Shapemaker

A Game-based Introduction to Programming

Nicholas Masso and Lindsay Grace
Armstrong Institute for Interactive Media Studies
Miami University
Oxford, OH
massonj@muohio.edu

***Abstract*—As emphasis on Science, Technology, Engineering, and Math (STEM) initiatives increase, students are at a fragile point in their education. It is imperative that students learn core skills like computer programming and procedural logic necessary to living in today's increasingly digital society. Our new game Shapemaker solves these problems in game-based learning that incorporates an engaging augmented reality interface. ShapeMaker utilizes a table-based tactile interface that bridges physical actions with digital results, allowing players to learn basic concepts in programming by playing a card game. Shapemaker aims to transform the process of introducing new students to the world of programming and create a new, wider demographic of computer programmers. This article is a preliminary study of Shapemaker, documenting its concept, current design, and directions for development.**

Keywords-serious games; educational games; computer science education; augmented reality;collaborative learning;

I. INTRODUCTION

As emphasis on Science, Technology, Engineering, and Math (STEM) initiatives increase, students are at a fragile point in their education. It is imperative that students learn core skills like computer programming and procedural logic necessary to living in today's increasingly digital society [1]. Unfortunately, computer science students often find the basic concepts of computer programming difficult to learn. This fundamental educational challenge is widely recognized by educators as particularly important in introductory computer science curricula [2]. The challenge is to create situations where students are motivated to learn the basic concepts and foster enduring curiosity. So far, traditional approaches have had limited success in creating these situations [2, 3].

Game-based learning offers a promising new method to teach these basic programming concepts. This is due to the highly engaging nature of digital games and their ability to teach content in a way that students enjoy [4, 5]. Researchers have demonstrated that these games' designs can be as Prensky and Gee argued—motivating, enjoyable and effective [6, 7, 8].

The difficulty in engaging beginner students of programming emerges from the sharp differences between the digital world and the analog world. Beginning programmers obviously have limited experience in the digital world and need a truly introductory experience. Current attempts to teach introductory programming are either too detailed or too exploratory in nature.

Educational games like "Saving Sera", "The Catacombs", and a modified version of "Kernel Panic" all incorporate the syntax of programming into gameplay [6, 7]. These games force players to write and implement code as part of gameplay to entice them into real-world computer programming languages. But there are two drawbacks to this approach. First, the use of syntax limits gameplay. "Catacombs" asks players to construct code through a series of multiple choice prompts, a task almost as tedious and frustrating as coding [3]. Other proposed solutions present a simple programming language that players then utilize in the same manner as a professional programmer, but within gameplay. Incorporating syntax into a game is an admittedly difficult task that can only be used in a very narrow range of scenarios. Second, the use of syntax alienates players new to the concept of playing a game around syntax. If a player doesn't know computer programming syntax, asking them to use syntax as part of a game is a daunting requirement. This is because it requires them to learn the rules of a new language and use that language almost literally at the same time. In addition to this, they must also acquire new concepts within this new language. Typically, introductory philosophy and economics are not taught in a language unfamiliar to the student, yet we continue to teach function calls and memory allocation with strict syntaxes in foreign computer languages. It is then apparent why interest in computer technology often withers when students are introduced to computer programming [2, 3]. This is particularly true of children and secondary education students.

Environments like "Scratch" [9] and "Color Code" [10] bypass the syntax of code for users, but don't fold the experience into the structure of the game. In Scratch, users can create programs by dragging and dropping blocks and objects on screen, while in Color Code, color crayon drawings are used. Both solve the challenges of syntax by simplifying it to an extremely intuitive level. In Scratch and Color Code, users cannot write nonsense code. Randomly tinkering with their color palette or reordering their programming blocks produces new mappings, not a jumble of syntax followed by a compiler error. Their ability to easily get results makes these environments successful in engaging new programmers. But without goals, competition, and structure, these introductory environments lack the motivation and benefits held by games.

Shapemaker seeks to bridge these two separate approaches into an innovative approach to introduce students to programming.

II. OVERVIEW

Our new game Shapemaker resolves these problems in an engaging augmented reality interface. Shapemaker is a table-based tactile interface that bridges physical actions with digital results. Players learn basic concepts in programming by playing a card game. The order and location of game cards are interpreted by a custom computer-vision algorithm, to render on screen results. Players can learn the basic concepts of programming the same way they would learn to play Rummy or Go Fish. The system encourages trial and error, and shows clear connections between programming concepts and the physicality of the game.

