MazeStar: A Platform for Studying Virtual Identity and Computer Science Education

Dominic Kao
Massachusetts Institute of Technology
77 Massachusetts Ave
Cambridge, MA 02139
dkao@mit.edu

D. Fox Harrell
Massachusetts Institute of Technology
77 Massachusetts Ave
Cambridge, MA 02139
fox.harrell@mit.edu

ABSTRACT

This paper presents an overview of the *MazeStar* platform for Computer Science education. *MazeStar* is both a game (*Mazzy*) that teaches programming concepts like loops and conditionals, and a game editor which allows players to create and share their own game levels. By playing and creating, players are using computing concepts (e.g., block structuring, parallelism, etc.) and computing practices (e.g., debugging, iterative prototyping, etc.). To date the *MazeStar* platform has been used in controlled user studies involving > 10,000 participants. Here, our goal is to detail the different components of the *MazeStar* platform, and how we have/are leveraging these components to study the interplay of education, games/game-making, and virtual identity.

CCS CONCEPTS

• **Human-centered computing → Human computer interaction (HCI)**; • **Applied computing → Education**;

KEYWORDS

Educational platforms, educational games, virtual identity, avatars

ACM Reference format:
Dominic Kao and D. Fox Harrell. 2017. *MazeStar*: A Platform for Studying Virtual Identity and Computer Science Education. In *Proceedings of FDG'17, Hyannis, MA, USA, August 14-17, 2017,* 6 pages.
DOI: .1145/3102071.3116221

1 INTRODUCTION

The well-known theory of constructionism, that building knowledge is most effective through construction of shared artifacts [47], is having something of a heyday in popular media forms. Today, we are witnessing a veritable rise of videogames and virtual environments that could be considered "constructionist" platforms. For instance, games like *The Elder Scrolls V: Skyrim*, *Minecraft*, and *LittleBigPlanet 3* all have or have evolved to have "modding" (user-driven game modifications) at their core (e.g., [49]). *Counter-Strike*, *Team Fortress*, *League of Legends*, and *Dota 2* are all popular games that themselves are direct descendants of mods. *Roblox* is a game marketed for children and teenagers aged 8-18 with 15 million monthly active users (as of July 2016 [53]) and has extensive affordances for creating levels and avatars. While game modding has been practiced since the 1980s [48], systems and processes have gradually been put in place by developers to both lower the barrier to entry and to incentivize the act of building. Games like *StarCraft*, *Warcraft*, *Trackmania* (and many others) all shipped with official level editors, and could be reskinned using either official or community-generated tools. Games like *The Sims* and virtual worlds like *Whyville* and *Second Life* have all had a significant metagame around making, e.g., "face-parts" in *Whyville* [22], animated textures in *Second Life* [55], clothes in *The Sims* [19], etc. While platforms like the Steam Workshop have dominated the commercial realm of user-generated content (as of 2016 supporting almost 500 titles [60]), educational platforms for Computer Science education rooted in constructionism are emerging [7, 8, 34].

In this paper we discuss the *MazeStar* platform, a platform that both teaches computing and computing-related practices through gameplay and game-making, but also serves as an experimental testbed. Specifically, the MazeStar platform is a novel contribution along three axes:

- An **experimental setting** for studying the impacts of virtual identity and other phenomena, along with robust data tracking and a number of possibilities for virtual identity creation, with over 10,000 participants having taken part in controlled studies.
- Within a **framework of maze-solving**, combines gameplay and game making–extending to a wide array of computing concepts from basic programming like loops and conditionals, to human-computer interaction, design, and iterative prototyping, to more theoretical topics like search algorithms, all with heavily streamlined features like built-in image search and automatic website creation for sharing made games.
- A **focus on virtual identity** as a key component to students' trajectories as computer science learners.

The remainder of this paper is structured as follows. In Section 2, we discuss our theoretical framework. In Section 3, we outline and describe the different components of the *MazeStar* platform. In Section 4, we give an overview of our crowdsourced studies and associated published findings, as well as some of our work in progress that is ongoing in Boston classrooms. In Section 5, we make concluding remarks.

FDG'17, Hyannis, MA, USA

978-1-4503-5319-9/17/08…$15.00
DOI: .1145/3102071.3116221

2 THEORETICAL FRAMEWORK

2.1 AIR Project

The Advanced Identity Representation (AIR) project [17] constitutes approaches to analyzing and designing social categorization systems across diverse forms of virtual identity ranging from avatars to social media profiles. It is grounded in approaches to cognitive categorization and social classification from cognitive linguistics and sociology, along with HCI approaches for implementing and evaluating results. The AIR project [17] identifies several common limitations in computational systems, such as "Attributes are reduced to statistics," "Community membership is a binary model," etc. Many of these run parallel to the ones in educational systems. The AIR project is one lens through which we begin to critically analyze these media.

