# All Rosy in Scratch Lessons: No Bugs but Guts with Visual Programming

Pia Niemelä, Pervasive Computing, Tampere University of Technology

***Abstract*—This case study addresses motivational issues in the elementary computing that follows UK National Curriculum of Computing (UKNC) at one of the international schools in Asia. The study examines different motivations and their impact on learning outcomes. Started in Year 8, Scratch was used as a computing primer, followed by the Khan Academy's JavaScript, and Python basics. In order to study the learning process, surveys, interviews, and the analysis of the Scratch coursework were employed. Based on the results, Scratch provides a useful tool for scaffold programming basics and for fostering motivation in all student groups. The discontinuity point from visual to textual programming appears to be problematic: textual programming with JavaScript and Python seems to engage mathematically talented students who developed intrinsic motivation, and disengage several others, mainly because of felt incompetence. A few students with authentic interest areas, such as design, animation, or social media, engage inadequately after transition. In planning the syllabus, it is crucial to address motivational aspects as well.**

***Keywords*—K-12 computing syllabus; visual programming; Scratch; JavaScript; Python; SDT theory; intrinsic motivation**

## 1. Introduction

Our way of working and living is rapidly changing and school curriculum must adapt accordingly. Familiarizing students with computing basics should begin already in primary school. The secondary level then strengthens the gained skills further and enables differentiation. In learning, the attitude toward the subject and the perceived sense of self-efficacy are crucial in terms of good learning outcomes. In contrast, students' attitudes towards computing tend to become adverse in many early adapter countries, which reflects in the learning motivation. After including the subject in the elementary school curriculum, the enrollments of computing courses decreased alongside the attenuation of attitudes; examples include South Korea [1], the UK [2] and U.S. [3]. Although, U.S. has recently managed to switch the declining trend with determined actions, such as initiatives of Hour of Code, CS4All, and Hack Clubs [4]. In addition, an appealing tool selection, such as visual programming, seems to lure more students into CS classes, and to increase motivation and self-efficacy [5].

Negative attitudes are common not only among students, but also among teachers [6]. Rapid changes in society and new requirements induce resistance to change. Were the attitudes favorable, low computing competence would prevent from progressing [7]. Well-designed tools for novices lower the learning threshold by hiding complex syntax of programming languages. Brown (2013) lists the need for more substantial support for computing teachers at school context, and the shared vision with clear subject goals among school personnel [8]. In the United Kingdom, computing teachers have founded a Computing-At-School (CAS) community, where they can share their concerns and get advice [9].

As an active member and contributor of the CAS community, Crick (2011) noted that the increasing amount of disengaged students calls for more excitement and wider perception in the whole science-technology-engineering-math (STEM) domain [10]. Implementing visually rewarding projects in other STEM subjects, demonstrates applicability of programming skills also outside computing classes. The CAS guide for secondary school (2014) states that creating presentations, websites or videos, are a viable way to add excitement providing public access [11]. Demonstrating the utility of skills in concrete applications feeds motivation. [12]. Since being motivated and engaged correlates with good learning outcomes [13], attempts to construct a good intrinsic motivation are essential for education.

Visual programming circumvents many disadvantages of traditional textual programming languages, such as susceptibility to errors. Fewer errors and a robust environment increase productivity, which in turn adds the self-efficacy of students. However, after the students have tasted the convenience of visual programming, they might need some extraneous assistance in moving to textual programming [14], [15], [16]. The present study examines the development of motivation and focuses on transition from visual to textual programming.

### 1.1. Research questions

In this study, I focus on the motivational factors behind the differences of learning outcomes in computing. By analyzing the surveys, interviews and coursework, the study answers the following questions:

1) What motivation categories can be found?
2) What is the role of visual programming in engaging students?
3) What are the means to maintain motivation with textual programming?

First, I review the motivation theories and correlation between motivation and learning outcomes. The Research Methods section opens up the context of this study and application of mixed methods. The Results section describes

978-1-5090-5920-1/17/$31.00 ©2017 IEEE

the motivation categories based on the content analysis and the interview results. Analysis of Scratch coursework completes the Results. To conclude, the motivational aspects of learning are emphasized.

## 2. Development of Intrinsic Motivation

This study examines the development and manifestation of the motivation and early computing craftsmanship among school students. Differences in performance tend to grow during school years. The Self-Determination Theory (SDT, [17]) justifies this tendency by explaining the mechanisms of the development of the intrinsic motivation and partly by referring to the Intentional Learner Theory [18].

The innate psychological needs and structure of the person's psyche affect the development of intrinsic motivation, whereas external factors affect extrinsic motivation. External factors consist of environmental pressures, such as control and expectations, and after completing a task its rating and other anticipated feedback. Alternatively, the reward may also be more abstract or remote, for example, the appreciation of the community or a study place in the desired field. The Intentional Learning Theory [18] considers the long-term objective of gradual knowledge building more comprehensive than the intrinsic motivation, and denotes a "serious student" that sets subject-specific lifelong learning objectives. Regardless of motivation type, long-term goals sharpen the learning.

Extrinsic and intrinsic motivations are not mutually exclusive but complementary. Motivation type varies on each stage of learning. For example, in the beginning and at junction points, the extrinsic motivation is a trigger for the development of an intrinsic motivation so that a successful intrinsic learning process gradually takes over from the extrinsic one. Counter-intuitively, a number of studies [19], [20] claim that external rewards may even slow down the development of intrinsic motivation. For example, a student may start to think that a subject itself is not worth studying, if a teacher must reward the effort. In addition, rewards are seldom open-handedly distributed throughout the process. Was the motivation dependent on rewards, it would start to decline when rewards tail away.

