Education and Information Technologies (2022) 27:4887–4904
https://doi.org/10.1007/s10639-021-10811-w

Teacher observations of programming affordances for K-12 mathematics and technology

Niklas Humble[1]

Received: 13 September 2021 / Accepted: 11 November 2021 / Published online: 17 November 2021

Abstract

With future shortage of professionals with programming and computing skills, many countries have made programming part of kindergarten – grade 12 curriculum (K-12). A possible approach is to make programming part of an already existing subject. Sweden has chosen this approach and in 2017 programming was integrated in the subject content of K-12 mathematics and technology. Integrating programming is at the expense of extra workload on teachers. Teachers affected by these changes will face new challenges in their teaching and learning activities. The aim of the study is to examine K-12 teachers' use and perceived affordances of programming as a tool for teaching and learning activities in mathematics and technology. Data were collected through focus group discussions with three teacher teams in mathematics and technology from three K-12 schools in the mid Sweden region. 21 teachers participated in the study. Thematic analysis with a mixture of deductive and inductive coding were used to analyse the data. Theory of affordances was used to structure findings in themes of interests and answer the study's aim and research questions. Results show that the teachers use a variety of programming tools in their teaching and learning activities. The use of programming in mathematics and technology can be understood in five main perceived affordances: 1) Play, 2) Discovery, 3) Adaptation, 4) Control, and 5) Freedom; which relate to both student motivation and subject content. Teachers also perceive obstacles and opportunities in using programming, that relates to different programming tools' ability to support teaching and learning activities. The findings of this study can be drawn upon by teachers and other stakeholders in the integration of programming in K-12 education, and in the design of teaching and learning activities with programming.

Keywords Programming · Teaching and learning · K-12 education · Affordances · Teacher perspective

✉ Niklas Humble
niklas.humble@miun.se

[1] Department of Computer and System Science, Mid Sweden University, Akademigatan 1 – Building Q, 831 40 Östersund, Sweden

1 Introduction

During recent years, programming has been integrated in K-12 education (kindergarten to grade 12) in many countries (Nouri et al., 2020). This is in line with the expected need of professionals with competence in programming, due to an increased automation of jobs on the future labour market (Smit et al., 2020). Two possible approaches to introduce programming in K-12 education are to establish programming as its own subject or to integrate programming in an already existing curriculum (Nouri et al., 2020). Sweden is one of the countries that has chosen the latter approach and as of 2017 the Swedish government has accepted a revised school curriculum where programming has been integrated in the subjects of mathematics and technology (Heintz et al., 2017).

The path to integrate programming in an already existing school subject enables an interesting opportunity to use programming as a tool for teaching and learning activities in that subject. The use of programming as a tool for learning has a long history. A well-known example is that of Seymour Papert and the programming language Logo. Paperts (1993) idea, which he developed in the theory of *Constructionism*, is that a path to learning is to introduce the learners to microworlds, in which they can use and develop their skills and knowledge in a natural way. In the same way that you go to France to learn French, you should go to *Mathland* to learn mathematics (Papert, 1980). The programming language Logo was developed to support learners in this discovery and to move the role of the teacher to a co-learner, where the teacher and student learn and discover together (Papert, 1999).

Since Papert and Logo, there has been an expansion on the market of educational programming tools and languages. Educators and students are no longer limited to make their choice of programming tool only between different *textual programming tools*, such as Logo, Python, Java and C#. But they can also choose to use a programming tool where the code is already put together in blocks and all they need to do is snap them together, the so-called *block programming tool* (Papadakis & Kalogiannakis, 2019). It is also possible to not use a computer at all, so-called *unplugged programming* (Bell & Vahrenhold, 2018). One of the more well-known programming tools in the category of block programming tools is Scratch (Zhang & Nouri, 2019). Scratch expands on the idea of Logo but makes the programming easier with the graphical interface of blocks and, with the web-based solution, adds a big library of programming solutions for the users to be inspired by (Resnick et al., 2009).

With such a rich variety of different types of programming tools, one could be led to believe that the integration of programming in K-12 education would be an easy process. However, the integration is at the expense of putting extra workload on a group of teachers that often have no prior knowledge and experience in using programming as a tool in their subject (Rich et al., 2019; Szabo et al., 2019). As seen in previous research, the integration of new technology in educational context does not only provide educational affordances but also issues that need to be addressed (Andreas et al., 2010; Bower & Sturman, 2015). The teachers affected

by this change will have to face and solve a set of new challenges in their teaching and learning activities. How and to what do you use programming in K-12 mathematics and technology?

This study focuses on the teachers' perspective in integrating programming in an already existing school subject; and how they perceive programming as a tool for teaching and learning activities. The aim of the study is to examine K-12 teachers' use and perceived affordances of programming as a tool for teaching and learning activities in mathematics and technology. The study has been guided by the following research questions:

RQ1) How do K-12 teachers use programming as a tool for teaching and learning activities in mathematics and technology?
RQ2) Which are K-12 teachers' perceived affordances of programming as a tool for teaching and learning activities in mathematics and technology?

