



# Evaluating young children's creative coding: rubric development and testing for ScratchJr projects

Apittha Unahalekhaka<sup>1</sup>  · Marina Umaschi Bers<sup>1</sup>

Received: 26 August 2021 / Accepted: 16 December 2021

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

## Abstract

Project-based assessment has been used to evaluate coding projects created by students for a long time. Nevertheless, there is a lack of rigorously tested project-based coding rubrics that are developmentally appropriate for early childhood. This study presented the development and testing of a coding rubric to evaluate children's creations with the popular ScratchJr app for early childhood, as well as results from field testing of the rubric. This paper first presents the ScratchJr Project Rubric development phases, and then a field test on 228 ScratchJr projects from 1st and 2nd grade students ( $n=87$ , aged 6–7 years old) across three time points. The results showed that the rubric demonstrates validity and reliability, and can measure changes in the project quality across time points. While the rubric was designed for researchers and teachers to evaluate ScratchJr projects, the design and conceptual framework is applicable to other programming languages for children that invite creative coding.

**Keywords** Early Childhood Education · Elementary Education · ScratchJr · Computer Science · Project Assessments

## 1 Introduction

In the last decade, an expanding number of coding tools and applications make it possible for young children to learn programming in creative ways by making their own projects (Sullivan & Bers, 2019). For example, the free ScratchJr app is one of the most popular developmentally appropriate programming languages for young children. As of October 2020, almost 60 million ScratchJr projects have

---

✉ Apittha Unahalekhaka  
apittha.u@tufts.edu

Marina Umaschi Bers  
marina.bers@tufts.edu

<sup>1</sup> Eliot-Pearson Department of Child Study and Human Development at Tufts University, Medford, MA, USA

been created (ScratchJr – DevTech Research Group, 2020). The animated projects that children create display both expressive design and coding skills (Refer Fig. 1), and thus it is important to have a reliable and valid tool to assess those projects.

The use of rubrics as tools for project-based assessments in computer science education is not new; however, there are shortcomings in the design that should be addressed. First, there needs to be more focus in testing rubrics for early childhood creations, as the focus has mostly been on middle school level and above (Basu, 2019; Denner et al., 2012; Grover, 2020a, b; Moreno-León & Robles, 2015; Wilson et al., 2013). Second, many rubrics do not capture enough aspects of the project that truly reflect children’s learning. For example, some rubrics only assess the coding and not the design or mostly evaluate the quantity and not the quality of the codes (Basu, 2019). Third, rubrics that were used to assess coding projects at only one time point were not able to reflect children’s coding capabilities (Brennan & Resnick, 2012; Salac & Franklin, 2020).

This study addressed the aforementioned gaps in project-based assessments literature for computer science education. Moving towards the younger grade levels, this study focused on the early childhood and early elementary students by designing and testing the ScratchJr Project Rubric. The ScratchJr Project Rubric is a project-based assessment that can be used to evaluate ScratchJr projects meaningfully and holistically. Particularly, this rubric assesses both the quantity and quality of the coding commands, as well as the project design components. Additionally, the rubric also looks for purposefulness, or the intentionality in adding different components to the projects. For example, the voice recording added to the project should have a clear relationship to the overall project theme to receive the highest score. Lastly, the rubric was tested across three time points in this study to make sure that it is capable of capturing subtle changes in project quality.

The paper describes the ScratchJr Project Rubric development phases and field testing. The project development phase involved the rubric’s design, iteration, and finalizing. Moreover, the field test phase involved the use of the ScratchJr Project Rubric to assess 228 projects created by first and second grade students. The study presents in this paper examined some of the basic psychometric properties



Fig. 1 Left: Painting Tool. Right: Coding Palette

of the ScratchJr Project Rubric including validity and reliability, as well as the rubric's subcategory score distribution. In particular, this study addressed the following research questions in the rubric development phases and field testing:

- RQ1: How can a coding education model that includes the acquisition of coding concepts, purposefulness, and design concepts be operationalized into a rubric for scoring young children's coding projects?
- RQ2: Does the coding project rubric described in RQ1 have acceptable psychometric properties, reasonable score distribution, and sensitivity across time?

## 2 Background

### 2.1 Early Childhood Coding with ScratchJr

Research shows that computational and digital competency skills can be acquired through coding and are crucial skills for the twenty-first century (Nouri et al., 2020). In addition, coding can support young children to acquire Computational Thinking (CT), which is a highly transferable skill for everyday life involving critical thinking, problem solving, and planning (Relkin et al., 2021).

With the advancement in technology, various tools have been developed to allow young children to code on tablet apps and with robots (Angeli & Valanides, 2020; Clarke-Midura et al., 2019; Sullivan & Bers, 2019). As illustrated in Fig. 1, ScratchJr engages children through its visual and audio features. It is designed so children can use their fingers to drag coding blocks, paint characters and background, record their voice, and take photos. Most importantly, ScratchJr teaches fundamental programming concepts such as Sequencing, Repeat, Conditionals, Parallelism, Coordination, and Number Parameters (Flannery et al., 2013; Rose et al., 2017; Strawhacker & Bers, 2019).

The ScratchJr programming language is block-based (Refer Fig. 2), which makes it developmentally appropriate for young children to understand the commands through symbols instead of words (Flannery et al., 2013). Compared to text-based programming languages, block-based languages are easier for children to grasp. Block-based languages remove the syntax, or a set of rules and structures in writing coding commands such as commas, apostrophes, or brackets at fixed positions (Grover, 2020a, b). ScratchJr has six coding block categories consisting of "Triggering", "Motion", "Looks", "Sounds", "Control", and "End", shown in Fig. 2. Each category has a different function and can be differentiated by color. From ScratchJr's functions, children can apply their creativity and expressivity to combine programming blocks and design backgrounds and characters for their animated stories.

