Abstraction Fashion:
Seeing and Making Network Abstractions and Computational Fashions

Nicholas Gwyn Shulman

A Thesis
in
the School
of
Graduate Studies

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Arts (Individualized Program in Fine Arts) at
Concordia University
Montreal, Quebec, Canada

September 2020

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: Nicholas Gwyn Shulman

Entitled: Abstraction Fashion: Seeing and Making Network Abstractions and Computational Fashions

and submitted in partial fulfillment of the requirements for the degree of

Master of Arts (Individualized Program in Fine Arts)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final Examining Committee:

______________________________ Examiner

Aiman Hanna, PhD

______________________________ Examiner

Jason E. Lewis, MPhil

______________________________ Supervisor

Ana Cappelluto, MEd

Approved by ____

Rachel Berger, PhD, Graduate Program Director

September 15, 2020 ____

Effrosyni Diamantoudi, PhD, Interim Dean of Graduate Studies

Abstract

Abstraction Fashion:
Seeing and Making Network Abstractions and Computational Fashions

Nicholas Gwyn Shulman

Human life today is enmeshed with network organisms. What we value, the ways we talk, and the subject matter we pay attention to are all dependent on and depended upon by the networks that dominate our imagination. The internet, private social platforms, and the virtual and physical supply chains that create the hardware, software, and memetic abstractions with which we think are all examples of network organisms. Each has found a viability mechanism that permits it to survive and thrive in the present moment. Each viability mechanism creates its own unique incentives for self-perpetuation, which drive the outward appearances with which we are familiar. These incentives manifest as product forms, interface abstractions, and socially optimized beliefs and identities. To grapple with what drives the abstractions these network organisms output, this dissertation builds a worldview for seeing and making with computational networks. Computing machines are composed of abstractions, simulate abstractions, and project their abstractions onto the world. Creating in this medium requires resources that can be acquired through attention manipulation and fashion performance. The text culminates in an appendix documenting ewaste club, an art research-creation project that combines wearable cameras, supply chain inspired fashion, and disposable computers. Through a mixture of practical projects, historical analysis, and technical explanation, this dissertation proposes several new concepts linking fashion, the arts, and computation to making in the time of networks.

Acknowledgements

I would not have been able to complete this dissertation and research-creation project without the generous support of many people and institutions.

Several funding bodies provided crucial support that made this research project possible. Thank you to the Fonds de Recherche du Québec Société et Culture for supporting this project through their Multidisciplinary Master’s Research Scholarship (B1) progamme and the Bourse pour stage en milieu de pratique, which supported an internship at the Fab Lab du PEC. Thank you to the Concordia University School of Graduate Studies for awarding me the Campaign For A New Millennium Graduate Scholarship. Thank you to the Faculty of Fine Arts for awarding me the Faculty of Fine Arts Fellowship. Thank you to the Faculty of Fine Arts and the Interdisciplinary Programme for awarding me the Fine Arts Travel Award, which supported fieldwork in Austria. Thank you to Interdisciplinary Programme for awarding me the Research Travel Grant, which supported fieldwork in China. Thank you to the Textiles and Materiality Cluster at Milieux Institute of Arts, Culture and Technology at Concordia University for awarding me two Individual Grants, which supported my fieldwork in Austria and China.

Thank you to my supervisor, Professor Ana Cappelluto, to whom I am so grateful. From the time I applied to the FRQSC grant through to the final days of editing the dissertation, Ana has encouraged me to believe in myself. Ana guided me through coursework, helped me find several homes at the university, and enabled me to learn and see so much. Ana introduced me to so many of the names and places listed in these acknowledgements. Although this advising relationship was a surprise to us both, I am forever indebted to her kindness, generosity, emotional support, and artist spirit, which has been a great influence on me. When feeling low about this project, just hearing her voice would convince me to keep going. Ana, your confidence and trust helped me to develop as an artist and to find my voice as a writer. Thank you, I could never have done it without you. I am so happy to have made it through with you.

Thank you to the faculty members who have supported me throughout this research project. Thank you to my dissertation committee examiners Professor Aiman Hanna and Professor Jason Edward Lewis, who provided valuable feedback on my research writing and whose unique perspectives fortified this multidisciplinary research effort. Thank you to Professor Rachel Berger, Professor Charles Reiss, and Darlene Dubiel of the Interdisciplinary Programme for their continued encouragement and support of my work, from the start of this process through to the end. Thank you to Professor Barbara Layne, Professor Kelly Thompson, and the members of the Textiles and Materiality Cluster who welcomed me into their

home and supported my research even though I did not know the first thing about fibres when I walked in the door.

I am grateful for the support of many mentors and collaborators who taught me to see and make in new ways. Thank you to Laura Acosta, Head of Costume Shop at Concordia University's Theatre Department, for teaching me to draw, cut, and sew patterns, and to boldly make clothes and trust my hand. Thank you to Geneviève Moisan, Equipment Support Specialist at the Textiles & Materiality Cluster, for introducing me to digital embroidery and thinking with fibers. Thank you to Angela Liu for bringing me along to factory visits in China. The success and ease with which I travelled in China is thanks to you and your family. Thank you to my colleague Raphaël Demers at the Fab Lab du PEC for teaching me so much about fabrication, and even more about learning with calm fascination. Your research practice is ceaselessly inspiring. Thank you to Christopher Novello, whose vision is unparalleled, and who helped me to see so many of the concepts I discuss in the dissertation. This dissertation owes a great deal to our conversations. Thank you to Marc Beaulieu, Head of Infrastructure and Technical Support at Milieux, for supporting my research with equipment access and wonderfully positive energy.

Thanks especially to my family and friends for supporting me throughout this process. Thank you to my mother Deborah Shulman, for listening and providing important feedback, both academic and emotional. I do not think I would have finished writing this text without your support over the phone and reading drafts. Thank you to Allan Shulman, my father, for making sure I am eating right and always helping to alleviate other burdens so I can focus on my work. Thank you to Jane Shulman and Charles Shulman, my siblings, for contributing so much to my upbringing. I would not be who I am today without your presence in my life. Finally, thank you to Alice Sanz for holding my hand through to the final edit.

Table of Contents

List of Figures

List of Tables

Introduction

Network software is a part of nearly everyone's life today, yet few understand how viable software, networks, and platforms are made and maintained. Popular computing has encouraged lay people to dream in the language of software abstractions such as the app, however inexperienced onlookers often lack a nuanced understanding of the mechanisms at work underneath the sleek interfaces with which they are familiar. These weak mental models are borne out by the way people talk about software. "I have an idea for an app" says someone over a family dinner. "It's an app for finding lost objects" or "it's an app for making lunch plans with friends." While a given idea might work as a feature of an established service, on its own, the app that directly solves one person's problem is unlikely to survive long term in the wild. Why exactly is this the case?

Unsophisticated software dreamers confuse the product with the organization that creates it. Software and service interfaces, such as apps, are just the visible façade of larger structures of human labour. A given software binary is one output of a process that must be sustained over time if the software is to continue working. For instance, a reliable app that is available over a period of several years requires more than a stroke of genius and a single late-night coding session. While these actions may be enough to put the first version of the app into an app store, environmental pressures will soon make it obsolete. Apps fall out of date all the time as the software and protocols that they depend upon change. If the operating system or app store vendor makes breaking changes or adds new requirements, then the given app must be updated to stay available and continue working. Who will be responsible for making these changes, and what will motivate them to do so?

Making software that lasts is not just about making tools that solve problems or entertain. Making software that lasts, and especially popular software, is about finding a viability mechanism that will drive that software's development over time. To create reliable software that works well today and into the future, one needs to perform and reperform its development as the surroundings change. This process demands that developers and other contributors be motivated to dedicate their skills and attention to a project. Sometimes, the incentive to work on a project is financial. A company with a working business model may generate revenues and pay its developers salaries in turn. In other cases, contributors may be motivated by reputation, or the desire to see some project come to fruition. A person might contribute to an open source project, for example, out of personal interest in its outcomes or to build a portfolio that demonstrates their expertise. No matter which particular mechanism it lands upon, to perpetuate itself into the future, a software project must find a viable incentive scheme that motivates contributors and participants to engage and ensure its survival. Building these types of viability engines is a multidisciplinary activity that can involve fashion myth making, organization building, coding, and more.

We live in a time of unprecedented network communication. Of the 7.8 billion people on Earth, 5.2 billion (67%) have a mobile phone, 4.5 billion use the internet [1], [2, p. 11], [3]. Those with internet spend an average of 6 hours and 45 minutes online per day [1]. In 2019, 298 million people gained access to the internet. This represents 7% year-over-year internet access growth [1]. Smartphones and highly available wireless internet have delivered on the promise of twentieth century techno-optimism, which foresaw networked computing as a popular technology for learning, working, conversing, and so much more [4]–[6]. A message can now spread, at least technically speaking, from one person or group to nearly two thirds of the world's population in a matter of seconds. Today, personal networked computing is a truly popular technology.

Some, like entrepreneur and investor Peter Thiel, claim that technological change has slowed in recent years, and that popular internet computing is a disappointment rather than a victory. Promoting his venture capital fund Founders Fund, Thiel famously quipped that "we were promised flying cars and instead what we got was 140 characters" [7]. As others have argued, flying cars take an existing concept and imagine applying it directly in a new domain [8]. Global communications, on the other hand, stand to enable step changes in learning, research, collaboration, discovery, and invention—all of which are responsible for innovations just like the flying car. Instead of scoffing at popular and widely accessible computationally-enriched global communication networks, we might consider what positive role these media could play in ameliorating social relations.

The emergence of popular internet computing devices is especially convenient now, because the most urgent issues facing the world today are all problems of global coordination. At present, the world faces the looming threats of irreversible climate change, climate related mass migration, biodiversity collapse, nuclear war, and pandemic [9]–[13]. These current and imminent crises are tragedies of the commons that demand collaboration across industrial, national, and theological boundaries. These problems require cooperation between parties whose near term self-interests do not align with the collective good. These problems of social coordination are well suited to the social technologies unique to this moment.

However the window for spreading ideas widely is quickly closing. Never before have so many people been connected to the same network, yet the internet may soon shatter into disconnected pieces. For years, a handful of countries including China and Iran have banned American platforms that refuse to conform to domestic censorship, such as Google, Facebook, and Twitter [14]–[16]. More recently, democratic capitalist countries, once the relative bastion of free speech and global exchange, have joined the fray and begun banning platforms based on their provenance. In 2020, India banned hundreds of Chinese platforms and services, such as TikTok, WeChat, UC Browser, and PUBG [17], [18]. The US soon followed suit, threatening to ban TikTok and WeChat [19]. Although internet communication can still travel between these countries through other platforms and protocols such as e-mail, it is now more conceivable than ever that the internet may balkanize into disconnected networks.

Despite the prevalence of internet-connected software today, we live in a time of mass network illiteracy. Fast paced change at the intersection of network technology and new computational techniques far outpaces the average person's understanding of computation and its capabilities. While technology education researchers and advocates push programs for basic computer use, such as the International Computer Driving License, network computing reconfigures our relationship to the tropes of life before the internet. Data mining and machine learning research has shown, for example, that biometric data can be extracted from otherwise innocuous seeming sources [20], [21]. It is common knowledge that face identification systems are increasingly available, but how many people are aware that their fingerprints can be extracted from photos that include their hands, that the speed at which a person types their password is uniquely identifiable information, or that their walking gait can be used to identify them from 50 metres away [22]–[27]? Camera data is just one source of information that human and computer networks crunch. Ancestry.org, for instance, operates a DNA sequencing business, which it presents to customers as a genealogy exploration tool [28]. How can a customer give their DNA away with informed consent when even the company cannot know the implications such a data set will have in ten years' time? Work and leisure today increasingly implicate lay people in flows of data and network relationships of which they are completely unaware. This data, in turn, is used to modify human behaviour. Beyond basic computer literacy, people must urgently acquire the knowledge to navigate these unprecedented circumstances.

For the scale of their influence, there is also surprisingly little pop cultural mythology about network making itself. We all know who made Facebook, but do we know *how* he made Facebook, beyond the single reductive interpretation provided by Aaron Sorkin? What percentage of Google users know who Larry or Sergey are, or that their project started as graduate research? Discontent about advertising business models and surveillance have grown in recent years, but non-professionals lack the context to imagine viable alternatives [29]. It is fair to criticize existing software and networks for the externalities their incentive schemes drive them to wreak, but who is equipped to invent new alternatives? Although schools teach the trades required to keep networks running, such as programming, project management, and marketing, no popular scholastic discipline teaches students to think across the various viability mechanisms that power the networks that mediate our thinking.

To imagine new types of network viability mechanisms, more people must enter the conversation and become network literate. We live in a moment of incredible technological means, and yet we lack the slang, humour, and culture required to profit from these inventions because so few are able to engage them directly. More people must participate in the conversation if we are to invent social systems that mitigate needless suffering and provide paths to dignified living for all. We do not have the language to think of the network incentive schemes we need because too few people are able to participate in the conversation. A meaningful network discourse is one in which conversant parties are equipped to do more than just complain. A more informed public is more likely to imagine the vocabulary we currently lack, and invent new network incentives with more equitable ends.

Network computation is a matter of literacy because it mediates access to all other subjects. Compared to other parts of modern life, I believe that network computation is the most similar to reading, writing, and basic numeracy, because networks media are gateways to all other discourses. While it would be great if more people understood chemistry, plumbing, or household electricity, the means for learning each of these disciplines is transmitted through communication networks. These networks are not neutral pipes; the viability mechanisms that perpetuate these networks drive their interface designs and algorithmic recommendations. Learning and comprehension today are bound up with network computation, and so it is important to understand these networks to critically assess their outputs.

Talking about networks and software can be confusing, because these words are understood differently in different disciplines. To some, software is only binary code running on a digital computing machine. To others, any system of procedural instructions through which one steps is a kind of computer. Networks, too, have a variety of meanings. In sociology a network may describe a set of connected people, while in information technology it might mean a protocol for communication and the hardware infrastructure that speaks that language. In startup and internet circles, a social network means a software service for connecting, while in supply chains, one might have a network of suppliers.

In this thesis, I introduce a meta-category that I call *network organisms*. Network organism is a term I have chosen to group the various types of centralized network institutions, decentralized protocols, software packages, and the networks of people who connect through them. I use the term network organisms because it brings together various types of entities that are typically separated by disciplinary boundaries, and brings new focus to their naturally selecting quest for survival through a variety of viability mechanisms [30].

As I see it, there are a handful of network organisms that all rely upon one another. First, there are companies, which are profit-driven entities that sell attention and consumption of services and products in exchange for revenues, which they use to pay employees in exchange for their labour, such as computer programming. Companies can be privately held or publicly listed, and they can make money through a variety of business models. At present, these companies are often described as platforms, which can mean either that their services are open to extension by other actors in the ecosystem, or that they are a platform for connecting users or distributing content, which does not imply this extensibility. Facebook, Microsoft, and Apple are all examples of network organisms with a company model. Second, there are protocols, which are standards and syntaxes for communication. Protocols can be centrally organized, and governed by a non-profit body, or decentralized, and ruled by some mechanism of voting or consensus based on participation. TCP/IP, HTTP, and Bitcoin are examples of protocols. Third, there are open source projects, which gather the labour of volunteer and paid contributors to create code that others can use according to the given project's license, which often means the freedom to use and modify the code for free. Linux and the Wiki platform are examples of such projects. Fourth, there are the networks of people who make use of the private platforms and products, protocols, and open source software. This last category is distinct from the institutions that make the software. The network of people who communicate through a protocol or platform are what make it uniquely valuable, and impossible to replicate simply by reproducing the software or standard in use. These networks of people give the software and the institutions that produce or enable them value. So for instance, in the case of Facebook, there is Facebook the corporation, Facebook the software binary running on a given machine in a given moment, and the Facebook network of users that makes the rest meaningful.

The concept of a network organism draws attention to the various viability mechanisms that perpetuate software, institutions, and network participants into the future. No matter whether a network organism is centralized or decentralized, corporate or non-profit, virtual or implicated in the physical supply chain, it must find a way to survive, much like a biological organism in an ecological environment. As we shall see, each network organism's viability mechanism is the primary factor influencing their behavior and design over time.

In 2013, Jack Ma, the co-founder of Alibaba, delivered a speech to the Stanford Graduate School of business entitled “Technology and Ideas Can Change the World” [31]. In the final moments of his speech, Ma explained why his company bought Yahoo! China from its American parent. “Once you buy a company, it becomes a part of your own growth. And we grow bigger and bigger.” Ma raised his hand, with fingers pinched, to his mouth. “We just ate Yahoo and digested it.” He spread his arms like wings, as if the acquired company’s life force spread through his body. Ma pantomimed the corporate merger: the body of Alibaba consuming the flesh of Yahoo! China, absorbing its energy, and growing stronger. In the corporate merger, the acquirer integrates the acquiree’s people, their knowledge, processes, and relationships.

The goal of this dissertation is to help you, the reader, to swallow and digest a carefully prepared body of knowledge about network computation, so that you, too, may grow stronger and more powerful. Just like Alibaba swallowed Yahoo! China and became able to achieve feats neither company could accomplish on their own, it is my hope that this text empowers you to take a more comprehensive view of the network computational media that shape our understanding of the world and ourselves.

This research-creation dissertation is driven by several related research questions. How do technology, media, and abstraction relate? How are technologies extensions of language? Are new interfaces designed or discovered? How do computer and human labor relate in the context of an interface? How does symbiosis with network computation feel? How does one summon a network or grow an ecosystem? How do novel interfaces and product forms become socially accepted? How do networks change the meaning of art and design practices? Is network making an art form?

This dissertation is divided into five chapters and an appendix, each of which approaches the subject of network computing from a unique perspective. The first five chapters provide essential theoretical background information for understanding computation, networks, and their role in life. The appendix documents a series of research-creation experiments, which I undertook simultaneous to the writing process. Each of the five chapters and appendix points at the subject of network computation from a complimentary perspective. Each presents a facet of the same subject. Each is an analogy for the rest.

Chapter 1: Abstraction Machines explores the relationship between computing and abstraction. This is the longest and most technical chapter, which lays the groundwork for thinking about computation as a medium of abstraction. The first two sections of this chapter discuss the role of abstraction in computing hardware and software. The third section addresses the abstracting effect software has upon the world beyond the machine. The fourth section proposes that software and money are interconnected systems of abstraction.

Chapter 2: Algorithm Theatre delves into the performative aspects of software making. Rather than writing and running, this chapter argues that software is a multidisciplinary performance in many media at once, and over time.

Chapter 3: Fashion & Attention deals with the art of grabbing attention and framing software and network initiatives. The first section of this chapter contends that networks are memes and memes are networks. The second section argues that social media and premium entertainment media are driven by the same attention-optimizing incentive, and that they are differentiated only by the origin and production style of the media objects they serve.

Chapter 4: Booting Networks deals head on with the subject of networks. Section one describes three theories of network effects, which are terminology one comes across often in this field. The second section contends that many successful networks begin by enabling socially transgressive behaviour.

Chapter 5: Supply Chain deals with the relationship between manufacturing, the supply chain, and the making of meaning. The first section argues that Apple's legacy of sleek hardware is marred by their financial motivation to sell disposable devices. The second section advances the perspective that people are often attracted to disposable goods because mass produced products function like parts of speech in the meaning making practice of life

Appendix 1: Research-Creation documents a few years' worth of practical experiments at the intersection of cameras and fashion. I chose to design new interfaces to network computation on the basis of three premises.

First, new communication technologies change cultural norms in ways that cannot be predicted before their popularization. It is impossible to precisely predict how technologies will be adopted, because the hardware roadmap alone does not indicate how people will use and abuse a given bundle of scientific discoveries and human labour. For this reason, I chose to get my hands dirty and reflect upon what I discovered myself, in the hopes of finding something new.

Second, applications of new technologies are nevertheless grounded in the behaviours, tastes, and norms that precede their popularization. Successful applications provide comprehensible interfaces to unfamiliar technologies. To be legible, a software or hardware design must be visible to the present day while catalyzing the cultural change that invents tomorrow. With this idea in mind, I set out to use the memes around me to imagine new product forms and experimental directions.

Third, novel technology packages are popularized through collaboration between their creators, their early users, and the environment. A product or technology package may correctly foresee the inevitability of a new category of human behavior, but its cultural positioning can still fail to generate sufficient positive social interest to reach long term viability. To become parts of life, new product forms must build enough memetic momentum to become culturally relevant and socially acceptable. With this in mind, I conceived of my projects as *network* experiments. The work would have to be a collaboration between me, the informal group of early users who play with my prototypes, and the unanticipated influences I encounter in my environment upon handling my own creations.

To test these conceptual premises firsthand, I challenged myself to design and iterate new network interfaces, in both hardware and software, and to document this experience in qualitative autoethnographic style. Although my experimentation process employed engineering materials, such as electronics components and 3D printing, my approach more closely resembled an art practice than engineering optimization. To avoid hewing too much to tradition, my project aimed to humbly find meaning through prototyping, rather than chase prematurely defined functional goals.

The appendix documents this research-creation process. I discuss the various experiments I pursued, and I also describe the unanticipated prominent influence of fashion in the research work.

In sum, this thesis is intended to be an unconventional primer for approaching network computation from an active stance, in theory and practice.

Note on "Computer Science"

In the academic world, the study of computers has found a home under the banner of "computer science." Computer experts with post-secondary degrees in the discipline refer to themselves as computer scientists.[1] But is the study of computers really a science?

Within computer science, there is a tradition of criticizing the field's name for exaggerating the discipline's scientific legitimacy. The concern that computer science is not a science is typified by a joke that comes up often in computing circles. The earliest documented form of this joke that I have found occurs in *An Introduction to General Systems Thinking* (1975), where author Gerald M. Weinberg paraphrases mathematician Frank Harary. Weinberg writes that "any field that had the word 'science' in its name was guaranteed thereby not to be a science. [Harary] would cite as examples Military Science, Library Science, Political Science, Homemaking Science, Social Science, and Computer Science" [32]. Variations of this quip have been attributed to Max Goldstein, Richard Feynman, and others [33], [34, p. 215].No matter the critique's source, there is a recognizable unease within the discipline about the scientific grounding for computer science.

In the twentieth century, early in the discipline's formal history, some argued that computers could not be a science because computers are a human invention, whereas the sciences study natural, physical things. Over time, this argument has lost favor. In 1948, Claude Shannon proposed his theory of information, which has become one of the scientific pillars of computer science and one of the most important scientific discoveries of the last century. Although Shannon discovered information theory through his work on binary logic and telephone networking, his mathematical formulae for understanding the encoding and communication of information over noisy communication lines has been observed in domains that far predate and exceed humanity's domain. As former Association for Computing Machines (ACM) president Peter Denning pointed out in a 2007 interview with Ubiquity, geneticists and physicists have observed that information theory has explanatory power for describing the most beguiling aspects of the universe, like the mechanics of DNA in organic life and the formation of stellar bodies like black holes [35]. The study of computation has led to scientific discoveries that transcend the mechanical and

---

[1] There are also academic programs in computer engineering. Experts sometimes refer to themselves as software engineers. This may be more à propos for those who learn to program from people holding PhDs in engineering, but the term engineer is used widely and is not restricted to such graduates. Some practitioners call themselves programmers and developers.

electrical machines with which computation has been associated over the past two hundred and fifty years. Information theory alone proves that the study of computation can lead to scientific discoveries with broad impact. However the presence of some scientific research within the computer discipline does not rescue the whole of the discipline from critique.

In 2012, Internet protocol co-inventor and then-President of the ACM Vinton G. Cerf published a letter entitled “Where Is the Science In Computer Science?” [36]. In this letter, Cerf argues that “in the physical world, science is largely about models, measurement, predictions, and validation”. To Cerf, any science should be able to model and make predictions about the behavior of hardware and software. This definition is in line with the mainstream philosophical definition of science, forged originally by Karl Popper. For Popper, the key to scientific inquiry is proposing and testing falsifiable hypotheses. A theory without a test that can prove it wrong is not scientific inquiry according to Popper. According to this line of reasoning, while information theory is a scientific pursuit, much of the rest of computer science is not science at all.

Cerf acknowledges that computer practitioners can effectively model, measure, and predict *hardware* performance, but the field is far from scientific when it comes to *software*. There are scientific and mathematical descriptions of the efficiency of search algorithms, for instance, but when it comes to writing and evaluating code, no such mathematical measures exist. Cerf notes that there is no rigorous way to predict how many bugs are in a given piece of code, to predict how long it will take to fix them, to measure the code’s security, or the evaluate probability of it being exploited.

Computation philosopher Ted Nelson has argued throughout his texts, speeches, and videos that code is not scientific, but historical and political. First, as an educational discipline, mainstream computer science teaches most students that popular paradigms in computing, such as the directory-file abstraction popularized by UNIX, are natural, logical, and inevitable [37]. In fact, this structure is just one of many ways of organizing information inside of a computer, and it has become popular for historical reasons rather than its technical superiority or scientific inevitability. Furthermore, Nelson points out that programming is not, itself, deterministic. Given a coding problem, a hundred engineers would return dozens of solutions [38]. Anyone who has programmed knows that, while there are more and less efficient ways to write an algorithm in a given language, the finer points of which approach is optimal is subjective, not scientific. A style that is best in one’s person eyes may be arcane to someone from a different tradition. Which language and programming approach is best depends upon underlying infrastructure that is subject to change, such as compiler or runtime optimizations. Code legibility is subjective, too. For all these reasons, writing software is not primarily scientific thus far. While the underlying mechanisms of computation may be scientific, actual programming bears a closer resemblance to work flowing than mathematical theorems or organic chemistry.

In the scope of human history, formal computation is a recent discovery, the implications and potential of which have only just begun to be explored. Computation is too important and far reaching to be cordoned off and reserved for the minority who feel at home within computer science culture. For these reasons, I attempt to speak to a broader audience that includes people who identify with computer science, as well as others who are made uncomfortable by that epistemological regime.

In this text, I avoid invoking the phrase computer science where possible, but I fall back upon it whenever readability would otherwise suffer. I am not principally concerned about whether or not computation is a science. I perceive aspects of computation where the scientific method and falsification can be applied, and many others where subjectivity dominates. As Denning notes in the aforementioned interview, “Computation is the principle, the computer is simply the tool” [35]. I am less interested in computer science, and more interested in the social, philosophical, and artistic potential of *computation*. My research-creation concerns are focused at the point where the science stops, and the clever and observant performance artist can begin their work.

