Patterns of play: Understanding computational thinking through game design

Gabriella Anton, Shannon Harris, Amanda Ochsner, Allison Salmon, Meagan Rothschild,
Matthew Berland, Kurt Squire, University of Wisconsin-Madison
gabby.anton@gmail.com, shannonharris.research@gmail.com, allison@learninggamesnetwork.org, amanda.ochsner@gmail.com, meagan.rothschild@gmail.com, mberland@wisc.edu, kdsquire@wisc.edu

Abstract: While computational thinking and computer science skills are among the most valuable, sought after skills in the digital age, they are also some of the most challenging to learn. Because a vast majority of youth play videogames, using game design as an entry point for programming skills shows promise for encouraging more young learners to pursue careers in computer science and technology (Hayes & Games, 2008). Our research team at the University of Wisconsin-Madison has developed a game design curriculum around the game design software *Kodu*, fostering interest-driven learning of computational thinking. Based on data from 76 participants, results show kode complexity increases with time spent designing in *Studio K*, regardless of gender or prior interest in game design or programming. These preliminary findings beg deeper analysis of the patterns of play that may facilitate deeper, more meaningful acquisition of core computational and computer science concepts.

Introduction

Computational devices are becoming increasingly more integrated into myriad facets of our lives. With billions of smartphones, tablets, laptops, and other digital devices, the masses are able to readily access computational powers that were previously only available to mainframes and supercomputers, if even thought to be possible at all. The prevalence of these devices in nearly all facets of modern life necessitates that more people acquire the navigational and computational skills that can enable them to most successfully participate, and even innovate, in society. To accomplish this, it is imperative that more people become literate with digital technologies. This kind of fluency with computation requires more than knowing how to use any particular application or operating system, or even how to program. Instead it requires knowing how to abstract from situations and think computationally (Wing, 2006). People need to understand what computation can do easily and what it cannot do, develop good intuitions about how computation operates within domains of practice (from social software to personalized medicine applications), and be able to use computation to achieve existing goals, and also articulate and execute, new ones. At the same time, the real power of computational thinking lies in that protean power of the computer: creation. If understanding computation is valuable, using computation to build something personally meaningful is quite possibly the best way to get there. Computational thinking (National Academies, 2011; Wing, 2006) describes an ability to answer the question, “What can I build to understand and/or solve this problem?”

Unfortunately, becoming proficient with computational thinking can be difficult, and there are many intimidating challenges that arise for people looking to develop these skills. There is very little instruction in the United States that teaches students how to cross-apply computational thinking. While there are several paths to acquiring computational skills, most involve learning complex computer programming logic and languages. Many students do not have access to computer science courses in high school, and many existing courses emphasize disconnected abstract principles that create powerful barriers to entry (Camp, 1997). Pulimood and Wolz suggest such barriers could be addressed through a pedagogical shift that includes 1) authentic inquiry through creative design; 2) collaborative work mirroring modern media practices; and 3) multidisciplinary approaches to content (2008).

Game Design as the Solution

One way to address the accessibility issues and pedagogical shifts described above is to relay the material through popular digital technologies that can facilitate the framing of computational concepts in more immediately approachable and meaningful ways. Learning environments that are rich in opportunities for students to connect their background knowledge and interests help to anchor and contextualize the meaning of new knowledge, and thus lead to a deeper, more robust form of transfer for students constructing new schemas and conceptual understanding (Bransford & Schwartz, 1999). Videogames, through their broad dissemination, have become more relevant than ever before in youth’s lives, regardless of gender. Games hold great potential to engage underrepresented groups from very early ages, and to help them learn key practices and concepts necessary to effectively pursue computer science careers later in their lives (Hayes and Games, 2008). Games effectively become a medium for change, with interest in the medium leading to an array of literacy practices and reciprocal apprenticeships which can be further applied to facilitate the understanding of concepts in core classes (Gee, 2003; Steinkuehler, 2007;

Black & Steinkuehler, 2009; Squire & Jenkins, 2003).

