Tica: An environment for exploring tangible vs. screen-based programming

John Wilkie
Department of Informatics
University of Sussex
jw478@sussex.ac.uk

Judith Good
Department of Informatics
University of Sussex
J.Good@sussex.ac.uk

Abstract

This paper describes **Tica**, an environment designed to explore the differences between tangible and screen based interfaces when teaching programming to children aged 5–7 years. **Tica** comprises several components: a tangible programming language and a screen based equivalent, an Android app to interpret the programming languages and record data, and a physical robot that can be programmed using either the tangible or screen-based language. **Tica** was designed using a learner-centred approach, with a specific focus on the needs and abilities of the target user group.

Once implemented, an initial pilot study was carried out with 14 adults. Although participants using the screen-based interface completed problems more quickly than those using the tangible interface, participants using the tangible interface reported a higher level of enjoyment. Next steps include a more extensive study with the target population as well as some refinements to the **Tica** environment.

1. Introduction

Globally there has been growing concern over the need to educate young people not only to be safe consumers of digital technology, but also to understand how it works and to become creators themselves (Howland, Good, Robertson & Manches, 2015). In September 2013, the English Department for Education made significant changes to the National Curriculum: what was previously known as "ICT" became "Computing Programmes of Study". As a result, from Key Stage 1 (ages 5 – 7), children are required to be able to create and debug simple programs and to understand what algorithms are and how to implement them. This has resulted in a large increase in the number of young children being taught to program, leading in turn to an increasing focus on how they are taught. One approach to teaching young children to program has been to use tangible programming languages (TPLs). TPLs use physical objects that represent instructions that can be manipulated to write programs in a physical manner.

Although there is an intuitive sense that TPLs may be more appropriate for teaching programming to young children than more traditional screen based systems (Wyeth & Purchase, 2003), little research has been conducted to assess the benefits (if any) of tangible programming over screen based equivalents. The system described in this paper, **Tica**, was designed to allow researchers to investigate these differences in a controlled environment. **Tica** comprises two functionally equivalent programming languages: both use programming blocks, however, one interface is screen-based, while the other uses physical programming blocks. Both languages can be used to program a physical robot. Given that care has been taken to remove potential confounding variables between the two languages, it provides an ideal test-bed in which to investigate the potential benefits of TPLs for children.

The rest of the paper is structured as follows: Section 2 provides a brief summary of relevant background work, while Section 3 provides a comprehensive overview of the **Tica** system. Section 4 describes two preliminary evaluation studies, Section 5 describes future work, with general conclusions drawn in Section 6.

2. Related Work

Tangible programming languages are not a new idea. Perlman (1976) noticed the difficulties that children experienced when learning Logo, and believed that one of the barriers was the user interface. He therefore developed two alternatives to inputting instructions via the keyboard: the 'Button Box' (Figure 1), and the 'Slot Machine' (Figure 2). The 'Button Box' was a set of 4 individual boxes, each box having a different set of buttons to control the turtle. Boxes could be connected together as the child mastered more complex controls.

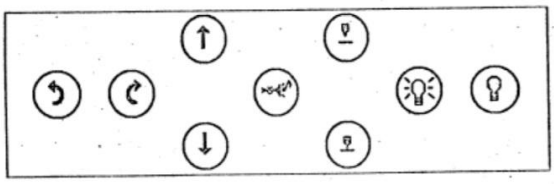

Figure 1 - Perlman's (1976) 'Action Box': one of 4 button boxes

The 'Slot Machine' had long rectangular boxes with slots into which the child could place instruction cards. On the boxes was a button that, when pressed, ran the instructions on the cards in linear order.

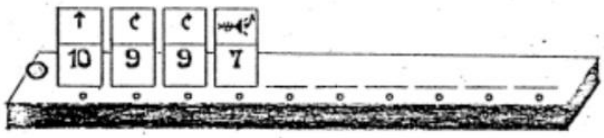

Figure 2 - Perlman's (1976) 'Slot Machine'

Since Perlman's initial work, a number of TPLs have been created which can be broadly categorised by 2 characteristics:

1) *The method the system uses to identify the blocks*: one family uses computer vision to identify blocks and their order, the other family uses in-built electronics allowing the blocks to uniquely identify themselves.
2) *What the TPLs control*: this can be split into one set controlling robots, and the other set being games running on tablets.

One of the most popular commercially available TPLs is Osmo coding (https://www.playosmo.com/en/coding/), which was originally developed as Strawbies (Hu, Zekelman, Horn, & Judd, 2015) for the Osmo (see Figure 3). Osmo is a system developed to enable tangible play with an iPad. It makes use of a mirror to reflect the area in front of the iPad into the front facing camera and then uses computer vision to recognise pieces placed in this area. With the Osmo coding app, the user can create programs using the tangible blocks. These programs control the movements of a monster character in the screen, who moves around a grid-like level to specified squares, collecting strawberries along the way.

