# Responsive Make and Play: Youth Making Physically and Digitally Interactive and Wearable Game Controllers

**Gabriela T. Richard and Yasmin B. Kafai**

**Abstract** Most research on game making has focused on screen designs leaving aside the potentially rich domain of making game controllers for learning. In this chapter, we illustrate how youth created wearable or e-textile-based controllers with physical and digital feedback by combining tangible and digital construction kits, using Scratch, the MaKey MaKey, the Lilypad Arduino, and ModKit. In an eight-session workshop, 14–15-year-old youth not only programmed their own Scratch games but also created wearable or physically reactive game controllers using sensors to activate a response on the screen, through the physical artifact, or in both interfaces. In the discussion, we address what we learned about tool and workshop design to facilitate and support the introduction of such playful interface design to novice programmers.

**Keywords** Tangible interfaces · Wearable games · Tangible games · Youth design · Electronic textiles · Game controllers · Bidirectionally responsive design · Scratch games · Makey Makey · Lilypad Arduino · ModKit · Hybrid crafting · Physical computing · Digital games

## 1 Introduction

Augmented game controllers, which measure pressure, movement, and touch, have been around since the early days of gaming, starting most notably with Nintendo's PowerPad and PowerGlove in the 1980s (Tanaka et al. 2012). Newer technologies have allowed for more robust and dynamic sensing that can measure pitch and tilt, and gesture and body movement. While controllers with these features are now

G.T. Richard (✉) · Y.B. Kafai
University of Pennsylvania, Philadelphia, PA, USA
e-mail: gric@upenn.edu

Y.B. Kafai
e-mail: kafai@upenn.edu

A. Nijholt (ed.), *More Playful User Interfaces*, Gaming Media and Social Effects,
DOI 10.1007/978-981-287-546-4_4

commonplace in many commercial gaming platforms, making them has been out of the reach for many players even though tinkering and building external gaming systems and controllers was an integral part of early days of digital gaming (Swalwell 2012). Increasingly, proprietary systems and costly development tools have made the design of such interfaces a complex activity, requiring extensive technical and material knowledge and skills (Millner 2010). With the recent availability of low-cost and easy-to-use tangible computational construction kits using Arduino as a platform, this barrier has been removed and put the design of augmented gaming peripherals back into reach of novice makers and game designers.

Making your own physically and digitally responsive game peripherals that use sensors to control action on the screen and can provide feedback to the physical controller itself is a complex activity, but one that potentially could enrich and extend game making for learning. In many youth online DIY communities, such as Scratch (Resnick et al. 2009), Kodu (Fowler and Cusack 2011), and others, making and sharing your games with millions of others is one of the most popular programming activities (Kafai and Burke 2014b). It is becoming an equally popular activity for serious gaming where games are seen as promising learning environments to promote problem solving, collaboration, and design thinking (National Research Council 2011). While most of the attention has focused on instructionist approaches, i.e., playing digital games for learning, current trends also focus on constructionist gaming, i.e., making games for learning (Kafai and Burke 2014a). It is here where serious gaming connects with efforts to promote programming (Kafai and Burke 2014b) and making (Honey and Kanter 2013) in schools. So far, most making and programming activities have been implemented outside of school, but designing physically and digitally reactive or responsive game peripherals could offer a promising opportunity to bring together crafting, computing, and gaming for school learning. First, studies of students designing their own touch pads (Davis et al. 2013; Lee et al. 2014) and augmented game boards (Vasudevan and Kafai 2015) provide indication that designing, crafting, and programming these peripherals can enrich constructionist gaming efforts.

In this chapter, we expand the constructionist gaming in two new directions: (1) by making controllers that are both physically and digitally responsive, and (2) by focusing on wearable or textile-based controllers. We present findings from a workshop in which high school youth ages 14–15 years learned not only how to program their own Scratch games but also how to make wearable controllers. We asked students to focus on bidirectionally responsive design, meaning that sensors would activate a response on the screen and through the physical artifact simultaneously. An example of such bidirectional design could be a wearable game controller glove, where touching the glove would not only result in a change in the digital game state but would also signal its responsiveness through changes in color or vibration on the glove itself, thus providing tangible feedback in the glove. To accomplish such responsive designs, students needed not only to program in Scratch but also to integrate textile elements using the Lilypad Arduino (Buechley et al. 2008) programmed via ModKit (Baafi et al. 2013), a visual block-based programming language similar to Scratch, and the MaKey MaKey (Silver et al.

2012). Using conductive materials such as special fabric or aluminum foil, they would integrate the Lilypad Arduino to create a physical response in the wearable or textile game controller (i.e., a light turning on or a vibration), and the MaKey MaKey to control elements of their game in Scratch. In the discussion, we address what we learned about tool and workshop design to facilitate and support the introduction of such playful interface design to novice programmers.

## 2 Background

Constructionist gaming has been a popular activity since the early 1990s when the idea was first introduced that children could program their own video games for learning about coding in an authentic and personally meaningful way (Kafai 1995). The designing of video games on 256k computers meant that students literally had to program every single pixel on the screen to produce the colorful graphics and interfaces that were popular with commercial video games sold by Nintendo and Sega. Today, most programming platforms such as Scratch, Kodu, Alice, Agentsheets, and many others use visual programming environments that remove the thorny syntax issues that traditionally plagued much of novice programming while providing visual feedback. Graphics can be imported from the Internet and modified to create professionally looking interfaces for their games. Game making is one of the most popular design activities in online communities where youth (alone or together) have created and shared thousands of games emulating commercial games or of their own creation. Numerous studies have examined the benefits of game making in terms of learning programming, collaborative and creative design, and many other aspects (Hayes and Games 2008; Kafai and Burke 2014a).

Noticeably absent from this research on constructionist gaming has been the inclusion of peripherals such as game controllers. While video games have always used a variety of peripherals beyond the keyboard, such as joysticks, powergloves, or dancing mats, the arrival of the Wii re-energized the search for new interfaces because it illustrated that game controllers could make gaming more accessible to a wider audience. Making such new interfaces, however, is a difficult enterprise, often limited to professionals with extensive technical skills and material knowledge, unlike in the early days of digital gaming and microcomputers. As noted above, modding software, building components, and hacking were often integral and encouraged aspects of digital game play. Reflecting on computer gaming in the home in the 1980s, Swalwell (2012) observed that "a number of early hobbyist microcomputers came in electronic kit form, requiring that users first assemble them" (p. 5). New toolkits and materials have begun to open up these "black boxes" (Resnick et al. 2000) in various ways (Tanenbaum et al. 2013). Most notable is here the Arduino (Kushner 2011) an accessible construction kit for creating physical computing and tangible design, and the Lilypad Arduino, the electronic textile counterpart to the Arduino (Qiu et al. 2013). The most recent addition is the MaKey MaKey (Silver et al. 2012), which seamlessly connects to the computer and allows the designer, without any programming, to replace keyboard functionality with conductive materials and alligator clips. These toolkits have helped to foster the “Maker Movement,” a do-it-yourself movement and ethos, which is transforming the way the individuals learn about crafting, construction, development, and design, often through nontraditional or informal learning spaces, such as Maker Faires.

