Investigating Children's Spontaneous Gestures when Programming using TUIs and GUIs

Abrar Almjally
Al Imam Mohammad Ibn Saud Islamic University
Riyadh, Saudi Arabia
aamjally@imamu.edu.sa

Kate Howland
University of Sussex
Brighton, UK
k.l.howland@sussex.ac.uk

Judith Good
University of Sussex
Brighton, UK
j.good @sussex.ac.uk

ABSTRACT
Spontaneous gestures produced during mathematics learning have been widely studied, however, research on the role of gesture in computing education is limited. This paper presents an investigation into children's use of spontaneous gestures when learning programming using either a tangible user interface (TUI) or a graphical user interface (GUI). The study explored the relationship between spontaneous gestures, interface type and learning outcomes in a programming lesson for primary school students aged 6-7. In the study, 34 participants engaged in a learning activity lasting approximately 37 minutes, using a TUI or a GUI. The study used a between-subjects design, and mixed methods. Pre-test and post-test data were collected, and sessions were video recorded and subsequently coded and analysed. A video analysis scheme, adapted from mathematics education research, was used to code the spontaneous gestures produced during the learning session. We found a statistically significant difference between the mean learning gains of high-frequency gesturers and low-frequency gesturers, with the top quartile showing significantly greater learning gains. There was no significant difference in the frequency of gestures between interface types. A qualitative analysis of representational gestures showed that some children use spontaneous hand gestures to demonstrate abstract computational concepts, providing evidence for the embodiment of children's offline thinking in the computing domain.

Author Keywords
Computer, education; Embodiment; gesture; abstract; metaphors.

CSS Concepts
• Human-centered computing~ Human computer interaction (HCI)~Empirical studies in HCI
• Social and professional topics~ Professional topics ~ Computing education ~K-12 education

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
IDC 2020, June 21–24, 2020, London, United Kingdom.
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7981-6/20/06…$15.00.
DOI: https://doi.org/10.1145/3392063.3394408

In response to calls to improve computer science (CS) education [47], various countries in Europe, such as Estonia, England, and Finland, have introduced CS into their primary school curricula [22]. However, concerns about the theoretical evidence base for computing education have been raised [21], highlighting the need to further develop our theoretical and practical understanding of how best to teach computing concepts to children. In other domains, such as language learning, physics and mathematics, learning environment design and implementation has been informed by theories of embodied cognition (EC), which highlights the importance of actions and metaphors [17]. As a related subject, mathematics teaching has implications for computing teaching [1], and many teachers believe that physicality can have a beneficial effect in computing [41]. The use of object manipulation in the form of tangible interfaces is common in computing education, but there have been mixed findings about the benefits [38]. Furthermore, there has, to date, not been any investigation of how tangible programming interfaces might influence learners' use of gesture. More generally, there has been little empirical research on the benefits of adopting EC approaches to foster the development of computing concepts in children. Further research is needed to better understand the issues young learners face in the early stages of learning computing concepts, and whether these can be addressed and supported using EC approaches.

The research described in this paper applies insights from EC, including the use of gestures and conceptual metaphors, to the computing education domain. Our study investigates the use of gestures in programming learning sessions when students use a TUI or GUI block-based programming environment to manipulate a physical robot. In this context, a TUI has physical blocks that users can manipulate; whereas a GUI has virtual blocks that are manipulated on a screen. This is the first study to investigate the role of spontaneous gestures in the development of primary students' computing skills. It addresses the following research questions: How does interface type (TUI or GUI) affect children's spontaneous gestures? What is the relationship between children's spontaneous gestures and their learning outcomes? What types of gestures, if any, do children demonstrate while solving programming tasks and for what purposes are the gestures used?

RELATED WORK

Embodied cognition

Increasingly, cognitive science research has emphasized the role of the body and the environment in cognitive processing (e.g. [4, 12]). There are different perspectives on embodied cognition (e.g.,[40, 44]), however the most relevant for this study is the psychological perspective known as grounded cognition [18, 34], which focuses on evidence of sensory representations and mental simulations. Grounded cognition advocates that a full understanding of a concept involves the ability to create a mental perceptual simulation of it when retrieving the information or reasoning about it [4–6].

Research in science, technology, engineering, and mathematics (STEM) education has suggested that EC approaches can be helpful in these disciplines, as they depend on representational systems that require sensory encoding, such as data visualization, and rely on high abstraction, such as mathematical formulae or programming code. These factors require learners to understand how to link sensory representation with abstraction [13,15].

Moreover, there is evidence that adopting EC approaches, such as using object manipulation or gesturing in education, can not only enhance a student's ability to connect the abstract to the concrete, but also improve memory and cognitive skills, such as strategic or spatial cognition and the reasoning abilities used in problem-solving [13,15]. In addition, educational research shows that motor activity supports retention of learned concepts, because this activity establishes additional cues that represent and recall knowledge [9,10,24]. In contrast with simply visualising or hearing information, supporting information interactions with actions promotes deeper processing that creates stronger memory traces, enabling learners to activate multiple memory retrieval avenues [14].

In computing education, most studies that address EC approaches focus on object manipulation using TUIs, and do not investigate the role of spontaneous gestures and their relationship to students' learning gains. Existing studies examining the benefits of TUIs for learning programming have reported mixed results, with no clear consensus [38,39,41]. Our recent work [2], which this paper builds on, investigated the relationship between interface type (TUI or GUI) and learning gains, attitude toward computing, and enjoyment of computer-programming activities with primary school students in Saudi Arabia [47]. We found that learning gains were significantly higher for students using a GUI programming environment than those using a TUI. However, there was a significantly higher increase in positive attitude towards computing for the TUI group. There was no difference in activity enjoyment scores, which were high for both groups.

Embodied metaphors

The use of metaphors in speech and language reflects human experiences of the body and influences communication and cognition skills in terms of thinking, performing, and comprehending abstract notions [23]. For example, the understanding of time is grounded in experiences of linear motion in space, so the future is conceptualised as something in front of people, and the past is imagined as something behind them [32]. Conceptual metaphor theory describes how people represent or explain abstract concepts using image schemas. For example, a learner can map goals, such as a desired object, visually as a metaphorical movement along a linear pathway. Thus, individuals use image schemas to understand abstract concepts, such as "I have almost reached my goal," within an unconscious knowledge construction metaphor through image schemas [3]. Metaphors are commonly used in computer science terminology [33], for example, the use of relations such as *parent*, *child*, or *ancestor* in object oriented programming. Similarly, in networking, the embodied metaphor of the *handshake protocol* is used, and applications *listen* on ports for connections with other programs.