Figure 3 - Strawbies (left) and Osmo coding (right)

The original Strawbies game made use of the TopCodes computer vision library (http://users.eecs.northwestern.edu/~mhorn/topcodes/). This library recognises ninety-nine unique small circular codes and can return an ID number and the location of each code. Strawbies is not the only tangible programming language to make use of the TopCodes library: Tern (Horn & Jacob, 2007)

and Quetzal (Horn & Jacob, 2006) also use it. Both Strawbies and Tern also use jigsaw-like puzzle pieces, ensuring the user connects the tangible blocks in the manner intended (see Figure 4).

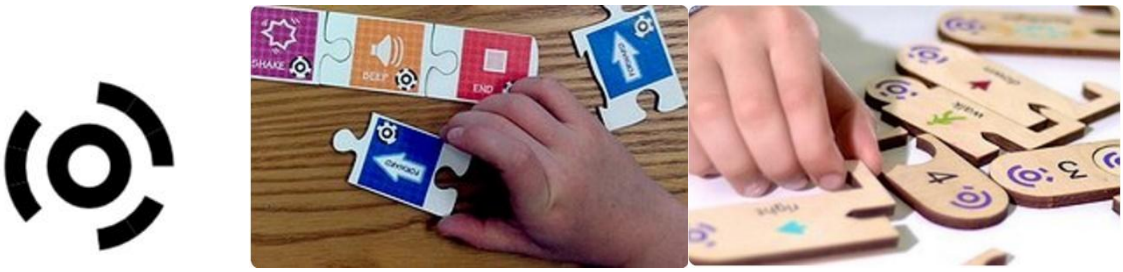

Figure 4 - A TopCode code (left), which is used by Tern (middle) and Strawbies (right)

One of the most recent developments in the tangible programming world is Project Bloks (https://projectbloks.withgoogle.com), a Google research project which is building a platform to allow others to create tangible programming experiences for children. This will potentially allow those with less technical experience to be able to easily create tangible interfaces.

Various studies have been carried out to compare the differences between tangible and screen-based interfaces for programming. Horn et al (2009) conducted a museum-based study in which researchers set up either a tangible or graphical interface to program a robot with simple instructions, then observed museum visitors. Overall, 260 visitors were observed, and 13 family groups interviewed. Their results suggested that visitors found both interfaces equally easy to use, however, visitors were more drawn to the tangible interface, and it was observed that the tangible interface led to more collaboration between group members. There was no significant difference in the time spent on either interface.

Sapounidis and Demetriadis (2013) carried out a study investigating children's preferences for a tangible programming environment (T_ProRob) or a screen-based equivalent for programming an NXT Lego robot. Three groups of children took part, aged 5-6, 7-8 and 11-12. Children first indicated their preference for either the tangible or graphical interface, then carried out a series of programming tasks using both interfaces. They were then asked which interface they found most enjoyable (at the mid-point of the study and again at the end), and which was easiest to use. Results showed that the tangible language was more attractive to the majority at first sight, and considered more enjoyable to use. In terms of ease of use, the two younger age groups reported finding the tangible interface easier to use, but the opposite was found for the eldest group. Observations showed that the tangible interface allowed multiple children to manipulate the system in parallel, hence programming was more group oriented than with the screen interface.

Sapounidis et al. (2015) performed similar testing while video recording sessions to document the time taken to complete tasks and the number of mistakes made. Five age groups participated: 6-7, 7-8, 9-10, 10-11 and 11-12 years. Results showed that tasks were completed more quickly with the tangible interface in all but the oldest group. Notably, the divergence in task completion time between the two interfaces decreased as age increased. The tangible interface also outperformed the graphical interface regarding number of errors and success of debugging.

Whilst the number of TPLs is increasing rapidly, there is a paucity of research into their potential advantages. The study by Sapounidis et al. suggests that tangible interfaces provide an advantage over screen based interfaces for younger children, however, in both the Horn et al. and Sapounidis et al. studies, users were only exposed to the interfaces for a single session. The next logical step would be to study the effect of interface type over multiple sessions, using a system designed to minimise extraneous differences between the two interfaces, and isolate the variables of interest. The next section describes the design and implementation of **Tica**, a system developed for this purpose.

3. The Tica System

The **Tica** environment comprises a number of components: a simple tangible programming language, a screen based programming language, a physical robot which can be programmed using either of the languages (Figure 5, left) and finally, a set of tiles of different types (e.g. 'start', 'finish') which can be formed into a grid to create tasks for the robot to complete (Figure 5, right). Either language can be

used, independently, to program the physical robot. Apart from the obvious tangibility of the tangible programming language, the two languages have been designed to be as similar as possible.