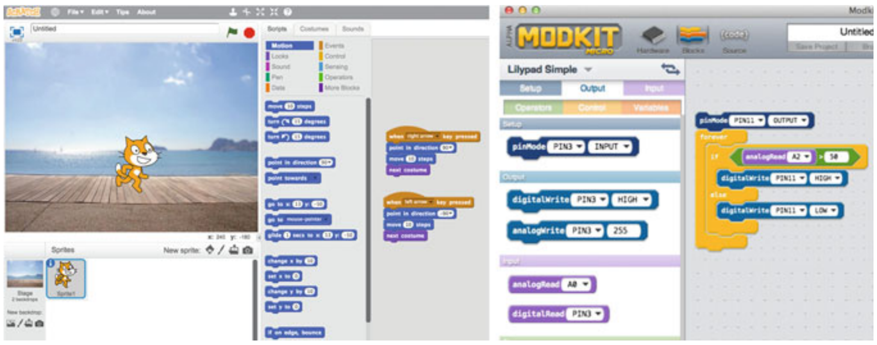
A few studies have began to use these toolkits to engage youth in making their own game peripherals, starting with MaKey MaKey, that does not require any prior programming knowledge. Examples have been to have middle school youth ages 10–12 years design their own game controllers for Scratch games (Davis et al. 2013; Lee et al. 2014) or making augmented board games (Vasudevan and Kafai 2015). In one of these projects, youth crafted their touchpads using conductive Play-Doh, paper, and pencil drawings to make new controllers that they connected with the Makey Makey to their online Scratch games. At the end of the project, youth set up an arcade with their games for other students in their schools, to playtest and receive feedback on their games and interfaces. In the other project, teams of high school youth designed board games with augmented gaming components such as digital dice that connected via the MaKey MaKey to Scratch. Other researchers have used Lego construction kits (Martinez 2014) to create augmented game boards that interact with computer projected screen interfaces. In all of these projects, the control or feedback between the controller and the computer screen has been unidirectional, going into only one direction.

In this chapter, we propose a new approach that we call bidirectionally responsive design, meaning that sensors can either activate a response on the screen or through the physical artifact. Past efforts have been made to teach physical computing (O’Sullivan and Igoe 2004) to youth and teachers in K-12 settings in order to help them design their own tangible learning modules, with digital and physical interfaces, beyond any specific scientific concepts (Richard 2008). This approach relates to recent work on bifocal modeling (Blikstein 2012), an extension of physical computing, which has used sensor technology in tandem with real-time visual simulations to help students understand physics phenomena by measuring data and displaying them on the screen. While bifocal modeling integrates two modalities, it is still unidirectional by having the information from the physical data go to the digital screen. There is no feedback loop or control from the screen to impact the actual measurements or data in the other modality. Our approach also combines two modalities, the physical and the digital, but is distinct in two dimensions: context and directionality. For one, we are using gaming, and not physics, as an authentic and personally meaningful context to situate data sensing and feedback for students’ engagement. In other words, rather than to focus on data representation, we focus on data use. For that reason in responsive design, the visualization is not captured in a responsive diagram but rather in a light or sound display providing feedback. This approach is much closer to efforts in teaching programming such as media computation that uses the manipulation of images and sound to help students learn about algorithm use (Guzdial 2013). Efforts have been made to provide for tools that can foster “hybrid crafting” (Golsteijn et al. 2014), or the more seamless creation of digitally and physically augmented interfaces through building with simple construction kits. Despite the term "crafting," these kits focus more specifically on a Lego-type interface with a portable screen that the researchers designed and prototyped with users. However, instead of focusing on creating a new toolkit, we focused on construction kits that were readily available and could combine two interfaces with conductive materials. We also situated the physical design in a wearable context thus moving out of the laboratory into the physical space, where much of gaming nowadays takes place. Second, we also aimed for including bidirectionality by having a dual response in the digital and physical space. This second dimension, while important, also added further complexity on the design of the interface, and as we found out in our study, proved to be challenging to implement for some novice designers.

In asking students to design wearable responsive controllers, we examined the following aspects—students' actual designs and their reflective responses. Our first line of investigation focused on the gaming peripherals that students created and how the crafting and coding related to the game design. This approach brings together the work on programming and making activities that, respectively, have mostly focused on screen designs or physical artifact. Here, we were interested in how students navigated the spaces of learning coding, related computational concepts and practices, together with the material demands of building physical artifacts considering conductive properties of materials. Our second line of investigation focused on students' perceptions and examined how students reflected on their learning experiences in making responsive designs.

## 3 Toolkits and Implementation

In order to design the wearable interfaces with digital and physical interactivity, students had to learn how to use four different toolkits. The digital construction kits or programming environments were Scratch 2.0 (Resnick et al. 2009) and ModKit Alpha (Baafi et al. 2013), while the physical construction kits were the MaKey MaKey (Silver et al. 2012) and the Lilypad Arduino (Buechley et al. 2008). Scratch is a media-rich programming environment, which uses a visual block-based coding language (see Fig. 1). Like Scratch, ModKit is a visual block-based coding environment, which is used for various Arduino platforms, including the Lilypad Arduino and other robotics platforms (see Fig. 1). The MaKey MaKey is a plug-and-play physical computing construction kit designed to turn almost anything with conductive properties into a physical interface (see Fig. 2). The LilyPad Arduino is a sewable microcontroller that can be used with conductive fabric and thread to make interactive textiles and wearables; for the workshop, we used the Lilypad Arduino Simple Board, which works well with ModKit and has a more simplified design (see Fig. 2). We felt it was incredibly important to keep the programming modality consistent to lower the complexity involved, given that the workshop only lasts a few weeks; hence, why ModKit was used instead of the Arduino line coding environment. Also, ModKit had successfully been used in similar past workshops



```
when right arrow key pressed
point in direction 90
move 10 steps
next costume

when left arrow key pressed
point in direction -90
move 10 steps
next costume
```

```
pinMode PIN11 OUTPUT
forever
    if analogRead A2 > 50
        digitalWrite PIN11 HIGH
    else
        digitalWrite PIN11 LOW
```

**Fig. 1** (*Left*) Scratch interface. (*Right*) ModKit Micro Alpha interface

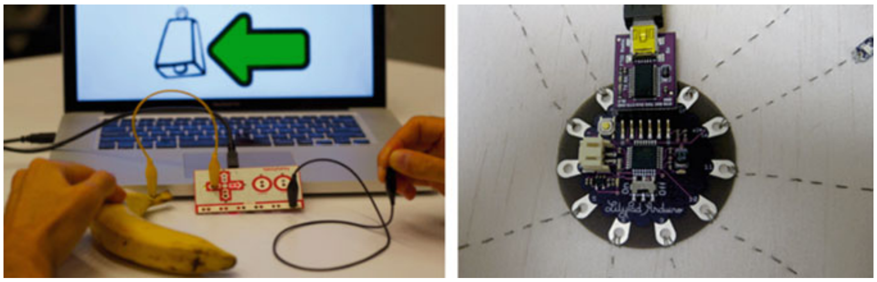


**Fig. 2** (*Left*) The MaKey MaKey, as connected to a banana, replacing the left arrow key and completing the circuit by using human capacitance to ground the connection. (*Right*) Lilypad Arduino connected to sensors and actuators with conductive thread

(Qiu et al. 2013), despite its beta nature. In this pilot of the curriculum, we utilized ModKit Alpha, which is a newer version of ModKit that has worked out some of its initial bugs with the Lilypad Arduino (see Fig. 2).

Asking students to learn and work with that many different construction and programming kits to achieve responsive designs required decisions about sequencing the introduction of tools and activities. We choose to start with the Lilypad Arduino to introduce students to physical computing activities and to get them out of the typical screen-based interactions they were used to (even though most students had neither prior programming experience nor knowledge of Scratch or ModKit). ModKit was used as an entry-level to visual object-oriented programming for the Lilypad Arduino so that students could focus on making connections between creating circuit designs and coding behaviors and interactions of sensors and actuators (Fig. 3). We used the MaKey MaKey after students had experimented with different Lilypad Arduino designs because it allowed an almost seamless connection between conductive materials and the computer to replace the keyboard and mouse. Scratch was incorporated at the same time as the MaKey MaKey in part because it is often taught in tandem with it due to its ease of use that facilitates experimentation.

