**University of Calgary**

**PRISM: University of Calgary's Digital Repository**

Science | Science Research & Publications

2018-07

# Astral: Prototyping Mobile and IoT Interactive Behaviours via Streaming and Input Remapping

Ledo, David; Vermeulen, Jo; Carpendale, Sheelagh; Greenberg, Saul; Oehlberg, Lora A.; Boring, Sebastian

Ledo, D., Vermeulen, J., Carpendale, S., Greenberg, S., Oehlberg, L., and Boring, S. (2018). Astral: Prototyping Mobile and IoT Interactive Behaviours via Streaming and Input Remapping (Report 2018-1107-06). Calgary, AB. : Department of Computer Science, University of Calgary.
http://hdl.handle.net/1880/107617
technical report

https://creativecommons.org/licenses/by/4.0
Unless otherwise indicated, this material is protected by copyright and has been made available with authorization from the copyright owner. You may use this material in any way that is permitted by the Copyright Act or through licensing that has been assigned to the document. For uses that are not allowable under copyright legislation or licensing, you are required to seek permission.

*Downloaded from PRISM: https://prism.ucalgary.ca*

# Astral: Prototyping Mobile and IoT Interactive Behaviours via Streaming and Input Remapping

**David Ledo[1], Jo Vermeulen[2], Sheelagh Carpendale[1], Saul Greenberg[1], Lora Oehlberg[1], Sebastian Boring[3]**

[1]Department of Computer Science, University of Calgary, Canada {first.last@ucalgary.ca}

[2]Department of Computer Science, Aarhus University, Denmark jo.vermeulen@cs.au.dk

[3]Department of Computer Science, University of Copenhagen, Denmark sebastian.boring@di.ku.dk

## ABSTRACT

We present *Astral*, a prototyping tool for mobile and Internet of Things interactive behaviours that streams selected desktop display contents onto mobile devices (smartphones and smartwatches) and remaps mobile sensor data into desktop input events (i.e., keyboard and mouse events). Interactive devices such as mobile phones, watches, and smart objects, offer new opportunities for interaction design– yet prototyping their interactive behaviour remains an implementation challenge. Additionally, current tools often focus on systems responding *after* an action takes place as opposed to *while* the action takes place. With Astral, designers can rapidly author interactive prototypes live on mobile devices through familiar desktop applications. Designers can also customize input mappings using easing functions to author, fine-tune and assess rich outputs. We demonstrate the expressiveness of Astral through a set of prototyping scenarios with novel and replicated examples from past literature which reflect how the system might support and empower designers throughout the design process.

### Author Keywords

Prototyping; design tool; interactive behaviour; smart objects; mobile interfaces.

### ACM Classification Keywords

D.2.2. Design Tools and Techniques – *User Interfaces*.

## INTRODUCTION

Smart interactive devices such as mobile devices, wearables and Internet of Things (IoT) objects vary widely in input and output, physical form, and development platforms. When prototyping interactive behaviors for these devices, designers are faced with two options: the first option is to build prototypes directly on the target device or platform. This involves programming, and in some cases, circuit building or soldering (e.g., physical computing, IoT devices). As a result, implementation details consume the majority of designers' time and resources (e.g., code setup, learning the platform at hand, programming basic visuals), instead of important early-stage high-level design decisions around interaction or animation or exploration of innovative alternatives [13]. The alternative option is to explore mobile device interaction with desktop-based prototyping tools (e.g. InVision, Balsamiq). Desktop applications often specialize in individual tasks (e.g. wireframing, prototyping standard UI widgets, creating animations as videos) without room for integrating different workflows [31]. Furthermore, desktop tools do not provide an accurate portrayal of what the interactions will be like on the target device [22], especially interactions that involve the target device's physical form, or possible tangible or embodied interactions (e.g. shaking the device).

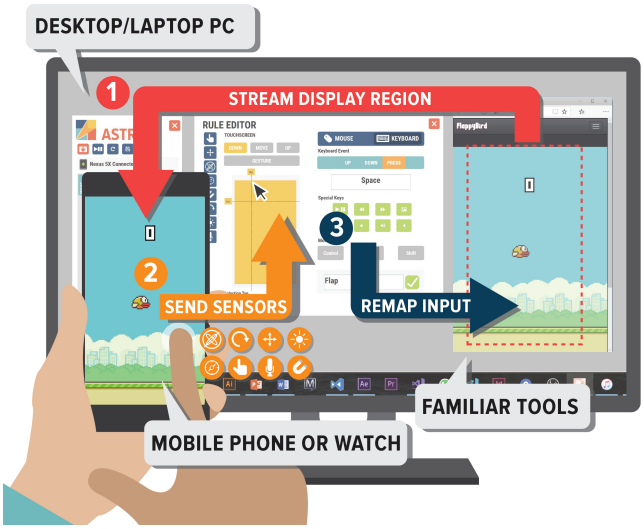


**Figure 1. Astral (illustration) allows designers to prototype interactive behaviours by (1) streaming contents of a desktop display to a mobile device, (2) sending mobile sensor data to the desktop, and (3) remapping the sensor data into desktop input (e.g. mouse and keyboard events).**

In this paper, we explore how streaming of display contents and mobile sensor data can enable designers to create interactive mobile and IoT prototypes.

**TECHNICAL REPORT (cite as)**

Ledo, D., Vermeulen, J., Carpendale, S., Greenberg, S., Oehlberg, L., and Boring, S. (2018). Astral: Prototyping Mobile and IoT Interactive Behaviours via Streaming and Input Remapping. Report 2018-1107-06, Department of Computer Science, University of Calgary, Calgary, AB, Canada, T2N 1N4. July.

## ASTRAL

We introduce Astral (Figure 1), a novel prototyping tool that uses streaming and input remapping to enable designers to author, test, and fine-tune interactive behaviour in mobile and IoT prototypes using familiar tools. We achieve this through: streaming of visual desktop content, streaming of mobile sensor data, and input remapping for interactivity.

### Streaming Visual Content to a Mobile Display

Astral consists of a desktop client and a mobile client application. Designers can use Astral to stream the contents of their desktop display onto a mobile device screen such as a phone or watch. As shown in Figure 1.1, the designer selects a region of a web browser running Flappy Bird to stream it to a connected mobile display. The mobile display is then updated live as the desktop display updates.

### Streaming Mobile Sensor Data to Provide Input

Next, the designer can select sensors on the mobile device that will provide input. Figure 1.2 shows how a designer can select multiple types of sensor data, such as touch, tilt, brightness or microphone data. In this case, the designer enables the touch sensor to be streamed back to the Astral desktop client. Every time the designer now taps the mobile display, the touch location is visualized on the Astral desktop client.

### Interactivity Through Input Remapping

Finally, as shown in Figure 1.3, the designer can remap the mobile sensor data to desktop keyboard or mouse inputs, and provide interactivity to the streamed display contents. The designer remaps touch events to spacebar keypresses, which control the bird in the web browser. Astral's input remapping allows the designer to stream visuals to the mobile display, which are affected by mobile sensor data, thus closing the loop and enabling interactivity.

More complex input remapping is possible, using different sensors, continuous and discrete actions, emulating states using sets of rules, or changing what is streamed to the mobile display, as we will demonstrate later.