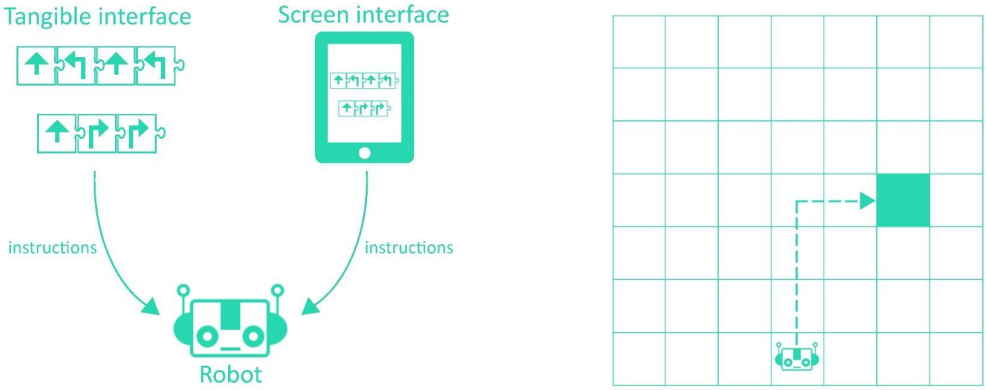

Figure 5 - The two programming interfaces (left) and a simple task for the robot to complete (right)

In designing **Tica**, a full list of functional and non-functional requirements was drawn up for the tangible components, the tablet computer application, and the robot. These requirements included a comprehensive consideration of the needs and abilities of the target users in terms of their cognitive and physical development (e.g. determining the appropriate size of the tangible objects based on typical fine motor skill development, font choices for ease of reading, appropriate use of icons, etc.).

3.1. The Tica App

The app provides functionality for a facilitator or teacher to design programming tasks for children and run sessions. It is used to communicate with the robot, create new tasks, and record and view data on task attempts. It also provides the platform used for programming using the screen interface.

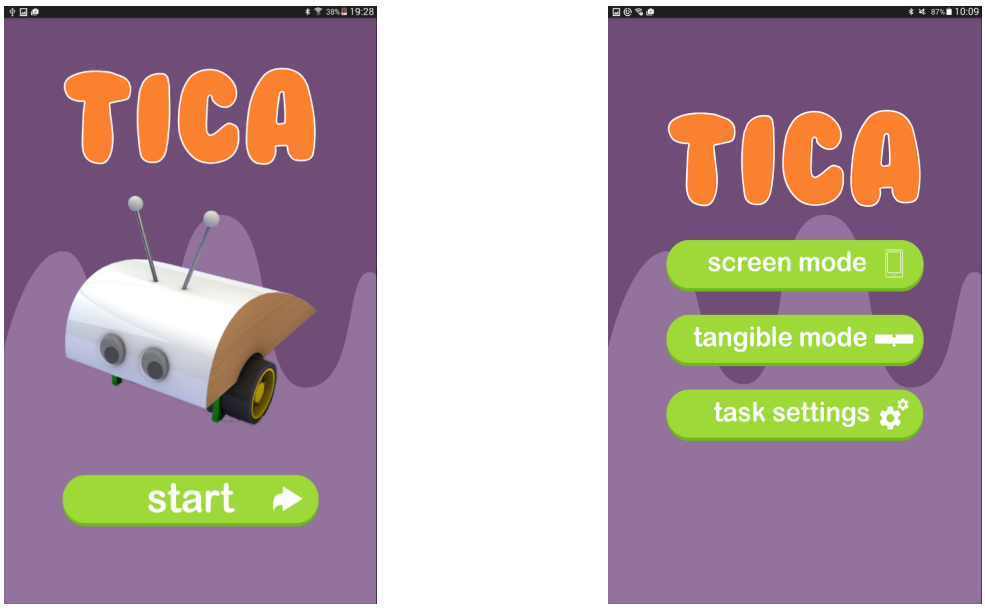

Figure 6 - Start Screen (left) and Main Menu (right)

Tica's start screen is shown in Figure 6 (left). While this screen is displayed, the system sets up the Bluetooth connection, and any problems making the connection are shown on this screen. The main menu (Figure 6, right) is designed for simplicity, and uses icons to support ease of understanding.

