Pers Ubiquit Comput (2015) 19:225–237
DOI 10.1007/s00779-014-0774-3

ORIGINAL ARTICLE

Evaluating children performance with graphical and tangible robot programming tools

Theodosios Sapounidis · Stavros Demetriadis · Ioannis Stamelos

Received: 27 July 2013 / Accepted: 22 April 2014 / Published online: 1 May 2014

Abstract This paper presents a cross-age study exploring children's performance on robot introductory programming activities with one tangible and one isomorphic graphical system. Both subsystems are parts of an innovative system, namely the PROTEAS kit. The tangible subsystem consists of cube-shaped blocks that represent simple and more advanced programming structures. Users may interconnect the cubic-shaped commands and so create the robot programming code. The graphical subsystem presents onscreen an isomorphic to the tangible programming space. Children ($N = 109$) of five different aged groups were let to interact in pairs with the two operationally equivalent programming subsystems with the scope to program a NXT Lego robot. Three variables associated with children performance upon tasks and four variables related with performance during free interaction were studied. Data analysis based on computer logs and video recordings showed that children produced fewer errors, made more effective debugging and younger children in particular needed less time to accomplish the robot programming tasks with the tangible subsystem. Moreover, during free interaction, elder children were more engaged, created more complicated programs and explored different commands and parameters more actively in the tangible case. Finally, interpretation of the findings is provided and the advantages of tangible user interfaces in children's introductory programming are discussed.

Keywords Computer science education · Educational technology · Educational robot · Introductory programming · Tangible programming · Tangible user interface

T. Sapounidis (✉) · S. Demetriadis · I. Stamelos
Department of Informatics Aristotle, University of Thessaloniki, Thessaloníki, Greece
e-mail: teo@auth.gr

S. Demetriadis
e-mail: sdemetri@auth.gr

I. Stamelos
e-mail: stamelos@auth.gr

1 Introduction

Since early 1960s, children programming has been a wide field of research. Drawing on Papert's constructionism ideas [1], a number of programming languages for children and novice users have been developed [2]. At the beginning, the programming tools were based on text and graphical user interfaces (GUIs) and later on tangible user interfaces (TUIs). Using keyboard or mouse, text and GUI systems allowed users to create programming structures either by typing commands or by dragging and connecting icons on a computer screen. Later, taking away keyboard and mouse, other developers based on TUIs created languages where children could acquire programming experiences by simply interacting with physical objects [3, 4]. Some of the most influential languages of this kind include AlgoBlock by Suzuki and Kato [5], Electronic blocks by Wyeth and Purchase [6] and finally, Quetzal–Tern by Horn et al. [7].

Despite the various design efforts and the theoretical frameworks [3] related to TUIs and children education, there is a lack of (a) empirical research investigating the possible advantages of the TUIs against graphical interfaces [8, 9] and (b) design guidelines and available tools that combine tangible and graphical programming capabilities [10]. Thus, this work presents a series of design

guidelines aiming to offer ideas and directions to future designers, and in addition, comparing tangible and graphical programming tools provides empirical evidence showing the possible performance advantages and disadvantages of tangible against graphical programming for children.

2 Background

TUIs in general may be considered as physical objects whose manipulation may trigger various digital effects, providing ways for innovative play and learning for children or novice [11]. Developers, taking advantage of this playfulness, created applications in fields such as quiz representation [12], storytelling [13], home automation [14] and introductory programming for children [4, 15].

Programming appears a quite challenging process for beginners of all ages [2]. Users have difficulties not only in learning a rigid syntax and commands with confusing names, but also in using the programming environment as well [16]. Reducing learning difficulties originating from the programming environment is considered to be one of the TUIs advantages, since TUI users do not have to familiarize with keyboard or mouse, while they are naturally skilled to direct manipulate physical objects, such as cubes and puzzles [17]. Consequently, it is believed that tangible systems reduce the required cognitive effort to learn how the system works, and users' attention may be exclusively devoted on the programming itself [18].

The systems that seam to have influenced the development of tangible programming are (a) AlgoBlocks [5] that was the first that introduced the cubic-shaped commands, (b) Tangible Programming Brick by McNerney [19] that was the first that clearly introduced the use of parameters, (c) Electronic Blocks [20] that let children build programmable robots or simple mechanisms by simply snapping together the block primitives and (d) Tern [21] that was the first that employed a scanning system in order to identify the connected commands and so create the program sequence through an automatic identification.

Although pioneering work has been already done in the domain of construction of such systems, there is still a lack of available tools [10] and design guidelines.

2.1 Comparison studies between graphical and tangible programming languages

A number of studies attempting to compare tangible against isomorphic graphical systems in various domains have reported controversial results, and this strongly emphasize the need for further research in order to examine the circumstance under which each type of interface offers more benefits in a real-class context [22, 23]. Particularly, in the domain of programming, there is a notable lack of such empirical works [3]. Comparative works on tangible and graphical systems with analogous features (such as similar shape, appearance, functionality) produced interesting findings, focusing primarily on fun and enjoyment [24]. Kwon et al. [10] performed a comparison between Algorithmic Bricks with Scratch [25]; however, the systems were not isomorphic. Horn et al. [26] compared a passive tangible against a similar graphical programming language in informal learning setting at the Boston Museum of science. This study revealed some advantages of the tangible programming language against the graphical. For instance, it was found that the passive tangible language was more attractive and supportive for active collaboration.

Finally, Sapounidis and Demetriadis [22] explored children's opinions regarding tangible and graphical programming. The results showed that the tangible interface was considered more attractive especially for girls, more enjoyable and finally easier to use only for younger children.

Even though tangibles are believed to be more efficient than GUIs, there is limited research that systematically explores the cognitive and social advantages of TUIs compared to GUIs [8, 27]. In particular, the impact of tangible environments and the conditions under which the handling of tangible objects can be more efficient for children or novice, in various domains, such as programming, have not been studied sufficiently and remains mainly unexplored [2, 18].

2.2 Research motivation and questions

Taking into account the limited empirical research [9, 28] on TUIs and particular in relation to quantitative outcomes [29], the present work sets a twofold objective. First, to illustrate a series of design principals in order to offer new ideas and directions to designers of such tools; second, to provide empirical evidence on the possible (dis-) advantages of TUIs (compared to isomorphic GUIs), when used for children introductory programming; The measurements and data analysis aimed to address children's performance on predetermined tasks and during free interaction. In detail, regarding performance upon tasks the research questions are:

RQ1. Do children perform significantly better with one specific interface, when task-solving time is considered?
RQ2. Do children perform significantly better with one specific interface when number of errors is considered?
RQ3. Which interface is more appropriate for efficient debugging when an error occurs?

