MakerArcade: Using Gaming and Physical Computing for Playful Making, Learning, and Creativity

Teddy Seyed
University of Calgary
Calgary, AB, T2N 1N4
teddy.seyed@ucalgary.ca

Peli de Halleux
Microsoft Research
Redmond, WA
jhalleux@microsoft.com

Michał Moskal
Microsoft Research
Redmond, WA
michal.moskal@microsoft.com

James Devine
Lancaster University
Lancaster, UK
j.devine@lancaster.ac.uk

Joe Finney
Lancaster University
Lancaster, UK
j.finney@lancaster.ac.uk

Steve Hodges
Microsoft Research
Cambridge, UK
steve.hodges@microsoft.com

Thomas Ball
Microsoft Research
Redmond, WA
tball@microsoft.com

ABSTRACT

The growing maker movement has created a number of hardware and construction toolkits that lower the barriers of entry into programming for youth and others, using a variety of approaches, such as

CHI'19 Extended Abstracts, May 4-9, 2019, Glasgow, Scotland, UK.

ACM ISBN 978-1-4503-5971-9/19/05.
https://doi.org/10.1145/3290607.3312809

KEYWORDS

construction kits, physical computing, tangible user interfaces; game design; maker-activities; block-based programming, STEM; STEAM

gaming or robotics. For constructionist-like kits that use gaming, many are focused on designing and programming games that are single player, and few explore using physical and craft-like approaches that move beyond the screen and single player experiences. Moving beyond the screen to incorporate physical sensors into the creation of gaming experiences provides new opportunities for learning about concepts in a variety of areas in computer science and making. In this early work, we elucidate our design goals and prototype for a mini-arcade system that builds upon principles in *constructionist gaming* – making games to learn programming – as well as physical computing.

1 INTRODUCTION AND RELATED WORK

Over the past several years, efforts to promote 'making' – cross disciplinary activities that involve DIY culture, electronics, science, engineering and craft – have expanded greatly [1]. Building upon these efforts, a number of construction toolkits have been created (e.g. LilyPad [2], Lego Mindstorms [17]) to lower barriers of entry into programming and introduce computing concepts across using a variety of contexts (and domains), such as robotics or e-textiles. Many of these toolkits and subsequent efforts have shown significant success in attracting, engaging and teaching youth in STEAM (Science, Technology, Arts, Engineering and Mathematics) activities [3].

One rapidly growing area for these efforts utilizes gaming. Due to the ever-growing popularity of gaming among children today (especially for entertainment), researchers and educators have begun using gaming to facilitate positive learning experiences [7], building upon core game design principles (scaffolding, interactivity and productive failure) [12]. A more recent approach within this context that has already shown promise is *constructionist gaming*, which involves students (and others) making or programming their own game for learning [9]. Some tools that embody this approach (e.g. Scratch, Kodu Game Lab [18]) enable those with limited game development or programming skills to create games. By enabling children and others to learn by both playing and making games, computational thinking and problem solving can be improved [10].

Although constructionist gaming approaches, tools and associated research is growing rapidly, there is still a large focus on designing around a singular screen (or gaming) experience [11]. Alternatively, commercial gaming has already begun moving beyond the screen into the physical world [5]. For example, the Nintendo Labo platform uses different approaches to combine elements of making and construction into gaming experiences [#ref]. More recently, construction kits have been used to link the design of games with the design of gaming interfaces, and this combination approach has shown promise in fostering collaborative learning, creative expression, and learning computational concepts [11]. This combination approach has revealed challenges with designing around constructionist gaming, especially with regards to construction kits. Specifically, there is not an equal emphasis on both computing and crafting, meaning designing on and off the screen experiences haven't been made equally important for constructions kits for gaming, which differs from other approaches such as those for e-textiles, where screen-based activities are (typically) limited to programming and downloading code onto a wearable [11].