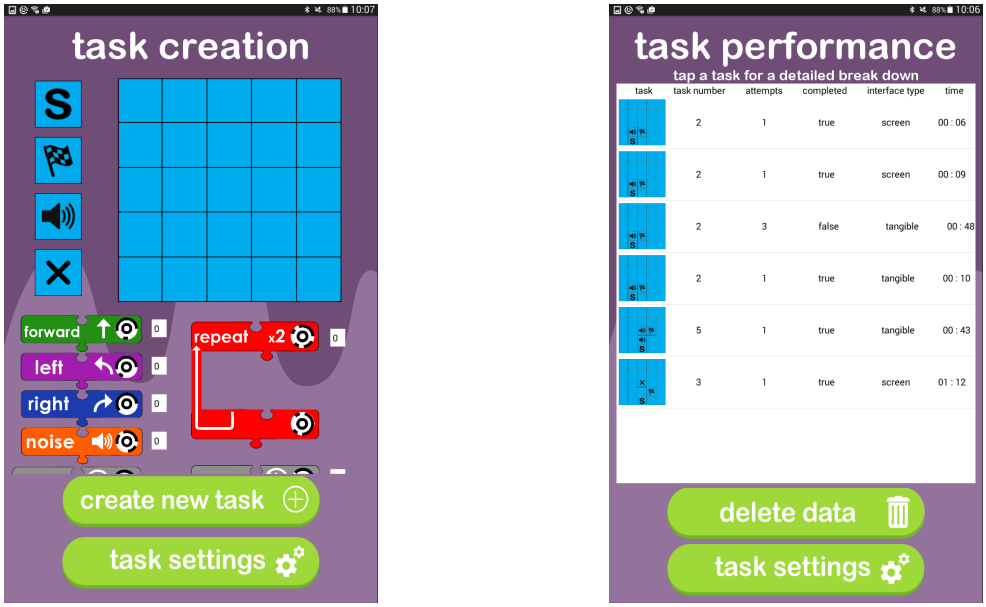

Figure 7 –Task Creation Screen (left) & Task Performance Screen (right)

The task creation screen (Figure 7, left) allows the facilitator to create a new task by choosing specific tiles for the grid, as well as the instructions needed to complete the task. The system will check if the task is well-formed (i.e. has a single start and finish tile, and at least one instruction), and will alert the user if this is not the case.

Tapping on a task in the task performance screen (Figure 7, right) allows the user to view details on that task attempt (e.g. instructions used), shown in a popup window. The user can also delete all current performance data by tapping the "delete data" button.

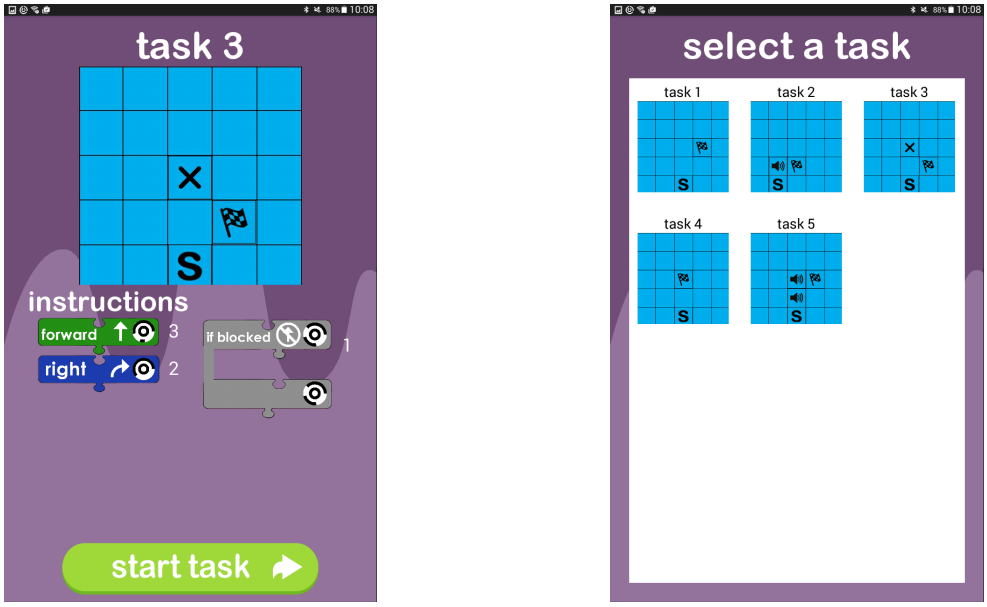

Figure 8 - Task description screen (left) and task selection screen (right)

The task description screen (Figure 8, left) shows each individual task created by the facilitator, while the task selection screen (Figure 8, right) shows thumbnails of the task grids that have been created to support ease of task selection. If there are more tasks than fit on the page, the area becomes scrollable.