Regarding free interaction session, the research questions are:

RQ4. When children may freely use either a GUI or a TUI programming interface, do they engage more with one specific interface?
RQ5. When children may freely use either a GUI or a TUI programming interface, do they produce longer programs with one specific interface?
RQ6. Do children explore more commands and parameters when working with one specific interface?
RQ7. When children may freely use either a GUI or a TUI programming interface, what is the complexity of the code they develop?

Answers to the above questions would provide insight into the possible TUIs versus GUIs (dis-)advantages in introductory programming for children.

3 The tools implemented

3.1 Design consideration guidelines

Although TUIs is generally believed to be easier to learn and use, it is more difficult to design and build than other traditional interfaces [30]. In designing and constructing our system, we took into account the suggestions, the advantages and the deficiencies of the previous design approaches as they appear in the related literature.

In detail the issues we addressed and took into account were:

Both tangible and graphical interface The development of a system that offers both tangible and graphical programming capabilities may give the possibility of a fluid and balanced transition between TUI and GUI, in relation to the age and user experience [7]. Furthermore, the possibility to program with two isomorphic environments gives a unique opportunity to study the advantages of tangibles in relation to GUIs (e.g., [8, 31]).

Cost and portability Because of cost and complexity issues, the construction of tangible systems for programming is considered to be resource consuming for research purposes [30], a fact that might explain the lack of evaluations in real classrooms settings [10]. To reduce the cost and simultaneously achieve high portability, we developed an active tangible programming language based on reliable microcontrollers and low cost D9 and D25 connectors. This way the programming activity can be done in any surface allowing to freely develop the activity in a more dynamic way [32].

Reduction in the physical and conceptual distance between input and output To eliminate the distinction between tangible actions and their effect, which are the results of the implemented program, both actions and effects should be presented in the same physical place [15, 32–35]. To achieve a better physical coupling between performed actions and their effects, in our approach, users program in the real environment with real cubes and inspect the outcome of their program in the same physical space with a real robot [36, 37].

Collection of commands and parameters Supporting procedural programming with an enhanced set of commands and parameters makes system usage interesting and challenging to even elder children [16]. In order to satisfy the above, our system was designed to support a plethora of commands beyond just the "move" commands and more parameters than just numbers; moreover, it allows users to familiarize with concepts such as procedure, repetition and condition [2]. Furthermore, the system introduces the functionality to save and reuse the program code which had been created by someone else [38, 39] or exchange it with other users [40].

Interaction with the user on the interface An increased user–system interaction is a desirable aspect in this kind of systems [16, 34]. For this purpose, we developed two additional functions; the first informs the user about the internal state of the system and second informs the user about any possible syntax error. Both actions occur with appropriate indications on the interface and so no external screen or other means are needed.

Physical properties and characteristics The development of a system that has properties and physical characteristics which may be useful during interaction. In our approach for the tangible subsystem, we used a combination of connectors in order to set the appropriate constraints on the users [41]. The goal is to prevent users from plugging a parameter cube in the place of a command cube or connect the blocks in the other way round. This physical property may reduce the users' cognitive load because the interface itself "informs" users for what they should not do. Finally, it seems that the inclusion of some material characteristics in the interfaces (temperature, shape, color, texture, sound, etc.) may bring the interfaces closer to the concepts and the operations they depict [42, 43]. For this reason in our tangible subsystem, cube weight has been adequately adjusted for example the parameter 4 has two times the weight of the parameter 2.

Availability To increase system's availability, the whole design was built upon common sensors and an easy to find, in many schools, Lego NXT robot. Finally, batteries are not required for the blocks and in general for the whole system, so continuous operation is assured without recharging.

Fig. 1 Father box and an indicative program with T_ProRob

Reliability To increase system reliability, serial protocols were employed and consequently the number of commands that can be connected is not limited by the I/O ports of the microcontrollers used [3]. Furthermore, the cubes have embedded intelligence and are able to carry out self-check procedures concerning various issues such as quality of power supply and connectivity with neighboring blocks. If a problem is detected, each cube tries to deal with it on its own. In any case, the user gets an indication of proper operation on the blocks.

Affordability The ability to depict a wide variety of concepts [34, 44]. To this end, by making some minor modifications on the computer software and changing the photos on the boxes, cognitive tasks can be implemented with the same graphical and tangible system [45]

3.2 PROTEAS kit

PROTEAS (PROgramming TangiblE Activity System) is an assembly including one graphical (V-ProRob) and two tangible robot programming tools (T_Butterfly) [46] and (T-ProRob). In the following, we present the tangible T-ProRob and the graphical V-ProRob subsystems which have been used in our study.

T_ProRob The T_ProRob (tangible) subsystem consists of Plexiglas command and parameter cubes. By combining the cubes, users are able program an NXT Lego robot. An indicative program structure is shown in Fig. 1.

In this program, the Lego's NXT robot will do the following: (a) two steps backwards (b) make a sound (c) then a nested loop will executed, the robot for three times will make a delay and then with the inner loop will move in a square route (d) when the nested loop has been completed, the robot will carry out a check using the light sensor. If no light is detected, the robot's lamp will turn on.

Users can perform robot moving control actions such as "move one step forward/backward," "turn left/right" and also commands such as "turn off the light" and "make a sound." Furthermore, repetition and condition programming structures are available, supporting more complicated combinations like nested repetitions and conditions. Finally, a special cube, where users can save their program code and reuse it later, as a function in other programs, completes the set of commands.

After the connection of the desired commands and parameters with the "father box," user may initiate the program execution by pressing the run button located on the top of the "father box." Then, the father box "reads" and forwards the program to a remote computer using an RS232 cable or Bluetooth. The computer records the program in a database and after compilation transmits the code for execution to the Lego NXT robot using Bluetooth. The bidirectional communication between robot and cubes allows for increased user–system interaction. While the program is executed, the robot can, for example, report to the condition command cube that the result of a measurement was positive or negative. Then, the condition command informs the user by turning on the appropriate LED on the cube. Furthermore, users are informed, in synchronous mode, of a potentially "wrong" (non-acceptable) parameter connection through a LED indication on the parameter cube.