## 4 Workshop Design

The *Maker Innovators* workshop took place in a local public school associated with the museum in a large Northeastern city in the USA. The workshop ran for eight weeks, occurring once a week for 2 h, between February and April 2014. Thirteen ninth grade students (6 male, 7 female), aged 14–15, from a public, science magnet high school choose to participate in the workshop. The racial makeup was 62 % White, 15 % Black or African American, 15 % Asian American, and 7 % biracial (Black or African American and White). Ten of the 13 students consented to be interviewed, and all except one female and one male student consented to documentation. The workshop was designed and implemented by the first author with two assistants helping students with project designs and documenting with video class activities and designs.

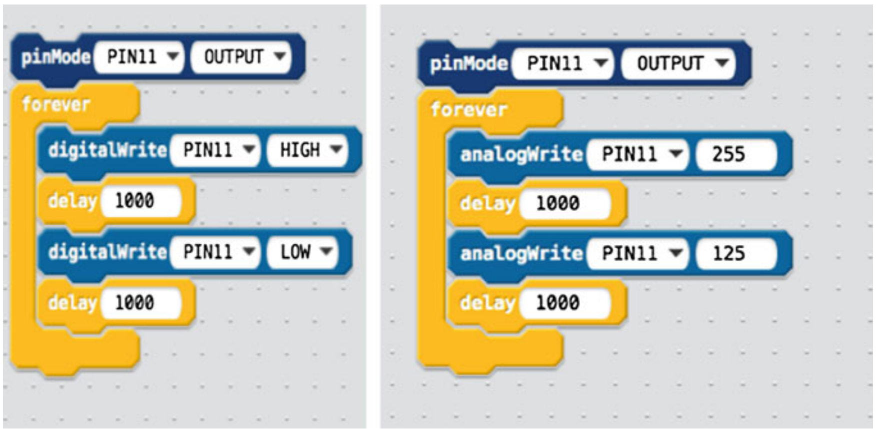
The workshop was broadly divided into two parts: during the first four sessions, students learned how to use the different tools, while in the second half, students worked in teams on designing their own controllers (see Table 1). In the first session, we showed examples of videos of responsive interface designs by other youth or students who had worked with either Scratch, the Lilypad Arduino, or the MaKey MaKey. We also included here the promotional video of the MaKey MaKey. Since we could not locate examples of physically interactive electronic textiles integrated with Scratch and the MaKey MaKey, we asked student teams to brainstorm ideas for their interface designs.

Over the course of the next sessions, we transitioned from learning different toolkits to focus on understanding circuitry, coding, and computation. For example, we started off with ModKit to understand block-based object-oriented coding and the Lilypad Arduino to understand basic circuits. Students then would work on exercises with the various toolkits and were given reference and troubleshooting guides they could refer to throughout the workshop. During the first session, students learned to code in ModKit, starting with very basic coding to light up an LED. Students learned about basic circuits and electricity, as well as how to connect their Lilypad Arduinos to the LEDs with alligator clips. During week 2, after learning some basics about initiating pins, and sending out digital signals to light up an LED, they learned about analog signals and changing the brightness of LEDs. The students then experimented with block coding in ModKit to change the brightness of their LEDs. Following this, they learned about variables, conditional statements, forever loops, and functions in order to continuously change the brightness of their LEDs. They were also introduced to sensors and how they worked like our hands and eyes. They played with light and temperature sensors, and we made connections to how they worked like variables when coding. In this sense, when they added a new AnalogIn value in ModKit, it worked similar to a variable, storing, and sending different values that could affect the code. They also started working with conductive thread and felt to sew their connections together and understand how conductive thread worked like a live wire (Fig. 4).

**Table 1** Weekly overview of workshop activities

| | Summary | Activities |
|---|---|---|
| Week 1 | Videos, discussion of possibilities, starting with the *Lilypad Arduino,* and electronic textiles | • Inspirational videos of wearable design projects created with the Lilypad Arduino or physical computing projects created with the MaKey MaKey<br>• Brainstorming session on combining the Lilypad and MaKey MaKey<br>• Lesson on basic circuits, sensors, and actuators<br>• Working with the Lilypad Arduino and ModKit to design a basic circuit with an LED<br>• ModKit used as a model for block-based coding<br>• Coding exercises with loops |
| Week 2 | Continuing work on the Lilypad Arduino and electronic textiles. Programming the *Lilypad* with *ModKit* | • Coding exercises with variables, loops, and conditional statements and reading in analog values from sensors received through the Lilypad<br>• Playing with different sensors of their choosing (temperature, light, or slide switches), with different actuators (LEDs, buzzers, and vibrators) |
| Week 3 | Working with *Scratch* and the *MaKey MaKey* | • Working with the MaKey MaKey and concepts on basic circuits<br>• Playing with different materials, such as Play-Doh, graphite (from pencil drawings), aluminum and conductive fabric to understanding conductivity<br>• Remixing and creating code in scratch to add different functionality |
| Week 4 | Combining scratch, the *MaKey MaKey* and the *Lilypad Arduino*; Brainstorming and discussing ideas on teams | • Integrating the Lilypad with the MaKey MaKey to control a game in scratch with LEDs<br>• Sewing conductive thread with the Lilypad<br>• Brainstorming and discussing ideas for final projects |
| Weeks 5–8 | Putting together design ideas through iteration; On week 8, showcasing work for an audience of experts and peers during the last hour of class. | • Working on design teams for final projects<br>• During the final session, they showcased projects in their current form |

During week 3, they worked with Scratch, creating a basic music game and a basic action game, using code that was explained during class time. They were encouraged to experiment and modify the code as they saw fit. They were then given MaKey MaKeys to play with (see Fig. 5), along with Play-Doh, or pencil



```
pinMode PIN11 OUTPUT
forever
    digitalWrite PIN11 HIGH
    delay 1000
    digitalWrite PIN11 LOW
    delay 1000
```

```
pinMode PIN11 OUTPUT
forever
    analogWrite PIN11 255
    delay 1000
    analogWrite PIN11 125
    delay 1000
```

**Fig. 3** (*Left*) Basic digital code to turn on and off an LED in ModKit. (*Right*) Basic analog code to vary the brightness of an LED in ModKit

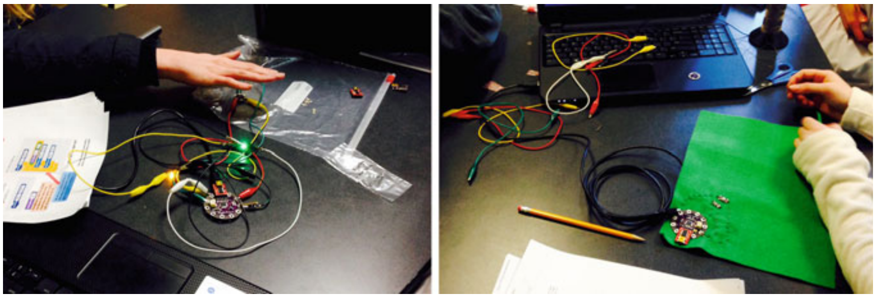


**Fig. 4** (*Left*) A student playing with a light sensor to vary the brightness of an LED with the Lilypad. Students used alligator clips to connect sensors and LEDs during prototyping. (*Right*) Students sewing electronic textiles on felt with conductive thread