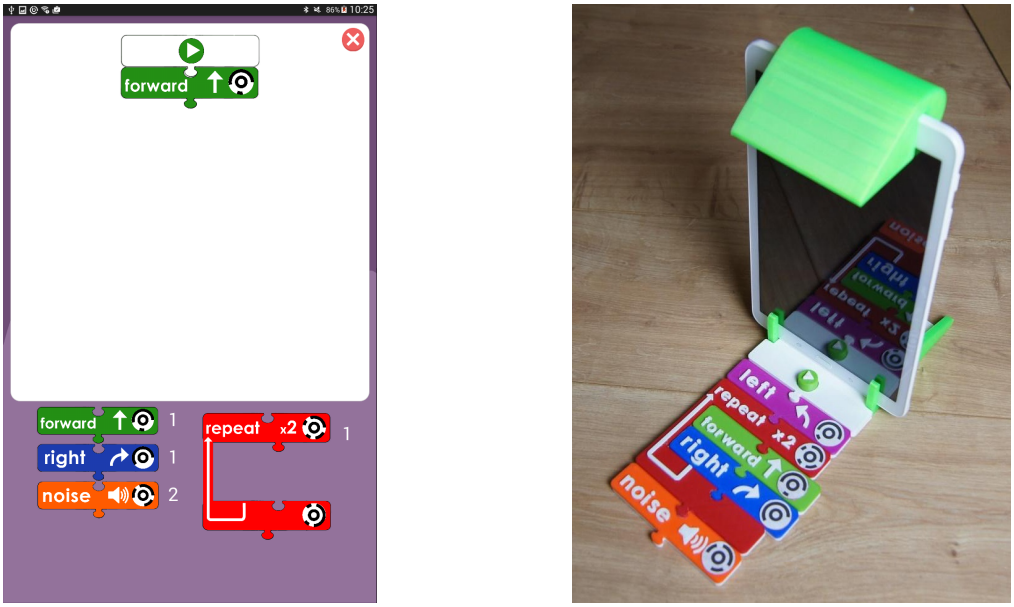

Figure 9 – Task attempt: screen interface (left) Tangible system (right)

The screen-based programming interface (Figure 9, left) has a construction pane at the top of the screen along with the inventory of instructions at the bottom. Instructions are selected by dragging them into the construction pane. In comparison, when using the tangible interface (Figure 9, right), users select instructions by physically placing them in front of the system. The play button on each respective interface will run the instructions, and a dialog will appear to inform the user the robot is running. Once the robot stops running, the system will either move to the task complete screen if the instructions were correct, or alert the user if they were incorrect.

3.2. The Tica Tangible Language

The tangible blocks were designed using key concepts from existing TPLs, and were created using a 3D printer. Seven instructions were created (see Figure 10): three instructions for controlling the direction of the robot, one instruction to generate noise from the robot, an iteration instruction and two control flow instructions. The condition for the control flow instructions is decided based on whether or not the path of the robot is blocked.

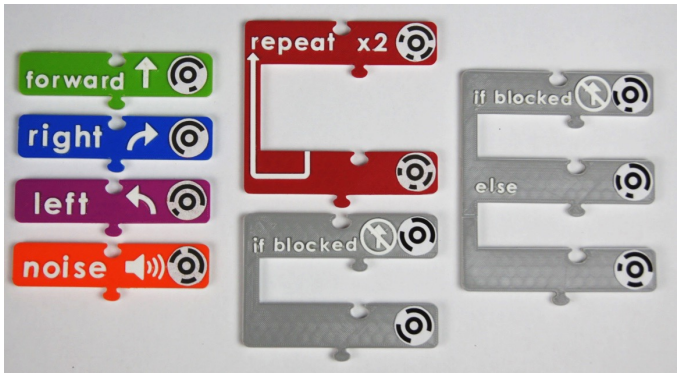

Figure 10 – Tangible instructions

Instructions are identified using a TopCode code. During the development of the instructions a prototype was built using OCR (optical character recognition) in order to identify instructions, but this resulted in a poorer identification rate.

A means was required to alert the Android application to read the instructions. A stand was built that incorporated a "play button" (Figure 9, right). Pressing this button signalled the application to process the current instructions via Bluetooth. The application then takes a photo that can then be processed to

identify the instructions. In order to reflect the image into the tablet's front-facing camera, a mirror is used. This use of a mirror was inspired by the Osmo (https://www.playosmo.com/en/).

3.3. The Tica Robot

The **Tica** robot is built around an Arduino Uno prototyping board and uses an Adafruit v2 motor shield to control two stepper motors. Stepper motors were chosen to provide semi-precise, repeatable movements. Communication with the robot is via Bluetooth. The case is a combination of wood and 3D printed plastic. The decision to design the case with a bug-like appearance was two-fold: firstly, so that the front and the back of the robot could be easily identified, and secondly, to seem somewhat less mechanical and more friendly, potentially allowing children to create a narrative around the robot.

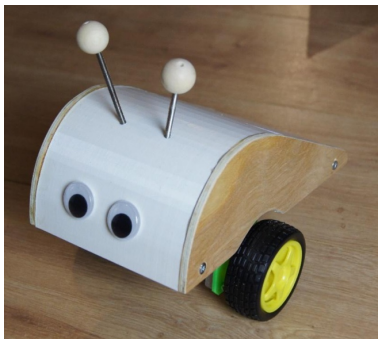

Figure 11 – Tica robot

3.4. The Tica Grid

In order to lay out tasks for the robot to complete, a set of tiles was required that could be used to create a grid, representing a task. Five tile types were created (see Figure 12).

Figure 12 – Tile types

The tile with the "S" is the tile the robot starts on. The tile with the finish flag marks where the robot must finish. The tile with the speaker icon is a tile the robot must stop on and make a noise (using the noise instruction). The tile with the cross represents a location that the robot must not enter. The empty tile is a normal tile with no special attributes.

3.5. Overall System Setup

The system can be set up to use either the screen based (Figure 13, left) or tangible interface (Figure 13, right). Switching between interfaces was designed to be quick and simple.