2.2 Virtual Identity

There is an abundance of work that demonstrates that avatars (or "blended identities" [17] as in practice users cognitively project aspects of their physical-world identities onto them) are impactful on a variety of student measures, such as learning, task performance, and engagement [3–5, 16, 21, 28, 36, 50, 54]. For instance:

- The "role model effect" is one in which participants using a famous role model avatar (particularly famous scientists) led to improved educational outcomes [24, 31].
- That "successful likeness" representations, avatars that are abstract during debugging and failure, but likenesses of the player during success, are especially effective [28].
- That abstract avatars (such as a shape) can provide several suggested benefits as compared to other avatar types: a) less embellished, therefore less distracting, b) greater detachment, therefore greater dissociation from unfavorable outcomes, and c) less identity features, therefore less likely to trigger phenomena such as stereotype threat [32].

A well-known phenomenon is that of the "Proteus effect", which describes an individual's tendency to conform to behavior typically associated with how an avatar appears [65]. Two of the earliest studies found that participants with taller avatars were more aggressive, and that participants with avatars they deemed more attractive were more confident (this persists to some extent after leaving the virtual world [66]). Through crowdsourced studies in our *MazeStar* platform environment, we have studied more than 10,000 individual users and how different virtual identities can either empower or disempower users. Some of these will be summarized in Section 4.

2.3 Constructionism

Constructionism is a theory of learning in which learners construct mental models for understanding the world. Cornerstones of this theory include student-based discovery learning, whereby students learn via bridges to their pre-existing knowledge and learning through production of shared artifacts [47]. Seymour Papert said of learning that it "happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity, whether it's a sand castle on the beach or a theory of the universe" [47]. Papert felt strongly that the "instructionist" approach towards education (similar to what Freire would term a "banking" concept of education [14]), which involved explicit verbal instruction, was a deficient educational approach.

In the seminal book *Mindstorms*, Papert describes "Turtle Geometry", an environment for programming an icon of a turtle trailing lines across a computer display, as drawing upon the child's preexisting pleasure and knowledge of motion. Papert described early experience with "Turtle Geometry" as a good way to "get to know" more formalized subjects through some of its powerful ideas [45]. This is similar to what Lave and Wenger term "legitimate peripheral participation" [39], what Crowley and Jacobs consider "islands of expertise" [12], and what Shaffer terms an "epistemic frame" [56]–all of which describe how beginners can slowly become experts, with their expertise extending far beyond the boundaries and consequences of the original activities. Constructionism places a heavy emphasis on breaking knowledge up into "mind-size" bites–similar to James Gee's "incremental principle" [15]–making knowledge more communicable, assimilable and "constructable" [45]. Almost three decades later, Papert's original ideas on constructionism remain relevant and have become ubiquitous in how learning theorists and educators aim to revamp traditional teaching methods.

2.4 Computational Thinking

Computational thinking is most widely understood through Cuny, Snyder, and Wing's definition [63]:

> **Computational Thinking** is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent.

Historically, computational thinking was a term first used by Seymour Papert in 1980 [44, 46], and in the ensuing decades has taken on different aliases albeit with philosophically similar definitions–computational literacy, which focused more on computing as a medium for exploration [13], and procedural literacy, which focused more on computational thinking in the context of new media art and design [6, 42, 57]. In a context of design-based activites in Scratch, Brennan and Resnick define their own computational thinking framework: computational concepts (e.g., actual programming concepts), computational practices (e.g., practices such as debugging and iterative development), and computational perspectives (e.g., perspectives on computation and the world more generally) [7]. We leverage this framework for describing how *MazeStar* teaches computational thinking.

2.5 Other Systems/Games That Teach Computer Science

Other games and systems have been used to teach programming and/or CS principles. Non-exhaustively, these include the Logo programming language and associated turtle graphics [41], the Scratch environment [52], Alice [11] and Storytelling Alice [35], NetLogo [62], MIT App Inventor [64], Gidget [40], LightBot [1], CodeCombat [2], BOTS [20], RoboBuilder [61], Greenfoot [38], AgentSheets and AgentCubes [51], Code.org exercises [10], the Arduino [8], Kodu Game Lab [59], Game Maker [9, 43], Gogo Boards [58], the STELLA programming language [37], and others [18].

