Augmented-Reality Scratch: A Tangible Programming Environment for Children

Iulian Radu
Georgia Institute of Technology
Atlanta, GA 30332 USA
iulian@cc.gatech.edu

Blair MacIntyre
Georgia Institute of Technology
Atlanta, GA 30332 USA
blair@cc.gatech.edu

ABSTRACT
In this paper we introduce AR Scratch, the first augmented-reality (AR) authoring environment designed for children. By adding augmented-reality functionality to the Scratch programming platform, this environment allows pre-teens to create programs that mix real and virtual spaces. Children can display virtual objects on a real-world space seen through a camera, and they can control the virtual world through interactions between physical objects. This paper describes the system design process, which focused on appropriately presenting the AR technology to the typical Scratch population (children aged 8-12), as influenced by knowledge of child spatial cognition, programming expertise, and interaction metaphors. Evaluation of this environment is proposed, as well as foreseeable impacts on the Scratch user community.

Author Keywords
augmented reality, children, interaction design, metaphors, programming environments

ACM Classification Keywords
H.5.1 [Multimedia Information Systems]: Artificial, augmented and virtual realities, H.5.2. [Information Interfaces and Presentation]: User interfaces.

INTRODUCTION
Over the years, various researchers and designers have imagined that Augmented Reality (AR) technology would be well suited to children's programming environments. AR technology allows users to view virtual objects overlaid on a real-world context, and to control the virtual environment through direct, tangible interaction with objects tracked in the physical space. The blurring of the line between real and artificial can support the creation of fantastical experiences, such as stories in which otherworldly characters inhabit the space in front of the reader [4], and environments, such as ones where music is shaped through simple physical interactions [11]. Unfortunately, until now, a significant amount of technical expertise is required for creating such experiences, an insurmountable barrier to many potential users. Children, in particular, are both incredibly creative but lack the technical, mathematical and abstract thinking skills needed to work with typical AR programming tools. Thus, they are typically viewed as consumers, rather than producers, of such rich interactive experiences.

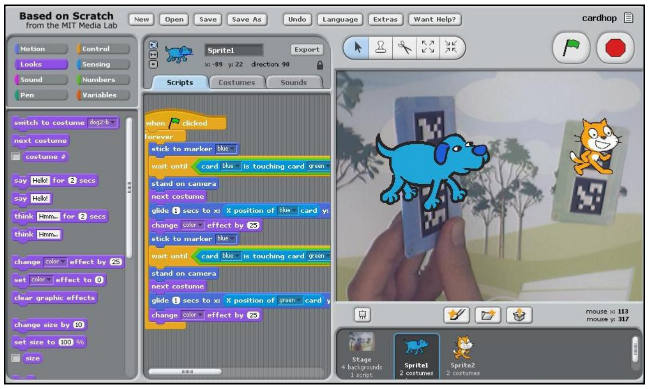
There exist a variety of programming environments which allow children to create digital experiences. Typical environments such as Alice [1], Microworlds LOGO [8], and Scratch [10], support the creation of applications that exist within the confines of the computer screen and are controlled through standard computer interfaces. Other environments, such as LEGO Mindstorms and PicoCrickets, allow children programmers to influence physical robotic objects. However, none of these environments allows virtual and physical objects to inhabit the same space.

Our current research introduces the first augmented-reality programming environment for children. This environment, using augmented-reality techniques and the ARToolkitPlus [13] software, supports mixing physical and virtual environments to create a coherent view of the merged space. Children see virtual objects overlaid on the real-world space seen through a camera, and they can control the virtual world through physical interaction with special markers that are detected and tracked using the same camera. By integrating AR technology into an age-appropriate programming environment and using metaphors and concepts that match the child's cognitive level, we hope to provide children with the functionality necessary to create such experiences.

DESIGN INFLUENCES
The goal of this work has been to create a programming environment in which children can author augmented-reality applications. Rather than design a new programming environment, we examined a number of successful programming environments to see if one could be extended with AR functionality (Alice, LOGO. and Scratch). Each was evaluated based on a number of criteria, including programming simplicity (it should be easy for children to create impressive applications in the specific environment), spatial simplicity (the environment should put minimal load on spatial cognition, with 2D being preferred over 3D), and ease of expansion (the environment should be open to expansion with AR functionality). The Scratch platform (see Figure 1) best satisfied our criteria, since it provides a simple drag-and-drop interface, generates 2D applications, and it is open-source. Furthermore, Scratch has a well-developed user community [10] which may be useful in

informing research on tangible and augmented-reality interfaces, as well as directly benefitting from this research.