V-ProRobL V_ProRob was designed based on T_ProRob's functionality and offers a reliable graphical isomorphic equivalent. The subsystem presents onscreen the same features and operation as T_ProRob does.

In Fig. 2, an indicative program structure with V_ProRob is depicted. The functionality of the program is identical with

Fig. 2 Indicative program with the graphical interface

the program previously presented with T_ProRob. With this graphical subsystem, users can create program sequences by arranging the available commands and parameters with a 'drag and drop' interaction technique. V_ProRob also supports bidirectional communication between robot and graphical environment. The communication is achieved using Bluetooth and allows the subsystem to provide feedback to the user over the icons of the commands and parameters, much like the tangible interface does.

4 Method

4.1 Participants

The study was conducted in a public school at the area of Thessaloniki, Greece. One hundred and nine children of five age groups, 6–7 ($N = 20$), 7–8 ($N = 25$), 9–10 ($N = 14$), 10–11 ($N = 25$) and 11–12 ($N = 25$) years participated in the study.

All children volunteered to participate as part of their every day school activities, and they were randomly assigned to work in pairs. All participating children were not familiar with the systems and spoke Greek as their native language. All children had some familiarity with the mouse. Figure 3 shows children interacting with the tangible subsystem.

4.2 Setting and procedure

Experiments were conducted in classrooms and were appropriately arranged so that the two subsystems were equally accessible for all children. Three subject-matter experts, one in the front and two in the back along with video–audio recorders and computer logs, recorded the whole process.

Fig. 3 Children interacting with tangible subsystem

Children, guided by the researcher, first filled out the questionnaires about their age, gender, familiarity with computers and computer programming knowledge. Then, the NXT Lego robot was presented, and following a simple scenario, the researcher presented to the children how they could program the robot with both systems. To rule out any possible sequence effect, we counterbalanced the presentation.

Then, two missions (*Task1* and *Task 2*) were assigned to children, while a third mission (*Task3*) was assigned to the elder children. The first mission (*Task1*) was a simple sequential program up to six commands which involved cubes such as "move forward/backward," "turn on/off the light" and "make a sound." The second mission (*Task2*) was a more advanced sequential program with parameters; it was up to six commands and involved cubes such as "move forward/backward," "turn right/left" "turn on/off

the light" and "make a sound." The third mission (*Task3*) was the most complex; it was up to five commands and in addition it involved the repetition (loop) structure.

Children were asked to accomplish the programming missions using one interface (selected so that half of the pairs started programming using the graphical interface and the other half using the tangible); subsequently, children had to accomplish missions of the same difficulty using the other interface.

After the interaction process with the tasks, the researcher let children free to make two more programs using each system successively. During this free session, children created programs with out any predetermined task scenario or time limit.

4.3 Measurements on tasks

The data collection procedure employed video–audio recordings and system databases logs. Video–audio recordings were analyzed by three experts (each one was responsible for one measured variable). By transcribing the video–audio recordings and database logs, we measure the three variables referring to children's programming performance for the two systems, namely (1) time to accomplish tasks (*TAT,* the time needed to accomplish the programming tasks the soonest possible), (2) errors (*ER*, the number of erroneously executed programs) and (3) the debugging stages (*DS,* the debugging stages reached after errors).

For assessing the debugging stages the Carver and Klahr model [47] was used and a three-level ordinal variable was created. Using the available video–audio recordings and computer logs, we determine whether the children after a wrong execution (a) arrived at the correction of the bug (full debug), (b) noticed that a specific error occurred, but they were unable to locate and correct it (partially debug), (c) did not notice any discrepancy between the goal and the actual outcome and no debugging took place (no debug).

The above-dependent variables were examined as a function of gender and age.

4.4 Measurements on free interaction

By transcribing the video–audio recording and computer logs, we measure the four variables referring to children's free programming performance for the two systems. These variables are (4) *free interaction time-engagement* (i.e., the time children spent to freely interact with the systems in order to create their programs), (5) *program length*, (6) *program vocabulary* and (7) *program complexity*.

For assessing the *program length* and *vocabulary,* we used the Halstead software metrics [48]. In our case, according to Halstead, the *program length* is the total number of commands and parameters, while *vocabulary* is the total number of unique commands and parameters within a program. Furthermore, in order to examine the *program complexity*, the logical structure of the program was analyzed by using the McCabe cyclomatic complexity measure [49]. All the above metrics are independent of the programming language itself [50]. To be fully unbiased, during our analysis, we set the maximum length of the programs for both interfaces to the maximum length of the program a user can see on a computer screen.

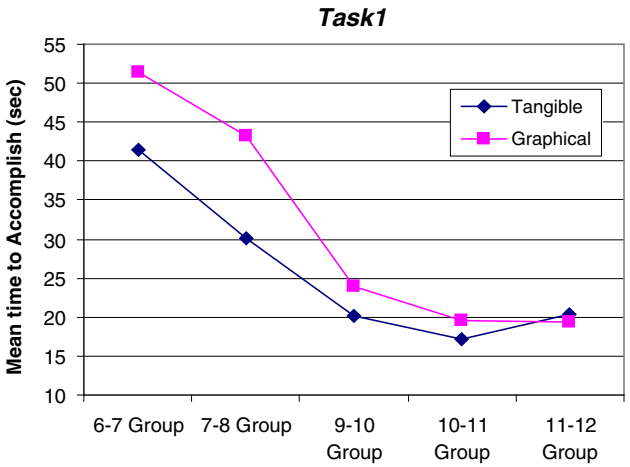

Fig. 4 The mean time children needed to accomplish Task1 as a function of group age for both interfaces

5 Results

5.1 Task performance analysis

5.1.1 Task1

The time to accomplish Task1 (*TAT1*) had a negative Pearson's correlation with age for both interfaces, $r = -0.63$, ($p < 0.001$) and $r = -0.68$ ($p < 0.001$) for the tangible and the graphical, respectively. These relations are also depicted in Fig. 4, which shows the mean time children needed to accomplish Task1 with both interfaces as a function of the group age. A decrease in *TAT1* is observed as the group age increases, and it reaches a plateau after 10–11 years age group.

