PingPong++: Community Customization in Games and Entertainment

Xiao Xiao[1], Michael S. Bernstein[2], Lining Yao[1], David Lakatos[1], Lauren Gust[1], Kojo Acquah[1], Hiroshi Ishii[1]

[1] MIT Media Lab
75 Amherst St.
Cambridge, MA 02142
{x_x, ishii}@media.mit.edu

[2] MIT CSAIL
32 Vassar St.
Cambridge, MA 02139
msbernst@csail.mit.edu

ABSTRACT
In this paper, we introduce PingPong++, an augmented ping pong table that applies Do-It-Yourself (DIY) and community contribution principles to the world of physical sports and play. PingPong++ includes an API for creating new visualizations, easily recreateable hardware, an end-user interface for those without programming experience, and a crowd data API for replaying and remixing past games. We discuss a range of contribution domains for PingPong++ and share the design, usage, feedback, and lessons for each domain. We then reflect on our process and outline a design space for community-contributed sports.

Categories and Subject Descriptors
H.5.1 Multimedia Information Systems

General Terms
Design, Documentation

Keywords
Sports Interfaces, Exertion Interfaces Crowd-contribution, Community Customization, Open-Source, DIY

1. INTRODUCTION
While software games like CounterStrike and Minecraft thrive on community customization and community-contributed content [16], sports and games in the physical world are fundamentally disconnected from "the crowd". This separation is not surprising: Shirky argues that crowds succeed when the Internet can lower the cost of organizing groups [19], but sports activities are highly local and physically embodied, and these physical elements are difficult to share online. As a result, sports cannot share the same community-driven benefits as online games: new arenas and rule sets, experimentation, remixing and hacking.

In this work, we explore open hardware and software platforms as a potential solution to this challenge. The core contribution of this paper is *community-contributed sports*: translating the concepts of community customization and community contribution into the world of physical sports and play. In this paper, we introduce *PingPong++*, an augmented ping pong table that embodies these Do-It-Yourself (DIY) and sharing principles in its design.

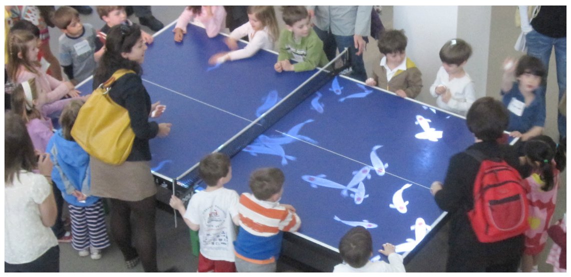


Figure 1. Children playing with the Koi Pond visualization in PingPong++. The fish swim away from the disturbance when the ball hits the table. This visualization was remixed using the PingPong++ API and an existing Processing sketch on openprocessing.org [18].

PingPong++ reinvents the PingPongPlus project from 1997 [8], which could track and visualize ball position during play, into a community platform. It does so through a range of contribution opportunities:

- **Open-source Hardware:** PingPong++ is made using commodity parts and can be copied by taping sensors to the bottom of a table and aiming a projector at the table surface. The open-source Arduino platform and vibration sensors are available cheaply and widely. For those without electronics expertise, we also make pre-printed circuit boards available.
- **Visualization API:** any member of the community can use a straightforward API on top of the popular Processing platform to build new visualizations that get projected onto the table. Contributors have created visualizations ranging from ambient art to training tools.
- **Crowd Data API:** all games played on PingPong++ tables are uploaded to a community server, where the data can be remixed, replayed, or used to answer questions like "Where on the table are hits most likely to score points?"
- **Instant Customization:** we have created a walk-up-and-use kiosk next to the table that allows visitors to author visualizations on the fly, without any knowledge of programming.

Members of our local community have already begun extending the platform and contributing new visualizations.

To succeed with PingPong++, we found it necessary to overcome significant motivational and effort barriers to contribution. Online communities reach enough people to build a core group of members even if only a small percentage of visitors actively participate. However, relatively small physical communities like office buildings (where PingPong++ is likely to be located) are less likely to survive with the same participation rate. This situation encouraged our design to focus on contributions with low thresholds [15] and high payoff, and to use online tools to share digital information like visualization code and crowd data between tables.