Gestures in learning and teaching

Spontaneous gestures produced when speaking are a pervasive element of human communication and can reflect what people are thinking [37]. Such gestures allow humans to construct complex explanations by reducing information processing and cognitive load [39]. This section addresses research on how gestures can support learning and teaching. We focus on studies that explore the role of gesture in mathematics learning, given that the subject is closely related to CS, and studies in the computing education field are limited. The literature suggests several benefits of gestures, as described in the following sections. They reflect the content of the semantic model more faithfully. It is therefore possible to catch a glimpse of students' understanding of a given concept in the gestures they adopt even when having difficulty articulating their understanding

Revealing implicit knowledge

Gestures can convey knowledge that is not expressed in speech [20] and understanding gestures can therefore provide teachers with insights into learners' thinking [31]. For example, Novak and Goldin-Meadow describe how a child discussing a Piagetian conservation problem stated that a given amount of water changed when it was poured from a tall, thin container into a short, fat container, indicating that the child did not understand the concept of conservation [30]. The child justified this belief by saying, "This one is taller than this one" while producing a C-shaped gesture representing the narrow width of the tall container, followed by a wider C-shaped gesture representing the wider width of the short container. The child emphasised the height dimension of the containers while speaking but used nonverbal gestures to express thoughts about the width dimension. These gestures conveyed unspoken information. Only by accounting for such gestures is a full understanding of the child's knowledge possible.

Enhancing problem solving and learning

Gestures not only give teachers insight into students' current knowledge, but they can also help students generate new ideas and enhance their learning. Broaders et al. [7] found that students who were asked to gesture when solving mathematics problems produced more correct and novel strategies when finding a solution. Additionally, children who were asked to gesture achieved higher learning gains than children who were not asked to gesture.

There is also evidence that gestures which occur without prompting have a beneficial effect for learners. Chu and Kita [10] found that learners who face difficulties when solving spatial visualisation problems spontaneously produce gestures to help with problem solving, and that this has a beneficial effect in their performance. The benefit of gestures appears to be distinct from the benefit of physical actions. Miriam et al. [29] compared physical action performed on objects with gestures. They found that children trained to gesture performed better on near and far-transfer problems than children asked to perform physical actions with objects.

Teacher gesturing enhances learning

There is evidence that when teachers use gestures, students can benefit more from instructions [11, 43] Ping and Goldin-Meadow found that children given instructions with gestures performed better when solving Piagetian conservation problems, compared with children who received instructions with no gesturing, regardless of the physical presence of the objects during the instruction. This research suggests that gesturing might facilitate the grounding of the lesson's abstract language in the concrete physical environment, and convey ideas through its "representational form" [19, 35]

Gestures in computer science learning

Very few studies have investigated the role of gestures in computer science learning. Manches et al.[36] examined gestures used by university-level computing students by asking 16 participants to explain three computing concepts (algorithm, loop, and conditional). Despite not explicitly being asked to gesture, the students generated 368 representational gestures, suggesting that CS concepts are commonly embodied through gestures. Manches and colleagues concluded that students built their gesturing on two embodied metaphors. The first metaphor is computing as a physical object. In this case, students simulated manipulating physical objects when referring to a range of computing constructs. The second metaphor was computing processes as motion along a path, in which participants moved their hands along one of three body-based axes when referencing temporal sequences [26]. However, this study did not examine how, or whether, certain gestures related to students' learning. This could be important in evaluating the potential of teaching gestures to assess and support learning [25].

Solomon et al. [42] conducted a study in which they observed novice students learning programming in high school. The researchers noted the gestures performed by the students and categorized them according to McNeill's taxonomies, which include deictic, iconic, metaphoric, and beat gestures [28]. They reported the challenges of fitting McNeill's taxonomies to the computing gestures produced by students and the need to develop a conceptual framework to support gesture analysis in computer education. They reported that gestures are potentially used as a problem-solving strategy and a way of communicating students' understanding and abstract concepts.

No published work has previously investigated the use of gestures in young children's programming or examined their effect on students' computing learning gains. In addition, there has been no investigation of the extent to which object manipulation in the form of tangible programming interfaces influences children's spontaneous gesturing. In our previous paper [47], we reported our finding that learning gains were significantly higher for children using a GUI programming interface than those using a TUI. In this paper, we explore the relationship between primary school pupils' gestures and learning gains, and the extent to which interface type (TUI or GUI) influences spontaneous gesture usage. Our work also presents an analysis of the type, location and meaning of the gestures that students and their teachers use while solving programming tasks.

METHOD

The study used mixed-methods, between-groups design. Interface type (TUI or GUI) and frequency of gesture (low or high) were the independent variables. Participants completed a pre-test, a 45-minute learning activity (which included programming a robot) and a post-test. In the learning activity, the participants were encouraged to explain their planned solution before they used the programming environment. Participants were not asked or encouraged to use any gestures – their use was therefore was spontaneous and unprompted. Student gestures, such as using their hand(s) or body to explain their solution, debug their code, and/or try to understand computational issues, were subsequently coded. The researcher also used gestures freely during the session. Given that previous research suggests that teachers' use of gestures can benefit student learning (as discussed in the previous section), researcher gestures were coded in order to check that there was no difference in gesture frequency between groups. The dependent variables were 1) normalized learning gain, measured by the difference between pre-and post-test scores, 2) participants' gestures.

Participants and setting

The study took place in a Saudi Arabia primary-school during normal hours. Two single gender classes took part in the learning activity, with a total of 44 students (22 males, 22 females) aged 6-7. Of these, 42 students' parents gave informed consent for their children to participate in the study. The participants were paired with children of the same-gender, because the classes are gender segregated in Saudi Arabia. They were paired based on their average scores in

math and science (subjects selected because they are related to computational thinking skills [25]). As there are no standardised tests for students of this age in Saudi Arabia, students' average coursework scores for the current semester were used. Participants were ranked based on their scores and assigned to conditions using blocked randomisation. In one case a request was made by a parent for two friends to be paired together, which was granted. The groups were manually adjusted to account for this, while maintaining roughly equivalent scores across groups

The pre-test was completed in participants' classrooms. The learning activities and post-test were completed in a room in the school. Each pair completed the learning activity session, which was recorded. Unfortunately, of the 21 sessions, the recordings for 4 sessions were damaged, therefore 17 sessions were coded (with a total of 34 participants). Table 1 shows participant distribution across conditions.