Although there is a wide variation of user ages and skills, the majority of users of Scratch are pre-teens aged 8-12 [7]. We were particularly concerned with how children in this age range understand 3D augmented reality and tangible interactions. Early in the design process, we were interested in understanding what is known about children's cognitive abilities, so that we could present the AR technology at an age-appropriate level.



```
when clicked
forever
  stick to marker blue
  wait until card blue is touching card green
  stand on camera
  next costume
  glide 1 secs to x: X position of blue card y:
  change color effect by 25
  stick to marker blue
  wait until card blue is touching card green
  stand on camera
  next costume
  glide 1 secs to x: X position of green card y
  change color effect by 25
```

Figure 1. The Scratch programming environment contains a library of functions (left-hand column) usable in actor's programs (middle column), which execute in the right-hand display.

Spatial Cognition

We looked at the literature on spatial cognition and tangible-user interaction to see what kinds of spatial interactions children understand. Spatial cognition literature informs us that adults can use three types of reference systems when processing spatial relationships: in the ego-centric mode, directions and distances are determined by using the body as the reference point (e.g., the spoon is on my left); in object-centric mode, the reference point is an external object (e.g., the spoon is on the left of the fork); in the environment-centric mode, the environment is used as reference (e.g., the spoon is north) [9]. The research in [9,6] reports on the development of spatial cognition in English–speaking children. In these children, a basic form of object-centric cognition develops first in the early years, whereby children are able to segment objects into constituent parts (such as front and back) and determine if other objects are near these parts; this occurs even before children are able to articulate this information [6]. Ego-centric thinking typically develops after this phase by the age of 5, and full object-centrism is developed by age 12, whereby children are able to describe what is in the "line of sight" of an object [9].

One study of children interacting with an augmented-reality book [4] found that children of age between 6 and 7 are adept at controlling 2D actor sprites with handheld paddles. Problems were detected when motions of physical objects did not directly map to those of virtual actors – many children had trouble with a mirror camera image, where upward motions appeared as downward on the screen.

Based on this literature, we decided that our initial design should be very conservative, placing minimal demands on children's spatial cognition by (1) not adding a 3rd dimension to the Scratch environment, and (2) continuing to follow the screen-centric frame of reference. Children would be able to control actors with motions of physical objects, but actors would remain in the 2D plane of the screen; furthermore, spatial properties of physical objects (such as their rotations, or angles between objects) would be provided in a screen-centric perspective which is aligned with the ego-centric perspective of the programmer. Based on what we find from this first version, we hope to add 3-dimensional viewing of objects later in the development process, after we better understand how children perceive and think about 3D spatial relationships.

Children can control applications using two types of physical objects: inscribed (shown in Figure 2) and non-inscribed (shown in Figure 4.c). Inscribed objects contain a surface pattern which is used by the ARToolkitPlus software for accurately detecting object position and orientation. This surface is called a "card" for simplicity. Playing cards with special surfaces were designed first, primarily because they are simple, familiar and versatile; furthermore, they can be easily installed by parents by printing and sticking the patterns onto typical playing cards. Knobs were found potentially useful to children because of their simplicity in affording turning interactions, suitable for intuitively controlling characteristics such as the size or color of virtual characters.

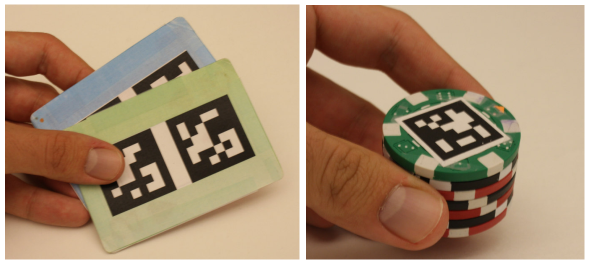


Figure 2. Tangible objects for application control: playing cards (left) and knobs (right).

Non-inscribed objects can also be used to affect program actors, as Scratch provides functionality for sprites to react when they touch specific colors. Figure 4.c. shows a Pong game created on this principle.

Programming Level

Finally, we wished to understand the complexity of programs created by the majority of Scratch users before designing an extension to Scratch. By surveying the applications created by children on the Scratch Forums website, we found that a majority of programs are driven by the users moving actor sprites through mouse or keyboard inputs. Furthermore, behaviors of game sprites are synchronized through the mechanism of event broadcasting. Children who create these programs appear to understand how to integrate environmental events in their programs (e.g., button presses), and appear to focus on control of sprite location as a central driver of their applications.

