AFTERIMAGE: COLLECTING AND REPLAYING GEOSPATIAL MEMORY

A Thesis

Submitted to the Faculty

in partial fulfillment of the requirements for the

degree of

Master of Science
in
Computer Science with a concentration in Digital Arts

by

Hyun Ji Seong

DARTMOUTH COLLEGE

Hanover, New Hampshire

May 15, 2018

Examining Committee:

Chair _____
Lorie Loeb

Member _____
Xia Zhou

Member _____
Tim Tregubov

_____
F. Jon Kull, Ph.D.
Dean of Graduate and Advanced Studies

ABSTRACT

AfterImage is an interactive wall consisting of light detecting units controlled by microprocessors. Each light-sensing unit determines user proximity through the intensity of the shadow being cast and turns these shadows into bright snapshots portrayed through built-in LED lights.

AfterImage extends the concept of memory to a space and location by imitating two unique characteristics of human memory: a lingering afterimage effect of intense stimuli, and a flashback of events that happened in the past. To create a more organic representation of flashbacks, a native data structure is also explored.

Through the collection and reproduction of geospatial memory, *AfterImage* attempts to change a user's perception of existence and connection between users across time. This paper discusses the development and production of *AfterImage* as a piece of interactive art.

Acknowledgements

To Lorie Loeb, my advisor who always pushed me in the right direction and encouraged me to continue.

To Xia Zhou for guidance and materials for the project.

To Tim Tregubov for words of encouragement when I needed them the most.

To my family, who never cease to believe in what I can do.

To my friends who stood by me and would check in on me even at the worst times.

To Greg and Janet at the Dartmouth Woodworking Studio, who helped me create the largest project in my life.

To the Neukomm Institute for providing me with a wonderful studio space.

And to all the people in my life who have been a part of my life, making me who I am now.

Contents

Introduction 1
Previous Work 3
Implementation 6
    Shadow Detection 6
    The Afterimage Effect 9
    Memory Replay 11
    Network 13
Fabrication 18
    Wiring 18
    Wood Structure 21
User Interviews 24
    Interaction Patterns 24
    Comments 25
Conclusion and Future Work 26
Appendix A: Role of Light in Everyday Life 28
Appendix B: Breadboard Layouts 29
Appendix C: Phototransistor sensitivity 30
Bibliography 31

List of Tables

1 Data size and communication rate at different levels in the network 17

List of Figures

1 Detail view of *AfterImage* 1
2 negative correlation between Arduino light reading and the LED output voltage 6
3 Graph of arduino readings with various smoothing methods 9
4 Various smoothing methods with the lowest reading bolded 11
5 Diagram of connectivity between devices. Graphic elements courtesy of Fritzing. 14
6 Wiring overview with segmentation and numbers used to identify each unit 19
7 A wired unit with minimal functional connections for light sensors and LED lights 20
8 Drawing of groove with measurements and photo of test piece with groove . 21
9 Drawings showing tenon and mortise joinery and picture of real joinery on the frame 22
10 Drawing of base with different colors for different sections and photo of actual base 23
11 Mind map of "the role of light in everyday life" 28
12 Explored breadboard layouts 29
13 Phototransistor readings with blocking applied at different distances 30

INTRODUCTION

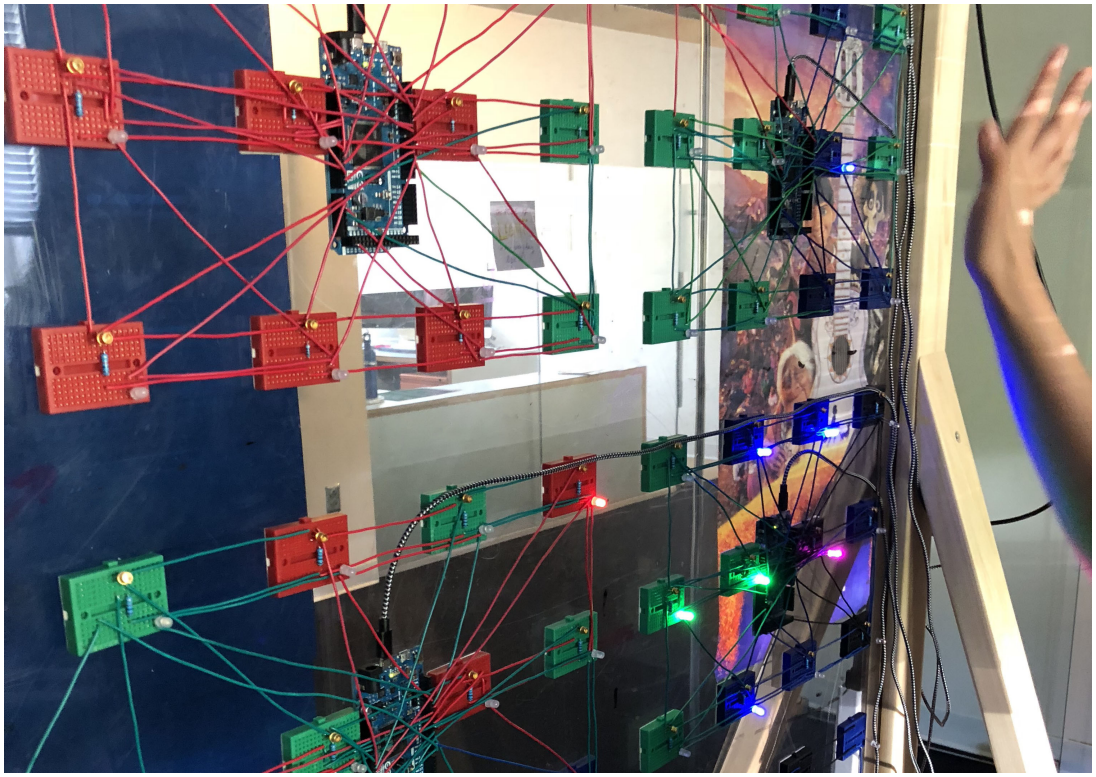

Figure 1: Detail view of *AfterImage*

Memory has always been a topic of interest to me. Some memories fade away or change with time, but some stay crystal clear in people's minds. Some happenings in time are remembered as distinct memories, whereas some things are never noticed and forever lost. The same set of events can be remembered completely differently according to the person. But events of the past and their memories define a person's present identity and existence. This fragile, volatile, yet impactful characteristic of memory is what makes it so precious and fascinating.

In September 2017, I was put in contact with the DartNets Lab at Dartmouth College. The lab had been looking for a way to create something different from the materials they had used to create *LiSense*, a testbed that utilizes visible light communication to promptly reconstruct the three-dimensional skeleton of the user [Li et al., 2015]. When I was provided a set of phototransistors to work with, I took some time thinking about the role of

light in the everyday lives of people and how it can be tied to the topic of memory and existence.

Light is needed for vision, which allows people to perceive and recognize objects. An object communicates its existence in space by bouncing off light. A person's spatial existence inadvertently changes the trajectory of the light rays and creates a shadow.

With the help of phototransistors and microprocessors, *AfterImage* embodies the idea of geospatial memory and existence. An acrylic sheet wall with light-detecting units acknowledges a user's existence through the shadows and remembers the interactions that happened in the space right in front of it. Just like how a person recalls a happening in the past, *AfterImage* will replay interactions by a user a while after the event happens.

You are sharing your space with someone who passed by some time before you. Although you may not be able to see them in the same space at the given moment, the space changed in some way due to their presence. No matter how small or short-lived the change might have been, it definitely existed. *AfterImage* prolongs the lifespan of such ephemeral interactions using the concept of geospatial memory, connecting what existed in the past with the present.

AfterImage provides a unique experience that combines the responsiveness of a computer system and the spontaneity of human interaction. By extending the concept of memory to a space and location, can a set of lights and sensors embodying the idea of geospatial memory create a sense of existence and connection between users?