	GUI-F	GUI-M	TUI- F	TUI -M
N	**10**	**8**	**8**	**8**
Pairs	**5**	**4**	**4**	**4**
M&S M	**84.5 %**	**84.3%**	**82.5 %**	**82.5 %**
M&S SD	**4.4**	**6.2**	**6.5**	**4.6**

Table 1: Group characteristics

Materials

The materials used for the study were the programming environment, an activity sheet, a pre- and post-test, and a GoPro camera to record the learning sessions.

The programming environment

The programming environment is built on a system called Tica [24], chosen because it provides TUI and GUI programming blocks that are identical aside from the physical instatiation. Tica consists of TUI code blocks, GUI code blocks presented on a tablet touchscreen, a physical robot that can be programmed using the TUI or GUI interface, a mat on which the robot moves, and various tiles (e.g., 'start', 'finish') that can be placed on the grid on the robot mat so as to create tasks for it to complete. Both sets of blocks are designed to be as similar as possible and are the same size and colour (see Figure 1). There are three action blocks, used to control the robot's movement: 1) forward (the robot moves forward one square), 2) right turn, 3) left turn; one block for generating a sound (in this case, a buzzer sound); and an iteration block (which repeats the blocks placed within it two times). The TUI blocks were created using a 3D printer.

Figure 1: The TUI setting (left) and the GUI (right)

The app provides the GUI interface and is used to communicate with the robot. The app also records task data, including number of activities completed per participant, activity completion time, number of attempts per activity, and the commands used in each attempt. The GUI has a construction pane at the top of the screen, and a block inventory at the bottom. Blocks are selected by dragging them to the construction pane. In the TUI, users select blocks by physically placing them in front of the system. The play button on each interface runs the instructions, and a message pops up to inform the user that the robot is running. Once the robot stops running, the system moves to the 'task complete' screen if the code is correct, or alerts the user if it is incorrect. The robot used is an mBot, a commercial educational robot based on Arduino [12] selected for its suitability for the participants' age group. An orange arrow was placed on the top of the robot and used in the activity sheets to signify the direction the robot faces (Figure 3). The robot moves on the mat, which has a 5x5 square grid, and five types of re-configurable task-specific tiles: 1) 'S' (the starting tile), 2) a finish flag (the end goal), 3) a speaker (tile where the robot must stop and make a noise), 4) 'X' (a square that is blocked to the robot) and 5) grey tiles used to make a specific path for the robot to follow (see Figures 2 & 3).

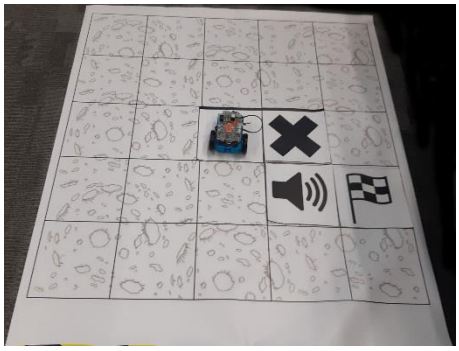

Figure 2: The mBot robot and the robot mat

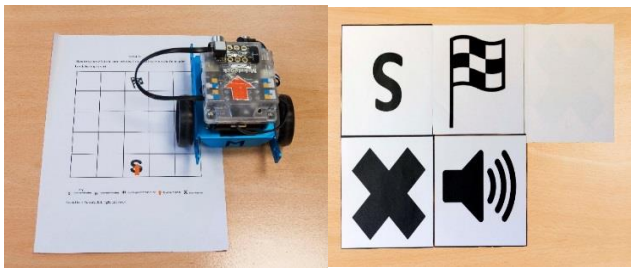

Figure 3: mBot with orange arrow and activity sheet (left); tile types (right)

Learning outcomes

The learning outcomes were based on the English Curriculum for Computing (specifically, the Key Stage 1 curriculum), since England is one of the few countries that teaches computing in primary school as standard (there is no standard primary computing curriculum in Saudi Arabia). The following learning outcomes were defined, targeting program creation, debugging, and comprehension:

A. Create and debug simple programs:
   - Describe: articulate the program goals.
   - Simple coding: write a simple program by selecting the correct commands and placing them correctly using one or more actions/types of action.
   - Complex coding: write an advanced program using the correct sequence of commands in one iteration.
   - Debug: predict what will happen, find out exactly what does happen, pinpoint and correct bugs.

B. Comprehension: explain the behaviour of simple programs by identifying suitable solutions.

Attainment test

Two versions of the attainment test were developed, which were semantically identical, differing only in terms of structure of the maps. Approximately half of the participants received version A as the pre-test and version B as the post-test, while the other half received the tests in reverse order. Both focused on testing the learning outcomes. All of the images in the tests, such as the grids and blocks, were visually similar to the learning activity. Each test consisted of six questions (two multiple-choice questions with four potential answers and four open-ended questions) and was worth 27 points in total.

The questions were as follows:

1. Look at the picture below (the arrow shows where the robot is, and which way it is facing) (Figure 4): [3 points]
   A. What shape will the robot make to get to the Finish square? [Intended to familiarise participants with questions and grid.]
   B. How many moves to the Right will the robot have to do to go to the Finish square?[Describe]
   C. What are the first 2 moves the robot will have to do to go to the Finish square? [Simple coding]
2. Which of the programs below will get the robot to make noise and go to the Finish square? [1 point / Debug]
3. Which square of the grid will the robot go to when following the program below? Use the given sticker (Figure 4) to show which will be the Finish square. [5 points/ Debug]
4. Write a program to make the robot go to the Finish square using stickers stickers (**Error! Reference source not found.**) [8 points/ Simple coding]
5. Now make the robot go to the Finish square using 5 blocks only using stickers [8 points/ Complex coding]
6. Which solution is more advanced/ better and why? [1 point / Comprehension]