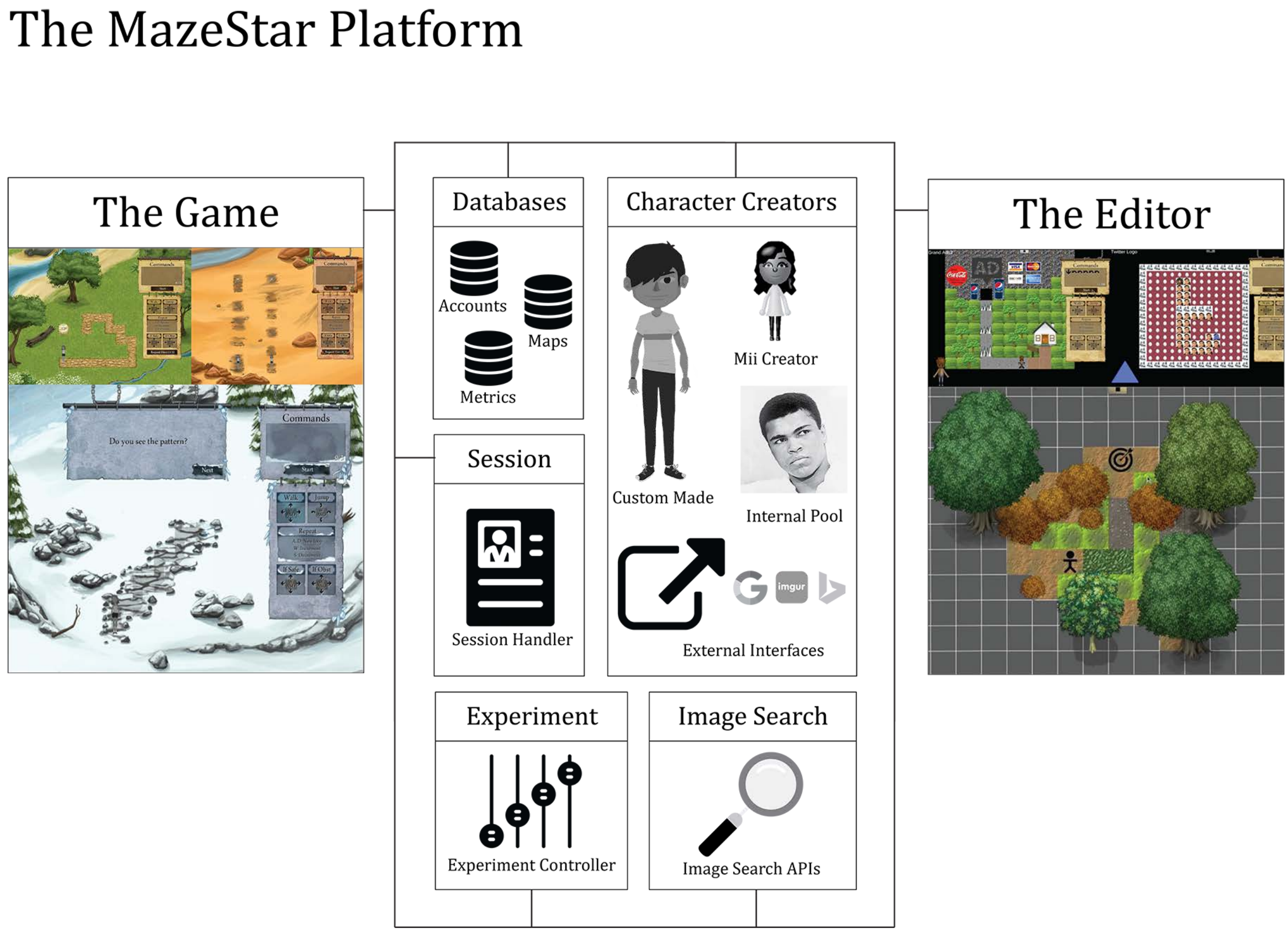


Figure 1: *MazeStar* platform components.

3 MAZESTAR

In this section, we describe the *MazeStar* platform in more depth. See Figure 1 for an overview. We begin by describing the game (*Mazzy*), then the editor, and finally some of the more important shared components between the two.

3.1 The Game

The *MazeStar* platform contains a STEM learning game called *Mazzy* [25][1][2]. *Mazzy* is a game in which players solve levels by creating short computer programs. In total, there are 12 levels in this version of *Mazzy*. Levels 1-5 require only basic commands. Levels 6-9 require using loops. Levels 10-12 require using all preceding commands in addition to conditionals. *Mazzy* has been used previously as an experimental testbed for evaluating the impacts of avatar type on performance and engagement in an educational game [23, 24, 26–31]. See the footnote for gameplay videos.