PREVIOUS WORK

Visible Light Communication *LiSense* is a system that enables real-time human skeleton reconstruction of the user standing on top of a 3 m by 3 m testbed [Li et al., 2015]. By using "light beacons" of various frequencies that shine on the testbed at different angles, the system is able to separate the shadows cast by the different light beacons and infer the 3D user skeleton using the separated shadow maps. The skeleton can then be used as a controller, allowing users to control devices or play games just with their body. *AfterImage* uses a similar approach of using the subject's shadow to detect the user input but is a standalone device with a built-in display that does not need the data to be sent to a different program. Also, *AfterImage* does not require a specific light setup, permitting the piece to be in a larger variety of spaces.

Collaboration and Art *CLOUD* is an interactive sculpture created from thousands of incandescent light bulbs and pull switches [Brown and Garrett, 2012]. By physically switching the lights on and off, the user can determine how the piece looks. At some instances, strangers would collaborate to turn all the switches on simultaneously. *CLOUD* was especially inspiring in showing how interactive art pieces could bring out the collaborative nature in its users. *AfterImage* extends the idea of the user actively being involved in creating the visual outcome, but facilitates the user's involvement even more by a hands-free interaction and encourage collaboration that can happen non-simultaneously.

Interactive Sensor Art With increased access to sensors and systems, interactive art pieces utilizing user participation have been a popular topic of research for the past two decades.

Hello.Wall is a good example of an early approach to interactive art using computer systems in the wall format [Prante et al., 2003]. *Hello.Wall* is composed of 124 cells that include LED clusters and short-range transponders, making the wall a visual display and a

radio message transmitter and receiver. In order to interact with the radio communication aspect of the wall, however, the user needs to be equipped with a radio device that is configured to work with the system. Thus, a passerby with no prior knowledge of the wall will not be able to gain the full interactive experience. *AfterImage* can be interacted by anything with a physical form, providing an inviting experience for anyone.

Digiwall is a climbing wall that combines elements of computer gaming with the physical climbing experience [Liljedahl et al., 2005]. The climbing holds of *Digiwall* have LED lights that can serve as visual cues and capacitive sensors that allow the system to acknowledge when a hold is being handled. Using a combination of visual and aural methods, the system hosts several interaction models including games, competitions, and challenges. This setup offers a rich physical interaction, but is highly goal-oriented and does not leave room for an ambient, reflective experience.

Transition Soundings is an example of a public sound art piece that uses proximity sensors and light sensors to create an immersive sound experience at a bus stop [Birchfield et al., 2006]. The case study highlights a few things to consider about interactive art that is different from art that is hosted in galleries or museums. Since interactive art is prone to unexpected contact from the engagement of the viewers, more attention needs to be paid to maintenance and durability. The creators of *Transition Soundings* designed the system so that even when some parts of the piece break or malfunction, the rest of the piece could continue to run smoothly. Also, there needs to be prior research and planning around the environment to ensure a meaningful user experience within the context. This is especially true for art hosted in public spaces since members of the audience might not be voluntarily "seeking an art experience." The introduction of the piece allowed room for bus riders to rediscover a space that they are familiar with and even interact with strangers talking about the art. *AfterImage* has a similar goal, but instead of limiting the experience to the current time, interactions from the past are stored and retrieved work together with interactions happening in the present, creating a collaboration that transcends time.

Bystander had a very careful approach using “moving bodies” as the primary input in their interactive experience, using infrared cameras to sense audience movement [Robertson et al., 2006]. The movement data were analyzed to determine the engagement of the users within the exhibition, and engaged users will be rewarded with more information. The creators of *Bystander* started with role-playing, improvisation, and "bodystorming" to gather an idea of the kinds of movements that might occur in the space hosting the art. Movement-oriented personas and scenarios from these activities helped the designers gather ideas on how to orient the user experience and later enact scenarios based on movement schemas to carefully test the experience. Rather than collecting and analyzing the data over time, *AfterImage* offers a quick, direct response to the user input to incentivize the user to spend more time with the piece.

IMPLEMENTATION

Shadow Detection

AfterImage consists of a 16 by 16 grid of breadboards arranged on a six-foot-high, six-foot-wide acrylic sheet. Each breadboard—referenced in this paper as "pixels"—is equipped with a Darlington phototransistor (Honeywell SD3410-001) and a 5mm LED light diode. The whole piece is controlled by 22 Arduino Due units that govern twelve or fewer pixels.

The basic underlying mechanism of the interaction is simple: when an object exists in a space that has light, the physical form blocks light and creates a shadow. The closer the object is to the surface that the shadow is being created, the sharper and darker the shadow will be. Thus, by reading the amount of light that is blocked, the light sensors can serve as pseudo proximity sensors.

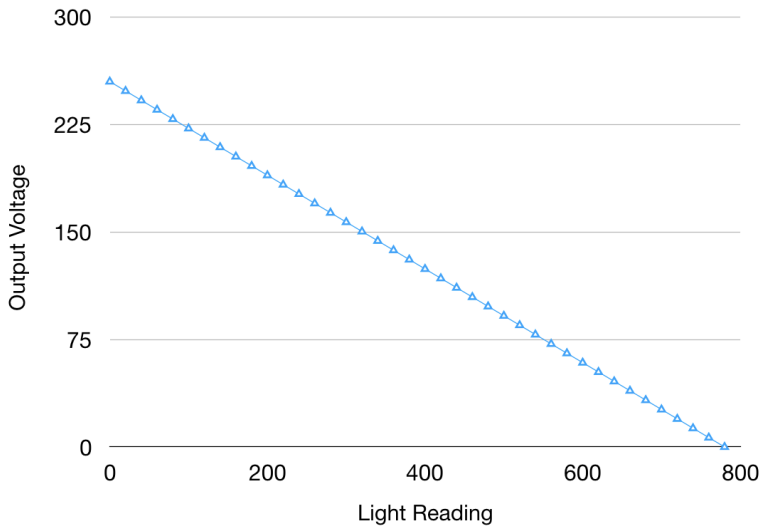

Figure 2: negative correlation between Arduino light reading and the LED output voltage

Thus, *AfterImage* utilizes a negative correlation between the light reading and the brightness of the LED lights—the closer the user, the stronger the shadow, and the brighter the pixel will light up. To accommodate the low density of the pixel grid, *AfterImage* uses a

continuous variation in the brightness of the lights rather than binary on and off states. This allows the display to have a fading effect that creates a more natural animation of the lights.

Light Reading To get data from the phototransistors, *AfterImage* uses the same method from *LiSense*: "cascade the photodiode and a resistor and measure the resistor voltage," which gives a value from 0 to 1023, 0 being the darkest and 1023 being the lightest [Li et al., 2015]. The specific phototransistor used for *AfterImage* has a field of vision of 90^{o}, detecting light and shadows up to 45^{o} from the straight line from its center. For best results, the light source in the space should be placed facing the sensing side of the wall so that there is a significant amount of light that travels into the phototransistors.

The phototransistors are connected in a parallel circuit, sharing the same power source and ground. The Arduino Due operates at 3.3V, so the collector of the transistor is connected to the 3.3V pin on the Arduino, which limits the Arduino reading to lower than the highest possible reading of 1023. In order to make sure that the readings are as accurate as possible, the sensing units of the circuit are detached from the rest of the wiring, not sharing the power source nor the ground with the other elements such as the LED lights or the serial communication connections.

Although the Arduino readings through the phototransistor are directly connected to the perceived light, the data is very sensitive and needs to be filtered. When the light readings were taken in raw and mapped straight to the lights, some of the pixels had a very noticeable flicker. Plotting out the readings of a single pixel revealed a lot of noise that caused the output voltages on the lights to vary. The phenomenon is more visible when artificial light is used as the main source of light, due to the utility frequency of the power source of lights. Studies have proven that an average person can perceive flickers happening at 50 to 90 Hz [Davis et al., 2015]. Although the utility frequency of 60 Hz may not be directly visible from the light source, the light mapped to the small LED light is more evident.

To tackle this noise, simple moving average (SMA) and exponential moving average (EMA) equations were tested. Simple moving average and exponential moving average can be represented as the following:

$$SMA_t = \frac{1}{n}\sum_{i=0}^{n-1} p_{t-i} \tag{1}$$

$$EMA_t = \begin{cases} p_1, & \text{if } t = 1 \\ \alpha \cdot p_t + (1-\alpha) \cdot EMA_{t-1}, & \text{otherwise} \end{cases} \tag{2}$$

where n is the sample size, α is a constant smoothing factor between 0 and 1, and p_t is the value at time t. When the α is higher, the more weight is given to the most recent reading of p_t.

