Augmenting Coding: Augmented Reality for Learning Programming

Nathan Dass
College of Computing
Georgia Tech
ndass6@gatech.edu

Duen Horng (Polo) Chau
College of Computing
Georgia Tech
polo@gatech.edu

Joonyoung Kim
College of Computing
Georgia Tech
jkim936@gatech.edu

Sam Ford
College of Computing
Georgia Tech
sford100@gatech.edu

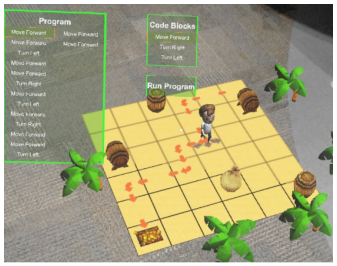
Sudeep Agarwal
College of Computing
Georgia Tech
sagarwal88@gatech.edu

Chinese CHI 2018 Poster, April 21-26, 2018, Montréal, Canada
ACM 978-1-4503-4656-6/17/05.
http://dx.doi.org/10.1145/3027063.3053103

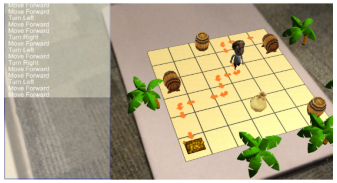
Abstract
Augmented reality (AR) is breaking into every industry and is finding a home in many unique and novel applications, due in part to its ability to engage users and their physical surroundings in potentially immersive means. We present our early investigation into whether these qualities of AR may be leveraged to help people learn coding more easily and with more fun. Using a within-subjects design with 9 participants, our pilot study evaluated two interactive AR coding environments: (1) head-mounted AR with Microsoft HoloLens, (2) mobile AR with ARKit on an iPhone; together with a conventional 2D touch interface using Swift Playground on an iPad as baseline. Participants enjoyed using mobile AR the most, and they also completed programming tasks the fastest when using it. Our current results suggest AR may have potential in enhancing beginners' learning experience for coding, especially for tasks that are more interactive and benefit from visual feedback.

Author Keywords
Augmented reality, teaching, HoloLens, ARKit

ACM Classification Keywords
H.5.m [Information interfaces and presentation (e.g., HCI)]: Mixed / augmented reality



a. Head-mounted AR
(Microsoft HoloLens)



b. Mobile AR
(Apple ARKit)

c. Conventional 2D touch interface
(Swift Playground on iPad)

Figure 1: (a-b) A user working on the *path-finding* task on two interactive AR coding environments; and (c) a conventional 2D touch interface. The user directs an avatar to navigate from the top right corner of a virtual game board to the goal in the bottom left, along a path coded by the user.

Introduction

The teaching of programming languages has become increasingly prevalent in curricula around the world. Coding has traditionally been taught in two dimensional environments. Recent improvements in immersive technologies such as virtual and augmented reality (AR) [1], however, have opened the door to more dynamic platforms for teaching how to code. This enables richer and more engaging learning experiences which allows learners to perceive and interact with content in a more natural way. While the novel and unfamiliar nature of AR environments results in users needing more time to adapt, the dynamic and interactive nature of AR helps improve performance in certain tasks [3]. There is also increasing evidence that supports AR's positive influences in education. For example, a recent study showed that AR technology promoted positive attitudes among university students towards science laboratories and helped them develop better laboratory skills [2].

We present our preliminary findings of testing augmented reality as a viable, novel platform for learning how to code. We focus on two interactive AR coding environments, as shown in Figure 1:

1. **Head-mounted AR**, using Microsoft HoloLens
2. **Mobile AR**, using Apple's ARKit on iPhone

We conducted a pilot study with 9 participants, using a within-subjects design to quantitatively and qualitatively evaluate the environments, where participants directed avatars to navigate virtual worlds using program commands, via spatial perception and interactions.

Code Learning Platform

To compare the performance of the two AR environments, we built a unified interactive code learning platform built with holograms. We drew design inspiration from Apple's Swift Playground. For example, they share the common objective of guiding an avatar to a goal (see Figure 1) on a 3D game board.