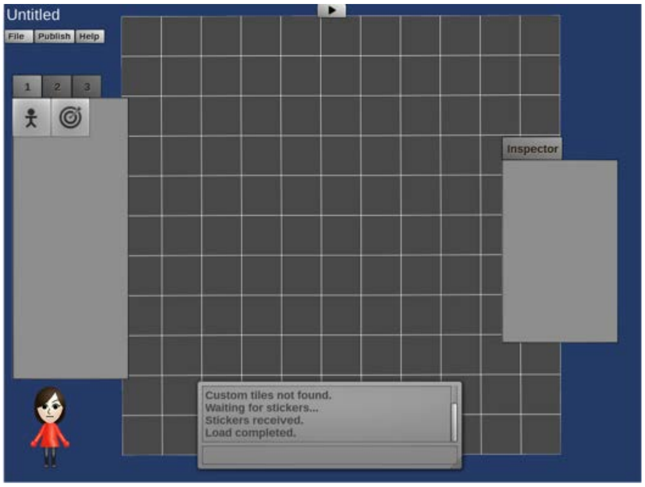
3.2 The Editor

At a high-level, the editor allows players to create their own *Mazzy* game levels, and then share those levels through links and automatically generated webpages. Each map consists of a grid of tiles, each of which can be textured separately and modified logically to be a safe or unsafe tile for the player to step on. The maps can be any size (from 1x1 to any size that fits in browser memory). See Figure 2.

3.2.1 Editor Basics. Within the editor, players move the view of the current working map using W, A, S, and D on the keyboard. They can save maps and open previously saved maps. On the left hand side is a panel that allows players to add different elements to the maps. In the first tab of this panel, players can set the start and goal location of the player (the start being where the player will initially spawn, the goal location being where they intend the player to try to reach, though the latter is not necessary for playing the map). The second tab contains textures for the tiles themselves, which can be placed on each of the grid squares.

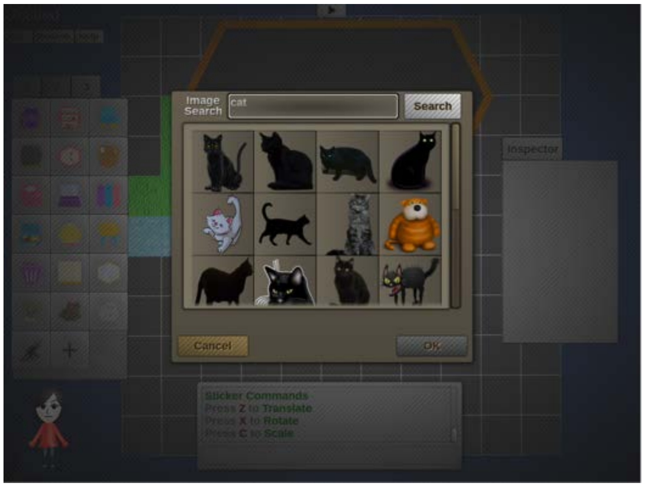
3.2.2 Stickers. The third tab contains textures for stickers, which are aesthetic images that appear overtop of the grid and do not

[1] Current Mazzy Version: http://youtu.be/n2rR1CtVal8
[2] Older Mazzy Version: http://youtu.be/j0TI4MH2rsY



```
Custom tiles not found.
Waiting for stickers...
Stickers received.
Load completed.
```

Figure 2: Blank 11x11 map in the editor.



```
Sticker Commands
Press Z to Translate
Press X to Rotate
Press C to Scale
```

Figure 3: Searching for the image "cat".

affect the game logically. These stickers can be translated, rotated, and rescaled using the Z, X, and C keys respectively to switch modes.

3.2.3 Custom Images. Players can not only use the tiles and stickers that are pre-loaded with the editor, but also search for images and import them directly. See Figure 3.

3.2.4 Testing a Map. Maps are periodically saved automatically to prevent data-loss in the event that the user should accidentally quit the browser without saving or in the event of a CPU crash. To test their maps, players can click on the play icon at the top-center of the screen. This simulates playing the map that they've created.

3.2.5 Sharing a Map. When satisfied with their map, players can then share their map either using: a) an automatically generated tinyURL link, or b) an automatically generated website. In the latter case, the website is a permanent record of their map and does not change (unless the user re-generated the website in which case the old one is overwritten). The automatic website generation involves the user filling out a dialog box with the entries "About Me," "Artist's Statement," and "Level Instructions," then a website is automatically generated via browser-side communication with our server using PHP. In both cases, using a link or webpage, visiting players can play the created map directly–similar to sharing a file on Google Drive or Dropbox publicly.

