Program Learning for Beginners: Survey and Taxonomy of Programming Learning Tools

Authors Name/s per 1st Affiliation (*Author*)
line 1 (of *Affiliation*): dept. name of organization
line 2-name of organization, acronyms acceptable
line 3-City, Country
line 4-e-mail address if desired

Authors Name/s per 2nd Affiliation (*Author*)
line 1 (of *Affiliation*): dept. name of organization
line 2-name of organization, acronyms acceptable
line 3-City, Country
line 4-e-mail address if desired

Abstract— **Programming is taught around the globe because it has become a vital skill. Occasionally a game or visual programming language tool designed for programming education is used to teach programming. In general, these tools have various attributes, which inhibit a great learning effect if the tool and learning objectives are not aligned. However, which tool is most appropriate for a given objective remains unknown. In this research, we propose a taxonomy table to evaluate program learning tools and demonstrate its usefulness by researching and comparing 43 kinds of program learning tools in the taxonomy table. This research should contribute to the selection of suitable tools for program learning.**

Keywords—Programing learning, Taxonomy, Programming learning tool

I. INTRODUCTION

Program learning environments such as Alice [1] or Scratch [2] are often used to teach programming to first time learners. Because these tools vary with regard to visual languages, game-software, compatibility with hardware, tangible-devices, and unplugged devices, an appropriate tool must be selected for users (learners and educators), attributes (ages and programming experience), and learning objectives. Which attribute or tool type is best for a specific purpose is poorly understood. Consequently, creating a taxonomy would help users select the appropriate tool. This survey addresses the following research question:

- **Research Question (RQ)**: Can a taxonomy group, evaluate, and compare programming learning tools effectively?

The contributions of this research are:

- A taxonomy table, which can compare and evaluate tools based on a standard protocol, is created.
- The taxonomy table aids users in selecting tools with appropriate attributes for the learning objective.

The rest of this paper is organized as follows. Section 2 describes the background of this research. Section 3 overviews the research method, while section 4 proposes our taxonomy. Section 5 shows the classification results. Section 6 discusses the RQ and the threat against validity. Section 7 introduces a relevant study, and section 8 provides the conclusion and future work.

II. BACKGROUND

A. *Programing learning environments*

Generally, a programming environment is used to learn to program, but the specific environment varies according to the purpose (e.g., business or learning).

We focus on programming environments (especially, tools for children) for beginners learning to program. For example, there are visual programming environments such as Scratch [2] and Alice [1] as well as are game software environments such as CodeCombat [3] and Lightbot [4]. Hence, various programming environments for learning exist. These environments have been applied to teach programming to beginners. Several studies have shown the learning effects by programming learning [5][6]. However, one study showed a difference in the learning effect due to the programming method and expression [6]. Qualitatively capturing the characteristics of each environment is an important factor to expand the learning effect.

B. *Environment survey*

Caitlin Kelleher et al. [7] investigated dozens of programming environments by classifying them into categories. Then programming environments were evaluated using the same taxonomy. Unlike Kelleher et al., which included numerous programming environments, this study focuses on programming learning environments for children to create a taxonomy table optimized for helping users [educators and learners (children)] select the environments referred to in [7]. Additionally, we evaluate the programming learning tools intended for programming education with our taxonomy table. The tools targeted in this thesis are visual languages, game software, and other software that work alone on PCs (including tablets and other devices) because the available tools have drastically increased.

III. RESEARCH METHOD

Many programming learning tools have been developed. We characterize these tools and investigate the types of tools that are currently available.

A. *Method to Select Tools*

To develop a method to survey program learning tools from the literature, we referred to the study by Kai Petersen et al. [8] because it is often used for comprehensive investigations of the literature. First, we searched the Web using a Google Custom Search API with eight sets (Japanese: 4 sets, English: 4 sets) of keywords. Table 1 shows the keywords, where keywords in the same row have same meaning in Japanese and English. The top 100 search results for each set of keywords were used, providing a total of 800 results. Then we extracted the programming learning tools by morphological analysis and visual observations. We identified 54 tools. These tools were further refined by only considering software working on a device such as PC or tablet. Therefore, we surveyed 43 kinds of tools. We divided the tools into three fields: visual programming environments (Visual), game software (Game), and other educational software (Other). Tools were classified according to the text on each tool's official website. Table 2 shows a list of survey tools.