Focusing on the differences in *TAT1* between the two interfaces, for the same age group, a *t* test and a nonparametric Wilcoxon Signed Rank Test were used. Both showed that in the case of the tangible interface, *TAT1* measurements were significantly lower for age group 6–7 years ($p = 0.049$), age group 7–8 years ($p = 0.030$) and age group 9–10 years ($p = 0.012$), while no statistical differences were observed for age groups 10–11 years and 11–12 years.

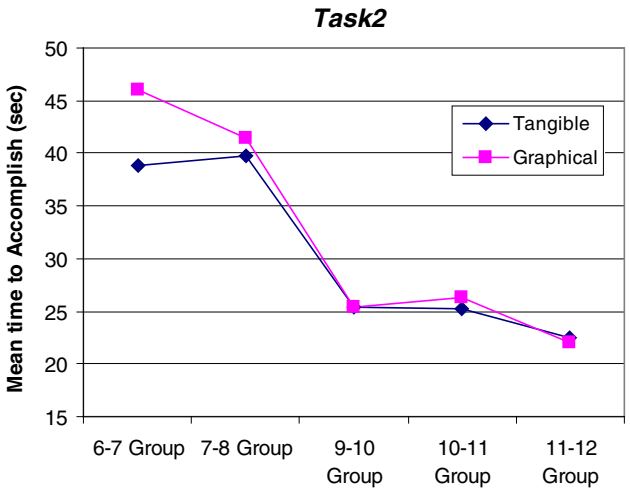

Fig. 5 The mean time children needed to accomplish Task2 as a function of group age for both interfaces

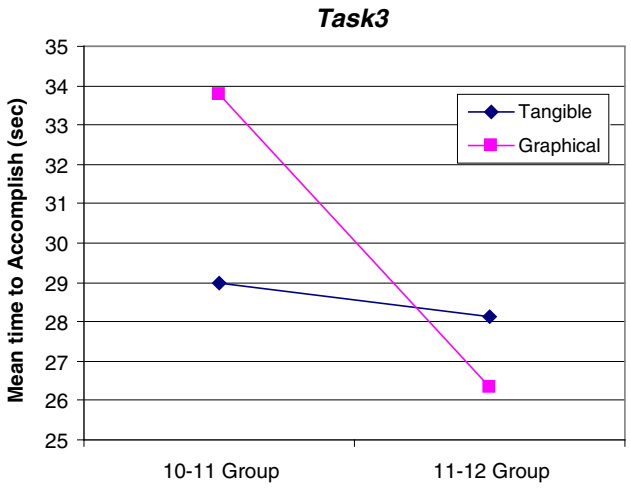

Fig. 6 The mean time children needed to accomplish Task3 as a function of group age for both interfaces

5.1.2 *Task2*

The time to accomplish Task2 (*TAT2*) had a negative Pearson's correlation with age for both interfaces, $r = -0.42$, ($p < 0.001$) and $r = -0.44$ ($p < 0.001$) for the tangible and the graphical, respectively.

These relations are also depicted in Fig. 5, which shows the mean time children needed to accomplish *Task2* as a function of group age. A decrease in time is observed as the group age increases, and it reaches a plateau after 9–10 years age group.

Focusing on the differences in *TAT2* between the two interfaces, for the same age group, both a *t* test and a nonparametric Wilcoxon Signed Rank Test showed that no statistical differences exist for all age groups.

Fig. 7 The percentages of erroneous tasks in each interface

5.1.3 *Task3*

In *Task 3*, which was the most demanding and difficult, only age groups of 10–11 years ($N = 7$) and 11–12 years ($N = 17$) participated.

Figure 6 shows the difference between tangible and graphical cases for the two age groups. A *t* test with bootstrap ($p = 0.001$) and a nonparametric Wilcoxon Signed Rank test ($p = 0.038$) showed that the *TAT3* in tangible interface was statistically lower than the graphical case for the 10–11 years age group, but not for the 11–12 years age group.

5.1.4 *Errors*

Figure 7 shows the percentages of erroneous tasks in each interface. In all cases, more errors occurred with the graphical interface. Examining the differences, we found that only in *task 2,* statistical significance exists ($\chi 2 = 3.96$, $p < 0.05$).

5.1.5 *Debugging*

Focusing on the error correction process, the debugging variable was analyzed after an unsuccessful execution in both interface settings. The two distributions are presented in Fig. 8, which shows a significant difference between them ($\chi^2 = 8{,}79$, $df = 2$, $p < 0.05$). Thus, for the TUI case, it is more likely that the errors are fully corrected, while for the GUI case, the errors are more likely to be overlooked.

Concerning gender, non-significant effects were found for all dependent variables.

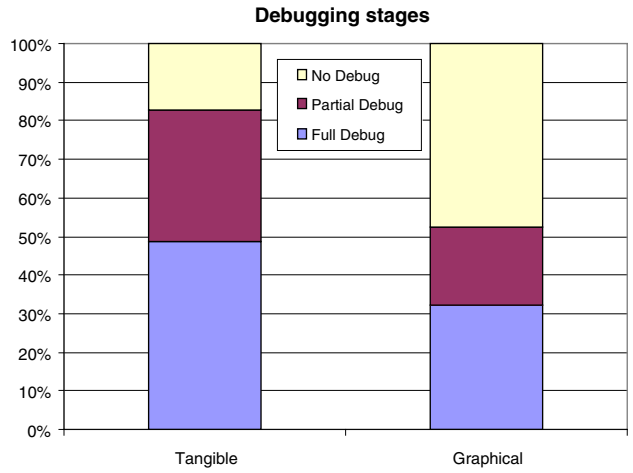

Fig. 8 Distributions of errors for both interface settings

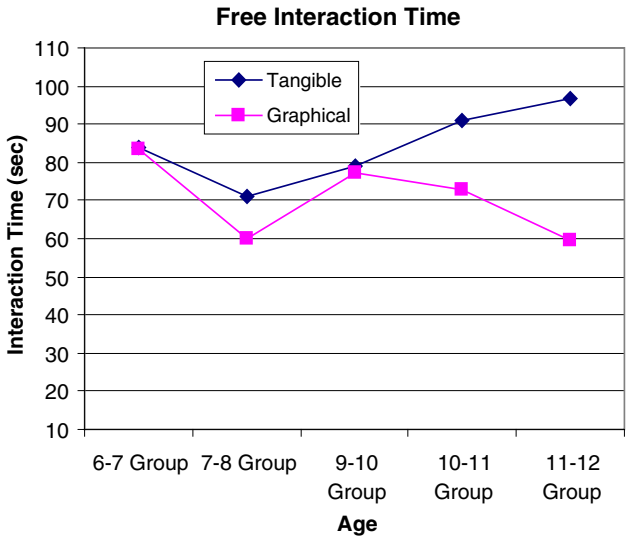

Fig. 9 The mean free interaction time–engagement as a function of age

5.2 Free interaction analysis

5.2.1 Interaction time-engagement

If *free interaction time-engagement* is considered, no correlation with age exist. To focus on the differences between the two interfaces, a *t* test and a nonparametric Wilcoxon Signed Rank Test were used.