2 Theoretical framework

The theory, and concept, of *affordances* was coined by James J. Gibson and has been frequently used in research on technology in educational context (Bower & Sturman, 2015). According to Gibson (1977), *affordances* are what the environment (encompassing substances, objects, surfaces, places, medium, and other animals) provides for the animal (or the observer); they are not physical or phenomenal, they are based on information from both the environment and the observer. For example, a mailbox can be used to mail letters whether a person understand that it can be used for that or not; however, the affordance of *mailing letters* can only exist if there is a person to utilise it (Bower & Sturman, 2015).

Donald A. Norman is, besides Gibson, a well cited author in the field of educational technology (Bower & Sturman, 2015). Norman (1999) describes *affordances* as possible relationships that exists naturally between actors and objects. They are worldly properties, although they do not need to be known, desirable or visible (Norman, 1999). Norman also introduces the concept of *perceived affordances* which emphasis what affordances the user perceives, and which actions therefore are performed (Bower & Sturman, 2015).

According to Norman (1999) it is important to distinguish *affordances* from *perceived affordances* and *conventions*. The distinction is important in product design, especially with screen-based products, where the designer is generally more interested in what actions are perceived to be possible by the user. What the designer can control in screen-based interfaces are primarily *perceived affordances*, since the *affordances* of the computer system are already built in. When designing the graphical layout of a screen-based product, or discussing the use of *affordances*, it is *conventions* that the design and discussion rely on. (Norman, 1999).

Conventions, or *cultural constraints*, is part of four different classes of constraints that Norman (1990) introduces in *The Psychology of Everyday Things*. Constraints are important because they constitute the limitations to what actions are possible to the actor. *Cultural constraints* limit the possible actions to which are allowable

(based in the actor's cultural background). *Physical constraints* limit the possible actions to which are physically possible (based in the properties of the physical world). *Semantic constraints* limit the possible actions to which are meaningful (based in the actor's understanding of the situation). Lastly, *logical constraints* limit the possible actions to which are logically possible (based in the actor's reasoning). (Norman, 1990).

3 Related work

In a report from 2015, based on a survey with 21 Ministries of Education (20 European countries and Israel), it is reported that 16 countries had integrated programming at a local, regional, or national level; and additional 2 countries had plans to integrate programming (Balanskat & Engelhardt, 2015). In the same report, 13 countries integrate programming in a specific ICT/Technology course, while several countries integrate programming in an existing subject, mostly in mathematics (Balanskat & Engelhardt, 2015). In a literature review from 2019, the authors conclude that most initiatives of introducing programming in K-12 education are targeted towards middle and high school students, and a wide range of programming tools are used (Szabo et al., 2019).

In the UK, programming has been integrated in K-12 education through the subject of Computing, which is composed of information technology, digital literacy, and computer science (Royal Society, 2017). In a report by the Royal Society (2017), computing education across UK is labelled "patchy and fragile" and at risk of damaging future generations education and nation economy. A reason for this being shortage of computing teachers, and that most of those teaching Computing are unfamiliar with the subject and does not receive adequate support (Royal Society, 2017). Some of the recommendations given for addressing these issues are ensuring that all pupils receive Computing education and that teachers are supported in their teaching and professional development (Royal Society, 2017).

Finland and Sweden are two of the countries that have integrated programming in existing subjects (Pörn et al., 2021; Heintz et al., 2017). In a study with 91 grade 1–6 mathematics teachers in Finland, it is concluded that the teachers have a diverse view on programming, where some claim teaching material is lacking or insufficient, and that this may lead to education inequality (Pörn et al., 2021). The study further suggests that there is a need for educational efforts by those who work with K-12 teachers (for example teacher education and producers of educational material) to support teachers in making the connection between programming and mathematical content clear (Pörn et al., 2021).

A common distinction between different programming tools for educational use is whether they have a text-based (for example Python) or block-based (for example Scratch) interface (Lindberg et al., 2019). Block programming tools are usually considered to be easier for beginners because they use drag and drop with a mouse instead of typing (Lindberg et al., 2019). Studies show that students find block programming tools engaging (Adams, 2010), and that they minimize misconception about programming concepts compared to the use of textual programming tools

(Mladenović et al., 2020). While block programming tools may be engaging and easier to use, research has shown that the gains students make in block programming tools does not automatically transfer to professional textual programming tools (Weintrop & Wilensky, 2019). Textual programming tools could attract students with higher expectations of programming since the tools are used in professional settings (Garneli et al., 2015). Previous research has also investigated the use of a structured approach in teaching textual programming in K-12 education, the PRIMM approach (predict, run, investigate, modify, make), with positive results for both the students and the teachers (Sentance et al., 2019).

Another approach used in K-12 education, especially the lower grades, is unplugged programming (Otterborn et al., 2020). Their study show that K-12 teachers use unplugged programming to give students a more concrete experience of programming, which can later be drawn upon in the use of other programming tools (Otterborn et al., 2020). A study by Bell and Vahrenhold (2018) recommends that unplugged programming should not be used in isolation, but rather as a stepping-stone to help and motivate students for programming.