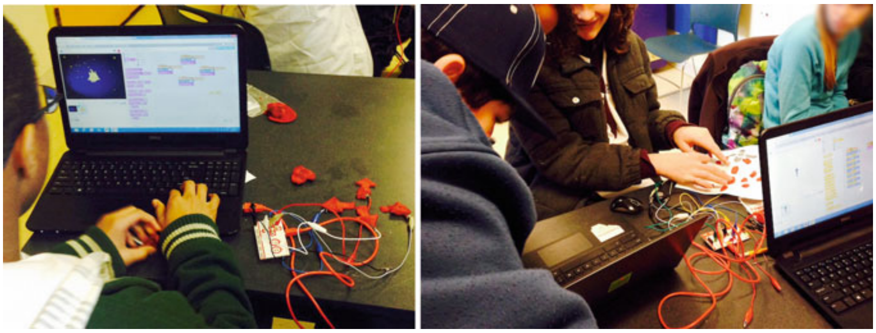
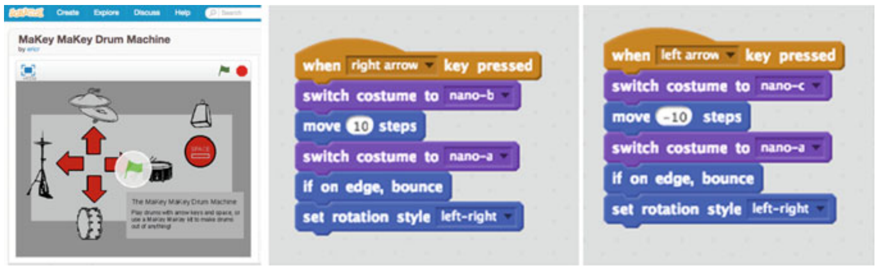


**Fig. 5** Students working with the MakeyMakey, scratch, Play-Doh and pencil drawings to understand physical computing and conductivity



```
when right arrow key pressed
switch costume to nano-b
move 10 steps
switch costume to nano-a
if on edge, bounce
set rotation style left-right
```

```
when left arrow key pressed
switch costume to nano-c
move -10 steps
switch costume to nano-a
if on edge, bounce
set rotation style left-right
```

**Fig. 6** (*Left*) Basic MaKey MaKey Scratch music game students remixed. (*Right*) Basic action game code we presented to learners to modify

drawings (graphite is conductive). They were also introduced to remixing code, first by working with the MaKey MaKey Scratch game (see Fig. 6), then by choosing any Scratch program they wished to play with. Finally, during weeks 4 and 5, they were reintroduced to the Lilypad Arduino, ModKit, Scratch, and the MaKey MaKey. They learned how to work with conductive fabric to change the brightness of an LED through the Lilypad and control movement in a game in Scratch.

During the second half of the workshop, students would work on self-selected teams with 2–4 members each, mapping and crafting their own responsive and wearable controllers. Beginning in week 5, they were asked to brainstorm project ideas and work in teams to create a project that would combine both environments. For the next few weeks, they worked on their projects in teams, figuring out what kinds of interactions they wanted in their physical textile interface and in Scratch. They were given free reign to come up with ideas and see them come to fruition. Author 1 was the primary course instructor who helped students think through their design concepts and work through issues that came up during their design process. Students were given handouts with all of the sample code used in class, a list of Scratch codes, and how to sew with electronic textiles. Five teams were formed among the thirteen students, of which four teams created fully working prototypes of varying complexity. The last class session was organized like a demo fair with students setting up their designs at individual tables. We had invited outside visitors —a professional board game designer and two education researchers (among them the second author)—to provide an audience and feedback along with their peers. Each team spent about 10–15 min for demonstrating and explaining their final designs to the visitors and other students in the workshop.

We used video and observation notes to document class activities and student interactions and design. We also documented all of the final projects created in the class. At the conclusion of the workshop, we conducted interviews with all of the consenting students. These interviews were transcribed and then analyzed to understand how youth conceptualized of their designs, their design process, and their understanding of bidirectional responsiveness.

## 5 Findings

So to what extent were the student teams able to realize wearable responsive game controllers? Implementing bidirectional designs turned out to be a formidable challenge despite all good intentions. Of the five teams, only one team succeeded in conceptualizing, coding, and crafting a design with full bidirectional functionality, while all other teams limited their designs to unidirectional functionality (mostly due to time limitations or being more focused on one utility versus another). Students conceived of the designs themselves, and the instructor (author 1) helped them to brainstorm and think through the bidirectional development process. Only one team of three students was unsuccessful in creating a working prototype by the final class (this team also suffered from multiple absences, excluding one member). However, these designs demonstrated the potential for wearable interfaces and also underscored students' understanding of the utility and potential of bidirectional feedback. In the following section, we provide case studies of some of the unidirectional and bidirectional designs. As part of our presentation, we include students' reflections on their choices and learning.

### *5.1 Unidirectional Wearable Game Controllers*

Examples of unidirectional designs included a jousting game (see Fig. 7, left) by a team of three students (2 male, 1 female) who created wearable vests with different conductive parts to them. They also included swords out of found materials (coat hangers). They connected the MaKey MaKey to the swords and the vests. Play involved trying to block your opponent from tapping the conductive part of the vest. When unsuccessful, the matching Scratch game, would respond by awarding the opponent a point and reflecting a reaction between the characters in the game, complete with funny sounds (see Figs. 8 and 9). Another example was the finger flappy star controller (see Fig. 7, middle), which was designed by a team of three students (2 male, 1 female). It included a wearable, partial glove controller that used two fingertips and conductive fabric to control a version of the then popular Flappy Bird game that was remixed from someone else's version found in the Scratch site.

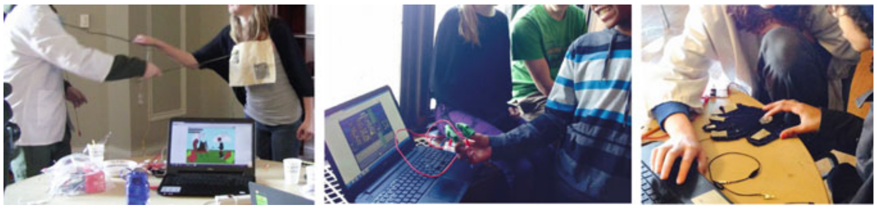


**Fig. 7** Snapshots of select unidirectional designs: jousting sticks (*left*), flappy star game and controller (*middle*), and sensor glove (*right*)

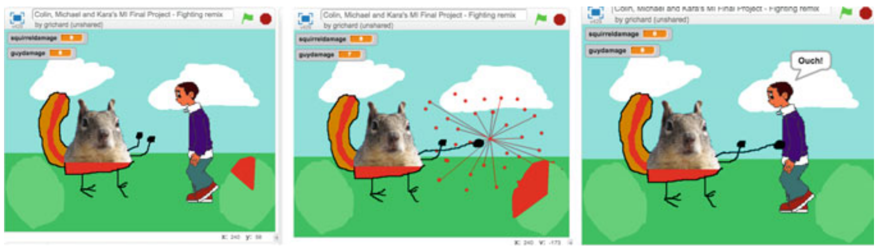
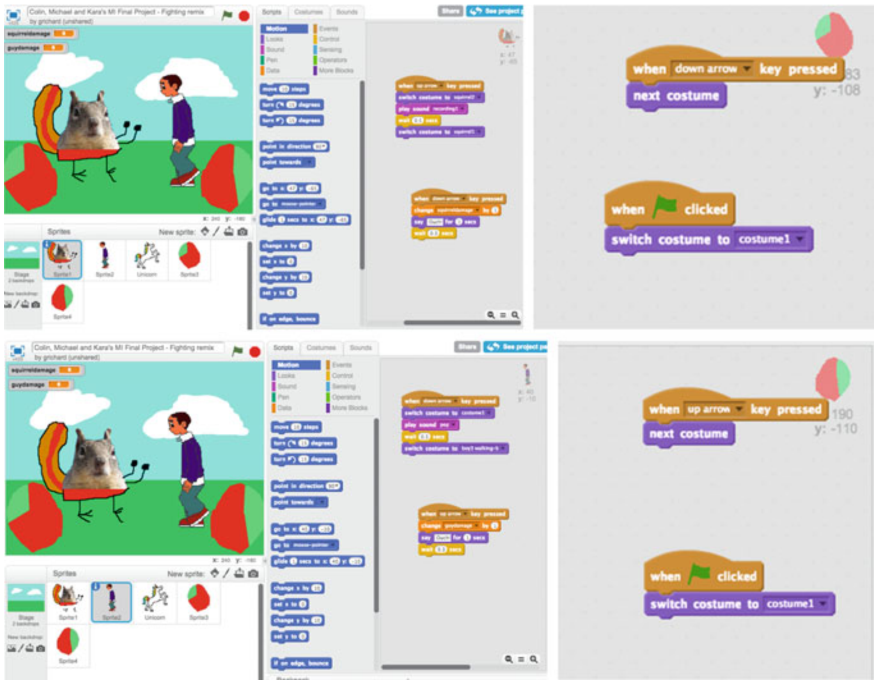