Chapter 1: Abstraction Machines

In 1842, Ada Lovelace realized that computers are not number crunching calculators, they are abstraction machines.[2] Lovelace, an English countess and mathematician, was perhaps the first person to put into words the peculiar power of a machine that works upon abstractions. In her now famous notes on Charles Babbage's Analytical Engine, Lovelace described her realization upon reviewing the first design for a general-purpose computer:

> *Many persons who are not conversant with mathematical studies imagine that because the business of [Babbage's Analytical Engine] is to give its results in numerical notation, the nature of its processes must consequently be arithmetical and numerical, rather than algebraical and analytical. This is an error. The engine can arrange and combine its numerical quantities exactly as if they were letters or any other general symbols*
>
> …
>
> *[The Analytical Engine] might act upon other things besides number, were objects found whose mutual fundamental relations could be expressed by those of the abstract science of operations, and which should be also susceptible of adaptations to the action of the operating notation and mechanism of the engine. Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent* [40, p. 33].

Lovelace is remembered as the first software programmer because she understood that computing machines can simulate any system of related symbols, no matter what they represent. Although machines like the Analytical Engine, and even modern computers, appear to operate on relations between numerical values, these numbers are merely the mechanism of the machine's operation, and not its final product. Lovelace recognized that computation can be performed on anything that can be expressed as a mutually related set of symbols, from musical notes to poetic forms, and beyond.

---

[2] The term "abstract machine" would later be used in the field of computer science to describe a hypothetical machine to test the computability of a problem. The most famous abstract machine in computer science is the Turing Machine. "Abstract machine" was also used by Gilles Deleuze to describe the power of diagrams to affect thinking [39, p. 32]. In this chapter, I discuss four ways in which computers are abstract machines.

This chapter is dedicated to a simple contention: *computers are abstraction machines, inside and out.* In the pages that follow I will discuss four ways the concept of abstraction helps us understand the cause and effect of computation.

The first two sections establish a common grounding: how are computers made up of abstractions, and how does that make them different from prior machines. The latter two sections examine how computation affects perception of the world, and how this in turn affects computation.

The first section of this chapter is about the physical abstractions that make up a computer. It sketches how chemicals can be turned into hardware devices, and how those hardware devices can be made into a computing machine. In this portion of the text, I attempt to recapitulate the essential logic of computing hardware. This section physically grounds the rest of the text and shows how important abstraction is to building computers.

The second section is about software, or the simulation of systems of symbols. This section reviews the special property that distinguishes computer hardware from other machines. What does it mean to work in a virtual medium? How has Turing's vision of computation framed the subject of software? This section describes the widely accepted formal definitions of programming that are popular amongst computer people.

In the third section, I turn my attention from the abstraction inside of computing machines to the abstraction they perform outside. Section three is an original reading of network interfaces as ideological lenses. In this section, I argue that computers are not only composed of abstractions, but that computers also make abstract the world beyond their packaging.

The fourth and final section contends that even the most abstract forms of computing, such as social media, can be seen as programming interfaces, too. In this section, I argue that locked-down interfaces intended to inhibit end-users from programming remain open to symbolic manipulation. Despite popular formal definitions of programming and computation, even very high level interface abstractions can be considered programmable.

Computation terminology and ideas are now, more than ever, bleeding into the popular consciousness. The transition to computer consciousness is only beginning to be widely recognized now that its interfaces have burst out of the infrastructural and subcultural domains and into the living rooms, cars, pants pockets, purses, and ear canals of nearly everyone.

The purpose of this chapter is to examine four ways computers and abstraction relate.

Section 1: Computers are Made of Abstractions.

From a distance, computing machines appear too complicated to comprehend. Even pedestrian computing devices today contain intricate circuity and countless components at a physical scale that is difficult to imagine and impossible to see.[3] How is it possible for humans to juggle the interconnected architectures of these machines in their heads?

To understand how a computer is built, it helps to break the machine down into functional parts. The most common way to divide up the parts in a computer is to distill each to its ideal form, which is called an abstraction. In the context of computers, an abstraction describes the essential properties of a component, regardless of the specific brand or manufacturing techniques used to create any particular instance of that part [41, p. 3]. In computers, abstractions describe what an entity does, without the messy details of how it is achieved. These abstractions conceal the complexity inside, and reveal only what is necessary to make use of the entity.

Traditionally, computer abstractions are conceived of as a belonging to a vertical hierarchy [41, p. 3]. The most essential, simplest parts are at the "bottom" of the hierarchy of abstraction. These are the "low-level" abstractions, like transistors. By combining low level abstractions together, we can build complex configurations that yield new behaviors. These more sophisticated configurations of simple parts can be grouped together into a new, "higher-level" abstraction. This hierarchical structure is an important conceptual tool that makes it possible for humans to keep the design straight in our minds.

---

[3] Search for images of "MEMS SEM," or microelectromechanical systems scanning electron microscopy, for a glimpse of the tiny machines at work in all our devices. These miniscule systems are smaller than a human hair is wide. Their applications are numerous: they push the ink in inkjet printers, measure the orientation and movement of phones, and angle the equally small matrix of mirrors in DLP projectors. Ghostly images of these vanishingly small mechanisms remind us that alien technology is here, and we are the space lifeform responsible for its invention and manufacture.

Table 1 Layers of abstraction in computing machines.[4]

	Software and Beyond	Social Media	Applications (Web Apps)
		High Level Languages	Applications (Native Apps, Browsers, Games)
“High Level”		Operating System	
		Compiler	
		Virtual Machine	
		Assembler	
	Hardware	Machine Language	
		Computer Architecture (Von Neumann)	
		ALU	Memory Elements
		Boolean Arithmetic	Sequential Logic
		Boolean Logic Gates	
		Transistor	
“Low Level”	Material Processes	Material Preparation (Silicon Doping)	
		Material Processing (Refining)	
		Material Extraction (Mining)	

From Chemistry to Hardware

To get a grasp on this concept of abstraction, let us imagine building a computer system from “bottom” to “top.” In the subsequent paragraphs I will briefly describe how chemicals can be combined to create hardware, which can be combined to run software.[5] This journey from atomic ingredients to programmable computers will demonstrate that abstraction is a crucial action in the physical and conceptual construction of computing machines.[6] To begin, let us consider the modest switch.

---

[4] Computing machines are composed of layers of abstraction [41, p. xii].Material processes (bottom) and social layers (top) are not usually included in computational abstraction stack, but they are relevant to hardware and software architecture thinking today.

[5] N.B. Computers are made of opinionated design abstractions within a larger field of possibility. In this section, I will describe common computer system components. For every assertion that I make, there are alternate designs that deviate from the descriptions given. Rather than hedge every declarative sentence, I hereby disclaim that the computer abstractions I define are in no way the only or the best approaches to making computation machines.

[6] N.B. Although they are listed above, I will not discuss material sourcing and several intermediate layers of abstraction because they are beyond the scope of this project. They remain fascinating and fruitful areas for further research and they are absolutely relevant to the abstract character of computation.

A regular mechanical switch, like the common light-switch, works by connecting or breaking an electrical circuit.[7] When a light switch is in its *off* position, the circuit is broken and electrons cannot move through the light bulb.[8] When the switch is in its *on* position, the circuit is complete and electrons can travel back and forth through the light bulb, causing it to light up.

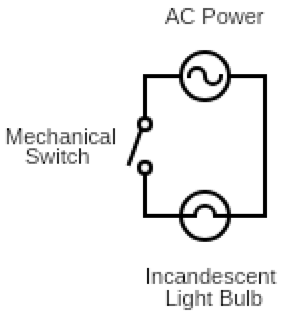


Figure 1 Circuit diagram of an alternating current mechanical light switch.

A transistor is an electronic switch with *no moving parts*.[9] Like the mechanical switch, in its *on* state, the circuit is complete, and in its *off* state the circuit is broken. As opposed to the mechanical switch that we use to control the lights, a transistor performs no physical action to break or make the connection in the circuit. Instead, transistors can be switched on or off with electrical current alone.

Silicon & Doping

To make a transistor, we have to start with chemistry. Transistors are made of positively and negatively charged materials, called P-type and N-type for short. In chemistry, charge describes the quantity of electrons in a given material. Electrons are negatively charged, so the material with *more* electrons is negatively charged, and the material with *fewer* electrons is positively charged.

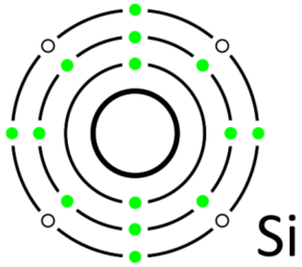


Figure 2 Bohr model of a neutrally charged silicon atom.

Both the P-type and N-type materials are composed of at least two ingredients each. The first ingredient is a semiconductor, such as silicon or germanium, which exhibit an electrical conductive property in-between conductors, such as copper, and resistors, such as glass. To understand better how Silicon Valley got its name, and why it is dubbed the semiconductor industry, let us take a closer look at silicon.

---

[7] This example is sure to age poorly, as internet-connected light switches replace the type I describe above.

[8] In the light switch example, the circuit draws its energy directly from the municipal electrical grid. This circuit generates light in the bulb by alternating current. The generators that power the grid push and pull electrons fifty or sixty times a second. In contrast, in the transistor examples, and all other computer examples hereafter, the electrical power is direct current, meaning that the electrons travel in only one direction. Alternating current is more efficient for transmitting power over long distances, but direct current is more easily stored in batteries.

[9] A transistor is also an amplifier, but amplifiers are not germane to my argument, so I will leave that to the side.

Silicon is the 14th element in the periodic table. Neutral atoms of silicon have 14 electrons. If we imagine an atom of silicon according to the electron shell model, its first and second electron shells are filled with ten electrons, and the atom's remaining four electrons are in its third electron shell, which can hold a total of eight electrons.[10] As a result, silicon atoms create covalent bonds with each other. Each silicon atom shares its four valence electrons with four neighbouring silicon atoms, creating a neutrally charged covalently bonded structure.

To create P-type and N-type materials, engineers mix silicon or another semiconductor with atoms of another element, called the dopant, which introduce a charge to the compound [42]. To make N-type material, the semiconducting silicon is doped with atoms of elements that have five valence electrons, or five electrons in their outermost electron shell such as phosphorous and arsenic. To create P-type materials, the silicon is doped with atoms of elements with only three valence electrons, such as boron or aluminium. When mixed together, the silicon and the dopant form bonds with one another. In N-type materials (negatively charged), the silicon forms bonds with the pentavalent atoms (those with five valence electrons), such as phosphorous. The silicon atoms have only four empty spaces for electrons in their outermost electron shell, so the fifth electron of the dopant atoms does not bond with any atoms. These unbonded electrons give the material a negative charge. For P-type materials, the opposite is true, and the holes left in the bonds between the semiconductor and trivalent elementary atoms make the P-type material positively charged.

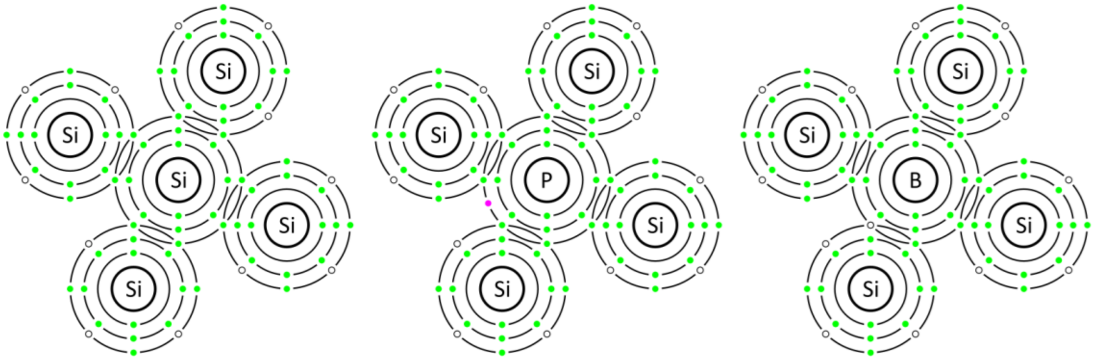


Figure 3 From left to right: pure silicon, silicon doped with phosphorous, and silicon doped with boron.[11]

---

[10] In the diagram at right, the central circle is the nucleus, which contains the protons and neutrons. Electrons are represented as green circles, while empty spaces in the valence shell are depicted as black outlined circles.

[11] These figures show pure and doped silicon. Pure silicon (left) creates covalent bonds with its neighboring atoms to create a neutrally charged structure. To create N-type, or negatively charged semiconductor (middle),

Diodes

When placed side-by-side, P-type and N-type materials form P-N junctions, the smallest unit of abstraction for making transistors. P-N junctions are also called diodes. When P-type and N-type materials are placed side by side, the free floating electrons in the negatively charged material travel into the electron holes in the positively charged material (see figure below). This occurs primarily at the boundary – also called the interface – between the two materials. When the negative charges move from the N-type to the P-type material, they create a region of slightly negative charge in the P-type material, and a slightly positively charge in the N-type material. These charged areas at the interface between the materials form a boundary that prevents further movement of electrons from one material to the other. This electrically charged boundary is called the depletion layer.

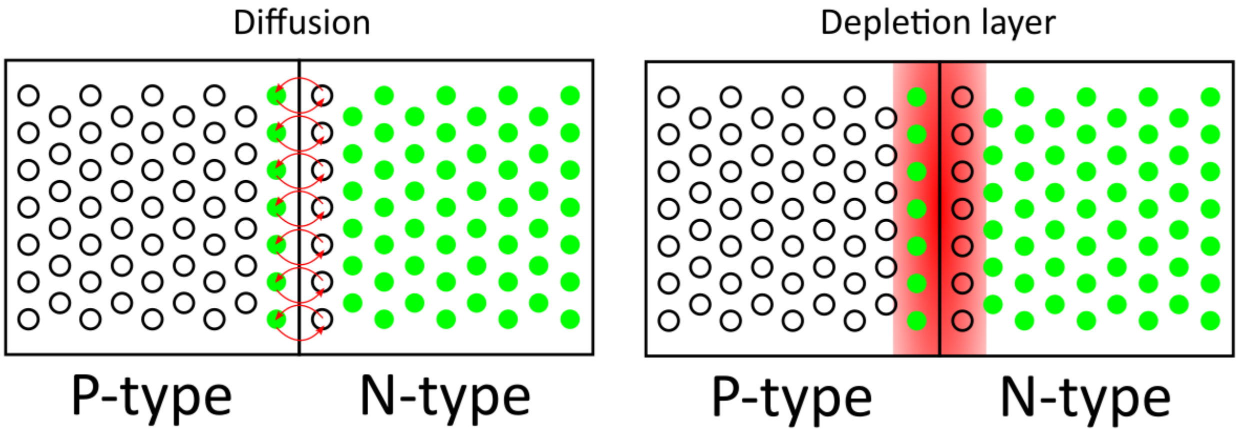


Figure 4 In P-N junctions, electrons move from the negative substrate to the positive substrate, creating a depletion layer.

Electrical current can only flow in one direction through a P-N junction. This property is what makes transistors possible! But how does it work? In a P-N junction, electrons can only travel across the depletion layer when an energy source is connected in one of the two possible orientations [42]. When a power supply's positive terminal is connected to the P-type material, and the negative terminal is connected to the N-type material, the electrons in the N-type will be attracted to the positive terminal, and the holes in the P-type will be attracted to the negative terminal (see figure below, left). This expands the depletion zone and no electrons cross the junction. If instead, the power source is connected in the opposite

---

phosphorous (element 15) can be introduced to the silicon in a process called doping. Each atom of phosphorous (element 15, middle) introduced into the silicon adds a single free floating electron, shown in purple. To create P-type, or positively charged semiconductor (right), silicon can be doped with boron (element 13). Boron has only three valence electrons. Each atom of boron creates a hole.

orientation, the electrons in the P-type material will be repelled by the power source's negative terminal, and the electrons in the N-type material will be attracted by the power source's positive terminal (see figure below, right). If the power source has sufficient voltage, electrons will be pushed and pulled across the depletion layer. This directional property of P-N junctions is called forward-bias and it is the reason that diodes and other forward-biased electronic components will only conduct electrons in one orientation. But how does this help us make a transistor?

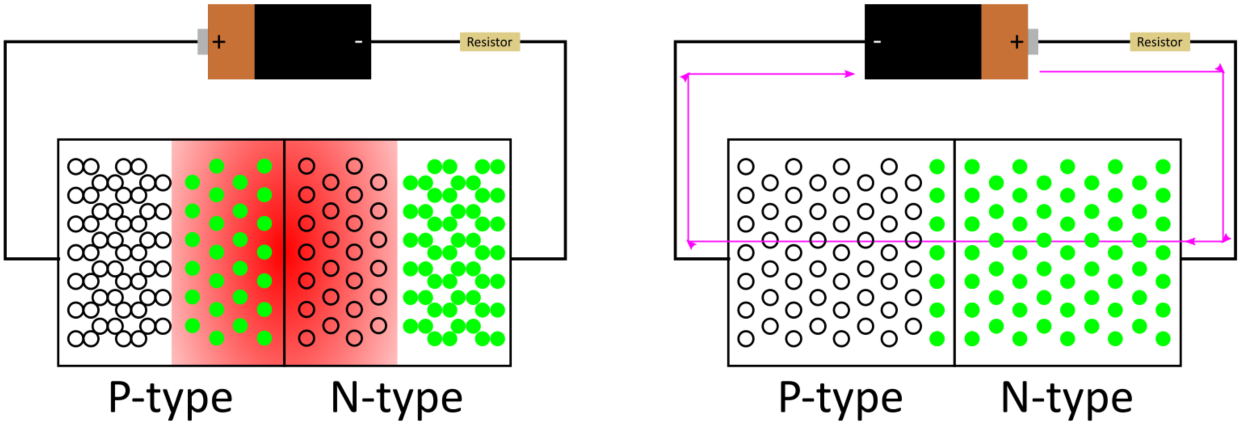


Figure 5 Electrons will only move through a P-N junction in one orientation, called the forward bias.

Transistors

A transistor is composed of a pair of P-N junctions. One common type of transistor design is the Bipolar Junction Transistor. The BJT is composed of two P-N junctions, sandwiched together into P-N-P or N-P-N formations. When a power source is connected to either extremity (called the Collector and Emitter) of a BJT transistor, the back-to-back depletion zones formed by the sandwich of two P-N junctions forbids the flow of electrons, no matter which orientation the power source is attached (see figure below, left).

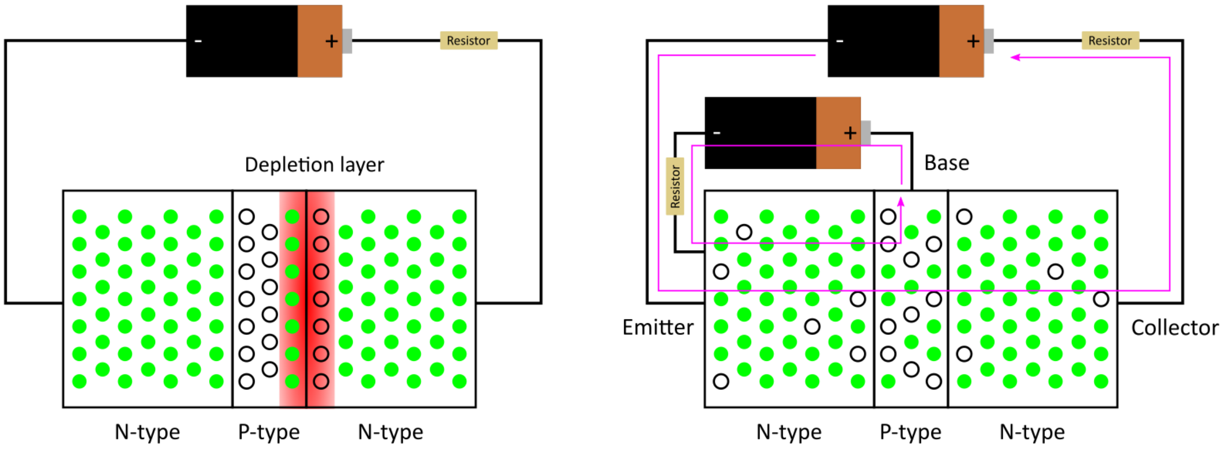


Figure 6 NPN sandwich (left) and N-P junction (right).[12]

However, if we connect another power source to only one of the P-N junctions in the forward-biased direction, this power source will push and pull electrons across the depletion layer, allowing current to flow from one extremity of the transistor to the other (see figure above, right). By applying a small voltage to just one of the P-N junctions, we allow current to flow from the collector through to the emitter. The conductive lead to this forward-biased P-N junction is called the *base* or the *gate*.[13] When a current is applied to the base, electrons from a power source connected to the collector are able to move all the way through the N-P-N (or P-N-P) sandwich. We now have all the necessary ingredients to make a transistor: a switch with no moving parts!

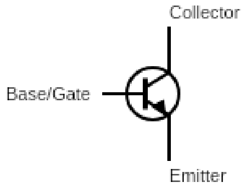


Figure 7 Electrical circuit diagram symbol for a BJT NPN transistor.

Figure 8 Electrical circuit symbol for a mechanical switch.

The transistor, or solid state switch, is a bit different from the mechanical light switch discussed above. When we flip a mechanical switch *off*, we break the circuit—we introduce a gap in the conductive wire connecting the lightbulb to the positive and negative leads in the wall outlet. Flipping the switch *on* completes the circuit, allowing current to flow through the bulb.

---

[12] No matter which orientation the battery is connected to the extremities of the NPN sandwich (left), its depletion layers will block electrons from passing through the circuit. In a BJT transistor, a second, smaller voltage is connected in the forward-biased orientation to one of the N-P junction (right). This smaller power source draws electrons across the N-P junction boundary. A small Base current enables a larger Collector current.

[13] I will refer to it as the *base* to avoid confusion with logic gates, discussed below.

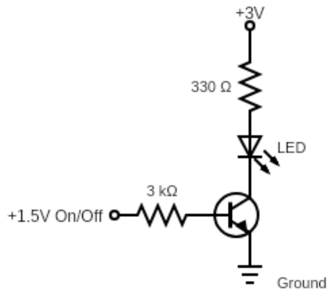


Figure 9 A circuit for controlling an LED with a transistor.

If we were to control a light with a transistor instead, the light would be connected in series with the transistor's collector and emitter terminals. Instead of physically breaking the circuit, as in a mechanical switch, the circuit would be broken by the aforementioned depletion layers in the transistor. Electrical current applied to the base terminal controls the current flowing between the two other terminals. When there is no current applied to the base, electrons between the collector and emitter are immobilised by the depletion layers and the lightbulb is *off*. If we apply a small current to the base terminal, current will flow from the collector to the emitter, and the light will turn *on*. With transistors, we apply electrical current to switch between on and off states much more quickly than is possible with a mechanical switch.

From Hardware to Logic

Transistors can be arranged in clever ways to create logic gates, devices capable of computing Boolean algebra with electricity. To understand why logic gates are important to computing, we must take a brief detour into Boolean algebra.[14]

Boolean Algebra

Boolean algebra is a branch of algebra that deals with logical relationships. In Boolean algebra, all values are binary: either True or False. These binary values can also be represented as 1 and 0.

Boolean algebra is powerful because it allows us to represent and compute logical relationships between Boolean values with the help of Boolean operators, such as AND, OR, and NOT. These operators take binary values as inputs and return binary values as outputs [41, p. 8]. Boolean operators are a deterministic mathematical formalism: given a certain input or set of inputs, an operator will always return the same result.

---

[14] The figures in this and the following subsections are inspired by graphics in Crash Course Computer Science [43].

For instance, the Boolean operator AND takes two input values and returns one output value. If input *A* is True and input *B* is False, then *A* AND *B* is False. The AND operator will only output True if both inputs are True. Boolean operators like AND can be combined with other operators to create Boolean expressions.

Table 2 Truth Table for the Boolean expression AND.

A	B	A AND B
False	False	False
False	True	False
True	False	False
True	True	True

Nineteenth and early-twentieth century computers represented numbers as decimals. These computing machines recreated base 10 mathematical notation in physical hardware devices. Base 10 notation is the most common notation today. In base 10, there are ten possible states for each digit. The first digit place is multiplied by 1, the second is multiplied by 10, and so on (see figure at right). Computers that simulate decimal notation encode at least ten distinct states in their hardware.[15] Decimal notation has some intuitive appeal, because humans usually have ten fingers and ten toes, but other notation schemes including binary have been used popularly in different societies throughout history. Early computer inventors naturally imagined that computer hardware should compute numerical relationships the same way they (Western humans) did. For almost 100 years, many computers were built to simulate decimal mathematics. Babbage's unfinished Analytical Engine (1837) represented decimal notation in mechanical gears and axels, and the ENIAC machine (1946), mapped decimals to a network of 18 000 vacuum tubes [44], [45, p. 168].

Table 3 Base 10 notation of the decimal number 192.

Hundreds	Tens	Ones
1	9	2

In the 1930s, researchers discovered that binary switching devices could be used to solve Boolean algebraic expressions. The discovery was made independently by researchers in Russia and America. In 1935, Russian logician Victor Shestakov discovered that Boolean logic could be mapped to binary switches [46, p. 260]. Shestakov imagined that electromechanical relay switches popular in the telephone network could solve Boolean expressions. His findings were not published until 1941. In his 1938 Master's thesis, Claude Shannon made the same observation and proved that telephone switching relays could in fact compute Boolean expressions [47]. This discovery, that Boolean algebra could be simulated with fast binary switches, opened the door to modern digital computing—so-called for its basis in binary hardware and symbolic representation. However both researchers were inspired by the electromechanical telephone relays of their day. Although the technology enabled them to perceive the possibility of binary

---

[15] Additional bits can encode whether the number is positive or negative, for example.

computing, it was not until the next decade that fully electronic switches with no moving parts were to be invented.

In the late 1940s, researchers at Bell Labs invented the transistor, the solid state electrical switching device explained in the previous subsection. Transistors are an excellent medium for performing Boolean algebra for a variety of reasons. First, transistors can switch states at electric speed—much faster than mechanical devices such as the electromechanical relay. Second, transistors can also be made much smaller than antecedent technologies. Third, they are much more durable than comparable technologies, such as vacuum tubes, which makes them more cost effective. The invention of transistors helped launch the world into the present era of abundant and increasingly inexpensive computing machines.