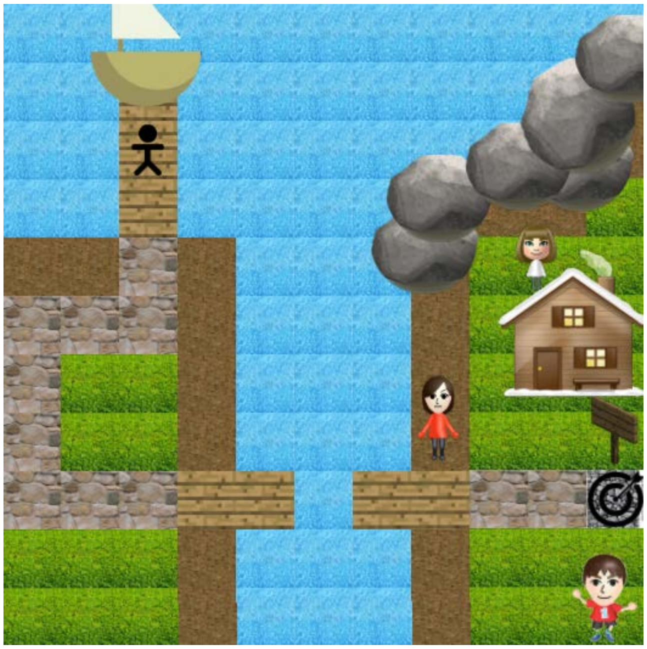


Figure 4: "HomeRoad" features a variety of assets. M/29.

Figure 5: "Garden" is a long one. M/21.

3.3 Example Player-Created Maps

In this section, we share 4 Amazon Mechanical Turk player-created maps. Average creation time for these 4 maps was 22.1 minutes (SD = 22.9). Players played Mazzy for as long as they liked, then were given a brief tutorial (mean time to complete the tutorial was 3.5 minutes, SD = 1.8) on how to use the editor. The tutorial introduced basic functionalities of the editor: panning/zooming, play-testing, searching for tiles/stickers, sticker manipulation using scaling/rotation/translation, and creating blank maps. In their version of the editor, no default images were provided for tiles/stickers (all images as part of their maps are searched for by players themselves through the editor's image searching functionality). These maps were selected on the basis that they appeared be effective and/or creative. See Figures 4, 5, 6, and 7. The player-given map name, gender, and age are in each caption.

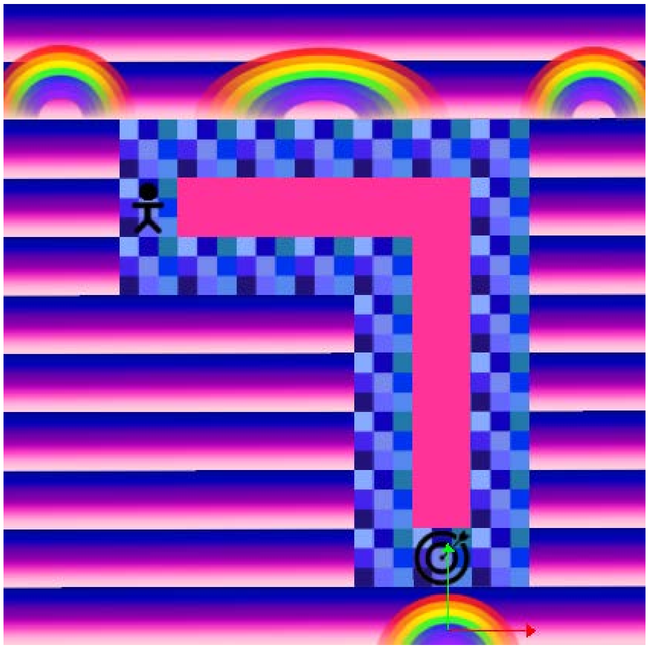


Figure 6: "jennymap" is colorful. F/29.

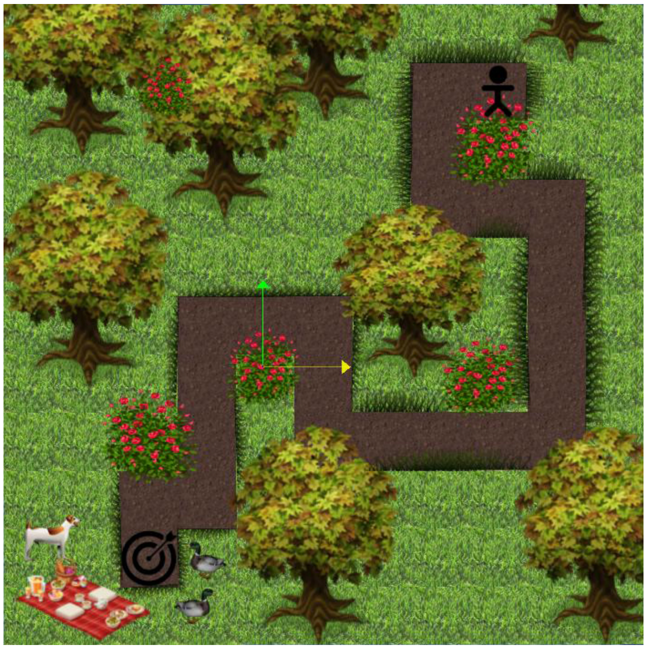


Figure 7: "Picnic Time" creatively uses stickers to make the path "fuzzier". F/24.