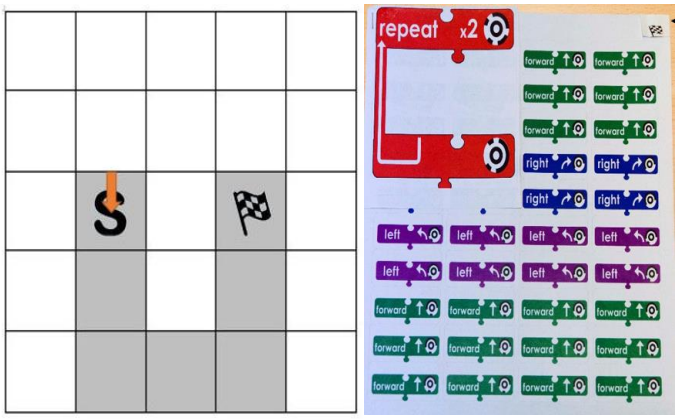

Figure 4: Example attainment test Question 1 (left) and the stickers (right)

Learning Activities

There were six activities with different difficulty levels (Figure 5). Each activity introduced or developed computing concepts, such as sequences, iteration, comprehension, or debugging, and aimed to fulfil the learning outcomes. Activities were presented to participants on the robot mat using the tiles and the activity sheet (shown in Figure 2 and Figure 3). In Activity 1, the robot must go from start to finish by moving forward four times. The aim was to familiarise students with the task and activity sheet and for them to gain confidence by solving simple problems requiring the repeated use of one type of block. In Activity 2, students must use the turn and noise commands to complete the task. In Activity 3, students must use turns to find the shortest path to the flag and avoid the 'X' tiles (shown in Figure 2). In Activity 4, the robot must make a noise twice using the repeat construct. Activity 5 requires the use of turns and the repeat block. In Activity 6, the robot must follow the grey squares to reach the flag, requiring a more complex program that includes turn blocks, some of which need to be placed within a repeat command.

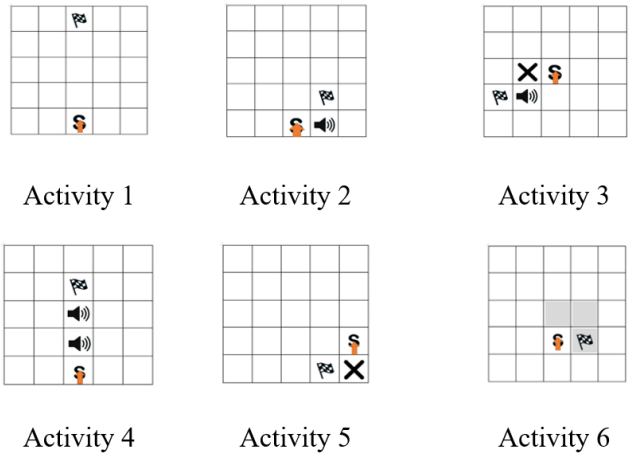

Figure 6: The learning activities

Procedure

The study took place November and December of 2018. Participants completed two sessions. In the first session, lasting 25 minutes, the researcher introduced herself, explained the study structure, and distributed the pre-test to participants (with one class receiving version A, and one class receiving version B). The stickers required for answering Questions 3 – 5 were given out with the test sheets. The researcher read the questions and multiple-choice options aloud to the whole class, as students were not fluent in reading, but students completed their test sheets individually at their desks. No time limit was given for the tests. The researcher made sure that everyone had finished answering each question before moving on to the next question.

The second session took place in a quiet room in the school and lasted approximately 60 minutes. In this session, pairs of participants completed the learning activities, followed by a post-test, which they completed individually. The procedure for both interfaces were the same. Participants were first given an overview of the session and the programming environment, and an explanation of the functionality of the action blocks. They were then asked to solve Activities 1–3, before an intervention to explain the functionality of the repeat block, and were finally asked to solve Activities 4–6. Before starting programming for each activity, participants were required to explain orally what they wanted the robot to do, discuss their solution together, and work together to program their solution. The aim of asking participants to explain their solution was to capture and examine their use of spontaneous gestures, however, the researcher did not ask participants to use gestures in any part of the study. In cases where the pairs disagreed about the solution, the researcher asked one participant to explain why they thought that the chosen block was correct, and then asked the other participant if they were convinced by their partner's explanation. If so, participants were asked to choose the robot's next move and again justify their choice. If not, the researcher asked them to explain why they were not convinced and to propose a different solution. There were only two cases where such an intervention was required.

Participants were assisted when they asked for help, with the researcher explaining the current issue and encouraging participants to find the solution themselves. The number of attempts participants could make was not limited. The researcher clapped and gave encouraging feedback each time an activity was completed, and participants generally showed enjoyment and excitement about their accomplishments. For Activity 4 to 6, participants were asked to solve the activity twice: once without the repeat block and once with the repeat block. Following the learning activities, the post-test was completed individually, with questions read aloud by the researcher, as in the procedure for the first session.

ANALYSIS

Overview of coding rubrics

Of the 17 videos that were coded, the mean duration of the activity session was 37 minutes. The sessions included three actors: the researcher and two participants. The videos were coded using the Elan video annotation software [48]. A unit of gesture was defined as the duration from the start of the movement until the hand returned to its resting position [28]. We excluded participants' self-adapting motions, such as scratching their heads and changing their hand positions from the lap to the desk, from the analysis, as they lacked semantic attachment.

Gestures were coded according to type, location, and purpose. We first created a coding scheme to identify the gesture types, based on the mathematic gesture taxonomy developed by Alibali [1], shown in Table 2. The locations where the gestures were performed are shown in Table 3, and the apparent purpose of the gesture was coded using the scheme shown in Table 4.

The first author coded all the video data. A random sample of 20% of the data was generated for second coding. Inter-rater reliability was determined using Cohen's Kappa scores, with strength agreement based on Fleiss [16]. Results were as follows: Type =.724 Good; Location =.897, Excellent; and purpose = .742 Good.

