Code Notes: Designing A Low-Cost Tangible Coding Tool For/With Children

Alpay Sabuncuoğlu
Koç University - Department of Computer Engineering
asabuncuoglu13@ku.edu.tr

Merve Erkaya
Koç University – Arçelik Research Center for Creative Industries (KUAR)
meerkaya@ku.edu.tr

Oğuz Turan Buruk
Koç University – Arçelik Research Center for Creative Industries (KUAR)
oburuk@ku.edu.tr
& Gamification Group, Laboratory of Pervasive Computing, Tampere University of Technology, Tampere/Finland

Tilbe Göksun
Koç University - Department of Psychology
tgoksun@ku.edu.tr

IDC '18, June 19–22, 2018, Trondheim, Norway

ACM ISBN 978-1-4503-5152-2/18/06 $15.00
https://doi.org/10.1145/3202185.3210791

Abstract
Programming has become an essential subject for today's education curriculum and as a result, the importance of creating the right environments to teach is increasing. For such environments, featuring tangible tools enhances creativity and collaboration. However, due to their high prices, current tangible tools are not reachable by most of the students. We developed Code Notes as a low-cost, attainable and tangible tool aimed to motivate children to support programming education. Code Notes is comprised of an Android app and code-cardboards to teach the basic concepts in programming. We continue to develop the platform with insights gained from children. This paper shares the design phases of Code Notes and observations from our two-month programming project. We also presented some future concepts of Code Notes that offer an active and embodied interaction with the teaching material.

Author Keywords
Affordable systems for education, Collaborative learning environments; Mobile learning; Tangible blocks.

ACM Classification Keywords
K.3.1. Computer uses in education: Collaborative learning; H.5.2. User Interfaces: Interaction styles

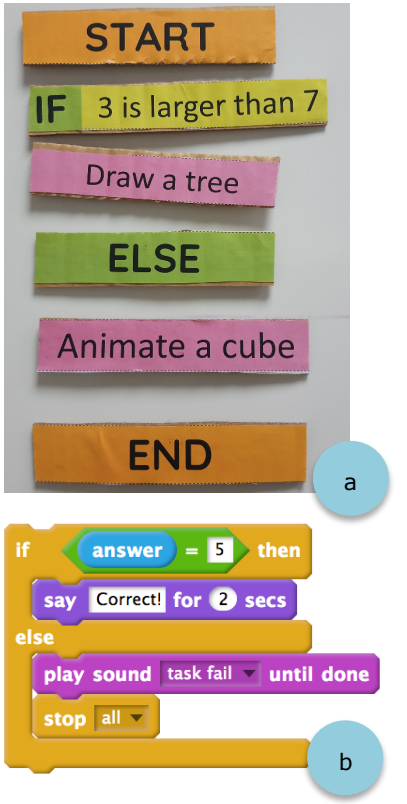


Figure 1: Different programs with If/Else structures. Programming languages: (a) Code Notes, (b) Scratch, (c) Python

Introduction

Programming has become an essential skill to be practiced for today's education. With that consideration, building learning environments to create computational thinking abilities gains importance. As a method, using tangible user interfaces (TUI) increases the playfulness and helps students to channel their mental effort on the current activity. As Zuckerman stated, computationally enhanced tangible tools encourage learning of abstract structures via hands on modelling [1]. Lego Mindstorms and Google Blocks are well known examples of these kind of learning environments. Although these are good for fostering creativity and collaboration, they are usually costly due to the electronic and mechanical components in the products. We believe that a low-cost tool is needed to reach children worldwide. Some financially reachable examples are Scratch, yet it is not a tangible programming platform, but uses the tangibility at the system design. In addition, Code Bits [2] and Tern [3] are inexpensive portable tools to boost computational thinking abilities. All of these platforms aim to build an active learning environment to teach computational skills for every child in the world.