Similarly, studies of youth designing and developing their own games have showed gains in a wealth of literacy and critical thinking practices (Peppler, Diazgranados, & Warschauer, 2010), and suggest that these practices facilitate the development of computational thinking skills like logic, debugging, and algorithm design (Berland & Lee, 2011). As other 3D programming and design environments, such as *Alice* (Pausch et al., 1995), *Toontalk* (Kahn, 1996), *AgentSheets* (Repenning, 1993), and *Scratch* (Maloney et al., 2010), have shown, a) providing an ever-growing reference collection of fun games to be deconstructed and built upon; b) enabling peers to help one another to progress in skill from beginner to expert; and c) creating safe spaces to learn to program together without fear of harassment or judgment (Resnick et al., 2009) can be powerful supports for learning to think computationally.

Studio K

Building off the successes of previous programming environments and design environments, as well as curricula on game design, our research team at University of Wisconsin-Madison has developed a similar experience around the 3D visual-based programming software, *Kodu*, integrating game design and programming curriculum within an online community to foster interest-driven learning (see Figure 1).

**Figure 1: On the left, the Studio K Mission layout webpage.
In the middle, the initial Kodu interface and design environment.
On the right, programming tiles in Kodu, 3D visual-based programming software.**

Working in *Studio K*, young designers play, revise, and create games in *Kodu* as they navigate a series of design challenge missions targeting fundamental concepts distilled from the rich literature on videogame design (see Figure 1). Unlike other visual programming languages, *Kodu* enables users to quickly and easily create seemingly complex, aesthetically pleasing games with only minimal knowledge of programming. By inversing the goal of the program from acquiring programming skills to developing games, *Kodu* can enable deeper engagement and more meaningful connections with the content, using interest in and enjoyment of videogames to promote use of increasingly more complex programming and computational thinking skills. The *Studio K* curriculum orients to these goals, using playful, open-ended challenges to ensure that students acquire content in individually relevant and engaging ways.

Methods

The data set for this paper represents learning analytic data from four club settings with a total of 76 students. Each club ranged from 2-4 weeks of active participation with *Studio K*, with 45-90 minutes spent per session. Data are derived from logged interactions with *Studio K* curricula and a modified version of Kodu used to build our ADAGE telemetry architecture (Halverson & Owen, in press). Logged and survey data were analyzed using ANOVAs and linear regression models. Descriptive statistics and correlation data were examined as well. Survey data were collected from students before they began using *Kodu* and the *Studio K* curriculum.

Key survey variables analyzed included sex, age, and 5-point ranged Likert scale statements measuring interest and identity components. These statements included: “I hope to get a job doing something with video games,” “It’s important to know how to program,” “I would be interested in taking more classes on programming,” and “It’s important for me to know how to design video games” while asking users to self-report their answers ranging on scales from strongly agree to strongly disagree (see Table 1, below).

	Strongly Disagree (1)	Disagree (2)	Undecided (3)	Agree (4)	Strongly Agree (5)
I hope to get a job doing something with video games.					
It's important to know how to program.					
I would be interested in taking more classes on programming.					
… design video games					

Table 1: Pre-Curriculum Identity and Interest Survey

Key logging data variables harvested from in-world actions included: timestamps, time spent on specific tasks, number of pages of kode (code in *Kodu)* developed, number of revisions made to kode, frequencies of actor switching (changing from one in-world object to another), unique sensor counts (kode developed that included passive coding concepts), unique action counts (kode developed that included active coding concepts), and kode complexity, which was operationalized by aggregating the unique sensor and action paired variables. Variables were parsed using time stamps; anonymized user IDs; and a combination of time stamps, user IDs, and user plus the specific in-game kodu being manipulated at a given time.

