Droplets (an Interactive Design Project)

Mariah Forbes

Senior

mariah.forbes@umontana.edu

Abstract

This project was created to be an interactive installation. Basically, it is intended to be an interaction between the user and the program, in which the user is able to "wipe away" raindrops from the cloth surface. I use programs such as Processing 3 and Arduino to achieve this, as well as an actual Arduino, sonar sensors, and a projector hooked to my laptop. The actual hardware of the product is cobbled together not-quite-professionally, and from what I had with me at the moment mixed with things I ordered from online and bought at Walmart. It is intended to engage the user and make them feel as if they could actually be wiping off raindrops. In this paper I will discuss how I came about the idea and why it seemed so novel to me, the beginning of my coding process, the struggles I had with the coding, the building of the physical platform, and how I had to compromise when building it.

Introduction:

"Droplets" was a project intended as an installation art piece. I have never done an installation piece before, much less an interactive one, and I wanted to see how it worked and if I could do it. I also liked the idea of using cloth as a part of an installation work, but it didn't end up being nearly as important as the rest of the elements of the project.

In this project, it was my idea that users would be able to use their hands to "wipe" away water from a screen without actually getting their hands wet. I liked the idea of creating a sort of "touch" screen, without actually having a lot of expensive technology to do it with. I wanted people to be able to clear it themselves, like they might do in real life. My goal was to make it feel like you were doing it, even though it wasn't really real, and may not necessarily have looked like it.

Not everything worked out exactly as I planned, but that seems to be completely normal for me. Instead, the development of my project occurred in several crucial steps.

Background:

I had no real background for this project, unless you count a little circuit board assignment in an Intro to Tech class in high school and a lab concerning resistance in college Physics. Our lessons in class actually helped me a lot when wiring up the Arduino and figuring out how it ran on the Arduino sketch.

I knew Processing from earlier projects for this class, and the Arduino sketch program was quite similar to it. Otherwise, everything was pretty new to me.

For the Arduino sketch, I had help from a website that had information about how to set up the sensors and provided code to run them and obtain information from them. I modified it so that I could use it for my own sketch, but I learned its rudiments from the online example.[1]

[1] Nedelkovski, 2015.

Process:

Inception:

I came up with the idea for the project when I was driving back to campus from home. It was raining out, and I liked the way the water drops on my windshield shone in the sun, and how clean everything behind it looked (yes, I probably should have been concentrating more on the road). Anyway, each time my windshield wipers would go over and wipe them off. No, it wasn't perfect and was sort of messy in a sloppy way, but I thought it would be cool if someone could do the same thing with their hands on an interactive board.

At first, I had an idea of using a generic image of a window as the background image and having the drops fall on that. Then I thought that might be too boring, and had an idea of having a miniature scene below the glass, so that it would look like you were looking down through the top of a tank and into it. With this idea, I thought it would be cool for the viewer to be able to see a swamp scene, and to have to clear off the glass in order to see it properly. That jumped to the idea of turning it into some sort of window into another world, where the scene would look like an alien planet, and you would have to wipe off the drops in order to see the flora and fauna there. Needless to say, I didn't get as far as moving images, and I am not sure if I could have anyway, with the way my processing sketch ended up being set up. But this was my idea at the beginning, so I naturally had to change it as I discovered what exactly I could and couldn't do.

Utility:

My original idea for the piece was quite similar to how it turned out, except I played with the thought of using Plexiglas rather than cloth, if it would project well enough onto it. However, I soon scrapped that idea when I thought I had discovered how I wanted to do my user input.

The weekend before there had been a small earthquake in the area my family lives, and almost everyone had felt it (except us, apparently. We live on some sort of separate bedrock, but people all around us felt it). I remembered that earthquake epicenters were located using triangulation of three sensors. I already kind of knew I wanted to use cloth, so I thought the perfect sensor would be some sort of spring-scale sensor. It would be able to measure the amount of force from the hand on the cloth, as would the other sensors. Then, using the numbers from all three sensors, I would ideally be able to triangulate the position of the hand on the cloth that was causing the disturbance.

Unfortunately, as I would discover later, they don't make spring scales that can input into Arduinos. So I had to look for something else.

At first when I searched for a scale for an Arduino, a sensor called a strain gauge popped up. At first I thought I might be able to do something with it, but then I realized that none of them were sensitive enough to be able to pick up the weight of a hand on cloth.

Oddly enough, this led me to flexor sensors, which the site I looked at claimed were used in some type of gaming glove. I ordered three of them, sticking with my idea of triangulation, and then looked at another way of sensing movement in case that didn't pan out either.