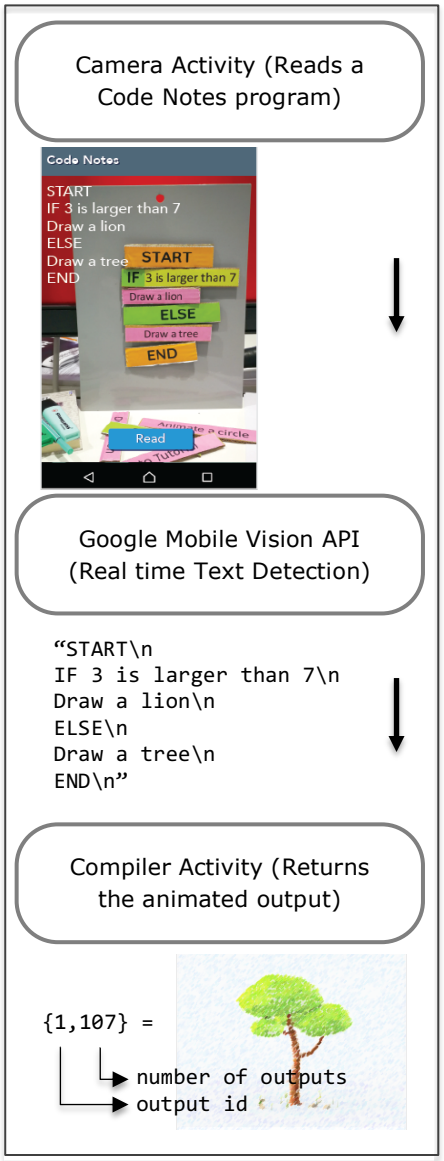
To that end, we have developed an open-source tangible programming tool, Code Notes, which is comprised of an Android app and programming cardboards. These cardboards, as seen in Figure 1a, correspond to blocks that can be combined to create different kinds of algorithms. The programming phrases are printable cards shared online, therefore are low-cost and accessible. To begin coding, learners can vertically order the printed and pasted cardboards to create meaningful algorithms. When the ordering is done, learners scan these cardboards by the real-time camera compiler, included in the Android app. Upon scanning, Code Notes App recognizes the blocks and compiles them into an action that can trigger different events (i.e., drawing a tree, animating a cube). The entire project is also made accessible on GitHub and explained in Oppia, the open source tutorial platform.

We used Code Notes in our two-month project to understand if children enjoy it and whether if this tool can be an introductory educational material prior to teaching a text-based language like Python. At the design process, we frequently received feedback from children and promptly developed some design ideas. During this process, we released two versions of Code Notes. In this paper, we presented the design process of these two releases, specifications of Code Notes language, curriculum and application. Furthermore, we discussed our preliminary observations obtained from children and shared insight for further studies.

Design Chronology

Before our two-month introductory programming course with 12- to 13-year-old students, we started the design process and then developed Code Notes as a supportive element to warm up with hands on modelling before the lectures. The platform is used to comprehend the basic concepts of programming before teaching them with Scratch or Python. For example, in Figure 1a, 1b, and 1c, different if/else structures can be seen, which are built with Code Notes, Scratch and Python. While covering if/else structure on lecture, we followed the steps: (1) telling how the structure works, (2) a quick demo with Code Notes, (3) creating a Scratch game, and (4) building a Python program.

Working with three different types of programming languages (text-based: Python, block-based: Scratch,



cards: Code Notes) led us to observe key interactions between the student and the programming environment. Some of these interactions were; the impact of programming language architecture on learning, learner-friendliness of development environment, and effect of classroom medium on motivation. These observations improved our understanding of how children learn programming and showed how we could improve our design. After this part of the paper, we shared the design process of Code Notes and discussed our observations from the studies.

Design Before Studying with Children

In our two-month project, besides teaching a programming language, our main goal was to teach the algorithm concept and integrate it to children's daily lives. First, we started to plan the curriculum to see the content we needed to cover with our programming language and mobile app. Our curriculum was based on three main grounds: (1) the depiction of coding algorithms with daily life tasks, (2) teaching main functions (if/else-for-while), and (3) practicing these functions in Scratch and Python. On the first ground, we presented the algorithms as daily life tasks. For example, we built an algorithm for brushing teeth in four simple steps: (1) Open the toothpaste, (2) Put a piece of toothpaste onto toothbrush, (3) After opening your mouth, brush your teeth with circular movements, and (4) if the teeth are clean, stop brushing and gargle. This way, the children were more aware of the logic of programming and practiced this understanding daily.