### 2.2 Early Childhood Coding Assessments with ScratchJr

With an increasing demand for early childhood computer science education, different types of assessments have been used for young children's coding competency such as game-based (Pila et al., 2019), interview-based (Wang et al., 2014),





























Categories	Blocks
Triggering	 Start on Green Flag  Start on Tap  Start on Bump  Send Message  Start on Message
Motion	 Move Up  Move Down  Move Left  Move Right  Turn Right  Turn Left  Hop  Go Home
Looks	 Say  Grow  Shrink  Reset Size  Hide  Show
Sounds	 Pop  Play Recording Sound
Control	 Wait  Set Speed  Stop  Repeat
End	 End  Go to Page  Repeat Forever

Fig. 2 28 ScratchJr Coding Blocks by Categories

task-based (de Ruiter & Bers, 2021), observational (Fessakis et al., 2013). Two specific examples of ScratchJr programming language assessments are Solve-It and Coding Stages Assessment (CSA). Solve-Its are a set of open-ended tasks for children to recall ScratchJr coding blocks and create programming sequences according to the ScratchJr projects that were shown to them (Strawhacker & Bers, 2019). The objective of Solve-Its is to assess ScratchJr coding knowledge by asking children to circle correct coding blocks on the paper or order cut-out paper coding blocks into sequences.

Similarly, CSA also assesses children's ScratchJr programming language proficiency with additional focus on children's expressivity and purposefulness (de Ruiter & Bers, 2021). *Expressivity* is how well the children can turn their abstract ideas into creating coding sequences, while *purposefulness* is whether they can create according to their intentions. With the CSA, testers will ask children to explain their answers or create coding sequences on the ScratchJr app according to the given task. Then children's coding competencies will be categorized into one of five coding stages: "Emergent", "Coding and Decoding", "Fluency", "New Knowledge", and "Purposefulness".

Solve-Its and CSA are examples of assessments on ScratchJr programming language proficiency (de Ruiter & Bers, 2021; Strawhacker & Bers, 2019). However, unlike project-based assessments, they do not capture young children's naturalistic *ability to apply* their coding proficiency into creating their own open-ended and expressive coding projects. The next sections of this paper will introduce

project-based assessments and describe the process through which the ScratchJr Project Rubric was developed, and field tested.

### **2.3 Project-based Assessment: Coding Project Rubrics**

Project-based assessments have been used by researchers and educators to study how students demonstrate their skills in making authentic products (Cateté et al., 2016). There are different names for these types of assessments such as artifact, performance, or portfolio-based assessments. Project-based assessments serve to evaluate how children apply their knowledge and integrate interdisciplinary skills using rubrics (Chen & Martin, 2000). Particularly, they allow assessors to differentiate children's programming proficiency by evaluating their open-ended coding artifacts (Basu, 2019; Seiter & Foreman, 2013; Sherman & Martin, 2015). Furthermore, assessing a collection of projects over time can be beneficial to illustrate children's coding mastery progression and identify concepts that need additional support (Basu, 2019; Brennan & Resnick, 2012). It has been recognized that traditional coding assessments are sometimes too lengthy for the short attention span of young children (Relkin & Bers, 2019). Project-based assessments are naturalistic and can be done outside of class time. In another word, it provides an ease of administration under time constraints where children will not be taken away from their learning time.

Two studies presented how coding rubrics can give us insights on children's computer science learning. The first study tested a rubric with 160 middle schoolers' coding projects created in Scratch and App Inventor, which are coding apps with block-based programming languages (Basu, 2019). This study reported that students created projects with a wide range of complexity levels. Although most students only used simple coding concepts in projects, their project scores were moderately correlated with their Computer Science class grades. The second study showed that most of the coding projects created by first to sixth grade students used the basic "motion" and "appearance" coding blocks. However, only students in higher grades utilized the more advanced concepts of variable referencing and assigning (Seiter & Foreman, 2013).

However, there are studies that pointed out limitations of the project rubrics on assessing children's computational capabilities. A study found a very weak correlation between children's (ages 7–12) coding projects analysis (number of block usage) and coding concept assessments (Salac & Franklin, 2020). Using more loop blocks in projects does not mean that children understand how loops work. The researchers concluded that assessors should be cautious with what they can learn from the artifact analysis alone, as "Artifact analysis shows that a student built something—not that they understood something" (Salac & Franklin, 2020, p. 478). One approach that might help address this limitation is to assess not only one project but also a collection of projects over time to provide a richer view on how children can create differently (Brennan & Resnick, 2012). Furthermore, project rubrics should focus on the complexity of coding concepts rather than assessing just the quantity of blocks used in each concept (Basu, 2019).

The main advantages in using coding rubrics over the other forms of assessment are the rubric's naturalistic nature and its practicality for teachers to evaluate their students' work in their available time. However, most of the research using coding project assessments was conducted on late elementary and middle school students (Basu, 2019; Denner et al., 2012, 2012; Wilson et al., 2013). Thus, there is a lack of coding project rubrics to understand how young children create their coding projects. This study aimed to develop a ScratchJr Project Rubric for researchers and teachers that can assess the ScratchJr project creation with two main domains: *Coding Concepts* and *Project Design*.

### 3 Methodology: ScratchJr Project Rubric Development Phases

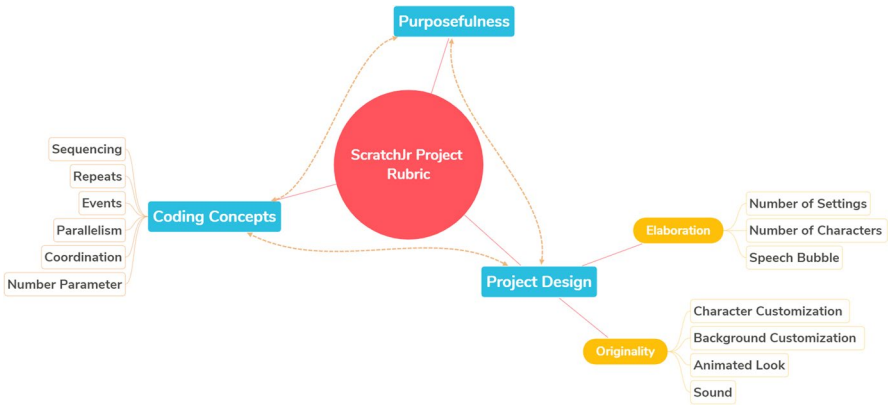
Table 1 shows three phases in the ScratchJr Project Rubric development, consisting of Rubric Design, Iteration, and Finalizing. This study used 287 testing ScratchJr projects provided by the DevTech Research Group, which were created at an unknown date by young children or teachers that joined the research group's activities.