TABLE I. KEYWORD LIST

Japanese	English
プログラミング 学習 子ども　ゲーム	Programming learning tool game
プログラミング 学習 子ども　ツール	Programming learning tool children
プログラミング 教育 子ども　ゲーム	Programming education game children
プログラミング 教育 子ども　ツール	Programming education tool children

TABLE II. TOOLS LIST

ID	Name	Field
T1	Alice	Visual
T2	Ardublock	Visual
T3	Blockly	Visual
T4	MOONBlock	Visual
T5	Pyonkee	Visual
T6	Scrach	Visual
T7	Scratch Jr.	Visual
T8	SmalRuby	Visual
T9	Viscuit	Visual
T10	Greenfoot	Visual
T11	Hopscotch	Visual
T12	Kodu	Visual
T13	LearnToMod	Visual
T14	Programin	Visual
T15	BetaTheRobot	Game
T16	Bo1 Island	Game
T17	BotLogic.us	Game
T18	Code Monster	Game
T19	Code Studio	Game
T20	CodeCombat	Game
T21	Code-Girl Collection	Game
T22	CodeMonkey	Game
T23	Crunchzilla	Game
T24	Daisy the Dinasaur	Game
T25	Empire of Code	Game
T26	Erase All Kittens	Game
T27	Flappy	Game
T28	HackforPlay	Game
T29	Junior Coder	Game
T30	Lightbot	Game
T31	Move the Turtle	Game
T32	Penjee	Game
T33	RoboMind	Game
T34	Run Marco!	Game
T35	Tech Rocket	Game
T36	The Foos	Game
T37	Tickle	Game
T38	Turtle Academy	Game
T39	JointApps	Other
T40	Learn Python	Other
T41	Squeak	Other
T42	Swift Playgrounds	Other
T43	Tynker	Other

IV. TAXONOMY

A. *Taxonomy classification*

We created a taxonomy table to evaluate program learning tools qualitatively (Table 3) by referencing Kelleher et al. [7]. Specifically, we optimized Kelleher's table for learning tools and added the following categories: Game Elements and Requirements. We added game elements because playing a game is a suitable method to learn programming, especially programming concepts. In addition, the number of the games to learn programming such as CodeCombat [3] and Lightbot [4] has increased. In this survey, we considered game elements that deal with games. We used Rule/Restriction, Goal, and Reward (the common parts of the definition by Katie Seaborn et al. [9] and Juho Hamari et al. [10] to define game elements. From the viewpoint of multi-play, we also added Cooperation [11]. The classification in the taxonomy table has 11 categories for the 43 items. Classification details are explained below.

B. *Taxonomy details*

Style of Programming (C1) has six entries, which explain the program style built into the tool. These include procedural, functional, object-based, object-oriented, event-based, and state machine-based.

Programming Construct (C2) reflects the programming construct, which can be learned in a tool. Items include conditionals, loops, variables, parameters, procedures/methods, user-defined data types, pre-and-post conditions, and recursions. In this survey, all types of loops are lumped together because they are the same from the viewpoint of teaching the concept of a loop. We added recursion as some tools teach this concept.

Representation of Code (C3) explains how to display programs. Items include text, pictures, flow charts, animations, forms, finite state machines, and physical objects.

Construction of Programs (C4) describes how to input programs. Items include typing code, assembling graphical objects, demonstrating actions, selecting/form filling, and assembling physical objects.

Support to Understand Programs (C5) focuses on help understanding programs. Examples include back stories, debugging, physical interpretations, liveliness, and generating examples.

Designing Accessible Language (C6) explains the functions to make programming languages easier to learn. Items include limiting the domain, selecting user-centered keywords, removing unnecessary punctuation, using natural language, and removing redundancy.