Gesture types	Code	Definition
Pointing (Deictic)	Pointing-gesture	Deictic gesture, reflecting grounded cognition in the physical environment e.g. pointing to the code, or the robot path
	Robot	Use Robot to physically demonstrate its movement

Representational	Hand gesture	Representational gesturing* with hands e.g. using hands to indicate turn (includes one instance using a foot)
	Body gesture	Representational gesturing* with body e.g. rotating the whole body to indicate turn

* Representational Gesturing: A gesture handshape or motion trajectory, depicting aspects of their meaning either literally or metaphorically either by hand or body [1]

Table 2 Gesture Type

Location	Definition
Robot mat (RM)	An A0 mat on which the robot moves (see Figure 2).
Activity sheet (AS)	An A4 paper, which shows the task that the participants need to solve (see Figure 3).
Code	The programming blocks participants use to solve the task
Air	Gestures performed in the air, without reference to the AS or RM

Table 3 Gesture location

Purpose	Definition
Direction	Demonstrate the direction of robot movement such as forward, backward, left turn, right turn, or double turns
Indicate object	Indicate the tiles in programming task such as a square. flag, noise, robot
Simulation	Simulate the route the robot will follow to complete the task
Counting	Counting squares
Iteration	Represent an iteration

Table 4 Gesture Purpose

RESULTS

How does interface type (TUI or GUI) affect children's spontaneous gestures?

An independent samples t-test was conducted to determine whether interface type affected children's spontaneous gestures. The mean number of gestures produced by the GUI group (M=28.56, SD=12.55) was lower than that produced by the TUI group (M=32.13, SD=16.41), but this difference was not statistically significant. An independent samples t-test was also run to determine whether there was a difference in the frequency of researcher gestures across the interface types. The mean number of gestures produced by the researcher was largely the same across the GUI group (M=43.67, SD=15.88) and the TUI group (M=43.38, SD=15.08), and the difference was not statistically significant.

What is the relationship between children's spontaneous gestures on and their learning outcomes?

Given the positive benefits of spontaneous gestures for learning reported in the literature, we were interested, firstly, in whether there were participants in this dataset who could be characterised as either "high gesturers" or "low gesturers", and secondly, in whether there was a relationship between gesture frequency and learning gain. We therefore divided the dataset by quartile according to frequency of gesture, characterising participants in the upper quartile as "high gesturers" and those in the lower quartile as "low gesturers". Normalized learning gain was calculated by taking in to the account the maximum possible gain or loss given the pre-test score [27], using the following formula: 100 × (post-pre)/ (100- pre). An independent samples t-test was conducted to determine if there is significant different between the groups in the pre-test. The mean of the scores in the "low gesturers" group (M=4.48, SD=2.56) was slightly higher than in the "high gesturers" group (M=4.07.13, SD=2.56), but this difference was not statistically significant.

The mean number of gestures for "high gesturers" was 50.25 (SD=10.32), while for "low gesturers" it was 13.75 (SD=2.91). The mean normalized learning gain for "high gesturers" was 11.38 (SD=4.96), and for "low gesturers" it was 5.64, (SD=3.6). An independent samples t-test showed a statistically significant difference in learning gains between the high-frequency gesturing and low-frequency gesturing groups (t (14) =2.68, p=.019). A further independent samples t-test was run to determine whether there was a difference between the researcher gestures in each frequency group. No statistically significant difference was found. Researcher gestures were considered because teacher gestures might benefit learners [11, 43].

What types of gestures, if any, do children, demonstrate when they solve programming tasks?

We coded 1020 participant gestures. The gesture type used most frequently by both participants (P) and the researcher (R) was pointing (P=695, R=592), followed by hand movement (P=258, R=100), body movement (P=60, R=27), and using the robot to point (P=6, R=7). Gestures were performed most frequently on/near the activity sheet (P= 627, R= 334), followed by near/above the robot mat (P=297, R=197), the code (P=63, R=173), and in the air (P=34, R=22). The purposes of the gestures were Simulation (42.7%), Direction (37.8%), Indicate object (11%), Counting (6.8%), and Iteration (1.6%).

How does gesture type relate to gesture purpose, and do identifiable patterns emerge when participants use specific types of gestures?

To identify patterns that emerged according to participants' gesture types and purpose, we used a cross-tabulation table

that showed the data for a given gesture's purpose and the most frequent type of gesture used. The table below indicates the percentage of gesture types used for specific purposes.

Purpose \ Type	Pointing (Diectic)		Representational		
	Pointing	Robot	Hand	Body	Total
Simulation	52.5%	66.7%	19.%	30%	436
Direction	20.9%	33.3%	76.7%	68.3%	386
Indicate object	16.1 %	0%	0%	0%	112
Counting	9.9%	0%	0%	0%	69
Iteration	.6%	0%	4.3%	1.7%	16
Total	695	6	258	60	1020

Table 5 cross-tabulation for gestures type and purpose

Most of the intentions analysed demonstrate a simple movement or simple physical concept such as simulation, direction, object, and counting. However, it is interesting to examine the use of gestures to represent iteration, which is an abstract computing concept. Abstract concepts are very common in computing, and noticing young children representing them spontaneously through gesture is noteworthy. Therefore, the next subsection discusses the gestures that were used to represent the iteration concept.

Representational gesture: Insights on the representation of an abstract concept (iteration)

In eight sessions, across 1020 gestures and 34 participants, only 10 participants performed gestures that represented the abstract concept of iteration (12 hand and body gestures in total). Pointing to the Repeat block was not included in this analysis as we were specifically interested in investigating the use of representational gestures for abstract concepts. Table 6 categorizes the representational gestures exhibited by each participant while solving the programming task. No significant differences in learning gains were found between the participants who used gestures that represented an abstract concept and the others.

Action	Description	Example
Hand	Performing a circular motion.	 **Figure 7 P1**
Hand		 **Figure 8 P2**
Hand	Repetitive circular movement as if he is stirring something.	 **Figure 9 P3**
Hand	Both hands, in rolling movement.	 **Figure 10 P4**
Hand	Repeated the action command inside the repeat which is Forward using tilted hand, twice.	 **Figure 11 P4**

Hand	Circular gesture that indicates a container.	 **Figure 12 P5**
Body	Opening the arm in a circular shape and taking one step forward.	 **Figure 13 P6**

Table 6 Iteration representational gesture description

DISCUSSION

The aim of this study was to explore the relationship between spontaneous gestures, interface type, and learning outcomes in a programming lesson for primary school students aged 6-7. We conducted an analysis of spontaneous gestures generated by 34 participants who engaged in the learning session. They used either a TUI or a GUI and took a pre and post-test to measure the learning outcomes.