Results

Analysis of ADAGE in-world telemetry data reveal that measures of complexity and game/program quality are not significantly correlated with external measures of interest (See Table 2, below). Specifically, measures of code complexity, revisions, and the number of objects programmed are not significantly correlated with the interest measures in our survey (*likelihood of future courses on programming*, *importance of learning to program*, and *interest in having a job related to game design*).

			revision count	actor switching	Kode_Complexity
Spearman's rho	revision count	Correlation Coefficient	1.000	.249**	.291**
		Sig. (2-tailed)	.	.000	.000
		N	4988	4988	4988
	actor switching	Correlation Coefficient	.249**	1.000	.159**
		Sig. (2-tailed)	.000	.	.000
		N	4988	4988	4988
	Kode_Complexity	Correlation Coefficient	.291**	.159**	1.000
		Sig. (2-tailed)	.000	.000	.
		N	4988	4988	4988
	How well do the following statements describe you?–I would be interested in taking more classes on programming.	Correlation Coefficient	.098	-.008	-.158
		Sig. (2-tailed)	.399	.943	.172
		N	76	76	76
	How well do the following statements describe you?–I hope to get a job doing something with video games.	Correlation Coefficient	-.022	-.131	-.087
		Sig. (2-tailed)	.853	.259	.454
		N	76	76	76
	How well do the following statements describe you?–It?s important to me to know how to program.	Correlation Coefficient	.015	-.137	-.097
		Sig. (2-tailed)	.895	.240	.406
		N	76	76	76

Table 2: Logged in-World and Survey Response Correlation Data

In addition to finding no significant correlations among complexity of game design projects and interest in game design/programming, we also found no significant relationship between gender and kode complexity. In fact, the girls created slightly more complex code than the boys (See Figure 2, below).