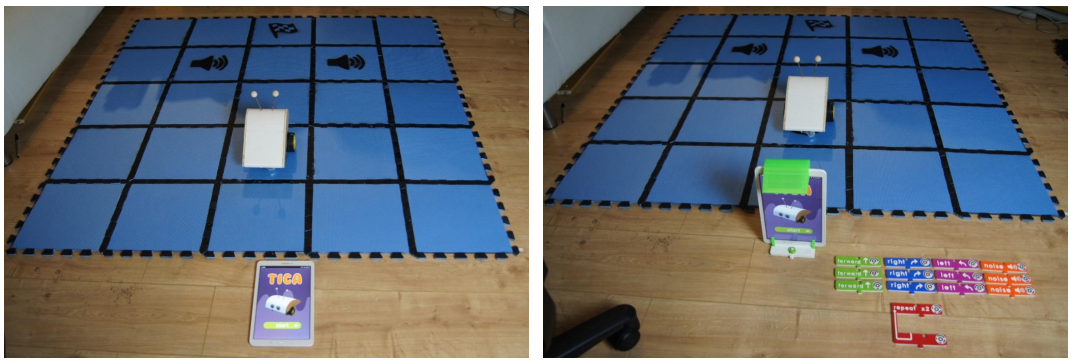

Figure 13 – Screen interface set up (left) and tangible interface set up (right)

4. Evaluation

Due to project time constraints, it was not possible to conduct an evaluation study with the target users (although this is planned as future work). Instead, a pilot study was carried out with adult novice programmers. Additionally, a further pilot study investigated the use of the system from the perspective of the facilitator/teacher. These studies are described below.

4.1 Novice programmer pilot study

The primary aim of this study was to examine the use of the system 'in the wild' in order to identify necessary improvements and ensure that the system's data recording mechanisms were accurate. In addition, the study aimed to gather preliminary data on the relationship between the interface type (screen-based or tangible) and task performance, as well as perceived enjoyment.

4.1.1 Participants

Opportunistic sampling was used to select fourteen adult participants with no prior programming experience. Participants ranged in age from 18 to 64 (mean age = 37.4, SD = 17.1). All but two users (aged 61 and 63) had used a tablet prior to the evaluation.

4.1.2 Materials

The materials used for the session were a Samsung Galaxy Tab E tablet on which the **Tica** app ran, the robot and floor tiles, and the tangible interface components (instruction blocks, mirror holder and tablet stand). Pre- and post-test questionnaires were also developed. The pre-test contained two questions, both of which showed a picture of a task grid and a set of instructions. Participants were asked to decide whether or not the robot could complete the task with the instructions given. The post-test included these same two questions along with a further two questions of the same type (but of increasing difficulty). An additional question included in the post-test required participants to select the correct set of instructions for a given task from a set of three. Finally, participants were asked to rate their enjoyment of the session (on a scale of 1 to 5) and to describe the best and worst aspects of the session.

4.1.3. Procedure

Participants were randomly assigned to either the screen-based or tangible condition (seven per condition). They were first given a description of the session and asked to complete the pre-test. They were then asked to complete six tasks using either the tangible or screen-based interface (see Figure 14). Participants were observed as they completed the tasks, and the time taken to complete each task was recorded. Participants then completed the post-test and were debriefed.

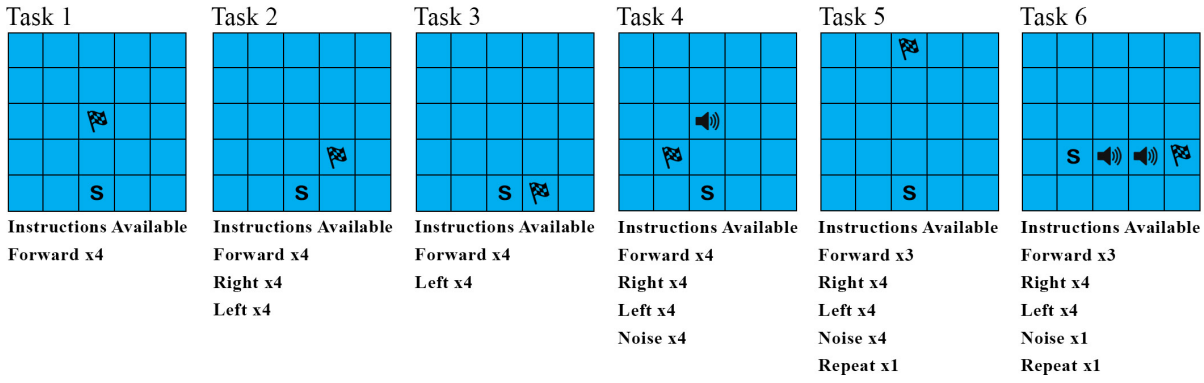

Figure 14 – Tasks for novice programmers

4.1.4. Results

For the two questions which appeared on both the pre- and post-tests, two participants incorrectly answered one of the questions on the pre-test (one from each interface condition), however all answered correctly in the post-test. Although there is insufficient data from which to draw conclusions, it suggests that learning occurred during the session (but shows no difference between the interfaces).