We investigated how interface type (TUI or GUI) affects children's spontaneous gestures. We found that there was no significant difference between the two interfaces groups in terms of gesture frequency. There is no evidence from this study that object manipulation encourages gesturing, but as both interfaces had the same physical output (a robot), further investigation of the role of physicality of the task and gesture could be warranted. This could be investigated by comparing a GUI programming environment that controls an on-screen robot and a TUI programming environment with a physical robot.

We investigated the effect of spontaneous gestures on children's learning outcomes. We characterised participants in the upper quartile as "high gesturers" and those in the lower quartile as "low gesturers". We found a statistically significant difference in the learning gains score between the high-frequency gesturing and low-frequency gesturing groups. This is a promising finding, which suggests that gestures have a role in supporting learning in computing [3, 8]. This is in line with the empirical work that has been carried out in mathematics [1]. Additionally, in computing education, Solomon et al. [42] found that students who did not use gestures when they explained control flow could not correctly predict the output. This might mean that encouraging students to be active and use gestures leads to improvement in their reasoning about the answers. Further work is needed to investigate the role of gestures in computing classrooms, examining the role of specific gestures instead of just observing spontaneous gestures.

Additionally, we investigated the different types of gestures that children produce when solving programming tasks. We found that participants used pointing or 'deictic' gestures most frequently. The main purpose of deictic gestures is to reflect the grounding of cognition in the physical environment; according to McNeill, these gestures physically link speech and associated mental processing gestures to the physical environment [28]. This was found in mathematics and described by Alibali [1]. In computing education, Solomon et al. [42] found that pointing gestures help to trace the code. In our study, student use of pointing gestures was particularly high when simulating the robot's actions. Deictic gestures were used to help students clarify the direction and object they were referring to while talking. Finally, pointing was used to count the number of squares that the robot would follow, which is a simple illustration of the grounded cognition theory that tools from the environment are used to off-load cognitive work [46].

The second type of gesture used was representational gestures using hand or body to represent an abstract concept. This type of gesture was used to demonstrate direction, such as a turn or forward action, or to simulate the robot's movement. This is similar to the use of iconic or representational gestures to represent a concrete idea. In mathematics, they are used to simulate an action or perception [1], and in computing, these gestures might be useful to facilitate communication and simulate action. Solomon et al. [42] found that participants who used pointing gestures rather than representational gestures for abstract concepts talked about their code at a low level, suggesting that students' gestures matched the abstract level of their understanding. Further work could investigate how different types of gestures might help improve students' learning of abstract concepts gradually.

Finally, representational gestures were used to represent an abstract concept. In our study, although the use of representational gestures was limited to the iteration concept, we show the possibility of young children generating representational gestures spontaneously when completing a programming task. In our study, students represented iteration as a circular shape; for example, Figures 7,8,9,10,12 and 13 suggest that the program will return to the point from where it started. This is a similar pattern to that found in [25] when CS students were asked to represent iteration. Another representation was the repetition of action blocks inside the repeat, such as in Figure 10 and 13 One student (Figure 12) considered the repeat a container, which also related to the image schema found in [25]. Future work may be able to

identify further patterns of gestures with more computing concepts and a larger representative sample.

Additionally, in mathematics, representational gestures are used to reflect conceptual metaphors that underlie mathematical concepts [1]. In computing, the use of this type of gesture may support learning of abstract concepts by externalising them and might highlight student misconceptions that might be difficult to perceive using other methods [42]. More research is needed to examine how representational gestures could reveal students' misunderstandings about an abstract concept and how teachers might be able to communicate with the students to correct their understanding.

There are some limitations to the study described in this paper. First, we explored how the frequency of spontaneous gestures supported learning and identified general gesture types. However, mathematics studies [29] have shown that the use of abstract gestures and concrete gestures can lead to differences in the depth and transfer of learning. Future work should investigate gesture types in these terms, to identify how they can be used most effectively in CS classrooms. Moreover, our work was limited to investigating the participants' use of gestures; however, the literature shows promising results of the meaningful use of gestures by teachers and how it supports learning in other fields. Further work could evaluate what types of gestures teachers can use to communicate computer concepts, understand the current status of their students, and reveal new ideas. In addition, although CS uses metaphors extensively, such as programming data structures (*stacks*, *queues*, *trees*, *pipes*), our study was limited to the concept of iteration because of our target users. Other work could examine the role of metaphors in computer pedagogy more broadly. Additionally, a sample size of 34 children was appropriate for this context but did not allow for conclusive statements about the effectiveness of the spontaneous gestures, which would require a larger sample size. Finally, even though participants were enrolled in the same first grade class, children differ in their ability to effectively communicate and engage in turn-taking behaviors that advance the common goal based on their development. Thus, failure to learn and failure to use gestures may both reflect a particular child's younger development rather than supporting evidence of gestures leading to greater comprehension.

CONCLUSION

In our previous work we investigated the role of object manipulation and the use of tangible objects to foster the learning of programming skills for first year primary school. In this paper we present our analysis and findings regarding the role of spontaneous gesturing and the types of gestures children use when solving programming tasks. This is the first study in computer science education which investigates the role of spontaneous gestures in supporting learning for young primary school students. Our next study will explore the role of spontaneous gestures in explaining programming concepts, adapting the approach used in [25] to explore whether and how children use meaningful gestures when doing so.

In terms of implications for design, although this research is preliminary, it suggests that there is a positive relationship between gesture frequency and learning gain. At the same time, however, the introduction of physical manipulatives (in the form of tangible programming blocks) does not seem to have an effect in increasing the frequency of gestures. Given the seeming importance of gestures for learning, this suggests that other means for supporting children's use of gesture should be explored. The suggestions for future work, described in the section above, will provide us with a deeper understanding of how-to best support student gesture and, in turn, the ways in which this support can be implemented in an environment for learning programming.

SELECTION AND PARTICIPATION OF CHILDREN

This study was approved by our institution's relevant ethics committee. The school head teacher was sent a consent form that describes the study; they signed the form and agree to host the study. Then, the head teacher contacted with parents of all Grade 1 students and sent them a parents/carers consent form. The parent of the participant read the combined information sheet and consent form and explained the study to their child. The adult signed the form acknowledging that they gave consent for their child to participate and returned the form to the school. At the start of each session, the researcher briefly explained the research to the child and asked for a verbal agreement to participate. If a child wanted to stop for any reason, the researcher would stop immediately and exclude the data from the study.