Implementation For front-end holographic designs, we used Unity3D, and for back-end game processing logic, we used C# and Swift, which were compatible with the HoloLens and ARKit, respectively.

Interface Design When a user selects a code block from the *Code Blocks* panel, that code block is added to the *Program* panel, which maintains the list of code blocks that the user selects. When the user adds code blocks to the program, orange arrows are added on the board which indicate the path that the avatar would follow if the current program was executed. The user executes the code by tapping on the Run Program button.

User Study Design

Our main objective at this stage of research was to learn about the two AR environments' potential in helping people learn to code, and how they compare qualitatively and quantitatively. However, we were also curious about how AR would compare with conventional "non-AR" 2D environments, such as Swift Playground. Thus, we decided to use a within-subjects design with three main conditions for completing tasks: head-mounted AR, mobile AR, and Swift Playground.

Participants We recruited 9 participants, aged 18 to 23, from our institution through advertisements. Two participants were female and the rest were male. All participants were screened to make sure they had minimal knowledge of coding. Each study lasted for 90 minutes. The participants were paid $15 for their time.

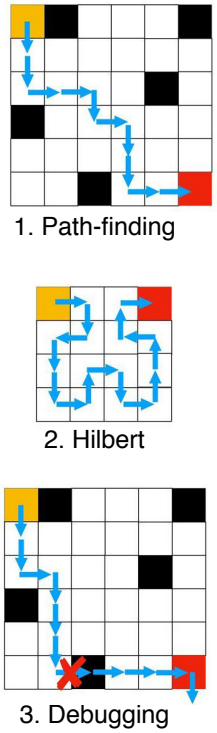


Figure 2: The three tasks each participant completes: (1) Path finding, (2) Hilbert, and (3) Debugging. Only one variation of each task is shown. Dark yellow box is the avatar's starting position; red box is the goal. The red "X" mark in (3) indicates the wrong path segment that the participant must fix.

Experimental Design We used the same room and identical lighting for all conditions. Each participant completed a set of three tasks in one condition, before moving to the next condition. The order of conditions was counterbalanced. We used three task sets, each containing three tasks with varying difficulty (Figure 2), to ensure that participants would not remember their solutions between sets (i.e., each task has three variations).

Procedure Before using each device, the participants would watch a usage demonstration given by the experimenter. Additional tutorial videos were provided for HoloLens. The participants would then try out the devices and familiarize themselves with the interaction gestures. Participants were given 8 minutes to perform each task during which we recorded audio, video, and time taken. If a participant failed to complete the task within the allotted time, the experimenter marked the task as a failure, and recorded 8 minutes as the task completion time. Participants completed an exit questionnaire asking for their subjective feedback, such as the tools' immersiveness, using a Likert scale (Figure 3).

Tasks The first task was *path finding* (Fig 2.1), where the participant would code a program that would direct an avatar to follow a path (in blue) to reach a destination (red), on a 6x6 game board with a few obstacles on it. The second task (Fig 2.2) also involved tracing a path, but a much more complex one based on the Hilbert curve, on a 4x4 grid. Due to the large number of turns in a smaller space, the Hilbert task required the participant to orient the avatar carefully at every step to complete the task. The third task (Fig 2.3) asked the participants to debug pre-written code that did not lead to a desired destination, by removing one command to fix the path. We designed this task to study how AR may help participants visualize and correct coding errors.

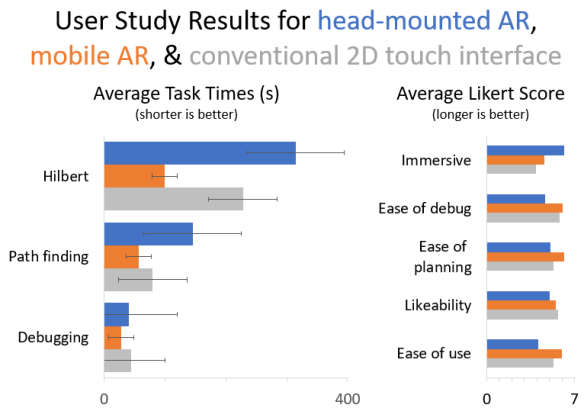


Figure 3: Average task completion times and likert scores for each platform. Mobile AR is statistically significantly faster across all tasks. Error bars represent one standard deviation.