In digital computers, Boolean algebra's True and False values are mapped to High and Low voltage readings of transistors. Let us imagine a transistor in a circuit running at 5 volts. If voltage is applied to the transistor's base, electrons will pass from the collector through the transistor, and a voltage reading taken at the emitter will show around 5 volts. If no voltage is applied to the base, electrons will not be able to pass from the collector through the transmitter, and a reading measured at the emitter will show around 0 volts.[16] As Shestakov and Shannon showed, these High and Low voltage states can be used to represent the two possible values in Boolean algebra in voltage alone. This is the basis of binary, the essential language of digital computing.[17]

---

[16] Each computer quantizes the voltage to High or Low readings differently. In 5 volt Arduinos, for example, any voltage reading above 3 volts is considered high, while anything below 1.5 volts is considered low [48].

[17] It is, in fact, possible to make use of the intermediate voltages between, for example, 0 and 5 volts. Early computers discerned three and five states between 0 and 5 volts. These are called ternary and quinary computing systems. These systems are, however, more susceptible to electrical interference. The more states allowed, the more ambiguous the transistor's state becomes. Binary solves this problem by quantizing the readings to only one of two extreme states, High or Low, On or Off [49].

Logic Gates

In computers, transistors are organized into special circuits called logic gates, which are physical devices that implement Boolean functions [41, p. 11]. For instance, a pair of transistors and three resistors can be connected as shown in the diagram to create a circuit that replicates the logical operator AND with electricity [50]. For current to pass from the power supply (the point labeled +5V) to the point marked Out, both *A and B* transistors must have a charge applied to their bases.

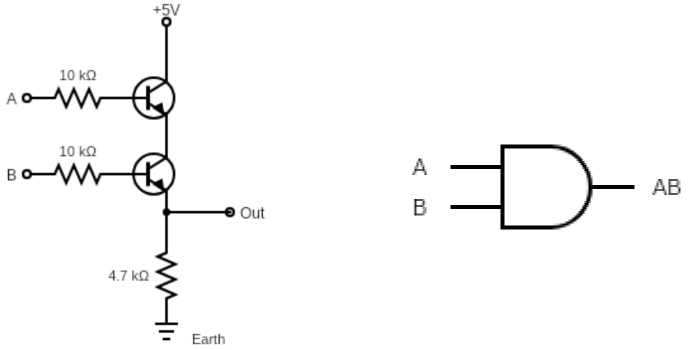


Figure 10 AND logic gate circuit diagram (left) and AND gate symbol (right).

NAND logic gates are particularly important in computing. NAND stands for “NOT AND” gate. NANDs are composed of an AND gate, which we saw just above, and the NOT gate, which has only one input and always returns its opposite as output. NOT True becomes False and vice versa. It follows that NAND returns the exact opposite Boolean values of the AND gate (see Truth Table at right) [41, p. 19]. In other words, NAND takes two inputs, and returns True in all cases *except* when both inputs are True. This is the exact inverse of the AND gate. The circuit diagram below shows how to create a NAND gate with only two transistors and three resistors.

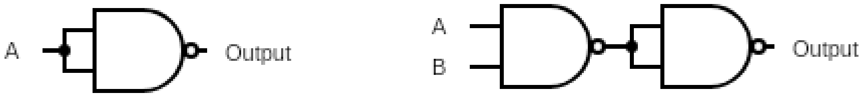


Figure 11 NAND-based AND gate (left) and NAND-based NOT gate (right).

NAND and NOR gates exhibit a special property called functional completeness. Functional completeness means that one can build all other gates using either of these gates, alone. Many computers today are often composed exclusively of NAND gates, because they require less parts and so they are cheaper to manufacture at scale than NOR gates. How does this work in practice? For example, if we tie the A and B inputs of the NAND together, then it behaves like a NOT gate. If we put a regular NAND gate *before* the synthetic NOT gate we just created, then the pair will behave like a regular AND gate. All other logic gates can be implemented with five NANDs or less.

A	B	A NAND B
False	False	True
False	True	True
True	False	True

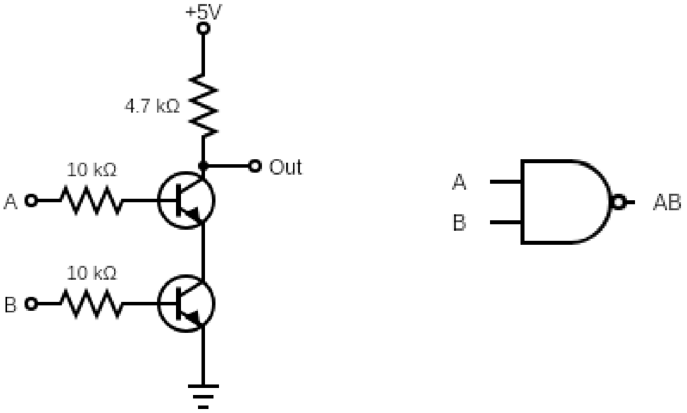


Figure 12 From left to right: NAND truth table, NAND circuit diagram, and NAND logic gate symbol.

From Logic to Calculation and Memory

Logic gates can be combined to create circuits that perform operations essential to computing, such as calculating sums and determining whether a value is even, odd, or equal to zero.

Adders

To create a simple calculator out of logic gates, we start by building a Half-Adder. Half-Adders add two binary inputs together. To build the Half-Adder, we will need one XOR gate and one AND gate.[18] If we represent the Boolean values as binary digits, instead of True and False, we can see that the XOR gate is already quite close to being an adding machine (see truth table) [51], [52]. XOR returns the correct output for a Boolean adding machine, except when both inputs are 1. To solve this problem, we can connect both inputs to a parallel AND gate (below left). If the XOR returns 0, but the AND gate

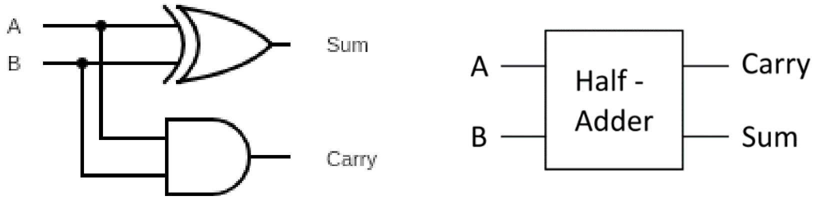
Table 4 XOR gate truth table.

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

---

[18] In built computers, these would be made of NAND gates, as shown above.

returns 1, then we can call the result a carry bit, which will “carry the one” into another Half-Adder, as we shall see in a moment. Once we know how they work inside, Half-Adders can be abstracted into a black box device that takes two binary inputs and returns their sum and a carry bit.



Inputs		Outputs	
A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Figure 13 From left to right: Half-Adder circuit diagram, Half-Adder black box abstraction, and Half-Adder truth table.

To add more than two binary values to one another, we need a Full-Adder. Full-Adders are made with two Half-Adders, and an OR gate. Full-Adders can sum three digits at once. This is crucial for summing larger numbers.

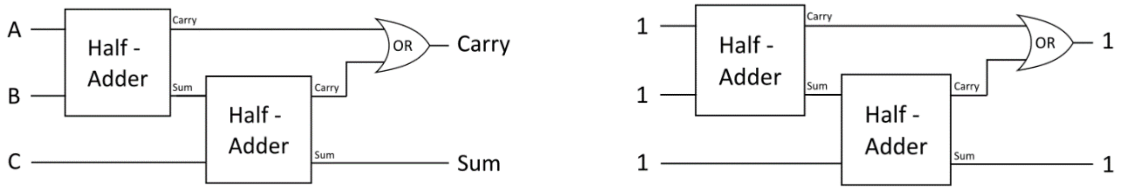


Figure 14 Full-Adder composed of two Half-Adders and an OR gate (left) and a Full-Adder summing three binary digits (right).

In a Full-Adder (above left), the first two bits (A and B) are input into a first Half-Adder. Its sum is fed into another Half-Adder. Its carry bit is passed into an OR gate. The second Half-Adder takes as input the sum output of the first Half-Adder and the third input value (C). The second Half-Adder’s carry bit is also passed into the OR gate. The Full-Adder outputs a two-bit binary value. The sum is the final digit and the carry is the first digit. For example, if we add together three binary 1s, the result is 11, or 3 in decimal notation.[19]

Half-Adders and Full-Adders can be combined to make a logic circuit capable of summing larger numbers. The 4-Bit Adder (see above) shows a circuit that can sum two four bit numbers. This architecture can be scaled up with more Full-Adders to accommodate larger values. As in the previous example, the sum bit becomes the last digit output, but in these larger adding circuits, the carry bit is fed into another Full-

---

[19] Visit https://www.falstad.com/circuit/e-fulladd.html to manipulate a Full-Adder simulation [53].

Adder. This architecture can be expanded with more Full-Adders to sum even larger binary numbers. If the final Full-Adder's carry bit is 1, then the two input values exceed the adding circuit's capacity. When the inputs are too large, the result is called an overflow.

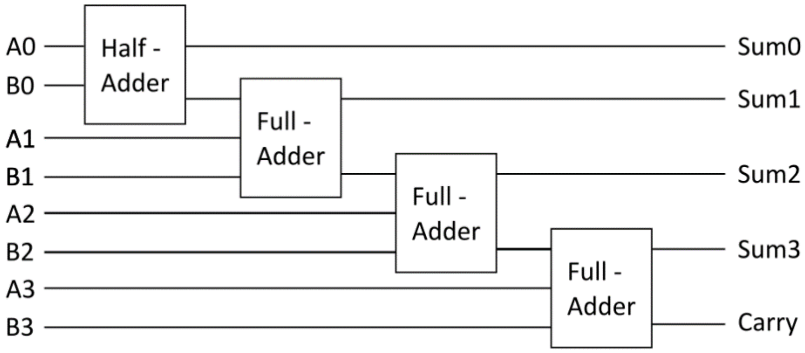


Figure 15 A 4-bit Adder.

Logic Unit

Besides arithmetic, logic gates can also be combined to create a *logic unit*, which can evaluate logical relationships between larger binary values. Logic gates, which we saw earlier, take one-bit binary values (0 and 1) as inputs and produce one-bit outputs. Logic gates perform useful operations, like determining if two inputs are the same in the case of the AND gate. A logic unit can perform similar and more elaborate operations *with multi-digit binary inputs*. That is to say, a logic unit performs logical operations on binary input values greater than 0 and 1. Logic units can determine if two input multi-bit values are equal, for instance. Logic units can also perform numerical tests to determine whether a value is equal to zero, even, odd, or negative. A logic unit achieves this feat by combining several logic gates into a more sophisticated abstraction.

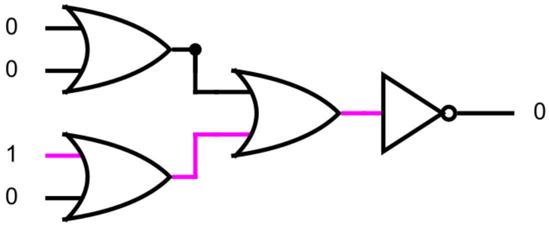


Figure 16 Diagram of a Zero Output logic circuit. Purple lines indicate 1 value, black lines indicate 0 value.

To create the logic unit circuit that determines if a value is equal to zero, for instance, we need only a handful of OR gates and one NOT gate. Let us look at an example that determines if a four digit binary input is equal to zero (see figure). If we pass each digit of the binary value into two OR gates, then the outputs of those OR gates into another OR gate, we will have a circuit that outputs 1 when any of its four inputs is 1. If we add a NOT gate to the end of our circuit, we will have a circuit that outputs a 1-bit only when all of its inputs are equal to zero [51], [54]. So, if the input binary number is equal to 0000, then the output will be 1. If the input is anything other than 0000, the output will be 0. This same architecture can be expanded with more OR gates to accommodate larger input values.

The logic unit enables a computer to perform logical operations and numerical tests on multi-digit binary values. The logic unit provides building block operations that are essential to performing abstract computations, as we shall see shortly.

Arithmetic Logic Unit

Together, an adder and a logic unit form an Arithmetic Logic Unit, or ALU. The ALU is at the heart of a basic computing machine. It can perform three types of operations. First, the ALU can execute arithmetic operations, such as adding, subtracting, and incrementing. Second, it can perform logical operations, which are also called bitwise operations, such as AND and OR. Finally, ALUs can bit shift, which is to say they can move bits left or the right in wider binary spaces.

The basic ALU has three inputs, several outputs, and is represented by a V-shaped symbol (see below). To control the ALU, voltages are applied to its input terminals. Two input values, called operands, are passed into the input terminals marked A and B. The size of the operands depends on the ALU. Early ALUs took 4-bit inputs, but industrial devices can handle 512-bit inputs. The operation code, or opcode, is a short binary value that controls what operation the ALU performs. ALUs with 4-digit opcodes, for example, can perform a maximum of 16 operations. Circuitry inside the ALU interprets the given binary opcode and causes the ALU to perform the associated logical operation upon the operands. For instance, if we pass two values into A and B, and pass the opcode 1000, which might represent the Addition operation, then the ALU will output the sum of A and B (see diagram bottom) [51]. ALUs also output a series of flags, which are single-bit outputs that give further information about the output value. The zero flag, for instance, makes use of the equals-zero circuit implemented in the Logic Unit section above. If the output of the operation performed on A and B equals zero, then this flag will output 1. Other flags will be “raised” if the output value is negative, even, odd, or if the result overflows the adder’s maximum capacity.

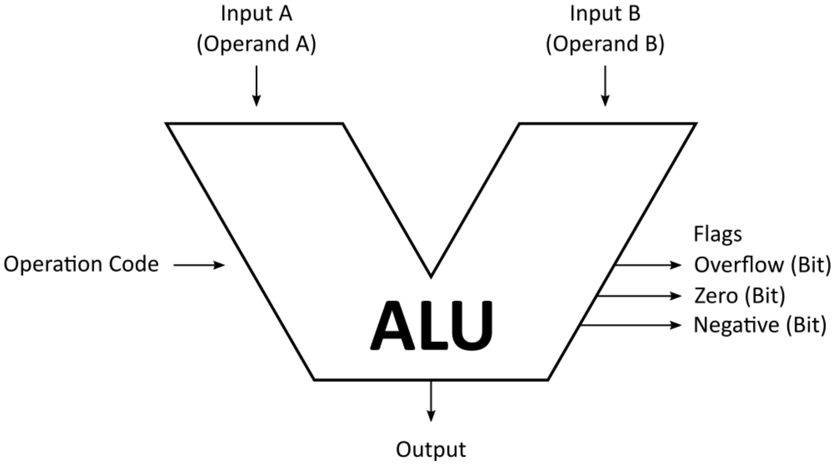


Figure 17 The Arithmetic Logic Unit (ALU) symbol used in block diagrams.

ALUs are capable of performing fundamental logical and mathematical operations upon input values, but they cannot simulate arbitrary software programs on their own. ALUs can only perform the operations for which they have opcodes. To compute and execute programs, a computing machine must also be able to store values in memory.

Memory Latch

Memory, or retained information, is the other essential primitive in building computing machines. Until this point, all of the circuits we have discussed fall into the category called *combinatorial logic*, which means that they do not store any information. In combinatorial logic circuits, the output depends solely upon its present inputs [55, p. 101]. When the inputs change, so too do the outputs. Combinatorial logic does not retain any information *state*—another term for memory.

By contrast, logic circuits that retain information are called *sequential logic*. The output of sequential logic circuits depends upon present and prior inputs [55, p. 102]. In other words, a sequential logic circuit remembers the past and is the basis for computer memory. How can a circuit built with rudimentary components like logic gates retain information?

A gated latch is a circuit that can store a single bit of information [41, pp. 41–43]. Latches can be constructed a variety of ways, with either NAND or NOR gates. The gated latch is a volatile memory device, which means that it retains information so long as the circuit is powered. To store information that will not disappear when the power goes out or the battery dies, we would need a persistent memory device, which I will not cover here.

To demonstrate how memory is made, let us build a gated latch with NAND gates. I will describe the circuit using True and False rather than 1 and 0 because they are easier to understand in written text. I have colour-coded the diagrams below with green for True and red for False. I have provided a version of the circuit diagram for each of its five possible states. First, let us recall the NAND gate's truth table (at right), which shows that a NAND gate always outputs True *except* when both of its inputs are True.

Table 5 NAND truth table.

A	B	A NAND B
False	False	True
False	True	True
True	False	True
True	True	False

Now we are ready to wire four NAND gates together according to the circuit diagram below. The latch is controlled by two inputs, *Input* and *Write enable*. The value held in memory is read from the *Output* pin on the right side of the diagram. Points *a, b,* and *c* help us to understand the state of the circuit at the output of each NAND gate, but these are neither inputs nor outputs. As the colour-coded diagrams show, the *Set* pin controls whether the memory bit can be written to. When *Write enable* is True, as in the first two diagram states, the *Input* pin's state is reflected in the *Output*. When *Set* is False, the *Input* pin's state does not affect the circuit, as in the subsequent three diagrams. I will refrain from verbosely describing each of the possible states, as it will be easier for the reader to confirm that the diagrams are correct by applying the NAND truth table (above) in each circumstance (below). As you can see, the 1-bit memory can be achieved with only a handful of simple parts, which were themselves made up of previously described abstractions.

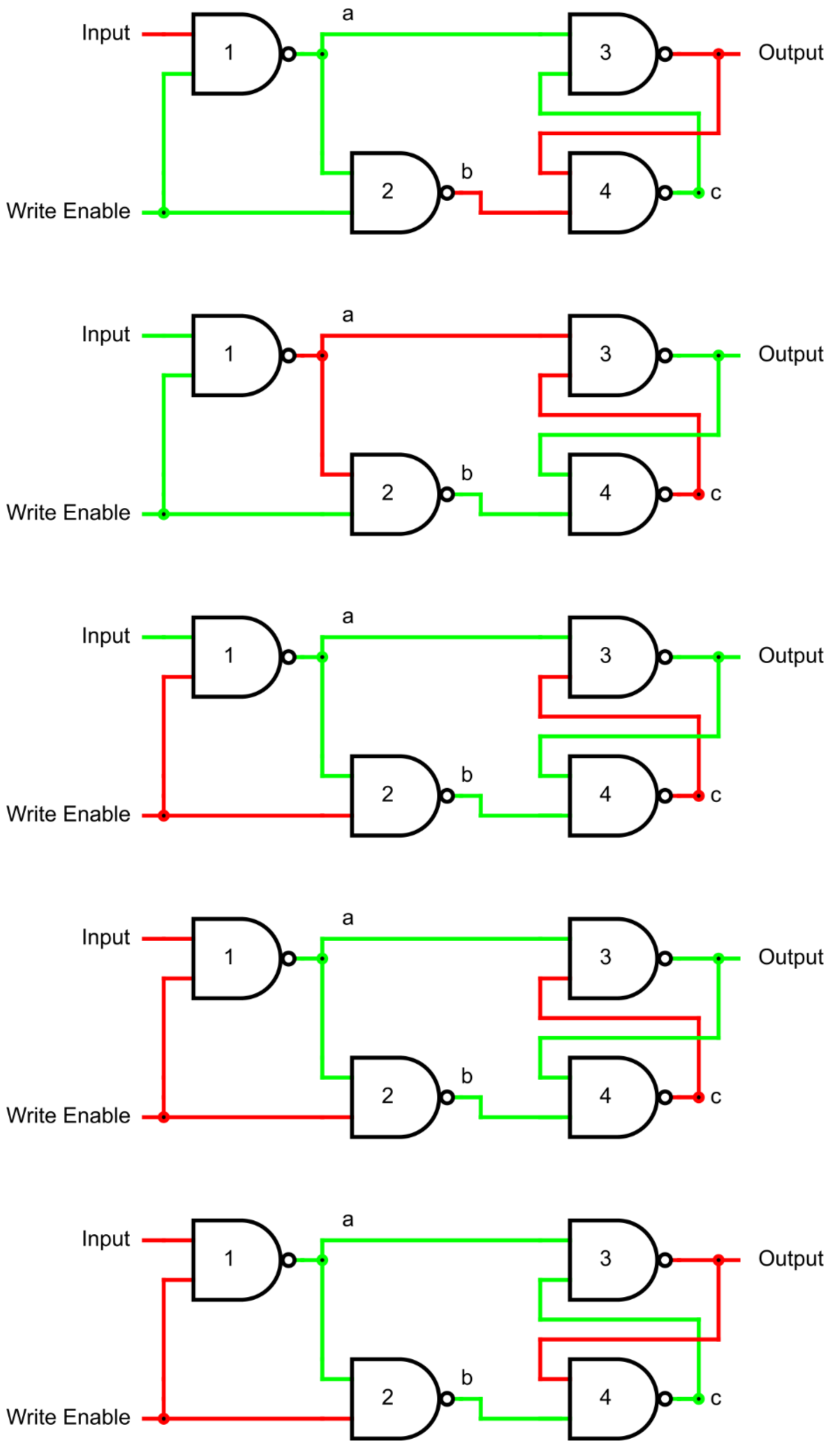


Figure 18 Five possible states of a NAND gated latch or 1-bit memory circuit.

Larger Memory Devices

Multiple latches can be strung together to create volatile memory devices large enough to serve practical applications in modern computers. First, a handful of 1-bit latches can be connected to create *registers*, which are memory devices capable of storing multi-digit binary values. Four or eight latches can be placed side-by-side and wired together to create a 4-bit or 8-bit register, for instance. The number of bits that a register can store is called its *width* [43].

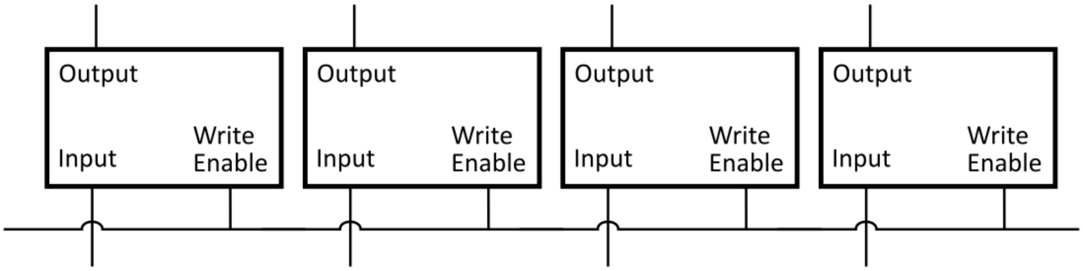


Figure 19 A 4-bit register composed of four latches.[20]

For higher density memory application, latches can be organized into a 2-dimensional matrix (a grid). For instance, a matrix of latches with 16 rows and 16 columns is able to store 256 bits of information—a familiar quantity for anyone who has worked with RGB colors.[21]

A latch's location within the matrix is called its *memory address*. Memory addresses specify the row and column intersection of each latch in binary. The length of these addresses depends on how many latches are in the matrix. If the matrix has 16 rows and 16 columns, then the address for each axis will have to be 4 bits long. The first column's address is 0000. The last column address is 1111 in binary.[22] Anything less than four bits would be too small for such a large matrix. Together, the row and column addresses in this example make an 8-bit address.

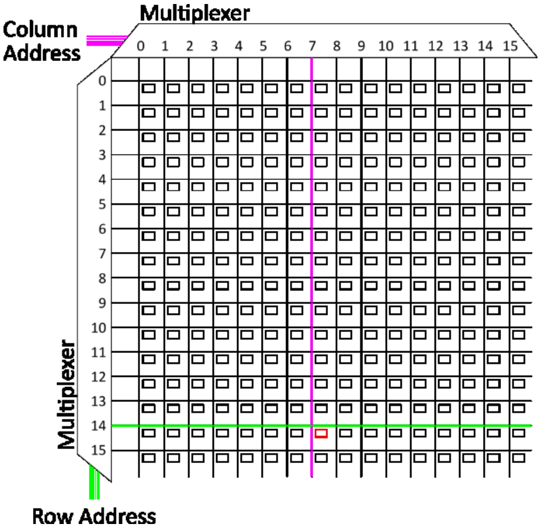


Figure 20 A 16 x 16 latch matrix controlled by two 4-bit multiplexers.

To read and write to the memory, devices such as the CPU, which we will see in a moment, send binary memory

---

[20] The latches share a single Write Enable wire.

[21] In image manipulation and vector art applications, colors are often represented by values of red, green, and blue. The amount of each colour is described by a value between 0 and 255, which corresponds to the 256-bit space of possibilities in an 8-bit value. 8-bits can also be called a byte, or an octet.

[22] Binary 1111 is the 15 in decimal notation. The ultimate column is column 15 because the first column is column 0.

addresses to the memory's two *multiplexers*, which act like a switchboard to the latches.[23] One multiplexer controls the column wires and the other controls the rows. The multiplexers are composed of combinatorial logic circuits that interpret the addresses and turn on the appropriate row and column wires. Only the latch at the intersection of the active row and column wires will be enabled.

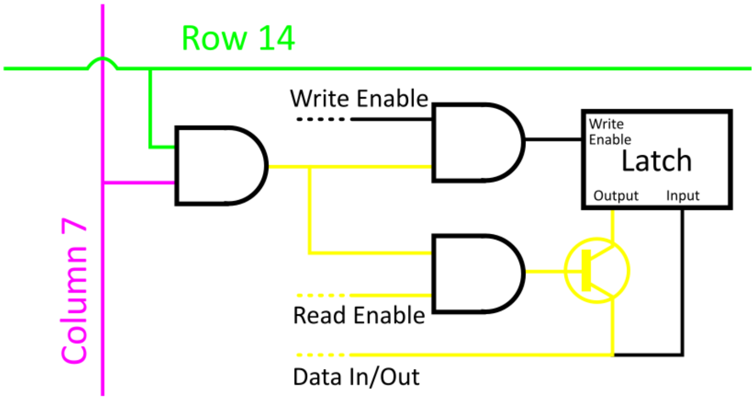


Figure 21 Detail view of a latch in the 16 x 16 latch matrix.

In contrast to registers, the latch devices in a latch matrix require a few additional logic gates to interpret signals coming from the multiplexers. First, it must be noted that all latches in the matrix share a single data wire, which handles both input and output, a single read enable wire, and a single write enable wire. How can 256 latches share the three wires that control their operation? To save on circuitry, memory modules are made so that only the latch at the intersection of active row and column wires, coming from the multiplexers, is activated. To make this possible, each latch is preceded by an AND gate, into which both the row and column wires are connected. If both the row and column wires are active, then the AND gate outputs True to a pair of AND gates preceding both the Write Enable and Read Enable wires. If the row and column wires are active, and the globally shared Write Enable or Read Enable wire is active, then the given latch will be activated for that operation. For example, to read data from the latch in the seventh column and the fourteenth row (see figure above), we output a 1 on the Read Enable wire and pass the value 0111 to the column multiplexer, and 1110 to the row multiplexer.[24, 25] The latch activates and we read the contents of its 1-bit memory on the Data In/Out line.