According to the SDT theory, motivation is at its strongest in the overlap of three main components, competence, autonomy and relatedness. A motivated student feels more knowledgeable, being in control of his learning, and connected to other students. Competence is understood holistically to entail not only computing basics and specific tools, but being able to solve problems by decomposing them into smaller subproblems, identifying patterns, modeling systems and then gracefully implementing a desired artifact as a well-managed and timely delivered project [21].

Being able to set authentic goals supports autonomy, i.e., authenticity relates to autonomy. Authentic goals enhance engagement and empowerment. However, the goal must be achievable, so that a student either possesses or can acquire necessary skills to succeed on schedule.

Lastly, relatedness helps in transforming from extrinsic to intrinsic motivation. While intrinsic motivation is yet in its infancy, a student's relations to the significant others (i.e. friends, classmates, parents) may trigger and uphold motivation. Additionally, peers with the same level of competence scaffold the student. Relatedness is especially helpful in the beginning when the intrinsic motivation is still weak. For example, Haarala-Muhonen et al. (2011) recommended investing in the students' preparedness from the very beginning by using both student and teacher tutors [22]. Haatainen et al. (2013) decreased computer science dropouts by scaffolds, tailored for the first computer science course (CS1) at university [23]. The components of intrinsic motivation function in co-operation and influence each other in a spiral manner: competent students show more initiative and are eager to take on tasks, which again increases autonomy – autonomous learners take more responsibility for their own learning – which develops competency.

## 3. Research methods

The research questions are answered by surveying and interviewing Year 10 students of Hope International School in Cambodia in year 2016, and by analyzing their Scratch coursework. The students had followed National Curriculum computing syllabus [24] for two years by the time of conducting this research. They started with Scratch and continued with textual programming languages of JavaScript and Python. To get a grasp of the executed syllabus, the computing teacher was informally interviewed. His approach to teaching resembled genetic epistemology that necessitates progressions from concrete to abstract exercises (e.g. from Legos to Minecraft then to exploiting Python APIs that enable a programmatic construction of Minecraft artifacts) [25].

The UKNC syllabus defines the overarching learning targets, such as '*think creatively, innovatively, analytically, logically and critically*' and '*analyse problems in computational terms through practical problem solving, including designing, writing and debugging programs*' [26]. The syllabus does not dictate the use of any specific languages. Nevertheless, the selected programming languages hold an established position supported by Computing-at-School UK, as a course organizer and certifier. The author of this study did not influence teaching instructions, but carried out the interviews and examined the coursework retrospectively.

### 3.1. Qualitative analysis of surveys and interviews

The survey consisted of two parts: in the first part, the students answered open-ended questions regarding language preferences and the most useful computing skills. In the second part, they completed sentences to reveal the attitudes towards computing: "Coding is like...", "Scratch is...", "Scratch works best in...", "I like to code because...", "I do not like to code because...", and "Anybody needs ICT skills because..." The selection of survey informants was made by simply exploiting the group which was the easiest available,

i.e., the Extended Math class (N=16) that the author was teaching.

The content analysis of survey data aimed at identifying the underlying motivations. Similar arguments were grouped and the most descriptive words were selected as identifiers of each motivation category. The analysis led to the appropriate theoretical framework: the SDT theory and its intrinsic motivation part appeared to best explain the suggested categorization. Because of the content-rooted attachment of the theory, the analysis had a flavor of a grounded theory.

After these preparatory phases, the focus group (N=6) of the most skilled computing students were interviewed. The selection applied the snowball method: the teacher selected only the first student in the chain, and each student in turn named the next student of the opposite gender until a group of six was in size. During the interview, one of the students acted as a moderator and ensured everyone's contribution; no other people but the group were present. The questions were distributed beforehand, the first task being the review of initial motivation categorization. The review feedback shaped the final motivation categories.

The focus group interview was recorded and transcribed. The moderator proofread the transcription and completed the parts not caught by the audio typist. Consisting of the most prominent and intrinsically motivated programmers of Y10, the focus group and its interview could shed light on the genesis of intrinsic motivation. The most illustrative comments were quoted.

### 3.2. Quantitative analysis of Scratch coursework

Furthermore, the Scratch coursework provided means for triangulation in order to test the accuracy of intrinsic motivation predicting good performance. In this study, activity in Scratch is thought to reflect intrinsic motivation pursuant to the Free-Choice Paradigm [27]. The paradigm measures a student's intrinsic motivation based on his behavior when he considers not being observed. In this context, free-choice activities comprise extra projects, sent messages, comments, favorites, and followers/followees as a demonstration of intrinsic motivation. Similarly to the grading system with A-E that assumes the Gaussian distribution of human qualities, only one-fifth of the students are anticipated to have an 'A-level' intrinsic motivation. The activity score was reduced as intrinsic motivation simply by defining a threshold to result in one-fifth of the students. Activity that exceeded the threshold was regarded as an intrinsic motivation of value one; other users were marked as not intrinsically motivated, the value of zero. In comparison, the binomial data of intrinsic motivation correlated with Scratch scores illustrated with logistic regression. Logistic regression is apt for classifications, defining a student as intrinsically motivated or not.