To follow, we review related work, then describe PingPong++ and its opportunities for community customization. As we share each customization design, we report on its usage, design feedback we received, and how it led to the next design. Finally, we reflect on the design space of community-contributed sports.

2. RELATED WORK

In this section, we review related work in sports and exertion interfaces, as well as research on open-source and community-contributed interfaces.

Sports and entertainment systems have begun integrating sensors and visual feedback. These systems often visualize progress to onlookers, for example in martial arts [3] and skiing [12]. In training, sports technologies are divided into those that train pure technique and those that develop strategy. For technique, Baca et al. developed GUIs for training ping pong hit accuracy and serving speed [1]. Interfaces for strategy have focused on video collection and the semi-automatic analysis of the video [1, 10]. Visualizations built by the PingPong++ team span this space: ambient visualizations like the Koi Pond reflect game status, while visualizations based on recorded games offer tactics and strategy.

Many projects in sports technology aim to enhance the enjoyment of physical activity [8, 20]. Several projects expanded the social space of players by enabling sports across distance [13, 14]. There has been a sharp divide in sports interfaces between the serious and the recreational. Serious sports interfaces focus on providing data for experts or those who wish to become experts while recreational interfaces focus on enjoyment but not improvement.

PingPong++ also draws on related work in DIY (do-it-yourself) and community contribution. Kuznetsov and Paulos survey several DIY communities and identify opportunities for engagement for HCI researchers, including for future projects [9]. Likewise, Ghosh found that contributors to open-source software are often primarily motivated by learning and developing skills [7]. Industry has pursued these questions through "open-source hardware" such as Arduino [6], as well as through community contribution of game content in games like CounterStrike, StarCraft and Minecraft [16]. Ducheneaut et al. argue that the performing for an "audience" in massive multiplayer online role-playing games (MMORPGs) like World of Warcraft incentivizes player contribution of game content [5]. Similarly, Bartle identifies a class of players in MMORPGs who build structures in the virtual worlds to gain social respect [2].

Our design philosophy follows a Batteries Included philosophy by including working starter code, pre-built boards, and wizards to create new visualizations. These approaches have seen success in electronics construction [11, 21] and in enabling children and other novices to quickly reach results [17]. The placement of the original PingPong++ table in a widely trafficked public space enable contributors to showcase their work to a constant audience of passersby.

To our knowledge, we are the first to translate the concepts of DIY to sports interfaces, and likewise to introduce sports interfaces to the domain of community contribution.

3. DESIGN AND EVALUATION

There are four ways users may contribute to the PingPong++ experience: open-source hardware, a visualization API, an instant-customization kiosk, and a web endpoint that accesses aggregate data from the ping pong table. For each design, we will describe the motivation, design challenges, user-end customization efforts, reflections from users, and what we have learned through the process.

PingPong++ followed an iterative design process, and so the evaluations reported in this paper are not summative, final user studies. We focus our evaluations on feedback from real deployments of the system and open PingPong++ workshops. Controlled evaluations of the designs are important future work.

3.1 Open-Source Hardware

Many sports interfaces require expensive hardware and significant expertise to build and recreate. Our goal with PingPong++ was to create a platform that could be easily duplicated by others. We did so by creating a sensing system that is both robust and easy to build using off-the-shelf components. PingPongPlus and PingPong++ use pieozoelectric sensors taped to the underside of the table to detect vibrations, which are used to track the ping pong ball position whenever it lands on the table. A nearby projector can then display feedback to the players by projecting directly onto the table.

For easy duplication, we redesigned PingPongPlus's hit position detection circuitry to interface with the commonly used open-source Arduino and Processing platforms. The Arduino calculates timing differences between pairs of sensors, and our Processing code derives the ball position based on the timing differences. The estimation training is identical to PingPongPlus [8].