### Benefits

Astral has several benefits over prior work:

1. ***Display contents and sensor data are streamed live***, allowing designers to test, iterate and refine interactive behaviour on the target device. The liveness aspect also enables designers to compare and explore different variations in input/output mappings.

2. ***Designers can use and repurpose familiar design tools*** through streaming and input remapping. One could, for example, combine Astral with video editing tools to create interactive mobile or IoT prototypes (see Scenario 4).

3. Astral enables designers to use the ***same approach to prototype both the interactive behaviour of mobile applications as well as IoT devices*** as enabled through mobile displays and sensors [28]. For instance, Astral could be used to prototype IoT devices such as a smart speaker (see Scenario 5).

## RELATED WORK

The goal of our prototyping tool is to author nuanced interactive behaviour on mobile devices. In this paper, we consider 'interactive behaviour' to be the 'feel' of a prototype [39] that cannot be easily conveyed through a physical sketch. The 'feel' emerges from not only the outputs once interactions happen, but also *as* interactions happen – a dynamic interplay discussed in past discussions of progressive feedback [57] and animation applications [16,17].

We situate our work among extensive prior work on toolkits and prototyping tools that help interaction designers define interactive behavior. The specific approach of our prototyping tool leverages past work on streaming sensor inputs and display outputs across devices, and remapping inputs across devices. Based on previous work, we identify a series of Design Goals (***DG***) for Astral.

### Prototyping Tools

In particular, we draw on prior work in prototyping tools where designers can quickly customize interactive behaviours [61] and work with sensor input [19], while leveraging existing tools [3] thus reducing the need to program.

Fast prototyping not only relies on expressiveness, but also the speed at which designers can preview and evaluate their designs. Many interface prototyping tools highlight live prototyping as a feature that helps designers create interactive applications in both mobile contexts [33, 35, 45] and physical computing contexts [19, 20, 28]. Gummy Live [33] allows designers to create mobile interfaces live and see them reflected in the mobile device ready for modification. Similarly, de Sá et. al [45] created a tool capable of transitioning from sketches on the target mobile device, to Wizard of Oz [26] prototypes, to higher fidelity ones. Our first goal is to create a tool that ***prototypes live interactive behavior on the target device (DG1).***

One approach to supporting physical computing authoring is through specialized hardware (e.g. [3, 14, 55, 63]). Arduino [62] in particular requires programmers to build circuits from scratch, which can be difficult for novices and introduces more opportunities for bugs [4]. Alternatively, designers can repurpose existing sensors in their authoring [28, 47, 49]. These approaches to prototyping allow designers to address each sensor individually; however, they also rely heavily on low-level programming. Our second goal is to ***provide an end-user interface that allows designers to explore variations among mobile sensors (DG2).***

When working on top of existing infrastructures, toolkits can leverage already existing functionality to quickly explore new types of interactions. Olsen [42] discusses how working with common infrastructures enables new technology combinations to support new solutions; for example, when pen input works as a mouse, mouse-based applications can now be used with a pen as the input device. Many prototyping tools in the research literature use this approach. Exemplar

[19] and MaKey MaKey [3] discuss the possibility of mapping mouse and keyboard events, and Gummy-Live [33] and D.Macs [34] leverage streaming. Other tools (e.g. InVision, Adobe XD) accommodate designers working in their platform by allowing them to import elements from other applications (e.g. mockups drawn in Photoshop). Our third goal is to further extend these principles and ***allow designers to work with existing, familiar prototyping applications (DG3)***.

### Streaming Inputs & Outputs Across Devices

We intend to address ***DG1*** (*prototype live interactive behavior on the target device*) by both streaming desktop display output to a mobile device, and streaming mobile sensor input back to a desktop machine. The basic principle of streaming a display to another device has previously been used to create interactive systems in HCI research. For instance, Sikuli [59] takes screenshots of UI elements so that they can be annotated, or to automate different tasks. Transmogrifiers [6] show how dynamic content of the desktop (e.g. images, websites, videos) can be transformed on-the-fly to create free-form visualizations. Prefab [12] treats the desktop as a set of pixels, which can be reverse-engineered and thus enables new behaviour implementations on existing interfaces (e.g. Bubble Cursor [15]).

In addition to streaming a captured screen to another device, we can also extend interaction across multiple devices (e.g., TeamViewer[1], VNC[2]). Semantic Snarfing [37] showed how a mobile device could act as a laser pointer to retrieve contents of a desktop. Myers [36] also examined how mobile devices could act as additional inputs to PCs (e.g. as a number pad). A screenshot can also hold information about the system's state – DeepShot [9] shows how to use a mobile device photo to migrate tasks across devices. The mobile device can also provide input to displays at a distance through its camera as done in Touch Projector [5], or extend the display space for further interaction opportunities as done in Virtual Projection [2]. We apply the idea of streaming in both directions: the contents of the desktop are sent to the mobile display, and the sensor data from the mobile device are sent to the desktop. Our last step is to remap mobile sensors into keyboard and mouse commands.

### Remapping Inputs Across Devices

For designers to prototype interactive behaviors, they must not only see continuous effects from their input (***DG1***), but also examine and modify *how* those effects take place. Thus, Astral needs to ***fine tune mappings between inputs and outputs (DG4)*** to prototype nuanced interactive behaviors. One way to achieve this is through *easings*. Easing was a term used by Adobe Flash [61] to refer to the slow-in and slow-out principle of animation [53], in which the number of in-between frames are increased or decreased at keyframes between poses to create the illusion that an object is speeding up or slowing down. Adobe Flash incorporated easings as a default linear tween that could be applied to motion tweens. Penner [43] created scripts to change the character of the easing through mathematical functions (see Figure 2). We extend Penner's [43] easing functions, which describe how an animation can play over time. We apply these functions to the continuous input values so that designers can explore how the output will perform as a function of the input action. The easing functions can serve for aesthetic experiences, as well as more utilitarian elements (e.g. balancing the sensitivity of an input's effect).

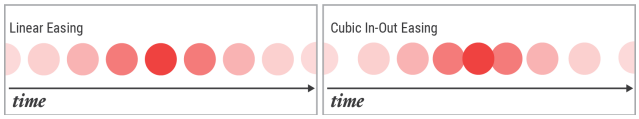


**Figure 2. Examples of Linear (left) and Cubic In-Out (right) easing functions [43].**

## ASTRAL'S CENTRAL COMPONENTS

The overarching idea of Astral (as illustrated in Figure 1) is to allow designers to quickly prototype interactive behaviours on mobile devices. To ensure that designers can use or repurpose familiar desktop applications to author and test mobile interactive behaviours (**DG3**), we designed Astral as a desktop client that communicates with a mobile client, for which designers want to author interactive behaviours. In this section, we provide an overview of Astral's individual components.

### Main View

When starting Astral, a simplified view is shown (Figure 3.1). This view provides basic functionality once a device is connected. Upon connection (by starting Astral on the mobile device), a designer can begin authoring the intended interactive behavior through the use of rules and rulesets (see following sections). At the same time, the designer can choose the region of the desktop (where they designed the intended user interface through means of their preferred design application) that should be streamed to the mobile client.

To select that region, the designer clicks the camera icon, and an overlay appears, showing exactly which part of the desktop will be streamed. This overlay can be scaled and translated to select the region of interest. While this selection window is shown, the contents are already streamed to the mobile client, so that a designer sees the changes and can assess when the selection is sufficient for the task.

