Building a Toolkit for Fabricating Interactive Objects

Despite the recent proliferation of easy-to-use personal fabrication devices, designing custom objects that are useful remains challenging. RFID technology can allow designers to easily embed rich and robust interaction in custom creations at low cost.

By Andrew Spielberg, Alanson Sample, Scott E. Hudson, Jennifer Mankoff, and James McCann

DOI: 10.1145/2889427

For years, industry analysts have been predicting the breakout of in-home consumer fabrication devices such as laser cutters, desktop mills, and most prominently, 3-D printers. Despite falling costs and bold predictions of their increasing ubiquity, consumer demand for these devices remains relatively low. Behind this low demand is the fact that rapid fabrication devices alone are not replacements for mechanical design expertise or electronics knowledge, making it difficult for laymen to design objects that are interactive. This limits the typical design space to static objects or simple machines, in turn limiting the usefulness of these fabrication devices.

Upcoming electronic 3-D printers, such as the Voxel8, coupled with booming online maker file sharing communities (such as Thingiverse and 123D Make) offer one possible solution to this problem, potentially putting entire suites of electromechanical capabilities a mere download away. While downloading designs may be easy, modifying them would still likely require the same mechanical and electronics expertise needed to design them in the first place. To truly empower makers around the world, a better solution is needed.

The solution must be affordable for hobbyists; it cannot require excessive expert knowledge to design with; it must take up little extra space so it doesn't excessively constrain the design process; and any end user should be able to incorporate it into their designs regardless of fabrication method. Most importantly, the solution's interaction mode needs to be fast in order to guarantee users true interactivity.

That solution might just be a new spin on an old classic; a sensor that has long found limited use in the consumer transaction and commercial warehousing industries, but is still a relatively young player in the space of making and interaction: radio frequency identification (RFID). By incorporating RFID tags in the design process and managing interactions with these tags intelligently in software, we can easily make almost any object interactive with little user expertise. As an added bonus, RFID tags are powered wirelessly, meaning designers have no messy circuitry to deal with and no batteries are required.

AN EXAMPLE APPLICATION

In order to motivate the types of interactive objects we want designers to be able to make, consider the following scenario. A designer wants to fabricate a physical Tic-Tac-Toe game board with X and O game tokens that are interactive (see Figure 1). When a player places a token on the board, a computer running a companion application provides auditory feedback. When a player picks