All three types of programming tools mentioned above (textual, block and unplugged) can be combined with tangible objects, so called *tangible programming tools*, to support learning. Research on an unplugged approach with a tangible robotic arm, showed that it could raise K-12 students' interest and attitude towards engineering (Miller et al., 2018). Inspired by block programming tools, Koushik et al. (2019) developed a tangible block-based programming tool (StoryBlocks) to support blind and visually impaired students learning basic programming.

According to Hammond (2010) there is a strong case for affordances when talking about information and communications technology (ICT) for teaching and learning. However, there must be a greater consensus on the concept if it is to be useful and Hammond (2010) suggest a definition of affordances as:

> "the perception of a possibility of action (in the broad sense of thought as well as physical activity) provided by properties of, in this case, the computer plus software. These possibilities are shaped by past experience and context, may be conceptually sophisticated and may need to be signposted by peers and teachers. […] Affordances provide both opportunities and constraints. Affordances are always relative to something and, in the context of ICT, relative to desirable goals or strategies for teaching and learning." (Hammond, 2010)

A common topic in previous research on affordances in educational context is design of technology to support learning, for example technology for computer-supported collaborated learning (Feyzi Behnagh & Yasrebi, 2020), supporting dyslexic readers (Antonenko et al., 2017), and 3-D virtual learning environments (Dalgarno & Lee, 2010).

There are also studies that investigate affordances in relation to programming education in K-12 settings. Block programming tools are often considered to be easier for novice programmers, compared to textual programming tools, because of the drag and drop approach to programming (Lindberg et al., 2019). This has been highlighted as an important affordance in block programming tools since it reduces the challenges of programming syntax (Sengupta et al., 2013). In a literature review,

Papavlasopoulou et al. (2017) examines the affordances of tangible programming and reflects on the practice and design of such tools. They conclude that the affordances of tangible programming tools are less ambiguous and therefore the objects are perceived as easier to predict and manipulate compared to virtual objects, which could potentially make programming more attractive for students at any age (Papavlasopoulou et al., 2017).

4 Method

The study was conducted as focus group discussions to uncover the range of experiences and perceptions by the participating teachers on the studied topic (Hennink, 2013). To create a non-threating and safe environment for discussion, the focus group discussions were conducted with teams of teachers that are custom to meeting each other regularly at their school workplace (Hennink, 2013). Further, the participating teachers are of similar power-positions within their organisation, since the focus groups consists of teachers that teach the same subjects at the same schools (Krueger & Casey, 2014).

4.1 Data collection

The focus group discussions were conducted during the spring and autumn semester of 2019 with three teacher teams at three different K-12 schools in the mid Sweden region. The teams consisted of teachers that have in common that they teach mathematics and/or technology in grade 7–9. The teacher teams were contacted through an e-mail invitation to all K-12 schools in two big municipalities in the mid Sweden region; and through an invitation post on a course forum for an introductory programming course for K-12 teachers in mathematics and technology at the [Mid Sweden University]. Three teacher teams, that met the criteria of teaching mathematics and/or technology in grade 7–9, answered the invitation and were selected to participate in the study.

21 teachers participated in the focus group discussions. The age of the teachers ranged from 33 to 61, with an average age of 45.8. 12 of the teachers are male and 9 are female. All of the teachers teach mathematics in grade 7–9, and 8 of them teach technology in grade 7–9. All participating teachers are familiar with the concept of programming since the introduction of programming in K-12 curriculum, but only 8 of them consider that they have a competence within programming prior the change in the curriculum. All focus group discussions were conducted in a semi-structured form and recorded. The average length of the recordings is about 48 min. The following questions were used as a guideline for discussion: How far have you come in the integration? How do you integrate programming in mathematics and technology? What programming tools do you use? What challenges and opportunities do you perceive? However, the participating teachers were encouraged by the moderator (the author) to add their own questions to discussion.

Focus group 1 consisted of 9 teachers, where all teach mathematics and 3 teach technology. 4 of the teachers consider themselves to have competence in programming prior the integration of programming in mathematics and technology. (Table 1) The focus group discussion was conducted at the teachers' workplace during one of their teacher team meetings in the spring semester of 2019. The focus group discussion was recorded and lasted for about 53 min.

Focus group 2 consisted of 6 teachers, where all teach mathematics, and none teach technology. 2 of the teachers consider themselves to have competence in programming prior the integration of programming in mathematics and technology. (Table 2) The focus group discussion was conducted at the teachers' workplace during one of their teacher team meetings in the autumn semester of 2019. The focus group discussion was recorded and lasted for about 51 min.

Focus group 3 consisted of 6 teachers, where all teach mathematics and 5 teach technology. 2 of the teachers consider themselves to have competence in programming prior the integration of programming in mathematics and technology. (Table 3) The focus group discussion was conducted at the teachers' workplace during one of their teacher team meetings in the autumn semester of 2019. The focus group discussion was recorded and lasted for about 40 min.