**Fig. 8** Screenshots of different game states of the jousting game



```
when down arrow key pressed
next costume

when [flag] clicked
switch costume to costume1
```

```
when up arrow key pressed
next costume

when [flag] clicked
switch costume to costume1
```

**Fig. 9** Screenshots of the jousting game scratch project. (*Top Left*) Game interface and code for the Squirrel sprite on right. (*Bottom Left*) Game interface and code for the boy sprite on right. (*Top Right*) Code for the left pie-chart counter sprite. (*Bottom Right*) Code for the right pie-chart counter sprite

The conductive fabric was connected to the MaKey MaKey, such that the thumb piece was connected to ground, and the forefinger was connected to spacebar, which would be the normal controller for the game (see Fig. 10). As the player pinched their fingers together, the star would hop up on the screen. Finally, the sensor glove (see Fig. 7, right) created by a team of two students (1 male, 1 female). It included a multifunctional wearable device for controlling games in Scratch. Based on which thumb and finger combination was pressed together, there would be a different response in the game.

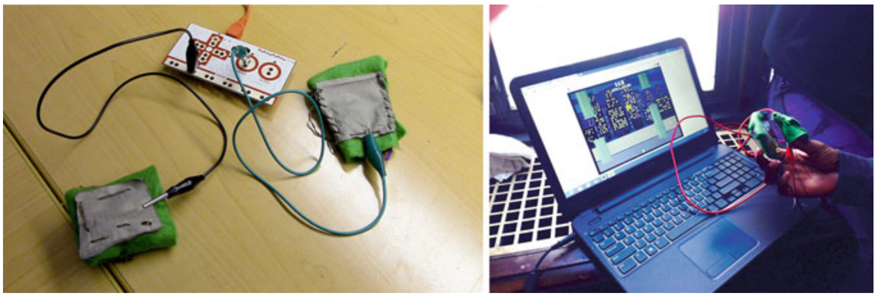


**Fig. 10** (*Left*) How the finger controller was connected to the MaKey MaKey's ground and space bar key, effectively replacing the space bar control for the game in Scratch. (*Right*) Students demonstrating their game's functionality. When the conductive fabric is touched, the star sprite leaps up. When they let go, the star sprite begins to fall

**The jousting game** was created by a team of three students, two male and one female. The project consisted of vests with conductive fabric, which were connected to different arrows on the MaKey MaKey, "swords" made out of metal coat hangers, which were connected through ground on the MaKey MaKey, as well as a fighting game, that kept score through a pie-chart interface, designed in Scratch (see Fig. 8). Each player would use the swords to block their opponents or hit the conductive parts of their opponents' vests, which would trigger a point in the fighting game, as well as show one of the on-screen characters hitting or reacting to the other character. Each character would react with different sounds that matched the creative character design.

Each team member contributed a unique skillset to the team, which made working together dynamic in meeting their design goals. According to Colin, "we all had different skills. I was better with the computers and Michael was good with the stitching and Kara was good with the sprite design." However, Michael also displayed a strong aptitude for working with crafts and materials, in general. As he explained:

> Making the wearable sensors was what I wanted to get out of the game. The partner I had - Collin - he did a lot of the computing and a lot of the programming. I did more of the outside stuff because that's where I felt like I was enjoying myself, creating stuff for the outside, rather than the programming…. I had thought about doing that the entire time with copper wiring and just using some wood, but I wanted something that was conductive. I saw a rack of hangers, and I was like, 'I remember when I was a kid and I used to use those to mess around.' I was like, 'Oh, I could use them.' So I ended up using the coat hangers as a second idea.

In coding the digital game, the team benefitted from having an experienced programmer. While Colin had never worked with Scratch before, he had previous experience line coding in Basic. He found Scratch to be accessible for the kinds of on-screen interactions they wanted to create:

> Scratch, is really easy to use. It was another building block thing. Pretty much did that and made some sprites, record[ed] some sounds. What I thought was interesting about that was

> that each program was tied to a specific sprite instead of all of the program being this general, the entire playing field.

Colin was able to create object responsiveness by utilizing object-oriented programming on the sprites themselves (see Fig. 9). He was also able to eliminate line coding, which he felt made the coding process easier.

The students on this team expressed that their original idea also included incorporating the Lilypad Arduino, by having lights that would indicate when a person was hit, thus reacting in a similar way to the trivia game. However, they experienced difficulty figuring out how to attach the Lilypad and the MaKey MaKey to power supplies while also having two mobile interfaces. Michael explained:

> We didn't get the Lilypad finished and on it, but we had planned to have like a little light so you know when you're hit. We had a small vest and we had conductive fabric and it was just a patch and another patch. Hit a certain area, put something into Scratch, and things happen on the screen. To know if you were hit or not without having to at least glance over at the screen, having a small light around your shoulder so it will flash up into your eye so you know when you're hit. That's what we were planning on incorporating it with the Lilypad, but it was taking a lot out of what we were already doing with the MaKey MaKey. We had a lot of electricity around, so it was kind of hard to get going with the amount of time we had.

Despite the limitations, both Michael and Colin appeared to understand the affordances of connecting the two interfaces and how they would be used. Kara, unfortunately, by her own admission, had missed too many classes and did not fully understand how all of the systems would work together; however, she and the other team members felt she contributed to the idea and added to the design by creating the inventive sprites. While some were skeptical about why the group created a fighting game, Michael's explanation demonstrates a heightened awareness of the affordances offered through the different toolkits and systems:

> …[We] were thinking of random games we could make, random genres, thinking of each one, and out of the ones that we had, it seemed to be most simple to make tangible pieces for because that's what we really had our minds on was the tangible pieces we could make for it, the fighting game. It originally started out an idea that you could attach something to your wristband. You conduct electricity. You hit a certain spot, but that kind of slowly grew… I felt we made like a very small simulator. I felt like we made a fighting game but simulated for being able to do it in the real world, rather than hitting buttons on a controller.

Michael's description fully articulates a design rationale that considers both the material and tangible affordances they were trying to imbue and felt a fighting game fit the genre best suited for the wearable and visual experiences they wanted to create with the tools they had available. While the group did not create a fully bidirectional design, their design rationale, and expression of limitations that hindered its fully realized development, indicate that they understood the utility of bidirectional responsiveness and may have been able to achieve it had they had more time.

**The flappy star game** was also created by a team of three students (2 male, 1 female). They created a wearable, partial glove controller that used two fingertips and conductive fabric to control a modded version of Flappy Bird, which they created by "remixing" someone's Scratch version of Flappy Bird. The conductive fabric was connected to the MaKey MaKey, such that the thumb piece was connected to ground, and the forefinger was connected to spacebar, which would be the normal controller for the game. As the person pinched their fingers together, the star would hop up on the screen (see Fig. 10).

While they remixed an existing Scratch project, they were committed to creating something unique in and that required a significant amount of work. Makhi[1] explained how remixing the Flappy Bird into a different object meant changing some of the sprite's properties, as well as the gravity that was preprogrammed in the original game:

> [The other male team member] drew the star… I changed the background in it, so the change of color of the stuff. Also with the gravity of the game, I had to change that, because the way the star was made, it wouldn't suffice. I learned how to change the way the game would have to react with the star size.