Our hypothesis was that mobile AR would achieve the shortest task completion times since participants would use mobile device touch gestures that they might be accustomed to. Task completion time was our dependent measure and it could be affected by: (1) *interaction environment*: AR (head-mounted or mobile), or conventional 2D touch interface; and (2) *usage order*: the order that the participant used each interaction device. Using a Latin square design, we created 6 randomly assigned participant groups.

Preliminary Results

Task Completion Time The task completion times were analyzed using a mixed model analysis of variance with fixed effects for *interaction environment* and *usage order*, and a random effect for *participants*. The only statistically significant effect was *interaction environment*, suggesting successful counterbalancing of *interaction*

environment order. The *interaction environment* had a statistically significant effect on the *Hilbert* times ($F_{2,6} = 17.210$, $p = 0.001$), and *path finding* times ($F_{2,6} = 12.336$, $p = 0.004$), but not for *debugging* times ($F_{2,6} = 1.213$, $p = 0.347$). The space-filling nature of the Hilbert path leads to a high number of turns; the Hilbert tasks took the longest time to complete (see Fig. 3). Debugging, which only involved removing a single command, was the quickest to complete.

Mobile AR was the fastest environment for all tasks, and head-mounted AR the slowest, yielding a completion time around 3 times slower than mobile AR for Hilbert and path finding tasks. This is not surprising because Mobile AR provides a new visualization environment with similar gesture interactions while the HoloLens presents an entirely new set of gesture interactions.

As mentioned before, our main goal was to compare AR environments; we decided to also include a touch interface (Swift Playground) in our study so that we could also study the potential differences in usability and capability between AR and "non-AR" environments. We found that the off-the-shelf Swift Playground was slower than mobile AR; this was largely due to some participants choosing to type in commands at times, instead of tapping command buttons; and the visual feedback offered by our AR coding environment, which was unavailable in Swift Playground.

Ease of Use Although most participants thought the head-mounted AR was the most immersive, they found it most difficult to interact with and also mentioned that the HoloLens gestures often required several tries before they got recognized. A few participants also found the HoloLens disorienting and felt dizzy after prolonged use unlike in other platforms. The participants found mobile AR to be the most enjoyable.

Visual Feedback Participants reported that the two versions of our code learning platform were in general immersive and intuitive to use. Most participants also agreed that the real-time visual feedback provided by the augmented reality environment enabled them to code and plan their next move better in completing the given task.

Ongoing Work

We presented our preliminary investigation into using augmented reality for coding education. Both the head-mounted and mobile AR platforms show positive signs of potential. We plan to further evaluate the AR platforms with more complex tasks that require fine motor movements or audio-visual perception in a larger participants pool. We will also add tasks that include more complex coding concepts to assess the effect of augmented reality environments in aiding the learning of such topics.

REFERENCES

1. Murat Akçayır and Gökçe Akçayır. 2017. Advantages and challenges associated with augmented reality for education: A systematic review of the literature. *Educational Research Review* 20 (2017), 1–11.
2. Murat Akçayır, Gökçe Akçayır, Hüseyin Miraç Pektaş, and Mehmet Akif Ocak. 2016. Augmented reality in science laboratories: The effects of augmented reality on university students' laboratory skills and attitudes toward science laboratories. *Computers in Human Behavior* 57 (2016), 334–342.
3. Benjamin Bach, Ronell Sicat, Johanna Beyer, Maxime Cordeil, and Hanspeter Pfister. 2018. The Hologram in My Hand: How Effective is Interactive Exploration of 3D Visualizations in Immersive Tangible Augmented Reality? *TVCG* 24, 1 (2018), 457–467.