For the remaining three post-test questions, there were five incorrect answers in total, suggesting that the majority of participants had a good understanding of the instructions by the end of the session. Of the incorrect answers, two were from participants in the tangible condition, while three were from

participants in the screen condition, suggesting that there was little difference between interfaces in terms of program understanding.

In terms of task performance, there were very few incorrect attempts, which was not surprising since the system was designed for children. All tasks were completed within two attempts regardless of interface. Users found it easy to recognise desired and erroneous results, based on their comments about the robot's behaviour. Participants in the tangible condition would often start to correct problems before the robot had finished the current (incorrect) attempt, something that was not possible with the screen interface.

The task with the most incorrect attempts was Task 2. Several participants who had an incorrect attempt commented that they expected the robot to move laterally to the right rather than turn 90 degrees to the right, when the right instruction was initiated - this area would benefit from some further consideration. Task 4 had the second highest number of incorrect attempts which may well be due to the fact that it was the longest program, requiring a minimum of seven instructions.

Of the nine total incorrect attempts, four occurred using the tangible interface, while five occurred with the screen interface (Figure 15).

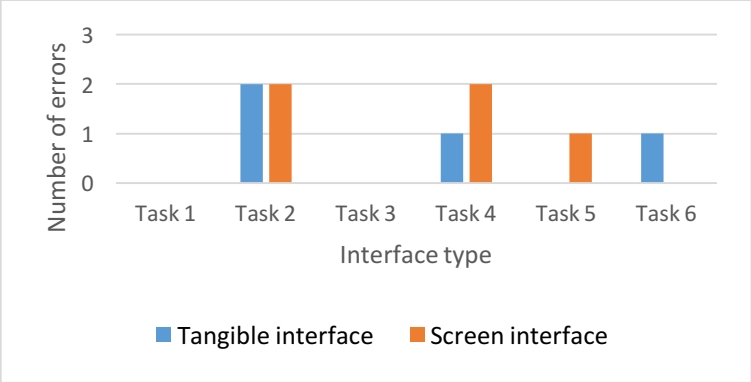

Figure 15 – Number of incorrect attempts by task and interface type

In terms of task completion time, the average time was longer in the tangible condition (mean = 26.9s, SD = 11.8s) as compared to the screen-based condition (mean = 23.6s, SD = 9.9s).

Unsurprisingly, the quickest task to complete was Task 1, which only required two instructions. The task which took the longest time to complete was Task 6 which, although it did not require the most instructions, was the most complex, involving the use of a repeat as well as three other instruction types.

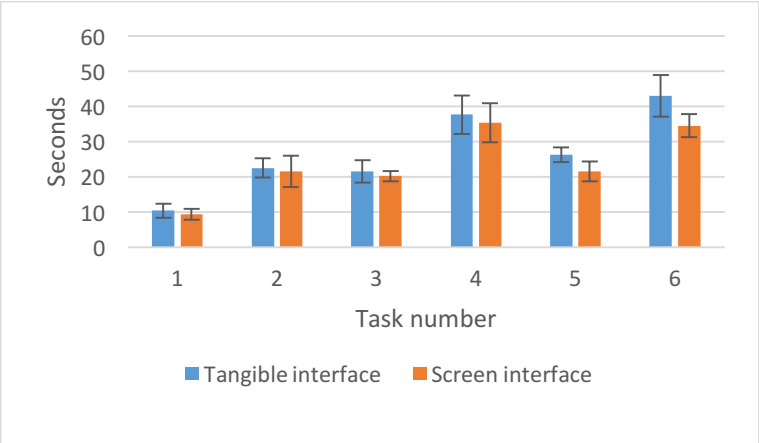

Figure 16 - Mean task duration for each interface

User-reported enjoyment was overall higher than expected, considering study participants were not the target user group. Enjoyment ratings (on a 5 point Likert scale) were higher for the tangible interface (mean = 4.25, SD = 0.76) than the screen interface (mean = 3.85, SD = 0.69). This may be in part due

to the novelty of the tangible interface. Research over a longer time period would shed light on whether enjoyment of the tangible interface is maintained over time.

4.2 Facilitator pilot study

The facilitator study was designed to explore the usability of the system and identify any improvements needed from the perspective of the facilitator user.

4.2.1 Participants

Five participants were chosen for this role, based on recommendations from Nielsen (2000). Participants were selected from personal contacts, and all were final year computer science students, as they would likely have a similar level of computer literacy to a facilitator.

4.2.2 Materials

The same materials were used as in the participant session, however the tasks revolved around setting up and controlling the system. Additionally, rather than pre-/post-tests, a single questionnaire asked participants to identify the task they found most confusing and explain why, and to identify two positive and two negative aspects of the system.