4 EXPERIMENTAL OVERVIEW

In this section we describe at a high-level the experiments we have conducted on the MazeStar platform.

4.1 Crowdsourced Studies

We have systematically explored the impacts of different avatar types on users in crowdsourced studies with over 10,000 participants. Our studies have revealed that avatars can support, or harm, student performance and engagement. A few notable trends are: 1) 'role model' avatars (in particular scientist avatars) are effective [24], 2) 'likeness' avatars (avatars in a user's likeness) are not always effective, 3) simple 'abstract' avatars (such as geometric shapes) are especially effective when the player is undergoing failure, e.g., 'debugging' [28]. We have also studied other topics such as the impact of level of embellishment in game backgrounds on performance, engagement, and self-efficacy in programming [33]. A full overview of the methods used in these studies is not possible here, so we ask interested readers to refer to the citations.

4.2 Classroom Studies

As part of an NSF-funded project, we have conducted a total of 4 workshops with public high school students in Cambridge and Boston in the last year and a half. These involved exploring the intersections of student identity (both social and virtual) and computer science learning, with a focus on underrepresentation in STEM. Students learned computational concepts (loops, conditionals, search algorithms, etc.), computational practices (the HCI design-create-evaluate cycle in increasingly complex iterations, debugging, etc.), and computational perspectives (the intersection of computational identity and themes of importance to them such as bullying, digital privacy, etc.). In the workshops, students both played *Mazzy* and created levels in *MazeStar* (starting from paper prototypes iteratively refining them within our platform). Students also discussed topics of importance to them and connected these topics via their constructed artifacts in *MazeStar*. Data analyses from our workshops is ongoing.

5 CONCLUSION

In this paper, we discussed the *MazeStar* platform. The *MazeStar* platform makes the following novel contributions as outlined in the introduction: 1) As an **experimental setting**, 2) As a **framework of maze-solving** which is both simple to introduce to students, but also highly extensible, and 3) A **focus on virtual identity**.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable feedback. This work is supported by NSF STEM+C Grant 1542970 and a Natural Sciences and Engineering Research Council of Canada (NSERC) fellowship.

REFERENCES

[1] 2008. LightBot (http://lightbot.com/). (2008).

[2] 2016. CodeCombat. (2016). https://codecombat.com/

[3] Ivon Arroyo, Beverly Park Woolf, James M. Royer, and Minghui Tai. 2009. Affective gendered learning companions. In *Frontiers in Artificial Intelligence and Applications*, Vol. 200. 41–48. DOI: https://doi.org/10.3233/978-1-60750-028-5-41

[4] Jeremy N. Bailenson, Jim Blascovich, and Rosanna E. Guadagno. 2008. Self-representations in immersive virtual environments. *Journal of Applied Social Psychology* 38, 11 (2008), 2673–2690.

[5] Al Baylor and Yanghee Kim. 2004. Pedagogical agent design: The impact of agent realism, gender, ethnicity, and instructional role. *Intelligent Tutoring Systems* 1997 (2004), 592–603. DOI: https://doi.org/10.1007/978-3-540-30139-4_56

[6] I Bogost. 2005. Procedural literacy: Problem solving with programming, systems, and play. *Journal of Media Literacy* (2005), 32–36.

[7] Karen Brennan and Mitchel Resnick. 2012. New frameworks for studying and assessing the development of computational thinking. *Annual American Educational Research Association meeting, Vancouver, BC, Canada* (2012), 1–25.

[8] Leah Buechley, Mike Eisenberg, and Jaime Catchen. 2008. The LilyPad Arduino: Using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. *CHI '08 Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems* (2008), 423–432.

[9] Kajal Claypool and Mark Claypool. 2005. Teaching software engineering through game design. In *ACM SIGCSE Bulletin*, Vol. 37. ACM, 123–127.

[10] Code.org. 2014. Code.org. (2014). http://code.org

[11] Stephen Cooper, Wanda Dann, and Randy Pausch. 2000. Alice : a 3-D Tool for Introductory Programming Concepts. *Journal of Computing Sciences in Colleges* 15, 5 (2000), 107–116.

[12] Kevin Crowley and Melanie Jacobs. 2002. Building Islands of Expertise in Everyday Family Activity. *Learning Conversations in Museums* (2002), 1–23.

[13] Andrea DiSessa. 2000. Changing Minds: Computers, Learning, and Literacy. *Journal of Teacher Education* 60, 1 (2000), 38–51. DOI:https://doi.org/Article

[14] Paulo Freire. 2000. *Pedagogy of the Oppressed*.

[15] James Paul Gee. 2003. What video games have to teach us about learning and literacy. *Computers in Entertainment* 1, 1 (oct 2003), 20.

