Strawbies: Explorations in Tangible Programming

Author 1, Author 2, Author 3
Anonymous
Affiliation
Address
email1, email2, email3

ABSTRACT

In this demo we present Strawbies, a real-time tangible programming game developed for ages 4 to 7. Strawbies is played by moving and connecting physical wooden tiles in front of an iPad. This interaction is made possible through the use of an Osmo play system, which includes a mirror that reflects images in front of the iPad through the front-facing camera. We combined this system with the TopCodes computer vision library to result in fast and reliable image recognition. Here we describe a set of principles that guided our iterative design process and discuss how testing with children informed the most recent instantiation of Strawbies.

Categories and Subject Descriptors

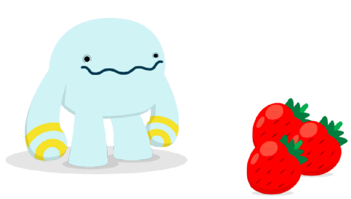
H.5.m [**Information interfaces and presentation (e.g., HCI)**]: Miscellaneous.

Keywords

Children; tangibles; games; strawberries.

1. INTRODUCTION

In this demo we present Strawbies, a tangible programming game designed for children ages 4 to 7. Strawbies is an iPad app that features Awbie, a character that children guide through a virtual world using wooden programming tiles (Figures 1, 2, 3). Our system combines the TopCode computer vision library [TOPCODES] with the Osmo play system [OSMO] to allow for real-time recognition of programs that children construct on a flat surface in front of the iPad (Figure 1). This results in an inexpensive, engaging, and portable tangible programming environment. Our design process has involved five major revisions of the game that were tested with children, parents, and teachers (see Figure 4).



2. DESIGN PRINCIPLES

Our design builds on prior work in tangible interaction and children's programming environments and was guided by the following seven principles:

1. Inviting: Through the use of tangible programming tiles, we hoped to create an inviting play experience that would draw children in to a collaborative play experience. The use of tangibles increases the visibility of game play, allowing it to move beyond the screen and spill out into the real world. In our testing sessions, children would often notice the game from across the room.

2. Playful and Open-Ended: In the spirit of many children's programming environments [PAPERT, ETC], we wanted to create a fun experience that would support children's open-ended exploration. This led us, over time, to an open-world style of game in which Awbie is free to roam around an infinite, randomly generated landscape. While the game is loosely structured around the task of guiding Awbie to eat and grow straberries, children are free to playfully explore the world in any way they see fit.

3. Simple+Complex: As with prior work on the development of programming environments for informal learning settings [Horn,

Figure 1. Children playing with Strawbies using tangible tiles, an iPad, and the Osmo attachment.