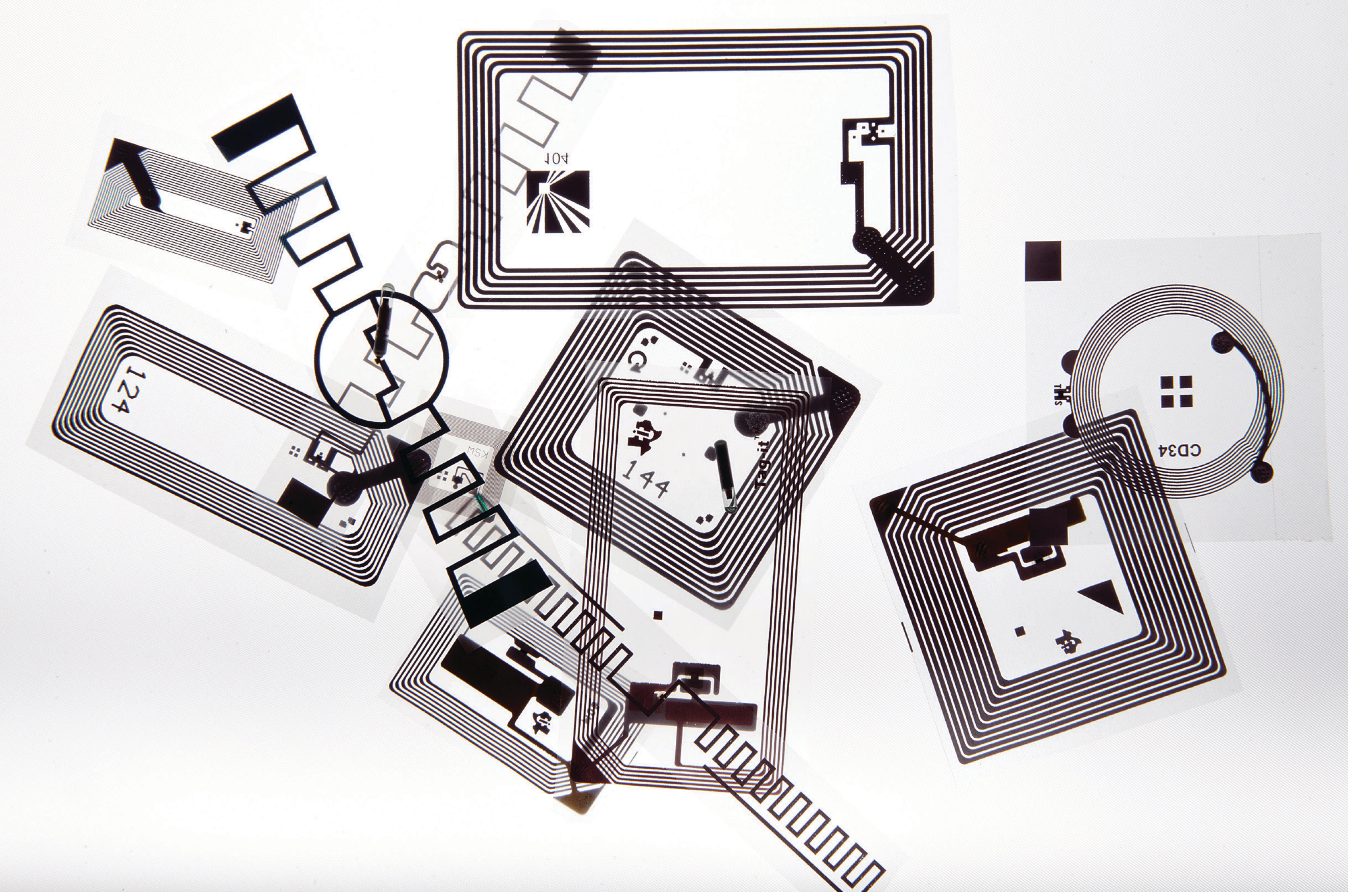


Image by Andrey Nikolaev

up a token, the system detects whether the token is an X or an O, and the application scolds the corresponding player to wait if it's not his or her turn. The application also keeps track of where X and O tokens are at all times, displays the current game with shiny graphics, and announces when a player has won or the game has ended in a tie.

There are two parts of this artifact that must be designed: the method by which a player interacts with the Tic-Tac-Toe board, and the specific Tic-Tac-Toe elements. The ideal solution should automate everything needed to implement the game's interactions—from geometric design of tokens through code needed to recognize token placement and motion—freeing up the designer to focus on the game mechanics and aesthetics. As you'll see, RFID tags can be the secret sauce for making this possible.

RFID

While you may not necessarily be aware of how RFID technology works from a technical perspective, you've almost undoubtedly experienced it at some point in your life. If you've ever seen or used a "tap to pay" credit or transit card, those interactions rely on embedded RFID tags to process the transaction. Large collections of physical media (such as libraries) use RFID tags for tracking inventories. Behind the scenes, RFID tags are used to track important parcels through storage and manufacturing processes.

Passive, ultra-high frequency RFID tags consist of millimeter scale computer chips attached to a (centimeter scale) antenna, typically embedded in an adhesive piece of paper. This chip holds data—a unique identifier and potentially kilobytes of storage. Radio signals transmitted by an antenna power tags wirelessly with transmit distances of up to 10 meters. When a tag receives a request for information, it reflects a signal to a reader with the appropriate response. (See Figure 2 for the components). This can be, for instance, its ID or any of its stored data. A single reader can be used to track populations of hundreds of tags. Tags are cheap, costing less than a 25 cents apiece.