In addition to the activity level, coursework is analyzed based on quality of the code. Initially, Dr.Scratch was the choice for scoring. Handily enough, the http://drscratch.org/ website requires only the project id as an input. The output comprises seven separate modules scored 0-3, resulting in the total of 21 points that divides into three levels: 'basic', 'developing', and 'mastery'. Nonetheless, Dr.Scratch did not provide a batch-processing mode necessitated by the number of projects. To execute analysis in larger quantities, an underlying Python plug-in, on which Dr.Scratch was built on, was exploited directly. This plug-in, called Hairball, claims to be lint-inspired [28]. Instead of a single indicator, it provides detailed information about the code. To reduce the amount, defining rubrics was indispensable, see Table 1.

Table 1. SCRATCH COURSEWORK RUBRICS

| Max score | Block | Definition |
|---|---|---|
| 3 | BlockCounts | Divided to basic and advanced:<br>Basic: Max score 2.<br>Advanced blocks consist of broadcast-message, clone and touch<br>Max score 1.<br>Totals 3 |
| 1 | DeadCode | $1 - \frac{lines \quad of \quad dead \quad code}{total \quad lines}$ |
| 1 | Variables | Max 0.5 from sprites as variables<br>Max 0.5 from other<br>Totals 1. |

Scoring should favor the utilization of desired and more advanced code structures, such as broadcast-receive message passing that enables concurrency combined with loops, e.g., a forever loop [29]. In concurrent programming, several agents are functional and may access the same resources simultaneously, which adds complexity. Agents and their interplay are central practices in agent-based programming, where Scratch categorizes in [30]. Scratch provides a clone operation for copying sprites. By cloning and setting rules for interactions, by sensing other agents (e.g. if touched), a student can implement simulations or games. Mastering advanced computing practices, such as concurrency and cloning, demonstrates progress in computational thinking skills of automation and algorithmic thinking, and prepares for multi-agent, event-driven programming with textual programming languages as well.

## 4. Results

The results are introduced chronologically, in the order of the survey (N=16), the focus group interview (N=6), and the Scratch coursework (N=54). In compliance with Scratch scores, other grades are reflected to widen the perspective. Based on the content analysis of the surveys and interviews, motivation categories are represented in the form of a diagram. From the categories, intrinsic motivation, a linchpin in developing expertise, is studied more closely by interviewing a focus group and analyzing Scratch coursework.

### 4.1. The survey and initial motivation categorization

The survey consisted of sixteen Year 10 students. The content analysis of the survey data divided motivations inherently into four categories characterized with following descriptions:

1) Intrinsically motivated students find coding rewarding per se, enjoy challenges, and problem solving
2) Intentionally motivated students are willing to invest in skills needed for further studies, jobs, and career. Good grades, positive feedback, and merits to show in a CV are highly valued
3) Design/art students pursue self-expression. The students aim at achieving something amusing or visually attractive that might be used as a design for industrial products, e.g. T-shirt, ads, and architecture
4) Internet/social media users are eager to connect, browse, and share ideas with their peers

The intrinsic motivation correlates most with the computing preference. Intrinsically motivated students enjoy programming, whereas students with other motivations need external triggers to ignite interest. A number of intrinsically motivated students are also keen on math. In sentence completions, these students often expressed their enthusiasm in compliance with highlighting the applicability of new skills, such as '*Computing...*':
Girl, intrinsic/intentional: ... *opens doors for me and enables me to make my way out of problems.*
Boy, intentional/intrinsic: ... *you can better understand what makes computers work and its logic.*
Boy, intrinsic: *I get the knowledge of how a computer works.*

In the interviews, disjoint categories of intrinsic and intentional motivations were clearly visible, but frequently manifested in the same students. Yet intrinsic motivation is more desirable from the subject's point of view, motivations are complementary: benefiting the skills excludes no genuine interest. However, a few intentionally (and not intrinsically) motivated students were capable of overlooking their dismay and frustration because of foreseen benefits, exemplified with the following comments that demonstrate this ambivalence:
Girl, intentional: *It is complicated to learn and has many different functions*, but on the other hand: *I know it will be an important skill to have in the future. ICT skills are important to have in your career.*
Girl, intentional: *Coding is very difficult and annoying when you are writing it but it is very relieving once you succeed. It is hard, you need to type a lot. But it is the trend nowadays and needed in society jobs, colleges, ..helps users think. People who use ICT skills have very developed thinking.*

The design and social media oriented students found their own territory of asserting themselves through Photoshop exercises, tuning images and animations, albeit sometimes these students indicate lower intrinsic motivation. This group with the negative computing attitudes gave the following statements:
Girl, design: *It is too hard and I do not understand computing*.
Boy, design: *It is a separate language that I do not understand. I find it to be boring, and it doesn't interest me.*

The teacher checked the identified categories and acknowledged them with '*Nail on the head!*'

### 4.2. A focus group interview and final categorization

Our focus group consisted of five intrinsically motivated and one design-oriented student. The group got the initial diagram for review; see Fig. 1. The globe with speech bubbles intended to illustrate the internet category, i.e. being connected and sharing ideas. However, the focus group struggled the most with this category. They associated internet with browsing, considered a no-brainer, which raised the question of a role of computing in it, '*Personally, I don't think number four (internet) applies very well if you know how to program, because if you know how to program, internet browsing shouldn't be a problem.*' The group was also slightly puzzled with 'design', '*design goes with the internet, cause the internet is in the end, and everything in the internet has been designed*'.

