13, 2021), Review Result (February 15, 2021), Accepted (March 17, 2021)

Print ISSN: 2205-8370, eISSN: 2207-5372

from major companies such as Apple, Microsoft, and Facebook. Integrating a variety of programming education tools, allowing learners to use visual programming languages, experience programming thinking in game situations, and learn programming methods. Moreover, as an important place for programming education, schools not only actively respond to the "One Hour of Programming" activity but also flexibly apply a variety of programming education tools to classroom teaching. For example, some elementary and middle school teachers integrate programming with other subject knowledge and use the programming education tool Scratch studio to explain animal characteristics and their habitats in biology classes.

The national education community has gradually attached great importance to the importance of programming education for young people. Social financing and various special funds have continued to flow into the field of programming education. Internet education companies have increased the research and development of programming education tools. However, the localization of transformation and application needs to be deepened, and it is necessary to develop and use programming education tools with national characteristics by the actual situation of the country. Therefore, this article uses literature research methods, comparative research methods, etc., to fully comb and analyze the programming education tools of various countries, to provide some suggestions for the development and use of national programming education tools.

2. Comparative analysis of programming tools

2.1. Computational thinking

Programming education tools are important teaching aids for cultivating children's computational thinking [4]. Studies by foreign scholars have shown that educational tools make programming easy to implement and very attractive, helping to teach children how to program and develop their algorithms and computational thinking skills [5][6].

Computational Thinking was first proposed by Professor Jeannette M. Wing in March 2006 in Communications of the ACM, an authoritative computer publication in the United States. Professor Zhou defines computational thinking as Computational thinking is a series of thinking activities covering the breadth of computer science, such as the use of basic concepts of computer science for problem-solving, system design, and understanding of human behavior [7]. For information technology education in elementary and middle schools, the views of Dr. Selby and Dr. Woollard of the University of Southampton on computational thinking are more applicable, and this view has been widely recognized [8]. They believe that computational thinking includes five elements: Algorithmic Thinking, Evaluation, Decomposition, Abstraction, and Generalization. The specific connotations are shown in [Table 1].

Table 1 Five Elements of Computational Thinking

Computational thinking	Algorithmic thinking	Algorithmic thinking can be seen as a form of thinking shared by humans and machines. It represents such a process: It is composed of a series of clearly defined limited operation steps and can solve a specific problem.
	Evaluation	Evaluation is a process that emphasizes whether a solution is appropriate, or when designing a solution, it is necessary to think about how to improve it to ensure that the solution is the best method.
	Break down	Decomposition refers to the decomposition of a complex problem or system into several small problems that are easy to manage and easy to solve.
	Abstract	The abstract is the process of filtering and hiding unnecessary details, grasping the main characteristics of things, to make things simple.

	Overview	Generalization is to find a general way to solve a type of problem and to be able to develop solutions to similar problems. It is a way to solve new problems based on previous problem solutions.

2.2. Educational and game features of programming game software

The main function of programming educational tools is to cultivate the computational thinking and problem-solving abilities of primary and middle school students, which must have a certain educational nature. Education is mainly reflected in the fact that when students use certain educational tools, they are generally motivated by specific learning motivations, such as training their programming skills or training logical thinking. Then, in the process of using, the knowledge and information fed back by the tool can promote the growth of students and cultivate their knowledge, skills, intelligence, emotions, attitudes, values, etc. In addition, teachers can use programming educational tools to assist teaching and flexibly apply them in the teaching process.

The existing programming education tools for primary and middle school students are mostly in the form of game breakthroughs, including a variety of game factors such as storylines, roles, and various challenging tasks. They have the characteristics of fun, challenge, and excitement of games. When learning with the help of programming education tools, these features can increase students' interest in learning and thus improve learning efficiency. It can be seen that programming educational tools are aimed at education and games are used as a means to meet the needs of students to "play" and to allow them to "learn" some knowledge while playing, achieving a blend of education and gameplay [9].