Interaction Metaphors

In integrating AR functionality to this environment, we closely followed the Scratch model of compartmentalizing functions. Scratch functions are grouped according to how they influence the current actor – typical sprite actors have functions for "Motion", "Looks", "Sensing", etc. Following this model, we first generated a list of basic functionality that AR technology offers, then grouped this functionality in two categories – a set of functions which are in charge of "Sensing" the real world objects, and a set of functions which direct sprite "Motion" according to the real world. The details of these sets of functions are described in the following section.

To present the functionality to users, we followed two simple metaphors which can help children conceptualize the technology in terms of ideas they are already familiar with. Firstly, we present the Scratch actors as things which can "stick" to the screen or to cards. When an actor is stuck to a card, its position changes to follow the position of the physical card as it appears on the screen. Commands to move the actor will cause it to move relative to the location of the card. To make an actor sprite stop following the physical object, it must be told to stick to the screen.

Secondly, the physical cards are presented as special sprites in the programming environment. In Scratch, a sprite actor can sense properties of other sprites, such as position, orientation, distance, etc. Functions corresponding to properties of physical cards follow a similar pattern, whereby a sprite can detect the location, orientation of a card, or relationship between sets of cards. Conceptualizing physical cards as virtual sprites should permit a wide range of existing programmers to grasp this technology.

AUGMENTED FUNCTIONALITY

Augmented-reality technology allows users to perceive computer-generated output as it is overlaid on real-world objects, and to control applications through real-world interactions. AR Scratch thus provides programmers with a library of functions which cover manipulation of both output and input.

Output in Reality

Scratch content is currently mixed with physical objects through two mechanisms. First, input from the computer camera is by default used as background for the application, allowing digital activities to be situated in the real-world space. Second, the programmer can instruct sprites to follow the position of a trackable card, while preserving the sprite as a 2D entity on the screen.

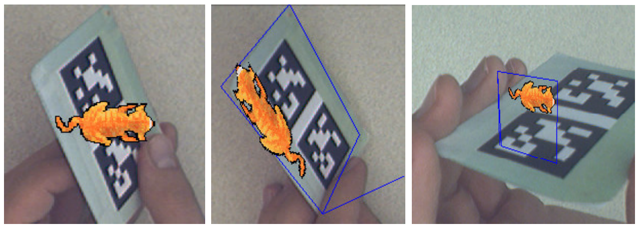


Figure 3. Appearance of sprite when it follows a card while staying on the screen plane (left), when it is projected on the surface of the card (middle), and when it is perpendicular to the card (right).

We have also constructed functionality to project sprites onto a plane parallel or perpendicular to physical cards, permitting sprites to exist as flat entities in a 3D world (see Figure 3). Furthermore, we envision that cards can be designed to have multiple reference points where sprites can be placed (eg: indicators for the "middle", or "top-right corner"), and also to indicate scaling factors. These features can support the creation of "pseudo-3D" applications without requiring the child to think or program using 3D concepts.

Control by Physical Interactions

Two sets of functions provide the programmer with information about the physical world. The first supplies data about individual objects, such as position, distance from camera, angles of rotation and tilt. The second gives information about pairs of objects, such as whether two cards are touching, as well as angles and distance between them.

Examples

By combining these simple interactions, children can quickly create complex applications. Figure 4.a. shows a drawing application created by programming a sprite to follow the card and leave a trail colored according to the card's rotation. Figure 4.b. shows a flower-growing game, where the user must first get a raindrop by touching the physical knob object to a virtual cloud, then tilt the knob past a certain angle in order to drop the raindrop, transforming it into a flower. Figure 4.c. illustrates a Pong game, created by bouncing the star when it hits the color yellow. Figure 1 shows a game where a dog rests on the blue card until the user tilts the card while touching the green card, causing the dog to walk to the green card and scare the cat.

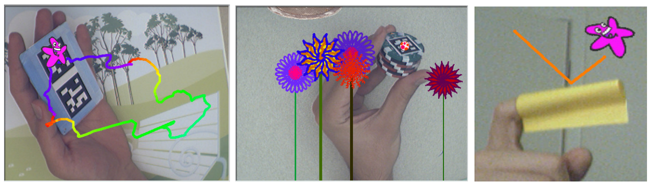


Figure 4. (a) Paint application. (b) Flowers game. (c) Pong game

EVALUATION APPROACH

Through several experimental sessions, we intend to study the extended Scratch environment in middle-school classrooms and allow children to become design partners in further developing the project. Our research aim is twofold: (1) to ensure that the AR technology is presented at a level appropriate for children's comprehension, and (2) to discover functionality which is desired by children but is not presently made available.