Shapemaker employs a table-based interaction that allows players to organize playing cards to draw shapes on the computer's screen (see fig. 1). The game challenges players to translate verbal criteria into computer graphics, programmed with cards. Players compete against each other to see who can fulfill each goal first. Whoever wins the most rounds wins the entire game. As an added challenge, players can also play cards to influence how their opponent's code is compiled by the computer. This helps make gameplay more engaging and fun by adding variability and increasing player interaction.

Players become familiar with programming syntax and structure through gameplay without having to learn the complex details that real programming entails. As players explore the concepts of computer programming within the physical domain of the game, they form a conceptual foundation for understanding computer programming.

Players also build rapport when they play together. This rapport is integral to the learning experience embedded in Shapemaker's design. Players learn from each other as they play. They may share code, sometimes when they win a level, sometimes by imitating methods of their opponents, and sometimes by using cards to render their opponent's code ineffective. After one session of gameplay, most players have interacted more than they would in six weeks of a lecture course. We expect this increased interaction will promote player's comfort in asking for help when learning more about programming. Shapemaker aims to help build relationships

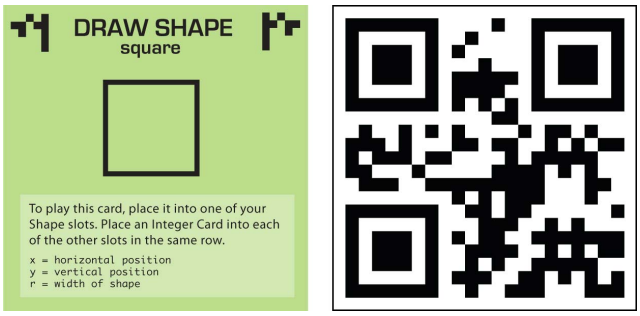


Figure 1. Front and back of a Shapemaker card

around a shared interest in programming, leading to greater educational benefits similar to those found in pair programming [11].

III. IMPLEMENTATION

Shapemaker was programmed with the Processing Java library and utilizes Daniel Shiffman's QRCode library [12]. The game has three basic components: physical input, screen based output and a data processing software application.

The physical implementation of Shapemaker includes two game boards, a Plexiglas table, and a webcam. Every card has a QRCode on its back. The front of each card has its name and instructions on how that card is played in the game (see fig. 2). The syntax for the represented Processing command is also printed on the front of each card. The cards were designed to be colorful but simple, combining motifs from computer programming and QRCodes with an appealing aesthetic design. The blocky designs are meant to evoke both QRCodes and the punch cards of early computers.

To play, the cards are placed, QRCode facing down, into slots in the game board. These slots are labeled to help players organize their cards. The slots also help align and order the QRCodes to promote fast and accurate parsing of the code data. Players stand on opposite sides of the game board, with a vertical shield to obscure each player's cards from the other.

A Mio webcam was positioned inside the Plexiglas table, on which the game board rests. The table was wrapped in black felt in the interior to reduce reflective light noise and a lamp was placed inside the table to improve image quality when

Figure 2. Setup of Shapemaker

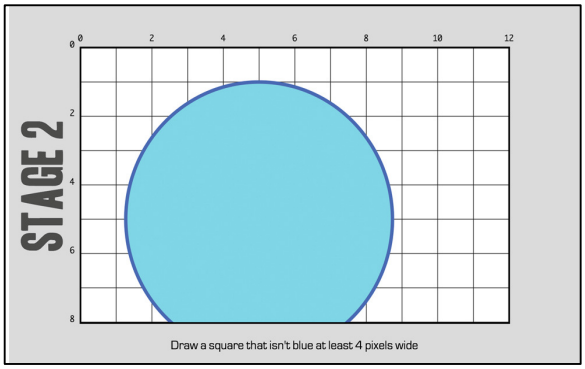


Figure 3. Digital Game Board

reading QR Codes. From within the table, the webcam photographs the state of the game board.

The screen based output on which results are shown is designed to be a lesson in computer programming conventions. This digital extension of the game board is a grid, labeled across the x and y axis. The grid is setup in typical computer graphics fashion—with an origin in the upper left-hand corner and normal image cutoff (see fig. 3).

The application component functions as a data processor for user input. When the program receives the webcam image, the Processing software splices the game board data into individual images. Each of these image slices is then processed in order by Shiffman's QRCode library. The library takes each image, decodes the QRCode into text to that then calls the commands. These commands can define variables, set shape fill colors, or draw specific shapes from player provided input, among other things.