I found sonar sensors, which I believe to be much less finicky than their visual cousins because they use ultrasonic frequency, which cannot be often confused. Along with these, I also ordered a breadboard, wires, an Arduino, and some resistors just in case I needed them.

In the end, the sonar sensors worked beautifully and much better than I'd imagined. I had to compromise by using something smooth and clear rather than my hand, but it turned out fine. The clear plastic cup ended up being nice as it not only gave accurate placement to the sensors, but also allowed the user to see what was beneath their hand.

Problems and Solutions:

This section goes hand in hand with my process. Many of my problems shaped how I changed my project over time and how it ended up. The solutions not only changed the project, but added a different feeling to it.

One problem was the raindrops in my code. At first, I wanted them to be able to change from a mist into larger and larger droplets until they were running randomly down the screen. However, that changed when I had problems just organizing them on-screen! I had to come up with a way to "layer" them so that they filled the screen and didn't interfere with one another. For that, I developed a very specific formula:

```
for (int i = 0; i < (cnt/2); i++){

   Drops[2*i] = new Drop((d*(i+1))-((width-
(d*7/9))*floor(i/(width/d))),(d*2*floor(i/(width/d))+
d), d, d);

   Drops[(2*i)+1] = new Drop((d*i)-((width-
(d*7/9))*floor(i/(width/d)))+d/2,
((2*d)*(floor(i/(width/d)))),d, d);

  }                                                    (1)
```

This formula alternated the rows that the drops were displayed on, and affected how they were placed on the screen. It offset one line slightly from the other.

I also had a problem when it came to the droplets themselves. At first, they always followed the same formula, so the droplets near the edges would get colors from off-screen, which usually ended with drops that looked grey. I fixed that by changing where the color was picked from on the background based on what half of the screen the droplets were on. Some of the edge ones I left with dark grey because actual droplets wouldn't distinguish where they got their colors from, and would probably reflect grey, too, depending on where the "border" of the window was.

In the beginning I wanted the droplets to grow, which I could do when the drops were called individually, but not as an array. Eventually, that idea was scrapped entirely as my drops were born the size they were, and didn't start off as a light fog. The vestiges of that idea are still there, however, particularly in the drop class where you can see the way I sized the inner circles to each other to make them look more like drops. This ended up working well due to my *fullScreen()* function, which would need the drops to be resized automatically.

I had problems with the "eraser" as well. In order to erase, I needed to be able to re-draw the background on top of the droplets. The only way I was able to do that, unfortunately, was by doing it pixel by pixel, as fill doesn't have an option to reference an image. Because of that, I had to use a for loop and 1/5 the size of the height of the canvass as an iterating number in order to re-draw the background pixel by pixel within the range I wanted. This made "stars" on the background where places were "erased". I thought it turned out OK, though.

On the Arduino side, I had problems reading the data that was being sent to Processing. First, I thought the solution was to hook up two Arduinos that would be sending data simultaneously, but then I realized that their signals could interfere with each other because I had no way of controlling whether or not the code was executing. I had to put both on one Arduino sketch so that they wouldn't interfere with each other, but then I had to figure out a way to distinguish the vertical from the horizontal.

To fix this new problem, I added another thing to the *Serial.println()* of the Arduino sketch: an h before the horizontal reading, and a v before the vertical. This way, I could use processing to read an array of four and the h and v could be used as a sort of true/false to help me determine which direction the numbers belonged to. I was able to read the information, transfer it the way I wanted, and use that data as a means of user interaction with the project.

For a while, the numbers weren't taking, but I discovered I had to use *if (val != null)* before anything else in order to get it to work. After that, it worked fine.

Discoveries

I found out a lot from this project. For example, I discovered that sometimes simpler really is better. I also learned a lot about using Arduinos to get stuff onto the computer, and about finding solutions to problems such as fitting a sonar sensor onto something that would keep it above the surface of the interactive region. I also learned how to use the materials I had to make something work, and how to compromise when it came to coding and managing my time. I believe I was successful in my project, as it did exactly what I wanted to, and I was able to discover a lot about the process and about myself along the way.

Future Work

In the future, I believe I would like to add more sensors to the project, as well as add a formula for the color picker/eraser that would allow me to make a circle when the user moved their object around. Maybe I could also implement some of the ideas I had earlier into another, similar project. These ideas would definitely make it more complicated, but the end result could be much more rewarding.

Conclusions

I had a lot of fun working on this project and learned a lot about myself and the work and effort it took to put into something like this to get it completed. I believe *Droplets* was successful, and I can't wait to use what I've learned in the future, both for myself and maybe even for others' enjoyment.

References

[1] D. Nedelkovski, "Ultrasonic Sensor HC-SR04 and Arduino Tutorial," *How to Mechatronics*, HowToMechatronics.com, July 26, 2015.