Strawbies Tangible Programming Game

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, 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.

FIG 1

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

Conference'10, Month 1–2, 2010, City, State, Country.

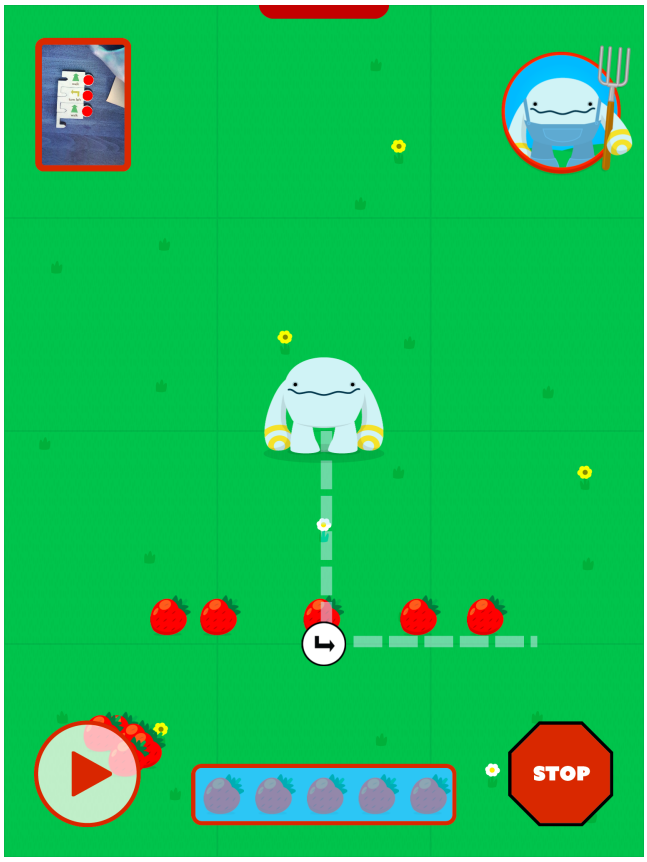

Figure 2. Screen shot of the iPad game. The game shows an image from the front camera to help reinforce the link between the tangible programming tiles and Awbie's movement in the world (top left). We also use arrows and icons to give children a preview of what will happen when their program is played.

4. 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.

5. 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].

6. 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.

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 ITERATIONS

4.1 Exploring developing for the Osmo

This version had two goals: making image recognition reliable and quick, and designing the blocks. The design of the blocks had several specifications. Connected blocks should slide around easily while maintaining structural integrity. Blocks should be easily insertable and removable at any point in the code. Blocks should also allow for customization, looping, and if/then logic.

Image recognition was simplified through the use of TopCodes. The Osmo was a perfect fit for TopCodes, as the image reflected by the Osmo could easily be distorted such that the codes were orthogonal to the camera. Using TopCodes provided reliable and responsive image recognition painlessly.

We went through numerous versions of these physical blocks before settling on a design which fulfilled all of our requirements. 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.

The game itself was 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 (4-10). Puzzles had only two or three possible solutions, and failure meant you had to restart the puzzle. We felt that the interaction opened up by the Osmo and tangible pieces allowed for more benefits than just the benefits typically associated with tangible programming, and we wanted to drastically rethink the game.