Both showed that in the case of the graphical interface, the *free interaction time-engagement* measurements were significantly lower for age group 10–11 years ($p = 0.002$) and 11–12 years ($p = 0.000$), while no statistical differences were observed for the other age groups. The effects are represented in Fig. 9 and show the interaction mean time, for free programming activities, as a function of the age group.

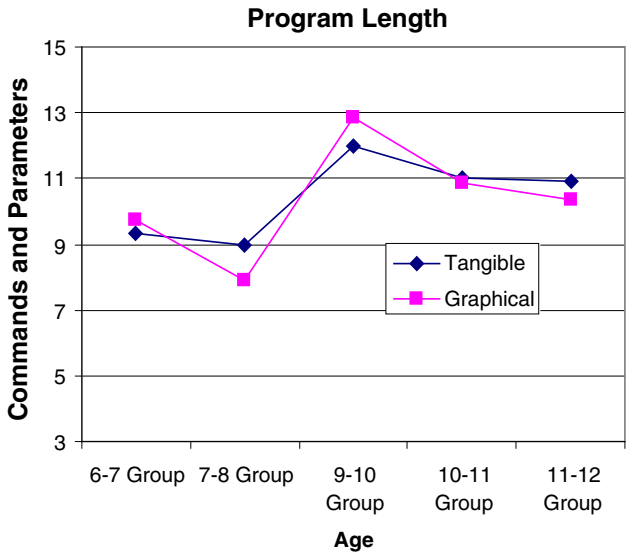

Fig. 10 The mean program length during free interaction

5.2.2 Program length

The *program length* had a positive Pearson's correlation with age for both interfaces, $r = 0.276$, ($p < 0.01$) and $r = 0.220$ ($p < 0.05$) for the tangible and the graphical interface, respectively. Studying the *program length* for the two interfaces during free interaction, no significant difference exists. The program length in accordance with age is presented in Fig. 10.

5.2.3 Program vocabulary

The *program vocabulary* had a positive Pearson's correlation with age for both interfaces, $r = 0.417$, ($p < 0.001$) and $r = 0.402$ ($p < 0.001$) for the tangible and the graphical interface, respectively.

Focusing on the differences in *program vocabulary* between the two interfaces, for the same age group, a *t* test and a nonparametric Wilcoxon Signed Rank Test were implemented. Both showed that in the case of the tangible interface, the *program vocabulary* measurements were significantly higher for age groups 6–7 years ($p = 0.041$), 10–11 years ($p = 0.036$) and 11–12 years ($p = 0.007$), while no statistical differences were observed for age groups 7–8 years and 9–10 years. The *program vocabulary* variable in relation with age is depicted graphically in Fig. 11.

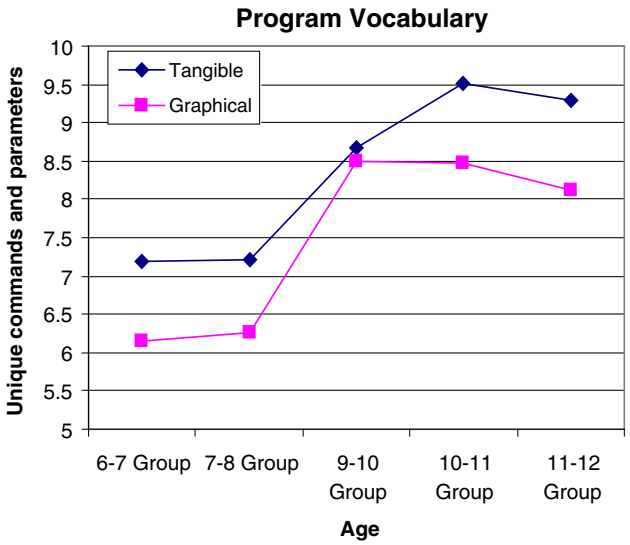

Fig. 11 Program vocabulary as a function of age

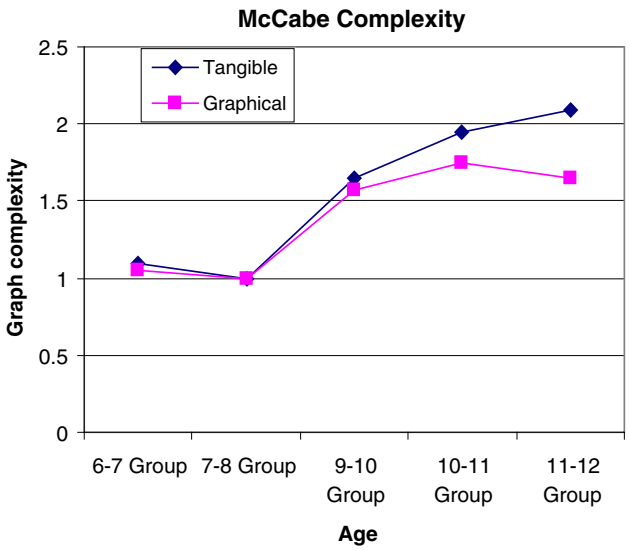

Fig. 12 The percentages of erroneous tasks in each interface

5.2.4 *Program complexity*

Analyzing the internal *complexity* of the program created during the free interaction, a positive Pearson's correlation with age for both interfaces, $r = 0.673$, ($p < 0.001$) and $r = 0.595$ ($p < 0.001$) for the tangible and the graphical subsystem, respectively, exists.

Focusing on the differences in *program complexity* between the two interfaces, for the same age group, a *t* test and a nonparametric Wilcoxon Signed Rank Test were implemented. Both showed that in the case of the tangible interface, the *complexity* measurements were significantly higher for age groups 10–11 years ($p = 0.017$) and 11–12 years ($p = 0.001$), while no statistical differences were observed for the other age groups. Figure 12 represents the complexity of the free program as a function of age.