[16] Rosanna E Guadagno, Jim Blascovich, Jeremy N Bailenson, and Cade McCall. 2007. Virtual humans and persuasion: The effects of agency and behavioral realism. *Media Psychology* 10, 1 (2007), 1–22. DOI:https://doi.org/10.108/15213260701300865

[17] D. Fox Harrell. 2010. Toward a Theory of Critical Computing: The Case of Social Identity Representation in Digital Media Applications. *CTheory* (2010).

[18] Casper Harteveld, Gillian Smith, Gail Carmichael, Elisabeth Gee, and Carolee Stewart-Gardiner. 2014. A Design-Focused Analysis of Games Teaching Computer Science Methodology for Examining Games that Teach Computer Science. (2014).

[19] J C Herz. 2005. Harnessing the hive. *Creative industries* (2005), 327–341.

[20] Andrew Hicks, Barry Peddycord III, and Tiffany Barnes. 2014. Building games to learn from their players: Generating hints in a serious game. In *International Conference on Intelligent Tutoring Systems*. Springer, 312–317.

[21] Amy M. Johnson, Matthew D. Didonato, and Martin Reisslein. 2013. Animated agents in K-12 engineering outreach: Preferred agent characteristics across age levels. *Computers in Human Behavior* 29, 4 (2013), 1807–1815.

[22] Y. B. Kafai, D. A. Fields, and M. S. Cook. 2010. Your Second Selves: Player-Designed Avatars. *Games and Culture* 5, 1 (2010), 23–42.

[23] Dominic Kao and D. Fox Harrell. 2015. Exploring construction, play, use of virtual identities in STEM learning. *Jean Piaget Society Annual Conference* (2015).

[24] Dominic Kao and D. Fox Harrell. 2015. Exploring the Impact of Role Model Avatars on Game Experience in Educational Games. *The ACM SIGCHI Annual Symposium on Computer-Human Interaction in Play (CHI PLAY)* (2015).

[25] Dominic Kao and D. Fox Harrell. 2015. Mazzy: A STEM Learning Game. *Foundations of Digital Games* (2015).

[26] Dominic Kao and D. Fox Harrell. 2015. Toward Avatar Models to Enhance Performance and Engagement in Educational Games. *IEEE Computational Intelligence in Games* (2015).

[27] Dominic Kao and D. Fox Harrell. 2015. Toward Evaluating the Impacts of Virtual Identities on STEM Learning. *Foundations of Digital Games* (2015).

[28] Dominic Kao and D Fox Harrell. 2016. Exploring the Effects of Dynamic Avatars on Performance and Engagement in Educational Games. *Games+Learning+Society (GLS 2016)* (2016).

[29] Dominic Kao and D. Fox Harrell. 2016. Exploring the Effects of Encouragement in Educational Games. *Proceedings of the 34th Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems (CHI 2016)* (2016).

[30] Dominic Kao and D. Fox Harrell. 2016. Exploring the Impact of Avatar Color on Game Experience in Educational Games. *Proceedings of the 34th Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems (CHI 2016)* (2016).

[31] Dominic Kao and D. Fox Harrell. 2016. Toward Understanding the Impacts of Role Model Avatars on Engagement in Computer Science Learning. *The annual meeting of the American Educational Research Association (AERA)* (2016).

[32] Dominic Kao and D. Fox Harrell. 2017. Embellishment & Effects: Seduction by Style. In *Avatars, Assembled: The Sociotechnical Anatomy of Digital Bodies*, Jaime Banks (Ed.). Peter Lang, Digital Formations Series.

[33] Dominic Kao and D. Fox Harrell. 2017. Toward Understanding the Impact of Visual Themes and Embellishment on Performance, Engagement, and Self-Efficacy in Educational Games. *The annual meeting of the American Educational Research Association (AERA)* (2017).

[34] Dominic Kao, D. Fox Harrell, Chong-U Lim, S. Veeragoudar Harrell, Maya Wagoner, and Helen Ho. 2016. Highlighting MazeStar: A Platform for Studying Avatar Use in Computer Science Learning Environments. *Games+Learning+Society (GLS 2016)* (2016).

[35] Caitlin Kelleher, Randy Pausch, and Sara Kiesler. 2007. Storytelling alice motivates middle school girls to learn computer programming. *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '07* (2007), 1455.

[36] Yanghee Kim and Amy L. Baylor. 2006. Pedagogical agents as learning companions: The role of agent competency and type of interaction. *Educational Technology Research and Development* 54, 3 (2006), 223–243.

[37] Kyu Han Koh, Ashok Basawapatna, Vicki Bennett, and Alexander Repenning. 2010. Towards the Automatic Recognition of Computational Thinking for Adaptive Visual Language Learning. *Visual Languages . . .* (2010), 59–66.

[38] Michael Kölling. 2010. The greenfoot programming environment. *ACM Transactions on Computing Education (TOCE)* 10, 4 (2010), 14.