#### 3.1 Rubric Design

The design of the ScratchJr Project Rubric's framework was inspired from previous studies with middle school students' coding projects (Brennan & Resnick, 2012; Denner et al., 2012; Wilson et al., 2013). For example, Denner et al. (2012) designed a rubric that has three main components including programming, code organization, and usability design. As ScratchJr was designed for younger ages, the ScratchJr Project Rubric has two main domains: Coding Concepts and Project Design as shown in Fig. 3. Both domains are evaluated in terms of the purpose they capture. That is, if a project has a random sequence of blocks and no coherent design, the creator sense of purposefulness will be indicated as low, resulting in a lower project score. While code organization and efficiency are often emphasized among older students (Denner et al., 2012), we think that they are not as important in early childhood as code correctness and purposefulness.

**Table 1** Timeline of the ScratchJr Project Rubric Development

Phases	Process Details
Phase 1: Rubric Design	Literature Review Subject-Matter Expert Feedback on the Rubric Scoring of Testing ScratchJr Projects
Phase 2: Iteration	Test and modify rubric with 245 projects • Inter-rater Reliability Testing (4 raters)
Phase 3: Finalizing	Test finalized rubric with 42 projects • Inter-rater Reliability Testing (3 raters)



**Fig. 3** ScratchJr Project Rubric’s two main criteria, Coding Concepts and Project Design, and their subcategories, which incorporated purposefulness

### 3.1.1 Literature Review: Coding Concepts

The ScratchJr Project Rubric’s Coding Concepts subcategories were primarily guided by the programming concepts gained from various studies that utilized ScratchJr (Flannery et al., 2013; Rose et al., 2017; Strawhacker & Bers, 2019; Strawhacker et al., 2018). In addition, when relevant, we adapted the coding concepts from coding rubrics for older children’s coding projects using Scratch (Basu, 2019; Denner et al., 2012; Wangenheim et al., 2018; Wilson et al., 2013). Consequently, the ScratchJr Project Rubric has six Coding Concepts subcategories consisting of “Sequencing”, “Repeats”, “Events”, “Parallelism”, and “Number Parameter” shown in Table 2. A project will receive a higher score in each subcategory if the coding sequence is syntactically correct and more conceptually advanced.

**Table 2** Coding Concept Subcategories Definition

Coding Concepts Subcategories	Definition
Sequencing	Able to form unique and functional coding blocks for the command to be executed as expected (number of coding blocks used)
Repeats	Using a command to make an action repeat multiple times
Events	Sending a command for an action to happen
Parallelism	Two or more coding sequences playing simultaneously to command multiple actions
Coordination	Controlling different characters to coordinate in an action
Number Parameter	Inserting the number of times that a command should be executed

**Table 3** Project Design Subcategories Definition

Project Design Subcategories	Definition
Character Customization	Changing the aesthetic of the character, which is defined as a character with more than one coding block
Background Customization	Changing the aesthetic of the background
Animated Look	Changing the animated appearance of the character with look blocks
Sound	Using pop block or voice recording tool
Number of Settings	Adding project pages with settings up to 4 pages
Number of Characters	Adding a number of characters with functional coding sequences
Speech Bubble	Entering words, phrases, or sentences with the speech block

### 3.1.2 Literature Review: Project Design

O'Quinn and Besemer (1989) defined three necessary elements for a creative product: originality, elaboration, and purposefulness. These informed the ScratchJr Project Rubric's Project Design criterion shown in Table 3. For originality, the rubric evaluates "Character Customization", "Background Customization", "Animated Look", and "Sound". For elaboration, there are "Number of Characters", "Number of Settings", and "Speech Bubble". The purposefulness component is incorporated across the Project Design criterion; for example, sound recording that aligns with the project would receive a higher score than an accidental sound recording that only plays a classroom background noise.

### 3.1.3 Subject-Matter Expert

After literature review was conducted on the existing coding rubrics for older children, the first draft of the rubric was created then reviewed by four experts in the field of early childhood technology and one ScratchJr developer. The experts gave suggestions on how to make the rubric scoring more developmentally appropriate for young children. For example, a nested loop should earn a higher score than a repeat loop. Further, to ensure that the rubric will capture a broad range of capabilities, the rubric scoring was altered from the score of 1–3 to 1–4.

### 3.1.4 Scoring

The score of each subcategory can range from 1–4 points according to the concept mastery level. The project must demonstrate the most challenging concept to obtain the highest score in each subcategory (Refer to Appendix 1). There are six categories for Coding Concepts, so a maximum score for this criterion is 24 points. Additionally, there are seven categories for project design, but the maximum score is limited to only 16 points. The goal was to give a higher weight



(60%) to coding concepts than to aesthetical design (40%) elements. Therefore, 40 points is the maximum score that each ScratchJr project can reach. The 60–40 weighted ratio in the ScratchJr Project Rubric was used according to scoring of the KIBO Project Rubric. The KIBO Project Rubric was developed and tested for evaluating early childhood robotic projects. The rubric also demonstrates validity and reliability (Govind & Bers, 2021).

### 3.2 Iteration Phase: Summary of Rubric Modifications

There were multiple rounds of rubric revisions due the diversity of ScratchJr projects and several raters were involved in the process. One of the most challenging aspects in the revision process was the design of the rubric to be broad enough to cover a wide variety of projects, but specific enough to have reliability across different raters. For example, a character can possibly have no code, non-functional codes, one functional code, or multiple functional codes. ScratchJr projects can be highly abstract especially when there is no word or narration to tell what the children intended to create. This observation aligned with the findings from Wright’s study (2010) that young children’s storytelling can be open-ended and ambiguous to outside viewers.

After seeing that the blocks’ functionalities were frequently misused by children, we revised the scoring criteria of many subcategories to capture purposeful creation. An example of two sequences that have the same command, *move right* then *move up*, is shown in Fig. 4. However, the top sequence has four blocks that did not make any changes to the program, and we identify these blocks as being misused. Specifically, the second block on the top sequence is an appear block, which will not make changes to the character, as this character has always been visible. Furthermore, the third block and fifth block are regular speed and regular size blocks.

To capture purposeful creations, raters only assessed blocks that make actions that can be seen or heard. Therefore, under the “Sequencing” concept, raters would count both sequences in Fig. 4 to have the same number of blocks (excluding unused blocks) and rate the same score. Furthermore, the rubric defines a *character* as having at least one coding block, while characters with no coding block are considered as parts of the background as we often saw that they were used as background



Fig. 4 Comparison between two different sequences with the same actions

decoration. For example, a table in the bedroom, a sun and stars in the sky, and static friends in the classroom.