Due to no saturation, the categories of 'Design/art' and 'Internet/social media' do not cover all the variety of potential interest areas. Were the students keen on audio/video editing or robots, the survey topics might contain 'Audacity', 'iMovie' or 'Arduino', which fit poorly in the current categories. By combining design and internet, and labeling the new category more generically as 'authentic self-expression' enables tackling the whole range of interests; the revised diagram below.

Intrinsically motivated students found coding (or the final product) so rewarding that an effort was made even if extrinsic motivation factors (support, feedback, rewards) were minor or squeezing. A remarkable part of the motivation was the foreseen applicability of computing skills in further studies and career, which complies with the intentional motivation. In accordance, external motivation factors played a role in shaping the motivation such as family pressure, good grades and the pressure of being tested. The next excerpt illustrates this: *I probably wouldn't stick with computing if I wasn't being tested and pushed by the fact that I'm doing it for the school as well.*

The focus group did not believe that becoming intrinsically motivated happens by accident. A computing course, an enthusiastic peer, or a tutor that paves the way was anticipated to trigger it. Even if the interview did not pronounce any submissive reasons for developing an intrinsic motivation, personal qualities seem to play a significant

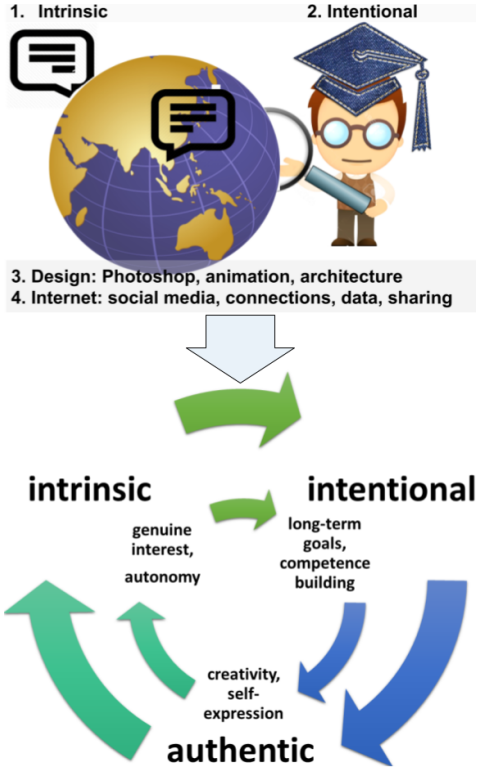


Figure 1. Motivation categories: the initial and revised versions

role. Math competence appears to correlate strongly and imply computing interest, which hypothesis is supported by grade correlations, the section 4.4, in particular between Additional Math and computing, as well as with intrinsic motivation indicators, such as activity.

In regard to motivation, the focus group was happy with moving from Scratch through JavaScript to Python, as addressed by a member *I would recommend using Scratch first, to show people how to think logically in an easy to learn environment. Then they should learn JS, as that's more like real coding, and can free people up from the limits of Scratch, while still being visual. Finally they should learn Python as that is more powerful than JS, but is easier to write without syntax errors.* Scratch prepares for textual programming without 'overwhelming' them. In contrast, the survey group preferred Scratch to JavaScript and Python '*definitely more exciting/fun than Python or Khan Academy*'.

### 4.3. Scratch activity predicts good performance

The quantitative part focuses on the effect of intrinsic motivation on learning outcomes by comparing Scratch activity and scores (N=54), where high Scratch activity is considered to reflect high intrinsic motivation. In addition to the number of projects, factors such as sent messages, comments, followers, and followees, increment the activity score. In determining intrinsically motivated students, the activity threshold was tuned to include approximately one fifth of the students. This threshold was raised three times higher with random MIT Scratchers revealing higher social activity of a typical user. Rather than for socializing, the Y10 students exploited Scratch for learning: they seldom contacted anyone else but their classmates or the teacher, and for any other reasons but advice. Compared with his students, the teacher was exceedingly active in giving them advice and preparing examples. In addition, he followed prominent Scratch contributors, thus scoring the highest in an activity-based evaluation.

One target of the Scratch coursework was to contribute and share projects publicly in the MIT database, where the visibility can be selected either public or private. Publicly shared projects expand the ever-increasing remix resources. Examining expert examples is an apt way of adapting new coding tricks. Demo selection provides opportunities of vocational growth for teachers as well. In addition to the Scratch community, the coursework provided cross-curricular contribution as well. It had agreed clients, who were, in essence, the STEM teachers of the school. They used projects to enliven their lessons, for instance, by showing simulations of laws of physics and chemistry.

The correlation between the intrinsic motivation and Scratch score was clear (see Figure 2). In the figure, *x*-axis

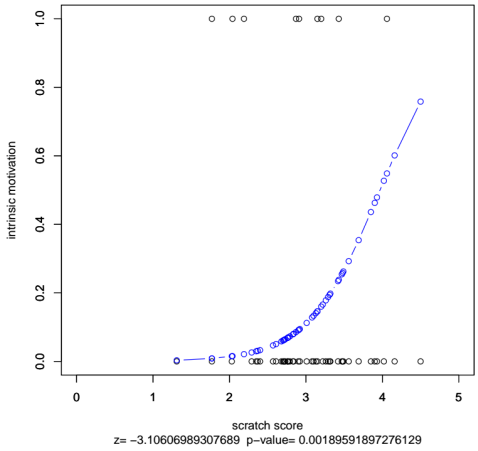


Figure 2. Intrinsic motivation (0,1), in *x*-axis the Scratch scores, p = 0.0019

shows Scratch scores within range 0-5. The distribution of scores was Gaussian, checked graphically with a histogram 4. The students' coursework was assessed by utilizing the Hairball plugin with the criteria defined in Table 1. The correlation between intrinsic motivation and Scratch scores was statistically significant (p=0.0019).