The main view is intentionally kept narrow, so that it occludes the least amount of screen real estate. It further provides access to the rule editing window, where designers can map sensor data from a mobile device to specific events on

[1] https://www.teamviewer.com/

[2] https://tools.ietf.org/html/rfc6143

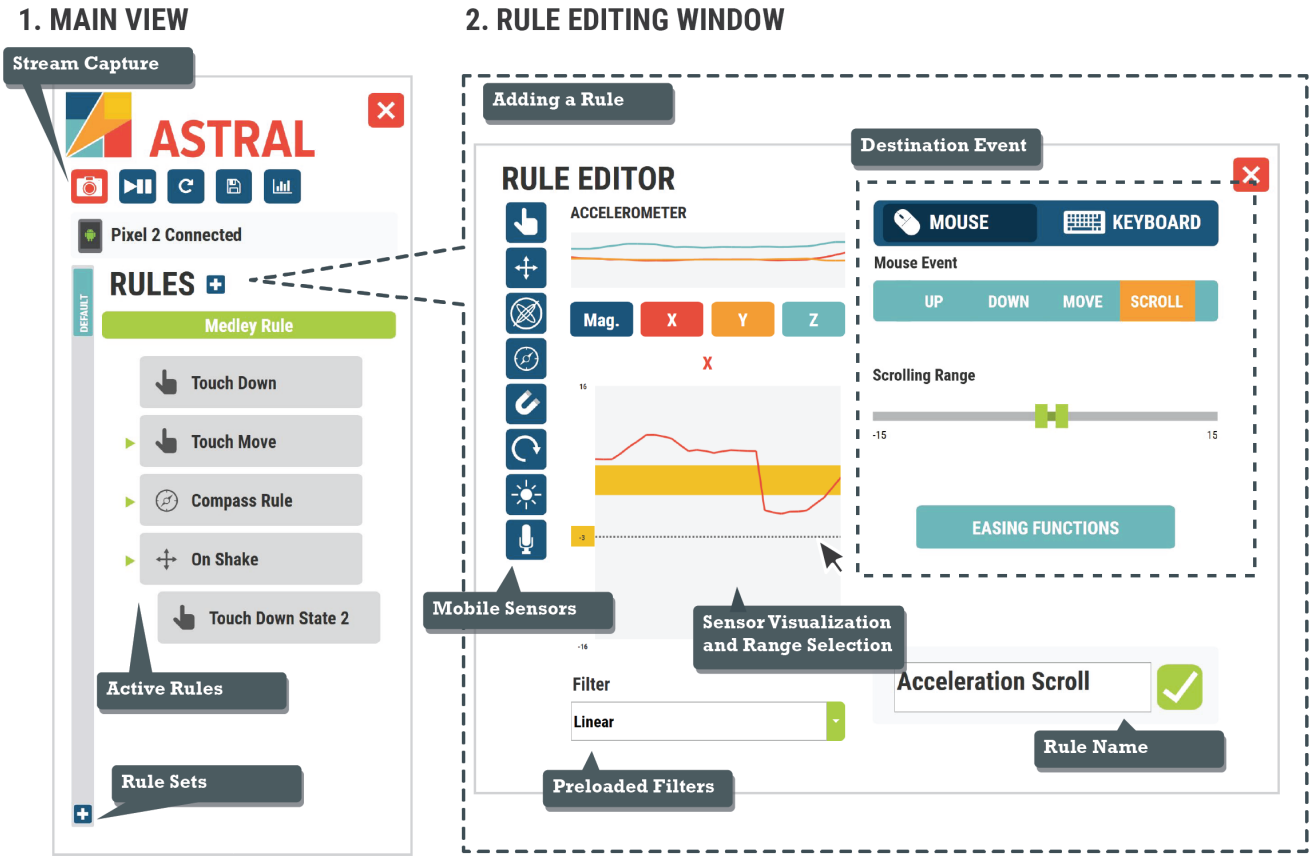


**Figure 3. Annotated Astral Interface. (1) The main window streams content to the mobile device and displays active rules. (2) When adding rules, the interface shows an interactive visualization where designers select the range of sensor values to for input remapping.**

the desktop (i.e., mouse or keyboard events). Other options include playing and pausing of active rules, saving the current rule sets, and displaying a visualization dashboard of the streamed mobile sensor data.

### Specifying Input Remapping through Rules

Once content is streamed to the mobile device, designers can author an interactive behaviour by defining a *rule*. A rule is a software abstraction that contains information as to how mobile sensor data is mapped to keyboard and mouse events. To do so, the designer first clicks on the 'plus' sign to open the *Rule Editing Window* – a guided interface to author or edit an input remapping rule. The rule can now be defined (see a particular configuration example in Figure 3.2). First, the sensor (e.g., the accelerometer) of interest must be specified, which brings up an individualized interface for each particular sensor (examples in Figure 4), often as a live visualization of that sensor's values. Based on the chosen sensor, the next step is to select which parameter to observe (e.g., the *x*-value of the accelerometer). The third step is then to constrain the sensor to a range of values (e.g., between -5 $m/s^2$ and 5 $m/s^2$). Note that sensor readings can be further transformed, such as by applying prepackaged filters. For instance, we can filter acceleration values to extract gravity and linear acceleration values.

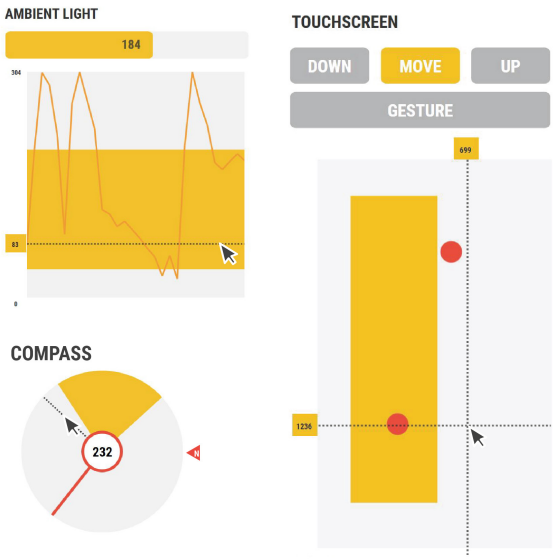


**Figure 4. Astral provides interactive visualizations for different sensors (clockwise from top-left: ambient light, touchscreen and compass).**

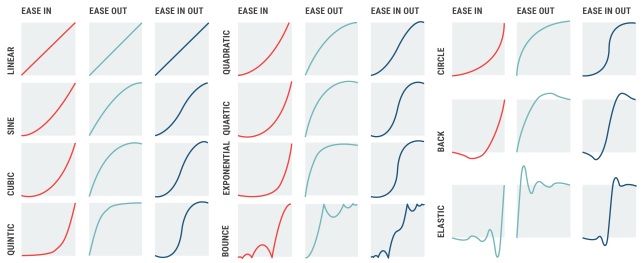


**Figure 5. Available easing functions in Astral.**

Now that the *sensor input* from the mobile device is specified, the designer should be able to map that input to the *desktop input* (e.g., a mouse move event). Mouse events can be constrained on the desktop, for example, only allowing mouse movements horizontally between 800 and 1000 pixels, or defining ranges for mouse wheel events (the default in Microsoft Windows being 120 pixels per step). For keyboard events, designers can specify a key event (i.e., down, press, or up) and the key that should perform that event (e.g., arrow left). Keys are either typed by the designer or selected from a list of operating system defined keys (e.g., volume controls, media playback, or print screen). Key selection allows for integration with existing applications through the use of shortcuts offered by that application.