By relocating the sensors on the table into a triangular pattern, we also improved position estimation accuracy. While the original PingPongPlus frequently yielded large errors (between 10 and 20

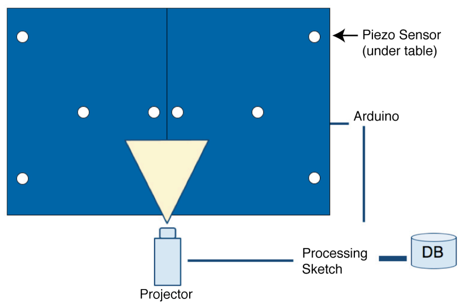


Figure 2. Hardware configuration: piezoelectric sensors in a triangular configuration under the table sense ball hits. The Arduino estimates ball location, then the Processing sketch visualizes the game state by projecting onto the table surface.

inches), our system reliably detects hits with an error of under 2 inches. In a trial where we tested 36 evenly distributed locations on the table that were not original training points, the mean error distance was 1.87 inches (std. dev. = 1.05). We have implemented an automatic calibration script written in Python with an instructional interface directing users to drop the ball at set locations on the table so that the sensors only need to be placed in approximate locations to duplicate the design.

Although our PingPong++ table runs on a breadboard circuit, we quickly learned that many sports enthusiasts would not want to wire a breadboard. To reach those with limited electrical engineering knowledge, we have designed a working printed circuit board (PCB) for the circuitry. This prefabricated board uses common electronics parts and costs under $30 to manufacture. With the PCB, building a table involves little more than taping piezo sensors under the table and running our calibration script.

Evaluation: To evaluate the difficulty of recreating the hardware, we deployed another instance of PingPong++ elsewhere on the MIT campus. We asked an undergraduate with almost no experience in electronics to wire the PingPong++ circuit on a breadboard, and she was able to build a working circuit in a single afternoon. Installation of the second PingPong++ took under two hours. We are currently working with two other undergraduates who are interested in installing PingPong++ in their own homes with our working PCBs.

3.2 Visualization API

Existing augmented sports interfaces have largely been single-purpose, where functionalities are envisioned and implemented by the creators of the interfaces. End-users may try out the interfaces and can provide suggestions to the creators, but the creators have the final word on the design. With the belief that a user of an augmented sports interface may have more domain expertise in sports than the creators, we aimed to empower the end-users of PingPong++ by giving them direct control over visualizations displayed on the table.

We pursued end-user customizability first by providing a software library written in Processing. Processing interfaces well with Arduino for hardware-software combination projects. It is also a visually-oriented language that is relatively easy to learn for programming novices: it was originally aimed at artists. Finally, there is also an existing community of Processing programmers who actively post example visualizations and code on websites like openprocessing.org. Many of these visualizations can be quickly adapted to the ping pong table, which makes it easy for new contributors to remix and edit existing code into new ping pong visualizations.

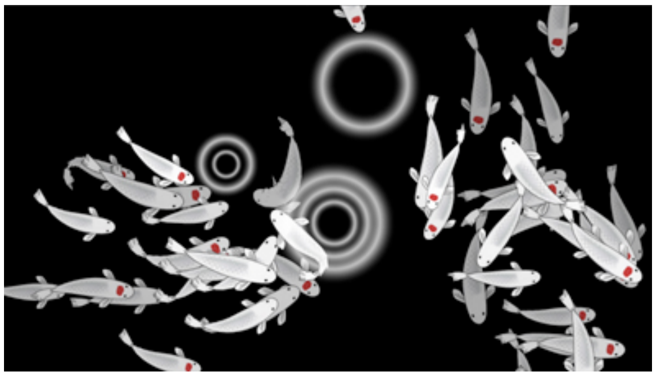


Figure 3. Screenshot of the Koi Pond visualization, which we provide as an example to volunteers wishing to write their own. Ping pong ball hits cause ripples in the table and scare the fish away. Based on Processing sketch by Sanchez [18]

To make visualizations easy to write, we created a Processing class called `BallPositionSensor`, which abstracts the math of the sensors from the user. Visualization authors only need to poll the current ball location `(x, y, sideOfTable)` to write visualizations. We then created a template and several initial visualizations for PingPong++ that users can reference. All of the PingPong++ processing visualizations can also be prototyped away from the ping pong table using mouse clicks to simulate ball hits.

