Gidget: An Online Debugging Game for Learning and Engagement in Computing Education

Michael J. Lee
The Information School | DUB Group
University of Washington
Seattle, Washington, USA
mjslee@uw.edu

Abstract—As interest in acquiring programming skills continue to increase, many are turning to discretionary online resources to learn programming. However, researchers and educators need more data to better understand who these learners are and what their needs are to create useful and sustainable learning technologies to support them. In my work, I investigate the factors that make a learning game engaging for users, and examine if playing through the game shows measurable learning outcomes. The game will be released the public, giving us the opportunity to collect large amounts of data. This data can be shared with other researchers to improve discretionary online tools such as educational games to support large-scale computing education efforts designed for a wide-range of users.

I. INTRODUCTION

Programming in the workplace is becoming more commonplace for many of today’s careers. Recent surveys have found that the computer literacy requirements have skyrocketed in almost every field [5], and that while there are about 3 million professional programmers in the United States, over 13 million more people say they do programming at work, and over 90 million use spreadsheets and databases [3,17]. As the need for programming becomes more commonplace, many people are turning to online discretionary learning resources to learn computer programming. Learners report that they enjoy these informal resources more than traditional classes because they allow for flexibility in how they learn, they give learners a better sense of retaining the material [1], and they are more motivating, engaging, and interesting than traditional classroom courses [6].

Unfortunately, many discretionary computing education resources has been slow to adapt, focusing on engagement, instruction, or scalability independently, but not on how to combine the three. For example, computer science distance education efforts such as Udacity.com, while scalable and instructive, can be isolating for students [2] and have high attrition rates. Similarly, Codeacademy.com, an interactive programming tutorial site, is scalable, but its effectiveness remains an open question because of its lack of evidence based instruction and assessment. Constructivist learning technologies such as Alice and Scratch [7], while engaging, are more difficult to scale because they require instructors at camps and after-school programs to promote learning [19].

Moreover, the majority of online learning resources typically share only the number of users that sign up for their site and little else. They rarely release information regarding the demographics of their users, what their users struggle with or succeed on, how many people continue to stay active on the site and actually complete all the tasks, or if users show any measurable learning outcomes. Having large-scale data of this kind of data would be invaluable to both researchers and educators by better informing them how to improve resources and materials for online computing education tailored to a wide range of people.

We believe that debugging games can be used to address these limitations by being engaging, instructive, and scalable for the following reasons. Debugging is a fundamental computational thinking skill and necessary for writing a correct program [12]. Our approach will be the first learning technology to teach debugging both explicitly before teaching programming. Finally, games are now a universal form of play for everyone: 91% of U.S. kids aged 2-17 play video games [7], with the average gamer being 34 years old and of all gamers in the U.S., 75% are 18 years or older and 40% are female [8]. Games are now also widely thought of as effective instructional tools [9], because they can provide concrete feedback about success and failure at reaching well-defined goals [10].

We will use a puzzle game as an instrument to learn more about the players of the game, including demographics, their progress through the game, and their knowledge and use of computing concepts before and after playing the game. To solve the puzzles, players will need skills that mirror those in debugging, including hypothesis formation and testing, mental simulation of programs, and evidence gathering. These skills will be taught explicitly through the game. Once the players complete the game, they will have the option to create their own levels using the Gidget language, which they can modify, share, and remix with their peers.
II. BACKGROUND

Our work uses an online debugging game called Gidget (shown in Figure 1) to teach programming concepts to novice learners. Players help fix faulty code provided by a robotic character to complete missions in the game. Several controlled experiments with online users have shown that is possible to translate debugging into engaging puzzle game mechanics that is appealing to a broad demographic [14,15,16]. More than 600 people between the ages of 18 and 66 years and of various genders, ethnicities, income-level, and education were recruited online and played the game. In addition, we had a total of 44 teens (between the ages of 13-18 years) play the game in a lab study and two summer camps [13].

Our work has demonstrated that novices can be highly engaged in learning programming concepts through a debugging game [14,15,16], that their initial negative attitudes towards programming can be changed [4], and that they can create their own programs after playing through the puzzles [13]. Across our studies, we have found that novices playing through our online game struggle largely with the same programming concepts that others have difficulties with in classroom settings [13,14]. However, we also observed that these novices were able to create novel, complex programs on their own by the time they completed the game [13].

III. KNOWLEDGE TESTS AND PUBLIC DEPLOYMENT

Gidget has been iteratively updated with improvements based on findings from each of our previous studies. Next, we plan to explicitly measure players’ learning after playing the game to complement our findings that players find the game engaging [14,15], even with integrated assessments throughout the game [16]. Afterwards, we plan to launch the game online, allowing us to collect user data at a potentially massive scale. Once deployed to the public, the data the game generates will shed more insight into who is attracted to this kind of learning technology, where they are coming from, and what they learned by playing the game. These contributions will provide a strong research base for the design of online discretionary computing education pedagogy.

REFERENCES