### 3.3 Iteration and Finalizing Phases: Inter-Rater Reliability

In the Iteration phase, the first author of this paper and three raters each assessed 245 projects, 176 practice scoring projects and 69 final scoring projects (with a finalized rubric). All criteria of the final scoring projects had a sufficient inter-rater reliability score (Krippendorff's  $\alpha > 0.80$ ), the averaged inter-rater reliability score was 0.89, ranging from 0.82–1.00.

In the Finalizing phase, two new raters did an inter-rater reliability testing with the first author of this paper. Although the main ideas of the finalized rubric remained intact, the wording of the rubric got revised for clarity. Then the new raters spent approximately 30 hours, each grading 100 practice projects as part of their training. Raters took up to two minutes to score each project depending on its complexity and length. On the final 42 new projects that have not been graded in the Iteration phase, the overall Krippendorff's alpha was 0.86, ranging from 0.71- 0.96. The graders met and resolved the misaligned ratings mainly due to human error in missing the small details on the rubric grading guidelines.

## 4 Methodology: Field Test

### 4.1 Participants

To field test the rubric, we worked with projects created by first graders (3 classes,  $n=48$ ) and second graders (2 classes,  $n=39$ ) from a public school in Minnesota, USA. This school was one of the 40 schools that we reached out to across the US from our research group e-list. Five classroom teachers were interested and participated with their students in this study. Consequently, students (aged 6–7) with consent from all five classrooms participated. While the racial breakdown of the students was unreported, 47% of the students were girls and 53% were boys. Participants were enrolled in a ScratchJr Coding as Another Language curriculum (Bers, 2019) implementation, which spanned from February 2021-April 2021. Due to students' absence from the COVID-19 pandemic and other personal issues, there were a total of 73 projects at Time 1, 79 projects at Time 2, and 76 projects at Time 3.

### 4.2 Procedure

This study was part of the Coding as Another Language (CAL) project, which implements a coding curriculum for K-2 students that integrates literacy with ScratchJr in public schools. A research-based CAL curriculum focuses on the role of artificial and natural languages for young children to express their ideas by combining programming language and literacy. Some activities in the CAL curriculum are unplugged, while most of the activities use the ScratchJr app as a programming platform (*Coding*

as *Another Language*, 2021). The CAL project in this study was implemented for approximately three months during a regular 45-min academic lesson that met twice a week. The school provided each student with a tablet (1:1 ratio) to code on ScratchJr during these class periods, which normally would have been scheduled for math or literacy.

From the 24 lessons included in the CAL curriculum, the research team asked the teachers to collect and send individual student's ScratchJr projects at three-time-points aligning with Brennan and Resnick's (2012) report that project assessments can be more effective when evaluated across times. At Time 1, students were taught with basic motion blocks to make characters move such as move forward and jump. At Time 2, students were taught all the project design features such as voice recording and creating multiple pages. At Time 3, students were taught all the advanced coding concepts in ScratchJr such as "Coordination", "Events", and "Parallelism". The CAL curriculum provided teachers with a short open-ended prompt for each student to create a ScratchJr project at each time point. For example, first graders were asked to create projects for their characters to do a "hokey-pokey" dance at Time 1, act in their classroom story at Time 2, and tell a story from "Where the Wild Things Are" book at Time 3 (Refer Fig. 6). "Where The Wild Things Are" is a children's story book written by Maurice Sendak, about a boy named Max that had a dream that he sailed to a monster land and had a dance party with them.

### 4.3 Data Analysis

The objective of the field test phase for the second research question of this study was to test the newly developed ScratchJr Project Rubric for some of its basic psychometric properties, score distribution and sensitivity across time. The first author of the paper used the ScratchJr Project Rubric to rate 228 projects across three different time points in the curriculum on all 13 subcategories (Refer Fig. 3). The projects that display more advanced coding concepts will receive higher scores under this criterion (Refer Appendix 1). Moreover, the projects that display more details and variety in their customization will receive higher scores in the project design criterion. As the activity prompt did not enforce children to use all types of coding blocks or design features on their projects, they may or may not end up having all the subcategories of the rubric. For this study, teachers were not involved with the project rating process; however, their involvement will be important to learn about the practicality of using the rubric in classrooms.

#### 4.3.1 Validity and Reliability Testing

To test for the rubric's criterion validity, a partial correlation analysis was conducted between the ScratchJr Project Rubric and the Coding Stages Assessment (CSA), controlling for students' grade level and gender. The CSA is an assessment to measure children's expressivity and technical level on the ScratchJr programming language (de Ruiter & Bers, 2021). The CSA demonstrated criterion validity in measuring Computational Thinking, and an excellent level of internal consistency, Guttman's  $\lambda_6 = 0.94$ .

In this study, all participants took the CSA test before and after the curriculum, but only the post-CSA weighted scores were compared to the final project scores at Time 3 ( $n=76$ ). The CSA covers all Coding Concepts and most Project Design elements in the ScratchJr Project Rubric. Although the rubric and the CSA do not measure the exact same learning areas (ability to create ScratchJr projects versus ScratchJr programming language competency), we expected both assessments to at least be mildly correlated. We did not expect a high correlation due to the open-ended nature of the ScratchJr Project Rubric, unlike the CSA that is task-based.

However, there should be some correlation, as children's measured expressivity and technical levels by the CSA are likely to be reflected in their ScratchJr projects.

Furthermore, we used Krippendorff's alpha for the inter-rater reliability calculation since the rubric development phase due to its ability to measure scores with more than two raters and capture missing values (Krippendorff, 2011). For field testing, the first author of the paper assessed all projects with an additional rater to assess 35% of the randomly chosen projects from each classroom at each timepoint.

### 4.3.2 Subcategories Frequencies and Means

This study calculated frequency and mean score of each subcategory only at Time 3 ( $n=76$ ), which was when students learned all the ScratchJr concepts. The purpose for this analysis was to determine whether children use a certain concept often, and whether this finding aligns with prior literature on early childhood coding. Moreover, it is important to note that children may not use all subcategories in their projects, and this study treated the unused category scores as missing values.