Game Elements (C7) is a new category because we think that the presence or absence of game elements affects the learning effect. It represents the game element included in a tool. Examples include rewards and goals, as explained in the previous section.

Supporting Language (C8) is the language used in each tool. This is newly added because whether users can understand the description of the tools or not is relevant to the learning effect. Supporting languages were classified as English, Japanese, and others.

Operating Environment (C9) is the environment where each tool works. We added category because how to start and use a tool is important aspect of usability. We classified the Operating Environment into Windows, Mac, Linux, Android, iOS, Web, and other.

Interface (C10) explains the suitable device to use the tools. We added this for the same reason as Operating Environment. We classified into PC, Tablet, and Smartphone, and other.

Experience (C11) explains whether each tool targets a novice programmer. This is added because our research aims to survey program learning tools for children without programming experience.

TABLE III. TAXONOMY

Style of programming (C1)	**Programming constructs (C2)**	**Representation of code (C3)**	**Construction of programs (C4)**	**Support to understand programs (C5)**	**Designing Accessible Languages (C6)**
procedural (i11)	conditional (i21)	text (i31)	typing code (i41)	back stories (i51)	limit the domain (i61)
functional (i12)	loop (i22)	pictures (i32)	assembling graphical objects (i42)	debugging (i52)	select user-centered keywords (i62)
object-based (i13)	variables (i23)	flow chart (i33)	demonstrating actions (i43)	physical interpretation (i53)	remove unnecessary punctuation (i63)
object-oriented (i14)	parameters (i24)	animation (i34)	selecting/form filling (i44)	liveness (i54)	use natural language (i64)
event-based (i15)	procedures/methods (i25)	forms (i35)	assembling physical objects (i45)	genereated examples (i55)	remove redundancy (i65)
state machine-based (i16)	user-defined data types (i26)	finite state machine (i36)			
	pre and post conditions (i27)	physical objects (i37)			
	recursion (i28)				
Game elements (C7)	**Supporting Language (C8)**	**Operating Environment (C9)**	**Interface (C10)**	**Experience (C11)**	
Rule/Restriction (i71)	Japanese (i81)	Windows (i91)	PC (i101)	unnecessary (i111)	
Goal (i72)	English (i82)	Mac (i92)	Tablet(8inch~) (i102)	necessary (i112)	
Rewards (i73)	others (i83)	Linux (i93)	Smartphone (i103)		
Cooperation (i74)		iOS (i95)	Web (i94)		
		Android (i96)	Other Interface (i104)		
		Other Environments (i97)			

V. RESULT AND ANALYSIS

A. *Overview of the results*

We surveyed the features of the programming learning tools. As a classification method, two people separately evaluated each tool using the following process: (1) Read the words on the official website of each tool. (2) Use each tool. (3) Verify the classification in the taxonomy table. (4) Cross-check the classification results of the evaluators.

Table 5 lists the taxonomy tables, which show the attributes of the tools. Furthermore, Figure 1 shows the corresponding number of tools for each attribute. Several tools have multiple attributes. Additionally, some attributes may be applicable to other fields (e.g., robot, unplugged tool). Therefore, additional research is necessary.

For the Style of Programming, procedural, which is the most basic concept, has the most entries as 25 tools are applicable. Visual programming environments have been applied to object-oriented in Style of Programming. Because procedural and object-oriented are basic styles of programming, many tools are being developed for these aspects.

For the Programming Constructs, five entries are supported by more than half of the tools: conditionals, loops, variables, parameters, and procedures/methods. These are important concepts for programming. In particular, 28 tools have incorporated conditions and loops as basic programming concepts, indicating that many tools teach the logic of programming.

For the Representation of Code, 90% of the tools use text. Such tools refer to general languages, allowing users to learn programming in a style that closely resembles regular programming or to understand programs in a natural language. Additionally, some tools such as Lightbot use pictures to represent programs. These tools allow programming to be more intuitively understood than text-based ones.