The selected desktop input can be *discrete* or *continuous* (as in Exemplar's categorization of sensor values [19]). Discrete input is triggered when the sensor enters or exits the range of values, while a continuous input interpolates the value from the range to the destination mapping. These interpolations can be altered through easing functions (Figure 5). Note that the system automatically determines the nature of input: *discrete rules* being mouse up and down, key up, down or pressed; and *continuous rules* being mouse move, or scroll.

For continuous input, designers can apply easing functions. We used Penner's set of easing functions [43] shown in Figure 5. To achieve this, a rule defines a range as a source selection (e.g. accelerometer low and high values) and a destination selection (e.g. mouse coordinates). The current value and the source selection are transformed and normalized into a unit rectangle (1 by 1 size), which then is interpolated into the selected easing function. The values are then remapped to the new destination selection, where the outputted value now has the easing applied.

The authoring process is dynamic: designers can immediately test and modify a rule as they author or edit it. If they do not wish for the rule to continue running (e.g., mobile device input taking over the mouse cursor), they can press the 'escape' key to play or pause the live mapping. When the designer is finished, they can name the rule, and it will be added to the active ruleset in the main application window.

### Merging Several Rules into Rulesets

Often, one interactive behaviour may require several rules (possibly relying on different sensors). Astral adds one layer of abstraction, called *rulesets*, which allows the combination of several rules. If a ruleset is active, all rules within that set will execute as long as the mobile device is streaming. This can be paused both on the interface by clicking on the play/pause button, or by pressing the 'escape' key.

To allow for testing variations of interactive behaviours, designers can create multiple rulesets and switch between them at any time by clicking on the tab (Figure 3.1 left and bottom). When there is an active ruleset, a newly created rule will be added to that set and stacked vertically.

There are two special conditions for discrete rules. A *medley rule*, which is a discrete rule that serves to switch between active rulesets, allows designers to test different variations of prototypes. Discrete rules can also have *children*, which allows individual transitions between rules (thus simulating multiple states). This means that a rule with a child becomes inactive once it is executed, then enabling its child to become active. A '+' button appears on hover when a child rule can be created. Rule hierarchy is shown through horizontal alignment.

### How Astral Satisfies its Design Goals

The central components of Astral strongly followed the previously identified design goals. It satisfies **DG1** with its ability of streaming sensor data from the mobile device and portions of the desktop display to the mobile device. Additionally, **DG1** is supported by running rules in real-time and dynamically switching them on and off.

The ability to configure interactive behaviour by creating and manipulating different rules without the need for programming allows Astral to satisfy **DG2**. In particular, the included visualizations provide a higher expressive match [42] to select the right values and properties for interactive testing.

Astral works with familiar desktop tools, such as PowerPoint or HTML to prototype interactive behaviour on mobile devices, thus satisfying **DG3**. Designers can remain in their natural design environment. Lastly, **DG4** is satisfied in that Astral allows for fine-tuning input mappings through the use of different easing functions.

## IMPLEMENTATION

The desktop client of Astral is implemented using C# and WPF, whilst the mobile applications are written in Xamarin to allow for cross-platform mobile development (iOS, Android, AndroidWear). To allow for reusable code and quickly adapting to newly added sensors of potential future devices, we developed all communication aspects in shared code, which uses the .NET Standard 2.0. Network connectivity is achieved through wireless LAN using TCP. We tested Astral on multiple phones (Nexus 5 and 5X, iPhone 7 and 8, Pixel 2) and on the Sony Smartwatch 3.

We are able to achieve relatively fast performance when streaming display content – 50 fps on iOS, 25 fps on Android – despite the mobile sensor data also being streamed back to the desktop (microphone, accelerometer, etc.). During testing and creation of the prototypes below, we did not experience any significant delay in the transfer of sensor data.

## INTERACTION SCENARIOS AND PROTOTYPES

In this section, we describe a series of interaction scenarios that demonstrate the expressive range and power of Astral. Each scenario describes an Astral prototype, implemented by the authors, followed by a discussion of concepts that are demonstrated by that particular prototype.

### Scenario 1: Exploring Map Interactions

This scenario depicts a designer, Alex, exploring possible one-handed physical interactions with a mobile map.

#### *A. Tilt to Move*

*Alex wishes to explore how to navigate a mobile interactive map with one-handed interaction. They open an instance of Google Earth in a web browser on their desktop and zoom into a location. Using Astral, Alex creates four rules* ***mapping keyboard commands*** *to the phone's accelerometer readings – tilting the mobile device to the right triggers a right arrow key, tilting to the left triggers the left arrow key, and similarly for up and down. The rules are set so that key commands are triggered when the acceleration passes a certain* ***range*** *(x: 4 to 7 triggers right, x: -4 to -7 triggers left, y: 4 to 7 triggers down, y: -4 to -7 triggers up). Because they chose a keypress event, when the accelerometer enters the specified range, the key is pressed down, and when leaving the range, it is lifted up. These values are determined live and by exploration. Alex can select the Google Maps window to test individual rules and fine-tune the sensitivity.*

This scenario replicates an example from d.tools [20] that originally required programming for continuous navigation; the Astral version leverages an existing web-based map (in this case Google Earth), and requires no programming at all. The designer can choose the device sensors and the corresponding keyboard interactions, and the sensor input can be fine-tuned to trigger keyboard interactions based on a specified threshold.

#### *B. Tilt-to-Zoom*

*Alex wants to modify their mobile map interaction to use tilt-to-zoom [21] (Figure 6). They create a new set of rules. First, Alex creates a "clutch" rule that starts the interaction – a touch down event that triggers the tilt-to-zoom operation and acts as a* ***parent rule****. They then create a rule mapping the y-axis of the accelerometer to a (mouse) scroll event. Alex selects an accelerometer range from -5 to 5, and scrolling range is set from -15 to 15 pixels. Astral linearly interpolates between both ranges by default. In testing the interaction, Alex is unsure if it is too sensitive, so they try an* ***easing function*** *(quadratic ease in/out), which makes the sensors less sensitive when the device is close to horizontal. Finally, Alex creates a rule triggered by a touch up event to end the tilt-to-zoom operation.*

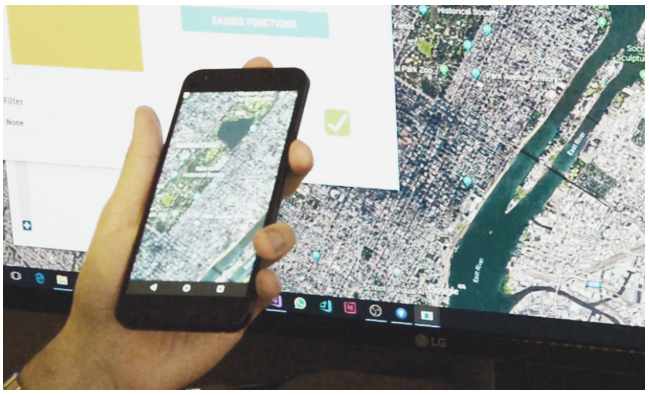

**Figure 6. Authoring the tilt-to-zoom prototype (Scenario 1B).**

This scenario replicates an example from Hinckley et al. [21], incorporating the concept of *motion in touch*, mapping more than one sensor to a single function. Touch is used as a means to explicitly switch to one-handed zooming using tilt. It also shows the value and role of easing functions, which allow designers to specify how the mapping is carried out based on continuous input. Finally, it showcases how Astral can map continuous mobile sensor actions to continuous desktop actions (e.g., scrolling) within a bounded number of pixels. Having a parent rule means that the child rule will not execute until the parent rule has executed, thus enabling different states.