### 4.3.3 Sensitivity To Change

Evaluating children's coding projects over time may give meaningful insights on children's learning (Brennan & Resnick, 2012). Therefore, the rubric's sensitivity was examined by a longitudinal regression across three timepoints. Particularly, this analysis investigated whether time was a significant predictor of the total project scores (outcome) accounting for grade level and gender. Cases with no project scores at a particular time point were assumed to be missing at random due to students' absence mainly from the COVID-19 pandemic. Consequently, 33 scores were treated as missing and got omitted from the model, resulted in 228 project scores across three timepoints. This study's final longitudinal model includes time as the level-1 predictor; grade level (0=Grade 1, 1=Grade 2) and gender (0=Female, 1=Male) as the level-2 predictors. Furthermore, this model only allows the intercept (project score at Time 1) to vary, while the slope of time is fixed across students as shown in Eq. 1.

$$\begin{aligned}
 Y_{it} &= \pi_{0i} + \pi_{1i}Time_{it} + e_{it} \\
 \pi_{0i} &= \beta_{00} + \beta_{01}(Grade_i) + \beta_{02}(Gender_i) + r_{0i} \text{ Intercept} \\
 \pi_{1i} &= \beta_{10} + \beta_{11}(Grade_i) + \beta_{12}(Gender_i) \text{ Slope : Time}
 \end{aligned} \tag{1}$$

The final model equation with the total project score as the outcome; time as the level-1 predictor; and grade level and gender as the level-2 predictors.

## 5 Results

### 5.1 Descriptive Statistics

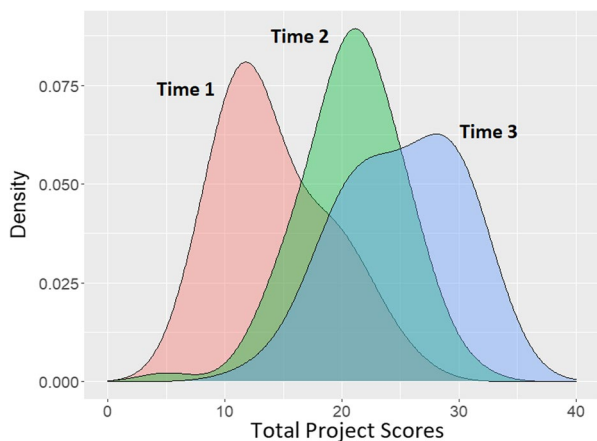
Figure 5 illustrates the total project score density plots across three-time points. They were approximately normally distributed with skewness and kurtosis values within absolute 2, ranging from -0.29 to 0.62 and -0.80 to 1.27, respectively. There were no outliers at Time 1 and Time 3. However, there was one outlier at the lower end of Time 2, which was far below the expectation from a 95% confidence interval for a sample size of  $n=79$ . Furthermore, only 57 students (65.51%) had all three projects, while 27 students (31.03%) had two projects, and three students had one project (3.45%). Out of 40 points, the average total project score from Time 1 to 3 was 14.23 ( $SD=4.63$ ), 20.82 ( $SD=4.26$ ), and 25.11 ( $SD=4.97$ ), respectively. The post-CSA weighted scores were also normally distributed with skewness and kurtosis values of 0.44 and -0.30, respectively.

### 5.2 Reliability and Validity of Projects from Field Test

For the reliability testing, the first author of this paper assessed all 228 projects while a research assistant assessed 80 projects (35%). There was a substantial agreement across all ScratchJr Project Rubric's subcategories; the average Krippendorff's alpha was 0.95, ranging from 0.81–1.00 for each subcategory (Krippendorff, 2018). The Background Customization category had the lowest agreement ( $\alpha=0.81$ ) due to the children's artwork that can be highly abstract, and therefore raters had difficulties noticing the intricate details. For example, a dark color background that is fully painted with also dark color small drawings and inserted shapes.

The ScratchJr Project Rubric intends to measure purposeful coding projects through its constructs. The rubric shows construct validity as the rubric's subcategories were derived from existing literature on coding rubrics and creativity. The rubric's criterion validity was shown by the significant positive correlation between the total project scores at Time 3 and the post-implementation CSA scores ( $r=0.35$ ,  $p<0.01$ ), despite children's grade level and gender.

**Fig. 5** The total project score distributions from Time 1 to Time 3





**Fig. 6** Left: Project with a high total project score. Right: Project with a low total project score

**Table 4** The Frequency and Mean Score of each Subcategory

Subcategories	Frequency ( $n = 76$ )	Mean	Subcategories	Frequency ( $n = 76$ )	Mean
Sequencing	100.00%	3.32	Number of Settings	100.00%	3.64
Number Parameter	97.37%	3.42	Number of Characters	100.00%	3.62
Events	92.11%	2.20	Background Customization	96.05%	3.23
Parallelism	92.11%	1.27	Character Customization	81.58%	2.08
Repeats	27.63%	2.62	Sound	38.16%	3.31
Coordination	22.37%	3.18	Animated Look	36.84%	2.54
			Speech Bubble	15.79%	2.17

Examples of projects with high and low scores in Fig. 6 demonstrated that this rubric can differentiate projects with varying complexities. Both projects in Fig. 6 were created by two first graders with the “Where the Wild Things Are” story theme. The project on the left contains highly complex coding concepts such as “Events” (sending messages) and “Parallelism” (not visible in the figure), while the project on the right is composed of simple motion blocks. Figure 6 (left) project has a scene where the main character, Max, gets off the boat, says “By” (bye) to the mouse then arrives home on the final page. Additionally, the project design in Fig. 6 (left) is highly elaborated with drawn characters and a picture of the “Where the Wild Things Are” graphic in the second-page background. Contrastingly, Fig. 6 (right) uses a less complex sequence, where Max moves forward and then backward before the project goes on to the next page. There is also minimal customization in Fig. 6 (right) such as changing the characters’ shirt colors.

### 5.3 Distribution by Subcategories

The frequency and mean score of each subcategory at Time 3 are shown in Table 4. Coding Concepts that appeared most frequently were “Sequencing”, “Number Parameter”, “Events”, and “Parallelism”. Out of a maximum score of 4, “Sequencing” and “Number Parameter” had the highest mean scores of 3.32 ( $SD=0.64$ ) and 3.42 ( $SD=0.91$ ), respectively. Although “Parallelism” appeared in most projects, it had

the lowest mean score of 1.27 ( $SD=0.68$ ). This finding suggests that children used “Parallelism” often but at the basic level: to have different characters run commands simultaneously. Furthermore, although “Coordination” only appeared in 22.37% of the projects, its mean score of 3.18 ( $SD=0.95$ ) was higher than many other concepts.

Projects on average scored higher on the Project Design than the Coding Concepts. Results demonstrated that all projects had at least a character with one functional code and a setting. Additionally, the projects’ backgrounds were more highly customized ( $\mu=3.23$ ,  $SD=0.70$ ) than the characters ( $\mu=2.08$ ,  $SD=0.75$ ). Children regularly customized their projects’ background with texts, drawings, and static ScratchJr characters without coding blocks (e.g., bed, chair, flower, and cloud).