In Construction of Programs, assembling graphical objects, which is a way to visualize language, has the most applicable tools. Although some tools demand users to type code, many tools enable users to input code by dragging and dropping. This is because the tools are developed for children who may not be proficient at typing or using a keyboard.

In Support to Understand Programs, physical interpretation has the most entries, meaning the code is expressed by a specific action such as “walk” or “jump” because these tools are developed for children.

In Designing Accessible Language, limit the domain, which is an attribute, has the most entries. Overall, 31 tools are applicable. Limiting the domain makes it easier for learners to understand programming.

In Game Elements, many tools include Rules/Restrictions and Goals. At least one game element is observed in 24 of the 43 software tools. Therefore, most tools are categorized as game software enabling users to learn programming by playing a game using game elements. The advantage of game software is that users can understand programs by watching an action rather than reading written instructions.

In Supporting Language, English was the most supported (36 tools). This is because many programming learning tools are often developed in the European and the American blocs. Some tools support many languages, enabling learners to learn in their own languages. Hence, this will lead to a better understanding of programming.

In Operating Environment, Web has the most entries. Because tools, which work on the Web, do not require a long time to prepare the environment, beginners can more quickly begin to learn to program. Additionally, some applications corresponding to tablets and smartphones are supported by some tools. These tools make program learning easier.

In Interface, PC has the most entries (33 tools). This means that most tools aim to teach users with a general language.

In Experience, over 90% (40 tools) of the tools can be used by beginners, suggesting that most tools are intended for beginners (especially for children).

B. *Results of each field*

1) *Visual programming environments*

There are 14 visual environments in the tools. Many visual tools are object-oriented programming environments and include basic programming concepts such as conditionals, loops, and procedures/methods. Learning to program is easier in a form close to real programming. As a representation of the code, text is used. Additionally, as a method of programming, many involve assembling graphical objects. Programming is possible by dragging and dropping. Not all visual tools possess game elements. In other words, these tools are not games, but are specialized to create programs.

2) *Game software*

There are 24 game software tools. Many of the game software are applied to procedural in the style of programming. or have Rules/Restrictions and Goals in the game elements. These game elements clarify learning goals. Therefore, game software is very suitable for introductory learning. In addition, we performed a cross tabulation with Game Elements, a newly added category, and Programming Constructs, the basic goal of programming learning. Table 4 shows the results. Many of the tools, which have Rules/Restrictions and Goals, include conditionals and loops. The reason for this is that showing the action of a conditional in a game helps users comprehend such concepts. Many games with these game elements, such as CodeMonkey[13] and Lightbot[4], are similar to Turtle Graphic. If users (educators and learners) want to learn conditionals and loops, which are logics of programming, they should select a game.

TABLE IV. RELEVANCE BETWEEN PROGRAMMING CONSTRUCTS AND GAME ELEMENTS

Game Elements	**Conditional**	**Loop**	**Variables**	**Parameters**	**Procedures/methods**	**Pre and post conditions**	**Recursion**
Rule/Restriction	12	11	8	10	9	2	2
Goal	15	13	11	13	12	2	2
Reward	6	6	5	3	5	1	1
Cooperation with Others	2	2	3	3	3	1	0

3) *Other educational software*

five tools are neither game software nor a visual language. There are many web services that gather programming learning applications. In addition, there is a tool to easily develop applications. Five programming expression tools are textual representations. For other items, there are individual characteristics for each tool. Furthermore, it is possible to break down the field of each tool.

C. *Summary of result*

Our taxonomy is sufficient to evaluate and compare tools because it contains attributes of each learning tool for programming. In addition, tools can be characterized by field (Visual, Game, and Other). Therefore, we can divide these tools into fields using this taxonomy.

VI. DISCUSSION

A. *Answer to the Research Question*

We investigated the following research question:

- **RQ**: Can a taxonomy group, evaluate, and compare programming learning tools effectively?