Table 1 Participating teachers in focus group 1

Participant	F1P1	F1P2	F1P3	F1P4	F1P5	F1P6	F1P7	F1P8	F1P9
Teaching mathematics	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Teaching technology		Yes				Yes	Yes		
Competence in programming				Yes		Yes		Yes	Yes

Table 2 Participating teachers in focus group 2

Participant	F2P1	F2P2	F2P3	F2P4	F2P5	F2P6
Teaching mathematics	Yes	Yes	Yes	Yes	Yes	Yes
Teaching technology						
Competence in programming				Yes	Yes	

Table 3 Participating teachers in focus group 3

Participant	F3P1	F3P2	F3P3	F3P4	F3P5	F3P6
Teaching mathematics	Yes	Yes	Yes	Yes	Yes	Yes
Teaching technology	Yes	Yes	Yes	Yes		Yes
Competence in programming		Yes				Yes

4.2 Data analysis

Collected data were analysed with thematic analysis to identify themes of interests in the material (Braun & Clarke, 2012; Bryman, 2016). A mixture of deductive and inductive approach was used in the process of analysis. First, deductive coding was used to identify themes of interests in the collected data, highlights were extracted and collected in a text document. This was based on the themes' relevance for the study's theoretical framework (Kiger & Varpio, 2020). The theoretical framework of *affordances* was used to identify and select themes of interests. Since the study's interest is in teacher use and perception, the concepts of *perceived affordances* and *constraints* (Norman, 1999; 1990) were used as the main concepts for identifying and selecting themes of interests in the data.

Second, inductive coding was used to regroup the identified themes of interest in the text document in categories based on their similarities in addressing perceived affordances and constraints. In this process, quotes in the collected data that represent the categories were chosen and translated to English. This work was derived from the collected data (Kiger & Varpio, 2020) and based in the themes' similarities. The categories were formulated as main perceived affordances that all the included themes related to. 5 categories, or main perceived affordances, where formulated in this process: 1) Perceived affordance of play, 2) Perceived affordance of discovery, 3) Perceived affordance of adaptation, 4) Perceived affordance of control, and 5) Perceived affordance of freedom. These categories are further used as sub-headings in the *Results and analysis*-section.

5 Results and analysis

The integration of programming in mathematics and technology is described to be in an early stage in all three focus groups. The participants describe that they are mostly planning and thinking about how to integrate programming in their subjects. In the discussions, many of the teachers discuss and gives examples of how they have started to implement programming with their students. Generally, the attitude towards programming is positive in all three focus groups. However, many of the teachers are critical towards how the integration has been planned by school leaders and other stakeholders. Common critical notions are not having enough time to learn and integrate programming, and that the instructions and guidelines for the integration are unclear.

5.1 Perceived affordance of play

In the focus group discussions, teachers gave examples of how they use programming as a tool for teaching and learning activities with their students. It was common in the discussions that the teachers reference fun and play when talking about the use of block programming tools in classroom practice. One such example is

where the block programming tool at Code.org is used to combine mathematical content of geometry with programming and Disney Princesses from the movie *Frost* (Quote 1).

> F1P4: In mathematics […] in Code.org when we were doing geometry last autumn, then there was this episode about the Disney Princesses in Frost, Anna and Elsa. And there were a lot of geometry. There were circles, angles, and a lot more. So, they [the students] got to run that. Everybody got to be a Disney Princess. It went really well. Even for the boys in 8th grade.
> Quote 1. Teacher describe the experience of using code.org in mathematics.

Unplugged programming was discussed in the focus groups but not used to a great extent by the teachers. It was mainly used as an introduction or warm-up for other programming tools. However, it is mentioned during discussions that unplugged programming is considered a fun and playful way of introducing programming to students. If it is used over a short period of time, otherwise there is a risk of it being perceived as repetitive and boring (Quote 2).

> F2P2: We have tested some [unplugged programming] […] only on easier instructions. […] I stood in the front and they [the students] would give me instructions to go and drink some water. […]
> Moderator: Is it something that you would consider using over a longer period of time? […]
> F2P2: As a warm-up in that case. To do unplugged programming over a longer period might be boring.
> Quote 2. Teacher describe the experience of using unplugged programming.

5.2 Perceived affordance of discovery

A popular programming tool among the teachers, especially in the subject of technology, is Micro:bit. It was mentioned during discussions that the way of combining block programming with tangible technology was considered an opportunity, since it provided students something to discover, while not being limited by programming syntax. According to a teacher, the experience of Micro:bit in classroom practice is that the students show a lot of engagement when using it. Which could be explained by that the students gets to try their way forward, what makes it sound, write or show something (Quote 3).

> F2P5: I have to say that I like Micro:bit. It was a lot of engagement in the class.
> Moderator: Why do you think that is?
> F2P5: I think that it stimulates their joy of discovery in some way. To get something to make a sound, or to make something write, or to make something show. They [the students] tried their way forward.
> Quote 3. Teacher describe the experience of using Micro:bit.