### Scenario 2: Input Variations in a Mobile Game

*Alex, our designer, wishes to test different input variations for a mobile game prototype (Figure 7). They open a web browser with the game 'Flappy Bird', which uses the spacebar or mouse click to make a bird flap its wings to fly between pipes. Alex streams the browser content onto a mobile phone and creates different rulesets that will trigger the spacebar on a threshold. The first one is a simple tap on the touch display, similar to the original mobile game. The next version uses a shake gesture on the accelerometer (when the magnitude is greater than 8). Alex tests the shake interaction and finds the magnitude threshold is too high and reduces it to 6.*

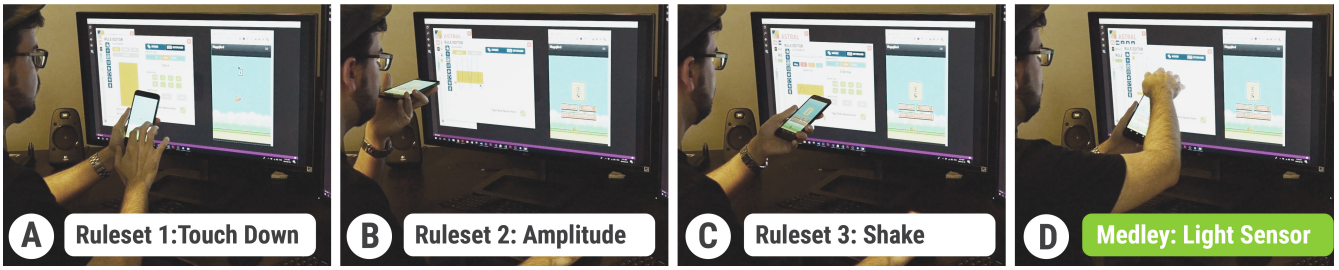


**Figure 7. Authoring different input variations for the game "Flappy Bird" where a bird flaps its wing when hitting the space key (Scenario 2). The following rules trigger the space key: (a) tapping, (b) blowing on the microphone, (c) shaking the device. When the (d) ambient light sensor detects a reading of 0, the rulesets are switched for variation testing.**

*Next, they create a rule set that uses the microphone's amplitude, where the player can blow into the microphone to trigger the spacebar. Finally, Alex creates a rule* ***to switch between rule sets****. The rule triggers when the ambient light sensors detect a reading of 0. When testing the game, the newly added rule allows Alex to immediately switch between the different input versions by covering the display.*

Designers can sequentially test a set of prototype alternatives to explore different solutions to the problem. Astral can use this approach to experiment with any variations of rules, including sensors, thresholds, easings, or desktop inputs. The designer can switch between rulesets by creating a medley rule (in our scenario, based on the ambient light sensor).

**Scenario 3: Iterative Design of a Media Controller**

This scenario describes iterative prototyping for a watch-based media controller that can play, pause, or change the volume of music on a computer (Figure 8). Each iteration uses a different strategy and shows how Astral supports moving from mockups to more refined high-fidelity prototypes.

*A. Interactive Paper Mockups*

*Alex draws a series of paper mockups for a watch media controller, takes photos of the mockups and opens them on their computer in a single image viewer window. Alex creates multiple states by creating child rules, specifying the transitions between the different states. For each state, Alex sets a new image capture region on the display around the corresponding mockup in an image viewer. Alex confirms that the volume buttons in their sketch are large enough to interact with and maps the touch location on the watch to a keyboard event (volume up and down keys).*

The designer can use Astral to create a first interactive prototype based on photographs of sketches. This prototype allows the designer to preview the interfaces on the target device and thus make early decisions (e.g. determining appropriate button sizes). In creating transitions, the designer can also preview the interaction flow, simulating state transitions as seen in other prototyping tools (e.g. d.tools [20]).

*B. Exploring States in PowerPoint & Adobe Illustrator*

*Alex wants to explore state transitions in PowerPoint using visuals from a higher fidelity non-interactive prototype created in Adobe Illustrator. They create one slide for each state; touch events on the device view correspond to click events on different regions of the desktop display, e.g. the slide thumbnails in Powerpoint. After testing the interaction, Alex considers it best to have a single screen given the small size of the smart watch.*

This scenario again demonstrates designers' abilities to work with existing tools, in this case PowerPoint and Illustrator, both of which are discussed as current standards [31, 60].

*C. Creating an Interactive Image*

*Alex takes one of the Adobe Illustrator designs and maps different locations of the image to playback controls – the previous button maps to the 'previous song' key, the next button maps to the 'next song' key, and the play button is mapped to the 'play/pause' key. Alex opens iTunes on their computer and tests the functional prototype on their watch.*

Because the designer assigns mappings, they can use multiple strategies to author a prototype. Here, Astral is used to create an interactive image, an alternative approach to creating user interface façades [50]. While the designer leveraged special keys provided by the operating system, they could also trigger other key combinations as well as hotkeys using modifier keys (e.g. control, shift).

*D. Creating a Smartwatch Prototype in Expression Blend*

*Alex wants to test the visual feedback of the buttons, and thus programs a prototype using Microsoft Expression Blend with buttons that change color when pressed. Alex streams the running Expression Blend program to the watch and maps touch events on the watch to clicks on the corresponding buttons in the program so that the buttons provide visual feedback when touched on the smartwatch. Instead of linking the Expression Blend prototype with C# code to program the media controls, Alex uses Astral to map those regions to keyboard commands for Play/Pause, Next and Previous as in the interactive image.*

The sequence of paper mockups, Powerpoint, Illustrator, and Expression Blend shows how Astral can support different fidelity prototypes as the designer iterates and refines their

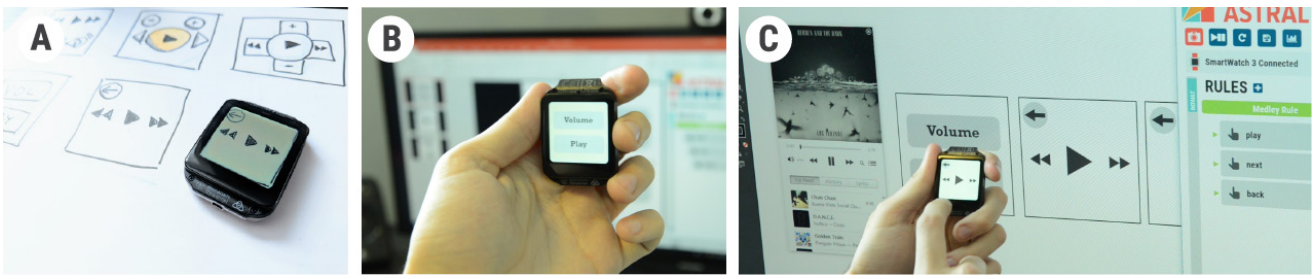


**Figure 8. Astral supports the design process in all stages by allowing (a) on-device rapid creation of interactive sketches (Scenario 3A), (b) using slideshows to transition between states (Scenario 3B), or (c) creating interactive images (Scenario 3C).**

ideas during the design process. Astral supports designers' work in familiar, powerful desktop software tools. Designers can use familiar GUI programming tools (e.g., web prototypes, Expression Blend) on mobile devices that do not natively support these applications.