The Scratch projects in the MIT database can be employed as a reference point and a means to review the validity. In excess of projects (currently over 20 mill.) a random sample of N=1000 was considered adequate.

In contrast with the Y10 students, random data indicated no correlation between activity and Scratch score; see Figure 3. The typical S-curve straightened, and the significance of p-value = 0.1 was less than the limit of 5 %, thus the correlation was lost.

Unlike the examined Y10 students, a typical MIT Scratcher focuses more on his social connections. In ad-

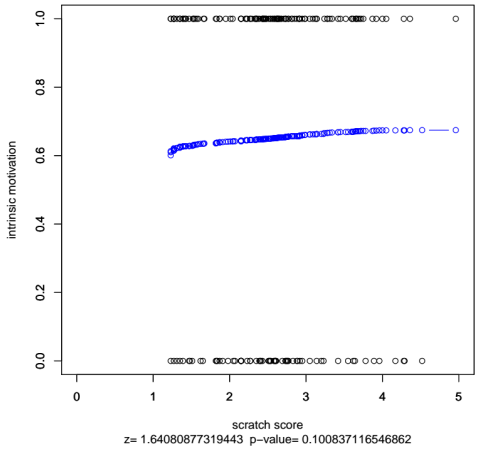


Figure 3. Random sample from MIT database revealed a different behavior, p = 0.1

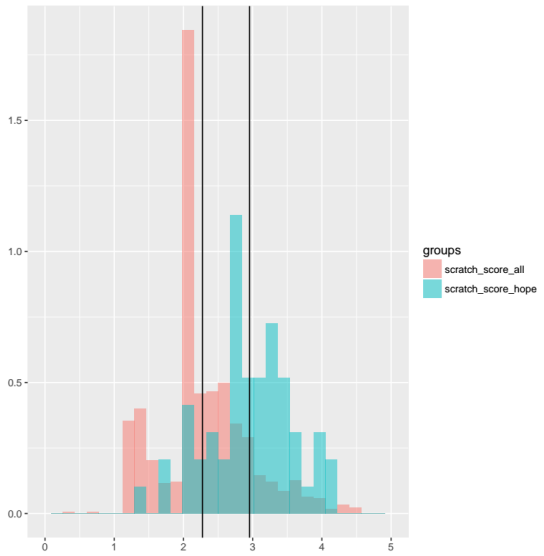


Figure 4. Histograms of Scratch scores; light blue bars describe the Hope School, whereas pink bars the random MIT sample

dition to a number of messages, comments and ratings, the followship of prominent contributors increases the activity score. The follower pattern of Scratch complies with preferential attachment theory [31], where followers distribute unevenly and cumulate in large quantities to few popular contributors. Including these experts, the Scratch user base is intensely heterogeneous, from kids to adults, many of whom just try out the tool. Evidently, a random Scratcher abandons a project easier without external pressure, that is, a teacher demanding the completion, getting a score, having a client, or sharing publicly. The amount of trash projects is high, thus selecting randomly results in a large amount of low scores. In this sample, the number of trash was remarkable.

In order to prove the significance of the quality difference, vertical lines illustrate the means of the samples. Moreover, a two-sample t-test determines the p-value. In the t-test, the null hypothesis states that the means are not significantly different, whereas the alternative hypothesis argues the opposite. The t-test gave p-value of 7.1E-11, i.e., indisputably significant. Fig. 4 reveals the bimodal nature of the random sample illustrated by the pink histogram, where the first peak resides between scores 1 and 2. This peak consists of trash projects that are not necessarily meant to be shared; the latter peak represents the decent projects. To improve comparability, the private projects should have been omitted, even if the web store allowed the download.

### 4.4. Other grades checked

For a validation of activity-predicted performance, the Scratch scores were contrasted with grades of other academic subjects. Letter grades of A-F were digitized in order to calculate the correlations: A+ equals to 4.33, A to 4, and F as “failed” equals to 0. Other grades linearly distribute in between. Being a member of the focus group was coded with the value of one, for non-members the value was 0. These substitutions enabled converting from nominal to interval data to calculate correlations.

The correlation between formal computing grades and global perspectives grades was the strongest (r=0.52), followed by geography(r=0.50). These two non-STEM subjects preceded correlations of science (r=0.495) and math (r=0.40). However, an essential aspect lost in calculations was the exclusiveness of Additional Math, selected only by five Y10 students, of whom four were in the focus group. The school provided three math levels in ascending order of complexity: Core, Extended, and Additional Math. To choose Additional Math, a student had to be an A-level student, thus the grades given in the Additional Math are not commensurate with the Core or Extended. Consequently, Additional Math participation was encoded as a dichotomy of on=1, off=0 to calculate the correlation, r=0.54, which was the strongest correlation with formal grades. After these correlation, activity, Scratch score, and focus group membership were checked. Activity and Scratch score correlated with computing grades poorly, activity r=0.05, Scratch score r=0.23; whereas with AddMath excellently, activity r=0.60, Scratch score r=0.56; and moderately with other math, r=0.31 and r=0.29. Being part of the focus group correlated with AddMath (r=0.66) the strongest, with other math strong as well (r=0.64), and quite strong with computing (r=0.54). Thus, correlations suggest a close linkage between intrinsic motivation and math as a subject.