Memory matrices like the aforementioned 256-bit memory modules can, in turn, be connected to one another to provide even greater memory capacity. A computer's RAM, or Random Access Memory, is

---

[23] Each of the 256 rectangles depicted in Figure 20 is a latch. The shared Write Enable, Read Enable, and Data wires are not depicted.

[24] Note that the first column and row is labeled 0, so the binary address 1 refers to the second column.

[25] In Figure 21, three AND gates and a transistor sit between the matrix and each of its gated latches. The Write Enable, Read Enable, and Data In/Out wires are shared between all the latches in the matrix. These shared wires are abbreviated in this graphic representation.

made exactly this way.[26, 27] Eight or more modules like the 256-bit memory described above can be wired together so that the same 8-bit memory address (column and row) can be used in all eight memory modules at once. In other words, each bit of an 8-bit value is stored in the same position in the grid across 8 different devices. This speeds memory access and simplifies addressing. To read the value of a given latch, we need only simultaneously input its memory address into all of the multiplexers, then read the data they output.

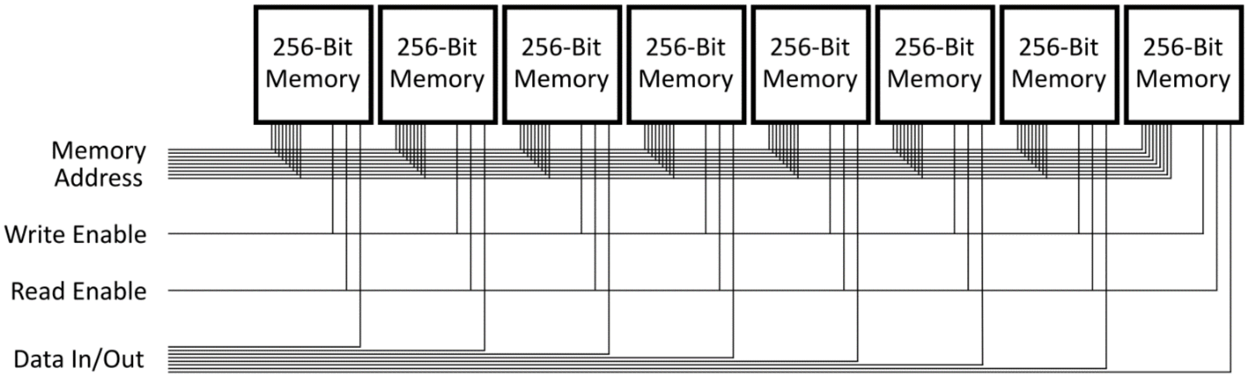


Figure 22 A 2048-bit RAM module composed of eight 256-bit memory matrices.[28]

To make RAM easier to think about, memory access is represented schematically with a simplified interface that abstracts away the difference between the matrix and the multiplexers. To the programmer and devices outside the RAM, memory is represented as a sequential list of addresses containing a given

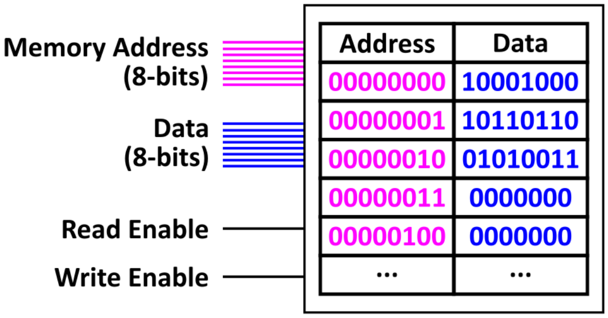
---

[26] RAM is so-called because its contents can be accessed out of order.

[27] The examples provided describe Static Random Access Memory. SRAM can be made with six transistors. There are other types of RAM, such as Dynamic RAM, which are made with one transistor and one capacitor. Different types of RAM have different performance characteristics, the difference between which are not significant to understanding the basic operation of computer memory.

[28] The eight matrices share Memory Address, Write Enable, and Read Enable wires. An 8-bit value is striped across the modules.

amount of binary values, such as 8-bits. In this way, computers are able to keep track of information by storing binary data in devices made up of many memory latches like the one described above.



Address	Data
00000000	10001000
00000001	10110110
00000010	01010011
00000011	0000000
00000100	0000000
...	...

Figure 23 Abstract representation of RAM with 8-bit binary addresses and 8-bit binary values.

From Calculation and Memory to Software

Central Processing Unit

The central processing unit, or CPU, is the device that does the essential work of computation. Every laptop, phone, microwave, and car has at least one, if not many CPUs. Together, the CPU and the RAM are the two halves essential to creating a basic computer today.[29] CPUs read program instructions and data from memory such as RAM, then evaluate logical operations and arithmetic expressions, and finally write data to memory.

The CPU is composed of an ALU, memory registers, a clock, and a control unit. While we have seen the ALU and basic registers already, the clock, control unit, and its many subunits such as the program stepper, instruction register, instruction address register, accumulator register, and temporary register are new. I have also not discussed the bus, which transports data between devices in the computer.

To understand how these components can be turned into a working computer, I will provide a high level overview of how the parts are connected, so that we may better understand how to create a hardware device capable of simulating systems of symbols—that is to say, hardware that can run software. Just like the components we have seen before, all of these components are made of logic gates, which are in turn made up of mere NAND gates.

Software is made of a sequence of *instructions*, and it is the CPU's responsibility to execute those instructions. Each instruction defines an operation to perform, and specifies upon which data to do so. A

---

[29] I will not discuss graphics processors or any other application specific architectures, despite their growing popularity and importance.

typical instruction includes an operation code (or opcode) and one or two operands, which are either binary values or memory addresses where binary values can be found [55, p. 211].

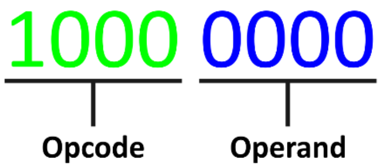


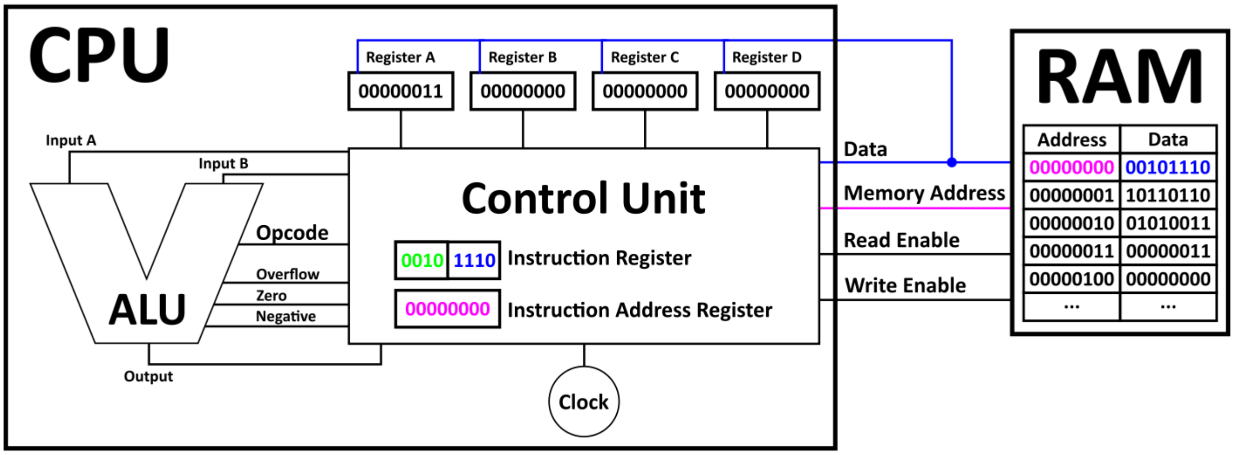
Figure 24 Instructions typically include an opcode and one or two operands.[30]

The opcodes we saw earlier, in the ALU section, are a subset of the CPU's opcodes. In addition to instructing the CPU to perform the operations provided by the ALU, such as addition, opcodes can also tell the CPU to access or modify data stored in RAM or other devices. In the hypothetical computer we are building, the leading bit of the opcode indicates whether the instruction specifies an ALU operation or a memory operation—1 might signify an ALU operation and 0 a memory operation [56, pp. 121–124]. Inside the CPU, the control unit's decoder subunit interprets the instruction and sends, retrieves, or processes data on the relevant wires. The operation associated with a given opcode is defined by the *instruction set architecture*, which differs for each CPU platform [55, p. 211].

There are three basic types of instructions [55, p. 213]. There are *data handling* instructions, which tell the CPU to read, write, or copy data to or from RAM, its internal registers, or an external hardware device such as a keyboard or display. There are *data processing* instructions, which tell the CPU to perform arithmetic expressions (e.g., add or subtract) or logical operations (e.g., compare) on one or more values contained in registers inside the CPU. Finally there are *flow control* instructions. Normally, the CPU executes instructions sequentially. When the CPU encounters an instruction that changes the *flow of control*, such as a JUMP instruction, it skips backward or forward to a specified position in the list of instructions in RAM.

---

[30] Some instruction set architectures include additional fields in each instruction.



Address	Data
00000000	00101110
00000001	10110110
00000010	01010011
00000011	00000011
00000100	00000000
...	...

Figure 25 The CPU and RAM work together to compute software programs.

We can walk our way through an instruction to see how the CPU and RAM execute a program.[31] For this example, we will execute an instruction already loaded into the RAM. Each step through the instructions is driven by the clock.[32] The first step of the program is called the *fetch* phase [57]. During the fetch phase, the control unit inputs the memory address in the *instruction address register* to the RAM's memory address input (in purple). When the program first starts, this address is 0000 0000.[33] The RAM returns the data stored at the address, 0010 0011, and it is copied into the *instruction register*.

At the next tick of the clock, the CPU begins its *decode* phase. In the decode phase, the control unit decodes the instruction in the instruction register [57]. The first four bits are the opcode, and in this case the opcode 0010 corresponds to the LOAD_A memory operation, which tells the CPU to load data into Register A. In the case of a LOAD_A instruction, the latter four bits, 0011, specify the memory address of the data to be loaded.

At the next tick of the clock, the CPU begins the *execute* phase. In the execute phase, the control unit performs the operation specified in the instruction's opcode [57]. To load a value into register A, the

---

[31] The following example is adapted from [57].

[32] This is a simplification. In fact, additional logic is used to subdivide each clock tick into smaller periods of time. These sub-ticks are fed into the stepper, the part of the control unit that executes all of the actions required to shift data from one register to another. In the main text, I describe data operations as single-step operations, when in fact in a real computer, each register or memory address must first be *enabled* to output its value on a shared bus, or set of eight wires, before the receiving register can be put into the *set* mode, in which it will update its stored value. For simplicity's sake, these details have been omitted. For more information on the stepper and instruction execution, see [56, pp. 93–112].

[33] I have added a space between the first four and last four bits in the octet to make this section more readable.

control unit sends a read enable signal to the RAM and sends the address 0011 on the memory address wire. The control unit also sends a write enable signal to register A. The RAM outputs the data at memory address 0011 to its data line and it overwrites the data in register A with 0000 0011. The instruction execution is complete, and the instruction address register is incremented to 0000 0001, which will be the next instruction executed.

Table 6 A sample Instruction Table.[34]

Instruction	Opcode	Description	Address or Registers
LOAD_A	0010	Load value from RAM into register A	4-bit RAM address
ADD	1000	Sum two registers and store the result in the second register	2-bit register ID, 2-bit register ID

Instructions like the one we just discussed form the basis of software. In addition to loading data from RAM into registers, the CPU can perform all of the operations we have previously discussed with the ALU, and more. No matter how sophisticated the piece of software, if it is executing on a CPU, then it is composed of individual instructions like these at the lowest level of abstraction. Despite their abstract appearance, apps, verbal interfaces, and games are all made up of simple instructions for loading, comparing, evaluating, calculating, and storing data.

The CPU we looked at here is an example of a *von Neumann architecture*. This architecture is named after computer scientist John von Neumann, who led the research effort that yielded this design in 1945. In the von Neumann architecture, program instructions are stored in the same memory device as program data. This contrasts with the *Harvard architecture*, in which program instructions are stored in one memory device and data is stored in a second memory device. By rough analogy, the Harvard architecture is like having all of one's apps on one USB stick and all of one's files on another USB stick, whereas the von Neumann architecture puts the programs and the data they work upon in the same place.[35] The von

---

[34] Real processors have many more instructions, including separate instructions for loading from each register, instructions for other ALU operations, and instructions for storing values in RAM.

[35] In fact, lots of devices today are actually

Neumann architecture treats program instructions the same as the data a program works upon or creates. Although this opens CPU to additional security vulnerabilities, it has the intriguing property of allowing the program to access and modify itself, as its own instructions are data just like the rest of the memory's contents.[36]

To simplify the process of programming, over the years, computer engineers and programmers have developed mnemonic abstractions for the binary opcodes seen above. Like all mnemonics, these abstract programming languages make the operations easier to remember. The lowest level programming language is called *machine language* [56, pp. 121–2]. We have already seen some machine language in the pages above. The instruction LOAD_A is machine language that translates to 0010 in the machine we have built. In the early days of digital computing, a programmer would write their code in machine language, and a computer program would *compile*, or translate, the word LOAD_A into the corresponding binary representation. The CPU must be directed with binary instructions, and not machine language, because instruction values are interpreted by the decoder and other hardware that operate on binary information only.

In machine language, one written instruction corresponds to one CPU instruction. In the years since machine language was invented, programmers have come up with increasingly abstract languages. One line of code in a higher level language might correspond to dozens or hundreds of CPU instructions. Just like machine code, programs written in higher level languages are compiled or interpreted into the same CPU instructions we have seen. The more high level of a language a person codes, the more quickly they can orchestrate a lengthy and perhaps sophisticated program. When using higher level programming languages, the programmer exchanges precise control at the level of the instruction for programming speed. Whether this is desirable or not depends on the programmer and their intentions.

In this section, I have shown that computers are composed of abstractions. From the beginning of our journey up the hierarchy of abstraction, we have seen a dozen times how knowledge in a particular

---

[36] In fact, things are even trickier than this. Many modern CPUs have aspects of both the von Neumann and Harvard architectures at once. These are sometimes referred to as "modified Harvard architecture," although that term covers a range of different architectures. For instance, to optimize for speed, specialty processors like digital signal processors (DSP) store some of their instructions in memory modules inaccessible to programs they run. Microcontroller CPUs like the AVR8 also store the program and data in different memory devices. This design decision allows the AVR8 to read program instructions and read or write data to memory simultaneously. CPUs can also be designed with separate memory modules inaccessible to program code to harden them against malicious code [58].

substrate can be packaged into a conceptual device—a named abstraction—that can in turn become the material basis for yet another more abstract system. We have seen together that these abstractions are not limited to any particular substrate. First, we learned, packaged, and abstracted away the chemistry of semiconducting materials into positive and negatively doped silicon—p-type and n-type, if you recall. We then used those chemical abstractions to create a BJT transistor, our first significant hardware abstraction. We used the transistor as the basis of logic gates, such as NAND, which we used to create adders and 1-bit latches. At this stage, our hardware abstractions began to reveal proto-virtual properties like memory, calculation, and multi-bit logical comparison. We then used these combinatorial and sequential circuits to create the ALU, which made possible the CPU, and the memory matrix, which made possible RAM. Finally, we saw how together, the CPU and the RAM fulfilled the minimum sufficient conditions for a computing machine. In the next section, we will explore what exactly makes this computing device so special. Why is it that a computing machine is unlike any other machine? In exploring the answer to this question, we will observe together the next abstract material transcendence, from hardware to software.

Section 2: Computing is Abstract Work

What makes a computer special? What makes a computer different from other machines?

A typical machine is "an apparatus constructed to perform a task" [59].[37] Such a machine's internal structure fits its function. The mechanisms of which it is composed are chosen and manufactured to fit perfectly together for the task it is intended to complete. A machine for clipping hair will be useless for clipping the lawn. A machine for washing clothes will be no good for cooking eggs.

In general, a machine's design anticipates its intended function. Of course, machines can be modified and made to do things outside of their intended purpose, but as built, a typical non-computational machine is fit for a preset range of purposes. A machine can be repurposed and recontextualized, as a *readymade,* for instance, but a traditional machine will not be able to complete the task of *every* other machine—at least not without great difficulty and extra parts.

A computer, by contrast, is a machine whose function is not fixed at the time of manufacture. Unlike lawn mowers, calculators, and forklifts, the computer is a machine that can be programmed to perform new operations long after it is made. This is what makes computers special. A computer is a machine that, at

---

[37] This sentence is sure to age poorly, for the idea of a typical machine being one without computation is unlikely to outlive even me.

least hypothetically, is fit to perform any computable function. For this reason, computers can be called general purpose machines. That is to say, a computer is a machine fit for an abstract purpose.

Computers are not magical, but their ingenious hardware configuration enables them to serve a huge variety of applications. Like other machines, computers are made up of components with fixed functions; however, unlike most other machines, computers are able to step through scripts of instructions that we call algorithms. These algorithms are sequences of actions.[38] Programming is the activity of defining algorithms, which the computer then executes. Traditionally, algorithms are understood as sequences of instructions for performing specific tasks, but really the computer is quite open ended. An algorithm could solve an algebraic expression, or it could just do a bunch of silly stuff for no reason—to the computer it is all the same.

A computing machine's electronic components provide the primitive operations that can be recombined to create an infinite set of applications.[39] As we saw in the prior section, computer hardware provides affordances for comparing values and performing calculations upon them, too. Computers can also read and write data, including the data of the program instructions they follow. These are the machine's operations, for which it has operation codes. These basic operations are sufficient to enable computers to perform any sophisticated algorithm that can be encoded in terms of those basic operations. This vocabulary is great enough to allow computers to perform a huge range of functions encoded *after* their construction. This is the basis of software, which can be installed, updated, removed, or shared, all without changing the underlying hardware. But what is software, really?

In computer circles, computation is most often understood in terms of the Church-Turing thesis. To understand this idea, we must first take a brief detour into history to learn the origins of the way computer people frame their medium today.

Turing Machines

In the early twentieth century, questions of the nature of mathematics and computation concerned mathematicians greatly. Since Gottfried Leibniz's invention of a calculating machine in the seventeenth century, he and other mathematicians alike had begun to wonder what logical or philosophical regime could underpin the study of mathematics [61]. Without a solid grounding, it was feared that the whole

---

[38] "A procedure or set of rules used in calculation and problem-solving; (in later use spec.) a precisely defined set of mathematical or logical operations for the performance of a particular task" [60].

[39] Infinite but not unlimited. There are things we know of that we are not yet sure can be computed.

edifice of mathematical reasoning was unstable. In the early twentieth century, as more and more areas of mathematics came to realize that they were subject to internal inconsistencies, or lacked grounding in a more fundamental mathematics, the foundational crisis of mathematics ("*Grundlagenkrise der Mathematik*") came to dominate conversation within the community [62]. In the 1920s, German mathematician David Hilbert posed a series of challenges, now known as Hilbert's program, which mathematicians would have to answer for the grounding of their discipline to be firmly established.

In 1928, Hilbert and fellow mathematician Wilhelm Ackerman posed the *Entscheidungsproblem* to the mathematical community. The *Entscheidungsproblem*, or decision problem, was the third challenge of the 1928 iteration of Hilbert's program, and the only aspect of it that I will engage in the present text [63, p. 91]. Hilbert's decision problem asked: Is mathematics decidable? In other words, is there a machine or a procedure that can, given *any* mathematical expression as input, accurately state whether or not the expression can be solved, or if it will result in an infinitely looping solution that reaches no conclusion? If such a machine existed, then every problem that could be reduced to formal logic could be solved definitively.[40]

Alonzo Church was the first person to answer this question. Between 1932 and 1936, Church invented and refined a system of mathematical notation called Lambda calculus, which he used to show that there could be no algorithm that would fulfil the decision problem's requirements. Lambda calculus is a notation for describing functions precisely. It is a means of expressing computable functions, and it is the basis of functional programming languages like John McCarthy's influential LISP (1958) [64]. I will not describe Lambda calculus or functional programming in any greater detail, but both merit additional study as they further illuminate the nature and capabilities of computation.

The second person to answer the decision problem was Alan Turing. Turing solved the problem from a completely different angle, and his argument for the impossibility of such a machine has deeply influenced the way computer people understand their field. Instead of inventing a formal notation for modeling computable functions, Turing solved Hilbert's decision problem with the help of an imaginary device that he called the automatic machine. Turing's machine model and Church's Lambda calculus are two ways of stating the same thing—they are logically equivalent. Church gave a glowing review of the paper in which Turing laid out his solution, and their combined effort came to be referred to as the Church-Turing thesis.

---

[40] In this context, machine and algorithm are equivalent concepts. Both follow a sequence of steps to yield a result.

Turing published his solution to the decision problem in 1937 [65]. Turing's solution to the decision problem relied upon his aforementioned automatic machine. Turing's machine has a handful of parts [66]. First, the machine has a memory, which Turing visualized as an infinitely long strip of tape. The tape is divided into squares, each of which can hold a single symbol. The machine also has a scanning head that can read-from and write-to the tape. Turing specified that the machine can move along the tape, in either direction, reading and writing as necessary. The machine also has an internal memory, or state, that can retain one symbol. Finally, the machine is equipped with a set of rules that it follows. The rules dictate what the machine should do given its internal state and what symbol it is currently reading. According to Turing's definition, the computing machine is an automatic machine, because its behavior depends uniquely upon the current state inside the machine, and the symbol that it is scanning at the present moment.

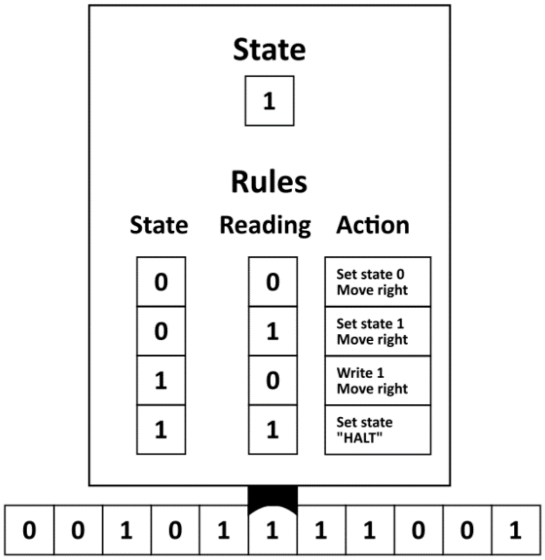


Figure 26 Illustration of a possible Turing machine, with its requisite internal state, rules, read/write head, and infinite memory tape (abbreviated for the purposes of illustration).

Turing's paper proved that the simple machine that he described could, if given enough time, compute any possible algorithm. Any algorithm, no matter how sophisticated or powerful, could be broken down into rules and read/write operations that should be executed dependent upon the current state of the machine. Given an endless amount of memory and time, such a machine could perform any task that could be expressed as an algorithm. This machine, which Church later rechristened the "universal Turing machine," came to define computation. Computing in this context means algorithmic processes executed upon data (information). Any machine that can perform the basic operations that Turing distilled is called

Turing complete. The crucial finding for understanding computing is that any Turing complete machine can compute the algorithm (or program) of any other Turing complete machine.

Turing applied his hypothetical machine to a variation of Hilbert's decision problem called the *halting problem* to show that there could be no algorithm that could satisfy either. The halting problem asks if it is possible to predict if an algorithm and data will halt (reach the end of the program) or if it will enter an infinite loop, without computing the algorithm. In other words, if we recall that a Turing machine is a combination of an algorithm (its rules and execution mechanism) and input data (its memory tape), then the halting question asks if one can construct a Turing machine that takes another Turing machine as input, and predicts whether it will run forever or complete the algorithm, without actually running its program. The halting problem is equivalent to Hilbert's decision problem, because it too asks if it is possible to create an algorithm (machine) that can *decide* whether an input algorithm is computable or not.

Turing showed that it is impossible to solve the halting problem, because computing machines necessarily have enough expressive potential to encode algorithms that will generate an infinite loop—the state in which a program recursively executes itself and never halts. Turing's explanation is a bit involved, and so I will give only the outlines here. In essence, Turing proved that it would not be possible for such a machine to exist by *reductio ad absurdum*. Turing proposed that we imagine a hypothetical machine, called *H*, that could perform Hilbert's desired task: it could take another machine's rules and input, or in mathematical terms the function expression and input variables, and return a yes or no answer as to whether it would halt, if run. Turing then proposed modifying the machine so that it would enter an infinite loop in every circumstance in which it returned an affirmative result. This modified H machine could only yield one of two possibilities, either it would say that the input algorithm and memory would *not* halt, or it would enter an infinite loop, thus not halting. Turing then proposed that if *this* machine's rules and memory tape were fed into H, then the machine could not possibly return a result, because any affirmative result would result in an infinite loop state. Thus, regardless of its internal mechanisms, even a machine that *could* decide whether an algorithm was computable would be subject to adversarial algorithms that would cause that machine to enter an infinite loop. Turing's solution recalls Gödel's incompleteness theorem, which showed that any mathematical notation system will inevitably be open to internal inconsistencies that make it possible to express nonsense that remains correct within the system's rules. The point is, if you can program the machine, then you can write a program that cannot be predicted to halt or loop infinitely, without computing it.

In the course of solving the halting and decision problems, Turing established the most popular definition of computability. Any algorithm that can be computed can be distilled to a series of rules and a given memory state that a simple Turing machine can follow. Although it is not possible to definitely decide whether all programs will successfully terminate without running them, Turing demonstrated that any machine that satisfies the criteria of a universal Turing machine will be able to compute any algorithm. This is the basis of how software is understood inside of the computing community. Computers are Turing machines (minus the infinite memory) that can, in principle, compute any algorithm [66].