ACKNOWLEDGMENTS

The authors thank the school head teacher, teachers, parents, and students who participated in this study. In addition, we want to thank the second coder Norah Sarhan for her time and effort in reviewing the videos.

REFERENCES

[1] Martha W. Alibali and Mitchell J. Nathan. 2012. Embodiment in Mathematics Teaching and Learning: Evidence From Learners' and Teachers' Gestures. *J. Learn. Sci.* 21, 2 (April 2012), 247–286. DOI:https://doi.org/10.1080/10508406.2011.611446

[2] Abrar Almjally, Kate Howland, and Judith Good. 2020. Comparing TUIs and GUIs for Primary School Programming. (2020).

[3] Alissa N. Antle and Alyssa F. Wise. 2013. Getting down to details: Using theories of cognition and learning to inform tangible user interface design. *Interact. Comput.* 25, 1 (January 2013), 1–20. DOI:https://doi.org/10.1093/iwc/iws007

[4] Lawrence W. Barsalou. 2008. Grounded Cognition. 59, 1 (January 2008). DOI:https://doi.org/10.1146/annurev.psych.59.1030

06.093639

[5] Lawrence W. Barsalou. 2010. Grounded Cognition: Past, Present, and Future. *Top. Cogn. Sci.* 2, 4 (October 2010), 716–724. DOI:https://doi.org/10.1111/j.1756-8765.2010.01115.x

[6] John B. Black. 2010. An Embodied/Grounded Cognition Perspective on Educational Technology. In *New Science of Learning*. Springer New York, New York, NY, 45–52. DOI:https://doi.org/10.1007/978-1-4419-5716-0_3

[7] Sara C Broaders, Susan Wagner Cook, Zachary Mitchell, and Susan Goldin-Meadow. 2007. Making Children Gesture Brings Out Implicit Knowledge and Leads to Learning. *J. Exp. Psychol. Gen.* 136, 4 (2007), 539–550. DOI:https://doi.org/10.1037/0096-3445.136.4.539

[8] Jerome S. (Jerome Seymour) Bruner. 1966. *Toward a theory of instruction*. Belknap Press of Harvard University.

[9] Kira J. Carbonneau, Scott C. Marley, and James P. Selig. 2013. A meta-analysis of the efficacy of teaching mathematics with concrete manipulatives. *J. Educ. Psychol.* 105, 2 (2013), 380–400. DOI:https://doi.org/10.1037/a0031084

[10] Mingyuan Chu and Sotaro Kita. 2011. The nature of gestures' beneficial role in spatial problem solving. *J. Exp. Psychol. Gen.* 140, 1 (2011), 102–116. DOI:https://doi.org/10.1037/a0021790

[11] Ruth Breckinridge Church, Saba Ayman-Nolley, and Shahrzad Mahootian. 2004. The Role of Gesture in Bilingual Education: Does Gesture Enhance Learning? *Int. J. Biling. Educ. Biling.* 7, 4 (2004), 303–319. DOI:https://doi.org/10.1080/13670050408667815

[12] Andy Clark. 1999. An embodied cognitive science? *Trends Cogn. Sci.* 3, 9 (September 1999), 345–351. DOI:https://doi.org/10.1016/S1364-6613(99)01361-3

[13] Paul G. Clifton, Jack Shen-Kuen Chang, Georgina Yeboah, Alison Doucette, Sanjay Chandrasekharan, Michael Nitsche, Timothy Welsh, and Ali Mazalek. 2016. Design of embodied interfaces for engaging spatial cognition. *Cogn. Res. Princ. Implic.* 1, 1 (December 2016), 24. DOI:https://doi.org/10.1186/s41235-016-0032-5

[14] Fergus I M Craik and Robert S Lockhart. 1972. Levels of Processing: A Framework for Memory Research 1. *J. Verbal Learning Verbal Behav.* 11, (1972), 671–684.

[15] D. DeSutter and M. Stieff. 2017. Teaching students to think spatially through embodied actions: Design principles for learning environments in science, technology, engineering, and mathematics. *Cogn. Res. Princ. Implic.* 2, 1 (December 2017), 22. DOI:https://doi.org/10.1186/s41235-016-0039-y

[16] Joseph L. Fleiss, Bruce Levin, and Myunghee Cho Paik. 2004. The Measurement of Interrater Agreement. In *Statistical Methods for Rates and Proportions* (Third). John Wiley & Sons, Inc, 598–626. DOI:https://doi.org/10.1002/0471445428.ch18

[17] Jennifer M.B. Fugate, Sheila L. Macrine, and Christina Cipriano. 2018. The role of embodied cognition for transforming learning. *International Journal of School and Educational Psychology*, 1–15. DOI:https://doi.org/10.1080/21683603.2018.1443856

[18] A M Glenberg. 1997. What memory is for. *Behav. Brain Sci.* 20, 1 (March 1997), 1–19; discussion 19-55.

[19] Susan Goldin-Meadow. 2015. From action to abstraction: Gesture as a mechanism of change. *Dev. Rev.* 38, (December 2015), 167–184. DOI:https://doi.org/10.1016/j.dr.2015.07.007

[20] S. Goldin-Meadows. 1999. The Role of gestures in communication and thinking. *Trends Cogn. Sci.* 3, 11 (1999), 419–429.

[21] S. Grover and R. Pea. 2013. Computational Thinking in K-12: A Review of the State of the Field. *Educ. Res.* 42, 1 (2013), 38–43. DOI:https://doi.org/10.3102/0013189X12463051

[22] Fredrik Heintz, Linda Mannila, and Tommy Farnqvist. 2016. A review of models for introducing computational thinking, computer science and computing in K-12 education. *Proc. - Front. Educ. Conf. FIE* 2016-Novem, (October 2016), 1–9. DOI:https://doi.org/10.1109/FIE.2016.7757410

[23] John M. Lawler, George. Lakoff, and Mark Johnson. 1983. Metaphors We Live by. *Language (Baltim).* 59, 1 (1983), 201. DOI:https://doi.org/10.2307/414069

[24] Robb Lindgren. 2014. Getting into the Cue Embracing Technology-Facilitated Body Movements as a Starting Point for Learning. (2014).