Both methods offer effective smoothing, but with different characteristics. As it can be seen in figure 3, SMA tends to have a slower response time to changes in the reading, especially when the sample size is larger. EMA has a less delayed representation of the current input but does not take into account how long the trends lasted before the change in the input. A combination of these characteristics can be used to create appropriate visual effects.

Initialization *AfterImage* is built to be a movable device that could function in a variety of spaces. According to the time of day and how controlled the environment is, the intensity of ambient light changes. And according to this change in ambient light, the reference point in which the sensor readings are considered significant enough to trigger a reaction will also need to be updated.

Before any light readings happen, *AfterImage* goes through an initialization process to gain a sense of what the readings tend to be with no user interaction. This initialization process, which happens when the device is turned on or when an Arduino is reset, collects the readings from each sensor for five seconds, populating the values for SMA calculation.

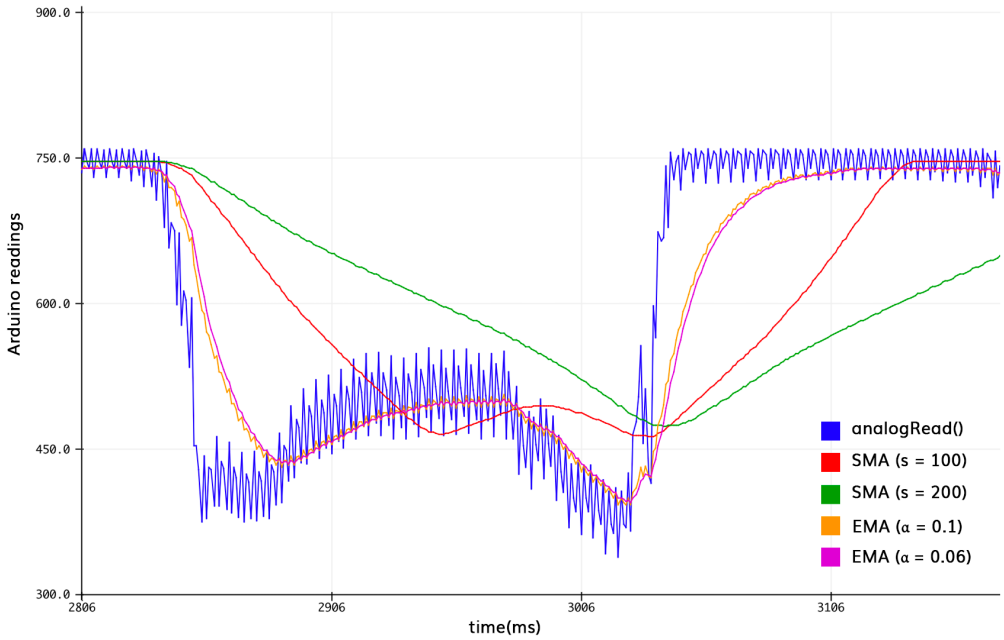

Figure 3: Graph of arduino readings with various smoothing methods

During this time, the lowest reading is constantly updated, being recorded in the system as the low threshold. At the end of the initialization process, this lowest value is multiplied by a constant of 0.97 to lower the value slightly and leave room for any unexpected noise that may be introduced.

The reference point created by the initialization is valid as long as the ambient light of the space does not change. But when the piece is placed in a room with ambient light that that fluctuates greatly over time, the low threshold recorded in the beginning may not be relevant at a later time. A possible current fix is to restart the system and allow the initialization process to run again and update accordingly, but this will cause the memory sequences saved for replay to be reset as well.

The Afterimage Effect

The visual effects of *AfterImage* are closely tied to the characteristics of human memory. When a "memorable" event happens, people tend to mull over the events soon after they

happen. This lasting of a stimulus after it stops is also observed as a visual phenomenon, such as camera flashes causing a dark spot in one's vision for a while after the actual flashing. This phenomenon, known as physiological afterimages or persistence of vision, inspired the name of *AfterImage* [Gersztenkorn and Lee, 2015].

To create a similar effect using the light readings, *AfterImage* uses a combination of the analog smoothing methods discussed in the previous section. Simple moving averages with a larger sample size can provide an insight into what trends were prominent during the period in which the data was collected. For example, if a series of low values was recorded for an extended period of time, the stored values will prevent the SMA from rising up too fast, acknowledging the significance of the low trend. In the case of *AfterImage*, there are two scenarios in which this delay can be seen: a delay in the downward trend when a shadow is introduced after a long time of high readings, and a delay in the rising of the value when a shadow is removed after an extended period of low readings.

In the first case, the delay in the downward reading causes the calculated value to lie above the low threshold level for some time after the shadow is introduced. This will cause the lights to turn on after a noticeable wait, or even prevent the lights from turning on at all if the shadow is applied only for a short amount of time. For a user engaging with *AfterImage* for the first time, this lag will cause confusion regarding the relationship between their actions and the visual response of the piece. Thus, the exponential moving average's ability to reflect the most recent change in a more punctual manner is more appropriate.

In the second case, the delay in the simple moving average can be used to intentionally cause a delay in the turning off of the lights. Just like physiological afterimages caused by strong stimuli, the pixels will have a tendency to remember low readings that have been recorded over a longer period of time. These lingering low readings will cause the lights to stay on for a while after the user leaves and fade out over time.

Thus, by taking the lowest of the averages at a certain time, *AfterImage* can achieve both a prompt response in switching the lights on and a lingering of the input to create the

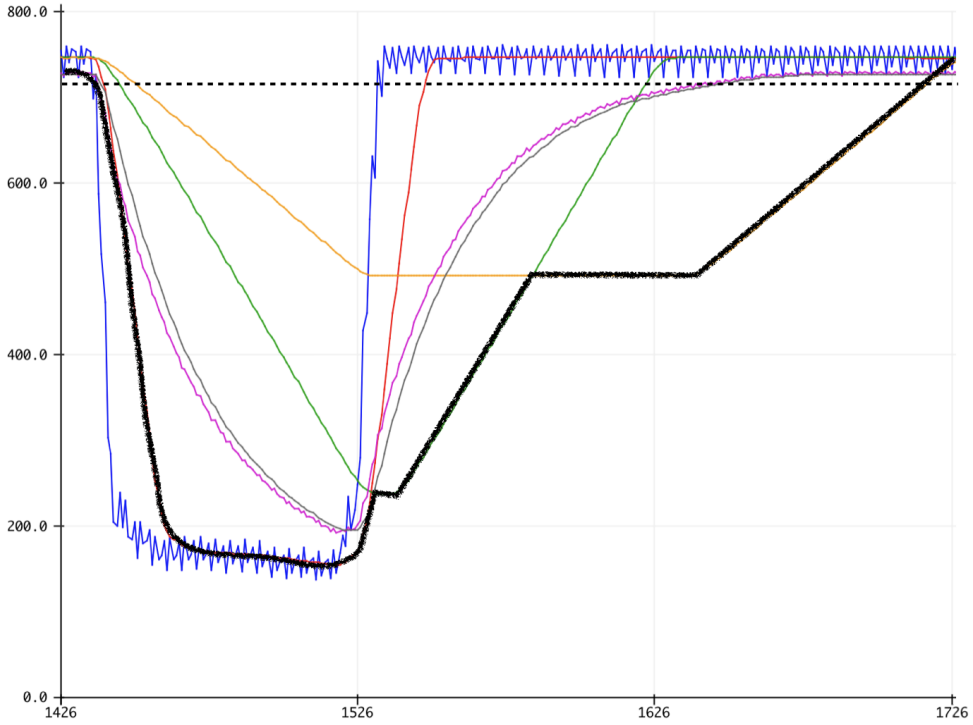

Figure 4: Various smoothing methods with the lowest reading bolded

afterimage effect.

Memory Replay

Another characteristic of memory that *AfterImage* portrays is the ability to create memory flashbacks. In order to replay interactions from the past over the 22 separate Arduinos and make the best use of the limited amount of memory, *AfterImage* takes advantage of the Arduino's ability to keep track of time and create a loop that runs based on an internal timer.