Turing's definition of computability helps us to see that computers are not only made of abstractions, computers also simulate abstractions. In the first section, we packaged chemical and manufacturing knowledge together in increasingly abstract packages. In section two, we have seen that those hardware abstractions culminated in a Turing complete machine (less the infinite memory). As Turing proved, a Turing complete machine is capable of working through any algorithm that can be encoded into memory. Software, then, can be understood as the layers of abstraction that sit atop those set in hardware. Software are those abstractions by which the programmer configures and reconfigures the machine in a purely virtual plane. To change the function of any Turing complete device, one need only change its instructions, which in the case of a digital computer means changing only the binary electrical signals stored in RAM.

A Note on Computing in Practice

While any Turing complete computing machine may be able to compute any algorithm in principle, in practice, hardware designers make choices between hardware abstractions to make a given machine more suitable for its intended task. In real world scenarios, a computer's hardware is always optimized for the range of tasks it is expected to perform. Hardware can be chosen to optimize price, performance, heat dissipation, energy consumption, robustness, or the data a machine is anticipated to work upon. For instance, application-specific integrated circuits (ASICs), such as those designed for gaming graphics or hashing algorithms, are specially designed to suit those algorithms' demands.

Computing machines can never fully be separated from the circumstances enabling their physical construction. Computers are tied to the human context in which they are manufactured.[41] As we shall see

---

[41] I owe thanks to Chris Beiser, who made this point clear to me in response to some remarks I made on Twitter. In a pair of tweets, he wrote "No machine can have function fully abstracted from its construction. I think it's

in Chapter 5: Supply Chain, real world computers are disposable goods that tend to be built for particular applications, despite their programmable nature. The Alexa puck sitting on the kitchen counter may theoretically be able to simulate and predict weather patterns, and in a network with all others of its kind it might even be a tractable problem, but it will never be dedicated to that task so long as that function does not serve the institution that controls what software it runs. In practice, computation today is a medium tied as much to the Church-Turing thesis as to other human institutions, and their diametrically opposite incentive schemes. More on this later.

Section 3: Computation Abstracts the World

In the first two sections of this chapter, I explained the two most prominent ways in which computers can be understood as *abstraction machines*. First, I described computers from the hardware and electrical engineering perspective. From this point of view, computing machines are made of a hierarchy of abstractions. Second, I described the mathematical perspective popular amongst computer scientists, which perceives computers as the set of machines equipped to execute any computable program. Computing machines simulate sequences of steps called algorithms, which are described by sequence of instructions. These algorithm or program instructions are encoded in a symbolic representations, which the computer can read and execute. Throughout both sections, I observed that the codes that map most closely to the machine hardware, such as binary and machine code, serve as the foundation for more abstract computer languages, which are often visualized as ”higher level” abstractions.

In the present section, I will discuss the third way in which computers can be considered abstraction machines. Now that I have provided a grounding for how a computer is made, and what distinguishes it from other phenomena, it is time to dive into this medium’s ability to affect human experience. In the pages that follow, I will expand upon computation’s capacity to abstract the world beyond the enclosure. To begin, I would like to take a look at apps, which are amongst the most popular software distribution abstractions today.

Since the launch of the modern smartphone app stores in 2008, apps have become the most popular form of end-user software [67]. Want to watch a video or check into a flight? Turn off the lights or start the car?

---

dangerous to think of a computer entirely as a ‘medium without qualities’—when we talk about a computer as a human form rather than an abstract ideal of computation, it's always tied tightly to certain qualities."

Check into a hotel or pay your taxes? Consult a therapist or play a game? Inevitably, "there's an app for that."[42] With a smartphone, one is only ever a few taps away from using an app.

Apps are abstractions that flatten modern network software into simpler mental forms. With apps, the consumer never has to think about the complexities of hardware and software that we explored earlier. The operating system, low level code, high level code, internet communication, platform-vendor vetting, in-app purchase commissions, and so on are all helpfully obscured by the app interface abstraction. Apps—and subsequent usable software product forms like verbally invoked algorithms—make it easier than ever to make use of new software packages.

Apps also abstract the world beyond the computer's packaging in at least two important ways. First, apps conceal elaborate social practices in a simple software product form. An app can simplify existing social infrastructure, or it can be the branded interface to new social practices. This is one of the app product form's greatest strengths, and the reason why ambitious projects can so easily be presented in the form "what if there was an app that did *x*?" Second, apps can create new systems of value that become integrated with human experience. Networks in the guise of apps inherently affect narrative formation. Each year it becomes more obvious how deeply apps can impact worldview formation. The way each person sees the world and sees themselves naturally in conversation with the media they experience habitually. Apps are not only a convenient software distribution scheme; they are also recurring foils in personal mythology making. An example of each will help elucidate the point.

Apps like Uber mask political and social maneuvering with a friendly app interface. Uber is a two-sided marketplace where riders and drivers exchange money for transportation services.[43] To the driver, Uber is a source of revenue, and to the customer it is an app for getting around. However Uber is also a human social practice. To make this private transportation market possible, Uber had to disrupt the regulated taxi cab industry in many of the cities where it now operates. The company knowingly broke the law to "open markets" to unlicensed taxi cab operation, much to the chagrin of the medallion-holding taxi cab companies and drivers the world over [69]. Despite its apparent simplicity, Uber the app could not exist without Uber the organization that dared to flout local transport regulation and push for regulatory innovations like "gig economy" independent contracting employment arrangements [70]. Uber the

---

[42] "There's an App for That" is a trademark of Apple Inc. [68]. The corporation popularized the phrase in a 2009 campaign promoting the iPhone 3GS. They won a US trademark for the phrase the following year.

[43] Riders and drivers could also be called clients and independent contractors.

company used venture capital backing, legal representation, and lobbyists compensated with company equity to overwhelm thousands of local taxi oligopolies [70]. The Uber app abstracts away all of this labour into a now-familiar software service product form.

Uber demonstrates that apps reduce complex human practices beyond the realm of computation into easily digested software product experiences. Apps are a form of abstraction that obscures the complexity within. Apps abstract all kinds of processes into software interfaces, no matter what multidisciplinary performances are required to keep them running. The app interface allows Uber to make abstract the variety of legal and financial maneuvers executed to make it possible to step into the backseat of a stranger's car and be whisked away to the chosen destination. To use this novel centralized marketplace, all we have to do is "take an uber." It is hard to imagine Uber achieving this feat without smartphone apps, not only because of their geolocation and ubiquitous internet affordances, but perhaps even more so because it is hard to imagine operating so many illegal cabs any other way.[44]

At the same time, apps like Instagram illustrate how habitual engagement with computation can shape the way we perceive the world and ourselves. Early Instagram appealed to the fashion and design-minded demographics who posted gorgeous pictures. Although they seem quaint now, co-founder Kevin Systrom was celebrated for his photo filters, which compensated for the low quality cameras popular at the time. When Instagram first launched, its camera was constrained to taking only square pictures. Instagram's square photos created not only a uniformly laid out feed, but also helped systematize content creation. The app's constrained camera subtly encouraged early Instagrammers to take photos that suited the square frame. The more that I used the app, the more I noticed myself perceiving Instagrammable vistas in the world beyond the screen. Even with the phone in my pocket, Instagram was residually affecting my visual perception of the world. Symmetric architectural features and picturesque landscapes brought the social media network to mind unbidden. Instagram's early square photo content creation constraint successfully coopted my visual processing. While I installed the app on my phone, through habitual use, I ended up installing Instagram in my mind.

Instagram is not the only software interface to color my thinking. Computational media experiences constantly insinuate themselves in my opinions, beliefs, and the ways that I experience life. Network

---

[44] It is interesting to note that, while the app stores will remove apps that threaten their monopoly (torrenting applications, payment schemes outside of in-app purchases), apps that are illegal according to federal laws (VPNs in China), and apps that do not suit their tastes (drone strike tracking app Metadata), they did not intervene in Uber and its kin apps, which clearly broke municipal laws.

software today is designed to hijack user attention and build addiction, and so the impact on users' offline perception is unsurprising [71]. Memes effortlessly inform my sense of norms and the world around me, and these memes are shaped by the computer media in which they are created, shared, and consumed. It is impossible to verify every statement we encounter, and it is equally infeasible to guard our minds from the effects of suggestion. Even if I do not believe what I am seeing, aesthetic experience incites emotional and embodied reactions beyond conscious control. The media we are exposed to impact how we imagine things are, what mechanisms are at work around us, whom we can trust and whom we can't [72], [73]. Once we accept that habitual experiences shape perception, addictive applications can no longer be written off according to twentieth century values as merely frivolous follies of youth or pop culture.

Unfortunately, the relationship between software interfaces, the complexes of social and political infrastructure they mask, and the impact of this social medium on discourse has entered popular conversation in extreme slow motion. The 2016 election of Donald Trump appears to have so wildly violated widely held assumptions about how politics is performed, and what role internet communication plays, that Facebook and other network platforms are now finally being discussed as important actors in society [74]. Unfortunately, the algorithmic selection of media and the engagement optimizing design processes by which communication networks are designed is so dimly understood that the informed criticism is out of reach for many. Critiques also generally fail to provide any medium-native constructive criticism, instead resorting to arguments for reasserting pre-transistor institutional power, such as government or publisher oligopolies. One of the essential contentions of the present text is that we will only be able to imagine new and socially positive ideas in the medium of social computation when more people understand the medium's essence. I will return to this theme in Chapter 4: Making Networks.

Software exhibits that strange property that Abraham Kaplan and others have observed of our tools: first we shape them, and then they shape us [75]. At a superficial level, apps promise to make easier a task that we were already doing—and so they snake their way into our lives. Over time, the software that we expose ourselves to habitually comes to affect what we see. Users naturally come to see the world in terms of the opportunities their chronic software provides. When the sky turns dark and it begins to drizzle, an Uber driver may imagine lucrative surge pricing.[45] Meanwhile, Instagrammers may find

---

[45] Surge pricing is Uber's mechanism for increasing compensation when rider demand outstrips driver supply. When it rains, drivers earn more money.

themselves bringing screenshots of face filters to cosmetic surgery consultations [76]. How can we anticipate the ethical value or political significance of software, if its very presence upsets the basis of our judgment? This is a theme we will revisit several times in the rest of the text.

Of course, this property of being installed in the mind is not unique to apps, or even computation. When a skateboarder traverses the built environment, they perceive opportunities for grinds and ollies, even without their board in-hand. Whether they have their board with them at the time or not, they come to perceive the world through the lens of its affordances. The aphorism that, to a person with a hammer, everything looks like a nail, comes to mind [77]. The productive constraints that we engage regularly become installed in our minds, no matter their medium. It just so happens that there has never been a more expedient means for spreading perception lenses as internet software. It is difficult to imagine what the consequences are when, instead of a skateboard, the designed goods with which we relate contain the dynamic and festering multitudes of human expression.[46]

Apps and other forms of software do not merely exist within the boundaries of the computing machine. Software is powerful specifically because it exceeds the boundary of the computer's enclosure all the time. Software simulations affect the world. At first this meant cracking Nazi encryption, and now it means collecting biometric data with camera apps that make you look like a beautiful unicorn. In this section, I described two ways that computation escapes the confines of the computing machine. First, I showed that Uber is not merely a new interface for mediating existing interactions, but that it is primarily a collection of disruptive social gestures that installed a new regime of labour relations, all the while dressed as an app. Second, I used Instagram as an example to outline how habitual software use installs new perception lenses in the mind. I have demonstrated that computers deserve to be called abstraction machines because they handily introduce new abstractions to the world and to the mind. Apps are powerful because they reshape how we understand the world.

Section 4: Computation is Labour and Money is a Programming Language

What is programming? It may seem academic, but the question of what counts as programming reveals a lot about how one conceives of the medium of computation. To many, humans working with computers

---

[46] Perhaps social computation is one way to understand the electric light that McLuhan noted "escapes attention as a communication medium just because it has no 'content'." Today, we are all utterly drenched in the electric light of content. It shines so powerfully, it penetrates the mind and is re-projected from our eyes onto the world [78, pp. 24–25].

can be divided into two categories: programmers and users. Programmers write code, whilst users simply make use of existing software. Although both work with computers, programmers are the more empowered of the two classes, because they cannot only use software, they can also write their own.

In this section, I will argue that, contrary to common belief, even end-user interfaces, the highest level interface abstractions, are programmable if we look with an open mind. First, I will revisit the hierarchy of abstractions that I described in the very first section of this chapter, "Computers are Made of Abstractions." Then, I will present two opposing perspectives on programming today. Finally, I will introduce a new approach, which maintains the widely accepted technical definition of computing and reveals that programming is already accessible to non-programmers via the same financial interfaces that have been intertwined with computation all along.

Traditional Perspective on Programming and Computation

The ladder of abstractions is a helpful tool for thinking about how computers are built, but this linear hierarchy does not comprehensively describe the flow of power, influence, or invention in computing. As we saw in the first section, the metaphorical ladder suggests that computers are made up of sequential layers of abstraction, from chemistry through hardware to software. This vertical visualization of computing as composed of "low" or "high level" abstractions is just one way of describing a maelstrom of activity that cannot be fully captured on a single axis. For a fresh perspective on computation as a medium, it is important to step outside of the received narrative and inspect ways in which the linear hierarchy of abstraction can be subverted by other virtual media. Which abstractions are programmable is one such access point.

Computer science trains students to associate computation with Turing completeness. Within the discipline, a computer is formally defined as any system that is Turing complete, as discussed in section two. Any machine or environment is Turing complete if it can execute an algorithm, or set of instructions, by reading and writing symbols stored in a memory device, so long as it can choose which instruction to execute depending upon the symbols read from memory. With these basic ingredients, a machine or environment will be able to execute any program that can be written and executed by any other Turing machine.

Turing machines are appealing because they provide a strict definition of computation while remaining agnostic about the computer's material implementation. There are Turing complete electrical digital computers, like the von Neumann architecture computers with their CPUs and RAM. There are biological computers, too. DNA is Turing complete, despite being made up of entirely different materials and architecture [79], [80]. There are also entirely virtual Turing machines, whose memory and reading/writing capabilities exist only in a simulated environment. These virtual machines are also capable of running arbitrary programs, despite their immateriality. So for instance, if a person wants to play Pokémon Yellow, but does not have access to a Nintendo Game Boy, they could download a Game Boy emulator, which mimics the hardware of the missing physical computer entirely in software. In this scenario, there are several Turing machines running atop one another. The binary running on the CPU and RAM constitute a Turing complete environment. The operating system running on that hardware also constitutes a Turing complete environment, because it too is programmable. Finally, the Game Boy emulator running atop the operating system is also Turing complete. Each of these systems could execute any program that could be encoded and executed by any other Turing machine, despite the differences in their construction. Turing machines can be electrical or biological, physical or virtual, and beyond.

Turing completeness is more common than one might imagine. Many systems can be wielded in unconventional ways to reveal latent Turing complete affordances. The *Magic: The Gathering* card game has been shown to be Turing-complete, as have Minecraft and a variant of the Windows game Minesweeper with an infinitely large game board [81]–[84]. Research has also proved that Microsoft PowerPoint provides sufficient affordances with its animation objects to create Turing machines inside of a presentation. This means that one can execute any program inside of a PowerPoint slideshow, albeit quite slowly.[47]

To a traditional computer person, an end-user interface that is not Turing complete is not itself a computing machine. Beyond a certain point of abstraction, interfaces become too limited to satisfy the criteria of Turing completeness. The usability oriented design that emerged in the 1980s and 90s popularized many such applications, which are open to user interaction, but locked down so that those same users cannot create infinite loops and other crash-inducing scenarios.[48] Today, it is natural to presume that software is written by professional developers and consumed by hapless users. This trend is so entrenched that glimmers of end-user programmability, like Minecraft Redstone, Roblox Studio,

---

[47] Some of these unconventional Turing complete systems require a human (or equivalent) actor to manually step through the program. For example, in the case of PowerPoint, a human must click a sequence of highlighted buttons to walk through the computation. Without the human's participation, PowerPoint is unable to proceed to the next action. In this case the human acts something like a clock in a von Neumann architecture computer.

[48] Word, iTunes, Acrobat Reader, iPhoto, etc.

PlayStation Dreams, Apple Shortcuts, and Zapier Zaps are received as innovative software that unleash the power of programming—when really that power had to be deliberately obscured in the first place.

In contrast to Turing, some argue that any interaction with a computer is programming. In their 2019 workshop "Always Already Programming," artist and educator Melanie Hoff contends that the distinction between programmer and user is political, not technical. Hoff claims that the technology industry has a vested interest in separating programmers from users, because this puts programmers in a position of relative power [85]. Instead, Hoff states that user land activities like manipulating buttons in a smartphone app or naming and organizing folders in a file system are legitimate acts of programming. Like traditional programming, these user actions also affect the state of the machine and ultimately compile down to binary code running in the CPU. Furthermore, Hoff observes, the static classes and functions available in a high level programming language like JavaScript are prewritten code borrowed from others. If JavaScript developers rely upon the work of others to author code, then perhaps interface manipulation should count as programming, too. This looser definition of programming aims to reclaim agency for those disenfranchised by the Western patriarchal computation paradigm. From this decolonizing perspective, any action that results in code execution could be called programming.[49]

To people with computer science backgrounds, it may be difficult to accept the deconstruction of the term programming. Turing completeness and the definition of programming that it implies are useful concepts because they distinguish machines that can run any arbitrary program from machines that can only execute a finite set of actions defined by their creator. To this point of view, a high level application that exhibits no Turing complete affordances cannot be programmed, because computer programming is defining information processing algorithms in a symbolic language with a strict minimum of expressive freedom. Under the formal definition, an Instagram user is not a programmer, because they are only making use of limited features defined by that application's software developers.[50] By contrast, a program like Excel is intended for a large audience of end-users, but its affordances meet Turing's definition of programmability, and so it is considered by many to be the most popular programming language [86].

---

[49] What human interaction could mean in a world where the network is constantly passively reading and internalizing human actions as subtle as lingering on a picture, lingering in a doorway, or lingering in bed, is anyone's guess.

[50] Instagram may be Turing complete in a roundabout fashion, like PowerPoint, but let us put that aside, as such a proof would not fundamentally change how most participants make use of the network.

Instead of exploding the idea of programming as an act of resistance, I would like to propose an alternate resolution to this question of programming agency in consumer software.

Programming with Instagram

I propose that it *is* possible to program using high level interfaces, without having to change the meaning of programming altogether. If programming means *to create a sequence of branching instructions that a computing machine can execute*, then there are already many social media creators who successfully program, despite the Turing incompleteness of their preferred interfaces. How is this possible?

To program with social media, a person must engage the affordances native to the medium. Programming with social media does not look like programming with traditional abstractions like text or visual programming environments. In many social media, attention is the most important measure of success. To create software with social media, one must begin by programming content—that is to say, "programming" in the television sense of the term. In the genre of social media, creators are encouraged by the interface and the algorithm to create media that click with an audience. Successful social media creators accumulate large numbers of views, likes, streams, or platform-specific equivalents. On their own, these attention metrics are not able to conjure novel programs.

To turn media stardom into programming, creators must exchange attention for a more liquid virtual asset: money. Advertising allows creators to convert engagement into dollars. Creators can swap clout for cash with platform monetization tools, like Google AdSense, they can sell merchandise likes clothes and cosmetics, or they can make their own paid content arrangements, such as sponsored episodes, ad reads, and product endorsements. Whichever monetization scheme they choose, creators can use the money they earn to hire programmers, who can write the algorithms their employer desires. Financial abstraction allows commercially successful influencers to transform views into code by way of business.

Kim Kardashian, for instance, has proved that it is possible to create software with Instagram. Kardashian's path to fame is well known. She acquired an initial following through promotional appearances, reality television, and tabloid magnetism. As social media platforms like Instagram rose in popularity, Kardashian adapted her brand to fit the format's constraints. Weekly episodes of her TV show were complimented by round-the-clock selfies, fashion photoshoots, and related content. Kardashian monetized her audience with paid content sponsorships and developed her own makeup brand—techniques that have since become standard in the influencer business playbook. Kardashian then used her following and the money that she generated from these commercial endeavors to finance the development of smartphone software, such as Kimoji, a premium iOS keyboard stocked with Kim K-inspired illustrated images [87]. Like all iOS keyboards at the time, Kimoji was written in Objective-C with Apple's XCode programming environment and in accordance with the iOS Software Development Kit and App Store Review Guidelines. Kimoji sold for $1.99 and the app grossed approximately $3 million in its first year [88]. Talk of the app and its financial success traveled across the English-speaking internet.

Money is a virtual abstraction that social media influencers can use to engage computer programmers. Although Kardashian is not a programmer herself, she made use of the Turing *incomplete* affordances of social media to commission software which in turn created more media attention and revenues. Kardashian generated attention value on Instagram, captured that value through advertising, and then allocated that newfound capital to software development. Her software, in turn, fed back into her celebrity and netted her significant financial gains. To launch her popular Kimoji sticker set, Kardashian did not need to pick up an O'Reilly manual on app development, nor tune into WWDC to learn about the latest update to the iOS SDK. Instead, she simply used her wealth to partner with programmer specialists, who worked to complete the software development aspect of the project. This template is instructive. From attention, to monetization, to USD, to software development, to further monetization and attention accumulation. In high level interfaces such as social media, money provides liquidity that enables attention magnates like Kim K. to produce software.

To create software with Instagram's affordances, however, does not look like the kinds of programming most often association with computing. Instagram is not programmable by the strict computer science definition, but it nevertheless provides ample opportunity to people with entirely different skillsets to turn their ideas into legitimate software. Contrary to popular opinion, the value one can accumulate in this more computationally restricted environment *is* sufficient to create programs that run and execute according to the definition laid out by Turing and widely accepted by the computer community today. To program with Instagram, one must break out of the confines of the computational abstraction hierarchy, and use parallel virtual media, like money, to address lower level computational affordances.

Computer science traditionalists will no doubt find it difficult to call Kardashian a programmer, because she is not the one responsible for writing the code that is compiled and run. Kardashian is the only essential person in the creation of Kimoji, for instance, yet it will seem wrong to some to say that she is a programmer because she did not write the code. Social media is not considered a programming interface because financial instruments are not often regarded as programming interface primitives. However, if we reminisce on the origins of the medium, digital computation has actually always depended upon money as a means of wielding abstract algorithmic labour.

Computation is a form of labour

For hundreds of years, computing has been adopted first as a means of automation for cost saving and efficiency gains, then later as a medium for different forms of creative expression. Joseph Marie Jacquard's 1804 punch-card looms combined several prior inventions to create an automated manufacturing technology capable of accelerating weaving in France [89]. Jacquard's looms inspired Charles Babbage to seek funding from the British government to build calculating and computing machines that could outperform human computers in speed and accuracy [90, p. 6], [91].[51] At first, these proto-computing devices were invented to accelerate execution speed and obviate costly human labour. Like domesticated animals, single-purpose machines, and combustion engines before them, computers were first and foremost a technology aimed at reducing human labour input while increasing the desired output. Over time, early computing machines made way to modern electronic devices, whose incredible electric switching speed made possible new applications for which human computing labour would never have been applied.

The inverse is also true: humans are also routinely deployed to replace computer labour. In circumstances where software algorithms are not able to perform a sophisticated but desired task, technology organizations regularly deploy low-wage human labourers to fill in the gaps. Author Astra Taylor has proposed the term *Fauxtomation* for these algorithmic tasks that are performed by humans, but branded as if they are done by machines. Taylor argues that fauxtomation props up the fantasy that "machines are smarter than they really are" [92]. Technology organizations hide or omit the human element in their processes both to avoid drawing attention to their labour practices and to fortify the veneer of inscrutable scientific magic that often allows them to skirt regulations facing other industries. Although fauxtomation is an important and undoubtedly increasingly relevant framing, in the present text, I am less concerned with the theatrical staging of human work as software output. Instead, I am more interested in the interchangeability of human and computer labour, and what it reveals about computation.

Examples of humans doing labour associated with computation abound. At present, humans remain involved in many classification and data cleaning tasks. For example, humans perform social media

---

[51] As every introductory text mentions, computers were originally humans (usually women) who performed mathematical calculations.

content moderation, voice transcription, and image and video tagging. The most famous example is perhaps Google's ReCaptcha service, which claims to verify whether a connected user is a human or a bot through image classification tests [93]–[95]. While testing users, ReCaptcha also uses this semi-voluntary -to-read characters in scanned books, Street View pictures, and other data set cleaning work. In a similar vein, human transcription services like the now-defunct Jott, used human workers to transcribe audio messages [96], [97]. Amazon's Mechanical Turk (launched in 2005) provides a convenient API for just this type of on-demand labour. One 2017 research study found that Turkers earn a median hourly wage of $2 [98]–[100]. MTurk is a convenient hosted service for easily engaging human labourers for tiny tasks that elude software.

Human labour optimization has been a part of manufacturing since the early twentieth century, long before computers were introduced to the workplace. In the 1910s, industrial researchers advocated for process optimization techniques, which they called scientific management. In the 1910s, Frederick Winslow Taylor and Frank and Lillian Gilbreth developed new methodologies for breaking human labour into short tasks that could be optimized for execution speed [101]–[103]. Taylor consulted on the Ford Motor Company's adoption of moving assembly lines into their car manufacturing factories [104], [105]. These assembly lines increased production speed and output, which allowed Ford to reduce manufacturing costs. Instead of a single person building a whole car, these optimization techniques emphasized the division of labour into sequences of efficient repetitive tasks. Today, techniques for maximizing manufacturing operations with human workers is called methods engineering.

Methods engineering techniques are a form of abstraction that transform skilled craftsmanship into algorithmic labour. In an efficient factory, complex projects are broken into the smallest possible parts. Completing these tasks does not require the labourer, be they human or machine, to understand how the wholly realized output functions. This approach allows factories to decrease costs and increase outputs. It also makes it possible to treat employees like interchangeable parts, because the tasks they will be completing do not require skills unique to any one person—although they may be very physically and mentally taxing. This type of labour is naturally compatible with computational execution. More importantly, the "rationalization" of manufacturing into bite-sized tasks is itself a form of abstraction. In fact, we have already seen a series of examples in which human labour is transformed into computer commodities, and promptly forgotten altogether.

Financial Abstractions in Traditional Computer Sciences

Financial abstractions are endemic to all applied computation. When working with computers, it is impossible to avoid the division of labour into specialities. Whether programming or building hardware, one is constantly using money to outsource work to others. In practice, one most often uses off the shelf parts and software for every abstraction beyond one's immediate interest. Outside of the most idiosyncratic of hobby projects, growing silicon crystals and etching CPUs is left to professionals who benefit from massive scale. Even the most dedicated hardware hacker who builds computers from scratch will purchase transistors and integrated circuits [106]. To work with computers, one must constantly outsource labour to others—be that buying integrated circuits or cycles on a cloud server. No matter the circumstance, money allows a person working inside of one abstraction to ignore the details of all the rest.