4.2.3 Procedure

Participants were first given an introduction to the study, and then asked to complete the 6 pre-defined tasks while being observed. Participants then completed the feedback questionnaire.

4.2.4 Results

Time taken to complete facilitator tasks varied (see Figure 17), however all were completed in under 60 seconds on average. Given that participants had not previously used the system, the results suggest that it was intuitive and easy to use.

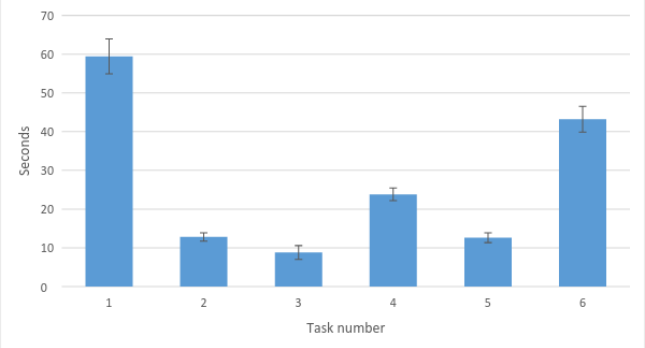

Figure 17 – Average task duration

Task 1 (connecting the robot and tangible system to the app via Bluetooth) took the longest to complete (59.4s on average), and was found to be the most confusing by all users. This was not surprising, as Task 1 required turning on two devices. Confusion over whether the robot and tangible system were turned on stemmed from the fact there is no clear indicator of this - a good point for future improvement.

5. Future work

The primary areas for future work involve, firstly, improving the robustness of the system and, secondly carrying out a full-scale evaluation of the **Tica** system.

The robustness of the **Tica** system could be improved in a number of ways, many of which revolve around the current version of the robot. Battery life is short, which restricts the length of sessions. This, coupled with the fact that the robot does not have a warning system for a low battery, can result in somewhat erratic behaviour when the battery does run low. A number of improvements could also be made to the app to improve usability and to show the status of the Bluetooth connection with the robot. The iteration instruction is currently fixed in the number of instructions it can hold, and so iterates only twice. Changing this to allow multiple iterations and a variable number of instructions would allow users to create more complex programs with more realistic control flow.

Having run a small-scale evaluation with adults, the next stage would be to run a classroom-based evaluation with children aged 5 - 7 years over multiple sessions. A comparative study of this nature

would allow us to determine the relative benefits and drawbacks of each type of interface, and to assess their impact on learning programming for children of this age group.

6. Conclusions

This paper presented **Tica**, a comprehensive environment for exploring the differences between tangible and screen-based interfaces for teaching programming to young children. Preliminary research suggests that the environment holds promise: next steps will involve researching the use of the system *in situ*.

7. References

Department for Education. (2013). Computing programmes of study: key stages 1 and 2. Retrieved May 21, 2017, from https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/239033/PRIMARY_national_curriculum_-_Computing.pdf

Horn, M. S., & Jacob, R. J. (2007). Tangible programming in the classroom with tern. In CHI'07 extended abstracts on Human factors in computing systems (pp. 1965-1970). ACM.

Horn, M. S., & Jacob, R. J. (2006). Tangible programming in the classroom: a practical approach. In CHI'06 extended abstracts on Human factors in computing systems (pp. 869-874). ACM.

Horn, M. S., Solovey, E. T., Crouser, R. J., & Jacob, R. J. (2009). Comparing the use of tangible and graphical programming languages for informal science education. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 975-984). ACM.

Howland, K., Good, J., Robertson, J., & Manches, A. (2015). Every child a coder?: research challenges for a 5-18 programming curriculum. In Proceedings of the 14th International Conference on Interaction Design and Children (pp. 470-473). ACM.

Hu, F., Zekelman, A., Horn, M., & Judd, F. (2015). Strawbies: explorations in tangible programming. In Proceedings of the 14th International Conference on Interaction Design and Children (pp. 410-413). ACM.

Nielsen, J. (2000). Why you only need to test with five users, Jakob Nielsen's Alertbox, March 19, 2000, Retrieved May 21, 2017 from http://www.useit.com/alertbox/20000319.html

Perlman, R. (1976). Using computer technology to provide a creative learning environment for preschool children. Logo memo no 24, MIT Artificial Intelligence Laboratory Publications 260, Cambridge, Massachusetts

Sapounidis, T., & Demetriadis, S. (2013). Tangible versus graphical user interfaces for robot programming: exploring cross-age children's preferences. Personal and Ubiquitous Computing, 17(8), 1775-1786.

Sapounidis, T., Demetriadis, S., & Stamelos, I. (2015). Evaluating children performance with graphical and tangible robot programming tools. Personal and Ubiquitous Computing, 19(1), 225-237.

Wyeth, P., & Purchase, H. C. (2003). Using developmental theories to inform the design of technology for children. In Proceedings of the 2003 conference on Interaction design and children (pp. 93-100). ACM.