**Scenario 4: Video-Based Prototyping**

This scenario demonstrates how designers can leverage video editing tools, such as Adobe AfterEffects (Figure 9). This strategy maps continuous input to different points in the video timeline to 'scrub' based on the current sensor data.

*A. Compass Interface*

*Alex wants to test the look and feel of a compass interface. They create a compass mockup in Adobe Illustrator and import the asset into AfterEffects. Alex creates a basic transformation in AfterEffects to rotate two Illustrator layers (the compass' needle and its shadow) within a 3-second window. Alex then opens Astral, connects the mobile phone, and maps the angle of the compass sensor to the position on the video timeline through a mouse move event. Pointing the phone in different directions now updates the compass interface.*

Maudet et. al discuss how designers often use high-fidelity videos to convey prototype ideas to developers [31]. While video can show state-based animations, it does not show the effect of the interaction as continuous inputs are taking place. This example shows how Astral enables the designer to quickly create a working compass prototype, thus going beyond traditional video prototyping. Moreover, small effects such as the changing shadow would be relatively complex to achieve through programming, while it requires little effort in a video editing application for a skilled designer.

*B. Phone Control Panel*

*Alex creates a video prototype of a phone control panel in Astral, which, on swiping down, progressively reveals different options. Alex maps a mobile touch move to the mouse to scrub through the first portion the video in a video editing tool (e.g. Adobe Premiere). Next, Alex tests several easing functions, including a playful bounce effect animation. Once dragged down, the panel reveals a brightness slider control at the top. Alex creates another portion of the video that demonstrates what happens when the brightness is decreased, mapping a horizontal touch move event to scrub through that part of the video, thus creating the effect that the slider is being dragged.*

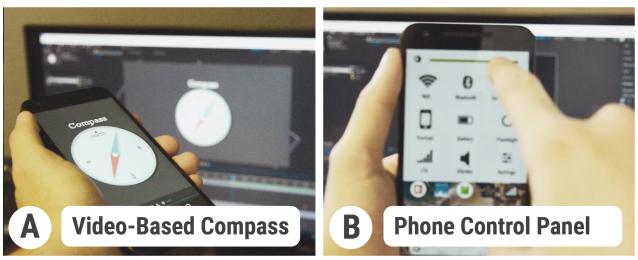


**Figure 9. Interactive prototypes created by Astral by mapping sensor data to mouse move events that scrub through video: (a) a compass application (Scenario 4A) and (b) a phone control panel (Scenario 4B).**

**Figure 10. "Smart Home Speaker" prototype (Scenario 5) consisting of a travel mug, a smart watch, a 3D-printed base to hold the watch and a translucent 3D printed lid that diffuses light.**

This example demonstrates the ability to create complex animations using familiar video editing tools, which may be far easier to achieve than using programming. Finally, this scenario demonstrates how designers can simulate multiple states by mapping different continuous input to different parts of a video timeline.

**Scenario 5: IoT Prototyping**

Astral can also extend beyond mobile prototyping into IoT applications (Figure 10). The following scenarios demonstrate how Astral supports authoring physical IoT prototypes (through Soul-Body prototyping [28]).

*A. Smart Speaker*

*Alex is prototyping applications for a "Smart Home Speaker". They create a Soul-Body prototype [28] by repurposing a travel mug and 3D printing a translucent lid that encloses a smartwatch (see Figure 10). Alex selects the microphone control in Astral and chooses the 'speech recognition' option, which they map to different videos to simulate a conversation with the smart speaker, showing nuanced animations, from loading, listening, to responding. Audio is output through the desktop speakers.*

This scenario demonstrates Astral's ability to leverage the displays and sensors of smartwatches and mobile phones to extend into IoT applications, using Soul-Body Prototyping [28]. Within the physical prototype, the designer can test different visual animations and responses.

### Scenario 6: Beyond the Desktop

*Finn, a graphic designer, is working on a layout which requires them to constantly work with the object alignment functions in Adobe Illustrator – and there are no keyboard shortcuts! Using Astral, they stream the Alignment toolbar onto a phone, rerouting touches to mouse clicks that select the different alignment tools. As they work, Finn can quickly access alignment tools on the left side of their desk. Finn can keep working with the mouse using their right hand and use the left to quickly trigger the different alignment functions.*

This scenario moves interface elements to another screen for customizable access as envisioned by Myers [36]; thus, it partially replicates Interface Facades [50], taken to an external, peripheral device [51]. The alignment functions in Illustrator do not have a default hotkey and often require designers to move their mouse back and forth between the object of interest and the control. Astral by default will move the mouse cursor back to the starting location when executing mouse down, up and click events. The mobile device moves the toolbar to the periphery and makes it readily accessible with the non-dominant hand, providing an opportunity for desktop-based bimanual interaction.

## DISCUSSION

In this section, we discuss our primary evaluation approach – demonstration [29] – as well as specific considerations when designing a prototyping tool such as Astral, including the use of states, scale, and input locks. We relate these to our design rationale and heuristics for systems research [38, 42].

### Demonstration by Example

We realized all twelve of the example prototypes discussed in our scenarios using Astral. These prototypes represent a combination of both novel and replicated past research systems [20, 21, 28, 36, 50, 51]. Our scenarios provide a perspective on how designers might work with Astral and show elements of threshold and ceiling [38]. Finally, we benchmarked the performance of the image transfer, which reached up to 20 fps on Android, and a consistent 45 to 50 fps on iOS.

### The Use of 'States'

States are a common approach followed in prototyping tools, as they can be used to quite intuitively describe the flow of the interaction. In Astral, states can be used to manage multiple sensors (one per sensor). Astral supports states in two main ways. First, different states can be configured by using parent rules, as shown in Scenarios 1B, and Scenarios 3A–3D. Second, Astral supports different states by scrubbing through different parts of a video, as demonstrated in Scenario 4D. Rather than supporting more complex states through a hierarchical state model, we focused on facilitating the fine-tuning of the mapping and interplay between the mobile sensor inputs and outputs (**DG2** and **DG4**). While a pure state model approach could enable more complex applications, it would complicate prototyping the 'feel' aspects of interactive behaviour, as state models tend to favour a trigger-action model.

### Scale

As demonstrated in our scenarios, Astral allows designers to quickly get started (*threshold*) and achieve fairly expressive results (*ceiling*). However, we have only examined a small subset of the range of interaction possibilities with these types of inputs and this type of tool. Since the input remapping is constrained by what is supported by existing applications, the level of complexity that can be supported by Astral prototypes is bounded by the capabilities of those applications. Yet, Astral's ceiling can be further increased by relying on scripts or custom coded applications that respond to desktop mouse and/or keyboard events.