Unsurprisingly, RFID technology has enjoyed plenty of breakthroughs in application to ubiquitous sensing; after all, one of the technology's killer apps, transaction processing, is essentially high proximity tag detection. But other prior work has demonstrated RFID tags' potential for tracking human interaction [1], measuring gestures such as tag touches, swipes on tag surfaces, tag motion, and even tag localization. Adding RFID tag-based interaction can potentially add great breadth of interactivity to the artifacts that users fabricate. Further, RFID tags and their rich vocabulary of interaction modes can be directly translated into default widgets to be included into designs, simplifying the design process.

While techniques for measuring RFID-based interaction are becoming capable of robustly identifying larger vocabularies of interaction, increased robustness has come with a price. Reliably identifying interactions can take seconds, which can be prohibitively slow for many real-world interactions. If you were trying to design, say, a video game controller that uses RFID tag-based interaction for input, the

Figure 1: The target Tic-Tac-Toe game.

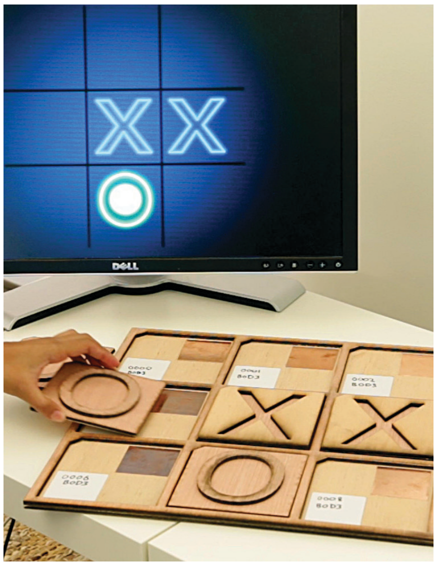
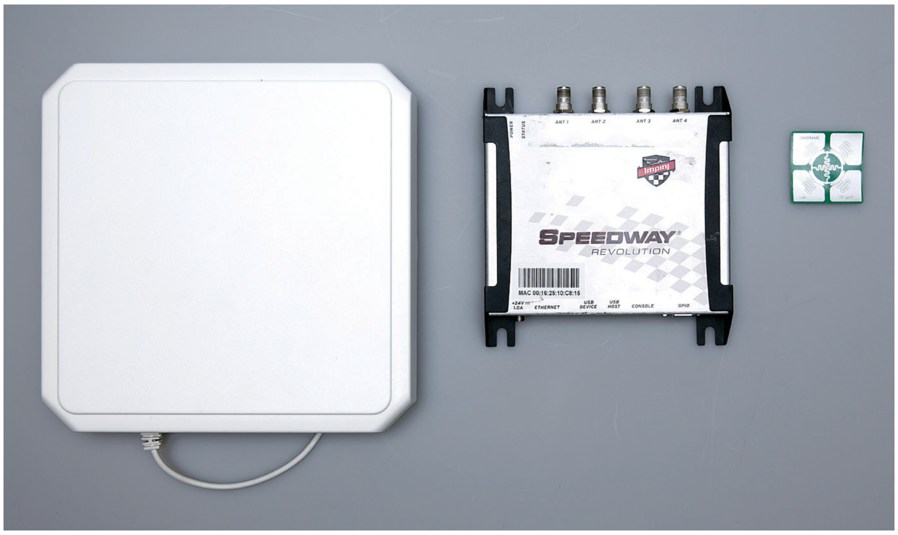


Figure 2: An antenna (left) emits radio waves to a 5.3 cm2 RFID tag (right), which reflects a response to the reader (center).



difference between a two millisecond and two-second input lag could be the difference between player success and player failure (and frustration). On the other hand, methods exist with faster response times, but these methods are historically less accurate. Misinterpreting one input for another could be similarly frustrating and damaging to usability. Thus, an important design trade-off arises for developers using RFID tags: Do you want your application to be fast or do you want it to be accurate?

It turns out there's a way to get the best of both worlds.