Evaluation. To evaluate the ease of use of our Processing library, we recruited 7 volunteers, all MIT undergraduates and graduate students, to design and implement their own visualizations for PingPong++. While the general technical expertise at MIT is high, we emphasize that these volunteers were far from expert programmers: only two planned to be professional software engineers, and they were both freshmen. We present three visualizations programmed by three different volunteers and discuss the volunteers' own comments on their experiences. These visualizations were created over the course of roughly a week in the participants' spare time.

- **Constellation** displays target points on the table that turns into stars when hit. When all the stars are "discovered", it displays the stars as a constellation. This visualization was created by a mechanical engineering undergraduate with a small amount of programming experience.
- **Blocks** transforms ping pong into an arcade-style game where each player tries to destroy sections of similarly colored blocks on the opponent's side to earn points. It was created by a graduate student with an industrial design background with almost no programming experience.
- **Munchkin Run** features randomly moving creatures on each side of the ping pong table that get captured to the other side when hit by the ball. It was created by a freshman with a small amount of programming experience in Java.

Learning how to better program was the motivation for these three volunteers and for most of the others who wrote visualizations. Volunteers were also motivated by their interest in PingPong++ and potentially seeing their work featured on a popular installation in the community. Our volunteers indicated that while our examples were relatively straightforward to mimic, it took some

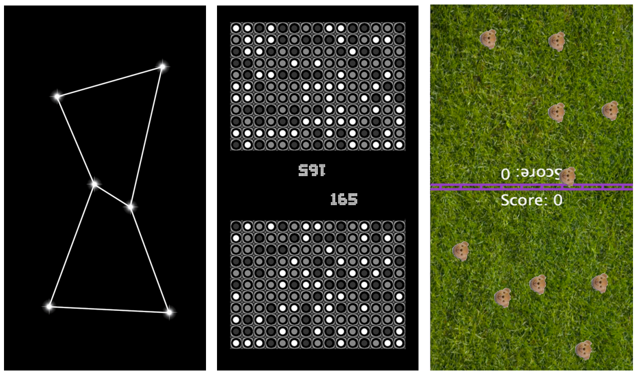


Figure 4. Screenshots of Constellation (left), Blocks (center), and Munchkin Run (right), three visualizations created by nonexpert programmers who had no experience with the PingPong++ API.

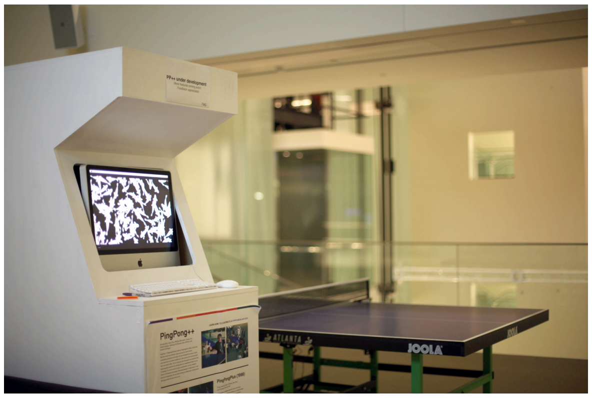
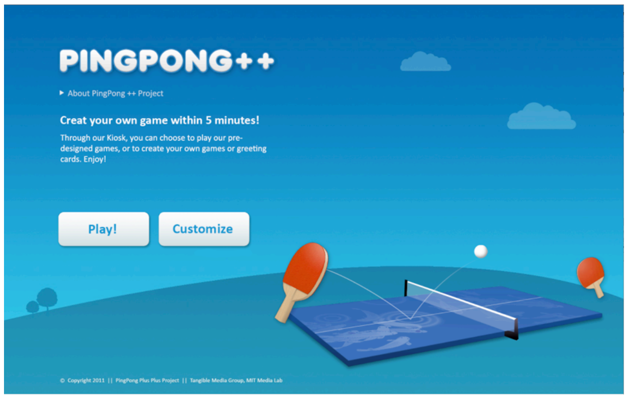


Figure 5. The instant customization kiosk next to PingPong++.