Computer hardware reference books confirm that division of labour is essential to even understanding how a computer works. "We're not interested in the physical layer and the atomic layers," writes the author of the 656 page *Principles of Computer Hardware*, "because that's the province of the semiconductor engineer and physicist" [55, p. 206]. Another book, subtitled *Building a Modern Computer from First Principles*, justifies its entirely schematic approach to computer design when it states that "today, hardware designers no longer build anything with their bare hands. Instead, they plan and optimize the chip architecture on a computer workstation, using structured modeling formalisms like *Hardware Description Language*" [41, p. 14]. This book, too, spends no more than a handful of sentences on the chemistry and manufacturing used to make computers real. Whether relying on reified human labour from the supply chain or the App Store, "authentic" computer programmers and electronics makers are constantly stepping outside of the linear hierarchy of abstraction to spend money and buy their way to working computer results.

Computation has always been entwined with financial interfaces to specialty labour—we just do not talk about it that way. The hierarchy of abstraction has two purposes. As we discussed, it makes it easier to understand all the parts, because it hides the inner details behind helpful names (which serve as interfaces). At the same time, these abstractions enable any single computing device to contain parts sourced from hundreds or thousands of manufacturers. The supply chain is fully integrated into digital electronics. Supply chain manufacturers furnish packages of abstract human knowledge and workmanship in the form of components. Money is not just a programming language for the rich, it is electrical engineering for hobbyists and professionals alike. Money grants access to the works produced by processes I do not or cannot engage on my own. The intricacy and depth of interrelated computer abstractions make this not just a choice, but a necessity.

Programming, too, is open to mediation by financial interfaces. In the hardware stack, one pays money to acquire working components. Why then, should it be any different for software? Programming labour is just as open to abstraction as the work and skill involved in developing hardware components. There is no question that a C or LISP programmer is a legitimate author of software processes, despite their disinterest in the hardware underlying their craft. Not only is it legitimate, it is conventional within the discipline to ignore the human and computer practices required to make programming possible. Following this logic through to its conclusion leads us to the realization that even a high level interface can be used to indirectly write and run software code by way of money.

Finance and computation are parallel and complementary virtual systems. From its origins in weaving, government, and military research, computation has always been bound up with money. In the preceding pages, I have argued that the same logic should be allowed of the highest level interfaces. If an application is closed to programing, but open to commercial extraction, then creative thinkers will find ways to exploit that escape hatch. If one finds oneself stuck in an abstraction with limited expressive horizons, it is perfectly legitimate to hack one’s way to the goal. Computation and financial value are neighbouring virtual towers of abstraction that have been strapped together from the start. Their proximity makes it possible to escape and jump from one virtual edifice to the other, only to step back in at whichever level we want. Acknowledging trap doors and opportune scaffolding between virtual architectures like these make us sharper network thinkers.

Money is a programming language, in the sense that it can be used to symbolically orchestrate programming itself, just like so many other parts of the digital computational stack with which we are familiar. Avowing this opens the door to a theme running through this and every other chapter of the present dissertation. The fundamental design challenge for writing network software is to create mechanisms that ensure the viability of a software authoring organism over time. By introducing money as a programming language, we have begun to acknowledge that the role of design in the making of software is not just the color palette, nor even the data structure, but the nature of the engine for paying the rent and expenses of the labourers. Computation and the network organisms I will discuss in subsequent chapters are tied just as much to the traditions of computer science as they are to the market and the media.

Chapter 2: Algorithm Theatre

Computation provides creators with endless open space. Software is a fluid medium that can ingest or output just about any material. When working with such an intangible virtual substrate, creators, myself included, can easily inherit assertions about the medium. Prior art can be the mark on the page that allows a creator to get moving, but at the same time, the dominant ways of talking about software can limit our collective imagination.

In this chapter, I argue for the similarities between network software making and the performance arts. I challenge some broadly held assumptions about the stylistic nature of creating software, and explore how alternate language and conceptual frameworks might open the door to new experiments, products, and art.

Performing Network Software

Programming Metaphors: Literature v. Performance

Software is a performance art. Despite other, more popular metaphors and analogies for the process of creating software, I believe that performance is a helpful and refreshing way to think about making code and establishing computation-oriented institutions. In this section, I will present three ways in which the practice of making network software resembles performance arts. First, I will discuss the role of revision and repetition in software making. Second, I will examine audience participation in network building. Third and finally, I will explore how network-building institutions engage consumer emotions through theatrical gestures.

It is easy to think of software development by analogy to writing. One unintentional consequence of this cultural association is that we speak about the software making practice using terminology borrowed from literary publishing. In fiction, code is usually depicted as text typed into a computer on a keyboard. Similar characterizations are common amongst practitioners, too. A programmer is someone who “writes” code. The creator of a piece of software is often called its “author,” and the culmination of a project results in its “publishing” to the web or App Store. The phrase “programming languages” suggests a written syntax for describing software.[52] Visually, this phrase evokes stock photography images of cryptic computer

---

[52] Although “programming language” might just as easily evoke verbal interfaces. For now, that association ranks far behind the dominant textual one.

syntaxes and technical-looking indentation schemes, perhaps represented by light-coloured text on a dark coloured screen. In sum, computer programming today is associated with text.

In fact, programming can just as easily be visual, physical, or any other modality one might imagine. As we saw in Chapter 1, what counts as programming is any arrangement of symbols that can be translated into sequences of instructions for a computing machine to execute. This is borne out by alternative interfaces to code today. There are node-based programming environments, such as Scratch, Unreal Blueprints, and Max, which enable the programmer to create software projects by connecting block-like nodes to one another [107]–[109]. There are also physical programming environments like Dynamicland, littleBits, and Google Bloks, each of which create tangible means for creating software [110]–[112]. These projects demonstrate that the link between software creation and writing is the product of a particular set of historical contingencies. Visual and physical modalities are just two options among an endless range of symbolic representations for programming.

In practice, creating network software is much more like performance art than prose publishing, because network software must constantly be performed anew to continue working. When writing a book, an author works with an editor to create a body of text that can be published. Although there may be minor adjustments in subsequent reprints or digital corrections, a traditional book is finished on the day it is published. If the author has more ideas to add, they can create an expanded second edition or write a new book altogether. Either way, the first edition and first pressing continue to function as intended at the time of publishing. This is exactly unlike network software. Software that is intended to run on the network inevitably depends upon myriad third-party services and standards to function. Over time, the protocols, operating systems, and distribution schemes upon which a piece of software relies will change. To continue working, the software must be updated to match the new circumstances in which it must run.

Software is like a recurring performance, because it must be regularly maintained to continue operating as intended. For instance, in 2018, Google Chrome introduced a new feature that would block webpages from playing audio unless the user demonstrated an intention to trigger an audio event with a mouse click. This restrictive feature was introduced to cut down on intrusive and manipulative advertising. The change was criticized by game and audiobook webapp developers, because it broke websites that did not update their code to match the new Chrome guidelines [113]. For active developers, the change required them to fix their webapps to conform to the new rules. For unmaintained sites, the change simply broke the audio aspects of those pages. In some cases, reengineering the application to conform to the rules would have been too lengthy a process for developers who had moved on to other projects. For projects with no remaining maintainer, the change broke those sites permanently.

Breaking changes like these happen on most platforms. Backwards incompatible updates to web browsers, web protocols, standards, and security practices are practically unavoidable. On proprietary software stacks, like mobile and PC operating systems, breaking changes are even more common. Every few years, Apple, for instance, requires that developers update their apps to support new screen resolutions, new processor instruction sets, or new legal documents like privacy policies. Non-compliant apps are removed in short order. At any moment, breaking changes in the platform, or other services upon which the software depends, threaten its operation. Unlike a book, which continues to work no matter how grammar or printing technology changes, network software relies upon the care and attention of willing developers. In all of these circumstances, the software developer's role is much more akin to that of a theatre company that puts on a show anew each night of the week. If the performers and supporting staff do not show up, the show does not go on. If the venue or city impose new requirements upon the production, the theatre company must adapt and adjust their work to fit the novel setting. Although a lot of software is written as text, its upkeep looks far more like production repetitions. Should the programmer(s) choose not to show up and perform, the network software will stop.

Network software that relies upon user engagement to thrive is especially like performance arts, because the product depends upon audience reaction. Nowhere is this more obvious than in social properties, such as social media, communications networks, and crowd-sourced projects. TikTok, Wikipedia, Apple, and others all depend upon their audiences' continued interest to sustain their development work and upkeep. Regardless of their business-model, these *network organisms* must maintain a keen interest in their audience's reaction to their output.[53] The relationship is perhaps most like that of a DJ and dancing concert-goers. The DJ is responsible for both bringing an energy that excites and entices the public, and also adjusting quickly should their musical choices diminish the audience's interest. The DJ and audience are in a relationship of constant feedback. Either party on their own is not sufficient to create a lively event. Together, they can create happenings of previously unimaginable social consequence. The same is true of networks makers and network participants.

---

[53] To keep things brief, I will hereafter refer to the superset of network companies, non-profits, and open source initiatives as *network organisms.*

Internet-native networks are especially reactive to audience behavior. Facebook, for instance, pioneered popular algorithmic attention products with their News Feed, which launched in 2006 [114]. With News Feed, Facebook introduced to their users a content aggregator that drew upon user-behavior to decide what to show next, rather than pure chronology. In contrast to blogs and RSS readers that had come before, News Feed's powerful recommendation engine took into consideration content clickthrough, likes, comments, and time spent when selecting what to show. News Feed was a powerful paradigm shift in network software design that made real time user engagement data a crucial ingredient for future products. More recently, ByteDance's TikTok has demonstrated that there remains room for software to grow even more addictive through user attention metrics and machine learning product design. TikTok's recommendation algorithm, which is based upon sister product Toutiao's news aggregating algorithm, has proved extremely effective at selecting the most engaging videos among the millions uploaded each day. TikTok combines traditional computational, machine learning, and human labor approaches to analyze and categorize videos by content [115]. TikTok's For You page measures user engagement such as likes, comments, and video completion to create a feed of unparalleled attention-grabbing short videos [116]. In these cases and others, network software products demonstrate superhuman reactivity to audience engagement. While authors may pay attention to their readership to some extent, the greater parallels to performance arts are clear.

Performing Institution Building

In her 1991 book *Computers as Theatre*, Brenda Laurel showed that the dramatic arts can be a helpful influence when designing software interfaces. Laurel's research is the confluence of her formal training in the theatre and her initially accidental career in the then-burgeoning field of user experience design [117, p. 16]. Throughout the late-1970s and 1980s, graphical user interfaces and consumer friendly computers like the original Apple Macintosh popularized the ideas of interface design, user experience, user research, and product usability [117, p. 3]. Laurel conducted research and product development at Atari, Apple, Activision, and Interval Research Corporation, where she developed her unique perspective on interface. In *Computers as Theatre*, Laurel argues that user experience designers can glean useful insights from the dramatic tradition. Beginning with classical Western dramaturgy, Laurel draws out language and analytical categories, then considers how each can be applied in HCI research.[54] Motivation, intention, potential,

---

[54] HCI is Human-Computer Interaction, the academic disciplinary framing of research about human and computer interfaces.

action, enactment, pattern, and time are just a few of the conceptual tools theatre scholars have developed to dissect a play. Laurel provides countless examples of dramatic elements and evidence of exemplars in software history.

I contend that Laurel did not go far enough, and that really the most crucial theatrical element of creating software is the staging of the organization that performs and reperforms the project itself. In the 1980s and 1990s, the relatively young field of interface design was ripe for more humanistic influences like Laurel's. Decades later, however, the focus for software building has necessarily shifted. Today, computation corporations concentrate wealth, power, and social influence like never before. In this moment, it is urgent that attention to theatrical gestures move from an individual product's usability to the design of the organizations that create these world-permeating products in the first place.

This meta-performance, of creating and positioning a *network institution,* is the most intriguing and undervalued performance art at this time. As I discussed in the previous section, network software most often requires developers to perform their work anew all the time, forever adjusting to the dynamic environment around them. Whether due to changes in dependencies or market conditions, network software makers must constantly react and update their work to fit the present and future. As such, the most important ingredient in launching and maintaining a network-infused project is to secure the interest of those people who can perform the software over time. Whether they be paid employees or open source contributors, it is of the utmost importance that the people who *make* the network be excited and engaged. This is not the type of performance art one is likely to find in an art school course catalog, and yet it demands the very same creativity to which artistic vitality is attuned. To accomplish this feat of building network organisms, one must venture into the psychological domain of brand.

A network organism needs an enthusiasm engine to inspire passionate volunteers or compensate employees. Regardless of their business model or institutional design, network organisms live and propagate thanks to enthusiastic acolytes and peripheral believers. Let us consider three types of organizations: a non-profit, a company, and a cryptocurrency. A non-profit network organism, like the Wikimedia Foundation, survives by monetary donations, which fund development and administrative costs, and volunteer labour, which creates Wikipedia's content. In the case of Wikipedia, the mission to create a free, public-editable, open encyclopedia touches many people who happily contribute one way or another. In the case of a company, such as 23andMe, the promise of an ambitious and potentially lucrative future business motivates investors and employees alike to get involved. If they are convinced by its mission, plan, and competency, investors will trade the money they manage for shares in the company. 23andMe's founders and executives can deploy this money to hire employees who can make the vision a reality. The stronger the company myth, the better the talent pool that will be available, and the more those employees will want to be compensated in company stock, instead of cash. Equity compensation arrangements like this allow the company to create golden handcuff clauses, which improve employee retention.[55] In the case of a company like this, a bold mission and promising financial prospects come together to create the enthusiasm engine at the heart of the institution. Finally, in the case of a cryptocurrency, similar mechanics are at work. In cryptocurrency, early adopters, volunteers, and investors are motivated by faith—or perhaps fear—that a new digital technology may transform the nature of assets and value altogether. In this third case, mythic hype drives all parties to participate in the creation of a new empire of value.

Cryptocurrencies exhibit some especially interesting growth mechanics, because they create the opportunity for early adopters to invest in their obsessions. Traditionally, being an early adopter has many downsides. Early adopters implicate themselves in technologies that may be imperfect, too early, or that will never pan out. Being too early to a technology or social phenomenon is also socially unprofitable, as the mainstream is not yet prepared to appreciate the foresight. Participating in a technology before it has "crossed the chasm" and reached early mainstream adopters, to borrow Everett M. Rogers' Technology Adoption Lifecycle terminology, opens an early adopter to criticism and social ostracization [118]. Cryptocurrencies turn these disadvantages around and mobilize early adoption two-fold. First, cryptocurrency early adopters stand to profit enormously, should the technology become popular down the road. Second, cryptocurrencies that are scarce assets also incentivize early adopters to promote the network through word of mouth, because more adoption increases the value of the early adopter's holdings. In both ways, early crypto adopters are financially motivated to suffer the hardships of public support for an as-yet unproved technology.

Elon Musk's approach to funding The Boring Company demonstrates how theatrical staging can bring ambitious goals within reach. According to his own retelling, Musk came up with the idea for The Boring Company due to his frustration with LA traffic [119]. To solve the gridlock, Musk reasoned that it would be necessary to build three dimensional layers of highways underground [120]. To make that possible, The Boring Company was founded first as a small project at SpaceX, then spun out into its own company

---

[55] Golden handcuffs are compensation arrangements that require employees to stay with their employer for a given period before they have access to their allotted stock or other compensation.

in 2016 [121]. Its goal is to build boring robots that can quickly and cheaply dig 3D networks of tunnels underneath big cities to alleviate road traffic and free up space on the surface for things other than cars. The company was officially created in 2016, but it was in 2018 that Musk performed the great gambit of futurist theatre that I would like to discuss in detail [122].

In January 2018, The Boring Company opened pre-orders for a limited edition collection of flamethrowers [123]. The flamethrower product launch followed hot on the heels of the company selling out of 50,000 baseball caps emblazoned with the company's name. The $20 hats netted the company $1 million, and inspired them to launch a follow up product. Musk took to Twitter on December 10, 2017 to entice his fans with a flamethrower pre-announcement [124]. The company, he implied, would soon begin selling assault rifle-style flamethrowers with a distinctly cinematic science fiction aesthetic. The tweets caused a sensation on social media and in the press. Pre-orders opened on January 27, 2018, and less than a week later the company had sold out of all 20,000 units. At $500 per flamethrower, the company grossed $10 million in sales with no significant marketing budget. Within a few days, journalists had figured out that the flamethrower not only failed to meet the technical definition of its namesake, but that it was actually composed of a standard issue roofing torch packaged in the enclosure of an airsoft gun [125], [126].

The Boring Company Not A Flamethrower spectacle is exemplary of the performative latitude open to creative network makers. This limited edition product used theatrical gestures like hype, scarcity, and collectability to emotionally engage the audience and motivate them to part with their money. Musk's persona and the pitch-perfect positioning of the product encouraged fans to identify with his heroic quest to improve the world through technology companies, while also baiting those susceptible to collector logics to join in funding his newest operation. Musk further excited his audience with self-aware humour like his January 28 tweet where he said "The rumor that I'm secretly creating a zombie apocalypse to generate demand for flamethrowers is completely false" [127]. To buy a Boring Company flamethrower was exciting, funny, crazy, thrilling, insidery, optimistic, and more.

The Boring Company is not a network technology company by any traditional definition, but to anyone with an open mind, it is clear that this gambit, and the network organism that Musk and his associates intended to set up, are deeply integrated with network software. The product was hyped on Twitter, sold online, and manufactured by strapping together two commodities issued from the supply chain, then branding them as something else with altogether greater viral potential. The flamethrowers became a meme and a comedic symbol of reckless young wealth. It is a recurring gag, for instance, in Teen Choice Award winner David Dobrik's YouTube videos.

The Boring Company flamethrower is a testament to Musk's marketing acumen, and his intuition for performing in the style of algorithmic theatre. Musk could no doubt fund the company by other means; a few months later, in fact, The Boring Company announced that it had raised over a hundred million dollars in venture backing [121], [128], [129]. More important, however, is the founding team's impressive insight into the staging and performance of futuristic meme theatre. Regardless of alternative sources of money, or the product they ultimately ship, Musk's Boring Company succeeded in capturing media attention through a lucrative gag that made it by far the most recognizable of tunnel digging brands in the world. This notoriety undoubtedly helps this modern network organism find future employees, investors, inbound sales inquiries, and eventually public shareholder enthusiasm. This is a trick Musk has repeated for each of his ventures since he exited PayPal, and it is as important aspect of his success.

Chapter 3: Fashion and Attention

As we saw in Chapter 2, making network software is a performance art. Network software must be performed over and over as it forever becomes obsolete. Its form and behavior must respond to the audience, sometimes many times per second. Perhaps most performance-like of all, network makers must improvise a convincing theatre act to create a viable institution that can steward their creation into long term viability.

If network making is a performance, then creating fashions and commanding attention are its stagecraft. This chapter deals with perception and immersion. In section one, I will describe the role of fashion in positioning networks in the public consciousness. This section focuses on the similarities between social networks and popular content. Despite superficial differences, I will argue that these two phenomena are in fact, quite alike. Section 2 is dedicated to *attention optimizing design*, the brand of design that many network organisms are incentivized to pursue. In this portion of the text, I will provide background information about traditional social media networks' drive to capture user attention, and how this informs their design choices. This section closes with an argument that even paid entertainment services are driven to monopolize subscribers' time, and so these networks, too, are incentivized to pursue attention optimizing designs.

Section 1: Fashion and Fame

Social Networking Platforms Are Memes

"Content" is what software barons call everything that is not software. Founders and investors in popular social networks often frame the contemporary media landscape as divided into *platforms* and *content*. Facebook, for instance, is a platform, and everything that is shared on Facebook is content. According to this scheme, social network platforms are infrastructure, or even utilities, while everything else is relegated to the all-consuming category of content [130], [131]. If we accept the software barons' framing, then human activity is suddenly divided into these two stark categories. There are social network platform makers, who reign over the means of communication, and there are content creators, who struggle to achieve sufficient notoriety to stand out amidst endless waves of competing signals.

The term platform can be a bit confusing, because it means different things in different disciplines. In economics, platforms are markets where buyers and sellers coordinate trade, whereas in engineering, platforms are modular technological architectures that can be recombined for new applications [132]. In consumer software, networks are called platforms for speech at some times, and platforms for the standardized presentation of information at others [133], [134]. The term platform has been adopted by the software industry for at least two reasons. First, its ambiguous definition provides sufficient malleability to adapt to the changing role networks have both in our lives and in the market. Second, and perhaps more critically, content distribution platforms have so far been exempt from legal liability for the content that third parties publish through their services [135]. In contrast to a book publisher, for instance, YouTube and Google have not been liable for the media they list, so long as they comply with legally binding takedown requests.

Contrary to the worldview proposed by the software elite, social networking platforms are actually content, too. Social networks are not simply utilities or mere pipes through which media objects flow. Social networks themselves are subject to the whims of public attention, just like the content from which software makers so often try to distance themselves. Despite their best efforts to suggest that social networks platforms are completely different to content, I will argue that *social network platforms are memes*, in the ways that they become known, grow popular, and decline. The network platform is itself a meme in the minds of prospective and active users.

Booting and sustaining successful network platforms requires creators to engage people's attention and interest, much like in the world of fashion. To grow from nothing to long-term viability, networks must employ the very same dynamics that popular social content does. Some networks, like Instagram, follow the lead of luxury fashion and goods when they promise active participants a chance to flaunt their lifestyles in public. Others, like Raya and Clubhouse, sell elite exclusivity through invite-only schemes [136], [137]. Meanwhile, networks like Kuaishou simply pay users to watch and post videos—a sure-fire strategy to draw public interest, at least [138]. Let us consider a couple of examples of how networks participate in trends to kick off viral growth, and how networks can just as easily be supplanted by more aggressively fashionable upstarts.

Musically, the network that would later become TikTok, was not always so popular. In 2014, co-founders Alex Zhu and Luyu Yang realized that Cicada, their short-form video education platform, had failed to gain traction. According to company legend, Zhu decided to pivot from education to entertainment while taking the Caltrain between San Francisco and Palo Alto [139], [140]. On the train, Zhu observed a group of teenagers making videos together on their phones. While one person performed, another recorded, and another found a song for the soundtrack. Zhu realized that if they could create a social app to facilitate collaborative music video making, they might just have a hit. The team reworked the project and relaunched it in August 2014 as Musically, an app for making music videos with friends. The new app attracted users, but it was not an instant success.

Musically achieved viral growth when it focused on lip syncing, then multiplied this advantage by amplifying growing memes to the whole app's user base. In the spring of 2015, Zhu and Yang discovered that app installs were spiking on Thursday nights [140]. They dug into the numbers and determined that the weekly influx of new users corresponded to the airing of a US primetime television show called Lip Sync Battle. Musically had been lucky. The team had included "lip sync" in the keywords of their App Store listing, and the new users had downloaded it by search result ranking chance. With this information in hand, the team zeroed in on the lip syncing use-case and updated their branding and features to match. To further increase user content generation, Musically developed featured hashtags called challenges, which encouraged users to lip sync a particular song or dance to a given audio track. The app prompted users with push notifications about new challenges, which created popular memes within the community and increased the app's stickiness. The moves to lip syncing and challenge prompts worked, and the app began to grow rapidly among American teenagers. Within two months Musically was the number one most downloaded app in the App Store.

For Musically, combining the right technologies was not enough; to break through and create a viable network, the product team had to focus on a popular trend. Through two prior iterations, Musically's founders struggled to find a product that clicked with an audience. The technologies they chose were primed for success. Short videos, quick editing, and social sharing were all correct bets, but these ingredients were not enough to kick start a network when pitched as an education streaming app, nor as a general purpose music video platform. The founders' unique insight was to focus their efforts on a specific niche to build a positive feedback loop that yielded content generation and increasing user adoption. Once they had nailed this formula, it was possible to grow and eventually expand into neighbouring categories.

Almost two decades earlier, Hotmail achieved a similar growth trajectory with its pioneering approach, which was later dubbed *viral marketing*. Sabeer Bhatia and Jack Smith founded Hotmail in 1996. Faced with firewalls that blocked them from connecting to their personal emails at work, Bhatia and Smith came up with the idea to create a free web-based email client—the first of its kind—which would allow them to access their emails from any browser [141, pp. 19–20]. To grow the service, Smith added a line at the bottom of every email saying "This message has been sent from Hotmail. Get your free email at hotmail.com" [141, p. 22]. Each email sent with Hotmail was also simultaneously an advertisement for the service. As more users joined, more ads were dispatched to their closest friends, and the network grew exponentially [142]. This simple tagline spawned the field of viral marketing. Without spending any money on traditional advertising, Hotmail grew from zero to 12 million users in a year and a half.[56]

Despite early success, Hotmail's failure to deliver sustained technological improvements ultimately cost it its position as the fashionable market leader only five years later. Google employee Paul Buchheit began work on Gmail in August 2001 [143]. Googlers at the time were inundated by email, years before a similar torrent would come to affect white collar workers outside of the technology industry. Buchheit, who had previously led the development of Google Groups, started by borrowing the fast search technology developed for his prior project [143]. With powerful search in hand, it became obvious that one would want to store an archive of emails, rather than deleting them or only storing them locally on the client. This led Buchheit and the engineers that joined the team to grant users an unprecedented 1GB of storage, so that they would never have to delete an email. This was 500 times as much storage as Hotmail provided [144, p. 153]. In addition to superior search and storage space, the third and final functional innovation in Gmail was its interface, which was written in JavaScript. As opposed to Hotmail and its contemporaries, which boasted pure HTML interfaces that reloaded the entire page with every click, Gmail was built with cutting-edge Asynchronous Javascript and XML (AJAX) techniques, which allowed the page to feel and respond more like a native application than a clunky webpage [143]. In addition to all these technical innovations, one final twist to early Gmail was perhaps most instrumental to creating buzz around the product.