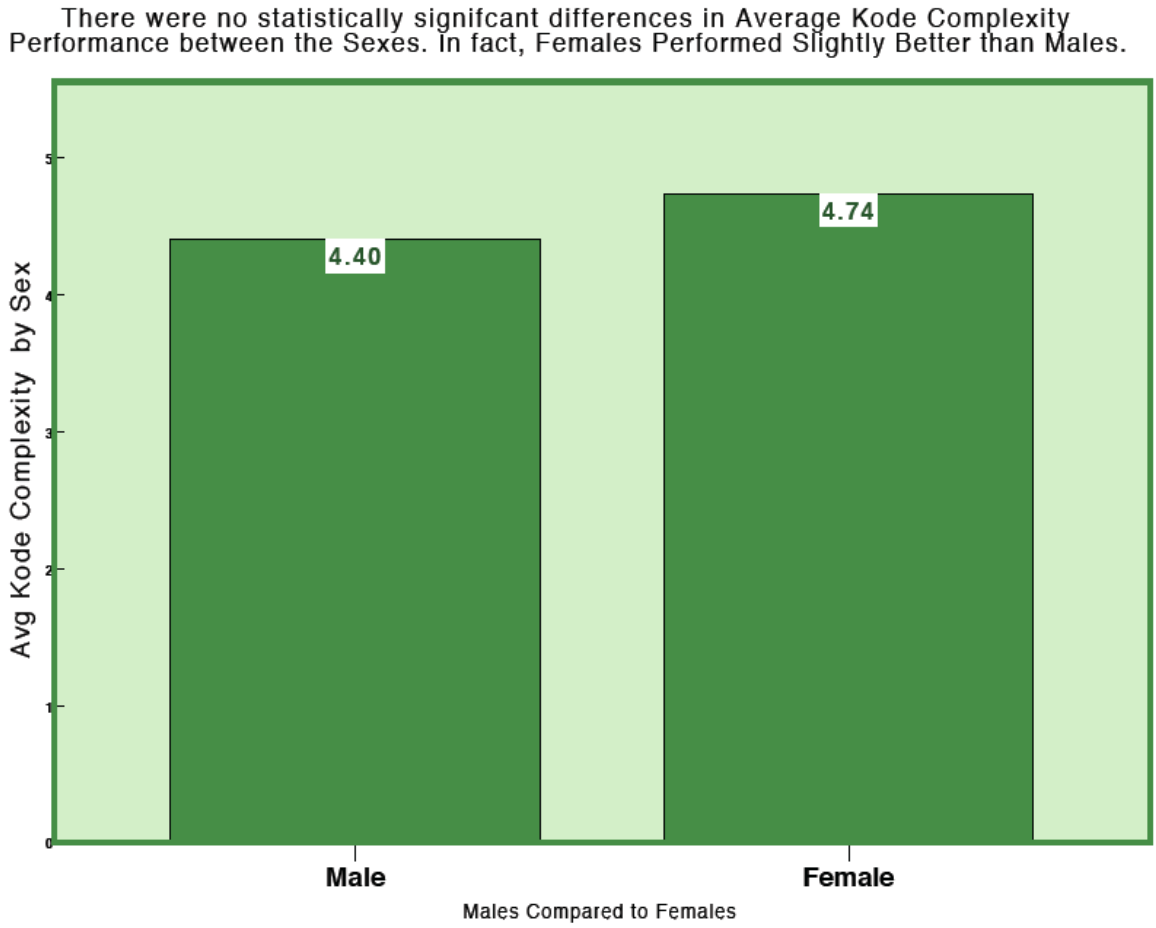

Figure 2: Male versus Female Average Kode Complexity Comparison

Additionally, a linear regression model demonstrates that kode complexity is likely to increase over time, with higher average skill levels developing across all users and significantly increasing with the number of days spent with the curriculum ($p < .001$). No significant findings were discovered using ANOVAs by comparing club sites, number of pages of total kode, or aggregated unique sensor and action counts

Discussion

Analysis of initial telemetry data reveals that as young learners spend more time making games in *Kodu* using the *Studio K* game design curriculum, the complexity of their kode increases. These initial results suggest that *Studio K* provides learning benefits for users, represented through the increasing complexity of kode. Interestingly, these improvements remain regardless of gender and regardless of initial interest in both game design and programming. These two final insignificant correlations, gender and interest correlated with kode complexity, are most interesting, as they suggest that *Studio K* may be effective with a broader audience than is typically seen in game development or computer science.

These results lead into other lines of questioning, begging analysis on examining other measures of learning throughout the curriculum and more concretely defining the learning practices that are occurring throughout use of the program. Next steps include examining the relationship between kode complexity and other computational thinking practices, like debugging patterns. As we begin to further break apart the patterns of play within game design, learning profiles may become illuminated. Since the nature of the program is learning through open-ended, interest-driven challenges, examination of these patterns may show the natural progression through these basic computer science concepts; information that may inform meaningful iterations on the structure and function of the *Studio K* curriculum.

The primary goal of *Studio K* is to provide a unique learning experience to foster interest in computer science or programming through the initial exploration of games. The playful, open-ended challenges may support a broader range of interests and learning styles resulting in a higher success rate among all students, regardless of gender or prior interests in the content area.

These initial results suggest that the initial *Studio K* framework can be leveraged to provide a more enriching experience targeting underrepresented populations. Future research and product iteration will focus on identifying and bolstering the features that support successful acquisition of practices across a wide audience and building new tools that may be absent from the current support structure. These future redesigns may be structured around providing more concrete, meaningful trajectories that players can take through the *Studio K* curriculum that will support their skills, interests, and learning patterns in game design and programming. Players may ultimately complete all curriculum challenges, but they may originally follow their own trajectory. This cyclical process of design, player testing and data analysis, and then another iteration of design allows us to incorporate many of the essential components of design-based research--flexible design revisions and the study of multiple dependent variables, as well as trying to capture the complexities that arise out of capturing and understanding social interaction (Barab & Squire, 2004). Our iterations seek to improve early designs by testing and iteration, informed by analysis of not just what can be concluded about the learners' reasoning, but also the learning environment as a whole (Cobb et al., 2003).