Many teachers expressed a strong resistance towards using textual programming tools in teaching and learning activities, because the tools were considered too difficult. This was mainly because the teachers themselves lacked the knowledge and experience in how to use the tools for teaching and learning activities in mathematics and technology. In the discussions, it is mentioned that other programming tools (block programming tools and unplugged programming) limits the discovery of programming because you do not have access to the full programming experience. Most teachers in the discussions stated that they would like to know more about textual programming tools to develop their own knowledge in programming (Quote 4).

F2P4: I would like to know it [textual programming].
Moderator: For your own sake or because you would like to use it in your teaching?
F2P4: Both. But also, because I feel that if I'm supposed to use this in a meaningful way in my teaching, I also need to know more myself. […]
F2P3: I agree.
Quote 4. Teachers discuss learning textual programming tools.

5.3 Perceived affordance of adaptation

During the discussions, it was mentioned that the adaptation in some block programming tools were perceived as an opportunity for teaching and learning activities. It was mainly the functionality to adapt the tools to be easier or provide more challenges to the students that were discussed as opportunities by the teachers. Code.org was mentioned as an example of how block programming could be adapted to provide more challenges for the students that want to advance faster (Quote 5).

F2P6: I would like to add an opportunity [to Code.org].
Moderator: Of course.
F2P6: That it also has the possibility for the students to, if you get this then you can advance fast and get more challenges. So, the degree of difficulty can always be increased.
Quote 5. Teacher describe the adaptation at Code.org.

Micro:bit was mentioned as yet another example of block programming that can be adapted to suit the needs of students that want to advance faster. Although most of the teachers, that used Micro:bit, stated that they used the block programming interface in Micro:bit, they perceived it as an opportunity that it is possible to shift the interface to textual programming with JavaScript. This can be used by the students with prior experience in programming, and as a point of comparison and discussion between student and teacher (Quote 6).

F1P9: […] you click on this, at least in Micro:bit, you can click and then see the text.
F1P6: You can shift it. There are like two modes.

F1P9: Yes, exactly. And then those that are a bit more skilled can go into that mode and see like: okey, it was that kind of statement. And then you can talk about that with those students.
Quote 6. Teachers describe the adaptation in Micro:bit.

A recurring obstacle with unplugged programming and textual programming tools in the discussions was their inability to be adapted. Unplugged programming is perceived to mainly be used for shorter introductions and warm-ups (Quote 2), since the lack of a computer limits its use for teaching and learning activities. Textual programming tools, on the other hand, are considered by many of the teachers to be too difficult for novice programmers. The concern is that the use of textual programming tools will have a negative impact on their students' motivation (Quote 7); and since the teachers do not see a possibility to adapt the tools to suit students with less prior experience in programming, they are hesitant towards using them.

F1P6: I keep coming back to this, we have to get everyone on board with this, and then block [programming tools] have the opportunity of making that possible. If you start with the other [textual programming tools] then you will lose. I'm sure that you will lose them [the students] straight away, half the class.
Moderator: If you go with textual [programming tools]?
F1P6: Yes.
F1P2: Absolutely.
F1P3: I also believe that. […] And to write textual [programming] is basically the same thing as what you get in a predefined block, but for the students it will feel like a huge difference.
Quote 7. Teachers discuss the obstacles and opportunities in using textual and block programming tools.

5.4 Perceived affordance of control

When comparing different programming tools in the discussions, it is mentioned that the level of control that the tool provides the teacher can be considered an opportunity for the teachers. Code.org is mentioned as an example that gives the teacher control over the students' use and progression, which some of the teachers appreciate. The teacher can create a course at Code.org for the students to take, and by this plan and monitor the students' activity (Quote 8). Code.org is compared to another tool that uses block programming, Scratch, and Code.org is perceived to be more user friendly by some of the teachers.

F2P1: On Code [Code.org] there is this kind of design that allows you to create a course for the students. And that feels more user friendly than Scratch. […] It's more strictly planned and you can follow the students' progression and see how they have solved the task.
Quote 8. Teacher compares Code.org and Scratch.

Textual programming tools are not perceived as programming tools that gives the teachers control over the students' use and progression. The lack of control over what the students do and can do in textual programming tools is mentioned as an obstacle in the discussions (Quote 7). However, it is also mentioned that the lack of control in textual programming tools might be of less importance in the future. According to some of the teachers, textual programming tools will be the appropriate tools to use in grade 7–9 mathematics and technology in a couple of years; since most students will already be familiarised with unplugged programming and block programming tools from lower grades by then. Textual programming, such as Python, will then be the next step of learning progression in grade 7–9, which have led some of the teachers to the conclusion that they might as well start using them now (Quote 9).

F3P2: They are running that [unplugged programming] in the lower grades and I think that is really good, when I hear my own children talk about school and so forth, and they are also introduced to block programming in 4th grade. So, I'm thinking when they come to us, they will hopefully already know this. The question then is, how much time should we put on establishing a way to work, if in 3 years we might be working with Python. Because they have already been through the other steps.
F3P6: And that's how it is, because I have read quite a lot of research about education and what they have tried and thought, and they have kept coming to that conclusion again and again. That is, block programming in middle school and younger and unplugged programming as an introduction. And then you use textual programming when you reach 7th, 8th, 9th grade and up. So that you will have a progression all the way.
Quote 9. Teachers discussing programming tools in relation to grade.