To begin investigating and addressing some of the challenges in designing and building a

Figure 1. An example of a completed MakerArcade system in the form of a miniature Arcade Cabinet running a user-created Flappy-bird like game with a default set of buttons mapped. This game (and others) can be mapped to different sensors (like an accelerometer) that a user can program and map as an alternative controller that is plugged into the cabinet which contains a headphone jack and uses our custom headphone jack-based hardware protocol.

constructionist kit for gaming, we created the MakerArcade system. MakerArcade draws upon concepts of physical computing using manipulatives, similar to [8], as it provides several benefits for novices (e.g. collaboration], visibility of work) [11]. It also draws upon programming concepts taught using simplified graphical, block-based user interfaces like MakeCode, as they simplify programming concepts, and block-shaped constraints help teach and enforce syntactically correct programming statements [16]. In this paper, we describe our preliminary system (Figure 1) that builds upon the Microsoft MakeCode Arcade (beta) software platform[1]. We believe this is an interesting step towards designing and building constructionist gaming experiences that incorporate both hardware and software aspects from the ground up

2 DESIGN GOALS

Informed by our own experiences in designing, building and gathering feedback for MakeCode Arcade and activities of our users from MakeCode Maker[2], as well as prior work in relevant areas of constructionist toolkits, tangibles and physical interfaces [13], we describe our design principles for a constructionist gaming system designed to facilitate play, learning and creativity.

Leverage the Gaming domain. Several construction kits have used their respective domains for design and inspiration, such as MakerWear [13] which utilized wearability and mobility for the design and experience of the platform. Similarly, we aim to draw upon and provide components that incorporate and are inspired by gaming, specifically retro gaming and the arcade culture. Furthermore, retro gaming has begun returning in popularity recently, with re-releases of consoles such as the Nintendo Classic Mini, providing a unique opportunity to blend playful making and learning.

Modularity. Prior work in modular kits for wearables has shown benefit [13]. Similarly, providing a system that supports plug and play with sensors and electronics, enables a broad range of activities for both on and off experiences for constructionist gaming. For example, a user can design, program and physically construct a custom controller for a game (which they've also designed and programmed) using different sensors, like an accelerometer and proximity sensor.

Equality in Physical and Digital Making. A key lesson from [11] with regards to constructionist gaming and the evaluation of existing tools was that there needs to be an emphasize on equality between crafting and computing for both on the screen and off the screen activities in constructionist gaming. This notion should be incorporated both on the physical (activities with electronics) and digital side (activities with the software) when considering the design of a construction gaming kit.

Low-Ceilings, High Walls. Building upon Resnick and Silverman's design cues [15] a construction gaming system should support users in the creation of increasingly complex games that combine both the physical and digital aspects, as further experience is gained.

Tinkerability. In wearable toolkits, an emphasis was placed on rapid tinkering and prototyping

[1] Microsoft MakeCode Arcade – http://arcade.makecode.com/beta
[2] Microsoft MakeCode Maker – http://maker.makecode.com/beta

Figure 2. Microsoft MakeCode Arcade (beta) that allows a user to create a game using block-based programming concepts. We provide packages that allow a user to program custom physical controllers that use buttons or sensors (e.g. sound) and map input to the game being programmed. For example, a user can map shaking input from an accelerometer (using a block) to the flapping of the bird. When the game is run on MakerArcade, the sensor is plugged in to play.

Figure 3. Microsoft MakeCode Maker (beta) that allows a user to program and prototype electronics. We provide packages which show a user how to physically create an arcade (with a screen) and buttons mapped to a microcontroller. We also provide mappings on how to create additional plug and play components (for custom controllers, sensors or modules), that use a custom headphone jack-based hardware bus protocol. These mappings when programmed by a user, are auto-detected when plugged into our MakerArcade system.

[2,13]. Similarly, our system should enable rapid tinkering and experimentation, as we aim to enable users (especially children) to easily experiment and play with different components. Given that expression is an important aspect of the constructionist approach [9,11], focusing on a wide range of tinkering activities (e.g. adjusting effects) in the creation and play of a game is important.

Fostering Collaboration. Another benefit seen from constructionist gaming activities, is the occurrences of collaboration and their associated artifacts (e.g. components customized with crafts), particularly in classroom settings [6]. This means that a system should both enable and capture (digitally and physically) these artifacts to foster collaboration.

3 MAKERARCADE

Our current prototype of the MakerArcade system is comprised of: (i) custom software experiences built upon the Microsoft MakeCode framework[3][1], (ii) a mini arcade cabinet that houses electronics and (iii) custom modular blocks that can be programmed. MakerArcade is heavily inspired by the retro gaming and DIY culture (through the cabinet), as well as prior work in modular blocks and devices, such as [14]. Next, we briefly describe our current implementation

3.1 Software

The software components of MakerArcade use customized (beta) versions of Microsoft MakeCode Arcade (Figure 2) and Microsoft MakeCode Maker (Figure 3). Both are open-source and are built upon the Microsoft MakeCode web framework, which enables a web-based programming experience for novices. It's editor uses block-based programming and code can be executed within the browser itself [4]. The MakeCode Arcade editor enables novices to learning programming by creating games in the browser using block-based programming (similar to [18]). Games are stylized in a retro 8-bit manner, with a small view dynamically rendering the game as it is being programmed. The MakeCode Maker editor allows novices to learn how to both program and prototype (via breadboard) with a variety of different sensors and electronics components.