## 5. Conclusion

**Which motivation categories can be found?** Categories include 1) intrinsic 2) intentional and 3) motivation based on authentic self-expression goals, often related to social media or design. Intrinsically motivated engage computing without special entertainments such as gamification or external rewards. Significantly, challenges that may hinder others have an opposite effect concerning intrinsically mo-

tivated students: hard as they struggle, the more motivated they become. As the teacher puts, '*Once they are engaged, you can get them to follow you through broken glass*'.

Intentional motivation comprises a flavor of strategic thinking and opportunism mixed in, pronounced by such excerpts in interviews as: '*An easy A for my paper*', '*Pushed by the tests*', '*Computing is applicable to the future*', and '*It's good for my CV*'. The group with authentic goals have special interest areas and distinct requisites for which they needed computing skills. In this sample, these interests varied from Photoshop, animation and architecture to social media and connectedness. By providing sharing, remixing, rating, and searching options, Scratch increases the anticipated community feel and adds social aspects to engage its users. These features induce a flavor of social media to Scratch. Besides socializing, however, the database of existing Scratch work offers means for learning from experts, when intentionally used in this manner. In addition to the Scratch on-line community, various APIs (such as api.scratch.mit.edu) proffer programmatic ways of handling Scratch data and tailoring new kind of applications.

Lack of any afore mentioned motivations appears as disengagement, which was not common among the examined students.

**What is the role of visual programming in engaging students?** The students appreciate Scratch as a tool. They value it high, because of no bugs and thus feeling more competent. The following snippet of the focus group interview demonstrates this: **Alpha**: *If you do Python or JavaScript, right, you have more chances of failing. When you have Scratch, you can have later that... super-multiple layers, you know what I mean? And when you finish it, it works, it is like (*a remarkable sigh*). Yeah, I'm the best! (pause). You know that feeling...?* **Drakvor**: *We get that feeling with Python, not with Scratch.* **Alpha**: *It is like if you have a wrong indent in Python everything goes wrong.*

As robust as it is, Scratch has its pitfalls. Meerbaum-Salant et al. (2011) discovered that Scratch might lead to bad programming habits, such as a bottom-up development and fine-grained programming [32]. The authors emphasized that Scratch requires appropriate teaching methods and materials to reach its full potential. Nonetheless, programming with Scratch appears to be a success story, whereas further steps with JavaScript and Python need adjustments to avoid decline in motivation.

**What are the means to maintain motivation with textual programming?**

In this study, the focus group (excluding the 'design' student) was eager to get grasp of more powerful and unconditional textual programming, whereas the motivationally heterogeneous survey group desired to keep Scratch, qualified as *'one of the fun things in your life'*, *'easy and visual'*, and *'the lego of coding'*. To boost motivation, the focus group advised to highlight the benefits, for instance, inspiring outcomes, **Alpha**: *If he (the teacher) shows what the final outcome, if we do it in that class, could end up like.. it increases the motivation.* **ActiveInFantasy**: *But you need show people achievable things.* **Alpha**: *Yeah, what they can achieve when they learn at that point..* **ActiveInFantasy**: *So if you start people on Python by saying 'Look at the Google Search Engine,' then they might be a bit overwhelmed.*

Applicability of the skills demonstrates in other school subjects as well, '*Our chemistry teacher had us make simulations of scientific experiments, which I think helped people see how their programming knowledge can be made useful in real life situations.*' In addition, higher employability demonstrates applicability, as the computing teacher emphasized, '*More and more in the world today, you can get jobs easier, if you have these sorts of skills.*'

Scratch may raise the threshold to transfer to textual programming: achievements may remain frustratingly modest due to frequent errors. To smooth the transfer, peer tutors are advantageous, and in addition to human help, tools and IDEs could provide extra scaffolding. Currently, active research focuses on tools bridging visual and textual programming with various angles of approach. In bridging, the mediated transfer between modalities of blocks and text can be either unidirectional or bidirectional. In a unidirectional tool, only the visual code, e.g. blocks, is editable (e.g. Blockly [33]); changes reflect automatically in the textual representation. A bidirectional tool modifies one representation based on the modification in another; examples include Droplet editor for Blockly [34], Pencil Code, and Code.org's App Lab [35].

In addition to textual code, illustrative visualizations scaffold gaining understanding of difficult concepts, such as lists and their indexing; difference of class and object; and control structures or the whole control flow of a program: Patch as a successor of Scratch provides such list visualization [36]. Frame-based coding offers an intermediate stepping stone in proceeding from visual to textual by structuring the code into blocks and by providing implementation hints [16], [37]. Greenfoot makes also such visualizations by elucidating e.g. the difference between an editable class and an instantiable objects [37]. Furthermore, visualizations need not restrict to code only, but block design can expand to cover electronic design as well, e.g. Scratch provides an Arduino expansion, Blockly goes with Picaxe that offers input/output simulations of circuits and flowcharts to visualize the control flow [38], and Microsoft MakeCode serves as a web-based programming platform similar to Scratch enabling the control of micro:bit components, which are embeddable and affordable microcontroller devices [39]. With the syntax guidance only, a student remains on the first steps of semiotic ladders, from where one should climb higher, to semantics and pragmatics. In climbing, the textual representation ought to be exploited to its full potential by bidirectionally connecting the block-based and textual representations and by explicitly transferring the gained programming experience, lest left untapped as a learning resource.