Through literature survey, the author refers to the comparative analysis of educational software and believes that programming game software is mainly researched from the two dimensions of education and gameplay. For example, the evaluation criteria proposed by foreign scholars Annetta and Lamb-Stone in 2011 include indicators such as interaction, feedback, immersion, sense of control, difficulty, rules, learning content, learning goals, and teaching effects [10]. Therefore，this article evaluates the two dimensions of education and play. Educational features include four sub-dimensions: clear and timely knowledge feedback, logical level design, reliable content, flexible and diverse forms, and clear learning goals; gameplay includes moderate challenges, reasonable incentives, optional learning content, attractive plots, and clarity Five sub-dimensions of the rules of the game. The specific connotation of each sub-dimension is shown in [Table 2].

Table 2. Educational and playfulness

Educational	Knowledge feedback is clear and timely	Whether the assessment tool gives learners' knowledge feedback information or learning evaluation results on time. If the number of stars obtained by the learner is used as the evaluation result, it is considered that the knowledge feedback is not obvious
	Level design is logical	Whether the design of the assessment tool conforms to learners' cognitive laws and general life logic, that is, knowledge from easy to difficult, from concrete to abstract
	Reliable content, flexible and diverse forms	Whether the knowledge contained in the assessment tool is accurate, scientific, and presented in various forms
	Clear learning goals	At the beginning of each level, clearly explain the knowledge points of the level, such as the programming topics involved, so that learners can know their learning goals
Gameplay	Moderate challenge	Moderate difficulty, not too high or too low, can stimulate learners' enthusiasm for learning
	Reasonable incentives	Set rewards to encourage learners to learn

	Optional learning content	Learners can choose their learning content
	Attractive plot	Learners can be attracted by the storyline in the process of using tools for learning
	Clear rules of the game	Learners can clearly know the rules of this tool, helpful hints

3. Comparative analysis of programming education tools

3.1. Collection and screening of programming education tools

First of all, this article collects several programming educations tools through the literature survey method. Secondly, by searching popular application stores, I paid attention to the easy-to-access software tools on mainstream mobile platforms and desktop platforms. Finally, collect more recommended tools from the public through international search engines. Through the above three ways, a total of 31 programming education tools suitable for elementary and middle school students were obtained. To ensure the usability and applicability of each tool, we tested all available tools, and browsed the tool introduction, usage, and comments to determine the basic information and programming topics of each tool, and removed outdated and inaccessible tools. Filter out, and judge the tool according to the determined evaluation dimension. Finally, 16 programming education tools have been determined and their basic information is provided, mainly including release time, type, platform, language, block or text, applicable age, and programming foundation, so that students, parents, or teachers can choose more suitable for their situation programming education tool. as shown in [Table 3].

Table 3 Basic information of programming education tools

Name	Time	Types	Platform	Language	Block or text	Applicable age	Programming basics
Cargo-Bot	2012	Puzzle	IOS	Eng	Piece	9-11/10+	Beginner
Daisy the Dinosaur	2016	Puzzle	IOS	Eng	Piece	6-8	Beginner
Kodable	2014	Puzzle	Web, IOS, Android, Windows	Eng	Piece	6-8	Beginner
Robozzle	2009	Puzzle	Web, IOS, Android, Windows	Eng	Piece	4+	Beginner
Ruby Worrior	2008	Puzzle strategy	Web	Eng	Text	Unlimited	Beginners, intermediates, and even programmers
Light-Bot	2011	Puzzle	Web, IOS, Android	30	Piece	4-8/9+	Beginner
Turtle Academy	2016	Puzzle	Web	Eng	Text	Kindergarten students, elementary school students	Beginner
Tynker Games	2016	Puzzle Adventure	Web	Eng	Piece	4-7/7-13/13 above	Divided into beginner, intermediate and advanced
Save The Animals: Coding Game	2016	Puzzle	Android, IOS	Eng, Chi	Piece	4+	Beginner