For our MakerArcade prototype, a user creates their game using MakeCode Arcade, which can then be loaded onto it (using USB) and played immediately. MakerArcade by default comes with 3 buttons (mapped to Start / Reset / B key), however, several games require more than 1 button for interaction (e.g. a Pacman-like game requires multiple keys for directions). This is where our software customizations arise, combining digital and physical construction through the use of Arcade and Maker. Using customized packages in Arcade, and Maker, users are able to program custom plug and play input using sensors (or buttons) for their game.

The mapping of custom input created by a user in Arcade is done automatically using our packages. This means that a user can not only create their game, but they can also create how they want their game to react with a chosen form of input. Given that sensors (and buttons) can be used dynamically, it allows a user to build their own controller experience that plugs into the MakerArcade

[3] Microsoft MakeCode – http://www.microsoft.com/en-us/makecode

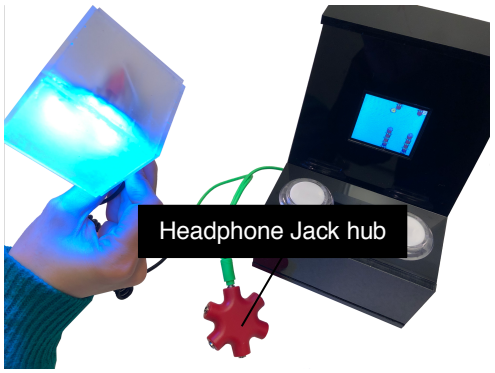

Figure 4. This custom designed accelerometer controller plugged into additional inputs using a headphone jack hub. This lets a user build their own controller and interactions for their game using MakerArcade.

Figure 5. Buttons from an Adafruit Circuit Playground Express (CPX) used as a controller for the Flappy-bird game. The CPX is coded using blocks (and a custom plug-and-play headphone-jack bus protocol) within MakeCode Maker. On MakeCode Arcade, a user simply programs additional buttons using our packages, which are auto-detected when plugged into the headphone jack hub.

cabinet, or even further customize their own MakerArcade, as instructions are provided on Maker that describe how to build an arcade using a screen, microcontroller and buttons on Maker (full list of supported hardware available on our website). In summary, we allow users to fully customize a gaming experience, not just with their programmed game, but also with their own plug and play controllers (built using their own choice of input) for our provided Mini-arcade cabinet, which can also be further customized.

3.2 Hardware

Our provided MakerArcade physical prototype consists of (1) a SAMD51-based microcontroller (Adafruit Trinket M4 Express), (2) a ST7735-based 1.8" TFT LCD, (3) 3 analog buttons and (4) an audio-jack port mapped to a pin on the microcontroller, all of which is housed in a laser-cut acrylic Arcade cabinet. The SAMD51-based microcontroller (and other supported microcontrollers) run custom bootloaders that enable it to both run games compiled on MakeCode Arcade, as well as the screen itself. The audio-jack port plays a critical role in the plug-and-play architecture for the constructionist components of gaming. We built upon JACDAC[41], which is a bus-based hardware protocol that utilizes audio-jack cables.

Generally, the supported parts serve as the base components for our designed MakerArcade system (and cabinet), but users are also able to design and build their own housing for these components, emphasizing more of the playful making and physical computing aspects. For example, a user could create a console-like design (e.g. Nintendo Classic) to house the base components.

With MakerArcade, a user can plug and play their own custom designed inputs (with mappings provided and created in Maker) into our supplied cabinet. These inputs use the JACDAC protocol which requires headphone jack cables, and with a Headphone jack hub, multiple custom inputs become plug and play (Figure 4). Furthermore, microcontrollers that are supported on MakeCode Maker (e.g. Figure 5) can serve as input (and eventually output) for games created on MakeCode Arcade. This means that users can fully prototype and build their own MakerArcade controller experiences with a variety of popular microcontrollers.