Google released Gmail to the public on April 1, 2004 under an invite-only system designed initially to limit traffic to their unfinished service. According to people who worked on the project, Google rushed to finish Gmail so it could be announced on April 1st as one of their annual April Fool's jokes. To meet the deadline, the Google team had to make due with underpowered servers. At launch, Google hosted Gmail on three hundred outdated Pentium III computers that could not handle too many users [143]. The team invited 1000 influential people in technology and journalism to join the service, and granted each of them a limited number of Gmail invitations so that they could fix any uncaught bugs and scale the servers to match controlled user demand [144, p. 154]. The invitation scheme and oddly timed announcement spurred intrigue and confusion among journalists and the public alike. The service also faced a privacy backlash over plans to scan emails for advertising purposes and to retain emails indefinitely [145]. In the

---

[56] Impressive numbers at the time.

long term, Gmail weathered the privacy storm, but the combined press attention it and the limited invites caused only increased the public's desire for access. In the week following the announcement, Gmail invites fetched as much as $200 on Ebay [146]. Together, these intentional and unintentional PR maneuvers generated huge excitement and rapid user growth. Gmail dropped its invitation scheme and opened sign-ups to the public in February 2007 [143]. It took six more years for Gmail to surpass Hotmail's user base, but it had already eclipsed Hotmail in terms of cache [147]. By October 2018, Gmail had accumulated 1.5 billion users, making it the most popular email service in the world [148].

In practice, networks are not just raw technologies, they are fashion objects subject to the whims of the audience. Just like memes, networks become popular and fade from relevance according to the public's perception of them as compared to competitors and the broader experiential landscape. Despite all the propaganda work the emissaries of current networks do in the press, the networks of today are not immovable utilities. Like the content in which they traffic, networks are themselves vulnerable to the opinions and behavior of their participants. To prospective and active participants, networks *are* memes. How one relates to a network depends on how one believes others perceive it, and how one perceives it oneself. Its name, logo, interface, and active users are all represented symbolically in the minds of the people, just like any concept or meme. A network can be the cool new kid on the block (e.g., Snapchat circa 2012), a place clogged with relatives (Facebook) or overrun with marketers (LinkedIn). Each of these aspects of public perception affect uptake and the behavior of those online. It is to be expected that dominant players will contend that they exist outside of the race for attention.

Memes are Networks

If networks are systems of interconnected nodes or interconnected groups of people, then perhaps memes are networks, too [149]. Today, memes are often associated with internet images and short videos, but the term's original meaning is a unit of shared culture [150]. A meme is a shared concept, be it an internet joke, a melody, or a belief. Memes are inherently social, because they exist in multiple minds at once, and they travel and propagate through communication. Memes also create networks, because they provide the shared context that structures new interaction. If I arrive at a party of strangers and loudly mention Joe Rogan, the ears of those people within earshot are bound to perk up. Whether others react positively or negatively, awareness of the meme creates new opportunities for communication. Shared concepts like notable people, events, and news are loci for social connection. By dint of their innate sociality, it is clear that memes form networks, too.

In the early twentieth century, long before the internet emerged, entertainment, business, and fashion discovered that celebrity has the power to draw people's attention and affect their decisions. The history of celebrity, marketing, and fashion in the twentieth century track humanity's increasing awareness that shared culture affects behavior. In this section, I will describe that history and argue that today's hegemonic software networks are just the latest in a lineage of attention-grabbing fashion phenomena. The origins of fame and influence show how prior network authors schemed to create and capture the value of human attention. The history of attention manipulation is instructive to network makers today, because network are memes and memes are networks.

The history of modern celebrity began with the invention of film stars in the early twentieth century. Motion picture film technology was invented in 1888, and for the first 22 years of cinema history, being a movie actor did not imply celebrity. At first, people in movies were small figures on the screen, the details of which were difficult to discern [151]. Film celebrities in this early period were people who had already achieved fame in other parts of life during the nineteenth century. Early movies offered a chance to see these people in motion, but cinema did not yet create its own new stars. Famed French theatre actor Sarah Bernhardt, cowboy showman Buffalo Bill, and escape artist Harry Houdini transitioned to film roles, but their fame was decidedly pre-cinema, according to film scholar Clive James [151]. In its infancy, filmmaking was unexplored territory that required new storytelling techniques. It took a full decade after the invention of motion pictures for filmmakers to invent the modern cut, where action continues from one shot into another [152]. It wasn't until two years later that the most important element of cinematic language would be invented.

The close-up was invented in 1900, and it changed fame forever. A film named *Grandma's Reading Glass* introduced the world to the cinematic close-up, which made actor's faces larger than life [153]. In the absence of recorded audio, which wouldn't reach movie theatres until 1927, close-ups made silent films relatable and immersive. Close-ups transformed cinema from a medium whose output looked like recorded theatre productions into the modern emotional medium we know today. On the silver screen, a close-up of an actor was larger than life [151]. Zoomed-in shots of actors' faces encouraged audiences to romanticize the narratives and the actors, too. Although this made actors into heartthrobs and heroes, those who were not already famous remained unnamed in the credits. Florence Lawrence, for instance, acted in 50 movies for D.W. Griffith's Biogaph Company between her debut in 1906 and 1909 [154]. Still, audiences called her "the Biograph girl," because they knew her face, but not her name [154].

In 1910, Carl Laemmle invented modern Hollywood celebrity. Laemmle recognized that actors drew audiences to the theatres, and that there was a financial opportunity in playing into the audience's infatuation. Laemmle lured Florence Lawrence away from D.W. Griffith's Biograph Company with promises of better pay and marquee billing for her films [155, p. 32]. He then orchestrated an elaborate publicity stunt. First, he spread rumours that Lawrence had died in a streetcar accident. Then, Lammle gave interviews to the press where he claimed that Lawrence's death was the "silliest lie yet circulated by enemies of the IMP," his new production company [154]. He used the sensational news articles to announce that Lawrence was in fact starring in a new IMP film. When she appeared in St. Louis, the world's first movie star was accosted by adoring fans who had bought fully into Laemmle's ruse. The movie producer, who would eventually found Universal Pictures, was good to his word. He made Lawrence the first Hollywood celebrity and increased her pay from $25 to $500 per week [154].

Florence Lawrence's celebrity signaled the beginning of the Hollywood star system, a marketing scheme designed cross-promote movies and increase revenues. Gone were the days of unnamed movie actors. As of 1910, movie studios recognized that audiences were drawn to the movie theatre by their interest in specific actors. Following Laemmle, other studios fostered their own movie stars to create obsessive fan followings for famous actors. The studios agreed to pay these actors much larger salaries in exchange for exclusivity arrangements which barred the actors from working with other studios without permission from their primary employer [151]. Under the star system, studios also managed the actors' personal lives and publicity; they produced backstories, romantic relationships, and glamorous private lives for the actors to increase their audience appeal.

The star system made it possible to connect disparate intellectual property through a web of familiar faces. Developments in film language, including the close-up and continuity between takes, made it natural for audiences to form irrational emotional relationships to the projected images [151]. Clever Hollywood businessmen used this spontaneous and irresistible passion to build oligopolistic control of the most advanced media technology at the time. Like hypermedia today, celebrity faces and the emotional connection they cast upon their audiences played a network-like role in attracting attention. In the second decade of the twentieth century, the invention of movie celebrity demonstrated that popular images and ideas, like the romantic icons of that era's cinema, could drive attention and lucrative changes in behaviour.

In the 1930s, Edward Bernays drew upon Freudian psychoanalysis to create advertising campaigns that used people's emotions to drive changes in consumer behavior. Bernays was born in Austria and moved to New York with his family as a teenager. He was the nephew of psychoanalysis pioneer Sigmund Freud. During the First World War, Bernays helped propagandize the American government's efforts to liberate humanity from war and "make the world safe for democracy" [151]. After the war, his uncle Freud sent Bernays a copy of his *General Introduction to Psychoanalysis*. The book, alongside his wartime experiences, convinced Bernays to apply the skills he had learned as a propagandist during peacetime. Propaganda had acquired a negative connotation during the war, so Bernays created an organization called The Council on Public Relations—the first time the phrase *public relations* was used—and set to work applying his skills to business problems [151].

Bernays used his knowledge of psychoanalysis and his intuition for mass behavior to associate people's unconscious desires with mass produced consumer goods. In the late 1920s, the president of the American Tobacco Corporation asked Bernays if he could find a way to break the taboo against women smoking in public, so that he could double the total addressable market for cigarettes [151]. Bernays consulted a respected psychoanalyst named Abraham Arden Bril, who told him that, to women, cigarettes represented male sexual power. To get women to smoke cigarettes, he would have to associate smoking with resisting male domination. Bernays staged his most celebrated public relations stunt at the 1929 Easter Day Parade in New York City. He convinced a group of debutantes to smuggle cigarettes into the parade under their clothes. Bernays leaked a rumour to the press that a group of suffragettes was planning to stage a protest at the parade by lighting "torches of freedom" [151]. At his mark, the young women took out their cigarettes and lit them simultaneously. The photojournalists eagerly took pictures that ran in newspapers around the country the next day under Bernays' prepared phrase, "torches of freedom" [151]. Bernays had successfully associated the irresistible symbol of young women smoking with a political position of women's liberation and freedom. The ploy was a success and it became more socially acceptable for women to smoke in public thereafter.

Bernays taught American businesses that people make buying decisions based on emotions, not facts. When he created public relations, most marketing sold products on the basis of factual information. Until the propagandistic innovation in the late 1920s, marketing appealed to potential customers as if they were rational agents making decisions informed mostly by the objective properties of one or another product. Bernays showed that, to make a sale, a marketer should speak to a person's unconscious feelings and desires instead. People are more likely to respond, Bernays proved, to products whose framing corresponds to their fears and dreams, than to products sold solely upon their physical merits. Smoking, for instance, contributed nothing material to women's liberation. It was irrational to associate an addictive product with personal freedom, but the symbolic association he created overpowered such logic. Over time, advertisers grew more sophisticated and learned to not only capitalize upon people's existing unconscious desires, but also to install new fears and expectations that matched clients' new product categories. Of course, consuming marketed products can never really make a person satisfied, because such a person will always be vulnerable to the next image of their own lacking satisfaction.

In the middle-twentieth century, fashion editor Diana Vreeland used artistic production to transform fashion in America into a genre of mythical storytelling. Vreeland was famous for her quick wit and unique perspective. She began her career as a columnist at *Harper's Bazaar* in 1936 [156]. Her column "Why Don't You…" caught readers' attention with its eccentric suggestions. "Why don't you…," wrote Vreeland, "wash your blond child's hair in dead champagne, as they do in France" [157, p. 92]. Her writing conveyed with perfect clarity the fanciful way she thought and spoke every day. Vreeland excelled at painting pictures of exceptional ways of living, and she was happy to bend the truth to give the reader something they could not get at home [158]. In the words of model and collaborator Veruschka von Lehndorff, Vreeland believed "don't tell a story, even if it's true, if it's boring. Invent something!" [158]. She was promoted to magazine fashion editor one year after she started writing her column, and she remained in that position until 1962 when she became editor in chief of *Vogue* [156].

Vreeland believed that fashion magazines could do more than just peddle clothes, they could create rich worlds full of passion and romance. When she took charge of Vogue in 1962, she reinvented the stodgy society magazine as a fabulous romp through the lives of artists and musicians, and the voyages of beautiful models to exotic locales [159]. Richard Avedon described her editorial direction as anecdotal, rather than explicit. On a shoot in Egypt, Vreeland sent him a memo in which she said, "Before you plan these pictures, just think about Cleopatra. Think about those hot nights. She's walking on the roof, everybody is *so old*, and she's just pacing that roof!" [158]. "She just sort of threw you a way of thinking," he reflected [158]. Vreeland often described her approach to magazine photography and layout with the phrase "the eye has to travel." She used color, text, and surprising juxtapositions to give rhythm to the story and make the subject leap off the page. It was only natural that larger than life actors would enhance the luxury of haute couture, and that stunning models would become memorable celebrities, in turn.

Vreeland glamourized American fashion by making celebrities into models and models into celebrities. Vreeland perceived earlier than most that celebrity and fashion were on an inevitable collision course. The fantasy lives of movie stars were made of the same stuff as the dreamlike fashion stories her team produced all over the world. She embraced the sixties obsession with human personality and encouraged photographers to make each model's idiosyncrasies their signature feature [158]. She turned the gap in model Lauren Hutton's teeth into an advantage, and she celebrated Barbara Streisand's famously unbutton-like nose with photographs shot in profile [158]. She believed that "at the same time mannequins became personalities in the sixties, personalities became mannequins. Ravishing personalities are the most riveting things in the world. Conversation, people's interest, the atmosphere that people create around them—these are the only thing worth putting into any issue" [158]. Instead of selling clothes, her magazines spun fantastic tales through images and words that carried the readers into new states of mind and interest.

Throughout the twentieth century, fame, fashion, and the unconscious were repeatedly exploited to make new ideas easier to digest. In the 1910s, Carl Laemmle capitalized upon heartthrob actors to draw audiences from one movie to the next. The star system that he created is a testament to the network properties of celebrity followings, and their powerful halo effect [160].[57] One decade later, Edward Bernays fashioned new memes to drive consumer behaviour into uncharted territory. The undeniable social impact of public relations proves that new ideas can shape how people relate to themselves, and one another. After the Second World War, Diana Vreeland turned fashion into a fanciful escapade where celebrity personality became a hub for human attention and behaviour modeling. Due in part to their network-likeness, public relations campaigns and celebrity endorsement became means for making new trends in consumer goods and lifestyle appear to be the inevitable direction of modern life. In all of these examples, what would now be called "content" was used to change the behaviour of networks of people. Far from outdated, memetic genres like fashion and celebrity remain powerful networks for driving new human behaviour today.

In the late 2000s, American talent manager Troy Carter realized that popular music stars are a kind of network. Carter is perhaps best known as the person who discovered and managed Lady Gaga during the first seven years of her career. Still reeling from the internet's effects on the music industry, Carter recognized that the business model for large music acts would have to shift from one centered around supply constrained physical media sales, to one that embraced infinite supply digital streams. In a 2014 interview, Carter explained that instead of selling vinyl or CDs, "it was more about building a platform on top of music—because music, we realized, sells everything but music" [161], [162]. Carter observed that

---

[57] The halo effect is the power of a positive impression of a given product to influence one's expectations about another product from the same brand.

emotionally resonant artists and entertainers *are a platform* for promoting parallel products. Gaga is a multimedia performance artist whose powerful brand could be channeled into much more lucrative endeavors than insisting on the legacy business selling songs and albums. As her manager, Carter organized brand partnerships and ventures to convert his client's tracks and stardom into lasting power and money. Gaga created her own social network, called LittleMonsters.com, and made partnerships and co-branded products with Polaroid, Virgin Mobile, MAC Cosmetics, and Beats by Dr. Dre [163]. Through encounters with Carter and the music industry machine, Gaga learned to play network capitalism like an instrument, and how to become a network platform herself.

Using pop stardom and celebrity as a platform is nothing new. Gaga's transformation from a musician and performer into a platform for products and experiences is reminiscent of the Hollywood star system discussed above. In recent years, music labels and concert promoters have adopted the Gaga-Carter model for other artists and have called it the "360 deal" [164]. In a 360 deal, the label or promoter give the artist larger sums of money and marketing support up front, in exchange for a percentage of their brand partnerships and merchandise sales. These deals not only recall early Hollywood, where actors would be paid handsomely in exchange for complete control of their lives, but also modern internet networks. Like Carter, internet network platform makers including Napster's Sean Parker and Facebook's Mark Zuckerberg recognized long ago that media (a.k.a. content) is just bits. Instead of attempting to charge for access to content that can be infinitely reproduced for no marginal cost, platform makers know that the network is where value accrues and can be captured. Gaga's army of fans and Facebook's billions of users are two networks that cannot be replicated.

Celebrity, memes, and fashion are three aspects of the same phenomenon. Each collects attention, influences behaviour, and provides shared context that facilitates communication. In the examples we have seen, clever people realized that they could turn ideas into social movements by suggesting to their audience that the product they were selling was *already* representative of a movement of which the audience would surely want to be a part. Memes like these have the power to shape our impression of what is normal, and what we should expect of life.

For network makers, it is important to recognize that networks are memes and memes are networks. Memes are socially expedient fragments of ideas that stand to change the way people behave and see the world. Rather than create more generic utilities with little chance of success, network makers would be wise to create niche memetic objects that correspond to the things that people care about, before expanding into wider use-cases. By focusing on the networks that spontaneously form between people interested in the same subjects, new networks and platforms may find the spark that separates them from the pack to become something truly special.

Section 2: Attention Optimizing Design

In this section I will argue that network organisms that survive by capturing attention are structurally required to practice *attention optimizing design*. I will show that both advertising-based networks, like Facebook, and for-pay networks, like Netflix, are structurally incentivized to maximize user engagement. I contend that attention is a more important paradigm for network designers to understand than the aesthetic regimes (skeuomorphism, flat design, verbal input, etc.) traditionally associated with user interface and user experience (UI/UX) design. This section will culminate with some remarks on how the rise of data driven attention optimization has transformed software interfaces into living real-time attention simulations, where user focus is bought and sold every time the user scrolls and content loads.

Is Attention the New Oil?

For years, the meme that “data is the new oil” has spread amongst people interested in technology and politics. The pithy metaphor contends that the various forms of data we generate through our activities are a valuable resource like petroleum. In 1997, Michael H. Goldhaber coined the phrase “attention economy” to describe the global economy’s shift away from material goods and towards attention [165]. More recently, the “data as oil” meme gained momentum as the swaths of data collected by Facebook and others came to the attention of mainstream media outlets during the early 2010s. This foregrounding coincided with the rise of “data science” and many businesses’ transition to internet advertising and internet business models. The subsequent discovery and invention of new data and compute intensive machine learning techniques in 2012 drove the perceived value of data to new heights.

The counter-meme, that data is not the new oil, has also been argued in numerous publications. The counter-argument states that the metaphor is inappropriate because data is unlike oil in most ways [166]. Data is not a limited resource, but an unlimited by-product of all measurable activity [167]. The ability to collect data is not universal, but data itself is not a rare or non-renewable resource. What’s more, for the privacy conscious, the absence of data collection is a saleable feature. For these reasons, it is not clear that comparisons to well-known commodities helps clarify the emerging significance of data.

Instead of data, I would rather compare attention to oil, in so far as it can be extracted, processed, and sold for profit.

In the following pages I will lay out an argument that both advertising and paid services fall under the umbrella category of *attention optimizing network organisms.* This class of network organism is inherently incentivized to refine crude attention into a saleable form. In the course of this attention processing, these internet lifeforms financialize attention and transform consumer network software design into a practice of real-time attention simulation. I believe this category of design activity is under-discussed and must be a key concern of future network makers.

Advertising-based networks

Social media platforms are advertising-based network organisms.[58] They subsist on advertising revenues. Companies like Facebook and Google make money by selling their users' attention to advertisers. These companies provide free services in exchange for vast amounts of user data. They use this data to develop detailed profiles of people, their tastes, and how they respond to stimuli. Rather than sell this information outright, these companies most often sell advertisers the attention of people with certain intersecting traits, in the form of display advertising.[59] Instead of selling data, these organisms sell access to attention.

Advertising-based network organisms make money when their users spend time looking at advertisements. The more time a person spends on Instagram, the more opportunities Facebook has to sell advertising slots. For this reason, these ad organisms are incentivized to keep users looking at their properties as much as possible. The frequency and amount of time a person spends using a service are two measures of *user engagement*. Engagement, however, is a flexible word. It can be used to describe the measurement of any behavior network makers wish to optimize.

Network organisms that subsist on advertising revenues perceive each user's engagement as a pie chart representing a 24 hour day.[60] The larger the share of the pie that a person spends on a Facebook property, the more revenue the company is able to generate. It behooves advertising-based networks to have the

---

[58] This is true of the most popular social media and free internet service providers today, like Facebook and Google, but it could, of course, change in the future. There is nothing intrinsic to social media or communications services like Instagram and Gmail that *requires* that they be advertising supported, but they have tended to manifest as such because it is easier to build a large network by giving the product away for free. For now, no one has figured out how to have a multibillion person internet-era social platform without data-harvesting and advertising.

[59] This data is also used to improve the efficacy of first party products, like recommendation algorithms, newsfeeds, and search results, to increase user retention and engagement.

[60] Of course, these sophisticated attention-oriented companies employ many analogies for representing users' time, not just pie charts. This representation is only a caricature, as activities often overlap. For instance, Netflix and Facebook, Driving and Listening to Podcasts, or Using Bathroom and Texting are all common overlapping engagement activities.

largest possible number of eyeballs trained on their products and services for as much time as possible per day.

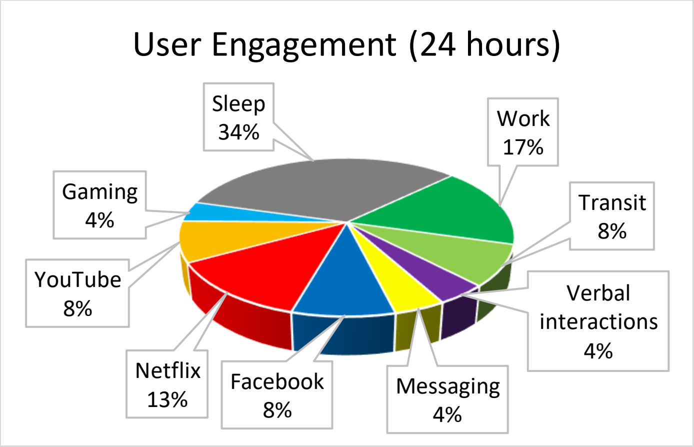


Figure 27 Hypothetical representation of a user's average daily engagement.

Advertising is an attention-extractive business, in which the limited resource of crude attention is collected, refined, and sold for a profit. Advertising-based networks like Facebook and Google capture and process crude attention into refined, fine-grained audience demographic segments. They do this in a multistage process.

First, the advertising-based networks capture billions of attention-hours per day with entertaining and useful free services. This attention generates engagement data, such as time spent, clicks, likes, follows, subscriptions, purchases, friends, age, gender, race, geolocation, camera data, health data, etc. The networks collect and marry this information to data purchased from data providers, such as purchase history, salary, criminal and government records.[61]

Next, the combined engagement and offline behavioral data must be processed. Computations are performed upon the data to transform undifferentiated user attention into segments according to target attributes, like gender, leisure habits, and political affiliation. This refined form of attention is much more valuable than undifferentiated attention, because it allows advertisers who wish to affect people's behavior to target their messaging at people exhibiting very specific intersecting traits. Sometimes,

---

[61] Facebook is known to collect data not only through its own properties, and other digital services that integrate Facebook applications such as the Like button, but also sources of financial and other offline data, which they purchase from unscrupulous data brokers. Google, too, has made itself able to monitor behaviors beyond Google owned properties with projects like Google Analytics, which offers website administrators free analytics tools in exchange for information about their site visitors [168]–[171].

attention providers even sell access to target audiences protected by anti-discrimination law. In 2016, Facebook was caught allowing landlords, employers, and financial services firms to target their ads based on race and gender [172]. Situations like this reveal the discriminatory essence of all targeted advertising.

Now that the users are grouped by fine-grained targetable demographic data, their attention is ready to be sold to advertising bidders in an attention auction. To participate in an attention auction, advertisers join an advertising network such as Facebook Ads, create an ad campaign composed of one or many advertisements, select one or more target audiences, set a daily budget and buying strategy, and upload their advertising content. Audiences can be targeted based on interest, demographics, or behaviors. Advertisers can also use programmatic advertising software to create ads and target audiences automatically based on their own data.

Ad networks like Facebook's employ a real-time bidding mechanism, which allocates advertising space as the page loads for the user. In a real-time bidding ad auction, as the Facebook user, for instance, scrolls an advertising spot into view, an auction between interested advertisers takes place in only a fraction of a second [173]. In this brief interval, Facebook's Ad Network decides which advertiser will get access to the user's attention. "Each time there's an opportunity to show an ad to someone, an auction takes place to determine which ad to show to that person. Billions of auctions take place every day across the Facebook family of apps" [174]. According to Facebook, the ad network takes three factors into consideration when deciding which bidder wins the auction. Of all the advertiser audiences into which a given user belongs, 1) who has bid the greatest amount of money, 2) which ad does Facebook predict is most likely to elicit user interaction (i.e., link clickthrough), and 3) which ad does Facebook rate as the highest quality, based on user feedback and automated visual inspection.

Together, ad networks and real-time bidding constitute a platform for the high speed allocation of attention. To create a clean well-functioning marketplace, both seller and buyer communicate through clear symbolic systems. The advertising-based network makes available its refined user attention in the form of segmented audiences. The advertisers translate their desire for access to user attention into a dollar value. Even the aesthetics of the ads are evaluated, quantified, and compared. With all ambiguities sorted, the internet-speed marketplace is able to assess and allocate attention for each user every time an ad slot is presented.[62]

Each time you or I are presented an ad online, or for that matter any piece of algorithmically selected content, an attention simulation runs on a server somewhere not too far away. In a few milliseconds, the simulation uses sorted and cleaned information culled from our prior behavior and those of our behavioral neighbors to estimate what content is most likely to keep us engaged. Access to our attention is financialized and sold to the highest bidder. There is nothing new about this technology—it has been in use for over a decade. The urgent point to be made here is that this simulation medium is the nature of Facebook's empire, not the kerning of its logo or the flatness of its icons. This mechanism, which is always in motion, is the crucial design upon which advertising-oriented attention optimizing networks are built.

This attention-orientation is not, however, reserved exclusively for advertising-based businesses. Paid services, like premium entertainment and news, also engage in attention allocation simulations. Although these organisms collect revenues differently, they remain engaged in similar design patterns.

Paywalled networks

Recently, a handful of premium internet services companies have adopted a marketing pitch that positions them as an antidote to the political polarization with which social media has become synonymous. This growing crop of companies market themselves to the public as a safe, healthy alternative to the insidious social media. Netflix, Disney+, and Apple News+ all present their products as editorially curated alternatives to the depressing free-for-alls of social media. These unsavory "democratized" media, we are told, are dangerous for democracy and bad for mental health. "We are responsible for what's in there," pitched Roger Rosner, head of Apple's App Store and News products during the launch of their paid subscription Apple News+ service. "We're not just going to let it be a total crazy land" [175].

However, Netflix and its ilk all practice the very same attention optimizing design as advertising-oriented networks.