A SOLUTION IN INTERFACE DESIGN

We suspected from previous literature and experiments that if we wanted interactions to be truly low latency to the point where they felt natural (say, less than 200 millisecond latency), intelligently managing this speed-accuracy trade-off would be at the heart of our solution. Interaction researchers and developers have developed a number of probabilistic methods for managing uncertainty for more traditional human-computer interaction (HCI) domains, including keyboard, mouse, and touch screen interactions. Could we apply these methods to our problem of designing RFID powered objects providing real-time interactions? Previous work on managing uncertain interactions by Schwarz et al. [2] represented all interactions as probabilistic events, and the applications built atop them were transformed into probability distributions over program states. For example, if, on your phone's touch screen keyboard, you touched halfway between the "F" and "G" keys, then their system might represent the word you were typing as starting with "F" with a 50 percent probability, and starting with "G" with a 50 percent probability. Because of this uncertainty, their system won't lock in that first letter just yet; instead it waits for more information. Subsequent letters then shift the confidence of the word starting with "F" or "G" appropriately. If the second input letter is an "H" with 99 percent probability, then the first letter was probably a "G." (No words in the English language start with "fh," while many—including "ghost," ghastly," and "ghoul"–start with "gh.") In other words, their system automatically defers the occasional decisions it's not confident about, while maintaining very low latencies for the typical decisions for which it's confident.

While a probabilistic representation of interactions is powerful, exposing it directly to an application developer is dangerous. The average person is bad at probability. Asking a user to calculate explicit probabilities of higher-level states (such as the probability of certain sentences in our phone example) would never gain traction in an API. So, the clever idea Schwarz et al. had was to have developers specify deterministic interactions, keeping the programming of states intuitive, and let the API automatically manage the probabilities behind the scenes. Their program state distribution is represented as a collection of potential *state samples*, and those state samples (and thus the distribution) are updated with *input samples* drawn from input events such as key presses. Those input events are also represented probabilistically. This sampling approach is also known as Sequential Monte Carlo sampling (SMC) or particle filtering.

SMC is not specific to any single type of input; it can even manage inputs from several input modalities at once. So naturally, the SMC framework could extend to our RFID-based scenario if we considered RFID tag reads (or the lack thereof) as individual inputs (similar to key presses in the phone example). Ideally, such a system could detect program states it were unconfident about, then it could defer making decisions about them—perhaps only by milliseconds—and avoid misclassification. SMC could help measure and manage these confidences. However, applying the SMC method would mean we would need to model interactive input with RFID tags probabilistically. What types of inputs would we want to detect, and where would their models come from?

TOUCH AND GO

We decided early on to focus on modeling two modes of interaction for fabricating interactive devices: 1) touch events, that is, when a user physically touches and covers a tag; and 2) motion, that is, the velocity with which a tag is moving relative to a reader. These inputs would allow us to build a design API that would allow for reasonably large variety in interactive objects, including touch menus, token-based games, spinners, sliders, accelerometers, and so on. For example, in our Tic-Tac-Toe game, these tag interactions alone could be used to implement both token placement identification (using tag touch/cover) and token motion measurement (using tag motion). Further, input and touch events lend themselves well to salient features measured by readers about tags. For touch, the time between each tag's consecutive reads provides a strong indicator as to whether or not the tag is occluded by a conductive material such as foil or a dielectric material such as skin. For motion, the rate of change of the phase of the received radio wave for each tag gives strong clues as to how fast any given tag is moving.

Consecutive inputs are correlated. If a tag is moving with a certain velocity during one tag read, it will likely be moving with a similar velocity at the next read. Since tag touch events are far less frequent than tag state reads, it is likely a covered tag will remain covered between reads (and likewise for an uncovered tag). Therefore, we decided to keep running measurements of tag states using Bayesian filters. A Bayesian filter fuses previous state estimates with new measurements to constantly provide robust measurements of noisy systems. It's called a Bayesian filter because it does this through recursive applications of Bayes' rule. Given a sequence of $i-1$, state estimates $x_{1:i-1}=\{x_1,x_2\ldots x_{i-1}\}$, and sequence of i observations $z_{1:i}=\{z_1,z_2,\ldots z_i\}$ a Bayesian filter estimates the i^{th} state as