Human vision retains an individual visual stimulus for fifteenth of a second, perceiving an image that is provided within a fifteenth of a second as a continuation of the previous image [Read and Meyer, 2000]. A frame rate higher than 15 frames per second will cause a viewer to take in a sequence of images as a continuous motion. By using the Arduino's internal timer and providing the system with a new snapshot every 60 milliseconds, the

stored memory can be replayed at around 17 frames per second, which is sufficient to create a sense of motion. Thus, instead of storing each and every moment of an interaction, *AfterImage* optimizes memory usage by storing snapshots at these 60-millisecond intervals.

Because the low threshold of the system can change at different times, the same light reading at may be above or below the low threshold according to the initialization results of a particular session. The resulting voltage output would be a better reference for the interaction snapshots.

The Arduino can control the output voltage of a pin using pulse-width modulation. The function `analogWrite` takes in an integer value between 0 and 255. This number range can be represented by an `unsigned char` or `byte` data type, which is smaller than an `int`. *AfterImage* remembers interactions by recording the output voltage of the pixels using a two-dimensional `byte` array, one dimension for the number of pixels being controlled by the Arduino unit and the other for time. To simulate memory replay, the time is traveled in a loop—once the end of the array is reached, the pointer is brought back to the beginning of the array, just like a clock handle that goes around in a circle. When the timer encounters data that is greater than 0, the stored output voltage will be mapped to the lights. Each pixel has its own timeline of voltage readings, which can be updated independently of other pixels. When a user interacts with one area on the wall while a replay is happening on a different part of the wall, only the pixels being interacted with will update their data. The next time a replay happens, both interactions will be visible.

For a more direct response to the user's interaction, the main process of the system operates every 5 milliseconds. Because the time interval of 60 milliseconds is much longer than the 5-millisecond interval at which the system runs, only the most recent valid input during the 60 millisecond period is put into the memory. If an interaction happens only for a fraction of the 60 milliseconds, the snapshot will still be stored since the memory is only updated when there is an output voltage higher than 0.

Rather than having an interaction sequence stay in the Arduino's memory forever, *Af-*

terImage fades out a memory by lowering the intensity of the stored output voltages every time a value is used in a replay. When the lowered number is below 5, at which the light of the LED is barely visible, the voltage stored in the array is reset to 0. The rate at which memory fades can be configured to control how long a sequence lasts within the system. When there is an interaction that is stronger than what is currently stored in the memory, the stronger value will get stored. For example, if a new user creates an interaction that is more recent and stronger on top of an old, faded memory sequence being replayed, the new interaction will determine the values being stored.

Network

Although the looping of memory at a constant time interval creates a simple and visually enticing memory recall system, it is not an accurate representation of how memories are recalled in real life. Flashbacks tend to happen in a more random and organic manner, both in terms of when they happen and what they are about.

In an attempt to create this intricate representation of memory, AfterImage utilizes a network structure that stores snapshots in an external server. Because the 256 pixels work together to make a single snapshot, it is crucial that the information is comprehensively in sync. The network allows data from the sensors to be stored and retrieved as a single data structure using a combination of serial communication and Bluetooth.

Starting from the Raspberry Pi server as the root, the network can be represented as a tree structure. A Raspberry Pi was chosen as the server due to its portability and performance. Figure 5 represents a diagram of the network involving a single Raspberry Pi and 22 Arduinos using serial communication integrated with Bluetooth.

The architecture of the tree is based on the fact that the Arduino Due has 4 built-in serial ports. One port is always saved for the parent, which leaves room for each Arduino to have no more than three children. Although serial communication speed can be customized by adjusting the baud rate, the highest possible rate is limited to the HC-06 Bluetooth module's

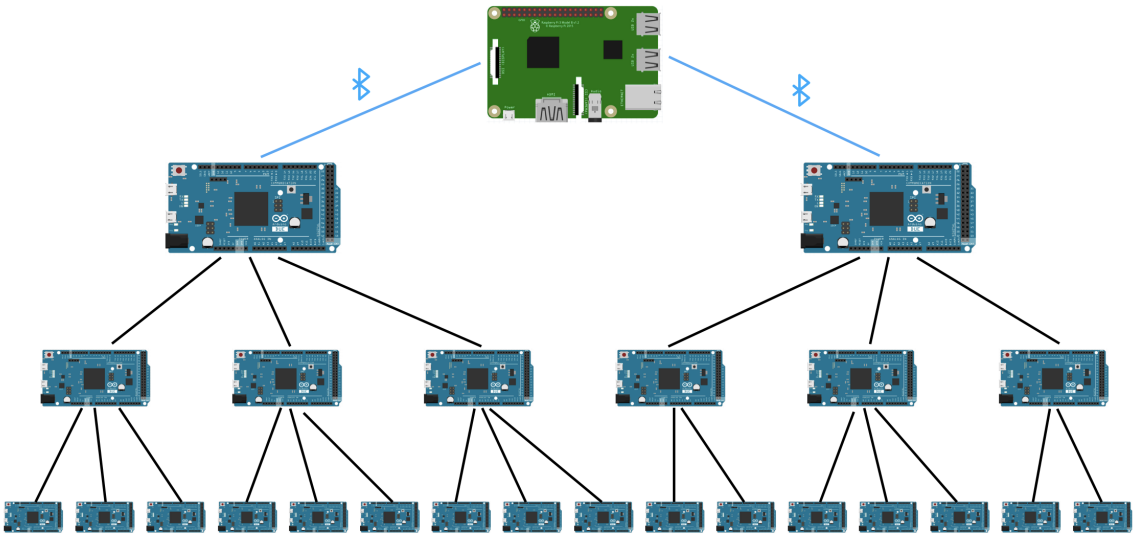

Figure 5: Diagram of connectivity between devices. Graphic elements courtesy of Fritzing.

maximum baud rate of 115200.

Data Representation As discussed in the previous section, since the environment in which *AfterImage* is presented is sensitive to differences in ambient light, saving the light readings is not a reliable way of representing the interaction sequence. Using the same model, the output voltages can be represented as a `byte`, along with an identification number for the Arduino since multiple Arduino units are involved in creating the whole image. The data from a single Arduino could be represented as a data structure like this:

```
typedef struct {
  int arduinoNumber;
  int voltageVal[12] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
} arduinoData_t;
```

This kind of data structure would take up 13 bytes, accumulating to 278 bytes for the 22 Arduino units and 256 pixels involved in *AfterImage*. To transfer this information through serial communication, the data structure is converted into a byte array and copied into the same data structure defined on the other side. In order to copy the data into a structure,

the specific byte size of the structure needs to be predetermined. Although this model is compact and fast, there is too much of a risk of data corruption. Due to the way Arduinos process serial data, in the case that a one or a few bytes get lost in the constant stream of simultaneous incoming data, following data will take the place of the lost data. This means that values that belong to one part of the data structure may be assigned to a different variable, and once this corruption happens, it continues to live in the system until it is reset.

To prevent such cases of data corruption, the network attempts to communicate the voltage data using specifically formatted strings. In the particular scenario of *AfterImage*, there are less than a hundred Arduinos, which means the `arduinoNumber` will always be a number that can be represented by less than two characters. The `voltageVal` ranges from 0 to 255, which is going to be a number that can be represented by no more than three characters.

Unlike the fixed sizes of the predetermined data structures, the string is going to have a varying length according to the numbers being output. Since serial communication reads in a stream of data byte by byte, there needs to be an indication of the beginning and end of the data set. In the case some data is lost through the communication, the start and end indicators can make sure that the next set of values are parsed from the right point. The data pieces are also separated using special characters, which can later be used to parse the data. Following these guidelines, the data can be represented in a way like this:

```
<2:0 0 0 0 0 0 0 0 0 0 0 0>
<12:202 194 209 189 177 103 102 204 206 102 109 111>
```

Here, the first number is the Arduino number, separated from the rest of the data with a colon. This is then followed by twelve numbers that are separated by spaces, which are the voltage values of the twelve pixels. The upper example, which is the smallest data possible with a single character Arduino number and single character voltage values, has a byte size of 27. The lower example, which is the maximum number of characters possible in this scenario, has a byte size of 52.