5.5 Perceived affordance of freedom

In the discussions, it is mentioned that an opportunity with using textual programming as a tool for teaching and learning activities, especially in mathematics, is that the tools allow freedom in the coding. According to the teachers, freedom in the programming activity is important for reaching a deeper understanding and for conducting mathematical calculations. Compared to block programming tools, it is mentioned in the discussions that textual programming tools allows deeper understanding and easier use of mathematical calculations because the tools are freer (Quote 10).

Moderator: So, you believe that it [textual programming tools] is a natural next step [after block programming tools]?
F1P7: Yes.
Others: Yes / Absolutely.

F1P4: Yes, because you do reach deeper with programming if you are coding more freely. Because, if it is block programming, then you are limited by the blocks you have. […]
F1P8: Yes, and some code can even be easier when writing with textual [programming]. […] When we calculate […] then it can be difficult with Scratch for example. […] Then it is much easier to just write [with textual programming] what you want.
Quote 10. Teachers discussing the opportunities with textual programming tools compared to block programming tools.

When discussing obstacles in using programming as a tool for teaching and learning activities in mathematics and technology, the limitations in block programming tools are mentioned. The concern is that block programming tools could prevent students from reaching a deeper understanding of programming (Quote 11). Because the code is hidden behind blocks and the students are limited to constructions that the tools support, this limits their ability to fully understand what could be done with programming. Further, the use of blocks was also mentioned in discussions to make block programming more difficult to use as a tool for teaching and learning activities in mathematics. When doing mathematical calculations, it was considered easier and faster to program these with textual programming tools than with block programming tools such as Scratch (Quote 10). Since block programming tools often requires the user to navigate through menus and finding and selecting the appropriate blocks to use.

Moderator: It sounds like you all are quite fond of the idea of block programming. Is there something negative with it? […]
F1P2: I think that you lose a deeper understanding of what it's about.
Quote 11. Teacher about losing deeper understanding with the use of block programming tools.

6 Discussion

An interesting finding in this study is that not all identified perceived affordances of using programming as a tool for teaching and learning activities in K-12 mathematics and technology are directly related to subject content. The perceived affordances of adaptation, control and freedom are often described in relation to subject content in the discussions. While the perceived affordances of play and discovery more often are described in relation to fun, motivation, and joy of discovery. This can be related to the concept of *microworlds* by Papert (1993). By engaging in programming with their students and finding it joyful and fun, they are learning and discovering together with them (Papert, 1999). Related to previous research on affordances in educational context, where a common topic is design to support different kinds of learning (Antonenko et al., 2017; Dalgarno & Lee, 2010; Feyzi Behnagh & Yasrebi, 2020), the perceived affordances identified through this study also highlights supporting learning. Whether it is supporting subject content (perceived affordances of

adaptation, control, and freedom) or supporting fun and motivation (perceived affordances of play and discovery). Which are both important aspects of learning.

Most teachers in the study where familiarised with several types of programming tools, most commonly textual programming tools and block programming tools. Their perception of these can be related to previous research. Block programming tools where generally considered to be easier and more engaging, as is also described by Mladenović et al. (2020), Lindberg et al. (2019), Sengupta et al. (2013), and Adams (2010). However, block programming tools where also described by the teachers to be limited and harder to use when engaging in more advanced subject content, especially in mathematics. In this regard, many of the teachers considered textual programming tools to be the better alternative to support subject content, and also support deeper knowledge in programming. This notion is not surprising since textual programming tools have an established use in professional development and could therefore attract students (and teachers) with higher expectations of programming (Garneli et al., 2015).

Some of the teachers in the study where also experienced with the use of tangible programming, especially in technology, and unplugged programming. Previous research on tangible programming in K-12 education has shown that it can be used to both raise students' interest and attitudes (Miller et al., 2018) and support special learning needs (Koushik et al., 2019). Similarly, the teachers that used tangible programming described that it engaged the students and stimulated their joy of discovery by trying their way forward with programming. Regarding unplugged programming, previous research describe that it can be used to give students a more concrete experience with programming, which can later be drawn upon in other programming tools (Otterborn et al., 2020). Similarly, the teachers that used unplugged programming used it as a fun warm-up to other programming tools, and not over longer periods of time since they believed that it would get boring. Many teachers also considered unplugged programming to be limited, which can be related to previous research that recommends that unplugged programming should be used in relation to other tools and not in isolation (Bell & Vahrenhold, 2018).