We derived a suitable taxonomy table based on Kelleher [1] to compare and evaluate programming learning tools as demonstrated by the fact that our taxonomy can classify all 43 tools. For example, many tools represent code by text and demand that code is inputted by assembling graphical objects. Tools with game elements are suitable to teach programming concepts. Therefore, an evaluation using a unified taxonomy is feasible. Moreover, tools can be selected by considering learning objectives.

B. *Threats to Validity*

One threat to validity is that the evaluation results may depend on the evaluator. Although two researchers crosschecked the findings in this survey, results involving more evaluators are necessary to confirm the conclusions.

In addition, the keywords used to extract the tools (Table 1) do not cover all tools for beginners. In this search, we targeted "children". However, an applicable tool may not be labeled as “for children”. From the viewpoint of the retrieval method, acquisition of high-quality data is a future subject.

Additionally, since we applied the results of a Google search, it is possible that older tools are excluded. These tools may have a more considerable influence than newer tools. Hence, classifying older tools is important.

TABLE V. CLASSIFICATION RESULTS

	C1						C2								C3							C4					C5					C6					C7				C8			C9							C10				C11	
	i11	*i12*	*i13*	*i14*	*i15*	*i16*	*i21*	*i22*	*i23*	*i24*	*i25*	*i26*	*i27*	*i28*	*i31*	*i32*	*i33*	*i34*	*i35*	*i36*	*i37*	*i41*	*i42*	*i43*	*i44*	*i45*	*i51*	*i52*	*i53*	*i54*	*i55*	*i61*	*i62*	*i63*	*i64*	*i65*	*i71*	*i72*	*i73*	*i74*	*i81*	*i82*	*i83*	*i91*	*i92*	*i93*	*i94*	*i95*	*i96*	*i97*	*i101*	*i102*	*i103*	*i104*	*i111*	*i112*
T1				x	x		x	x	x	x	x				x										x				x			x									x	x		x	x	x					x				x	x
T2	x						x	x	x	x	x				x								x									x									x	x	x	x	x	x					x				x	
T3	x						x	x	x	x	x				x								x								x	x									x	x					x				x				x	
T4			x				x	x	x	x					x								x						x			x									x	x					x				x				x	
T5			x		x		x	x	x	x	x				x								x						x			x									x	x						x				x			x	
T6			x		x		x	x	x	x	x				x								x						x			x									x	x		x	x	x	x				x	x			x	
T7			x		x			x		x					x								x						x			x									x	x						x	x	x		x			x	
T8			x		x		x	x	x	x	x				x								x						x		x	x									x			x			x				x				x	
T9			x					x								x							x							x		x												x	x		x				x	x			x	
T10				x											x							x						x														x		x	x	x				x	x					x
T11			x		x		x	x	x	x					x										x				x			x			x							x						x				x	x		x	
T12			x				x		x	x	x					x									x				x			x									x	x	x	x						x	x			x	x	
T13			x				x	x	x	x	x				x							x	x						x			x										x					x				x				x	
T14			x		x		x	x		x	x				x								x						x			x	x		x						x						x				x				x	
T15	x									x					x							x						x				x	x				x	x				x		x	x						x					x
T16	x						x	x							x	x							x				x		x			x			x		x	x	x		x	x	x					x				x	x		x	
T17	x									x					x	x						x			x		x		x			x					x	x		x		x					x			x					x	
T18	x						x	x	x	x	x				x							x								x								x				x					x				x				x	
T19			x				x	x	x	x	x		x		x								x						x		x	x	x	x	x		x	x			x	x					x				x				x	
T20	x						x	x	x	x	x		x		x							x					x		x			x					x	x	x	x	x	x	x				x				x				x	
T21	x								x						x							x					x										x	x	x		x						x				x				x	
T22	x						x	x	x	x	x				x							x			x		x		x			x					x	x	x	x	x	x	x				x				x	x			x	
T23	x						x	x	x	x	x				x							x								x								x				x					x				x				x	x
T24	x						x	x							x								x						x			x					x	x				x	x					x				x			x	
T25	x								x		x				x							x															x	x	x	x		x					x				x				x	x
T26										x					x							x					x			x							x	x				x					x				x				x	
T27	x				x								x		x								x						x			x			x		x	x			x	x					x				x				x	
T28				x						x					x							x					x		x								x	x			x						x				x				x	
T29	x						x	x			x				x	x							x				x	x	x			x			x		x	x	x			x						x				x			x	
T30	x						x				x					x							x						x			x					x	x				x		x	x		x	x	x	x	x	x	x		x	
T31	x						x	x	x	x	x			x	x	x									x				x			x					x	x	x			x						x				x	x		x	
T32	x						x	x	x	x	x				x							x					x	x									x	x				x	x				x				x					x
T33	x						x	x	x	x	x			x	x							x	x				x	x	x			x	x	x			x	x				x	x				x				x				x	
T34	x						x	x							x								x				x		x			x	x		x		x	x				x	x				x	x	x	x	x	x	x		x	
T35	x						x		x	x	x				x							x																x				x					x				x				x	
T36	x						x	x								x							x				x		x	x		x					x	x	x			x			x		x	x	x	x	x	x	x		x	
T37			x		x		x	x	x	x					x								x						x		x	x			x						x	x	x					x				x	x		x	
T38	x						x	x	x	x	x				x							x										x										x					x				x				x	
T39													x			x									x				x												x						x	x	x		x	x	x		x	
T40	x						x	x	x	x	x				x							x																				x					x				x				x	
T41															x							x																						x	x	x					x				x	
T42	x						x	x	x	x	x				x										x		x					x					x	x				x						x				x			x	
T43	x				x		x	x		x					x								x				x		x			x	x	x	x		x	x				x					x				x	x			x	x