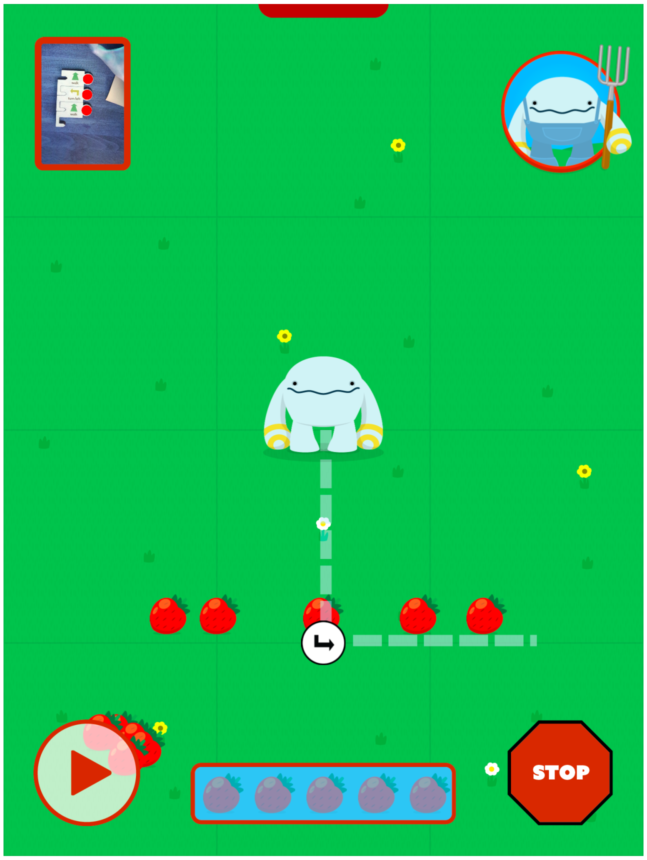


Figure 2. Screen shot of the iPad game. The game shows an image from the front camera to help reinforce the link between the tangible programming tiles and Awbie's movement in the world (top left). We use arrows and icons to preview what will happen programs are played.

Horn, CMU], we wanted to create a system that was simple enough for children as young as four years old to figure out on their own with little or no instruction. However, we had to balance this goal of simplicity against the ability to create relatively sophisticated programs that could lead to complex behavioral outcomes for Awbie.

4. Fluid and Responsive: SOMETHING HERE ABOUT REAL TIME

5. Developmentally Appropriate: We wanted to ensure that the content of our game was accessible to our target audience and that the game play was appropriate for children's cognitive, perseptual-motor, and social ability. To ensure that we achieved this, we conducted multiple rounds of testing with children in our target age range.

6. Pedagogically Aligned: One of the greatest challenges facing the adoption of developmenally appropriate technology in classrooms is that teachers must feel comfortable and confident with the technology themselves [CUBAN]. This includes making sure that technology aligns with the pedagogical philosophy of early childhood educators, a philosophy that emphasizes rich sensory-motor experiences, open-ended exploration, and social interaction. We see the use of tangible technology as an excellent way to introduce computational thinking activities in a way that evokes familiar cultural forms of teaching and learning [Horn].

7. Social: Along the lines of the previous design principles, we sought to design an activity that was inherently social in nature in that it would invite multiple children into the game, allow them to easily distribute their activity in collaborative play, and create space for multiple hands and bodies.

8. Adaptable: The difference in ability of a four-year-old and a seven-year-old child can be dramatic. In developing the game, we sought to create an activity that could grow with the child, perhaps with the proximal assistance of an adult or older child. For example, for pre-literate children, it might be necessary to read text out loud to help children interpret meaning and understand the graphical elements of the blocks. Or, teachers might choose to remove some of the blocks (such as the event triggers) from the activity until a child has mastered the more basic concepts. This is one example of the flexibility that tangibles afford—no customization of the software is necessary to instantly adapt the activity.

3. RELATED WORK

Our project builds on a long and rapidly growing tradition of programming environments for young children [Papert, diSessa, Weintrop, ScratchJR, Montemayor, Wyeth, Dr.Wagon, Tern, Stickers, Scratch, Logo] (see also [McNerney] for an excellent account of older work dating back to the 1970s). There is also growing momentum around the idea of supporting computational literacy activities throughout K-12 education and starting at the earliest grade levels. Our project contributes to this space by improving the flexibility, portability, and practicality of tangible programming for young children.

4. DESIGN & EVALUATION

Our design process involved several major revisions in which we explored different approaches and ideas for the iPad app and the tangible programming tiles. For the programming tiles, we wanted to make sure that blocks were easy to assemble and disassemble while also making sure that connected programs could be dragged around the table without falling apart. For programs, we included parameters, loops, and if/then logic. The blocks were split into several categories: verbs, adverbs and units of measurement. Verb blocks begin the start of each line of code, and the player can attach adverbs and units of measurements to the verb. For example: `walk (verb) + 5 inches (unit) + right (adverb)`. Verbs slide in and out of other verbs conveniently, while units and adverbs attach to verbs like puzzle pieces. Each string of verbs could start with an `Always` block (for looping behavior) or a `When` block (for if/then logic). Due to the limited field of vision of the Osmo and its more intuitive nature, we chose to use event-driven logic over sequential logic.