An advantage of this data structure is that the strings can directly be extracted and added on to additional data. On a parent Arduino, the data part of the message can be isolated by taking out the start and end delimiters. This data message is attached to the master message, and a special character '/' is added at the end to separate the data from additional messages that will be added. Messages from the other children are handled and added to the master message accordingly. Then, the local data is represented in the same way and appended at the end of the master message. This process can be done repeatedly, no matter how deep the tree goes. When the data is combined, data from one Bluetooth node on the tree looks something like this:

```
<12: 0 0 140 166 0 0 0 0 0 0 0 0/8: 0 0 0 0 0 0 0 0 90 189
145 73/10: 0 0 0 0 0 0 0 0 0 0 0 0/11: 0 0 0 0 0 0 0 0 0 0
0 0/2: 0 0 0 0 0 0 0 0 0 0 0 0/4: 0 0 0 0 0 0 0 0 0 0 0 0/
3: 0 0 0 0 0 0 0 0 0 0 0 0/7: 0 0 0 0 0 0 0 0 0 0 0 0/9: 0
0 0 0 0 0 0 0 0 0 0 0 0/5: 0 0 0 0 0 0 0 0 0 0 0 0/1: 0 0 0 0
0 0 0 0 0 0 0 0/6: 0 0 0 0 0 0 0 0 0 0 0 0>
```

In the end, the Bluetooth Arduino sends the collective data in the form of a string, line by line. The data is then received by the Raspberry Pi and stored in a file, which can later be retrieved and sent back to the Arduinos. The Arduinos can then parse the data using the delimiters and convert the data into the proposed data structure `arduinoData_t` and pull values accordingly.

Because serial communication is much slower than the Arduino processing speed, having a large amount of data traveling between machines will cause delay. The closer the Arduino is to the root of the tree, the more data is being sent and received through the Arduino. To accommodate the fact that parent nodes communicate three or four times more data with their parent than what they are receiving from each child, the baud rate is respectively increased for each level in the network, going up to 115200 for the Bluetooth Arduinos.

Level	No. of Arduinos	Data Size (bytes)	Baud Rate
1	1	27–52	19200
2	3–4	83–212	57600
3	11–13	307–688	115200

Table 1: Data size and communication rate at different levels in the network

Problems Through some testing, it was proven that this method of data processing is suitable for acquiring data from the Arduinos. Examining the incoming data received from the Raspberry Pi revealed that the exact number of Arduinos and pixels were represented in the incoming string. Also, when the formatted string was directly put into an Arduino unit using serial communication through a USB cable, the unit parsed the string properly and replayed the interactions of its respective part based on the data.

But when the same data was sent to the Arduinos via Bluetooth, only some of the units were responding accordingly. There seem to be some nested serial communication issues that cause some bytes in the serial data to be sporadically lost since the Arduino units are handling local data and information coming in from four different serial ports in the same sketch. Due to time limitations, the replay feature was not implemented successfully, but the reliable data collection through the proposed network and data structure opens up discussion about additional operations such as real-time data visualization.

FABRICATION

Wiring

A 6 feet by 6 feet panel of half-inch thick acrylic was used as the canvas for laying out the pixels. Acrylic has both reflective and transparent characteristics—when a user stands in front of the acrylic, they can see through the glass and also see a reflection of themselves. These features were deemed relevant to the topic of geospatial memory, allowing the user to gain a better sense of their existence in the space.

Using the resources from the DartNets Lab, the maximum possible square layout was a 16 by 16 grid. Multiple visual patterns were explored before a gradient-like layout was chosen to serve as a reference for the placement of the various colored breadboards. Making the overall aesthetics of the wall pleasant was important to attract more people to come closer and realize that the piece is, in fact, responsive to their actions.

The hardware was the main limiting factor in determining the connectivity of the layout. *AfterImage* uses the Arduino Due's twelve analog input pins for the light readings, enabling each Arduino to collect data from and control twelve pixels each. Once the breadboards were laid out, twelve or fewer breadboards were grouped into a section to be controlled by an Arduino. On each breadboard, a phototransistor and an LED light are placed, the phototransistor providing the input and the light creating the output accordingly.

The Arduinos are connected to a power source using a micro-USB to USB cable. These cables, which can be a little distracting due to their dark color, follow the edge of the breadboards and on to the edge of the wooden frame, hiding from the view of the plane as much as possible.

The wiring itself is a large part of *AfterImage*'s visual identity. The wires are like brushstrokes in a painting that connect areas of color to create the overall image of the piece. To create a circuit that is both aesthetically pleasing and functional, almost all of the wiring was done using solid hook-up wire that could be cut to desired lengths. By

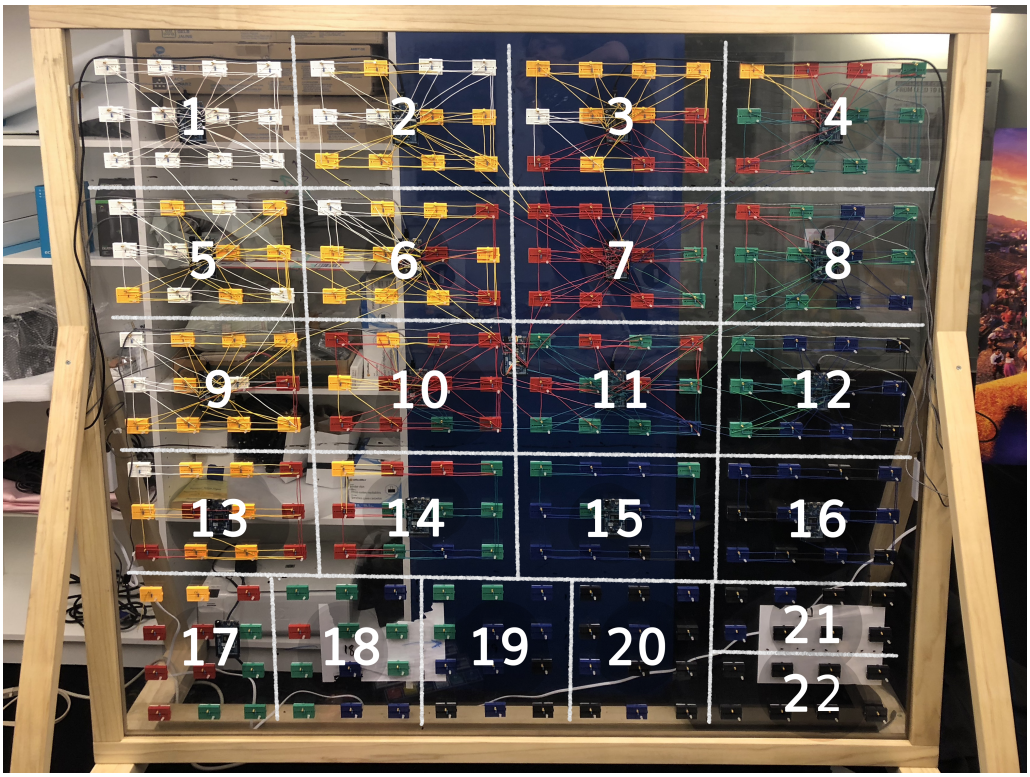

Figure 6: Wiring overview with segmentation and numbers used to identify each unit

making the wires taut and straight, the wiring was intentionally made to have a string-art like visual. Using the same color wires as the breadboards help enhance the color flow of the piece and create a more comprehensive visual connection. Figure 7 presents a fully functional Arduino unit with the wires necessary for performance—after the minimal functional wiring, additional wires were added in between units to visually connect the units more.