4 CONCLUSION AND FUTURE WORK

We introduced our early work towards building a constructionist gaming system that emphasizes designing, remixing, and customizing physical and digital components in building games. As it is early work, there are numerous directions and opportunities we intend to explore, including: (1) providing alternative and more customizable designs (e.g. console-like, hand-held or a wearable form factor); (2) explore providing modules with outputs that can be used in the creation and experience with a user programmed game (e.g. sound, movement); (3) participatory design sessions with children of different age groups and makers and (4) running user studies with a similar group to see how they use MakerArcade and what games, controllers and experiences they design, ultimately creating guidelines for creating constructionist gaming systems and kits.

[4] JACDAC – http://jacdac.org

REFERENCES

[1] Lisa Brahms. 2014. Making as a learning process: Identifying and supporting family learning in informal settings. University of Pittsburgh.

[2] Leah Buechley, Mike Eisenberg, Jaime Catchen, and Ali Crockett. 2008. The LilyPad Arduino: Using Computational Textiles to Investigate Engagement, Aesthetics, and Diversity in Computer Science Education. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08), 423–432. https://doi.org/10.1145/1357054.1357123

[3] Leah Buechley and Benjamin Mako Hill. LilyPad in the Wild: How Hardware's Long Tail is Supporting New Engineering and Design Communities. 9.

[4] James Devine, Joe Finney, Peli de Halleux, Micha\l Moskal, Thomas Ball, and Steve Hodges. 2018. MakeCode and CODAL: Intuitive and Efficient Embedded Systems Programming for Education. In Proceedings of the 19th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES 2018), 19–30. https://doi.org/10.1145/3211332.3211335

[5] Michael Eisenberg, Nwanua Elumeze, Michael MacFerrin, and Leah Buechley. 2009. Children's Programming, Reconsidered: Settings, Stuff, and Surfaces. In Proceedings of the 8th International Conference on Interaction Design and Children (IDC '09), 1–8. https://doi.org/10.1145/1551788.1551790

[6] Allan Fowler. 2017. Engaging young learners in making games: an exploratory study. In Proceedings of the 12th International Conference on the Foundations of Digital Games, 1–5.

[7] James Paul Gee. 2007. Good video games+ good learning: Collected essays on video games, learning, and literacy. Peter Lang.

[8] Audrey Girouard, Erin Treacy Solovey, Leanne M. Hirshfield, Stacey Ecott, Orit Shaer, and Robert J. K. Jacob. 2007. Smart Blocks: a tangible mathematical manipulative. 183–186. https://doi.org/10.1145/1226969.1227007

[9] Yasmin B. Kafai. 2006. Playing and Making Games for Learning: Instructionist and Constructionist Perspectives for Game Studies. Games and Culture 1, 1: 36–40. https://doi.org/10.1177/1555412005281767

[10] Yasmin B. Kafai and Cynthia Carter Ching. 2001. Affordances of Collaborative Software Design Planning for Elementary Students' Science Talk. Journal of the Learning Sciences 10, 3: 323–363. https://doi.org/10.1207/S15327809JLS1003_4

[11] Yasmin B. Kafai and Veena Vasudevan. 2015. Constructionist Gaming Beyond the Screen: Middle School Students' Crafting and Computing of Touchpads, Board Games, and Controllers. In Proceedings of the Workshop in Primary and Secondary Computing Education (WiPSCE '15), 49–54. https://doi.org/10.1145/2818314.2818334

[12] Manu Kapur. 2008. Productive Failure. Cognition and Instruction 26, 3: 379–424. https://doi.org/10.1080/07370000802212669

[13] Majeed Kazemitabaar, Jason McPeak, Alexander Jiao, Liang He, Thomas Outing, and Jon E. Froehlich. 2017. MakerWear: A Tangible Approach to Interactive Wearable Creation for Children. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17), 133–145. https://doi.org/10.1145/3025453.3025887

[14] David Merrill, Jeevan Kalanithi, and Pattie Maes. 2007. Siftables: Towards Sensor Network User Interfaces. In Proceedings of the 1st International Conference on Tangible and Embedded Interaction (TEI '07), 75–78. https://doi.org/10.1145/1226969.1226984

[15] Mitchel Resnick and Brian Silverman. 2005. Some reflections on designing construction kits for kids. In Proceedings of the 2005 conference on Interaction design and children, 117–122.

[16] David S. Touretzky. 2014. Teaching Kodu with Physical Manipulatives. ACM Inroads 5, 4: 44–51. https://doi.org/10.1145/2684721.2684732

[17] Homes - Mindstorms LEGO.com. Retrieved January 5, 2019 from https://www.lego.com/en-us/mindstorms

[18] Kodu | Home. Retrieved January 5, 2019 from https://www.kodugamelab.com/