Version 1: Puzzle Based

Our first game iteration consisted of a series of puzzles, much like code.org's Learn Programming series. From testing, we found that our players were easily frustrated by the game—especially the lower end of our target age range. Puzzles had only two or three possible solutions, and failure meant that players had to restart the puzzle. We felt that the interaction opened up by the Osmo and tangible pieces could be much more playful and open-ended.

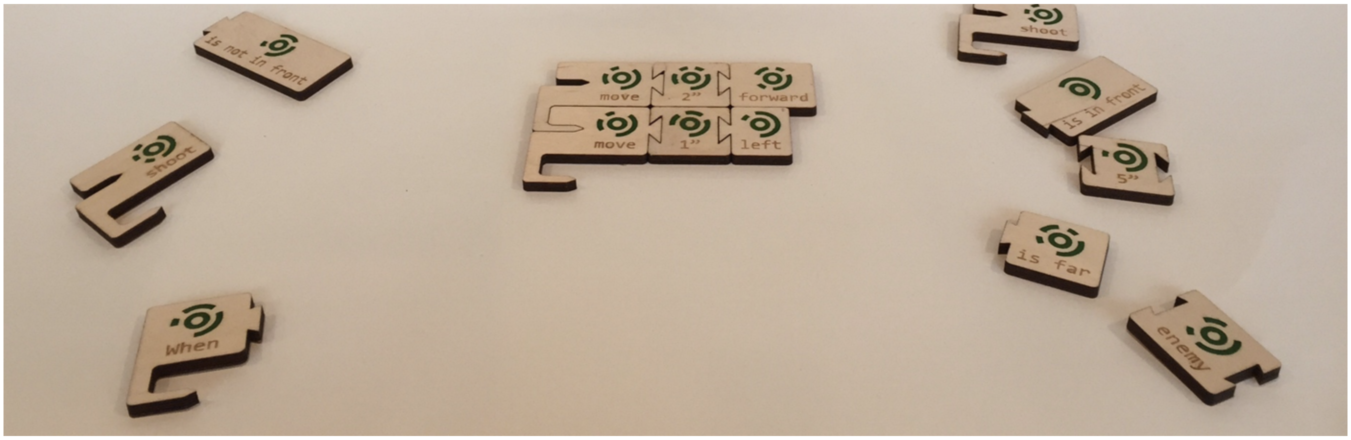


Figure 3. Collection of programming tiles available to guide the character in search of stawberries.

Our screen layout was also influenced by Scratch Jr. and other tablet coding games that do not use tangible pieces. Half of the screen was, therefore, dedicated to showing what the iPad was reading. This was unnecessary, as the physical code blocks in front of the iPad is an extension of the UI.

Version 2: Open World Environment

Realizing that the physical representation of programming tiles did not require screen real estate opened possibilities. Responding to user testing, we created an open world game in which the goal was to harvest "Strawbies". Because we wanted to create a fluid and responsive game that reacted in real time to kids' programs, every block that was recognized by the system would be immediately added to a pool of actions that the avatar could perform. The game would randomly chose from all tiles in front of the iPad and perform that action.

The structure of the game was provided by obstacles: water, trees, and bats. Trees obstruct movement, while contact with water or bats caused the game to end. Players could add events triggered by obstacles ("When water is in front, turn 90 degrees right and walk").

This game was better received by players, but there were a few issues. Blocks had too many qualifiers; a line of code could require up to three different blocks. The game was also too challenging for children as well as adults. Furthermore, collecting strawberries did not appear motivating enough to engage players in deeper explorations of programming concepts.

Version 3: Refining the Open World Game

In this version, we kept the open world concept and made additional refinements to improve fun and playability. We started by simplifying the action blocks and removing required qualifiers. Player can instead attach numbers to verbs to increase the number of times an action is performed. We also added room for graphical symbols on each block to increase the accessibility for young children. In this version, we also made looping an explicit block that can be used to cause actions to be repeated more than only once. To make the game less frustrating and more fun we added three special action blocks: rainbow, tornado and flashlight action. The rainbow flies the avatar to a different place in the world, tornado consumes all the strawberries in a short radius, and flashlight scares away rats that are trying to steal the avatar's strawberries. To give users an idea of progression and ownership, we added a separate farm game-within-a-game. As players collect strawberries, they can start to grow fruit bearing plants in a garden.