Climbing higher and striving for bigger and error-free systems necessitates design and testing. However, current Scratch supports poorly these attempts. In analysis phase, debugging tools should be as informative as possible: current state of variables and visualization of the control flow

enables finding errors effortlessly. Error messages should be descriptive and help in fixing. From new emerging transition tools, e.g. Patch addresses these issues by providing integrated tracing/debugging to assist algorithm development and implementation [36]. In preventing errors, test-driven development targets error-free code by mandating tests first, that students may first regard as work in vain. Hence, establishing the practice may require a disciplined approach from the teacher. The teacher should also focus on the most common misconceptions caused by block programming, such as '*loops are forever*' [5]. Misconceptions include indentation-related issues that seem to cause problems in moving to Python. In assessing the visual programming outcomes, automatic analysis could be exploited, e.g., Dr.Scratch to be passed with the level of 'developing' or 'mastery' before getting a grade. In addition, cross-debugging is worth trying.

**Future directions** Tailoring and differentiating the computing syllabus based on the skill level and authentic interest areas would provide meaningful learning goals for all student groups. The syllabus could have a separate and concise core part common to all computing students. Optional modules would comprise e.g. in-depth computing for intrinsically motivated students. Intentional learners would appreciate modules for study and future work skills including e.g. efficient data search and visualizations, whereas students with authentic interests would benefit from more autonomy and space to demonstrate their creativity by implementing assessable digital artifacts. With more fine-grained offerings, computing frustration was partly preventable.

Intrinsic motivation seems to be a remarkable asset in developing expertise in computing. The computing syllabus should take into account motivational aspects and attempt to foster intrinsic motivation in particular. The emerging trends (such as Internet of things, cloud services, coauthoring tools) expose the ubiquitous and pervasive nature of computing; in the future jobs will be more and more digitized. Everybody needs technological skills that are emerging as new citizenship skills, society should allow no digital dropouts. It is essential to take into account different motivations and engage students with authentic interests that may trail computing enthusiasm. This is important to note: clever engagement exploits students' authentic interests to increase their motivation.

Curriculum planners should pay more attention to motivational aspects also here in Finland, where the new curriculum including computing has been in effect since autumn 2016.

## Acknowledgments

I thank Hope International School, both the personnel and students, for opting me the interviews and Scratch coursework, and Pervasive Computing Department and my supervisors for their support of interdisciplinary research topics. In addition, special thanks to the Academy of Finland (grant number 303694; *Skills, education and the future of work*) for their financial support.

## References

[1] Jeongwon Choi and Sangjin An and Youngjun Lee, "Computing education in Korea — current issues and endeavors," *ACM Transactions on Computing Education (TOCE)*, vol. 15, no. 2, p. 8, 2015.

[2] Gillian M. Bain and Graham Wilson, "Convergent pathways in tertiary education. What makes our students succeed?" *ACM Inroads*, vol. 8, no. 2, pp. 37–40, 2017.

[3] Azad Ali and Charles Shubra, "Efforts to reverse the trend of enrollment decline in computer science programs," *The Journal of Issues in Informing Science and Information Technology*, vol. 7, pp. 209–225, 2010.

[4] Greg Thompson, "Coding Comes of Age: Coding Is Gradually Making Its Way from Club to Curriculum, Thanks Largely to the Nationwide Science, Technology, Engineering and Mathematics Phenomenon Embraced by So Many American Schools," *THE Journal (Technological Horizons In Education)*, vol. 44, no. 1, p. 28, 2017.

[5] Armoni, Michal and Meerbaum-Salant, Orni and Ben-Ari, Mordechai, "From Scratch to "real" programming," *ACM Transactions on Computing Education (TOCE)*, vol. 14, no. 4, p. 25, 2015.

[6] Balanskat, Anja and Blamire, Roger and Kefala, Stella, "The ICT impact report," *European Schoolnet*, vol. 1, pp. 1–71, 2006.

[7] Bingimlas, Khalid Abdullah, "Barriers to the successful integration of ICT in teaching and learning environments: A review of the literature," *Eurasia Journal of Mathematics, Science & Technology Education*, vol. 5, no. 3, pp. 235–245, 2009.

[8] Neil Christopher Charles Brown and Michael Kölling and Tom Crick and Simon Peyton Jones and Simon Humphreys and Sue Sentance, "Bringing computer science back into schools: lessons from the UK," in *Proceeding of the 44th ACM technical symposium on Computer science education*. ACM, 2013.

[9] S. Krpan, Divna and Mladenović and G. Zaharija, "Mediated Transfer from Visual to High-level Programming Language."

[10] Tom Crick and Sue Sentance, "Computing at school: stimulating computing education in the UK," in *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*. ACM, 2011, pp. 122–123.

[11] P. Kemp, Ed., *Computing in the national curriculum. A guide for secondary teachers*, 2014.

[12] Inkpen, Kori and Upitis, Rena and Klawe, Maria and Lawry, Joan and Anderson, Ann and Ndunda, Mutindi and Sedighian, Kamran and Leroux, Steve and Hsu, David, "" We Have Never-Forgetful Flowers In Our Garden": Girl's Responses To Electronic Games," *Journal of Computers in Mathematics and Science Teaching*, vol. 13, pp. 383–383, 1994.