Digital Puppet -Programming	2016	Puzzle-solving	Android	Eng	Piece	4+	Beginner
Run Marco	2016	Puzzle Adventure	Web, IOS、Android、Kindle	30	Piece	4+/6-12	Beginner
Blockly Games	2011	Puzzle	Web	50	Block/ text	5-12	Beginner
Coding games	2018	Puzzle	Web	Eng, Fran	Text	Unlimited	Certain foundation, even senior programmer
Code Karts	2017	Puzzle	iPhone, iPad	21	Piece	4+	Beginner
Code Combat	2013	Role-playing strategy puzzle	Web	50	Text	13+（Younger but need guidance）	Beginner（Beginner to intermediate level developer）
Kid's coding	2018	Puzzle	IOS	Eng, It, Po, Es	Piece	4+	Beginner
Scratch studio	2007	Puzzle	IOS, Web	47	Piece	8-16	Beginner

The role of programming education tools is to help students learn programming knowledge, should have basic programming topics-variables, conditions, loops, and methods. Regarding the programming topics of each tool, the author has made statistics as shown in [Table 4].

Table 4. Programming topics of programming education tools

Name	Variable	Condition	Cycle	Method
Cargo-Bot		X	X	X
Daisy the Dino		X	X	X
Kodable	X	X	X	
Robozzle			X	X
Ruby Worrior	X*	X	X	X
Light-Bot			X	X
Turtle Academy	X	X	X	
Tynker Games	X	X	X	X
Save The Animals：Coding Game		X	X	
Digital Puppet -Programming			X	X
Run Marco		X	X	
Blockly Games	X	X	X	X
Coding games	X	X	X	X*
Code Karts			X	X
Code Combat	X	X	X	X
Kid's coding			X	X
Scratch studio	X	X	X	X

Note: X means the characteristic is obvious; X means the characteristic is not obvious; blank means the characteristic is not reflected.*

3.2. Comparison of programming education tools

(1) Comparative analysis of basic information

Through comparative analysis of the type, platform, language, code presentation form, applicable age, and programming foundation of programming education tools, we can find that:

First, the types of tools are mostly educational and relatively single. Although the storyline, learning content, and challenging tasks are different, the operations and types of tools are similar. In particular, many block-based programming education tools have different shapes of code blocks, some are arrows pointing to different directions, and some are icons that characterize an action, but the specific operation is to combine these code blocks first, and then verify whether the movement process of the object is correct. It is worth mentioning that one of the tools fills in the blanks with code blocks. For example, at a certain level, a cartoon eye needs to be selected. When the learner writes the code of "select doll eyes", a series of small ball eyes will pop up for the user to choose. This form is relatively new and easy for learners to understand.

Second, the platform compatibility is strong. More than half of the programming education tools have mobile terminals, which are easy for learners to use anytime and anywhere. More than half of the tools have a web page, and you can use the tool when you enter the web page without special downloading. All tools support the IOS system.

Third, programming education tools only support the official language of the country and English, and only a few tools support multiple languages. This shows that the country still needs to vigorously strengthen its development of programming education tools.

Fourth, in terms of code presentation, 11 of the tools only support block-based programming, 4 only support text-based programming, and only 1 is both block-based programming and text-based programming. This type of programming education Tools can better meet the needs of learners and promote their understanding of programming knowledge. For example, Blockly Games first allows learners to use blocks for programming, and a prompt will pop up every time a level is completed. The prompt content is the text code corresponding to the code block used in this level. In the last few levels, let learners use text to program.

Fifth, from the perspective of applicable age, almost all tools are suitable for K12 students, that is, suitable for users aged 4 to 18 years old. More than half of the tools are over 4 years old. Relatively speaking, these tools are relatively simple and suitable for beginners.

Sixth, in terms of programming basics, some tools set different challenges for students of different ages. For example, Tynker Games is divided into three age groups, namely beginner, intermediate, and advanced. Secondly, the tools that are suitable for ages over 4 or 5 years old do not require a programming foundation. Some tools that are suitable for older age require users to have a certain programming foundation, such as Coding games. This tool even provides advanced programmers. Tier challenge.

(2) Comparative analysis of programming topics

Among the surveyed tools, the learning content of 6 tools included all the basic programming topics above, and the other 10 included only partial programming topics. In terms of the presentation form of programming topics, Light-Bot, Turtle Academy, etc. treat variables, conditions, loops, and methods as independent levels in the form of topics. Cargo-Bot, Scratch studio, etc. did not clearly separate them but integrated them into the actual game situation.