▸ About PingPong ++ Project

Creat your own game within 5 minutes!

Through our Kiosk, you can choose to play our pre-designed games, or to create your own games or greeting cards. Enjoy!

© Copyright 2011 || PingPong Plus Plus Project || Tangible Media Group, MIT Media Lab

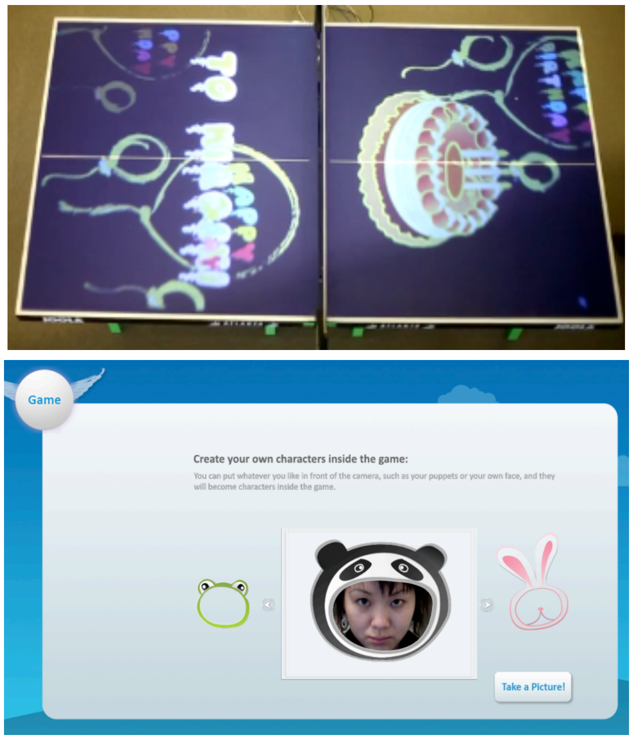
Figure 6. The kiosk homescreen allows users to choose from existing modes or create their own.

time to learn the basics of Processing itself given no prior experience. Even the volunteer with prior knowledge of Java (the language Processing is based on), required a few days to accustom himself to Processing by working through examples and tutorials.

3.3 Instant Customization Kiosk

Based on the experiences of our volunteers, we realized that writing a custom visualization for PingPong++ is not accessible for most users because it requires a level of fluency in programming and in Processing. We wanted to enable more people to be able to customize their PingPong++ experience, so we built an instant customization kiosk for the ping pong table.

Our kiosk sits next to the PingPong++ table and houses both the projector and a computer with a GUI interface. Using our interface, users can choose to play an existing visualization or customize their own. We noticed that many of the visualizations in our prior study could be abstracted into common templates. For example, one class of visualizations can be defined by having an image appear whenever the ball lands on the table. Another class of visualizations can be defined by moving existing images on the table toward the hit point. Our eventual goal is to provide the user with a set of templates that cover a range of interactions with the table and a library of graphical and audio assets. In the future, users will be able to build visualizations by choosing a template and modifying its visual and audio elements such as the background graphics, sprite images, or the sound that plays during a ball hit by selecting from the asset library.



Create your own characters inside the game:

You can put whatever you like in front of the camera, such as your puppets or your own face, and they will become characters inside the game.

Figure 7. Birthday mode in action (top) and the photobooth of the kiosk to customize Munchkin Run (bottom).

For the moment, our kiosk is still a work in progress. As a first step we have implemented two customizable visualizations that represent two degrees of customization capability. In our prototype, we have included a photobooth feature in our kiosk that allows users to input images from the physical world into PingPong++ visualizations.