On the second level, Code Notes was designed to teach the basic concepts of programming such as *if/else* structures, *for* and *while loops* and compiling simple outputs. We defined three rules for Code Notes: (1) Code starts with a START card and ends with an END card, (2) If a card uses a conditional, these cards (IF/FOR/WHILE) must be side by side, (3) Except for rule 2, all cards must be one under the other. These simple rules were intended to construct a foundation about the grammar definitions of programming languages. To compile this language, Code Notes has its own Android application. As explained at the side bar, Code Notes Compiler interprets the text to generate an output array to represent different outputs such as pictures, music or animations. We decided to keep the abstraction of these output statements at "high-level" because we wanted to focus on very basic concepts on programming and not want to distract students with details. 'Low-level statements' could be implemented for advanced programming and the learner would be able to insert the image and decide on its size and so on.

Observations During the Interaction with Children

Our curriculum gave insights to students about how algorithms work, how they can form algorithms for problem solving and how to create Scratch and Python programs. One of the advantages of working with different languages at the same time was that it spiked children's curiosity about how programming languages differ. Students mainly raised questions about the grammar rules of writing a Python program. For example, reasons behind the usage of indentation for different types of blocks and quotation marks for *Strings* were frequently questioned. These questions show that we succeeded in spiking curiosity.

We also observed the differences between studying with smartphone and computers. Students used their

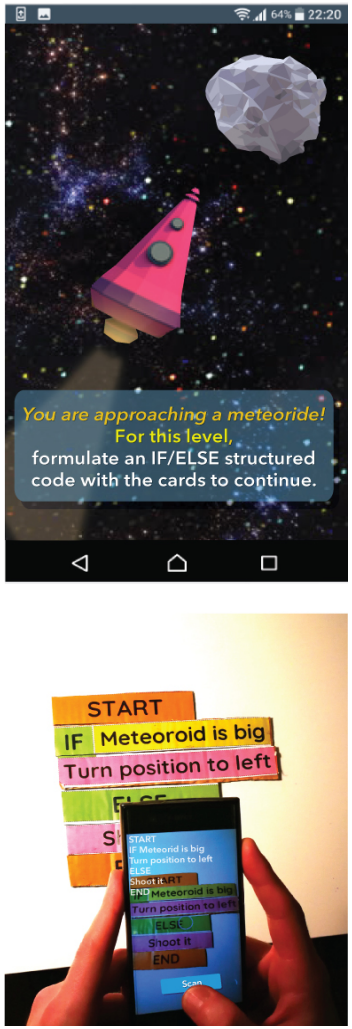


Figure 2: A space game prototype to teach basic programming concepts with Code Notes. For this figure, player need to build an if/else structure to destroy meteoroid.

Android phones to code and learn the basic concepts. Then, they used computers for creating simple programs with Scratch and Python. Using smartphones revealed two main advantages: relevance and interactivity. The children we worked with were more accustomed to smartphones than computers. So, when writing a code, they hesitated to use the computers we provided; many of them did not have access to personal computers and did not feel comfortable while working with them. Another advantage of smartphone was the increase of collaboration within classroom. Since they could pass smartphones around easily, interactivity improved. Additionally, using smartphones in coding allowed students to see these devices as educational materials.

The most valuable output of this phase was seeing the children's enjoyment from the process. For example, outside the classroom, they were sarcastically telling "*IF* I am older than you, I must eat your cake, *ELSE*, you can eat my cake." This is a good example showing how they understood and internalized the *if-else structure* and were amused to use it in daily conversations. To benefit more from this enjoyment, we asked the children about how this process could be more fun. They answered with the idea of a spaceship simulation game. In this game, player needs to solve problems by creating Code Notes programs, like in Figure 2. This game was one of the creative future products of this design phase.

A further evaluation: Development of Skills

We also wanted to see whether teaching programming via our curriculum influenced children's cognitive abilities. Previous studies done with interactive programming tools such as Scratch and Lego Mindstorms concentrated on the development of problem solving, logical reasoning of the participants [4, 5, 6]. After we got approval from ethical committee, parents and students provided consent to participate in this evaluation part. We tested whether interacting with tangible modules via Code Notes could improve children's cognitive abilities, mainly spatial reasoning. Our test included 15 questions with three sub-scales (mental rotation, spatial orientation, and spatial visualization). This test was an adaptation of a spatial reasoning test used by Ramful and colleagues [7]. Preliminary results had positive indications, suggesting that children's spatial reasoning was improved from 55% to 63%. However, we did not have a control group in this initial evaluation. Adding a control group and comparing the learning paces of our curriculum with different programming languages will provide further information about the effectiveness of Code Notes.