Different programming education tools require learners to master basic programming topics to different degrees. Take the loop as an example. Some tools only require the user to know that the loop needs to be used, but the user does not need to calculate the specific number of

loops. Most tools will allow users to consider the number of cycles needed when using loops. This method has higher requirements for learners than the previous one.

The presentation of the corresponding instructions in the programming theme of each tool is also different. First of all, as far as variables are concerned, in text-based tools, variables are defined directly in the form of declaring variables in a high-level language, including defining the type and name of the variable. In block-based tools, users only need to click the button to create a variable and give it a name without considering other rules. Secondly, the presentation form of the "condition" structure is also different. Conditional statements in text-based programming tools are similar to high-level languages and need to meet certain grammatical rules. In block-based tools, conditional statements can be represented by a specific code block. For example, Cargo-Bot, Robozzle, and Kodable use code blocks of different colors as conditions. In addition, in terms of the loop structure, text-based games directly present loops in the form of code, which is relatively abstract and not easy to understand. The block-based tool will present the cyclic structure with vivid and concrete graphics and icons. Finally, in terms of methods, most block-based tools limit the amount of code in the main function. When too many code blocks are needed to complete the goal, the user can only write some instructions in a method, and then call the method in the main function.

(3) Comparative analysis of computational thinking, education, and gameplay

According to the determined comparison framework, the following is a comparative analysis from the three aspects of computational thinking, education, and gameplay of programming education tools. Once proposed, computational thinking has attracted wide attention from all walks of life. It has been influenced by the International Computer Association (ACM) CC2001 (Computer Course) curriculum system (draft), and more and more attention is paid to cultivating children's computational thinking. Programming education tools are important teaching aids for cultivating children's computational thinking. Therefore, using it as a criterion for judging a programming education tool, the following analysis of programming tools is made according to the five elements of computational thinking-algorithm thinking, decomposition, abstraction, generalization, and evaluation, as shown in [Table 5].

Table 5. Computational thinking of programming education tools

Name	Algorithmic thinking	Decompose	Abstract	Generalize	Evaluation
Cargo-Bot	X	X	X	X	X
Daisy the Dino	X	X	X	X	
Kodable	X	X	X	X	X*
Robozzle	X	X	X	X	
Ruby Worrior	X		X	X	
Light-Bot	X	X	X	X	
Turtle Academy	X	X	X	X	X*
Tynker Games	X	X	X	X	X*
Save The Animals: Coding Game	X	X	X	X	X
Digital Puppet -Programming	X	X	X	X	X*
Run Marco	X	X	X	X	
Blockly Games	X	X	X	X	
Coding games	X	X	X	X*	X
Code Karts	X	X	X	X	
Code Combat	X	X	X	X	
Kid’s coding	X	X	X	X	
Scratch studio	X	X	X	X	

Note: X means the characteristic is obvious; X means the characteristic is not obvious; blank means the characteristic is not*

reflected

All the tools tested, reflect the cultivation of algorithmic thinking. That is, all tools allow users to solve problems through clearly defined steps. For problem decomposition, the above programming education tools almost all reflect the problem decomposition process, that is, learners need to decompose the task in advance, decompose a large goal into several small goals, and achieve the final goal by completing the small goals.

In addition to the above four factors, computational thinking also includes evaluation, that is, users' reflection and evaluation on the effectiveness and conciseness of their programs. Among the currently tested tools, only a small part can give corresponding feedback based on the running time and redundancy of the user code. For example, when completing a task, use the number of stars obtained as feedback on the quality of the user code. In addition, there are individual tools that will give more detailed feedback information, and in Cargo-Bot, users will be prompted with the best solution for each level.