This project will enable the team to aim for some of the true goals of design-based research--to not just show how a design works for a particular project, but to generate claims about learning and contribute new knowledge to the field (Barab & Squire, 2004). We hope that research on *Studio K* will not only reveal computational thinking benefits for users but also important insights about teaching game design and introductory computer programming concepts.

References

Kane, L., Berger, W., Anton, G., Shapiro, R.B., Squire, K. (2012). Studio K: A game design curriculum for computation thinking. In K. Squire, C. Martin, & A. Ochsner (Eds.), *Proceedings of the Games, Learning, and Society Conference: Vol 2.* Pittsburgh PA: ETC Press.

Barab, S., & Squire, K. (2004). Design-based research: Putting a stake in the ground. *Journal of the Learning Sciences*, *13*(1), 1–14. doi:10.1207/s15327809jls1301_1

Berland, M., & Lee, V. R. (2011). Collaborative strategic board games as a site for distributed computational thinking. *International Journal of Game-Based Learning*, *1*(2), 65.

Black, R.W. & Steinkuehler, C.A. (2009). Literacy in virtual worlds. In L. Christenbury, R. Bomer, & P. Smagorinsky (Eds.), *Handbook of adolescent literacy research (*pp. 271-286)*.* New York: Guilford Press.

Bransford, J. D., & Schwartz, D. L. (1999). Rethinking transfer: A simple proposal with multiple implications. *Review of research in education*, *24*, 61-100.

Camp, T. (1997). The incredible shrinking pipeline. *Communications of the ACM*, *40*(10), 103-110.

Cobb, P., Confrey, J., DiSessa, A., Lehrer, R., & Schauble, L. (2003). Design experiments in educational research. *Educational Researcher*, *32*(1).

Gee, J. (2003). *What Videogames have to teach us about learning and literacy.* New York: Palgrave Macmillan.

Hayes, E. R., & Games, I. A. (2008). Making Computer Games and Design Thinking A Review of Current Software and Strategies. *Games and Culture*,*3*(3-4), 309-332.

Kahn, K. 1996. ToonTalk™ – An animated programming environment for children. Journal of Visual Languages and Computing. (An abbreviated version appeared in Proceedings of the National Educational Computing Conference. Baltimore, MD, USA, 7 (June): 197-217, 1995.)

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch programming language and environment. *ACM Transactions on Computing Education, 10*(4).

The National Academies. (2011). *Report of a workshop on the pedagogical aspects of computational thinking.* Available from: http://www.nap.edu/catalog.php?record_id=13170

Peppler, K., Diazgranados, A., & Warschauer, M. (2010). Game Critics: exploring the role of critique in game-design literacies. *E-learning and Digital Media*, *7*(1), 35-48.

Pulimood, S. M., & Wolz, U. (2008, March). Problem solving in community: a necessary shift in cs pedagogy. In *ACM SIGCSE Bulletin* (Vol. 40, No. 1, pp. 210-214). ACM.

Repenning, A. (1993). *Agentsheets: A tool for building domain-oriented dynamic, visual environments.*(Doctoral Dissertation) University of Colorado at Boulder, Dept. of Computer Science

Steinkuehler, C. A. (2004). Learning in massively multiplayer online games. In Y. B. Kafai, W. A. Sandoval, N. Enyedy, A. S. Nixon, & F. Herrera (Eds.), *Proceedings of the Sixth International Conference of Learning Sciences* (pp. 521-528). Mahwah, NJ: Erlbaum.

Squire, K., & Jenkins, H. (2003). Harnessing the power of games in education. *Insight*, *3*(1), 5-33.
Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35.

Acknowledgments

This work was supported by a gift from Microsoft and AMD. Any opinions, findings, or conclusions expressed in this paper are those of the authors and do not necessarily reflect the view of the cooperating institutions.