6 Discussion

In this paper, we presented a series of design guidelines for tangible programming tools to inform design, and using PROTEAS kit, we carried out a comparative study between two isomorphic interfaces. The first assigned group of tasks (*Task*1) was chosen on purpose to be easily accomplished by the children. The second group of tasks (*Task2*) was more challenging, and for this reason, a number of programming errors occurred. The third group of tasks, the most difficult one (*Task3*), was assigned only to some elder children, and thus, we managed to further focus our study on these particular age groups. Finally, children were let freely to interact with both systems in order to create one to three programs with each condition. Accumulated programs of this free interaction session were analyzed using simple software quality metrics.

6.1 Time to accomplish the tasks

Regarding the time to accomplish the tasks, *TAT* decreases with the age in all tasks for both the tangible and graphical interface. This effect is anticipated considering the developed skills and intelligence of elder children. Focusing on the comparison between the two interfaces, *TAT1* for the tangible subsystem was significantly lower especially for younger children. This might be interpreted by considering the differences in children's familiarity with similar tangible games such as LEGO bricks and puzzles, and computer use. In elder children case, the experience with computers is increased and thus the *TAT1* differences between the two interfaces are expected to be diminished. Finally, with no statistical significance, the *TAT1* for graphical interface becomes even lower at the age of 11–12. This trend is evident in all tasks and might show that this particular age is the threshold where children reach the crucial computer skills that make them to further reduce their *TAT* with the GUI. To further investigate this effect, we implemented *Task3* engaging the two elder age groups. Although the differences, as we expected, became clearer, significance was reached only for the 10–11 years age group and not for the 11–12 years group.

In general, the present findings based on time to complete a task are consistent with the results reported in [22] and simultaneously extent the results by Xie et al. [8] who reported based on children's (aged 7–9 years old) self-report that it was easier for them to interact with physical

pieces than with the mouse. Summarizing, one may say that the familiarity with tangibles, and on the other hand, the accumulated experiences with computers are most probably the factors explaining our findings [22].

6.2 Errors

Regarding programming errors, children made more mistakes with the graphical interface in all tasks. However, in *Task2*, which was of moderate difficulty, the number of errors occurred in tangible interface was significantly lower. This significant difference was not observed in the other two cases. In the easiest *Task1*, a negligible number of errors occurred, while in the most difficult *Task3*, the statistically insignificant difference might be due the small number of children engaged.

The decreased number of errors while programming the robot with the tangible subsystem might be due to active participation and motivation. It is reported that children more actively participate in programming activities with tangibles than with graphical interfaces [7]. Several studies have shown that working together on specific collaborative tasks may increase children' enjoyment, engagement and motivation (e.g., [51, 52]). In the case of tangibles, since both participants can be simultaneously active, motivation is increased, contrary to the graphical case, where the one child is more likely to be a passive and maybe less motivated spectator [53].

6.3 Debugging

Children's interaction with tangibles appeared to facilitate debugging after an error occurrence. Using the tangible interface makes it more likely for a full debugging process and correction of the error to take place, whereas with the graphical one, it is more likely to be neglected. This finding might be attributed again to the active involvement of both group members in the case of tangibles, while in the case of graphical interface, one member uses the mouse and programs, as active participant, whereas the other is more likely to be a passive observer. Working with tangibles both users are equally participating [54, 55], so maybe they feel equally responsible to inspect the executed program. Furthermore, with tangibles, members can increase their visibility of the working plane [13, 56], and consequently both members can more easily detect the possible inconsistencies.

6.4 Free interaction time-engagement

Free interaction time, with a system, appears to be well correlated and so is usually interpreted as an indication of engagement [8, 26]. This measure in our case does not appear to be correlated with age, for both interfaces. Interestingly, focusing on the differences between the two interfaces, the elder groups (aged 10–11, 11–12 years) showed higher engagement with the tangible subsystem. Our results may support the results of our previous research [22] which showed, based on qualitative and quantitative analysis, that especially elder children were highly motivated and enjoyed tangible programming the most. The result of this measurement is raising questions related to the nature of the programming code that the children produced during their free interaction time, especially in the tangible case when children spent more time.

6.5 Program length

In the literature, one measurement of the programming quality appears to be the *program length*. This particular measurement in our case shows that, while growing, children create bigger programs. This is expected due to the higher mental skills developed by the elder children. If the *program length* is examined in relation to the tangible and graphical subsystems, no differences between the two cases appear to exist. This effect is fully in accordance with the results reported by Horn [26] who measured and compared the number of commands in programs created with a passive tangible and a graphical programming language in informal learning environment.

6.6 Program vocabulary

In terms of the different commands and parameters used within a program, the results interestingly denote that children in general preferred to explore the different capabilities available mostly in the tangible case. This particular trend is statistically significant for the very young and elder children (6–7, 10–11, 11–12 years of age).

The results for this dependant variable may partially give an answer to the question "what children did during their free interaction time?" Elder children (aged 10–11, 11–12 years) spent more time in the tangible case (Fig. 9), and it seems that they explored more alternative commands and parameters. On the contrary, younger children spent almost the same time in both cases, but the outcome appears to be different for the two cases. A possible explanation for the younger children can be found with a closer look at the *TAT1* variable (time to accomplish task1). Younger children are able to manipulate tangibles in less time in order to produce the same program with the graphical. Consequently, spending the same time for both the tangible and the graphical interfaces means that they have more time to explore the possible alternatives with the tangible subsystem. This particular result is in accordance with the results reported by Schneider et al. [29] who

reported that pairs using tangible interface were more explorative on alternative logic designs.

6.7 Program complexity

By measuring the number of linearly independent paths of the program source code, we measure the McCabe *program complexity* metric. This particular software metric is sensitive on the repetition and conditional programming structures. Examining the results depicted on Fig. 12, we may assume that younger children did not used enough repetition and conditional structures in their programs with both tangible and graphical subsystem. Maybe children of this age preferred to use simple structures or maybe they were not ready to use appropriately repetition and conditional structures. On the contrary, elder children (aged 10–11, 11–12 years) were able to use these complicated structures with both interfaces, but the programming outcome was significantly more complicated in the tangible case. This particular outcome may provide an additional answer to the question of what kind of programming code the children produced during this free interaction time. Elder Children not only spent more time in the tangible case (Fig. 9), but were more explorative, as explained before, and this exploration together with the extra time spent with the tangible subsystem lead them to produce more complicated programs. Our results are coherent with the results reported by Schneider et al. [29] who reported that tangibility trends to increase exploration and consequently enhance performance in simple logic tasks.