Things We Want to Keep

After all, engaging students actively in classroom environments and creating a playful interaction was the main purpose of our project. Our low-cost, mobile teaching tool revealed three insights that encourage building this environment: (1) increased number of questions raised by students, (2) enhanced interactivity through passing phones and tangible card blocks easily, and (3) the simplicity of development environment encouraged students to develop their own products. In addition to these advantages, Code Notes created a familiar and comfortable playing environment. This environment was a result of using two convenient materials that most children have. The first material, smartphones being more accessible to them than computers, created a more interactive, familiar, and creative environment. Therefore, we decided to

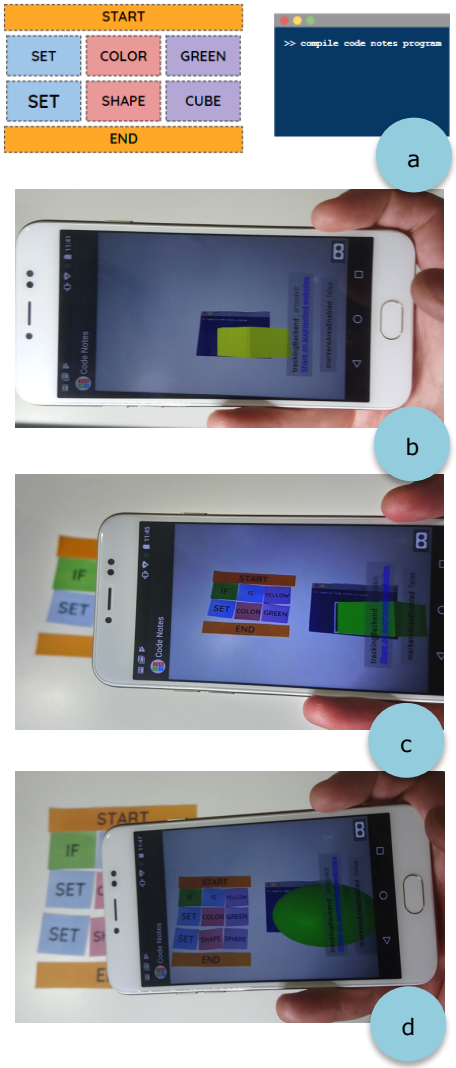


Figure 3: Compilation of new version. (a) Prototype (b) Terminal screen shows the last output: yellow cube. (c) After putting set-color statement, shape turns to green. (d) Then set-shape statement turns the shape to sphere.

continue developing on the mobile platform. The second convenient material was cardboards used as tangible blocks. The cardboards also created a sense of reliance that enabled playing with them freely, since children knew that the material was cheap, they could play with them as they wanted. The observations and possible future products favored to keep these main materials for future iterations.

Things We Want to Change

In the lectures, Code Notes was a great assistant tool. But this release of the Code Notes needs some improvements to cover programming concepts. For example, in this version outputs are drawing animals, opening other programs or playing music. But the learner cannot modify the output. Expanding the possible output number can help learners to create their unique ideas. Code Notes also needs an improvement on variable and function definitions. When the learners can create modular pieces with function definitions, they would be able to create their own stories and animations on Code Notes and better comprehend programming concepts.

Design of the Next Iteration

After considering all the pros and cons, we decided to keep the bone materials, but change the compiler and the generated outputs of Code Notes. This version of Code Notes was influenced by the space simulation game. As children wanted to change parameters and see the results at the same time, we added a terminal card to generate augmented reality outputs on this card at real time. In Figure 3, an example of a program execution is shown: (a) The language supports to show basic geometric shapes as output. Variables of the language are colors, shapes, and positions so that learners will be able to draw the defined shapes, modify them, and put them at different positions. On the terminal screen card, output renders at real time. (b) When the camera only sees terminal screen, application renders the last output. (c) When the code, IF – IS – YELLOW -> SET – COLOR - GREEN is placed, color changes to green at real time. (d) Then if we put SET – SHAPE – SPHERE, it will change to a sphere. AR feature allows us to use the smart phone capabilities in a more effective way the leads to more enjoyable interactions compared to the version that provides only 2D output.