LED Lights In order to create the fading effect of the lights for smoother animation, it was necessary to use analog outputs, which would change the output voltage going through the pin to a value between 0 and 255. The Arduino Due has twelve pins that are designed to create this analog output effect using pulse-width modulation (PWM). Although using RGB LED lights would have greatly expanded the visual capability of the project, a single RGB light has to use one PWM pin for each color channel. This would limit the number

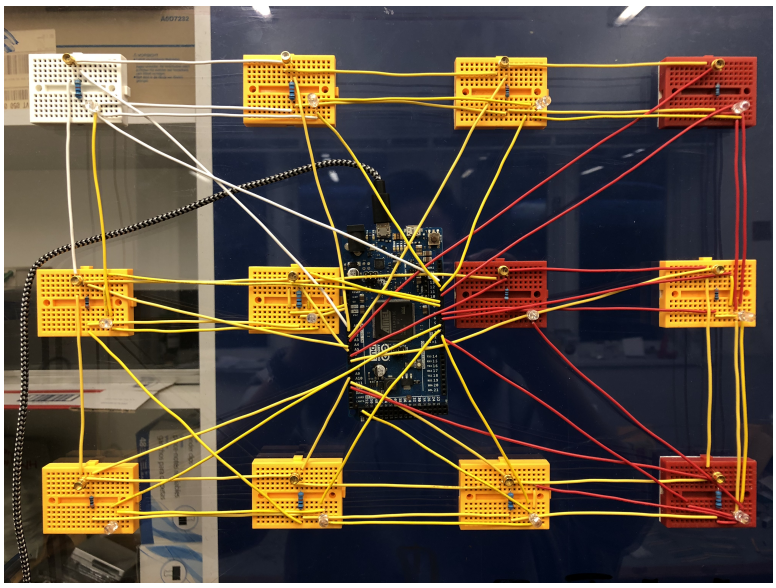

Figure 7: A wired unit with minimal functional connections for light sensors and LED lights

of lights that can be controlled by an Arduino to three.

To be able to control all the pixels with the limited number of Arduinos, *AfterImage* uses single-color LED lights. The light colors were matched with the underlying breadboard color for visual continuity. This facilitates the user experience by providing a direct correlation between what the user sees before and after they trigger the lights. Considering the dimensions of *AfterImage*, having a light turn on a pixel that is not in the vicinity of the user means the lights will have to be viewed at an angle. Clear LED lights tend to not be very visible from an angle, so diffused LED lights were deemed more suitable for the project.

Because PWM pins are used to determine the brightness of these LED lights, the voltage coming into the lights are always regulated. Thus, there is no need for a resistor between the voltage source and the LED light.

Wood Structure

In order to keep the acrylic sheet stable and add mobility to the project, *AfterImage* is encased in a wooden frame with wheels. The frame and base elevated the acrylic plane off the ground, making it around 6.5 feet from the ground. This granted the wall to cover a more active area since feet and leg movement tends to be less intense than hand and arm movement [Hoffmann, 1991].

Acrylic is very heavy: the piece used in AfterImage is calculated to weigh around 50 kilograms. With the wiring and additional parts assembled on top, the whole piece is expected to weigh more than 55 kilograms. Since interactive art is always prone to touch and occasional rough handling, it was of utmost importance to create a sturdy structure that could offer a safe experience to the users.

Frame The basic structure of the piece is like a moving wall or a movable partition—an upright plane attached to a base with wheels on the bottom. To tackle the flexible nature of the acrylic, *AfterImage* needed a frame that goes around all four edges of the plane of the acrylic and keeps the plane flat. A significant amount of the acrylic sheet needs to be enclosed within the frame to secure it in place. The frame has a half inch wide, 1 inch deep grooves that embrace the acrylic sheet.

Figure 8: Drawing of groove with measurements and photo of test piece with groove

A shorter test piece was used to check the fit of the acrylic into the groove. With confirmation that the grooves encased the acrylic perfectly, the actual frame was made by

creating the four edges separately and then assembling them together. Creating four pieces of the same length with a 45° cut on each end would have been the typical way of creating a frame, but this was not appropriate due to the size and weight of this project's size and weight. Having an angle at the joints meant that there could be some slipping and gradual disassembling of the frame.

Thus, a tenon and mortise joint was chosen to assemble the frame instead. A tenon was cut from a conjoining frame edge by shaving away the end of the wood piece to reveal a half inch thick, 1.5-inch high block, fitting into the grooves of the frame perfectly. By using the extension of the groove as part of the joinery, the frame could be assembled using just a single drywall screw at each corner.

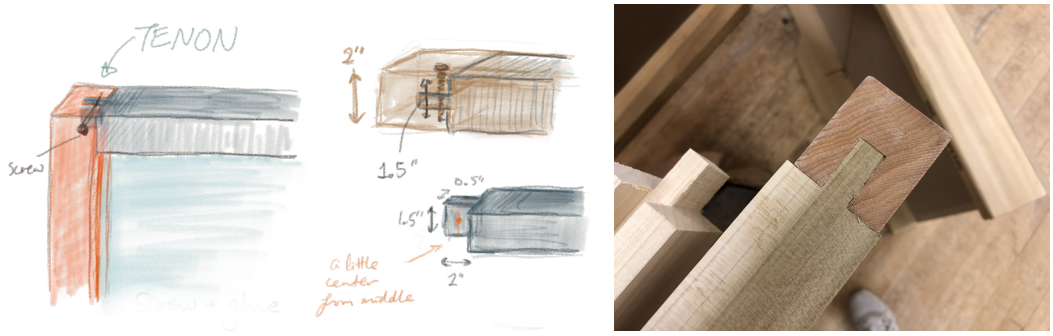

Figure 9: Drawings showing tenon and mortise joinery and picture of real joinery on the frame

Base Since the frame with the acrylic sheet is quite heavy, any tipping or introduction of an angle could cause the structure to fall. Having the frame connected to the base with braces and making sure that the frame is always upright can prevent the structure from having any moment force towards the front or back. The wider the angle of the braces, the more stable the structure will be.

But having a wider angle on the braces meant that the base had to be quite wide. In terms of the interaction, having a wide base is not ideal because the base could prevent people from getting close to the actual active area of the piece.

Thanks to the regular distribution of weight along the frame, the frame could be attached

to the base using just two endpoints. This meant that the base does not need to be a solid piece. Instead, a simplified base was created with two end structures and two bridges connecting the two to prevent any unwanted rotation. The bridges were assembled closer to the middle of the structure so that when the user interacts with the piece, they could get as close as they need to be, yet keeping them from getting too close to the piece.

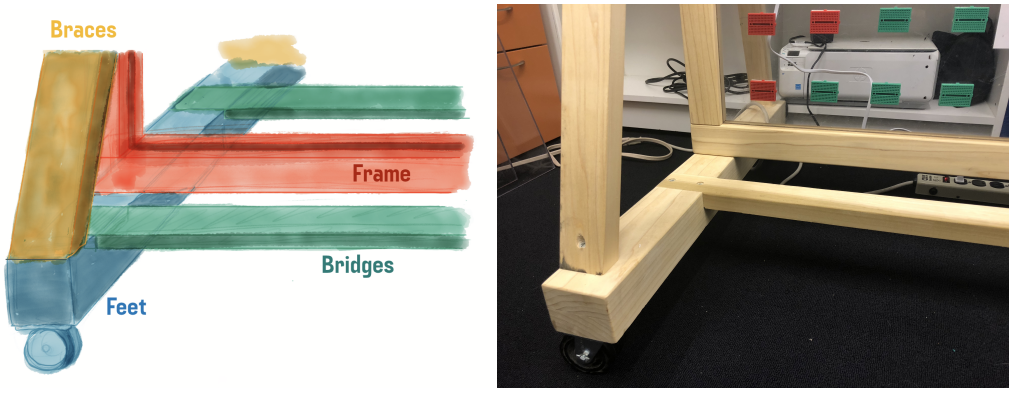

Figure 10: Drawing of base with different colors for different sections and photo of actual base

USER INTERVIEWS

Over the course of two days, ten individuals were invited to experience *AfterImage*. Four people interacted with the piece on their own, while others came in as a group of two people. Intended to be a test on how new users interact with the piece, they were introduced to the space with no background information. After a few minutes of interaction, the underlying idea was explained: a representation of geospatial memory in an attempt to create a sense of existence and connection.

Interaction Patterns

Although it was expected that users would use more hand and arm movement than leg movement, there was a surprisingly high number of participants who attempted to use their feet to interact with the wall. This pattern of action happened a lot especially when the participants were trying to take a video of the piece with their hands, limiting their movement to their feet. A lot of times, this caused the user to be further away from the surface, causing the lights to not turn on.