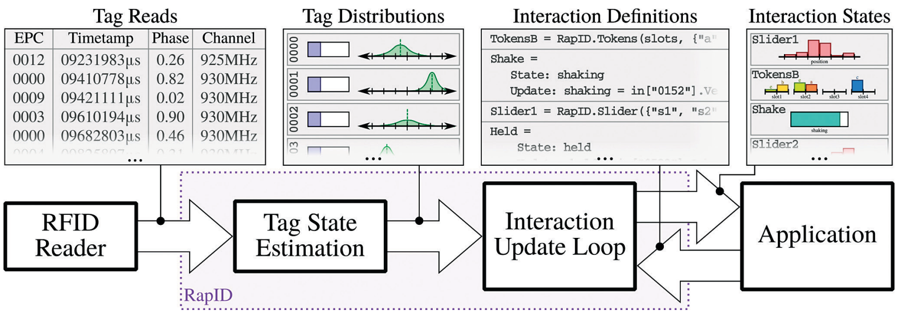
$$p(x_i|z_{1:i},x_{i-1})=\frac{p(z_i|x_i)p(x_i|x_{i-1})}{p(z_{1:i})}$$

Here $p(x_i|x_{i-1})$ is a hand-tuned Bayesian prior, which represents state evolution in the absence of observation, $p(z_i|x_i)$ is known as the measurement model, and $p(z_{1:i})$ is a normalization factor, constant over all hypotheses, which can typically be ignored. In our setting, the measurement model has a far bigger influence over the state estimate than the prior, and many priors work fine in practice. However, we found an intuitive solution is to simply bias the state estimate toward increasing uncertainty.

Now, you may be thinking that all of this is overkill. After all, RFID-based credit card transactions also rely on whether or not a tag is visible to detect interaction, and they trigger almost instantaneously as soon as a reader sees the tag. But that instantaneous transaction relies on a tag transitioning from a default state of *invisible* to a reader, to *visible*. Meanwhile, touch events begin when a tag is suddenly occluded, transitioning from a default state of *visible* to *invisible*. When touched, reads for a tag stop occurring, and so the presence of a touch event is actually described by the absence of data. When a tag is covered, the reader goes from to reading its presence every 50-200 milliseconds or so to every three seconds at best. Thus, as more time passes between subsequent tag reads, the less likely the tag is actually visible.

We performed a number of experiments over various tag population sizes where we recorded times between subsequent tag reads, including scenarios where the tags were covered and some scenarios where they were uncovered. From the data, we were able to build probability distributions of the times between tag reads in each of the covered and uncovered states, providing our measurement model. Coupled with our Bayesian filter and a prior, which rebiases the state estimate toward a 50–50 estimate of a covered/uncovered tag state, we can

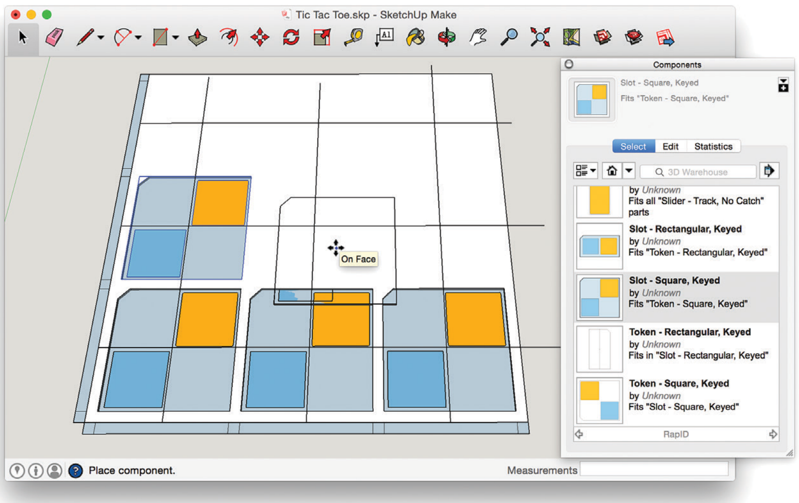
Figure 3. RapID's data-to-application pipeline.