- **Card** was inspired by a PingPong++ birthday card made by two graduate students in our group. It allows users to specify the background image, text, and the image to appear where the ball strikes the table. Our customized card draws balloons wherever the ball hits the table. Another card may replace the balloons with other images or modify the text.
- **Game** is a customizable version of Munchkin Run. Instead of the default munchkin graphic, a picture of the user's face is taken and used as the face of the munchkins moving around the table.

Card mode gives users several design choices while Game mode only requires users to take a picture of their face. We are currently developing additional templates for PingPong++.

We have not yet conducted any careful usage evaluations of the kiosk. However, we have already noticed that its physical presence next to the table has encouraged visitors to interact with it much more readily than they would visit a web site and install Processing.

3.4 Crowd-Data API

The techniques discussed so far are relatively simple, and do not have a very high ceiling [15]. In short, they only react to each instantaneous ball hit. We wanted to enable the creation of more advanced visualizations, both recreational and serious. To do so, we set up a server that aggregates data from all PingPong++ tables and returns information appropriate for visualizations.

Our server, called Pong, exposes one primary endpoint for recording a hit. Each time the table senses a hit, it sends the server information about where the hit was, when the hit occurred, and which table the hit was on. The server places this information in a MySQL database. We modified the `BallPositionSensor` class in the Processing code to upload this data automatically. The server itself is written in Python using Django.

We then created a set of Python classes to provide access to the data for programmers. Using these classes (based on Django's Object Relational Mapper), programmers can create complex endpoints and do historical calculations on the ping pong games. We believe that this high-threshold, high ceiling approach will open the door to deeply innovative techniques from the most dedicated contributors.

As examples of how to use aggregate data, we built the following three visualizations aimed to provide real-time pedagogical feedback on a user's ping pong playing:

- **Defensive Heatmap:** tells plays where on the table they should prepare to defend. Based on where a player hits the ball, Defensive Heatmap lights up areas of the table where the opponent is likely to return the hit. This data is aggregated from every past game. The visualization updates the player's side of the table every time the ball hits the opponent's side.
- **Offensive Spotlight:** When the ball falls on the player's side of the table, the offensive spotlight animates a circle to show where the player should return the hit to have the highest chance of scoring. The animated circle starts at a large radius and closes in on the recommended return location, making it easier to see in the middle of a heated game. Given a ball location, Offensive Spotlight searches the database to find all hits that happened immediately after a ball landed in that location. It then filters the return hits to find ones that ended a volley—meaning that the player scored a point on the play. Finally, the interface takes the single most successful return location and recommends that the user hit there.
- **Expert Arrows:** the Expert Arrows visualization focuses on learning from expert players. As the ball falls on the player's side of the table, Expert Arrows draws a large white arrow from the ball location to the point on the table where expert players tend to return the hit. Expert Arrows uses the same algorithm as the offensive spotlight; however, it only considers data from games played between two experts. We recorded games between the top 15 ranked players in our lab's ping pong ladder, out of 55 known individuals. This expert data consisted of nine full games, or 7,269 hits over a period of three and a half hours of gameplay.

Data from PingPong++ has also been used by the DoppelLab, a virtual model of the MIT Media Lab with a virtual ping pong table that plays a game of pong corresponding to hits from the physical table [4].

5. DISCUSSION AND DESIGN SPACE

PingPong++ followed a highly iterative design process. In this section, we reflect on the process and the design space that the visualization API, open hardware, kiosk, and hit database explore.