Originally, they were planning to have the light turn on in the physical glove when the conductive fabric was pressed. However, when they designed the final 2-finger controller, they realized there was not enough space to add the Lilypad, which they would have needed to control the LEDs. Even though they did not incorporate a physically interactive element in their wearable controller, they did find utility for it early on in the design process. As Makhi explained:

> [Working with the LED] would help us show if it was working or not. Because without the light, I remember you taught us that the light would indicate whether or not it's working. We had no other way to tell if it was working except by going on the game. But, we didn't make the game yet. We had to make sure the thing was working. That was just like our little guide to see. That was probably a big part because it helps us make the game what it is, now.

In other words, they were able to use the Lilypad Arduino, conductive fabric and an LED early on in the design of their wearable controller, and the activation of the LED was used for troubleshooting. Instead of creating a fully functional bidirectional design, they used different aspects of what was learned in class in their design process. However, Makhi felt that the workshop allowed him to think differently about how systems are designed and put together to be responsive in different ways:

> …because we were working on hardware… Now I'm thinking, because we know how it works, I'm thinking how does it work. I want to go in depth with it…. I'm thinking to myself, how did they know that this spark information would go there? How did they get everything to work so precisely? How did they get it to be so small?

**The sensor glove** was created by a team of 2 students (1 male, 1 female) who were excited to explore the possibilities inherent in different kinds of tangible interactions within a wearable interface. They were less focused on the digital

[1]The other two team members were not interviewed. One declined, and the other did not obtain parental permission.

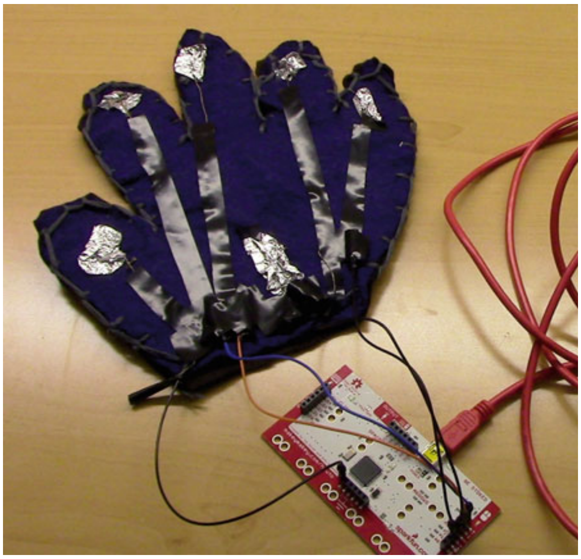


**Fig. 11** Close up of the sensor glove prototype with the MaKey MaKey

environment and wanted to spend time perfecting the glove controller, which could essentially work to control any game with arrow key functionality (Fig. 11). As Kia explained:

> Our team made… sensor gloves, with touch sensors on the finger tips that connected with the MaKey MaKey so that when you pressed the sensor on your finger, depending on which finger it was, one was up, down, left, and right, and those controlled, they were the sensors of the MaKey MaKey, like directions and thumb was jump… [through the spacebar] and that connected with the MaKey MaKey with our Scratch game that we made and all together, we got to control the game using the gloves that we made.

However, early on the team experienced difficulties fabricating the final physical prototype and seemed to take different directions with the design. During my observations, I noted that Liam was very focused on coding, and Kia was focused on creating the glove interface, though they often struggled to bring everything together when it came time to beta test. As conveyed by Kia:

> …since we were working on a separate computers sometimes there would be a miscommunication and I would be doing something and he would be doing something else and then they wouldn't be compatible with each other… I would have to go back and fix what we were doing and so I think working separately was kind of the problem, but we were able to make it work.

This most likely contributed to some of the technical errors they later discovered with the glove interface. I was able to observe it working to control the basic pong game in Scratch Liam had remixed, but, by the time they had to show it to outside observers, it had stopped working. Kia explained that "there were loose connections [in the glove] and that was mainly [the] problem." Despite the technical problems, Liam felt “if we had had about two more weeks then I’m sure we could have gotten it.” However, Kia was impressed that they were able to make a working glove controller that worked with Scratch through the MaKey MaKey, however, fleeting:

> We got it to work once and it was impressive for that one moment. And then everything fell apart, but I was really proud that we actually got this idea that we had in our heads and that prior to this I probably wouldn’t think that I could do [it]… I was proud that we could make the thing work… all by ourselves and it was pretty cool.

Further, through all of the difficulties encountered and troubleshooting, Kia, who had previously not worked with hardware and had no programming experience, felt she had a stronger understanding of the design process:

> I feel like I learned, not only a lot about technical skills, like programming and how to use Lilypad and Modkit and Scratch but I also learned how to troubleshoot, fix problems and how to… even when your project doesn’t work, how to make it so you can bounce back from that and if you can’t fix it, to think of another thing that can be used as a replacement and I think that was really important.

Moreover, she enjoyed the utility of working with all of the toolkits and systems, which she felt brought different skill sets together in a meaningful way:

> Our design was based off of the combination of [the Lilypad, wearable sensors, the MaKey MaKey and Scratch] and I think… if we were to take it away our design would have been completely different. I felt like that was what I liked about it… it was able to incorporate what different people liked and put it all into one project. It was able to get different skill sets. Again, Liam is very good at programming, I’m very good with craft stuff and it was good to merge those, otherwise it would have been a lot more challenging. I liked that.

## *5.2 Bidirectional Responsive Game Controller Design*

We now turn to a more extensive examination of one student team with Tuyet, a female designer, and Quinn, a male designer, who were able to successfully create a fully bidirectional responsive design. Their final project was an interactive trivia game in which players would have to answer questions related to historical questions appropriate for their grade level in school (9th grade). Some parts of this game were played on a colorful, flexible, felt game board made of conductive fabric, which, when triggered would roll digital dice on the screen in Scratch that would randomly generate a trivia question (see Fig. 12).

The design involved both the MaKey MaKey, which interfaced with Scratch and controlled the randomized trivia generating dice, as well as the Lilypad Arduino, which they hooked up to the same conductive fabric. When the conductive fabric was touched, two things happened: The interactive die would change on screen (and create a rolling sound), which would trigger the trivia question, and the LEDs on the game board would turn off to indicate that the fabric was touched, thus enabling a responsive element for players to respond to (see Figs. 12 and 13). They had also drawn unique characters, as game pieces, to go along with the game, though they

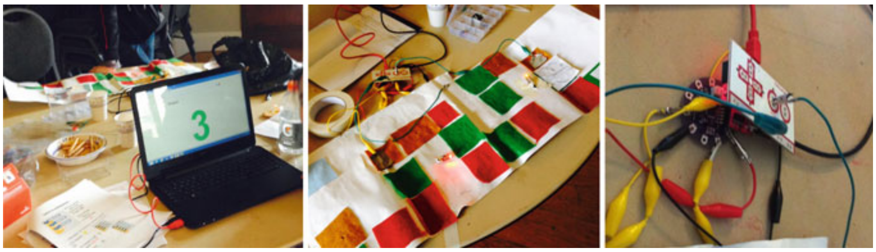


**Fig. 12** (*Left*) Demonstrating the digital dice functionality on the Scratch screen. (*Middle*) Close up of the felt game board with MaKey MaKey. (*Right*) Lilypad Arduino under MaKey MaKey with conductive parts of the board connected to both

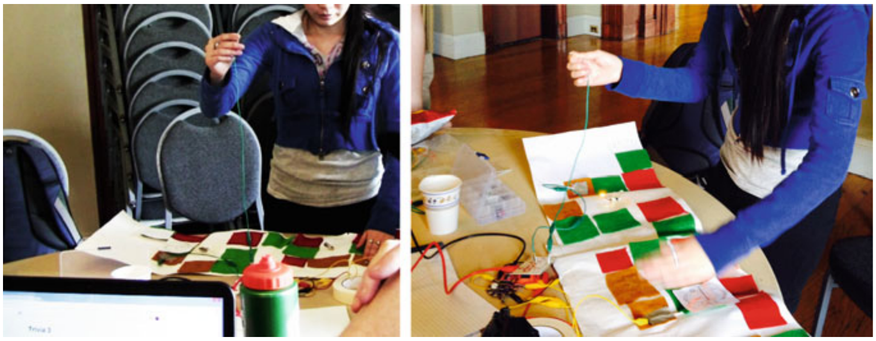


**Fig. 13** Tuyet controlling the game board. On *left*, when the game board is touched, the LEDs turn off. When she lets go, the LEDs turn back on

expressed that they were unsure how to make them interactive with the bidirectional interface, since they had been created on paper, and did not have time to add the conductivity to the individual pieces (Fig. 13).