In the process of program development, children will create mappings between real-world interactions and program activities by leveraging the library of functions outlined above. If children are to use AR technology, they must understand the functionality correctly, and be able to integrate it into their designs. The studies will determine if the functionality is presented at the appropriate level of complexity, and if the applied metaphors are beneficial to children's understanding. Additionally, we expect that children may wish to take advantage of other real-world interactions which the system does not presently capture, and that children will generate novel types of applications using these and existing interactions. We will determine if and how this functionality can be integrated into the existing environment.

This knowledge will be extracted through observations of children playing with the interface, as well as group interviews. Central to the experimental process will be the Constructive Interaction (CI) [3] and Peer Tutoring (PT) [5] methods. The CI approach pairs children into teams which collaboratively explore a system. In the PT method, children that have learned to use a technology teach their knowledge to others. The team aspect of these methods creates a setting of natural interaction in which children are likely to be unbiased when communicating their opinions and thoughts. Both these approaches help to determine the mental models and conceptualization processes which children employ while learning new technologies. These methods are potentially more effective in extracting cognitive information from children than the Think-Aloud or one-on-one interview approaches, primarily because children may have difficulty verbalizing their thoughts or may need frequent prompting [2,5,3]. The group interview approach has been selected, over survey or single-person interview methods, for maximizing the reliability of answers. It has been reported that children are highly perceptive of expectations and roles, and will provide answers as mediated by those perceptions [12].

CONCLUSION AND FUTURE WORK

This paper has presented an augmented reality programming environment for children, developed by extending the Scratch platform. The environment brings AR technology to child programmers by minimizing cognitive load and leveraging simple interaction metaphors.

In the future, we expect the AR Scratch project to be integrated into the Scratch platform, and become available to the Scratch Forums community. It is foreseen that the user community will generate a variety of applications which mix digital content with real-world contexts. In these applications, children will generate mappings between tangible interactions and behaviors of programs; thus, a set of frequently-used natural mappings will emerge from the community. Researchers may also create applications which make use of novel interaction techniques, and use the community to determine if and how children grasp these concepts. In this sense, we expect that the community will become a research partner for tangible interface research.

REFERENCES

1. Conway, M., Audia, S., Burnette, T., Cosgrove, D., and Christiansen, K. Alice: lessons learned from building a 3D system for novices. *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (2000), 486-493.
2. Donker, A. and Markopoulos, P. A Comparison of Think-aloud, Questionnaires and Interviews for Testing Usability with Children. *PEOPLE AND COMPUTERS*, (2002), 305-316.
3. Druin, A. Cooperative inquiry: developing new technologies for children with children. *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, ACM (1999), 592-599.
4. Dünser, A. and Hornecker, E. Lessons from an AR book study. *Proceedings of the 1st international conference on Tangible and embedded interaction*, ACM (2007), 179-182.
5. Höysniemi, J., Hämäläinen, P., and Turkki, L. Using peer tutoring in evaluating the usability of a physically interactive computer game with children. *Interacting with Computers 15*, 2 (2003), 203-225.
6. Judith R. Johnston. Children's verbal representation of spatial location. In *Spatial Cognition*. Erlbaum, 1988, 195-206.
7. Levins, Martin. Mitch Resnick at ACEC Canberra 2008. 2008. http://www.levins.net/users/martin/weblog/4d358/.
8. Logo Computer Systems Inc. LCSI - Solutions - MicroWorlds EX. http://www.microworlds.com/solutions/mwex.html.
9. Majid, A., Bowerman, M., Kita, S., Haun, D.B.M., and Levinson, S.C. Can language restructure cognition? The case for space. *Trends in Cognitive Sciences 8*, 3 (2004), 108-114.
10. Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., and Resnick, M. Scratch: A Sneak Preview. *Proceedings of the Second International Conference on Creating, Connecting and Collaborating through Computing*, IEEE Computer Society (2004), 104-109.
11. Poupyrev, I., Berry, R., Billinghurst, M., et al. Augmented Reality Interface for Electronic Music Performance. *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (2001), 805-808.
12. Read, J.C. and MacFarlane, S. Using the fun toolkit and other survey methods to gather opinions in child computer interaction. *Proceedings of the 2006 conference on Interaction design and children*, ACM (2006), 81-88.
13. Wagner, D. and Schmalstieg, D. ARToolKitPlus for Pose Tracking on Mobile Devices. *Computer Vision Winter Workshop*, (2007), 6-8.