7 Conclusion and future research

In this paper, we carried out and presented a comparison study of children's performance using the two isomorphic subsystems (TUI vs. GUI).

Data analysis upon task measurements showed that younger children needed less time to accomplish the programming tasks when using the tangible interface. On the contrary, elder children, who were more experienced computer users, needed almost the same time to accomplish the tasks with both interfaces. Furthermore, fewer programming errors occurred and better debugging was achieved in the tangible case.

Moreover, data analysis on free interaction showed that elder children were more engaged, explored more alternative commands and parameters and created more complicated programs with the tangible subsystem. On the contrary, younger children spent almost the same time in both cases, but it seems that they were more explorative with the tangibles.

The above findings could be explained by considering both children's familiarity with games that resemble tangibles and prior computer experience. Moreover, we have proposed that additional factors, such as the ability to better see, collaborate and actively participate have also implicitly affected our research results.

Regarding our initial design guidelines to have both TUI and GUI interfaces, our findings also support that a balanced transaction between the two alternative interfaces in relation to user experience and weather children interact upon tasks or freely may be beneficial for certain users. Our initial goal regarding portability and availability was proved also crucial to implement our experimentation, which took place in many classrooms and surface for a long continuous time. Additionally, our finding regarding complexity program length and program vocabulary highlighted the need to have a plethora and adequate collection of commands and parameters. In any case, more effort and focused research is needed to evaluate how the findings impact design.

In conclusion, the results supporting TUIs in introductory programming encourage further investigations on learning and performance and possible training applications, such as with adult novices or individuals with special needs. In addition, advanced modern programming techniques, such as pair programming or object-oriented programming, may be combined with tangibility and researched to provide additional insight to early programming.

References

1. Papert S (1980) Mindstorms: children, computers, and powerful ideas. Basic Books Inc., New York
2. Kelleher C, Pausch R (2005) Lowering the barriers to programming: a taxonomy of programming environments and languages for novice programmers. ACM Comput Surv 37(2):83–137
3. Orit S, Eva H (2009) Tangible user interfaces: past, present, and future directions foundations and Trends®. Hum–Comput Interact 3(1–2):1–137
4. Sapounidis T, Demetriadis S (2009) Tangible programming interfaces: a literature review. In: Proceedings of the 4th Balkan conference in informatics, Thessaloniki, Greece, pp 70–75
5. Suzuki H, Kato H (1993) AlgoBlock: a tangible programming language, a tool for collaborative learning. In: Proceedings of the 4th European logo conference, pp 297–303
6. Wyeth P, Purchase H (2002) Designing technology for children: moving from the computer into the physical world with electronic blocks. Inform Technol Child Educ Ann 2002(1):219–244
7. Horn M, Crouser R, Bers M (2011) Tangible interaction and learning: the case for a hybrid approach. Pers Ubiquit Comput 16(4):379–389
8. Xie L, Antle AN, Motamedi N (2008) Are tangibles more fun? Comparing children's enjoyment and engagement using physical, graphical and tangible user interfaces. In: Proceedings of the 2nd