Within a closed environment like an office building or common area, we found easily extensible software to be the first step toward building a community. The visualizations followed a "do

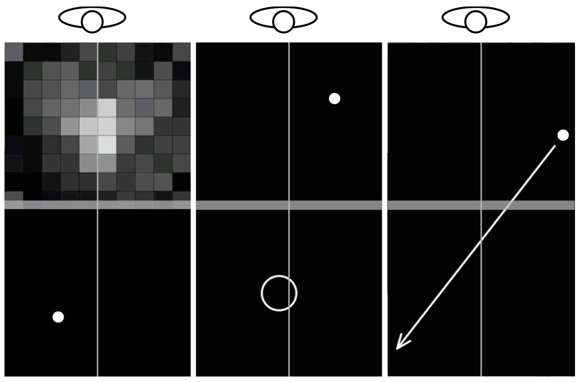


Figure 8. Defensive Heatmap (left), Offensive Spotlight (center), and Expert Arrows (right) with the ball hit that triggered the display. The player for whom the visualization is intended is at the top.

no harm" principle: at worst, even with buggy visualizations, the ping pong table would be just as playable. At best, the more engaging visualizations encouraged players to try their hand at authoring. Easily remixable starter code proved extremely important at helping visualization authors get started quickly. In addition, building on amateur programming platforms like Processing was important to bootstrapping a community of authors.

However, a visualization API requires authors to be motivated enough to go to their desks and write code. Instead, we found that in-band customization opportunities can engage a much broader crowd that is interested in tinkering with the visualization mechanics. Building the visualization API before the kiosk allowed us to learn common authoring patterns.

To spread the system beyond one prototype, it is important for the hardware to be easily replicable. Here we take inspiration from the blossoming DIY communities [6]. Where research projects typically rely on expensive hardware and advanced degrees in embedded programming, community replication needs simple instructions and off-the-shelf parts. We found that the circuit could be further encapsulated into a printed circuit board, which reduced the extra labor to mostly wire and tape. However, we believe that hardware spread is still a large challenge for community-contributed sports interfaces.

Finally, we believe that it is important to support community members who want to build complex applications on the platform. To this end, we automatically capture all games played on the

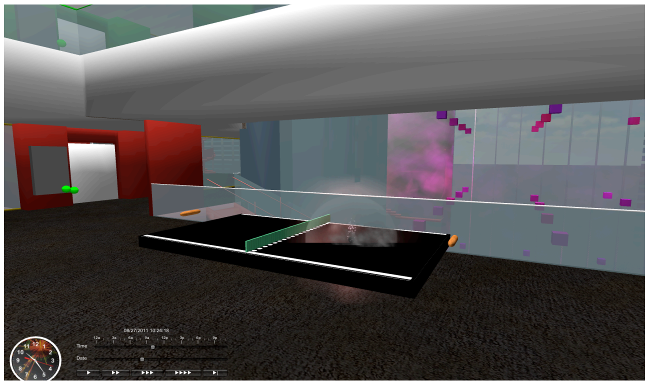


Figure 9. Screenshot of ping pong table simulation powered by PingPong++ data within DoppelLab [4]

table. This database endpoint has opened a broad range of other applications, like one off websites (e.g., a webpage "IsThePingPongTableBusy.media.mit.edu" which displays "YES" if the table has recorded a hit in the last 15 seconds), distributed play where players can see the activity on other tables in realtime, game replay, changing the spectator experience, and even the ability to keep score or change the rules of the game.

We believe that these elements – customizable visualizations, re-creatable hardware, and open data – are core to any community-contributed sports interface. While our system was built at MIT, most of the contributions described in this paper were not authored by technical people. We view this as a success. However, it will be important to do a longer-term study of the system's success outside of our immediate environment.

6. FUTURE WORK AND CONCLUSION

In this paper, we have introduced PingPong++, and with it, community-contributed sports and games. Where crowds have been successful at mobilizing for large tasks like Wikipedia and Ushahidi, it has been difficult to draw these benefits into the physical realm. We believe that this is possible using sports and games as a vector: communities play, experience the enhanced sports interface, and are motivated to personalize and improve the experience.

A yearlong deployment of the table, open hardware, visualization API, crowd data API, and more recently the kiosk have taught us much about the dynamics of community-contributed sports like PingPong++. Ping pong is an enjoyable way to learn to program, and a public showing of the resulting visualization is a high-payoff result from participating. It is important to support a range of motivations, from ten-seconds-to-experiment to overnight-hacking-session.