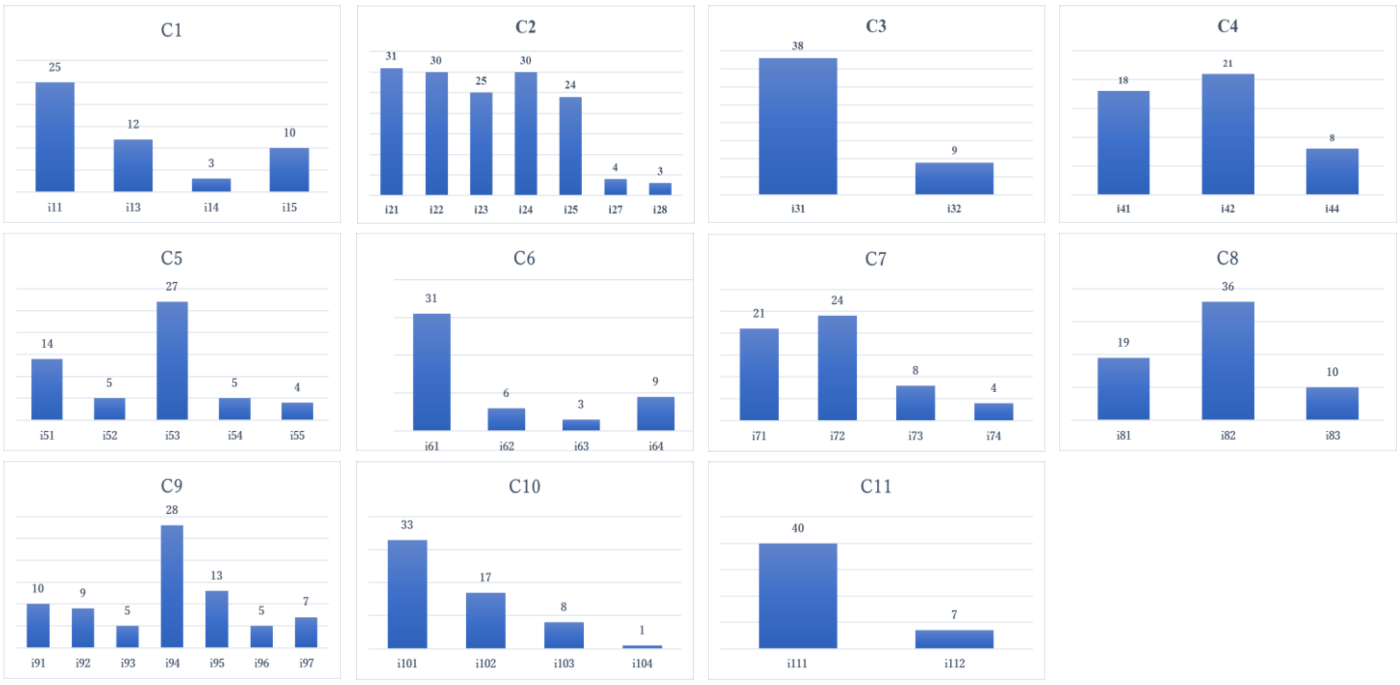

Fig. 1. Corresponding number of items by category.

VII. RELATED WORKS

Caitlin Kelleher and Randy Pausch surveyed programming learning tools, classified them with their original taxonomy, and created a table to explain tool attributes in 2005 [8]. Their survey and taxonomy were highly detailed, greatly contributing to resolving issues in this field. Due to advances in programming learning tools, a new survey is necessary to improve the taxonomy and incorporate new technology. In addition, the preceding survey targeted all kinds of programming education tools, which is extremely difficult today due to the greater diversity in tool types. Thus, our survey specialized in tools categorized as software developed for the purpose of education. This study successfully provides a taxonomy table for tools targeting beginners.

VIII. CONCLUSION

We surveyed 43 kinds of tools with an emphasis on visual language and software that works alone (except visual language) on PCs or other devices to create a taxonomy table for programming learning tools. The proposed table can evaluate and compare such tools. The experiment confirms that the classification and evaluation results are independent of the evaluator. Consequently, this taxonomy table helps users (learners and educators) select the appropriate tool based on their objective.

In the future, more than two people must verify the taxonomy table to confirm its reliability. Additionally, we will continue to investigate whether this taxonomy table helps users select the appropriate tool in actual situations.

REFERENCES

[1] Resnick, Mitchel, et al. "Scratch: programming for all." Communications of the ACM 52.11 (2009): 60-67.