international conference on tangible and embedded interaction, Bonn, pp 191–198
9. Zaman B, Vanden Abeele V, Markopoulos P, Marshall P (2012) Editorial: the evolving field of tangible interaction for children: the challenge of empirical validation. Pers Ubiquit Comput 16(4):367–378
10. Kwon D, Kim H, Shim J, Lee W (2012) Algorithmic bricks: a tangible robot programming tool for elementary school students. IEEE Trans Educ 55(4):474–479
11. Price S, Rogers Y, Scaife M, Stanton D, Neale H (2003) Using 'tangibles' to promote novel forms of playful learning. Interact Comput 15(2):169–185
12. Terrenghi L, Kranz M, Holleis P, Schmidt A (2006) A cube to learn: a tangible user interface for the design of a learning appliance. Pers Ubiquit Comput 10(2):153–158
13. Stanton D, Bayon V, Neale H, Ghali A, Benford S, Cobb S, Ingram R, O'Malley C, Wilson J, Pridmore T (2001) Classroom collaboration in the design of tangible interfaces for storytelling. In: Proceedings of the CHI01 SIGCHI conference on human factors in computing systems, Seattle, WA, pp 482–489
14. Blackwell A, Hague R (2001) AutoHAN: an architecture for programming the home. In: IEEE symposia on human–centric computing languages and environments, pp 150–157
15. McNerney TS (2004) From turtles to tangible programming bricks: explorations in physical language design. Pers Ubiquit Comput 8(5):326–337
16. Cockburn A, Bryant A (1997) A Leogo: an equal opportunity user interface for programming. J Visual Lang Comput 8(5–6):601–619
17. Smith A (2007) Using magnets in physical blocks that behave as programming objects. In: Proceedings of the 1st international conference on tangible and embedded interaction, New York, NY, USA, pp 147–150
18. Marshall P (2007) Do tangible interfaces enhance learning? In: Proceedings of the 1st international conference on tangible and embedded interaction, Baton Rouge, Louisiana, pp 163–170
19. McNerney T (2001) Tangible computation bricks: building-blocks for physical microworlds. In: Proceedings of the CHI01, ACM Press
20. Wyeth P, Purchase H (2003) Using developmental theories to inform the design of technology for children. In: Conference on interaction design and children, New York, NY, USA, pp 93–100
21. Horn M, Jacob RJK (2007) Tangible programming in the classroom with tern. In: CHI '07 extended abstracts on human factors in computing, San Jose, CA, USA, pp 1965–1970
22. Sapounidis T, Demetriadis S (2013) Tangible versus graphical user interfaces for robot programming: exploring cross-age children's preferences. Pers Ubiquit Comput. doi:10.1007/s00779-013-0641-7
23. Zuckerman O, Gal-Oz A (2013) To TUI or not to TUI: evaluating performance and preference in tangible vs graphical user interfaces. Int J Hum–Comput St 71(7–8):803–820
24. Sapounidis T, Demetriadis S (2012) Exploring children preferences regarding tangible and graphical tools for introductory programming: evaluating the PROTEAS kit. In: 12th International conference on advanced learning technologies (ICALT), Rome, Italy, pp 316–320
25. Maloney J, Resnick M, Rusk N, Silverman B, Eastmond E (2010) The scratch programming language and environment. Trans Comput Educ 10(4):1–15
26. Horn MS, Solovey ET, Crouser RJ, Jacob RJK (2009) Comparing the use of tangible and graphical programming languages for informal science education. In: Proceedings of the 27th international conference on human factors in computing systems, Boston, pp 975–984
27. Antle AN (2007) Designing tangibles for children: what designers need to know. In: Proceedings of the CHI'07 extended abstracts on human factors in computing systems, San Jose, CA, USA, pp 2243–2248
28. Zaman B, Abeele Vanden V, Markopoulos P, Marshall P (2009) Tangibles for children: the challenges. In: 27th International conference extended abstracts on human factors in computing systems, Boston, USA, pp 4729–4732
29. Schneider B, Jermann P, Zufferey G, Dillenbourg P (2011) Benefits of a tangible interface for collaborative learning and interaction. IEEE Trans Learn Technol 4(3):222–232
30. Shaer O, Jacob RJK (2009) A specification paradigm for the design and implementation of tangible user interfaces. ACM Trans Comput–Hum Interact 16(4)20:1–20:39
31. Sylla C, Branco P, Coutinho C, Coquet E (2012) TUIs vs GUIs: comparing the learning potential with preschoolers. Pers Ubiquit Comput 16(4):421–432
32. Fernaeus Y, Tholander J (2006) Finding design qualities in a tangible programming space. In: CHI '06 Proceedings of the SIGCHI conference on human factors in computing systems, Montreal, Canada, pp 447–456
33. Kitamura Y, Itoh Y, Masaki T, Kishino F (2000) ActiveCube: a bi-directional user interface using cubes. In: Proceedings of the fourth international conference on knowledge-based intelligent engineering systems and allied technologies, Brighton, UK, pp 99–102
34. Zuckerman O, Arida S, Resnick M (2005) Extending tangible interfaces for education: digital montessori-inspired manipulatives. In: Proceedings of the SIGCHI conference on human factors in computing systems, Portland, OR, USA, pp 859–868
35. Rekimoto J, Ullmer B, Oba H (2001) DataTiles: a modular platform for mixed physical and graphical interactions. In: CHI '01 Proceedings of the SIGCHI conference on human factors in computing systems, Seattle, WA, USA, pp 269–276
36. Patten J, Griffith L, Ishii H (2000) A tangible interface for controlling robotic toys. In: CHI'00 conference on human factors in computing systems, Hague, The Netherlands, pp 277–278
37. Cockburn A, Bryant A (1996) Do it this way: equal opportunity programming for kids. In: Proceedings of the sixth australian conference on computer–human interaction, Hamilton, New Zealand, pp 246–251
38. Kahn K (1996) Drawings on napkins, video-game animation, and other ways to program computers. Com ACM 39(8):49–59
39. Wyeth P, Purchase H (2002) Tangible programming elements for young children. In: CHI'02 extended abstracts on human factors in computing systems, Minneapolis, MN, USA, pp 774–775
40. Frei P, Su V, Mikhak B, Ishii H (2000) Curlybot: designing a new class of computational toys. In: Proceedings of the SIGCHI conference on human factors in computing systems, Hague, The Netherlands, pp 129–136
41. Ullmer B, Ishii H, Jacob RJK (2005) Token constraint systems for tangible interaction with digital information. ACM T Comput–Hum Int (TOCHI) 12(1):81–118
42. Blackwell A (2003) Cognitive dimensions of tangible programming languages. In: Proceedings of the first joint conference of the empirical assessment in software engineering and psychology of programming interest groups, Keele, UK, pp 391–405
43. Fishkin KP (2004) A taxonomy for and analysis of tangible interfaces. Pers Ubiquitous Comput 8(5):347–358
44. Zuckerman O, Resnick M (2003) A physical interface for system dynamics simulation. In: CHI '03 extended abstracts on human factors in computing systems, FL, USA, pp 810–811
45. Sharlin E, Itoh Y, Watson B, Kitamura Y, Sutphen S, Liu L, Kishino F (2004) Spatial tangible user interfaces for cognitive assessment and training. Biol Inspir Approaches Adv Inf Technol 3141:137–152

46. Sapounidis T, Demetriadis S (2011) Touch your program with hands: qualities in tangible programming tools for novice. In: Proceedings of the 15th Panhellenic conference on informatics (PCI), Kastoria, Greece, pp 363–367
47. Carver S, Klahr D (1986) Assessing children's LOGO debugging skills with a formal model. J Educ Comput Res 2(4):487–525
48. Halstead M (1977) Elements of software science (operating and programming systems series). Elsevier, New York
49. McCabe T (1976) A complexity measure. IEEE Trans Softw Eng 2(4):308–320
50. Sheng Y, Shijie Z (2010) A survey on metric of software complexity. In: Proceedings of the 2nd IEEE international conference information management and engineering (ICIME), Chengdu, China, pp 352–356
51. Scott SD, Mandryk RL, Inkpen KM (2003) Understanding children's collaborative interactions in shared environments. J Comput Assist Learn 19(2):220–228
52. Inkpen K, Booth K S, Gribble S D, Klawe M (1995) Give and take: children collaborating on one computer. In: CHI '95 conference companion on human factors in computing systems, Denver, CO, USA, pp 258–259
53. Stamovlasis D, Dimos A, Tsaparlis G (2006) A study of group interaction processes in learning lower secondary physics. J Res Sci Teach 43(6):556–576
54. Rogers Y, Lim Y, Hazlewood R, Marshall P (2009) Equal opportunities: do shareable interfaces promote more group participation than single user displays? Hum–Comput Int 24(1–2): 79–116
55. Falcão TP, Price S (2009) What have you done! The role of interference in tangible environments for supporting collaborative learning. In: Proceedings of the 9th international conference on computer supported collaborative learning, Rhodes, Greece, pp 325–334
56. Klemmer SR, Hartmann B, Takayama L (2006) How bodies matter: five themes for interaction design. In: Proceedings of the 6th conference on designing interactive systems, University Park, PA, USA, pp 140–149