#### 5.4 Sensitivity across Time

The final longitudinal model in Appendix 2 showed that the ScratchJr Project Rubric is sensitive to the changing total project scores. At Time 1, the model estimated that first graders would get 12.27 ( $p<0.001$ ) on their total project scores, while second graders would get 5.56 scores more ( $p<0.001$ ) than first graders. With one passing time point, the total project scores were estimated to be 7.04 points higher for first graders and 3.63 points higher for second graders. Furthermore, grade level but not gender was a significant predictor ( $p<0.001$ ) of the changing project scores over three timepoints. Lastly, the post-screening analysis on the final model showed that the residuals were normally distributed, homoscedastic, and independent. The final model included all necessary random effects, any additional random effects to the model resulted in a convergence error.

## 6 Discussion & Conclusion

With the growing popularity of coding applications for young children to create their own open-ended animated projects, a valid assessment tool to evaluate the project’s quality in terms of coding skills and design elements is critically needed. The first research question (RQ1) of the study is how to design a coding rubric for young children that assess the coding, design, and purposefulness aspects. The second question (RQ2) is asking whether the newly designed rubric in RQ1 would have acceptable psychometric properties. For RQ1, this paper describes the rationale and process in developing the ScratchJr Project Rubric and the rubric’s field testing on first and second graders’ projects. For RQ2, results from evaluating 228 projects demonstrated the rubric’s reliability and validity for assessing young children’s coding projects. Particularly, the rubric demonstrated reliability through the sufficient agreement between raters across three rounds of testing. The rubric showed criterion validity through the significant positive relationship between the rubric and the CSA. Construct validity was established as its domains were derived from prior literature on coding assessments. Additionally, the rubric’s subcategories score distributions reflected what previous studies have observed on young children’s programming competencies.

The ScratchJr projects analyzed in this study demonstrated that it was common for children in early elementary grades to create functional sequences with four to six

unique motion blocks. However, it was less common for them to utilize more advanced coding concepts such as “Events”, “Repeats”, “Coordination”, and “Parallelism”. For instance, some of the infrequently used blocks under Events were “Start on Tap,” “Start on Bump,” and multiple color message blocks. These findings aligned with a prior study, which found that many first graders were not able to construct sequences with the “Send Message” and “Open Message” blocks (Strawhacker & Bers, 2019). Message blocks are challenging for young children, as they require an understanding of “Control Flow”, which is similar to “Repeat” (Loop) and “Speed” (Coordination) blocks. Children may find the Control Flow concept less apparent as this command acts as a trigger or a modifier to be used with the other block categories, including Motion, Look, or Sound (Strawhacker & Bers, 2019). Aligning to the current study, prior research has also identified Parallelism as an advanced concept for children across ages. For example, a study with middle school students reported that only a third of the coding projects analyzed used Parallelism (Denner et al., 2012).

Children may develop valuable skill sets such as creativity and expressivity when integrating both coding concepts and project design in their projects (Ge et al., 2015; Liao, 2016). Thus, the rubric presented in this paper involves both, since coding scores alone could not account for the nuances of a ScratchJr project’s quality. Most children invested effort in expressing ideas and personalizing their projects. For example, most of the ScratchJr projects evaluated had extra customizations such as drawings, self-portraits, narrated voices, or spelled out project names. On average, each project got almost a perfect score under the Number of Characters subcategory (using six or more characters with functional sequences). The prevalence of aesthetic customization in young children’s projects from these findings supports pedagogical claims encouraging educators to provide sufficient classroom time for project design (Strawhacker et al., 2018).

Lastly, this study indicated that the ScratchJr Project Rubric is sensitive to change. Notably, the increasing project scores in the study aligned with the fact that children in both grades became more knowledgeable coders as they participated in the Coding as Another Language curriculum. The total project scores at Time 3 for both grades were approximately 65% of the maximum score; the highest score that a child reached was 85%. This percentage was not surprising as children must use the most challenging concept in ScratchJr to achieve each criterion’s top score. Ultimately, these outcomes suggest that the ScratchJr Project Rubric can provide a high ceiling for children’s mastery levels to grow, aligning with a report that an ideal assessment tool should be able to capture children’s wide range of capabilities (Relkin & Bers, 2019).

In conclusion, the ScratchJr Project Rubric is an assessment tool for children’s coding projects that demonstrates reliability and validity. Although the rubric has only been tested by researchers and not teachers, it is both practical to implement and sufficiently brief to be suitable for classroom use. The rubric’s subcategories can provide meaningful insights into how children create open-ended coding projects and can illuminate areas of the curriculum that need to be strengthened. For instance, teachers may consider providing extra support or encouraging children to test out concepts with low scores in the rubric such as “Parallelism” and “Repeats”. Finally, while this study specifically utilized the ScratchJr app, the rubric’s constructs still apply to other coding applications that use block-based programming with similar programming and design features to ScratchJr. Examples of these features include movement blocks, conditional blocks, and decorating tools.



## 7 Limitation and Future Directions

A single evaluation instrument cannot capture children's learning. Thus, one of the limitations of the present study is that the ScratchJr rubric alone cannot inform the process of how children create expressive coding projects. For example, Brennan and Resnick (2012) suggested that project rubrics do not capture the project creation process; therefore, cannot inform the assessors whether the child understands the coding concepts. Children may have received help from their teachers or friends to create advanced coding sequences; thus, the rubric might fail to capture children's actual ability. Furthermore, the curriculum may highly influence children's choice of coding blocks in their own open-ended projects. For example, the "Go to Page" block appeared in most projects evaluated in this study, which might be due to the curriculum's prompt to create a story with multiple pages. Alternatively, children rarely used "Start on Tap" and "Start on Bump," although these two blocks are on the same score level as the "Go to Page" block. Consequently, it is challenging to determine whether the "Start on Tap" and the "Start on Bump" are conceptually more complex for children than the "Go to Page" block.

Another study limitation is that it is not possible to determine whether what children created was intentional, without interviewing them. For example, children may be joining random coding blocks to create long functional sequences without understanding all the commands. These random blocks make it difficult for assessors to detect anything other than obvious errors with the ScratchJr Project Rubric—a sequence that has a repeat loop that does not repeat, or an example shown in Fig. 4. Contrastingly, children may also intentionally create coding sequences that do not make logical sense. For example, putting a "Hide" block at the beginning of a long sequence, meaning that all the following actions will be invisible.