[2] Cooper, Stephen, Wanda Dann, and Randy Pausch. "Alice: a 3-D tool for introductory programming concepts." Journal of Computing Sciences in Colleges. Vol. 15. No. 5. Consortium for Computing Sciences in Colleges, 2000.

[3] CodeCombat Inc., CodeCombat - Learn how to code by playing a game, at https://codecombat.com/, accessed on 6/20, 2016.

[4] Daniel Yaroslavski, Lightbot, at https://lightbot.com/, accessed on 6/20, 2016.

[5] Theodoropoulos, Anastasios, Angeliki Antoniou, and George Lepouras. "How do different cognitive styles affect learning programming? Insights from a game-based approach in Greek schools." ACM Transactions on Computing Education (TOCE) 17.1 (2016): 3.

[6] Saito, Daisuke, Hironori Washizaki, and Yoshiaki Fukazawa. "Analysis of the learning effects between text-based and visual-based beginner programming environments." Engineering Education (ICEED), 2016 IEEE 8th International Conference on. IEEE, 2016.

[7] Caitlin Kelleher and Randy Pausch, "Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers," ACM Computing Surveys, Vol. 37, No. 2, June 2005, pp. 83–137.

[8] Kai Petersen, Robert Feldt, Shahid Mujtaba, Michael Mattsson, "Systematic Mapping Studies in Software Engineering," Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE'08), pp.68-77, 2008.

[9] Katie Seaborn and Deborah I. Fels, "Gamification in theory and action: A survey," Int. J. Human-Computer Studies 74, 2015, pp. 14-31.

[10] Juho Hamari and Veikko Eranti, "Framework for Designing and Evaluating game Achievements," Authors & Digital Games Research Association DiGRA, 2011.

[11] Juul, Jesper. Half-real: Video games between real rules and fictional worlds. MIT press, 2011.

[12] J21 Corporation, CodeMonkey, at http://codemonkey.jp/, accessed on 6/20, 2016.