[39] Jean Lave and Etienne Wenger. 1991. Situated learning: Legitimate peripheral participation. *Learning in doing* 95 (1991), 138. arXiv:arXiv:1011.1669v3

[40] Michael J. Lee, Andrew J. Ko, and Irwin Kwan. 2013. In-game assessments increase novice programmers' engagement and level completion speed. *Proceedings of the ninth annual international ACM conference on International computing education research - ICER '13* (2013), 153. DOI:https://doi.org/10.1145/2493394.2493410

[41] Logo Foundation. 2017. Logo (http://el.media.mit.edu/logo-foundation/). (2017).

[42] Michael Mateas. 2005. Procedural literacy: educating the new media practitioner. *On the Horizon* 13, 2 (2005), 101–111.

[43] Mark Overmars. 2004. Teaching computer science through game design. *Computer* 37, 4 (2004), 81–83.

[44] S. Papert. 1980. *Mindstorms.* Vol. 15. 177 –181 pages. DOI:https://doi.org/10.1111/j.1467-8721.2006.00431.x

[45] Seymour Papert. 1993. Mindstorms. (1993).

[46] Seymour Papert. 1996. An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning* 1, 1 (1996).

[47] S Papert and Idit Harel. 1991. Situating Constructionism. *Constructionism* (1991).

[48] Hector Postigo. 2003. From Pong to Planet Quake: Post-Industrial Transitions from Leisure to Work. *Information, Communication & Society* 6, 4 (2003), 593–607.

[49] Hector Postigo. 2016. Modification. In *Debugging Game History: A Critical Lexicon*. MIT Press, 325.

[50] Jean A. Pratt, Karina Hauser, Zsolt Ugray, and Olga Patterson. 2007. Looking at human-computer interface design: Effects of ethnicity in computer agents. *Interacting with Computers* 19, 4 (2007), 512–523.

[51] Alex Repenning. 1993. Agentsheets: a tool for building domain-oriented visual programming environments. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*. ACM, 142–143.

[52] M Resnick and John Maloney. 2009. Scratch: programming for all. *Communications of the . . .* (2009). http://dl.acm.org/citation.cfm?id=1592779

[53] Roblox. 2016. We're Building the Future of Entertainment. (2016). http://corp.roblox.com/careers

[54] Rinat B. Rosenberg-Kima, E. Ashby Plant, Celestee E. Doerr, and Amy Baylor. 2010. The influence of computer-based model's race and gender on female students' attitudes and beliefs towards engineering. *Journal of Engineering Education* (2010), 35–44. DOI:https://doi.org/10.1002/j.2168-9830.2010.tb01040.x

[55] Michael Rymaszewski. 2007. *Second life: The official guide*. Vol. 2. John Wiley & Sons.

[56] David Williamson Shaffer. 2006. Epistemic frames for epistemic games. *Computers & Education* 46, 3 (2006), 223–234.

[57] B A Sheil. 1980. Teaching procedural literacy (Presentation Abstract). In *Proceedings of the ACM 1980 annual conference*. ACM, 125–126.

[58] Arnan Sipitakiat, Paulo Blikstein, and David P Cavallo. 2004. GoGo board: augmenting programmable bricks for economically challenged audiences. In *Proceedings of the 6th international conference on Learning sciences*. International Society of the Learning Sciences, 481–488.

[59] Kathryn T Stolee and Teale Fristoe. 2011. Expressing computer science concepts through Kodu game lab. In *Proceedings of the 42nd ACM technical symposium on Computer science education*. ACM, 99–104.

[60] Valve. 2016. Steam Workshop. (2016). http://steamcommunity.com/workshop/

[61] David Weintrop and Uri Wilensky. 2014. Program-to-play video games: Developing computational literacy through gameplay . (2014), 1–7. http://ccl.northwestern.edu/papers/2014/GLS-2014final.pdf

[62] Uri Wilensky. 1999. NetLogo. (1999).

[63] Jeannette M Wing. 2006. Computational thinking. *Commun. ACM* 49, 3 (2006), 33–35.

[64] David Wolber. 2011. App inventor and real-world motivation. In *Proceedings of the 42nd ACM technical symposium on Computer science education*. ACM, 601–606.

[65] Nick Yee and Jeremy Bailenson. 2007. The Proteus Effect: The Effect of Transformed Self-Representation on Behavior. *Human Communication Research* 33, 3 (jul 2007), 271–290. DOI:https://doi.org/10.1111/j.1468-2958.2007.00299.x

[66] Nick Yee, Jeremy N Bailenson, and Nicolas Ducheneaut. 2009. *The Proteus Effect Implications of Transformed Digital Self-Representation on Online and Offline Behavior*. Vol. 36. 285–312 pages. DOI:https://doi.org/10.1177/0093650208330254