### Overcoming 'Input Locks'

In some cases, we observed issues with input locks when using Astral. Once a mouse move event is selected, the mouse would start reacting to the incoming sensor data. It then became impossible to move the cursor with the physical mouse. To preserve the ability to test behaviours live as they are authored, we remedied this by adding a toggle with the 'escape' key to enable or disable the live preview.

## LIMITATIONS

We next list some technical limitations of the current implementation of Astral.

### Single Device

At the moment, Astral supports one mobile device per desktop system, which constrains and simplifies the workflow. This is also tied to a technical limitation of desktops, as mouse and keyboard commands only can be sent to a single focused program, meaning that a side by side comparison as done in Scenario 1-B would not be possible on the desktop.

### Device Relativism

Mappings of mouse and screen coordinates may not carry across different computers with different resolutions. One way to address this is to use device coordinates. Another potential concern is that window sizes are not fixed, so once the workspace has changed the mappings may no longer work. There are some workarounds to this latter concern: for instance, it is possible to store the position and sizes of the windows and associate them to the rules, so that when a ruleset executes it adjusts the window sizes. Finally, Astral currently does not support full-screen applications. Mobile phones and smartwatches also have a wide variation within their resolutions and sensors. Additionally, some sensors may not be available on each device, and some sensors may have device-specific readings.

## FUTURE WORK

Astral provides a starting point for richer potential applications for mobile prototyping and Internet of Things. We next discuss some potential avenues for future work.

### Increasing Complexity

One way to extend Astral would be to examine how to support more complex dimensions than the current mouse and

keyboard events. Additionally, it would be interesting to examine how to integrate Astral's authoring of behaviours onto the more traditional state model to afford more complex applications.

### Multimodal Output

Astral currently streams the desktop display contents only. It would be interesting to explore other possible mobile outputs. For instance, sounds (if used) are currently limited to be played on the desktop, but Astral could be extended to also stream these sounds to the mobile device.

### Evaluating Astral in Interaction Design Practice

We are interested in understanding Astral's utility for interaction designers. An observational study would help us understand *how* the workflow fits interaction designers. Ideally, this would involve an in-situ evaluation over a period of time, as designers all have individual setups (i.e. preferred applications, workflows, customized interfaces). A field study would unveil how Astral could be integrated into their existing practices, and common usage strategies.

## CONCLUSION

This paper presented Astral, a prototyping tool that allows designers to author and fine-tune interactive behaviours from mobile sensor data. By streaming selected desktop display contents onto the mobile device, and by converting the mobile sensor data into mouse and keyboard events, we empower designers to repurpose existing applications in new and interesting ways. We also allow designers to transform input through easing functions, so that they can fine-tune how the output changes as the input takes place. Our design decisions are informed by current literature, and we demonstrated the expressive power of Astral through a broad range of usage scenarios. Our scenarios included: exploration, fine-tuning and comparison; prototyping alternatives; supporting different stages in the design process; repurposing the video timeline; IoT prototyping; and beyond the desktop. We hope our exploration can propel designers' conversations around interactive behaviour and mitigate some of the challenges of transitioning from design to implementation.

## REFERENCES


1. Dominikus Baur, Sebastian Boring, and Steven Feiner. 2012. Virtual projection: exploring optical projection as a metaphor for multi-device interaction. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12). ACM, New York, NY, USA, 1693-1702. DOI: http://dx.doi.org/10.1145/2207676.2208297
2. Beginner's Mind Collective and David Shaw. 2012. Makey Makey: improvising tangible and nature-based user interfaces. In Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction (TEI '12), Stephen N. Spencer (Ed.). ACM, New York, NY, USA, 367-370. DOI: https://doi.org/10.1145/2148131.2148219
3. Tracey Booth, Simone Stumpf, Jon Bird, and Sara Jones. 2016. Crossed Wires: Investigating the Problems of End-User Developers in a Physical Computing Task. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16). ACM, New York, NY, USA, 3485-3497. DOI: https://doi.org/10.1145/2858036.2858533
4. Sebastian Boring, Dominikus Baur, Andreas Butz, Sean Gustafson, and Patrick Baudisch. 2010. Touch projector: mobile interaction through video. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10). ACM, New York, NY, USA, 2287-2296. DOI: https://doi.org/10.1145/1753326.1753671
5. John Brosz, Miguel A. Nacenta, Richard Pusch, Sheelagh Carpendale, and Christophe Hurter. 2013. Transmogrification: causal manipulation of visualizations. In Proceedings of the 26th annual ACM symposium on User interface software and technology (UIST '13). ACM, New York, NY, USA, 97-106. DOI: https://doi.org/10.1145/2501988.2502046
6. Tsung-Hsiang Chang and Yang Li. 2011. Deep shot: a framework for migrating tasks across devices using mobile phone cameras. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11). ACM, New York, NY, USA, 2163-2172. DOI: https://doi.org/10.1145/1978942.1979257
7. Morgan Dixon, Alexander Nied, and James Fogarty. 2014. Prefab layers and prefab annotations: extensible pixel-based interpretation of graphical interfaces. In *Proceedings of the 27th annual ACM symposium on User interface software and technology* (UIST '14). ACM, New York, NY, USA, 221-230. https://doi.org/10.1145/2642918.2647412
8. Saul Greenberg. 2007. Toolkits and interface creativity. *Multimedia Tools and Applications*, *32*(2), Springer, 139-159. https://doi.org/10.1007/s11042-006-0062-y
9. Saul Greenberg and Chester Fitchett. 2001. Phidgets: easy development of physical interfaces through physical widgets. In Proceedings of the 14th annual ACM symposium on User interface software and technology (UIST '01). ACM, New York, NY, USA, 209-218. DOI=http://dx.doi.org/10.1145/502348.502388
10. Tovi Grossman and Ravin Balakrishnan. 2005. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '05). ACM, New York, NY, USA, 281-290. DOI=http://dx.doi.org/10.1145/1054972.1055012
11. Rubaiat Habib Kazi, Fanny Chevalier, Tovi Grossman, and George Fitzmaurice. 2014. Kitty: sketching dynamic and interactive illustrations. In Proceedings of the 27th annual ACM symposium on User interface

software and technology (UIST '14). ACM, New York, NY, USA, 395-405. DOI: https://doi.org/10.1145/2642918.2647375