Future Work

Code Notes application is released at Google Play Store. The first step will be the further development of the AR version. After that, we want to adapt Code Notes to various classroom materials and enhance functionality. Modifiable variables and function definitions are the first steps of this future work. We will further explore how we can create different programs, by rethinking the size of cardboards and the camera's field of view.

After we saw that students were motivated to direct their own games and stories, we created the prototype, as a sample page seen in Figure 4. With adapting Code Notes to storytelling books, we can create interactive books for children to determine the direction of the story by using codes. Additionally, we observed that children asked more questions after interacting with Code Notes in the class. Thus, we need to further investigate the use of tangible cards with the mobile phones in other areas of education. Workshops with teachers on different topics can result in different types of conceptual tangible blocks.

Figure 4: A sample book page from the book that we can program stories with tangible cards.

(Example chapter from *Where the Wild Things Are* by Maurice Sendak)

Furthermore, affordability of Code Notes allows us to create low-cost environment to teach programming. We formed a collaboration to apply Code Notes Project with a non-governmental organization, the Educational Volunteers Foundation of Turkey (*TEGV*) (https://tegv.org). TEGV gives free education to more than a hundred thousand children in middle school, which will create a unique opportunity for us to observe the effects of using this platform. The project will combine other programming tools along with Code Notes, while focusing on computational thinking and problem-solving abilities.

Conclusion

We introduced Code Notes as a programming education tool that uses simple programming cardboards with the companion Android app to teach some essential programming concepts. Code Notes introduces a financially reachable and open-source alternative to other physical tools for programming. Furthermore, it enables to create a collaborative and engaging environment. We used this tool in our voluntary project to examine the interactions triggered by Code Notes in the classroom environment. Our paper presented how we designed the Code Notes system prior to and after working with children.

Code Notes, as it is easily reachable, can inspire teachers to adapt it to various applications. An example is integrating the tangible cardboards to storytelling books to enable children to program stories. It can also be used for teaching science, math and history by integrating cards and mobile phone concept to curriculum. Reachability of project quickly resulted with collaboration with an NGO to access more students nationwide. This collaboration shows that Code Notes can allow to create affordable programming teaching environments for developing countries.

References

1. Oren Zuckerman, Saeed Arida, and Mitchel Resnick. 2005. Extending tangible interfaces for education: digital montessori-inspired manipulatives. In Proc. SIGCHI Conference on Human Factors in Computing Systems (CHI '05). ACM, New York, NY, USA, 859-868.
2. Sidhant Goyal, Rohan S. Vijay, Charu Monga, and Pratul Kalita. 2016. Code Bits: An Inexpensive Tangible Computational Thinking Toolkit For K-12 Curriculum. In Proc. Tangible, Embedded, and Embodied Interaction (TEI '16).
3. Michael S. Horn and Robert J. K. Jacob. 2007. Designing tangible programming languages for classroom use. In Proc. Tangible and embedded interaction (TEI '07). ACM, New York, NY, USA, 159-162
4. Özgen Korkmaz. 2016. The effect of scratch- and Lego Mindstorms Ev3-Based programming activities on academic achievement, problem-solving skills and logical-mathematical thinking skills of students. Malays. Online J. Educ. Sci. 4, 73–88.
5. Lai A.-F., Yang S.-M. 2011. The learning effect of visualized programming learning on 6th graders' problem solving and logical reasoning abilities. *In. ICECE.*
6. L.W Gibbon. 2007. Effects of Lego Mindstorms on Convergent and Divergent Problem Solving and Spatial Abilities in Fifth and Sixth Grade Students. Doctoral dissertation, Seattle Pacific University, Washington, USA.
7. Ramful, A., Lowrie, T., & Logan, T. (2017). Measurement of spatial ability: Construction and validation of the spatial reasoning instrument for middle school students. *Journal of Psychoeducational Assessment*, *35*(7), 709-727.