5. EVALUATION

We brought the game for six play sessions at two preK-6 schools. We tested the game in two environments: a closed off space with two children at a time, and an open environment with many kids coming and going. Through these play sessions, the consistent observation is that the game is fun and engaging. Most play testers wanted to play more when asked told that their time was up. Our sessions were divided into 20-30 minute slots. The idea of growing a gardent of strawberry plants and and tapping on increasingly more strawberries seemed motivating.

The current iteration of the game is playable and fun. By refining these elements, the benefit of tangible programming becomes clear. Collaborative phrases such as "can you pass me this block", "don't do that, I want to do this", "we need to bring him down, do you see anything" are common as teams of play testers work towards a common goal. Physical collaboration is also common: for example, holding back another's hand or moving an undesired piece outside of the camera range.

When the play sessions were in open spaces, there were two occurrences of children coming up and asking what the blocks were. Once the children who came up started to play, other children would join in. When we tested in the open area, a group of 5 students eventually came to form around the iPad. We observed that though it was 5 students around 1 iPad screen, none of the children appeared bored. There were always pieces in front of them that they could play with, or slide in and out of the iPad.

Without the a tutorial of any sort, the children would interact with the blocks in very different ways. One boy in Grade 3 turned the block to face the iPad screen, so that the word "Walk" was facing the avatar on screen, and not himself. Another child thought that tapping on the block would send a signal to the iPad. However, with some hints (which can be replaced by tutorials), the children understood the how the tangible blocks worked without any more hints. Young children (ages 4-5) were able to remember the names of blocks such as rainbow and walk, despite not being able to read them initially.

A common behavior was to use only single actions. This was observed for young as well as older children (ages 4-10). While

Figure 4. We have developed four major revisions of the game that we have tested with children, parents, teachers.

we do not believe that this is an undesirable behavior as the children are learning the game and connecting blocks to actions, we are experimenting with ways for the next iteration to eventually encourage the players to use longer blocks and plan more strategically.

6. ACKNOWLEDGEMENTS

Omitted for blind review.

Kafai computational participation. Marina's papers David Weintrop Gross.

7. REFERENCES

[1] Bers, M. (2008). *Blocks to Robots: Learning with Technology in the Early Childhood Classroom*. Teachers College Press.

[1] diSessa, A. (2000). *Changing Minds: Computers, Learning, and Literacty*. MIT Press.

[1] Horn, M.S., Crouser, R.J., & Bers, M.U. (2012). Tangible interaction and learning: The case for a hybrid approach. *Pers. and Ubiq. Computing, 16*(4), 379-389.

[1] McNerney, T. (2004). From turtles to tangible programming bricks: explorations in physical language design. *Pers. and Ubiq. Computing, 8*(5), 326-337.

[1] Montemayor, J., Druin, A., Chipman, G., Farber, A., & Guha, M.L. (2004). Tools for children to create physical interactive StoryRooms. *Computers in Entertainment: Educating children through entertainment Part II, 2*(1).

[1] Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas.* New York: Basic Books.

[1] National Research Council. (2011). *Report of a Workshop of Pedagogical Aspects of Computational Thinking.* Washington, D.C.: The National Academies Press.

[1] Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas.* New York: Basic books.

[1] Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Sliverman, B. & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM, 52*(11), 60-67.

[1] Weintrop, D., & Wilensky, U. (2013). RoboBuilder: A Computational Thinking Game. In *Proc. ACM Technical Symposium on Computer Science Education*, 736-736.

[1] Wyeth, P. (2008). How young children learn to program with sensor, action, and logic blocks. *Journal of the Learning Sciences, 17*(4), 517-550.