12. Rubaiat Habib Kazi, Tovi Grossman, Nobuyuki Umetani, and George Fitzmaurice. 2016. SKUID: sketching dynamic drawings using the principles of 2D animation. In ACM SIGGRAPH 2016 Talks (SIGGRAPH '16). ACM, New York, NY, USA, Article 84, 1 pages. DOI: https://doi.org/10.1145/2897839.2927410
13. Björn Hartmann, Leith Abdulla, Manas Mittal, and Scott R. Klemmer. 2007. Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07). ACM, New York, NY, USA, 145-154. DOI: https://doi.org/10.1145/1240624.1240646
14. Björn Hartmann, Scott R. Klemmer, Michael Bernstein, Leith Abdulla, Brandon Burr, Avi Robinson-Mosher, and Jennifer Gee. 2006. Reflective physical prototyping through integrated design, test, and analysis. In Proceedings of the 19th annual ACM symposium on User interface software and technology (UIST '06). ACM, New York, NY, USA, 299-308. https://doi.org/10.1145/1166253.1166300
15. Ken Hinckley and Hyunyoung Song. 2011. Sensor synaesthesia: touch in motion, and motion in touch. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11). ACM, New York, NY, USA, 801-810. DOI: https://doi.org/10.1145/1978942.1979059
16. Lars Erik Holmquist. 2005. Prototyping: generating ideas or cargo cult designs?. interactions 12, 2 (March 2005), 48-54. DOI=http://dx.doi.org/10.1145/1052438.1052465
17. David Ledo, Fraser Anderson, Ryan Schmidt, Lora Oehlberg, Saul Greenberg, and Tovi Grossman. 2017. Pineal: Bringing Passive Objects to Life with Embedded Mobile Devices. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (CHI '17). ACM, New York, NY, USA, 2583-2593. https://doi.org/10.1145/3025453.3025652
18. David Ledo, Steven Houben, Jo Vermeulen, Nicolai Marquardt, Lora Oehlberg and Saul Greenberg. 2018. Evaluation Strategies for HCI Toolkit Research. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (CHI '18). ACM, New York, NY, USA (To Appear). https://doi.org/10.1145/3173574.3173610
19. Nolwenn Maudet, Germán Leiva, Michel Beaudouin-Lafon, and Wendy Mackay. 2017. Design Breakdowns: Designer-Developer Gaps in Representing and Interpreting Interactive Systems. In Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW '17). ACM, New York, NY, USA, 630-641. DOI: https://doi.org/10.1145/2998181.2998190
20. Jan Meskens, Kris Luyten, and Karin Coninx. 2009. Shortening user interface design iterations through realtime visualisation of design actions on the target device. In *Visual Languages and Human-Centric Computing, 2009. VL/HCC 2009. IEEE Symposium on*, pp. 132-135. 10.1109/VLHCC.2009.5295281
21. Jan Meskens, Kris Luyten, and Karin Coninx. 2010. D-Macs: building multi-device user interfaces by demonstrating, sharing and replaying design actions. In Proceedings of the 23nd annual ACM symposium on User interface software and technology (UIST '10). ACM, New York, NY, USA, 129-138. DOI: https://doi.org/10.1145/1866029.1866051
22. Jan Meskens, Jo Vermeulen, Kris Luyten, and Karin Coninx. 2008. Gummy for multi-platform user interface designs: shape me, multiply me, fix me, use me. In Proceedings of the working conference on Advanced visual interfaces (AVI '08). ACM, New York, NY, USA, 233-240. DOI: https://doi.org/10.1145/1385569.1385607
23. Brad Myers. Mobile devices for control. 2002. In *International Conference on Mobile Human-Computer Interaction*, Springer, Berlin, Heidelberg 1-8. https://doi.org/10.1007/3-540-45756-9_1
24. Brad Myers, Choon Hong Peck, Jeffrey Nichols, Dave Kong, and Robert Miller. 2001. Interacting at a distance using semantic snarfing. In *International Conference on Ubiquitous Computing*. Springer, Berlin, Heidelberg pp. 305-314. https://doi.org/10.1007/3-540-45427-6_26
25. Brad Myers, Scott E. Hudson, and Randy Pausch. 2000. Past, present, and future of user interface software tools. *ACM Trans. Comput.-Hum. Interact.* 7, 1 (March 2000), 3-28. http://dx.doi.org/10.1145/344949.344959
26. Brad Myers, Sun Young Park, Yoko Nakano, Greg Mueller, and Andrew Ko. How designers design and program interactive behaviors. 2008. In Proc. *Visual Languages and Human-Centric Computing, 2008. VL/HCC 2008. IEEE Symposium on*, pp. 177-184. 10.1109/VLHCC.2008.4639081
27. Dan R. Olsen, Jr.. 2007. Evaluating user interface systems research. In *Proceedings of the 20th annual ACM symposium on User interface software and technology* (UIST '07). ACM, New York, NY, USA, 251-258. https://doi.org/10.1145/1294211.1294256
28. Robert Penner. *Robert Penner's Programming Macromedia Flash MX*. McGraw-Hill, Inc., 2002.
29. Marco de Sá, Luís Carriço, Luís Duarte, and Tiago Reis. 2008. A mixed-fidelity prototyping tool for mobile devices. In Proceedings of the working conference on Advanced visual interfaces (AVI '08). ACM, New

York, NY, USA, 225-232. DOI: https://doi.org/10.1145/1385569.1385606

30. Valkyrie Savage, Colin Chang, and Björn Hartmann. 2013. Sauron: embedded single-camera sensing of printed physical user interfaces. In *Proceedings of the 26th annual ACM symposium on User interface software and technology* (UIST '13). ACM, New York, NY, USA, 447-456. http://dx.doi.org/10.1145/2501988.2501992
31. Valkyrie Savage, Xiaohan Zhang, and Björn Hartmann. 2012. Midas: fabricating custom capacitive touch sensors to prototype interactive objects. In *Proceedings of the 25th annual ACM symposium on User interface software and technology* (UIST '12). ACM, New York, NY, USA, 579-588. https://doi.org/10.1145/2380116.2380189
32. Wolfgang Stuerzlinger, Olivier Chapuis, Dusty Phillips, and Nicolas Roussel. 2006. User interface façades: towards fully adaptable user interfaces. In Proceedings of the 19th annual ACM symposium on User interface software and technology (UIST '06). ACM, New York, NY, USA, 309-318. DOI: https://doi.org/10.1145/1166253.1166301
33. Desney S. Tan, Brian Meyers, and Mary Czerwinski. 2004. WinCuts: manipulating arbitrary window regions for more effective use of screen space. In CHI '04 Extended Abstracts on Human Factors in Computing Systems (CHI EA '04). ACM, New York, NY, USA, 1525-1528. DOI: https://doi.org/10.1145/985921.986106
34. Frank Thomas, Ollie Johnston, and Disney Animation. 1981. *The illusion of life*. Abbeville Press, New York.
35. (Eds.). ACM, New York, NY, USA, 170-171. DOI=http://dx.doi.org/10.1145/223355.223485
36. Nicolas Villar, James Scott, Steve Hodges, Kerry Hammil, and Colin Miller. (2012) .NET Gadgeteer: A Platform for Custom Devices. *In Pervasive Computing. Pervasive 2012. Lecture Notes in Computer Science*, vol 7319. Springer, Berlin, Heidelberg. 216-233 https://doi.org/10.1007/978-3-642-31205-2_14
37. Jo Vermeulen, Kris Luyten, Karin Coninx, and Nicolai Marquardt. 2014. The design of slow-motion feedback. In Proceedings of the 2014 conference on Designing interactive systems (DIS '14). ACM, New York, NY, USA, 267-270. DOI: https://doi.org/10.1145/2598510.2598604
38. Tom Yeh, Tsung-Hsiang Chang, and Robert C. Miller. 2009. Sikuli: using GUI screenshots for search and automation. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology* (UIST '09). ACM, New York, NY, USA, 183-192. https://doi.org/10.1145/1622176.1622213
39. The Tools Designers are Using Today (2015 Survey) http://tools.subtraction.com/ – Accessed April 01, 2018.
40. Adobe Animate (formerly Adobe Flash) https://www.adobe.com/ca/products/animate.html – Accessed April 01, 2018.
41. Arduino http://arduino.cc – Accessed April 01, 2018.
42. Microsoft MakeCode https://makecode.com/ – Accessed April 01, 2018.