The team was successful in finalizing their bidirectional responsive design by making several interesting but also judicious design decisions. For one, they spend most of their time on designing the hardware instead of programming, so they had time to work out usability issues in their trivia game. Furthermore, they programed the game by remixing an existing Scratch program. Remixing is a popular and encouraged activity in the Scratch online community, where members can repurpose code from programs and use in their own projects (Kafai and Burke 2014b). Finally, they used the prototyping materials by keeping the original alligator clips used for prototyping, instead of creating the final fabricated design by sewing down all of the connections with conductive thread.

In order to create their final project, they had to engage in understanding and integrating material affordances (such as conductive and nonconductive materials), coding and computation, and the design process. Tuyet explained[2] how initial misgivings about material affordances meant redoing the design but also learning a lot about conductivity and usability in the process:

> We found out that paper is not so friendly. (laughs) We found out glue is not so friendly… We didn't realize that we can't put wire on top of paper and then put cloth on top of paper and then expect it to stay in place, so we had to puncture holes in it… We found out that you can't always have it the way you want the first time… So trying to keep it nice and clean for everyone thinking it's beautiful and yet underneath looks like a tornado of just random wires and poked holes and glue and tape.

While she and Quinn were frustrated by having to redo the project, and it meant limiting their final design, they also learned a lot about the design process, and how to present a final interface that is both functional and understood by users. As expressed, they were confused about material affordances when they first started to design, and the instructor had to remind them that they needed a textile element to work with the Lilypad Arduino. For example, while they initially started with felt, they wanted to draw an interesting game character, which they could only figure out how to do on paper. When they tried to piece it all together, it did not work as intended, and the instructor had to remind them about using conductive materials, so the MaKey MaKey and Lilypad could pick up the values.

Furthermore, they had to engage in understanding and integrating coding and computation. For example, they originally wanted the LEDs to turn on (not off) when someone touched the board. This actually was partially a result of the limitations of the ModKit software. While an original version of ModKit allowed for reading values from serial communication, the current one did not, meaning that students actually had to go back and forth between previously developed code in Arduino (the C-based, line coding programming language used with Arduino products) that allowed them to read in the changes to the values when the conductive fabric was touched and then to ModKit to adjust the code to reflect when the lights should turn on and off. Unfortunately, despite the prewritten code, this proved to be challenging for this team (as well as others) and added additional stress and time to the design process. As a result of this, another limitation to their design was that they did not have time to make either the physical side or the software side as polished as they intended.

As a means of trying to get a finished product, they engaged in computation practices, such as remixing and reusing, that expert designers often use (Brennan and Resnick 2012; Kafai et al. 2014a, b). While they originally wanted to do more with the Scratch interface design, it was through this expert practice that they were able to come up with a working prototype by the end of the workshop. Despite integrating a previously designed project, they felt they had to learn coding in Scratch in order for this to be successful. According to Tuyet, they had to learn "the codes… the building blocks of Scratch—we had to learn the different meanings and how to use them properly." However, they had all of the pieces of a working

[2]Quinn declined to be interviewed.

prototype, which would have only required a minimal amount of additional time to get it to where they wanted it.

Finally, they had to bring all of their amassed knowledge together into a representative system. Tuyet described how they learned to connect the different components to make their game bidirectionally responsive:

> We had to learn how to control the lights. We knew how to turn on the lights, because that's really simple, but then we had to make them blink [when touched]. We had to make them do certain things to go with touch and react to what we wanted them to do. The MaKey MaKey system, which was used primarily for the dice to control on the board—that was easy because we took [a game they remixed in Scratch] that was already made. We just created, or deleted certain parts of that dice and then we used that for the MaKey MaKey system and plugged that into the board… It ended up being just a couple of wires. But that's OK.

Despite the limitations and frustrations experienced during the design process, Tuyet expressed how working with all of the toolkits and coding environments added to her understanding of bidirectional design and their affordances:

> Yes [combining all of the systems added to our design]. Originally it was quite simple, our design for our project. Then as it grew we had multiple parts, which made it more dynamic in the fact that it was not just one simple board game that only had one function. It had multiple functions and you could do multiple things if you wanted it to… To combine all of them, it's not easy. They're quite interesting to use as a whole so when I used all of them… You got to see that they can also work cohesively with each other. They're just not one system does this, this system does this… They actually do work together. You just need to know how to use and connect the dots between them because they are not the same system whatsoever but they … can work together.

In conclusion, while only one team was successful in creating a fully bidirectionally responsive design, students from other teams revealed a better sense of the design process and had cleverly learned how to utilize prototyping and remixing strategies to test out functionalities and refine their design process. These case studies help to illustrate the trends we observed in the workshop: students not only got to engage in multiple design practices for the creation of complex projects with various physical and digital responses, but also learned to think about computation differently. Tuyet expressed how she came to think systematically about how all of the systems came together. The integration of multiple toolkits made their designs more dynamic and interesting. Further, by noting "they actually do work together," Tuyet expresses a sense of perceptual change fostered through the workshop and similarly expressed by other students, regardless of the outcome of their designs. For many students, the workshop did not just enable the potential for bidirectional responsiveness, but multiple entry points to learning about coding, computation, crafting, and tangible design. As a result of being able to integrate multiple skills and interests, and observing other projects that had been successful in achieving different aspects of responsiveness, most students walked away with a heightened understanding of bidirectionally interactive tangible design.

## 6 Discussion and Conclusion

Responsive wearable interfaces are becoming more commonplace in play, work, and school. Technologies such as *Google Glass*, *Oculus Rift, the WiiMote,* or even the *PlayStation 4* game controller, whose lights change to respond to on-screen game cues, have become highly visible and now also commercially accessible. However, designing such tangible and reactive interfaces is no trivial matter, and asking novice designers to do so turns out to be a challenging enterprise on both the tool and design level. In the following section, we discuss what we learned about tool and workshop design.

For creating the responsive interfaces, students needed to learn and design with four different tools—construction kits and programming languages—which included the MaKey MaKey that interfaced with Scratch and its block-based coding environment, along with the Lilypad Arduino that used ModKit its block-based programming language. We observed that most of the complexities for novices involved working with the Lilypad Arduino and ModKit, perhaps not surprising based on prior experiences we have of students learning to design with e-textiles (Kafai et al. 2014a, b). In contrast, the MaKey MaKey did not require any programming but let students engage with circuit design in a simpler way than the Lilypad Arduino. While learning Scratch programming seemed to come easier to the students, it is very possible this was due to learning about visual block-based coding earlier in the unit through ModKit. An additional affordance of working with Scratch was its integration of remixing, which allowed students to deconstruct the code behind other projects they liked in Scratch and wanted to incorporate with their workshop projects. While the tools were necessary, some participants clearly were more interested in the crafting and physical materials than engaging with serial communication and digital interaction design. Overall, a major constraint noted was the lack of time necessary to learn to master all the tools in order to make the kinds of fully functional designs students had conceived of.