A possible solution to the challenge that some programming tools are potentially better at supporting student motivation while others are potentially better at supporting subject content, is to use different types of programming tools in a progression strategy. This was suggested by some of the teachers in the study, start with the easier and more engaging tools (that is, block programming and unplugged programming) and as the students' knowledge develops introduce the more professional textual programming tools. A potential problem with this strategy is that previous research has suggested that the gains students make in block programming tools does not automatically transfer to the use of textual programming tools (Weintrop & Wilensky, 2019). However, this is not to say that it cannot be done if the teachers support their students in the shift of programming tools and make the connection between the tools, and the subject content, clear. A method for introducing textual programming, such as the PRIMM approach (Sentance et al., 2019), can support the teachers in this work. Some of the teachers spoke highly of programming tools that support both block and textual programming, which can be used to further facilitate the transition from block programming to textual programming.

Although the teachers in this study were generally positive towards programming, there were some critical notions that can be related to previous work. Teachers stated that they did not receive adequate support by school leaders and other stakeholders in integrating programming. Limited time and unclear instructions and guidelines were mentioned as obstacles in the integration. In the study by Pörn et al. (2021), Finnish K-12 mathematics teachers express similar concerns and Pörn et al. (2021) suggest that there is a need for further educational efforts to support the teachers. It was also mentioned by some of the teachers in the focus groups that, although they were positive towards programming as a skill for their students, they would have preferred if it was implemented as its own subject. K-12 schools in the UK took a different approach to programming than what was done in, for example, Sweden and Finland, by integrating programming through the subject of Computing. However, the report by the Royal Society (2017) highlights similar challenges in UK schools and suggest similar efforts in supporting the teachers. This indicates that the crucial point for integration programming in K-12 education is the support that the teachers receive, and not whether it is integrated through its own subject or part of an already existing subject.

7 Conclusion

This study has examined K-12 teachers' use and perceived affordances of programming as a tool for teaching and learning activities in mathematics and technology. The study has shown that the teachers use programming in a variety of ways. Programming is used for calculations, tangible tinkering, and monitoring learning progression. But programming is also used for activities that go beyond being a tool for teaching and learning activities in mathematics and technology, such as facilitating fun, student engagement and motivation. Further, this study has shown that the teachers' use of programming in K-12 mathematics and technology can be understood in terms of perceived affordances, which highlight what they perceive to be meaningful and possible actions for supporting teaching and learning. These can be summarised in five main perceived affordances: 1) Play, 2) Discovery, 3) Adaptation, 4) Control, and 5) Freedom. Although, there is not a single programming tool that is perceived to supports all five. The conclusion of the study is that the K-12 teachers' use and perceived affordances of programming as a tool for teaching and learning activities enables them to support both student motivation and subject content in mathematics and technology. In that sense, the integration of programming in K-12 mathematics and technology is productive. Although, as pointed out in this study and previous research, there are important challenges with the integration that needs to be addressed. The results of this study can be used by teachers and other stakeholders in designing teaching and learning activities with programming in K-12 mathematics and technology; and in guiding the introduction of programming in K-12 education.

8 Limitations and future research

An interesting next step of research would be to investigate K-12 teachers' use and perceived affordances of programming as a tool for teaching and learning activities on a larger scale. Since this study was quite limited, both geographically and in the number of conducted focus group discussions, it would be interesting to investigate additional emerging themes.

Many of the participating teachers in the study have limited experience in programming prior the change in the curriculum. An interesting next step of research would be to conduct a follow-up study. What has happened since the last visit? Are the use and perceived affordances of programming as a tool for teaching and learning activities in K-12 mathematics and technology still the same? Or have new programming tools and themes emerged?

Authors' contributions Not applicable.

Funding Open access funding provided by Mid Sweden University.

Data availability Not publicly shared to protect study participants. Contact author.

Code availability Not applicable.

Declarations

Conflicts of interest The author declare that there is no conflict of interest.

References

Adams, J. C. (2010, March). Scratching middle schoolers' creative itch. In *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 356–360).

Andreas, K., Tsiatsos, T., Terzidou, T., & Pomportsis, A. (2010). Fostering collaborative learning in Second Life: Metaphors and affordances. *Computers & Education, 55*(2), 603–615.

Antonenko, P. D., Dawson, K., & Sahay, S. (2017). A framework for aligning needs, abilities and affordances to inform design and practice of educational technologies. *British Journal of Educational Technology, 48*(4), 916–927.

Balanskat, A. and Engelhardt, K. (2015), Computing Our Future: Computer Programming and coding, Priorities, School Curricula and Initiatives across Europe: European Schoolnet (EUN Partnership AIBSL), Brussels.

Bell, T., & Vahrenhold, J. (2018). CS unplugged—how is it used, and does it work?. In *Adventures between lower bounds and higher altitudes* (pp. 497–521). Springer, Cham.

Bower, M., & Sturman, D. (2015). What are the educational affordances of wearable technologies? *Computers & Education, 88*, 343–353.

Braun, V., & Clarke, V. (2012). Thematic analysis. In *APA Handbook of Research Methods in Psychology: Vol. 2. Research Designs*, H. Cooper (Editor-in-Chief) Copyright © 2012 by the American Psychological Association. All rights reserved.

Bryman, A. (2016). *Social research methods*. Oxford University Press.