We believe that the most critical future work lies in wide deployment of the platform. We envision tens or hundreds of such tables, all networked together across the world. We are already beginning to publish the PingPong++ instructions online, but success will also depend on technical support and evangelism. We hope to see a day soon where community members will help each other set up new installations.

7. ACKNOWLEDGEMENTS

We are grateful to Nan-Wei Gong, Mark Feldmeyer, and Joseph Paradiso for their help with the hardware redesign. Thanks to Ricardo Sanchez for allowing us to adapt his Koi Pond code for our visualization. Thanks to the Tangible Media Group for their continual support.

8. REFERENCES

[1] Baca, A. 2008. Feedback Systems. In: Computers in Sport, P. Dabnichki and A. Baca, Eds. WIT Press, Boston, MA, 44-67.

[2] Bartle, Richard. 2003. *Designing Virtual Worlds*. Indianapolis: New Riders.

[3] Chi, E. H., Song, J., and Corbin, G. 2004. "Killer App" of wearable computing: wireless force sensing body protectors for martial arts. In Proc. UIST '04.

[4] Dublon, G., Pardue, L., Mayton, B., et al. 2011. DoppelLab. Ars Electronica.

[5] Ducheneaut, N., N. Yee, et al. 2006. "Alone Together?" Exploring the Social Dynamics of Massively Multiplayer Online Games. In Proc. CHI '06.

[6] Fried, L. and Torrone, P. Open Source Hardware. http://www.adafruit.com/blog/2009/03/28/open-source-hardware-overview-slides/. Accessed June 07 2011.

[7] Ghosh, R. A. 2002. Free/Libre and Open Source Software: Survey and Study, Part IV: Survey of Developers. International Institute of Infonomics University of Maastricht, Netherlands.

[8] Ishii, H., Wisneski, C., Orbanes, J., Chun, B., and Paradiso, J. 1999. PingPongPlus: design of an athletictangible interface for computer-supported cooperative play. In Proceedings of CHI '99. ACM, New York, NY, 394-401.

[9] Kuznetsov, S., and Paulos, E. 2010. Rise of the expert amateur: DIY projects, communities, and cultures. In Proc. NordiCHI '10

[10] Lames, M. 2008. Coaching and Computer Science. In: Computers in Sport, P. Dabnichki and A. Baca, Eds. WIT Press, Boston, MA, 100-119.

[11] Mellis, D., Gordon, D., and Buechley L. 2011. Fab FM: the design, making, and modification of an open-source electronic product. In Proc. TEI '11.

[12] Michahelles, F. and Schiele, B. 2005. Sensing and monitoring professional skiers. IEEE Pervasive Computing 4, 3 40-46.

[13] Mueller, F. 2008. Long-Distance Sports. In: Computers in Sport, P. Dabnichki and A. Baca, Eds. WIT Press, Boston, MA, 70-95.

[14] Mueller, F. and Gibbs, M. R. 2007. Building a table tennis game for three players. In Proc. ACE '07.

[15] Myers, B., Hudson, S.E., and Pausch, R. 2000. Past, present, and future of user interface software tools. *ACM Trans. Comput.-Hum. Interact.* 7, 1 (March 2000), 3-28.

[16] Postigo, H. 2007. Of Mods and Modders: Chasing down the value of fan-based digital game modifications. Games and Culture 2(4), pp. 300-313.

[17] Resnick, M. 1993. Behavior construction kits. Commun. ACM 36, 7 (July 1993), 64-71.

[18] Sanchez, R. 2010. Koi Fish Pond Processing Sketch. http://www.openprocessing.org/visuals/?visualID=28285.

[19] Shirky, C. 2009. Here Comes Everybody. Penguin Books.

[20] Wijnalda, G., Pauws, S., Vignoli, F., and Stuckenschmidt, H. 2005. A Personalized Music System for Motivation in Sport Performance. IEEE Pervasive Computing 4, 3 (Jul. 2005), 26-32.

[21] Wu, K. and Gross, M. 2010. TOPAOKO: interactive construction kit. In Extended Abstracts CHI '10.