[25] Andrew Manches, Peter E. McKenna, Gnanathusharan Rajendran, and Judy Robertson. 2019. Identifying embodied metaphors for computing education. *Comput. Human Behav.* (January 2019). DOI:https://doi.org/10.1016/J.CHB.2018.12.037

[26] Andrew Manches, Peter E. McKenna, Gnanathusharan Rajendran, and Judy Robertson. 2019. Identifying embodied metaphors for computing education. *Comput. Human Behav.* (January 2019).

DOI:https://doi.org/10.1016/J.CHB.2018.12.037

[27] Jeffrey D. Marx and Karen Cummings. 2007. Normalized change. *Am. J. Phys.* 75, 1 (January 2007), 87–91. DOI:https://doi.org/10.1119/1.2372468

[28] David McNeill. *Hand and mind : what gestures reveal about thought*.

[29] Miriam A. Novack, Eliza L. Congdon, Naureen Hemani-Lopez, and Susan Goldin-Meadow. 2014. From Action to Abstraction. *Psychol. Sci.* 25, 4 (April 2014), 903–910. DOI:https://doi.org/10.1177/0956797613518351

[30] Miriam Novack and Susan Goldin-Meadow. 2015. Learning from Gesture: How Our Hands Change Our Minds. *Educ. Psychol. Rev.* 27, 3 (September 2015), 405–412. DOI:https://doi.org/10.1007/s10648-015-9325-3

[31] Miriam Novack and Susan Goldin-Meadow. 2015. Learning from Gesture: How Our Hands Change Our Minds. *Educ. Psychol. Rev.* 27, 3 (September 2015), 405–412. DOI:https://doi.org/10.1007/s10648-015-9325-3

[32] Rafael E. Núñez and Eve Sweetser. 2006. With the Future Behind Them: Convergent Evidence From Aymara Language and Gesture in the Crosslinguistic Comparison of Spatial Construals of Time. *Cogn. Sci.* 30, 3 (May 2006), 401–450. DOI:https://doi.org/10.1207/s15516709cog0000_62

[33] D. Parmar, S.V. Babu, L. Lin, S. Jörg, N. D'Souza, A.E. Leonard, and S.B. Daily. 2017. Can embodied interaction and virtual peer customization in a virtual programming environment enhance computational thinking? In *2016 Research on Equity and Sustained Participation in Engineering, Computing, and Technology, RESPECT 2016 and 2nd Annual Conference of the IEEE Computer Society's Special Technical Community on Broadening Participation, STCBP 2016 - Co-located with the* . DOI:https://doi.org/10.1109/RESPECT.2016.7836179

[34] Diane. Pecher and Rolf A. Zwaan. 2005. *Grounding cognition : the role of perception and action in memory, language, and thinking*. Cambridge University Press.

[35] Raedy M. Ping and Susan Goldin-Meadow. 2008. Hands in the air: Using ungrounded iconic gestures to teach children conservation of quantity. *Dev. Psychol.* 44, 5 (2008), 1277–1287. DOI:https://doi.org/10.1037/0012-1649.44.5.1277

[36] Judy Robertson, Andrew Manches, and Helen Pain. 2017. "It's Like a Giant Brain With a Keyboard": Children's Understandings About How Computers Work. *Child. Educ.* 93, 4 (2017), 338–345. DOI:https://doi.org/10.1080/00094056.2017.1343589

[37] Wolff-Michael Roth. 2001. Gestures: Their Role in Teaching and Learning. *Rev. Educ. Res.* 71, 3 (June 2001), 365–392. DOI:https://doi.org/10.3102/00346543071003365

[38] Theodosios Sapounidis and Stavros Demetriadis. 2017. Educational robots driven by tangible programming languages: A review on the field. *Advances in Intelligent Systems and Computing 560*, 205–214. DOI:https://doi.org/10.1007/978-3-319-55553-9_16

[39] Daniel L Schwartz and John B Black. 1996. Shuttling Between Depictive Models and Abstract Rules: Induction and Fallback. *Cogn. Sci.* 20, (1996), 457–497.

[40] G. R. Semin and Eliot R. Smith. 2008. *Embodied grounding : social, cognitive, affective, and neuroscientific approaches*. Cambridge University Press.

[41] Sue Sentance and Andrew Csizmadia. 2017. Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Educ. Inf. Technol.* 22, 2 (March 2017), 469–495. DOI:https://doi.org/10.1007/s10639-016-9482-0

[42] Amber Solomon, Mark Guzdial, Betsy DiSalvo, and Ben Rydal Shapiro. 2018. Applying a Gesture Taxonomy to Introductory Computing Concepts. In *Proceedings of the 2018 ACM Conference on International Computing Education Research - ICER '18*, ACM Press, New York, New York, USA, 250–257. DOI:https://doi.org/10.1145/3230977.3231001

[43] Laura Valenzeno, Martha W Alibali, and Roberta Klatzky. 2003. Teachers' gestures facilitate students' learning: A lesson in symmetry. *Contemp. Educ. Psychol.* 28, 2 (April 2003), 187–204. DOI:https://doi.org/10.1016/S0361-476X(02)00007-3

[44] Francisco J. Varela, Evan. Thompson, and Eleanor. Rosch. 1991. *The embodied mind : cognitive science and human experience*. MIT Press.

[45] J.L. Weese. 2016. Mixed methods for the assessment and incorporation of computational thinking in K-12 and higher education. In *ICER 2016 - Proceedings of the 2016 ACM Conference on International Computing Education Research*. DOI:https://doi.org/10.1145/2960310.2960347

[46] Margaret Wilson. 2002. Six views of embodied cognition. *Psychon. Bull. Rev.* 9, 4 (December 2002), 625–636. DOI:https://doi.org/10.3758/BF03196322

[47] J.M. Wing. 2006. Computational thinking. *Commun. ACM* 49, 3 (2006).

[48] Peter Wittenburg, Hennie Brugman, Albert Russel, Alex Klassmann, and Han Sloetjes. 2006. ELAN : a professional framework for multimodality research. (2006), 1556–1559.

[49] Osmo – Award-Winning Educational Games System for iPad. Retrieved March 24, 2020 from https://www.playosmo.com/en-gb/