In addition to the important factor of computational thinking, the integration of education and gameplay is also the key and difficult point for the success of programming education tools. Excessive emphasis on education and lack of gameplay will not attract students' interest. Excessive emphasis on gameplay and the content of education is not organically integrated with game activities, it is easy to attract students' attention to meaningless game activities, thus failing to achieve the educational goal. Therefore, this article analyzes the educational and playfulness of programming education tools as shown in [Table 6].

Table 6. Educational and gameful nature of programming education tools

Name	Educational	Gameplay
	1. Knowledge feedback is clear and timely； 2. Level design is logical； 3. Reliable content, flexible and diverse forms； 4. Clear learning objectives	1. Moderate challenge； 2. Reasonable incentives； 3. Optional learning content； 4. Attractive plot； 5. Clear rules of the game
Cargo-Bot	1* 2 4	1 2 3 5
Daisy the Dino	2 4	1 5
Kodable	1* 2 3 4	1 2 3 4 5
Robozzle		3
Ruby Worrior	3 4	1 4 5
Light-Bot	2 4	2 3 5
Turtle Academy	2 3 4	1 3 5
Tynker Games	1 2 3 4	1 2 3 4 5
Save The Animals：Coding Game	2 3 4	2 4* 5
Digital Puppet - Programming	1* 2	1 2
Run Marco	2	1 4* 5
Blockly Games	1* 2 3 4	1 3 4 5
Coding games	1 2 3 4	3 4 5
Code Karts	2 3 4	1 2 5
Code Combat	2 3 4	1 2 4 5
Kid’s coding	2 3 4	1 3 4 5
Scratch studio	2 3 4	1 3 4 5

Through the above comparison, it can be found that in terms of knowledge feedback, only the knowledge feedback of individual tools is clear and timely. The specific performance is: at the end of the level, timely feedback information reflects whether the user's code is efficient. This can be flexibly applied to teachers' classrooms. The survey found that some foreign teachers ask students to complete the challenge on their own, and then discuss the best solution with each other based on the evaluation feedback given by the tool. In terms of level design logic, the difficulty of almost all tools is gradually increasing, and programming knowledge is gradually deepened, which conforms to the general cognitive law of students. In Ruby Worrior, the difficulty is divided into beginner and intermediate. The programming knowledge of the tools has a certain degree of scientificity, but the content and form are quite different. For example, Blockly Games has mazes, birds, tortoises, and other different themes to breakthrough, and the content is rich and diverse. Robozzle has a single form of content. In terms of learning goals, most tools have clear goals. Block-based tools are more focused on training students' programming thinking, and text-based tools are more about training students' ability to write code.

First of all, in terms of the difficulty of the tools, most of them are of moderate difficulty, and some of them are relatively difficult. For example, Coding games and the lower levels of Light-Bot are too difficult, which is likely to discourage learners. Second, most tools have certain incentive mechanisms. In Save The Animals: Coding Game, learners can get medals by breaking through levels, and there are some tools to reward stars, which can inspire students to continue to challenge. In addition, there are 10 tools to choose the content of learning, that is, users can choose the level independently, rather than just follow the established route to go through the level, which increases their control. The storyline of the 8 tools is interesting and attractive. For example, Tynker Games explores the universe and composes the storyline of music. Finally, in terms of rules, except for Robozzle and Digital Puppet-Programming, all other tools have clearer rules. Most of them are for the first time to guide novices, and only a few tools use special sections to introduce the rules.

4. Suggestions for the development and use of programming education tools

4.1. Suggestions for developing programming education tools

First, in terms of education, a complete knowledge system is required in programming education tools. Among the tools tested, only a few included all the most basic programming topics. If the theme of the tool is incomplete, that is, the knowledge system is incomplete, it will not be able to support the development of programming education.

In addition to the integrity of programming topics, programming knowledge needs to be added to programming education tools. During the tool experience, I found that many tools did not have knowledge explanations or help tips. For example: in the process of breaking through, when reaching a certain level, a loop structure will suddenly appear. The learner wants to use the structure to continue through the barriers, but its function and specific method of use are not explained. In this case, users with programming knowledge can easily pass, but beginners may be difficult to understand, and then feel confused about the rules of the game. For programming education tools, developers should not pay too much attention to gameplay and neglect education. Instead, they should take education as the purpose and use games as a means to guide students to learn and use programming knowledge in development and design.