Thus, in future studies the rubric might need to be used in conjunction with other forms of assessments to triangulate children's learning and allow differentiation between an educated guess and actual knowledge of the concepts. An essential next step is to observe ScratchJr lessons and interview children on what they created. Feedback on how and why children construct sequences can enhance our understanding of their actual competencies. Input from children will be beneficial for the next round of the ScratchJr Project Rubric revision. Additionally, instead of having researchers test the rubric on children's projects, it is necessary for teachers to do so. To improve the rubric's design for classroom use, it will be helpful to measure the time required for teachers to implement the rubric and obtain feedback. Like the automated project rubric for Scratch, called Dr. Scratch (Moreno-León & Robles, 2015), it is possible to automate most of the ScratchJr Project Rubric's concepts. However, some concepts that heavily emphasize young children's intentionality such as "Coordination" and "Sound", are less straightforward for machines as it is required to understand the themes of the projects. Lastly, future studies should explore the extent to which the difficulty of the open-ended project prompts influence students' performances. Nevertheless, the current rubric version is robust and sufficient to provide meaningful insights about children's ability to express and transform their coding and design skills into coding projects.

## Appendix 1

### Tables 5 and 6

**Table 5** Examples of Sequencing and Repeats Coding Concepts Grading Criteria

	1 point	2 points	3 points	4 points
Sequencing	1 block was used or none of the code runs	2–3 different blocks were used appropriately (Excluding end block, identical blocks next to each other, and unintentional blocks)	4–6 different blocks were used appropriately (Excluding end block, identical blocks next to each other, and unintentional blocks)	7 or more different blocks were used appropriately (Excluding end block, identical blocks next to each other, and unintentional blocks)
Repeats	Attempted but no correct usage of repeat block or forever block	Used repeat block with one other block inside or used forever block appropriately	Used repeat block with more than one other block inside appropriately	Used nested loop appropriately (two or more repeat blocks or repeat block with forever block)

## Appendix 2

**Table 6** Results for the Final Model with Time Variant and Invariant Predictors of the Total Project Scores

Parameters		Model		
		Est	SE	P
N = 228				
Models for the Means				
$\beta_{00}$	Intercept	<b>12.27</b>	<b>0.76</b>	<b>&lt; 0.001</b>
$\beta_{10}$	Time	<b>7.04</b>	<b>0.56</b>	<b>&lt; 0.001</b>
$\beta_{01}$	Grade Level	<b>5.56</b>	<b>0.97</b>	<b>&lt; 0.001</b>
$\beta_{02}$	Gender	0.54	0.94	0.57
$\beta_{11}$	Time X Grade Level	<b>-3.41</b>	<b>0.72</b>	<b>&lt; 0.001</b>
$\beta_{12}$	Time X Gender	-0.68	0.70	0.34
Model for the Variances				
$\tau_{U0}^2$	Random intercept variance		1.27	
$\sigma_e^2$	Residual variance		17.67	

Bolded values have  $p < 0.001$

**Acknowledgements** We would like to sincerely express our gratitude to the project grading team including Alan Bers, Anika Kawsar, Kianie Ramirez, Melanie Becker, and Patrick Nero. Special thanks to Sara Johnson, Emily Relkin, and Madhu Govind for reviewing and being extremely thoughtful on this manuscript. We would also like to give an appreciation to the teachers involved in this project and the DevTech Research Group.

**Funding** This work was generously supported by the Department of Education (DoEd), Grant # U411C190006

**Data Availability** Not applicable.

**Code availability** Not applicable.

## Declarations

**Disclosure of potential conflicts of interest** The authors declare that they have no potential conflict of interest.

**Ethical Approval** All procedures performed in studies involving human participants were in accordance with the ethical standards of the Tufts University Social, Behavioral & Educational IRB protocol no. 1810044.

**Consent to participate** Informed consent was obtained from the educators and parents/guardians of participating students.

**Consent for publication** Informed consent was obtained from the educators and parents/guardians for publishing the findings from this study.

## References

- Angeli, C., & Valanides, N. (2020). Developing young children's computational thinking with educational robotics: An interaction effect between gender and scaffolding strategy. *Computers in Human Behavior*, *105*, 105954. <https://doi.org/10.1016/j.chb.2019.03.018>
- Basu, S. (2019). Using Rubrics Integrating Design and Coding to Assess Middle School Students' Open-ended Block-based Programming Projects. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*.
- Bers, M. U. (2019). Coding as another language: A pedagogical approach for teaching computer science in early childhood. *Journal of Computers in Education*, *6*(4), 499–528. <https://doi.org/10.1007/s40692-019-00147-3>
- Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vancouver, Canada* (Vol. 1, p. 25).
- Cateté, V., Snider, E., & Barnes, T. (2016). Developing a Rubric for a Creative CS Principles Lab. *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, 290–295. <https://doi.org/10.1145/2899415.2899449>
- Chen, Y. F., & Martin, M. A. (2000). Using Performance Assessment and Portfolio Assessment Together in the Elementary Classroom. *Reading Improvement*, *37*(1), 32–38.
- Clarke-Midura, J., Lee, V. R., Shumway, J. F., & Hamilton, M. M. (2019). The building blocks of coding: A comparison of early childhood coding toys. *Information and Learning Sciences*, *120*(7/8), 505–518. <https://doi.org/10.1108/ILS-06-2019-0059>
- Coding as Another Language. (2021). Retrieved January 12, 2022, from <https://www.sites.tufts.edu/codingasanotherlanguage/curricula/scratchjr/>
- Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers & Education*, *58*(1), 240–249. <https://doi.org/10.1016/j.compedu.2011.08.006>
- de Ruiter, L. E., & Bers, M. U. (2021). The Coding Stages Assessment: Development and validation of an instrument for assessing young children's proficiency in the ScratchJr programming language. *Computer Science Education*, 1–30. <https://doi.org/10.1080/08993408.2021.1956216>
- Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education*, *63*, 87–97. <https://doi.org/10.1016/j.compedu.2012.11.016>
- Flannery, L. P., Silverman, B., Kazakoff, E. R., Bers, M. U., Bontá, P., & Resnick, M. (2013). Designing ScratchJr: Support for early childhood learning through computer programming. *Proceedings of the 12th International Conference on Interaction Design and Children - IDC '13*, 1–10. <https://doi.org/10.1145/2485760.2485785>
- Ge, X., Ifenthaler, D., & Spector, J. M. (Eds.). (2015). *Emerging Technologies for STEAM Education*. Springer International Publishing. <https://doi.org/10.1007/978-3-319-02573-5>
- Govind, M., & Bers, M. (2021). Assessing Robotics Skills in Early Childhood: Development and Testing of a Tool for Evaluating Children's Projects. *Journal of Research in STEM Education*, *7*(1), 47–68. <https://doi.org/10.51355/jstem.2021.102>
- Grover, S. (2020). *Computer Science in K-12: An A-To-Z Handbook on Teaching Programming*. Edfinity.
- Grover, S. (2020). Designing an Assessment for Introductory Programming Concepts in Middle School Computer Science. *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 678–684. <https://doi.org/10.1145/3328778.3366896>
- Krippendorff, K. (2011). Computing Krippendorff's Alpha-Reliability. Retrieved from [http://repository.upenn.edu/asc\\_papers/43](http://repository.upenn.edu/asc_papers/43)
- Krippendorff, K. (2018). *Content analysis: An introduction to its methodology*. Sage publications.
- Liao, C. (2016). From Interdisciplinary to Transdisciplinary: An Arts-Integrated Approach to STEAM Education. *Art Education*, *69*(6), 44–49. <https://doi.org/10.1080/00043125.2016.1224873>
- Moreno-León, J., & Robles, G. (2015). Dr. Scratch: A web tool to automatically evaluate Scratch projects. In *Proceedings of the workshop in primary and secondary computing education* (pp. 132–133).
- Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2020). Development of computational thinking, digital competence and 21<sup>st</sup> century skills when learning programming in K-9. *Education Inquiry*, *11*(1), 1–17. <https://doi.org/10.1080/20004508.2019.1627844>