Dalgarno, B., & Lee, M. J. (2010). What are the learning affordances of 3-D virtual environments? *British Journal of Educational Technology, 41*(1), 10–32.

Feyzi Behnagh, R., & Yasrebi, S. (2020). An examination of constructivist educational technologies: Key affordances and conditions. *British Journal of Educational Technology, 51*(6), 1907–1919.

Garneli, V., Giannakos, M. N., & Chorianopoulos, K. (2015, March). Computing education in K-12 schools: A review of the literature. In *2015 IEEE Global Engineering Education Conference (EDUCON)* (pp. 543–551). IEEE.

Gibson, J. J. (1977). The theory of affordances. *Hilldale, USA, 1*(2), 67–82.

Hammond, M. (2010). What is an affordance and can it help us understand the use of ICT in education? *Education and Information Technologies, 15*(3), 205–217.

Heintz, F., Mannila, L., Nordén, L. Å., Parnes, P., & Regnell, B. (2017, November). Introducing programming and digital competence in Swedish K-9 education. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (pp. 117–128). Springer, Cham.

Hennink, M. M. (2013). *Focus group discussions*. Oxford University Press.

Kiger, M. E., & Varpio, L. (2020). Thematic analysis of qualitative data: AMEE Guide No. 131. *Medical teacher*, *42*(8), 846–854.

Koushik, V., Guinness, D., & Kane, S. K. (2019, May). Storyblocks: A tangible programming game to create accessible audio stories. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (pp. 1–12).

Krueger, R. A., Casey, M. A. (2014). *Focus groups: A practical guide for applied research*. Sage publications.

Lindberg, R. S., Laine, T. H., & Haaranen, L. (2019). Gamifying programming education in K-12: A review of programming curricula in seven countries and programming games. *British Journal of Educational Technology, 50*(4), 1979–1995.

Miller, B., Kirn, A., Anderson, M., Major, J. C., Feil-Seifer, D., & Jurkiewicz, M. (2018, October). Unplugged robotics to increase K-12 students' engineering interest and attitudes. In *2018 IEEE Frontiers in Education Conference (FIE)* (pp. 1–5). IEEE.

Mladenović, M., Mladenović, S., & Žanko, Ž. (2020). Impact of used programming language for K-12 students' understanding of the loop concept. *International Journal of Technology Enhanced Learning, 12*(1), 79–98.

Norman, D. A. (1999). Affordance, conventions, and design. *interactions*, *6*(3), 38–43.

Norman, D. A. (1990). *The Psychology of Everyday Things*. Basic Books, New York, 1988. In paperback as *The Design of Everyday Things*. Doubleday, New York, 1990.

Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2020). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. *Education Inquiry, 11*(1), 1–17.

Otterborn, A., Schönborn, K. J., & Hultén, M. (2020). Investigating preschool educators' implementation of computer programming in their teaching practice. *Early Childhood Education Journal, 48*(3), 253–262.

Papadakis, S., & Kalogiannakis, M. (2019). Evaluating a course for teaching introductory programming with Scratch to pre-service kindergarten teachers. *International Journal of Technology Enhanced Learning, 11*(3), 231–246.

Papavlasopoulou, S., Giannakos, M. N., & Jaccheri, L. (2017, April). Reviewing the affordances of tangible programming languages: Implications for design and practice. In *2017 IEEE Global Engineering Education Conference (EDUCON)* (pp. 1811–1816). IEEE.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books.

Papert, S. (1993). *The children's machine: Rethinking school in the age of the computer*. Basic Books.

Papert, S. (1999). What is Logo? Who needs it. *Logo philosophy and implementation*, 4–16.

Pörn, R., Hemmi, K., & Kallio-Kujala, P. (2021). Inspiring or confusing–a study of Finnish 1–6 teachers' relation to teaching programming. *LUMAT: International Journal on Math, Science and Technology Education, 9*(1), 366–396.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, *52*(11), 60-67.

Rich, P. J., Browning, S. F., Perkins, M., Shoop, T., Yoshikawa, E., & Belikov, O. M. (2019). Coding in K-8: International trends in teaching elementary/primary computing. *TechTrends, 63*(3), 311–329.

Royal Society. (2017). *After the reboot: Computing education in UK schools*. Policy Report.

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies, 18*(2), 351–380.

Sentance, S., Waite, J., & Kallia, M. (2019). Teaching computer programming with PRIMM: A sociocultural perspective. *Computer Science Education, 29*(2–3), 136–176.

Smit, S., Tacke, T., Lund, S., Manyika, J., & Thiel, L. (2020). *The future of work in Europe*. McKinsey Global Institute.

Szabo, C., Sheard, J., Luxton-Reilly, A., Becker, B. A., & Ott, L. (2019, November). Fifteen years of introductory programming in schools: a global overview of K-12 initiatives. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research* (pp. 1–9).

Weintrop, D., & Wilensky, U. (2019). Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms. *Computers & Education, 142*, 103646.

Zhang, L., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education*, *141*, 103607.