The participants' interactions could be broadly divided into micro and macro interactions. Some participants were more focused on the overall image they were creating on the whole wall, while others were interested in the fading on and off of the individual pixels. All participants eventually attempted a mixture of both interactions over the course of their experience.

When they were notified of the fact that the wall can replay interactions from the past, many participants tried to make a very distinguishable move and sit back, waiting for their unique pattern to be replayed. Once their interaction appeared on the wall, users would display their satisfaction and continue interacting, adding a new layer to the time loop. This pattern was more noticeable among the group participants, who would often take turns in interacting with *AfterImage*.

Comments

Even before they were given an explanation of the concept behind *AfterImage*, participants gave some insightful comments closely tied to the idea of existence and connection. One participant mentioned that "it's pretty novel to be able to see the past and the present at the same time." The other participant who was in the same session mentioned that the piece "brings people together," how "you leave your mark and other people can leave theirs," creating a "collaborative burst of color." Another tester described the wall as the canvas and the users as the artists, making the art by determining what lights get turned on.

Some participants mentioned that the experience made them be "more aware of what they do." Sometimes, they would see a replay of their own interaction and realize that the outcome looked different from what they expected.*AfterImage* provided a method for users to be detached from their own actions and really see what existed.

Many users were deeply touched by the subtle details in the representation of memory in *AfterImage*. One participant mentioned that the fading of the previous interactions are very representative of how "you never remember memories in the same way" and that "once you recall something it's kind of distorted."

A large majority of the participants found the experience satisfactory and entertaining. There existed some confusion regarding how close the user needed to be for the display to be triggered according to the ambient light settings, but many participants commented they enjoyed the experience of seeing themselves in a different light.

CONCLUSION AND FUTURE WORK

Conclusion Despite its technical limitations, *AfterImage* has provided users with a unique opportunity to leave their mark in time and interact with happenings in the past. Quotes from user interviews strongly suggest that this representation of geospatial memory creates an arena for collaboration and expression, allowing users to explore the sense of existence and connection with the space and with other people.

Future Work Although a sparse grid of pixels was enough to give the users a sense of human motion, the images projected on the wall were very rough. It would be ideal to have a higher density grid of pixels, with additional microprocessors and sensors. Using more lights than sensors would also help with increasing the pixel density, possibly by assigning lights in pixels without sensors and interpolating data from the input data from its adjacent pixels. Although there is a hardware limitation of having only twelve PWM pins, it is possible to program the digital pins of the Arduino to act like PWM pins through code. This opens up space for 42 additional LED lights per Arduino unit.

Since *AfterImage* was built with a possibility of being used in multiple environments and collecting the spatial data for different locations, there was no controlled environment that regulated ambient light. The pixels would have been the most reliable and responsive if the piece had been built and set up in a controlled environment. Having the piece shown in a controlled space like a gallery and really fine-tuning and optimizing the experience would be an interesting next step.

Using the current initialization method, change in ambient light will not be automatically recognized. Thus, if the piece is powered on for a longer period of time without the triggering of the initialization command, some light sensors might mistake the decrease in ambient light as a shadow caused by a user. A potential way to prevent such false triggering would be to train the system to recognize patterns in the readings created by humans. For example, human movement tends to happen on adjacent pixels. When a pixel is turned

on in a randomly scattered pattern, it is more likely that this reaction was caused by light changes. Another possible scenario is having an increase in ambient light—when the light readings are significantly higher than the threshold. By teaching the Arduino to re-initialize itself when such events happen, only when there is no user interaction happening in the rest of the piece, the piece will be able to retain its responsiveness. To create a more accurate and comprehensive representation of the interactions, it might be favorable to create an input unit composed of a combination of light and proximity sensors.

With the data of the interactions are stored on the Raspberry Pi server, it would be interesting to use the data to create visualizations using languages such as Processing. Just like how soundscapes aurally represent the rich combination of events happening around an area, data from *AfterImage* may be reproduced as a visual counterpart of spatial identity. This graphic element could be created live as the data comes in and work together with the existing pixel visualization through a projection played in the wall behind and viewed through the acrylic sheet.

Appendix A: Role of Light in Everyday Life

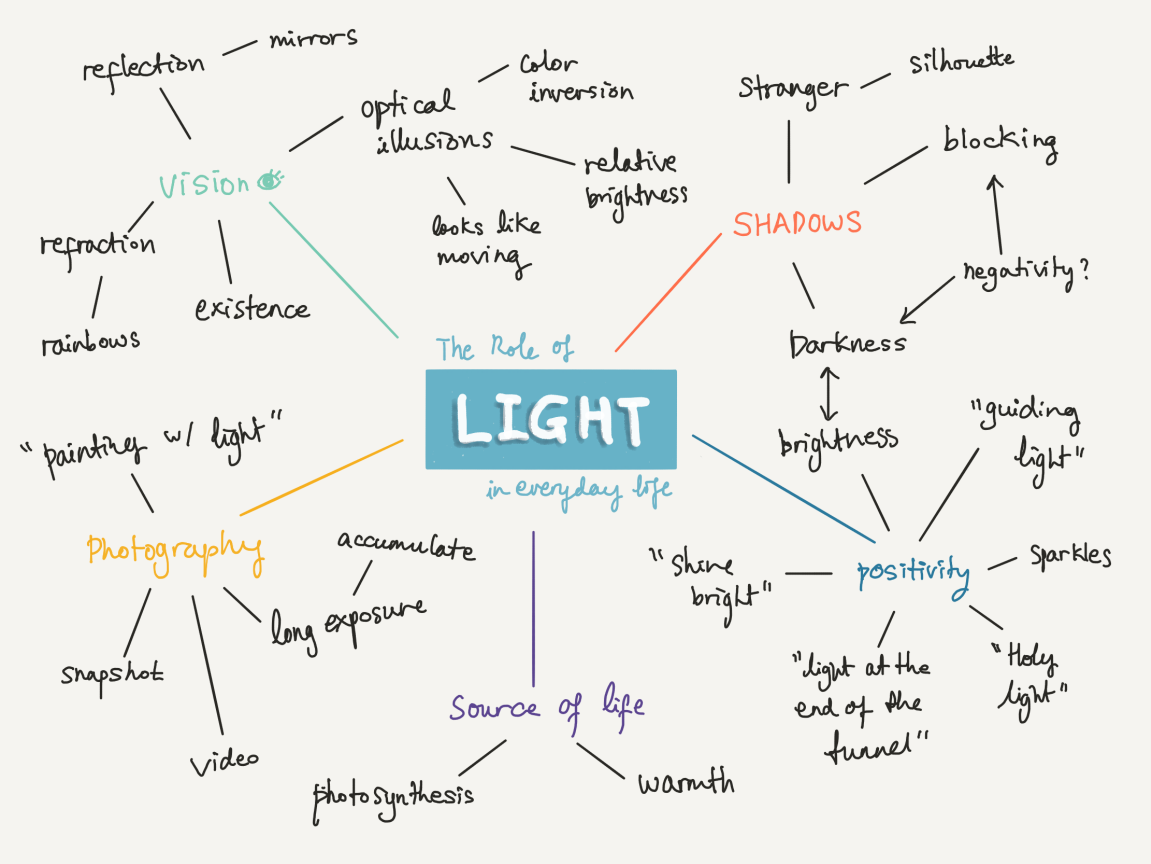

Figure 11: Mind map of "the role of light in everyday life"