Future implementation of such design workshops would benefit from a new generation of integrated tangible construction kits that could facilitate such responsive bidirectional designs. Rather than using four different tools, like we did in our workshop, students would only have to learn and interact with one tool, perhaps custom tailored for particular applications. For instance, the MaKey MaKey already simplifies bifocal design from what used to be an extremely complex process involving multiple sensors and back-end programming. There are already developments underway to achieve such designs. For instance, Sipitakiat and Blikstein (2013) have introduced the Pi-Topper, which adds to the Raspberry Pi by allowing it to work with sensors and actuators, creating physical computing capabilities in the Scratch programming and media design environment. However, understanding the utility of this toolkit for teaching bi-directional design still remains to be seen.

Providing integrated tools is one important support to introducing novice programmers to responsive design, while another equally important support is creating a workshop and curriculum structure that coordinates tool introduction and design phases. Based on the teaching experiences in this workshop and prior physical computing workshops (Richard 2008), a better model would involve using the previously created projects during this course as exemplars that can be deconstructed into project-based units. We are currently redesigning the curriculum such that it focuses on successively building from simple e-textile designs, to increasingly complex wearable interfaces that interact in both the physical and digital environments. These units will be used as buildable models of how to work with Scratch, the MaKey MaKey, the Lilypad Arduino, and ModKit, so that, once students come to build their own projects, they have done more focused building and troubleshooting within contained models they can use as building blocks to more complex design. Future work will explore how this newly tailored curriculum impacts students' designs and understanding of bidirectionally responsive and wearable game controllers and interfaces.

## References

Baafi, E., Reisdorf, C., Millner, A.: Modkit (Version Alpha) [Software]. Available from: http://www.modk.it/alpha (2013)

Blikstein, P. (2012). Bifocal modeling: a study on the learning outcomes of comparing physical and computational models linked in real time. In: Proceedings of the 14th ACM International Conference on Multimodal Interaction, pp. 257–264. ACM

Brennan, K., Resnick, M.: New frameworks for studying and assessing the development of computational thinking. Paper presented at the annual meeting of the American Educational Research Association, Vancouver, Canada (2012)

Buechley, L., Eisenberg, M., Catchen, J., Crockett, A.: 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, pp. 423–432. ACM (2008)

Davis, R., Kafai, Y., Vasudevan, V., Lee, E.: The education arcade: crafting, remixing, and playing with controllers for Scratch games. Proceedings of the 12th International Conference on Interaction Design and Children, pp. 439–442. New York, ACM Digital Library (2013)

Fowler, A., Cusack, B.: Kodu game lab: improving the motivation for learning programming concepts. In: Proceedings of the 6th International Conference on Foundations of Digital Games, pp. 238–240. ACM (2011)

Golsteijn, C., Hoven, E., Frohlich, D., Sellen, A.: Hybrid crafting: towards an integrated practice of crafting with physical and digital components. Pers. Ubiquit. Comput. **18**(3), 593–611 (2014)

Guzdial, M.: Exploring hypotheses about media computation. In: Proceedings of the 9th Annual International ACM Conference on International Computing Education Research, pp. 19–26. ACM Digital Library, New York (2013)

Hayes, E.R., Games, I.A.: Making computer games and design thinking: a review of current software and strategies. Games Cult. **3**(4), 309–322 (2008)

Honey, M., Kanter, D.E. (eds.): Design, Make, Play: Growing the next generation of STEM innovators. Routledge, New York (2013)

Kafai, Y.B.: Minds in Play: Computer Game Design as a Context for Children's Learning. Lawrence Erlbaum Associates, Hillsdale (1995)

Kafai, Y.B., Burke, Q.: Connected gaming: towards integrating instructionist and constructionist approaches in K-12 serious gaming. In: Polman, J.L., Kyza, E.A., O'Neill, D.K., Tabak, I., Penuel, W.R., Jurow, A.S., O'Connor, K., Lee, T., and D'Amico, L. (Eds.): Learning and Becoming in Practice: The International Conference of the Learning Sciences (ICLS), vol. 1, pp. 87–93. International Society of the Learning Sciences, Boulder (2014a)

Kafai, Y.B., Burke, Q.: Connected Code: Why Children Need to Learn Programming. MIT Press, Cambridge (2014b)

Kafai, Y.B., Fields, D.A., Searle, K.A.: Electronic textiles as disruptive designs in schools: Supporting and challenging maker activities for learning. Manuscript Submitted for Publication (2014a)

Kafai, Y.B., Searle, K.A., Fields, D.A., Lee, E., Kaplan, E., Lui, D.: A crafts-oriented approach to computing in high school: introducing computational concepts, practices and perspectives with e-textiles. Trans. Comput. Educ. **14**(1), 1–20 (2014b)

Kushner, D.: The making of arduino. IEEE Spectr. **26** (2011)

Lee, E., Kafai, Y.B. Davis, R.L., Vasudevan, V.: Playing in the arcade: designing tangible interfaces with MaKey MaKey for Scratch games. In: Nijholt, A. (ed.) Playful User Interfaces: Interfaces that Invite Social and Physical Interaction. Gaming Media and Social Effects. Springer, Singapore (2014)

Martinez, J.I.: Craft, click and play: crafted videogames, a new approach for physical digital entertainment. In Proceedings of the 2014 Conference on Interaction Design and Children, pp. 313–316. ACM Digital Library, New York (2014)

Millner, A.D.: Computer as chalk: cultivating and sustaining communities of youth as designers of tangible user interfaces. Doctoral Dissertation, Massachusetts Institute of Technology (2010)

National Research Council: Learning Science through Simulations and Games. The National Academies Press, Washington (2011)

O'Sullivan, D., Igoe, T.: Physical computing: sensing and controlling the physical world with computers. Cengage Learning (2004)

Qiu, K., Buechley, L., Baafi, E., Dubow, W.: A curriculum for teaching computer science through computational textiles. In: Proceedings of the 12th International Conference on Interaction Design and Children, (pp. 20–27). ACM (2013)

Resnick, M., Berg, R., Eisenberg, M.: Beyond black boxes: bringing transparency and aesthetics back to scientific investigation. J. Learn. Sci. **9**(1), 7–30 (2000)

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., Kafai, Y.: Scratch: programming for all. Commun. ACM **52**(11), 60–67 (2009)

Richard, G.T.: Employing physical computing in education: how teachers and students utilized physical computing to develop embodied and tangible learning objects. Int. J. Technol. Knowl. Soc. **4**(3), 93–102 (2008)

Silver, J., Rosenbaum, E., Shaw, D.: MaKey MaKey improvising tangible and nature-based user interfaces. Proceedings of the ACM Tangible Embedded and Embodied Interaction, pp. 367–370. Kingston, Ontario (2012)

Sipitakiat, A., Blikstein, P.: Interaction design and physical computing in the era of miniature embedded computers. In: Proceedings of the 12th International Conference on Interaction Design and Children, pp. 515–518. ACM (2013, June)

Swalwell, M.: The early micro user: games writing, hardware hacking, and the will to mod. In: Proceedings of Nordic DiGRA (2012)

Tanaka, K., Parker, J.R., Baradoy, G., Sheehan, D., Holash, J.R., Katz, L.: A comparison of exergaming interfaces for use in rehabilitation programs and research. Loading…, 6(9): 69–81 (2012)

Tanenbaum, J.G., Williams, A.M., Desjardins, A., Tanenbaum, K.: Democratizing technology: pleasure, utility and expressiveness in DIY and maker practice. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 2603–2612. ACM (2013)

Vasudevan, V., Kafai, Y.B.: Make and play for learning: computational thinking and participation in high school students' collaborative design of augmented board games. Paper presented at the annual meeting of the American Educational Research Association, Chicago (2015)