In addition, different learners have different cognitive development stages and cognitive structures. Therefore, when developing tools, it is necessary to clarify the applicable objects

and teach students by their aptitude. For example, younger or beginners can use a combination of blocks and codes to present programming knowledge.

Second, in terms of gameplay, developers need to design tools with more diversified types and contents. The current programming education tool type is relatively single, and the operation process of block-based programming tools is relatively similar, which makes most games lack innovation and interest, and it is not easy to highlight the tool features. The development team can appropriately add interesting game situations or elements to increase the fun of the tool. For example, Save The Animals: Coding Game combines the concept of protecting wild animals with programming, using rescue animals as a story setting, designing nature reserves and achievement sections, which can effectively stimulate students' interest in learning. In addition, although with the development of the times, many games tend to be idiotic, users can solve problems by exploring on their own. But for beginners, programming knowledge is a certain degree of difficulty. Developers need to provide more detailed game rules and help documents to prevent users from spending too much time exploring, or being stuck at a certain level and unable to move forward. Destroy the enthusiasm of users. The description and design of the game rules can take many forms and use a variety of media. For example, it can be a description document in the form of text or a combination of graphics and text, or a video describing the rules. Only in this way can it meet the diverse needs of different users.

Third, in terms of computational thinking, focus on cultivating students' ability to evaluate codes. From the analysis results, it is found that most programming education tools do not pay attention to the element of evaluation in computational thinking. For example, Digital Puppet-Programming uses the number of stars obtained as the feedback of the game. There are three stars in total. When the learner clears the level, the system will determine the number of stars obtained according to the efficiency of the code. Such feedback information is too simple and Can not highlight the characteristics of programming games, any breakthrough game can use the number of stars as feedback information. Assessment helps learners develop the habit of reflection and improve their critical thinking skills. Developers need to use feedback information in the game to remind students to evaluate and improve their code. The feedback information should be as detailed as possible. For example, you can point out the optimal code in this level, the optimal running speed of the code, and the running speed of the code written by the learner.

Fourth, in terms of curriculum integration, developers need to develop programming education tools that are highly compatible with the curriculum. At present, few tools are developed to match the curriculum, leading teachers to spend a lot of energy choosing the right tools when using them. Therefore, educational software companies can establish cooperative relationships with schools, understand the needs of schools, and develop tools that match the curriculum. You can also develop programming education tools with home, school, and students.

4.2. Suggestions for using programming education tools

First, choose appropriate programming education tools according to different age groups. The elementary stage (4-8 years old) is the younger students, they have almost no programming foundation, and the cognitive maturity is not high, they do not have strong abstract and logical reasoning ability. Therefore, this stage mainly focuses on the cultivation of enlightenment and thinking mode, so that they understand basic computer knowledge, cultivate interest in programming, cultivate cognition of logical sequence, and be familiar with and learn to use

procedural thinking, such as abstraction, classification, decomposition, etc. The requirements for learning programming languages are not high. At the intermediate level (8-12 years old), students already have a certain programming foundation, have a certain thinking mode, and can start to learn to program systematically. The learning content at this stage includes basic programming knowledge and advanced skills. Students can choose a suitable platform for systematic learning or select tools that can be used for the creation of works, such as Scratch Tutor, Kodable, etc.

In the advanced stage (over 12 years old), learners have a certain interest in programming and can choose a language for in-depth study according to their interests. However, programming has a certain degree of difficulty and involves a wide range of knowledge. Interest is not enough. It also requires perseverance to learn more knowledge (mathematics, physics, etc.) to continuously improve the learner's programming ability. Students at this stage can choose text-based programming, such as Coding games.

Second, make appropriate use of the gameplay of programming education tools to improve the fun of programming classrooms. At present, in programming teaching classrooms, teachers are more inclined to explain and impart basic knowledge of programming languages, classroom teaching is boring, and learners' learning enthusiasm is not high [12]. Gamified learning can effectively increase the interest of the classroom and enhance students' interest in learning. Many countries and regions in the world have also attached great importance to the role of games in education. Most teachers in the European Union, the United Kingdom, and the United States support the use of games in the classroom. Reasonable use of games can enhance the fun of the classroom, thereby enhancing the learning effect.