[13] Maarten Vansteenkiste and Joke Simons and Willy Lens and Kennon M. Sheldon and Edward L. Deci, "Motivating learning, performance, and persistence: the synergistic effects of intrinsic goal contents and autonomy-supportive contexts," *Journal of personality and social psychology*, vol. 87, no. 2, p. 246, 2004.

[14] Weintrop, David and Wilensky, Uri, "Between a Block and a Typeface: Designing and Evaluating Hybrid Programming Environments," in *Proceedings of the 2017 Conference on Interaction Design and Children*. ACM, 2017, pp. 183–192.

[15] D. Weintrop, "Minding the gap between blocks-based and text-based programming," in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. ACM, 2015, pp. 720–720.

[16] Brown, Neil CC and Altadmri, Amjad and Kölling, Michael, "Frame-Based Editing: Combining the Best of Blocks and Text Programming," in *Learning and Teaching in Computing and Engineering (LaTICE), 2016 International Conference on*. IEEE, 2016, pp. 47–53.

[17] Richard M. Ryan and Edward L. Deci, "Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being," *American psychologist*, vol. 55, no. 1, p. 68, 2000.

[18] Carl Bereiter and Marlene Scardamalia, "Intentional learning as a goal of instruction," *Knowing, learning, and instruction: Essays in honor of Robert Glaser*, pp. 361–392, 1989.

[19] Martin V. Covington, "Rewards and intrinsic motivation," *Academic motivation of adolescents*, pp. 169–192, 2002.

[20] Edward L. Deci, "Effects of externally mediated rewards on intrinsic motivation," *Journal of personality and social psychology*, vol. 18, no. 1, p. 105, 1971.

[21] P. Niemelä, C. Di Flora, M. Helevirta, and V. Isomöttönen, "Educating future coders with a holistic ICT curriculum and new learning solutions," 2016.

[22] Anne Haarala-Muhonen and Mirja Ruohoniemi and Sari Lindblom-Ylänne, "Factors affecting the study pace of first-year law students: In search of study counselling tools," *Studies in Higher Education*, vol. 36, no. 8, pp. 911–922, 2011.

[23] Haatainen, Simo and Lakanen, Antti-Jussi and Isomöttönen, Ville and Lappalainen, Vesa, "A practice for providing additional support in CS1," in *Learning and Teaching in Computing and Engineering (LaTiCE), 2013*. IEEE, 2013, pp. 178–183.

[24] English Department for Education, "National Curriculum in England: Computing programmes of study," 2013.

[25] J. Piaget and E. Duckworth, "Genetic epistemology," *American Behavioral Scientist*, vol. 13, no. 3, pp. 459–480, 1970.

[26] Department for Education, "GCSE subject content for computer science," p. 6, 2015. [Online]. Available: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/397550/GCSE_subject_content_for_computer_science.pdf

[27] Deci, Edward L, "8: Ryan, RM (1985). Intrinsic motivation and self-determination in human behavior," *New York and London: Plenum*.

[28] Bryce Boe and Charlotte Hill and Michelle Len and Greg Dreschler and Phillip Conrad and Diana Franklin, "Hairball: Lint-inspired static analysis of Scratch projects," in *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*. ACM, 2013, pp. 215–220.

[29] John Maloney and Mitchel Resnick and Natalie Rusk and Brian Silverman and Evelyn Eastmond, "The scratch programming language and environment," *ACM Transactions on Computing Education (TOCE)*, vol. 10, no. 4, 2010.

[30] van Krevelen, D, "Intelligent agent modeling as serious game," *Agents for Games and Simulations*, pp. 221–236, 2009.

[31] D. J. Price, "Networks of scientific papers," *Science (New York, N.Y.)*, vol. 149, no. 3683, pp. 510–515, Jul 30 1965.

[32] Orni Meerbaum-Salant and Michal Armoni and Mordechai Ben-Ari, "Habits of programming in scratch," in *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*. ACM, 2011, pp. 168–172.

[33] Code.org, "Learn to Code Javascript and Blockly."

[34] Bau, David, "Droplet, a blocks-based editor for text code," *Journal of Computing Sciences in Colleges*, vol. 30, no. 6, pp. 138–144, 2015.

[35] Bau, David and Gray, Jeff and Kelleher, Caitlin and Sheldon, Josh and Turbak, Franklyn, "Learnable programming: blocks and beyond," *Communications of the ACM*, vol. 60, no. 6, pp. 72–80, 2017.

[36] Robinson, William, "From Scratch to Patch: Easing the Blocks-Text Transition," in *Proceedings of the 11th Workshop in Primary and Secondary Computing Education*. ACM, 2016, pp. 96–99.

[37] M. Kölling, N. C. Brown, and A. Altadmri, "Frame-based editing: Easing the transition from blocks to text-based programming," in *Proceedings of the Workshop in Primary and Secondary Computing Education*. ACM, 2015, pp. 29–38.

[38] Alimisis, D. and Moro, M. and Menegatti, E., *Educational Robotics in the Makers Era*, ser. Advances in Intelligent Systems and Computing. Springer International Publishing, 2017. [Online]. Available: https://books.google.fi/books?id=tr1ZDgAAQBAJ

[39] T. T. Ball, "Physical computing for everyone," in *Proceedings of the 39th International Conference on Software Engineering: Software Engineering and Education Track*. IEEE Press, 2017, pp. 3–3.