Shapemaker also has a collection of simplified compiler errors—messages that let players know when they've executed incomplete or nonsense code. It checks for common mistakes like calling an undefined variable or missing input data in shape commands. The software also alerts players to trickier problems, like if their code will draw a shape off the grid. All of the error messages are delivered as friendly text alerts that appear over the digital game board. They were designed to avoid the typically technical and sometimes cryptic qualities of error messages in traditional programming languages. Instead, they are designed to be helpful and encouraging. A few of them are also expressed visually. For example, if a player codes a shape with no radius, Shapemaker will draw a minuscule version of their shape at their programmed coordinates. It also prints "Shape 1 still needs a radius (or width)" on the screen. If the player places inputs for a shape command, but doesn't play a shape command, a deformed combination of squares and circles is displayed, along with "Shape 1 still needs a shape command".

The feedback does not tell players how their shapes do not fulfill the current level's goals. As long as their code is properly constructed, the game leaves players to figure out how their shapes do not fulfill the level goal. The main challenge in Shapemaker is not constructing code that fulfills the level goal, it is the construction of code to reflect the players' wishes. The level goals are not very challenging by themselves—it is using the available tools effectively that is challenging.

IV. Game and Level Design

Shapemaker is begun by dealing a hand of seven cards to each player. The digital game board displays the current level goal (as seen in fig. 3). Players take turns by drawing a card. Then, they can either play a card, discard a card from their hand, or lay any number of cards down on the game board and run the interpreter. Players continue taking turns until one of them runs the interpreter and draws a graphic that fulfills the level goal.

The levels of Shapemaker are designed to slowly introduce the basics of the game, before letting players fully compete with one another. The first four levels have simple goals and a simplified, controlled version of the rules. For these levels, a players' starting hand of cards are predetermined and they do not draw or discard cards during these levels. None of the player interaction cards are used, since they would only distract and complicate the initial learning. Players do not use the entire physical game board either. A board overlay that hides unnecessary slots is included for each introductory level.

The first level has the goal "Draw something", players are given the four cards needed to draw a shape, and the overlay only shows the four slots needed to draw a shape visible. Players only have to deal with information that is directly related to their simple goal. This prevents them from feeling overwhelmed by the game's complexity and allows them to become comfortable with the basics before trying to accomplish more specific goals. The following three levels have slightly more complex goals, but also reveal more game board slots and include more advanced cards. After these four learning levels, the goals become more involved, players draw and discard cards, player interaction cards are included in play, and the full game board is open for use. Having learned the basics of the game's interaction model, players can then enjoy more open play with their opponent.

V. Evaluation

Shapemaker has undergone two sessions of informal user testing for feedback to guide development of the game, resulting in the iteration presented in this paper. The first session was conducted in an informal exposition at Miami University with an audience of game designers and game industry professionals. Twelve people played the game and provided qualitative feedback about the experience. The feedback from this evaluation session directed revisions to the instruction set and physical implementation. In the second session, five people volunteered for user testing, where two of them played a game of Shapemaker and three watched and offered comments on the experience. This user test was conducted in a controlled lab environment and led to revisions in the game's feedback system.

The first feedback we received from most volunteers in both sessions was excitement about the unique approach of the game towards education and programming. Professional programmers and their friends found it to be a refreshing interpretation of programming and were interested in having it implemented in a classroom setting.

Another common theme from the second testing session was that player's did not feel as if they knew how to play the game. The initial version of Shapemaker did not include any tutorial levels or instructions. Players were expected to make educated guesses, check their guesses, and modify their approach accordingly. But without any direct feedback, players had difficulty understanding what approaches were getting them closer to drawing shapes and which ones were not. This feedback motivated the creation of the tightly controlled learning levels that were discussed above. It also motivated the encouraging feedback system. These modifications provide simplified, clear environments for players to discover how the interaction model works with the guidance of feedback from the system.

One aspect of gameplay that was especially engaging for players in the second session was the player interaction cards. Despite other difficulties in gameplay, they clearly enjoyed playing cards that affected their opponent's code. It was also one of the few times that the two players (whom had never met each other before) directly addressed the other in the game. Even antagonistic plays seemed to create intimacy between the two players. Future versions of Shapemaker will include more of this type of cards to create richer player-to-player interaction and help in building rapport.

The developers also observed that the Plexiglas table was too tall for comfortable gameplay. Players had to stand because sitting down would put the table surface near their nose. While this was not a problem for testing, it would not work as an extended physical experience. This issue could not be easily resolved because the surface must be far enough away that the webcam can capture the entire surface at once, a distance of approximately three feet. There are two possible solutions to this problem. First, players could sit on stools. This would be an easy solution. Second, the physical interface itself could be changed to something more appropriate, like wall mounted cars that facilitate smartphone photography to read the players' cards. This would require much more work in technical development, but would ultimately result in a game that is more suited to mass production.