First, before explaining specific programming concepts, guide students to understand through tools. The concepts in programming are often abstract, such as loops, methods, and so on. But programming is maneuverable, and certain results can be obtained through debugging. Before teaching knowledge, teachers can let students use the tools and experience the corresponding programming topics, and provide students with advanced organizer materials to help understand specific concepts.

Secondly, grasp the balance point between students' self-action and knowledge explanation. Since that many tools do not have a knowledge system in line with cognitive development and are not developed for specific school courses, teachers need to add knowledge points while allowing learners to play games. The specific performance is the need to explain some knowledge in advance, and then let the students operate, or supplement the required knowledge in the process of the learner's operation of the tool.

Finally, focus on cultivating students' ability to accurately evaluate codes. Evaluation is one of the five elements of computational thinking, but among the testing tools, only a few tools guide learners to evaluate the code (block) they have written. Therefore, teachers cannot completely rely on the tools when using programming education tools. They need to guide students to evaluate and evaluate their codes in terms of code accuracy, efficiency (code execution time), conciseness (how much code), and convenience. Reflection.

Third, in the process of using programming education tools, cultivate students' self-control. Even if educational tools have a certain educational significance, they should not be overly addicted. These tools give students access to the Internet. They are easily lost in the virtual world of the Internet and cannot control their learning time. Teachers need to guide students to use them reasonably while using tools for teaching. The educational knowledge that programming education tools can convey is limited, and the real systematic knowledge learning still needs teachers to explain in the classroom.

5. Conclusion

This article collects typical foreign programming education tools, establishes the evaluation dimensions, compares and analyzes each tool, and puts forward suggestions for developing and using programming education tools based on the actual situation. However, the programming education tools collected are all from abroad, and there is a lack of comparative analysis of tools. In addition, the comparative analysis dimensions identified in this article are not comprehensive enough, and there is a lack of feedback from the users of programming education tools. Finally, with the development of the times, the application of programming education tools will inevitably become more and more extensive, and all countries need to strengthen their development and research so that programming education tools can be used flexibly in the classroom, and continue to promote the development of programming education in primary and secondary schools.

References

[1] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," Artificial Intelligence, **(2017)**

[2] Yi Yoojin and Euijune Kim, "Linkage among school performance, housing prices, and residential mobility, Sustainability, vol.9, pp.1075, **(2017)**

[3] A. Ahmed and A. Pollitt, "Curriculum demands and question difficulty," IAEA conference, Bled, Slovenia, **(1999)**

[4] S. N. Renny Lindberg, H. Teemu Laine, and L. Haaranen., "Gamifying programming education in K‐12: A review of programming curricula in seven countries and programming games," British Journal of Educational Technology, **(2018)**

[5] A. Khaled, T. Moanis, and R. Jamal, "Primary school pupils' attitudes toward learning programming through visual interactive environments," World Journal of Education, vol.6, no.5, **(2016)**

[6] C. Kora charkornradt, "TukTuk: A block-based programming game," Conference on Interaction Design and Children, ACM, **(2017)**

[7] M. Jeannette Wing, "Computational thinking," Communications of the ACM, vol.49, no.3, pp.33-35, **(2006)**

[8] K. M. Cammack, "A critical thinking activity on drug tolerance for undergraduate neuroscience courses," Journal of Undergraduate Neuroscience Education A Publication of Fun Faculty for Undergraduate Neuroscience, vol.15, no.2, pp.A157, **(2017)**

[9] R. Suzuki, J. Kato, and K. Yatani, "ClassCode: An interactive teaching and learning environment for programming education in classrooms, **(2020)**

[10] M. Paolieri, M. Biagi, and L. Carnevali, "The ORIS tool: Quantitative evaluation of non-markovian systems," IEEE Transactions on Software Engineering, no.99, pp.1-1, **(2019)**