The UI layout was influenced by Scratch and other tablet coding

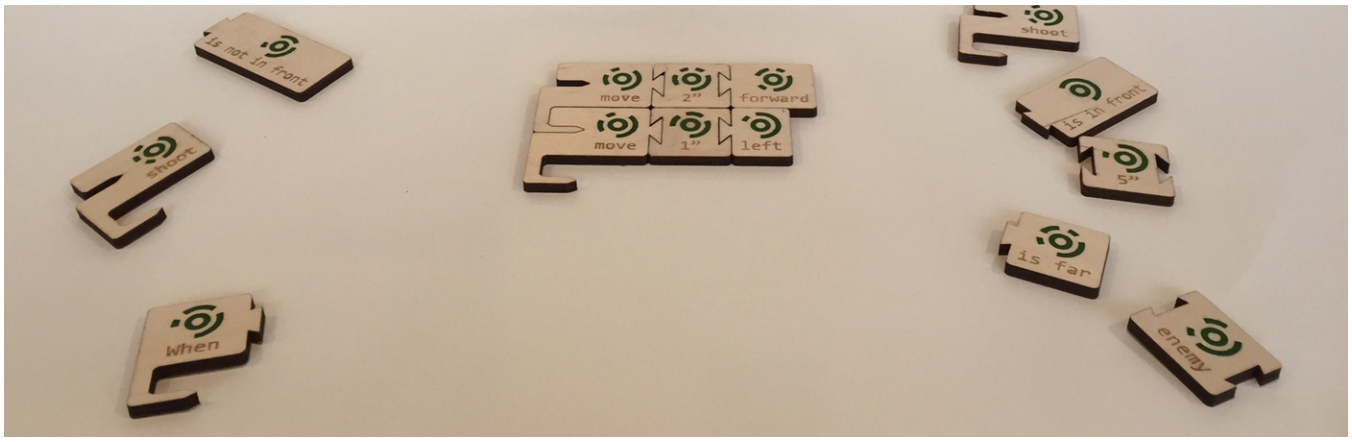

Figure 2. Collection of programming tiles available to guide the character in search of stawberries.

games which did not use tangible pieces. Half of the screen was 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.

4.1 Creating an Open World Environment

Realizing that the physical representation did not need to show up on screen opened up a lot of possibilities. Going off of testing which showed that puzzles with a binary answers was frustrating, we made an open world game where the goal was more open-ended (find "Strawbies"), and the solution even more so. We dedicated the entire tablet screen to the game itself.

We wanted to create a fluid and responsive game. This version reacted in real time to what was in front of the screen. Every block that was slid in front of the screen would be immediately added to a pool of actions that the avatar would do. The game randomly chooses from all available sets of instructions in front of the iPad, and performs the action. If a block was lacking qualifiers (for example: walk + 2 inches + ???), the game would pick one of the possible qualifiers automatically.

The game was open world and randomly generated. The structure for the game was provided by obstacles: water, trees and bats. Trees obstruct movement, and contact with water or bats causes the game to end. Players could add events that are triggered by obstacles. A possible even could be "When water is in front, turn 90 degrees right and walk".

This game was better received by players, but a few play sessions exposed a few issues. Blocks had too many qualifiers, a line of code can require up to three different blocks. The game was too challenging, for adults and children, and also far too reactive. Furthermore, collecting strawberries was not enough of a motivation.

4.2 Refining the Open World Game

In this version, we kept the open world concept as it showed optimistic results in testing. We made refinements many points in the game to improve the fun and playability of the game.

Simpler Blocks

We kept the same verb + qualifier structure, but we reduced each action to require no qualifiers. For example, the verbs are now: walk, turn left/right and special action words. The player can attach a number to verbs, which increases the number of times the action is performed.

Added symbols to blocks

We kept the features of the blocks, but redesigned them so there was room for symbols for each block. We added recognizable symbols to each block, as our target age range includes children not yet confident in literacy.

Looping no longer automatic

Looping is now an explicit behavior. The player must put a repeat block over the instructions to be repeated, or the player can attach an "infinity" qualifier to a verb. Unless stated otherwise, actions are now performed only once.

Fun special actions

We made the game less frustrating and more fun by adding in the rainbow, tornado and flashlight action. Rainbow flies the avatar to a different screen in the game, tornado consumes all the strawberries in a short radius, and flashlight scares away rats that are trying to steal the avatar's strawberries.

Real time projection coding

The previous version of the game would perform actions as soon as blocks were slid in front of the screen. We wanted to keep the responsive nature of the game, but also reduce the reactiveness. We added a level of immediacy between what is in front of the iPad and what the avatar performs by implementing a projection of the avatar's actions. Sliding blocks in front of the iPad affects the projection, but the player must tap on the screen to have the avatar perform the actions.

Adding a farmland

To give users an idea of progression and ownership, we added a separate farm game which can be populated with fruit bearing plants by collecting strawberries in the main game. The fruit dropped by the plants can be collected by tapping, which offers a break from the sometimes frustrating main game.

4.1 Further Iterative Refinements

I am not too sure how to give structure to these tiny refinements, will just list as bullet point for now.

- added leaf wiggle to trees when they are hit
- changed scoring system to be more apparent (5 strawberry

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

- progress bar at the bottom)
- added dirt lines in the farm game to give a clear sense of progression
- added tapping elements to the game to invoke a sense of empowerment over randomness
- changed the design of several blocks. walk had an arrow, which suggested direction
- added a little window in the UI which shows the camera feed and what is being recognized
- added a random number qualifier that could attach to verbs
- tweaked many little variables to produce a fun and challenging game

5. EVALUATION

We brought the game for six play sessions to two preK - 6 schools. We tested the game in two environments: a closed off space with two children, and an open environment with many kids coming and going. Through these play sessions, the consistent observation is that the game is fun and addictive. 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 their farm and tapping on increasingly more strawberries proved motivating. The graphics of the game were also polished, giving the children a familiar form (to their other iPad games).

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 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] 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.