EPC	Timetamp	Phase	Channel
0012	09231983µs	0.26	925MHz
0000	09410778µs	0.82	930MHz
0009	09421111µs	0.02	930MHz
0003	09610194µs	0.90	930MHz
0000	09682803µs	0.46	930MHz

```
TokensB = RapID.Tokens(slots, {"a"
Shake =
    State: shaking
    Update: shaking = in["0152"].Ve
Slider1 = RapID.Slider({"s1", "s2"
Held =
    State: held
```

Figure 4: Our Sketchup design environment; creating Tic-Tac-Toe.


Tic Tac Toe.skp - SketchUp Make
On Face
Components
Slot - Square, Keyed
Fits "Token - Square, Keyed"
Select
Edit
Statistics
3D Warehouse
by Unknown
Fits all "Slider - Track, No Catch" parts
Slot - Rectangular, Keyed
by Unknown
Fits "Token - Rectangular, Keyed"
Slot - Square, Keyed
by Unknown
Fits "Token - Square, Keyed"
Token - Rectangular, Keyed
by Unknown
Fits in "Slot - Rectangular, Keyed"
Token - Square, Keyed
by Unknown
Fits "Slot - Square, Keyed"
RapID
Place component.
Measurements

keep continuous estimates of tag state in the absence of tag reads, as well as update our distribution with more certainty when a tag read arrives. Using this method, we are able to measure up to 20 tags at once with latencies of at worst 200 milliseconds (and typically, our state estimation converges much faster than that).

Measuring motion is a bit different. Motion can only be measured when a tag is visible, so in this domain a reader is constantly receiving reads. Therefore, accurately measuring motion depends on processing information about the tag read, not just if and when a tag read happens.

In order to measure motion, we used some results in physics that state the velocity v of a tag is proportional to the change in phase $\Delta\phi$ between reads divided by time between reads Δt (normalized by the frequency f of the carrier wave). In other words:

$$v = \propto \frac{\Delta\phi}{f^{*}\Delta t}$$

This was a continuous domain rather than the discrete binary visible versus not visible domain of the touch event, so we had to record tag motion over many different velocity measurements. From this, we were able to build an empirical Gaussian of measured phase changes given known velocities, which, coupled with the running Bayesian filter, provides robust velocity measurements in practice (with built in error bars). In this case, we use a prior that maintains the previous velocity estimate and increases its variance, indicating increasing uncertainty in the estimate without information. Similar to the touch event scenario, updates are fast, and we are able to measure up to 20 tag velocities at once with latencies of at worst 200 milliseconds.

AN API FOR FABRICATION

With the technical details of how we could formulate RFID interactions as probabilistic events out of the way, we were able to carefully consider what our API (which we are calling "RapID") for fabricating interactive objects should look like. Figure 3 shows the entire pipeline of how data is translated into interaction with an application. But how should developers create experiences in such a paradigm? We knew we wanted developers to be able to create interactive programs, using RFID tags as inputs. We also knew the API for developing these programs should abstract the notion of probabilistic program states away from the users. Finally, in the same way we abstracted away much of the electromechanical design using RFID tags, we would need to abstract away as much coding as possible for novice programmers.

The powerful decision is to couple functionality with pre-defined physical widgets, combining the fabrication and interactive experience directly. This leads to two levels of fidelity with which developers could design physical user experiences.

At the high level, we developed a physical design environment as an extension to SketchUp, which is a CAD environment aimed at novice users. Figure 4 shows a user designing Tic-Tac-Toe with our design environment. We designed a number of physical widgets users can add in order to make their designs interactive: tokens, spinners, sliders, touchable surfaces, velocity sensors, and so on. After users finalize their design, they can export their designs to digital files for fabrication via 3-D printing or laser cutting, as per usual. RapID also exports two other important pieces of information.