Overall, the testing done thus far has revealed that people are excited by the concept and approach of the game. There are improvements to be made to the technology and the game mechanics. However, the current iteration of Shapemaker shows promise by delivering the key parts of the concept in the gameplay experience.

VI. CONCLUSION

The concept behind Shapemaker is the most important part of this project, one that we believe is worth sharing with other researchers. This approach has real potential to solve issues facing all computer science educators. We plan to continue developing the Shapemaker prototype to further demonstrate this. We plan to explore the spatial context within which players draw their shapes. More advanced levels might ask them to complete a graphic, like adding wheels to a graphic of a car driving down a street. We also plan to create more player interaction cards, as this was one of the highlights found in the play experience thus far. Instead of having players compete, there might be ways to add components of the game where they have to cooperate.

Shapemaker is an exciting step towards engaging a wider audience in computer science education. It simplifies the basic concepts in computer programming and presents them in an intuitive environment where players can easily gain experience working with those concepts achieve goals. At the same time, the game provides opportunities for players to see the syntax behind their actions without forcing them to use it as part of gameplay. As it is further developed, Shapemaker comes closer to realizing its potential to transform the process of introducing new students to the world of programming and creating a new, wider demographic of computer programmers.

ACKNOWLEDGMENTS

First, we must acknowledge and thank graphic designer Lauren Romano, who designed the aesthetics for the game, drawing inspiration from minimalist and postmodern aesthetics. She was instrumental in the creation of Shapemaker. Thanks also to Jacob Tonski for periodic feedback and technical consulting. Last, thanks to Miami University and the Armstrong Institute for Interactive Media Studies for resources provided to support this project.

REFERENCES

[1] Rushkoff, D. Program or be programmed: ten commandments for a digital age. OR Books, 2010.

[2] Falkner, K. and Palmer, E. 2009. Developing authentic problem solving skills in introductory computing classes. SIGCSE Bull. 41, 1 (March 2009), 4-8.

[3] Kinnunen, P. and Simon, B. 2010. Experiencing programming assignments in CS1: the emotional toll. In Proceedings of the Sixth international workshop on Computing education research (ICER '10). ACM, New York, NY, USA, 77-86.

[4] Gee, J. Learning by Design: Good video games as learning machines. E-learning, 2-1, 2005, pages 5-16.

[5] Prensky, M. Digital Game-based learning. ACM Computer in Entertainment, 1-1, 2003.

[6] Barnes, T., Powell, E., Chaffin, A. And Lipford, H. Game2Learn: improving the motivation of CS1 students. In Proceedings of the 3rd international conference on Game development in computer science education (GDCSE '08). ACM, New York, NY, USA, 1-5.

[7] Muratet, M., Torguet, P., Jessel, J., and Viallet, F. Towards a serious game to help students learn computer programming. International Journal of Computer Technology, 2009.

[8] Papastergiou, M. Digital game-based learning in high school computer science education: Impact on educational effectiveness and student motivation. Computer Education, 2009, 52-1.

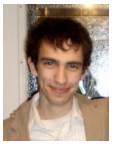
[9] Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., and Resnick, M. Scratch: a sneak preview. Creating, Connecting and Collaborating through Computing, 2004. Proceedings. Second International Conference on , vol., no., pp. 104- 109, 29-30 Jan. 2004

[10] Silver, J. MIT Media Lab: Lifelong Kindergarten. http://www.media.mit.edu/research/groups/lifelong-kindergarten

[11] McDowell, C., Werner, L., Bullock, H.E. and Fernald, J. 2006. Pair programming improves student retention, confidence, and program quality. Commun. ACM 49, 8 (August 2006), 90-95.

[12] Shiffman, D. Processing QRCode Library. http://www.shiffman.net/p5/pqrcode/

BIOGRAPHY



Nicholas Masso is an undergraduate at Miami University majoring in psychology and interactive media studies. He is a Harrison Scholar, a scholarship awarded to the most promising undergraduates at Miami. He combines design, conceptual art, and human factors to create novel physical interactions between people and technology.

Lindsay Grace is faculty at Miami University. He holds a dual appointment between the Armstrong Institute for Interactive Media Studies and the Miami University School of Fine Arts. He earned the Masters in Computer Information Systems and the Bachelor of Arts in English (drama) from Northwestern University. He also earned the Masters of Fine Art from the University of Illinois, Chicago's Electronic Visualization Laboratory. Lindsay spreads his efforts among scholarly writing and research, creative exposition of digital arts, and quality teaching in interactive design.