- O'Quin, K., & Besemer, S. P. (1989). The development, reliability, and validity of the revised creative product semantic scale. *Creativity Research Journal*, 2(4), 267–278. <https://doi.org/10.1080/10400418909534323>
- Pila, S., Aladé, F., Sheehan, K. J., Lauricella, A. R., & Wartella, E. A. (2019). Learning to code via tablet applications: An evaluation of Daisy the Dinosaur and Kodable as learning tools for young children. *Computers & Education*, 128, 52–62. <https://doi.org/10.1016/j.compedu.2018.09.006>
- Relkin, E., & Bers, M. U. (2019). Designing an assessment of computational thinking abilities for young children. In L. E. Cohen & S. Waite-Stupiansky (Eds.), *STEM for early childhood learners: how science, technology, engineering and mathematics strengthen learning*. New York: Routledge. <https://doi.org/10.4324/9780429453755-5>
- Relkin, E., de Ruiter, L., & Bers, M. U. (2020). TechCheck: Development and Validation of an Unplugged Assessment of Computational Thinking in Early Childhood Education. *Journal of Science Education and Technology*, 29(4), 482–498. <https://doi.org/10.1007/s10956-020-09831-x>
- Relkin, E., de Ruiter, L. E., & Bers, M. U. (2021). Learning to code and the acquisition of computational thinking by young children. *Computers & Education*, 169, 104222. <https://doi.org/10.1016/j.compedu.2021.104222>
- Rose, S. P., Habgood, M. P. J., & Jay, T. (2017). *An Exploration of the Role of Visual Programming Tools in the Development of Young Children's Computational Thinking*, 15(4), 13.
- Salac, J., & Franklin, D. (2020). If They Build It, Will They Understand It? Exploring the Relationship between Student Code and Performance. *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, 473–479. <https://doi.org/10.1145/3341525.3387379>
- ScratchJr – DevTech Research Group. (2020). Retrieved January 12, 2022, from <https://sites.tufts.edu/devtech/research/scratchjr/>
- Seiter, L., & Foreman, B. (2013). Modeling the learning progressions of computational thinking of primary grade students. *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research - ICER '13*, 59. <https://doi.org/10.1145/2493394.2493403>
- Sherman, M., & Martin, F. (2015). The assessment of mobile computational thinking. *Journal of Computing Sciences in Colleges*, 30(6), 53–59.
- Strawhacker, A., Lee, M., & Bers, M. U. (2018). Teaching tools, teachers' rules: Exploring the impact of teaching styles on young children's programming knowledge in ScratchJr. *International Journal of Technology and Design Education*, 28(2), 347–376. <https://doi.org/10.1007/s10798-017-9400-9>
- Strawhacker, A., Lee, M., & Bers, M. U. (2018). Teaching tools, teachers' rules: Exploring the impact of teaching styles on young children's programming knowledge in ScratchJr. *International Journal of Technology and Design Education*, 28(2), 347–376. <https://doi.org/10.1007/s10798-017-9400-9>
- Strawhacker, A., & Bers, M. U. (2019). What they learn when they learn coding: Investigating cognitive domains and computer programming knowledge in young children. *Educational Technology Research and Development*, 67(3), 541–575. <https://doi.org/10.1007/s11423-018-9622-x>
- Sullivan, A. & Bers, M. U. (2019). Computer Science Education in Early Childhood: The Case of ScratchJr. *Journal of Information Technology Education: Innovations in Practice*, 18(1), 113–138. Informing Science Institute. Retrieved May 24, 2021 from <https://www.learntechlib.org/p/216646>
- Wang, D., Wang, T., & Liu, Z. (2014). A tangible programming tool for children to cultivate computational thinking [research article]. <https://doi.org/10.1155/2014/428080>.
- von Wangenheim, C. G., Hauck, J. C. R., Demetrio, M. F., Pelle, R., da Cruz Alves, N., Barbosa, H., & Azevedo, L. F. (2018). CodeMaster—Automatic Assessment and Grading of App Inventor and Snap! Programs. *Informatics in Education*, 17(1), 117–150. <https://doi.org/10.15388/infedu.2018.08>
- Wilson, A., Hainey, T., & Connolly, T. M. (2013). Using Scratch with Primary School Children: An Evaluation of Games Constructed to Gauge Understanding of Programming Concepts. *International Journal of Game-Based Learning*, 3(1), 93–109. <https://doi.org/10.4018/ijgbl.2013010107>
- Wright, S. (2010). *Understanding Creativity in Early Childhood: Meaning-Making and Children's Drawings*. SAGE Publications Ltd. <https://doi.org/10.4135/9781446251447>