First, in order to ease the programming of the digital side user applications, RapID exports starter code, which can be immediately run. This starter code keeps running estimates of widgets—which tokens are placed in which slots, the position of sliders and spinners, how fast an object is moving, and so on—and registers callback functions to monitor changes in these widgets' states. All users need to do is define these callbacks deterministically to say how the program state should be updated when the tag states change. RapID then uses those deterministic functions to update the probabilistic program state using SMC. RapID widgets also provide visualization methods for providing on-screen visual feedback of the probabilistic state deterministically. For example, RapID can render objects using the mean state for velocity-based widgets, or for tokens, render based on the most probable placement configuration. Our API is built with Unity, making it easy to build beautiful interactive media built on top of it. Second, in order to ease fabrication and assembly, RapID annotates design files with the locations where users should place RFID tags, along with

Figure 5. Our Pong demo application and sliders.

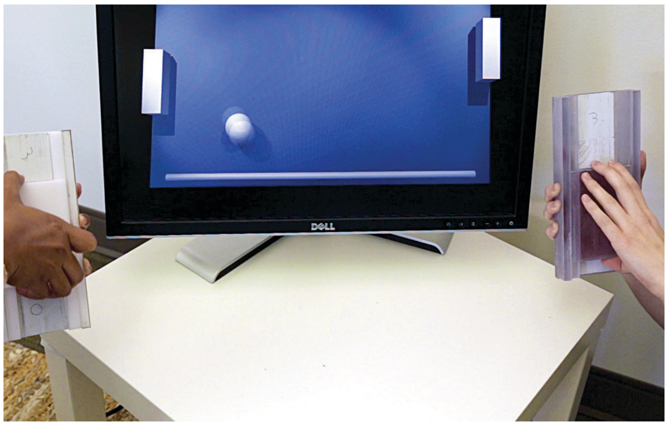


Figure 6. Our spaceship demo application.

the IDs with which those tags should be programmed.

At the low-level, though, we recognized our pre-defined widgets may not be expressive enough for all applications. That's why, for the experienced users, we exposed the lower-level API for interacting with the probabilistic program state. In order to make it possible for experienced users to develop their own physical widgets and their associated code.

While this project is by far not the first to allow users to build physical widgets that digital programs can be built on top of [3, 4], the fact that RFID tags, which are small and thin, have very few geometric constraints makes it very easy to place them anywhere in designs. This makes it easy to grow large, expressive widget libraries. In the future, it will be exciting to see how RFID tags and other similar, versatile sensors, will allow online communities to grow large widget libraries much in the way maker communities such as Thingiverse currently share pure .STL files.

PUTTING IT ALL TOGETHER

For now, we've created a few demo applications to show off the promise of a toolkit, which is a synthesis of our application pipeline (see Figure 3) and Sketchup Front-End (see Figure 4).

Using RFID tags on tokens and token slots, we were able to build a wireless, low-latency, physical game of Tic-Tac-Toe. Here, we used our token widget, which places tokens opposite conductive foil to measure whether or not token/slot pairs are visible. Using the IDs of the tags, we can identify which token is placed, when it is placed, and where it is placed. When the widget is added to the design, our Sketchup extension adds the appropriate token and slot geometry to the digital design files, and automatically generates all of the code for tracking this interaction. The only code the user has to add is the traditional deterministic game of Tic-Tac-Toe, and the visual and auditory feedback for the players, all of which can be written in fewer than 100 lines of C# code.

In another example (see Figure 5), we used our slider widget, which features a conductive cover that slides atop a line of RFID tags. Our automatically generated interaction code estimates the state of the slider based on which tags are visible to the RFID reader and which are masked by the cover. We 3-D printed one controller and laser cut the other (just to show we could), and painlessly coded up a flashy demo of the classic arcade game Pong using our sliders as wireless controllers.

As a final example (see Figure 6), we demonstrated our RFID tags' motion sensing capabilities with a simple spaceship-based demo. We designed a simple toy spaceship and placed a raw tag widget on the design, which, while not adding new geometry to the design, generated code for touch and motion callbacks. This demo is particularly friendly to novice programmers. Using our API, it was easy to translate our toy's motion to the digital on-screen motion of a virtual spaceship, only writing new code for on-screen animation. (Dong Nguyen, we eagerly anticipate your Flappy Bird port for our RFID-based system!)