Appendix B: Breadboard Layouts

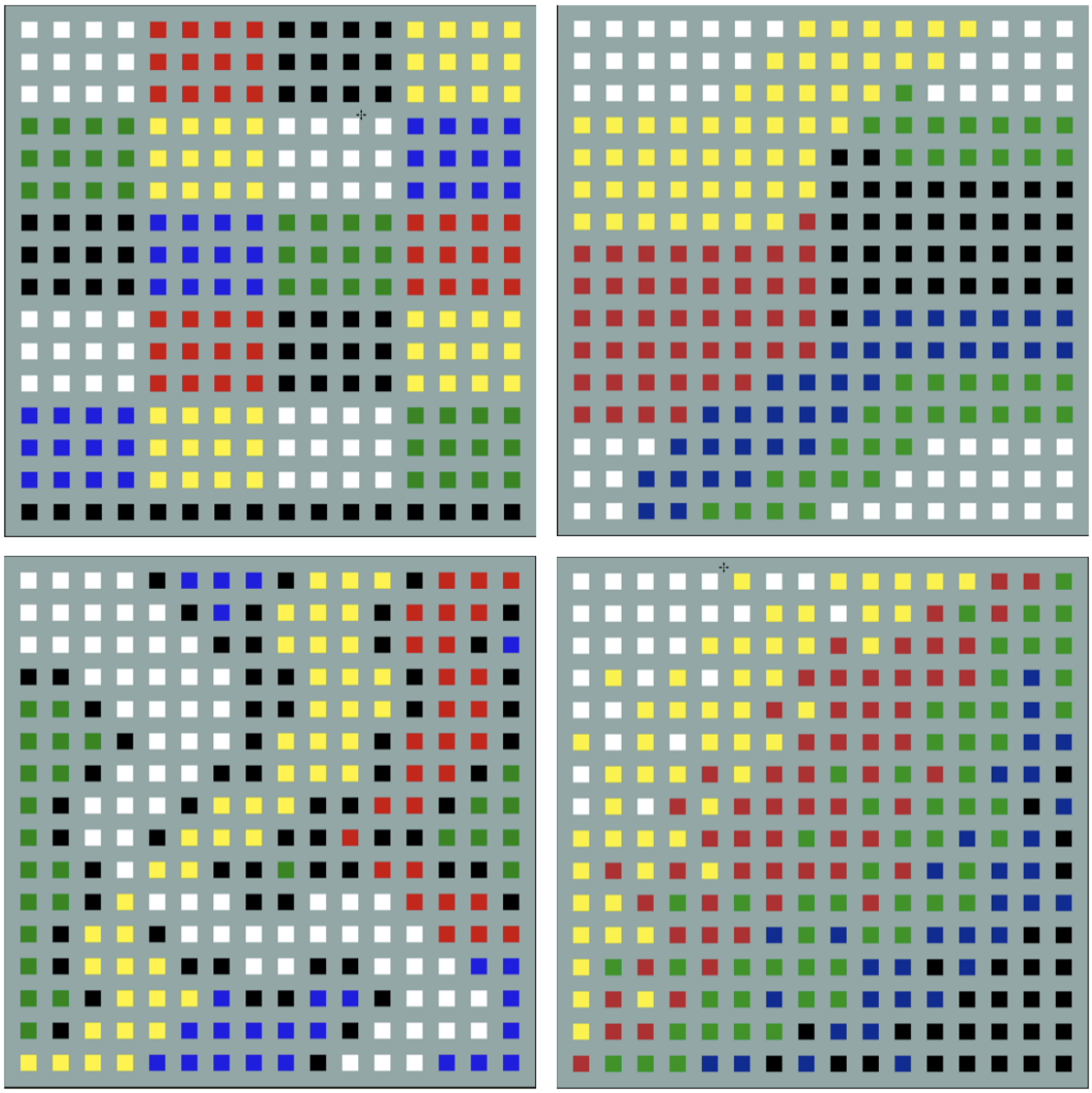

Figure 12: Explored breadboard layouts

Appendix C: Phototransistor Sensitivity

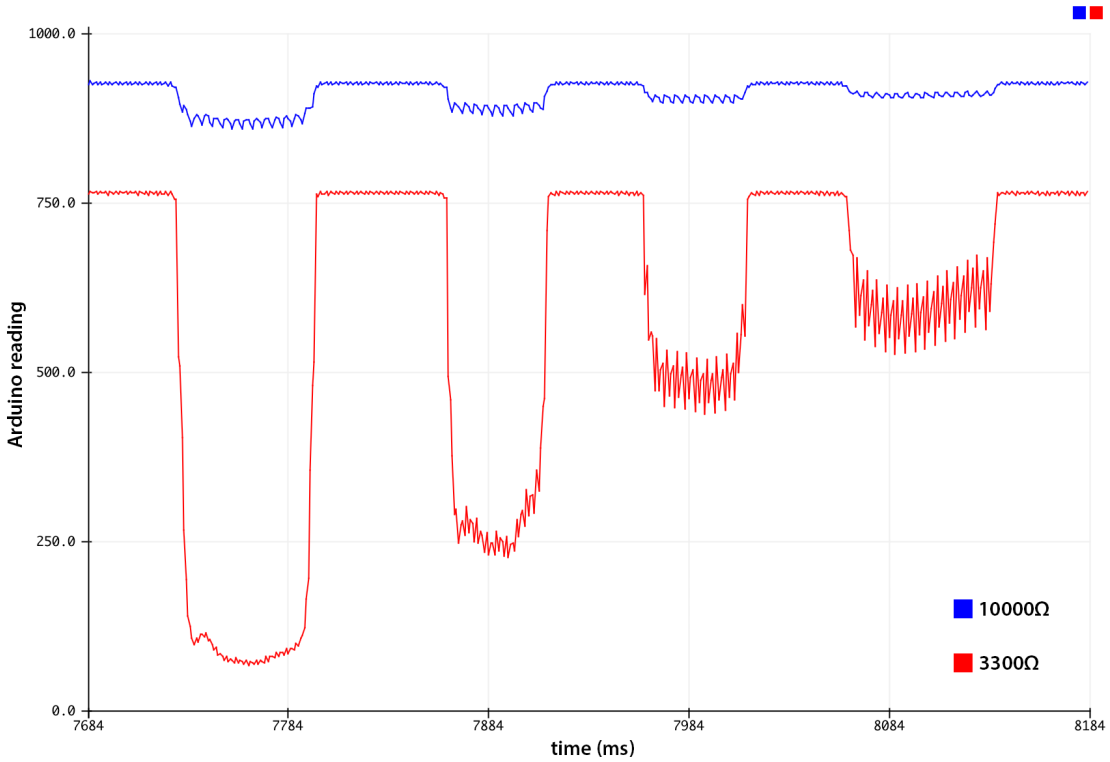

Figure 13: Phototransistor readings with blocking applied at different distances

It can be noted that the lower the resistor value, the more sensitive the readings are. According to the amount of ambient light, the most appropriate resistor level will be different.

Bibliography

Tianxing Li, Chuankai An, Zhao Tian, Andrew T. Campbell, and Xia Zhou. Human sensing using visible light communication. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, MobiCom '15, pages 331–344, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3619-2. doi: 10.1145/2789168.2790110. URL `http://doi.acm.org/10.1145/2789168.2790110`.

Caitlind R. C. Brown and Wayne Garrett. *CLOUD*, 2012. https://incandescentcloud.com/aboutcloud/ [Accessed: 2018].

Thorsten Prante, Carsten Röcker, Norbert Streitz, Richard Stenzel, Carsten Magerkurth, Daniel Van Alphen, and Daniela Plewe. Hello. wall–beyond ambient displays. In *Adjunct Proceedings of Ubicomp*, volume 2003, pages 277–278, 2003.

Mats Liljedahl, Stefan Lindberg, and Jan Berg. Digiwall: an interactive climbing wall. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 225–228. ACM, 2005.

David Birchfield, Kelly Phillips, Assegid Kidané, and David Lorig. Interactive public sound art: a case study. In *Proceedings of the 2006 conference on New interfaces for musical expression*, pages 43–48. IRCAM—Centre Pompidou, 2006.

Toni Robertson, Tim Mansfield, and Lian Loke. Designing an immersive environment for public use. In *Proceedings of the ninth conference on Participatory design: Expanding boundaries in design-Volume 1*, pages 31–40. ACM, 2006.

James Davis, Yi-Hsuan Hsieh, and Hung-Chi Lee. Humans perceive flicker artifacts at 500 hz. *Scientific reports*, 5:7861, 2015.

David Gersztenkorn and Andrew G Lee. Palinopsia revamped: a systematic review of the literature. *Survey of ophthalmology*, 60(1):1–35, 2015.

Paul Read and Mark-Paul Meyer. *Restoration of motion picture film*. Butterworth-Heinemann, 2000.

Errol R Hoffmann. A comparison of hand and foot movement times. *Ergonomics*, 34(4): 397–406, 1991.