CONCLUSIONS

As RFID tags become more robust and tag readers become cheaper with each passing year, RFID sensing is rapidly becoming a serious contender for making physical fabrication projects interactive. RFID sensing provides a platform that is easy to design with and even easier to interact with and use. A future where anybody can quickly fabricate wirelessly powered novel game controllers, smart-home devices, personal robots, and more is right around the corner. It will be exciting to see how other sensors can be hacked through similar data-driven methods, to go beyond their original intended purpose for use in interactive fabrication projects.

Through a combination of inexpensive, easy-to-use sensors, and more systems that marry physical design with digital design, people will finally feel empowered to make devices based on how they are meant to be used, and not just on how they are meant to look. Novice makers will finally be able to design and fabricate devices that fully capture the interactive nature of their imagination. And when interactive objects are as easy to make as static ones, the personal fabrication movement will truly be ready to take off.

ACKNOWLEDGMENTS

This work was conducted as part of a research project at Disney Research: Pittsburgh. This work is based on a paper to appear in the CHI 2016 proceedings.

References

[1] Li, H., Ye, C., and Sample, A. P. IDSense: A human object interaction detection system based on passive UHF RFID. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI 2015) (April 18–23, Seoul). ACM, New York, 2015, 2555–2564.

[2] Schwarz, J., Mankoff, J., and Hudson, S. E. Monte Carlo Methods For managing interactive state, action, and feedback under uncertainty. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Oct. 16-19, Santa Barbara, CA). ACM, New York, 2011, 235–244.

[3] Greenberg, S. and Fitchett, C. Phidgets: Easy development of physical interfaces through physical widgets. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology* (UIST 2001) (Nov. 1114, Orlando). ACM, New York, 2001, 235–244.

[4] Laput, G., Brockmeyer, E., Hudson, S. E., and Harrison, C. Acoustruments: Passive, acoustically-driven, interactive controls for handheld devices. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI 2015) (April 18–23, Seoul). ACM, New York, 2015, 2161–2170.

Biographies

Andrew Spielberg is a second-year Ph.D. student at the Computer Science and Artificial Intelligence Laboratory (CSAIL) at MIT, where he works in the intersection of fabrication and robotics. His current research exploits data-driven methods for optimizing the design and behavior of 3-D printed robots. His prior research has focused on automated assembly. Prior to joining MIT he received his B.S. and master's from Cornell University and spent time at The Johns Hopkins' Applied Physics Laboratory.

Alanson Sample is a research scientist at Disney Research, Pittsburgh where he leads the Wireless Systems group. His research focuses on enabling new guest experiences and sensing and computing devices by applying novel approaches to electromagnetics, RF and analog circuit design, and embedded systems.

Scott Hudson is a professor of human-computer interaction in the School of Computer Science at Carnegie Mellon University, where he serves as the founding director of the HCII Ph.D. program. He received his Ph.D. in computer science from the University of Colorado in 1986, and has previously held faculty positions at the University of Arizona and the Georgia Institute of Technology. Elected to the CHI Academy in 2006, he has published extensively on technology-oriented HCI topics, and recently received the Allen Newell Award for Research Excellence at CMU.

Jennifer Mankoff is an associate professor in the Human Computer Interaction Institute at Carnegie Mellon University. She earned her B.A. at Oberlin College and her Ph.D. in computer science at the Georgia Institute of Technology. Her research enhances the human experience with technology. Her goal is to combine empirical methods with technological innovation to construct middleware (tools and processes) that can enable the creation of impactful applications. Most recently, this work has focused on 3-D printing and its potential for creating custom assistive technologies for people with disabilities.

James McCann obtained his Ph.D. in 2010 from Carnegie Mellon University. His research hours are spent at Disney Research Pittsburgh developing systems and interfaces that operate in real-time and build user intuition; lately, he has been dabbling in the creation of physical objects. He also makes video games as TCHOW llc, including recent releases "Rktcr" and "Rainbow."