Netflix uses its viewership data to make content licencing and production financing decisions [176]–[178]. Netflix collects viewer location, time of viewing, search queries, duration of viewing session, menu

---

[62] While I have described this process using display advertising, the same could easily be applied to audio ads in podcasts, or as-yet uninvented ad formats.

browsing behavior, and a list of other Netflix-capable devices on the local network [179]. Netflix even keeps track of what happened in the show or movie just before a user disconnects from the service. The company correlates user behavior with other data it is able to capture from its own content, such as screenshots, changes in volume, and dominant color, and presumably details of the script and on-screen events that machine learning may be able to detect [180]. Netflix Originals, its content creation business, uses this information to select which productions to finance. With this information in hand, the company is able to finance productions at the intersection of several demographics' interests to create must-see properties for each audience that the company wishes to target [176], [177]. They can also use this information when promoting the productions, both on their own site and apps, and when marketing on social and traditional media, too [177].

Although Netflix is a paid service, it too engages in attention optimizing design by necessity. If Netflix fails to deliver the stickiest, most addicting entertainment experiences to its audience, then its growth will plateau and recede. As analyst Matthew Ball has written, Netflix's goal is to capture as much as possible of the four-and-a-half hours per day that the average American spends watching television [181]. The highly competitive entertainment environment online demands that Netflix use its information to license and produce content that keeps its customers coming back and defaulting to Netflix as their video entertainment provider—be that for movie night or just background noise while doing other activities.

Netflix's clever content release tactics often successfully create memes on social media that push people to subscribe and stay subscribed. In the wake of a particularly popular Netflix release, it is common for Twitter, Facebook, and offline watercoolers to buzz with talk of the latest series. Marie Kondo's "does it spark joy," miserable Aparna on *Indian Matchmaking*, *Tiger King's* Carol Baskins, and the dystopian antics of *Black Mirror* encourage viewers to share insider jokes on and offline [182]–[185]. Of course, Netflix is in no way the first paid service or product whose purchase people flaunt through insider language, but its tactics for spurring online conversation make it particularly notable. Releasing the whole of a season of television on the same day, for instance, encourages people to "binge watch" the programme, and in turn makes them more inclined to excitedly share their experience with fellow viewers. During the cultural moment of excitement following a release, Netflix's series are content that viewers can conspicuously consume on social media. To be in on the joke and participate in the collective post-release memetic frenzy, one must become a subscriber and watch the series at the same time as everyone else. This urge to participate is augmented by the ever-present threat of spoilers, and the concomitant "spoiler warning."

Like the paywall, spoilers reintroduce the scarcity of the unspoilt first viewing to the otherwise infinitely available streaming environment.

Paid services like Netflix should be grouped together with their advertising-oriented siblings as practitioners of engagement optimizing design. Just like Facebook, Netflix is a sophisticated interface to the spoils of refined user data. User engagement generates data which the platform collects, processes, and refines into either saleable ad space or addictive entertainment. Netflix, like Facebook, is an interface to an organism—a collective of individual laborers and processes—working to hold our attention. The main difference between Facebook and Netflix is that Facebook retains user attention by curating which items of user generated content to show to a user, while Netflix commissions professionals to generate content, which it serves in much the same way. It is wrong to separate the for-pay services from the advertising-based services, when in fact both use similar techniques to extract the very same limited resource from the population. For these reasons, I propose that we group all network organisms that are financially incentivized to maximize attention capture under the umbrella of *attention optimizing design*.[63]

In this section, I have argued that network organisms' designs are informed primarily by their incentive schemes. Whether they survive by advertising or paid subscriptions, attention optimizing network organisms are driven to increase user engagement at all costs. For creators of new networks and products, it is important to separate trendy discussions about outward aesthetics from the fundamental design task involved in birthing a network. To influence what designs are executed for the full lifetime of the network, one should focus on the incentive mechanisms embedded in the core of the institution one creates.

---

[63] Surprisingly, as of 2020-03-19 there is only one entry in Google's Search index for the phrase "attention optimizing design." This one result includes that string of words in an unrelated sentence. On Google Scholar, there are no results for the same search query. The similar phrase "engagement optimizing design" returns no results on either search engine. The phrase "design for engagement" is popular in the scholarly literature and non-scholarly sources, but it is employed in a variety of ways that do not specifically concern institutions whose viability depends upon attention and engagement optimization, which is my main area of interest [186], [187].

Chapter 4: Making Networks

It has been conceivable for thousands of years that humans might create stories that travel with the bards, or religions that attract the fealty of whole nations. The tools for creating these movements were, however, relatively unstructured and unpredictable. While narrative remains essential to launching networks, newly popularized technologies like the internet and mobile devices have made it significantly easier to distribute a message, a tool, an interface, and a protocol around the world. From this position, we may begin to wonder how one goes about making a new network.

Today, network organisms enable and mediate much of human activity. Apple, Google, Facebook, Wikipedia, Linux, and Bitcoin all play an enormous role in shaping the social, economic, and political spheres of life. Where did these networks come from? What distinguishes them from the large corporations and institutions of earlier times? How will the next networks be born?

Before we can consider how one grows a new network, I must first discuss what makes a network special.

Section 1: Network Effects

Networks are distinguished by a special property: the network effect. Network effects are a set of theories which purport that a network's value increases as a function of the number of people or machines in the network.[64] The telephone network exhibits a network effect. When a new phone line is activated, the network as a whole becomes more valuable to each customer, because all of them can now reach this new subscriber. Computer evangelists are wont to invoke one of a few theories which assert mathematical formulae for calculating the relationship between a network's value and the number of nodes on the network.

Sarnoff's Law

Long before bi-directional networks like the internet were popularized, radio and television magnate David Sarnoff asserted Sarnoff's law, which states that the value of a broadcast network (one-to-many) increases linearly with the number of connected receivers. This means that the value of the network is directly proportionate to how many people are listening. From Sarnoff's perspective, the observation was

---

[64] For brevity, the people or machines on the network can be called nodes.

obvious enough. Each time a new radio was purchased and tuned in, his broadcasting company gained one new receiver, which meant at least one new listener.[65]

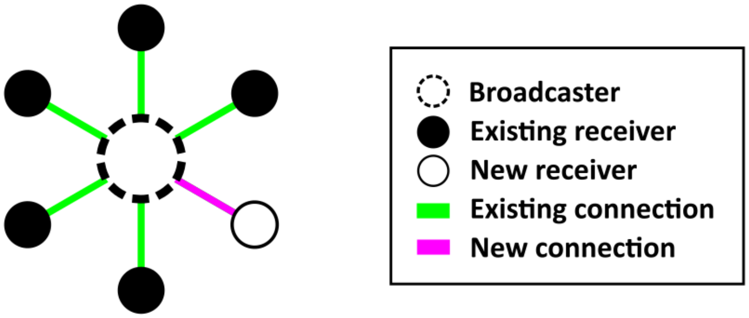


Figure 28 Sarnoff's law: Each new receiver creates one new connection. $V = n$

Metcalfe's Law

In digital networks, Metcalfe's law is one of the most popular models of network effects. Metcalfe's law asserts that the value of a communications network scales according to the number of connections between nodes on the network. The crucial insight of Metcalfe's law is that the number of connections scales much faster in a computer network than a broadcast network because of the computers' uniquely interconnected architecture. When a person buys a radio, the number of connections on the network increases by one—the radio broadcaster has one new potential listener. However, when a new node joins a computer network, new connections are created between that node and *all other nodes on the network*. If the network has three participants, and a fourth joins, three new connections are made.

---

[65] The other side of Sarnoff's equation is worth considering, too. While the value of his radio broadcasts increased as new listeners came online, the value of his broadcast *decreased* as other radio stations came online. More receivers increases the broadcaster's value, but so too would *reducing* the number of competing radio stations. This dynamic is native to broadcast networks, where there is limited radio spectrum and broadcasting rights, but similar dynamics have proved to play out in more decentralized, peer to peer networks like the web and the internet. On the internet, attention has still tended to collect at certain hub nodes, like Facebook and Google. While the technical infrastructure of the internet puts them on a theoretically even footing with any other peer node on the network, history shows that advertising-powered "free" services have allowed certain nodes to grow significantly more popular than the rest. Once Facebook, Google, and others have achieved a place of dominance, not unlike the broadcasters in the one-to-many networks, this other side of Sarnoff's law becomes more pronounced. It is in Facebook's interest to kill competing broadcasters, to increase their share of the internet network's wealth. This is borne out by walled garden initiatives like Facebook's login scheme and Google's AMP, Chrome, and Android projects, which seek to maximize engagement and limit competition.

In mathematical terms, Metcalfe's law states that the value of a network, as measured by number of connections, increases proportionally to the square of the number of participants. This is sometimes abbreviated as $V = n^2$ where V is equal to value and n is equal to the number of nodes in the network. While Metcalfe's law is asymptotically proportional to n^2, the correct formula must also subtract each node's connection to itself: $V = \frac{n*(n-1)}{2}$ [188].[66, 67] An example proves the point. A network of three nodes has three connections: A-B, A-C, B-C. If we follow the naïve simplified equation, then 3^2 would sum to nine connections, however $V = \frac{3*(3-1)}{2}$ yields the correct answer of three connections.

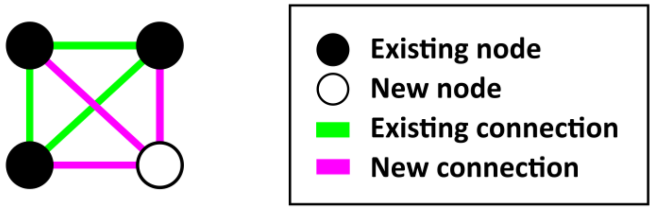


Figure 29 Metcalfe's law: Each new node creates a new connection with every other node. $V = \frac{n*(n-1)}{2}$

Robert Metcalfe, after whom the "law" was named, came up with basic concept of exponentially scaling network value to sell Ethernet networking cards, which he co-invented in the early 1980s [189]. In sales slideshow presentations, Metcalfe explained to customers that the more network cards they installed, the greater the return on their investment. At the time, Ethernet was principally used for creating local area networks—networks of a single company's computers. If a company had twenty employees and twenty computers, then purchasing only two Ethernet adapters would not have a significant effect on the business. If, however, the customer bought enough Ethernet adapters to reach a "critical mass" of networked computers, Metcalfe pitched, the cost of installing the network would be outweighed by the

---

[66] In fact, the accurate equation for Metcalfe's law asymptotically approaches $\frac{n^2}{2}$, which is a not the same as the simplified $V = n^2$

[67] From my research vantage point, it is arguable whether or not this assertion is true. Once a person or machine becomes a participant in a communications network, the presence of the network undeniably affects how the participant perceives themselves. Everyone who has ever sent themselves an email or a text message knows that a node can connect to itself. Participation on the network reshapes one's experience of self. A calculation like Metcalfe's that uses connections to quantify value may rightly calculate the connections from each node to itself as a part of the networks value, given that network participation invents this novel brand of introspection possible.

exponential growth in its value to the company. This pitch was popularized by magazine publisher George Gilder, who wrote about the concept and named it Metcalfe's law in 1993 [189].

Despite its intuitive appeal, Metcalfe's law cannot be correct.

First, Metcalfe's law fails to clearly define value. As he recalled in an article for Forbes magazine: "I was a little vague about what 'value' was, but in those days it had something to do with sharing expensive disks and printers, exchanging electronic mail within buildings […] My 3Com customers believed me and bought more Ethernets, which proved to be more valuable, and so they told their friends and bought even more" [189]. One can believe that a company is able to extract more value from office equipment when access is shared over the network, but this is not a rigorous definition.

Second, Metcalfe's network theory fails to account for qualitative differences between networks, which are crucial to understanding how new networks replace old networks. In the early 2010s, for instance, teenagers began to join Snapchat in favor of Facebook, because Facebook had become popular with parents and teachers [190]–[192]. This story echoes offline sub-cultural movements, where an underclass rejects the value judgments made by the dominant social group and creates new social and cultural value *outside* of the biggest networks. For teens, Facebook's larger network was a *detracting* factor that led them to instead join a new network with fewer nodes.[68] Snapchat's user interface was widely panned by the technology press and software-savvy young adults because its unconventional design was unappealing to the people attached to mainstream software aesthetics [190]. In many ways, Snapchat was the anti-Facebook, and it succeeded in this niche despite what Metcalfe's law would have to say about network scale. Metcalfe's law does not account for the different use cases and associations users have with networks, instead proclaiming that more users is always better. If this were the case, how could new networks like Snapchat get off the ground?

Third, Metcalfe's equation aims to make the value of a network into a calculable sum, but its failure to quantify qualitative differences makes the formula useless for comparing two networks. In May 2018, YouTube CEO Susan Wojcicki announced that YouTube had 1.8 billion monthly active logged-in users [193]. According to Metcalfe's law, the value of the network is equal to the one quintillion six hundred

---

[68] Although Facebook was able to stem Snapchat's growth by copying their key features in Instagram and WhatsApp in 2018, it remains to be seen if qualitative differences between the two networks will ultimately create *different* value for users of Snapchat, or another new entrant, with which Facebook cannot compete or replicate despite greater size.

twenty quadrillion (1.62e+18) connections it enables. By contrast, TikTok and its sister product Douyin have a combined monthly active user base of 900 million people [194], [195]. According to Metcalfe's law, TikTok's value is only four hundred and five quadrillion (4.05e+17). Are these values comparable? Sort of. Are they indicative of future growth rates? No. Are they indicative of user monetization or social impact? No.

If it were true that the utility of a network could be directly derived from the number of nodes or connections, then it would be impossible to explain how smaller networks displace larger ones. If a network's value approaches $\frac{n^2}{2}$, how can a large network ever be overcome by a smaller one whose value is measured by the same formula? The calculation strips networks of their qualitative differences, which removes a great deal of its utility.

Fourth, Metcalfe's law presumes that all network connections are of equal value. Inside of an office scale operation, this may make sense. There is a world of difference between two of twenty employees having networked computers, and twenty of twenty employees sharing a network connection. However, this is less obviously true when applied to very large networks. It is easy to presume that, as new nodes subscribe for phone service, buy a fax machine, or sign up for Facebook, the value of the network increases for all participants. By some hand-wavy mathematics, I can be convinced that the minute increase in value to each person on the network sums to a value that exceeds the cost of the individual device—Metcalfe's critical mass tipping point. It is less obvious, however, that this holds true once the network grows to be truly enormous. When the two-billionth person joins Facebook, does the network's value really increase for most other users? If even tens of thousands of people in a part of the world with which I have no truck whatsoever join up, does it really affect the value for me in any discernible way? Although each user may be said to increase the overall value of the network by some measure of value, there are diminishing returns once all people in a given node's sphere of communication are onboard, and Metcalfe does not account for this at all.

Reed's Law

In 2001, computer scientist David P. Reed proposed that the value of a network grows in proportion to the number of groups that can be formed within the network [196]. Reed observed that, while Sarnoff's law applies to one-to-many broadcast networks, and Metcalfe's law applies to peer-to-peer networks, there was another type of network architecture yet to be quantified.

Reed’s law applies to what he called group-forming networks (GFNs), in which users are able to create groups with only a subset of their fellow nodes. To understand the difference, let us consider a text messaging network like SMS. Metcalfe’s law purports to capture the value generated by all the possible two-person text message conversations. Metcalfe states that the value of a network is the same as the number of connections between nodes. Reed’s law, by contrast, quantifies the value of all of the potential *groups of nodes* on the network. In our text messaging example, this would mean counting all of the possible group chats, in addition to all of the two-person conversations. Reed’s law counts not only peer-to-peer bi-directional connections, but also the connections between a node and all possible unique groups of nodes to which it might belong.

According to Reed, in group-forming networks, the value of the network corresponds to how many groups the nodes can make with one another [196]. Reed introduced his theory as follows. “Let’s say you have a GFN with *n* members. If you add up all the potential two-person groups, three-person groups, and so on that those members could form, the number of possible groups equals 2^n. So, the value of a GFN increases exponentially, in proportion to 2^n.” A network of two nodes can only form one group containing the whole network. A network of three nodes, however, can form three additional subgroups, each containing a pair of nodes. A network of four nodes can contain eleven groups. To be precise, the function is formally written as $V = 2^n - n - 1$ to account for groups with only one member and one empty group. This is often simplified to 2^n.

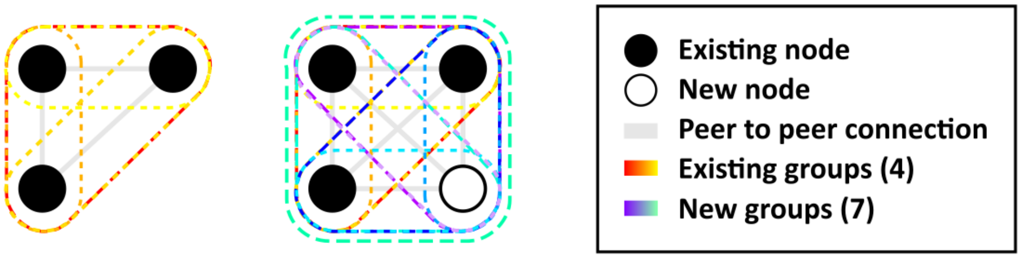


Figure 30 Reed's law: Each new node creates new groups with every possible subset of nodes. Groups must contain at least two members. $V = 2^n - n - 1$

Despite Reed’s intriguing contribution, his law suffers from the same problems facing Metcalfe’s law. Value remains vaguely defined, the sums it yields do not capture crucial qualitative information, its output is not useful for comparing or contrasting networks, and the formula does not account for diminishing returns when networks achieve global scale. This last point leads to especially absurd scenarios under Reed’s law: if someone in a faraway community joins me and the other 1.5 billion users on WhatsApp, does the network's value nearly double, because of all the new possible group chats that creates [196]? While the direction Reed's law points is interesting, it is difficult to seriously hold to its conclusions.

In summary, it is unclear how to quantify the value of communications networks in terms of their size, but simple observation shows that networks share an unusual growth-related property. This tendency, called a network effect, can drive more new participation at a super linear rate in the most successful cases. As nodes join the network, its value proposition becomes more attractive—at least in the early growth phase. Network effects are unique to networks; this property is not shared with manufactured goods and traditional services.[69] Until a network reaches an inflection point, which Metcalfe called critical mass, participation in the network may cost more than it is worth. For network makers, the important takeaway is that their network must be adopted by a large enough set of initial users before its value becomes apparent. Beyond that insight, the real mathematical value of network effect discourse is less clear, and perhaps mostly marketing.

Section 2: Booting Networks

Despite the temptation to attribute everything to the technology upon with which they are built, the origin stories of many of today's largest network organisms are rooted in social transgression as much as technological disruption. Although they are often called "tech companies," large scale network organisms are in fact multidisciplinary performances spanning the domains of software, law, and publicity.[70] To become capable of booting up new networks, we will have to inhabit a mindset that sees network creation as more than a purely technological activity.

The beginnings of the most successful social media platforms, internet marketplaces, and vertically integrated electronics brands can be traced to *creative* interpretations of corporate, labor, and intellectual property law, and a belief that "it is easier to ask for forgiveness than it is to get permission."[71] In this section I will explore a handful of criminal or socially unacceptable practices that launched famous internet brands and discuss why big networks sometimes come from bad behaviour.

---

[69] Economies of scale apply to manufactured goods, and so the price of manufactured goods falls as demand increases, but this appears to be a different subject. The connection between the two may be worthy of further research.

[70] I establish this point at greater length in Chapter 2: Performing Software.

[71] Grace Hopper [197].

The historical record shows that many successful networks achieve initial viral growth by facilitating prevalent but socially unacceptable behaviours, but why should this be the case? Before I examine a handful of examples, I would like to dwell for a moment on this question of "why?"

The first major challenge facing a nascent network is the cold start problem. As I discussed in Section 1: Network Effects, a network's value is often said to increase as a function of the number of participants. If this is true, then a new network with few users is not worth much to anyone. In the context of network building, the cold start problem describes the conundrum of attracting the first users to a network product, when the majority of its value will only exist once it has grown popular.

I first encountered the cold start problem up close when I interned at Color Labs in 2011. Color was a social network startup based in Palo Alto, California. When I joined in the summer of 2011, the company was just coming to reckon with its particularly hairy cold start dilemma.

Color was a geosocial photo network. The app allowed users to create collaborative photo albums with people nearby. In Color, users could *only* see photos recently posted within 100 metres of their current location. The CEO boasted that, in contrast to Facebook, this would be a social media for connecting with people nearby—a way to make new friends and connect with the people we see in the corridor every day but never meet.

The cold start problem was especially acute in Color's case. In a network of only a few hundred users, it was very unlikely that anyone had recently taken a photo nearby. If a network with a billion daily active users were to implement Color's product as a feature, it might have had a chance at success. On its own, however, there was little chance Color would grow because there was very little incentive to get onboard early in the network's life. The product worked well for everyone in the company office—it was fun to post a picture and see it appear on the phones of everyone in the room—but it was not very interesting to people without so many friends on the network.

It is inherently difficult to convince people to sign up for a new service or product for which there is no direct precedent. Joining a network is not fun in-and-of itself. Why would I want to install your app or join your network if it solves a problem I do not already have? In Color's case, while the shared photo albums feature was fun, it did not serve a profound or common enough desire to weather the early days of relative obscurity. Its 100 metre restriction was so severe that no amount of splashy advertising or corporate partnerships could have rescued it [198].

To successfully boot up and gain network scaling growth momentum, network products often first find a foothold within a specific niche. General purpose products that serve too wide a hypothetical audience typically fail to draw an initial group of enthusiastic users that make it possible to eventually serve a wide variety of use-cases. Facebook would never have taken off if it was first sold as the do-everything platform that it has now become. Without a zealous group of early adopters, there is little chance the network product will be shared by word of mouth. By contrast, when a network is able to attract an initial group of enthusiastic users, they tell their friends and family about the positive experiences they are having, and their activity allows the designers and developers to hone the product in a cycle of design, use, quantitative and qualitative data collection, redesign, and renewed use.

One battle tested way to successfully launch new networks is to appeal to human desires that long predate computers. In his talks to entrepreneurs and software designers, personal computing and interface pioneer Alan Kay frequently points to anthropologist Donald Brown's list of Human Universals as a guiding document for those seeking to attract the attention of the masses.[72] Human Universals is a list of over 300 human behaviours and concepts observed in all cultures around the world [200, pp. 435–439]. The list includes mental patterns, like binary cognitive distinction and private inner life, personal attitudes, like food preferences and manipulation of self-image, and taxonomies, like classifications of color, age, flora, fauna, and body parts. The list also includes familiar human interests, like anthropomorphization, gossip, hairstyles, play, music, and sexual attraction. Of course, it can be difficult to accept the idea of a list of human universal traits. What contemporary anthropologist would dare to claim to know what is universal to all humans, given their necessarily limited perspective and experiences? Nevertheless, if we raise our head from the page and look around at the media surrounding us, it is self-evident that practically speaking, a person can attract attention by servicing the most common observable human fascinations, whether they are truly universal or not.

In the pages that follow, I contend that many successful networks start with a sin.[73] Some break the law, while others break only social norms. One way or the other, many successful network organisms find their first users enabling transgression.

---

[72] Although Kay points to this document in a derisive, superior tone, I believe he is right that a person is more likely to successfully grow their product or network if it serves common themes of human interest [199].

[73] I am using the term sin poetically. I do not literally mean a biblical sin. When I say "sin," I mean a transgression of the social mores or legal norms of the time and place.

I would like to discuss two types of sinful origin stories. The first category are sinful prototypes whose remarkable popularity inspires their creators to revise their approach and create lawful variations on the original idea. Facebook is an example of this tendency, and its roots in privacy invasion and data misappropriation remain essential to understanding the company's actions today.

The second category are sinful products that facilitate bad behaviour until they have grown to network scale. Once they have secured market dominance, these networks either clamp down on the transgressive behaviour that initially drew users to the platform, or they cause a change in social norms so that those behaviours are no longer transgressive. YouTube and Uber are examples of this brand of sinful boot fuel.

A third category are networks that never cease to service transgressive behaviour, but I will not discuss these in this text. Napster and BitTorent are examples of this third category.

I believe it is important to acknowledge these transgressive behaviours at the root of network organisms, which I am grouping under the tongue-in-cheek umbrella of *sin*, because they are not one-off events but a pattern amongst break-out hits in this medium of network making. Why is it that flouting society rules proves to be such a powerful tool for attracting social activity?

Through series of examples, I will show that many successful networks start out by breaking the rules because sin is the easiest way to get people's attention. When booting a new network, it is difficult to convince people to cross the threshold and sign up, install, or buy. Why should I join a small network that offers a service I do not already want?

To bridge the gap and create initial user growth momentum, many networks enable people to do something that existing products and services will not let them do. If a network allows you to peep on your classmates or watch a free clip of pay TV, then it satisfies a human universal desire that no existing product can. A network that enables me to do something I want to do, but so far cannot, is attractive enough to be worth joining. A network that enables sinful behaviour is easier to boot than one that merely offers a new interface to something I can already get somewhere else. In the next pages, we will see this pattern reproduced several times.

Facebook: Privacy

Before Facebook, Mark Zuckerberg built Facemash, a social web application for comparing and rating his Harvard classmates. As we shall see, Facemash's viral success and subsequent legal troubles taught Zuckerberg that people wanted image-driven social media about people they knew, and that to do so successfully, he would have to reframe his early efforts to be more in line with intellectual property law and individual consent.

In Novermber 2003, Zuckerberg launched Facemash [201]. Facemash was a Hot or Not style voting game that presented users with two images and asked them to pick which was "hotter." While prior Hot or Not games pitted celebrities against one another, the Facemash interface displayed the student ID photos of two current Harvard students under the banner "Were we let in for our looks? No. Will we be judged on them? Yes" [201]. Visitors to the website were encouraged to pick which of the two students was more attractive, then repeat the process with another match-up ad infinitum.

To populate Facemash's picture database, Zuckerberg scraped his classmates' headshots from Harvard's password protected house webpages. Zuckerberg documented the project development in a journal he published on the website itself [201]. The audacious log described how, between 1 AM and 4 AM on October 28, 2003, he "hacked" and downloaded photos from each of the nine Harvard houses with online facebooks. He gloated that the hacking was "child's play," and he exclaimed that some of his classmates photos were so unbecoming that "I almost want to put some of these faces next to pictures of farm animals and have people vote on which is more attractive" [201].

Facemash was the buzz of the campus when it launched on Halloween, but the illicit means by which Zuckerberg acquired the photos quickly landed the soon-to-be founder in trouble with the College. According to interviews with The Harvard Crimson, Zuckerberg completed Facemash at 7:30 AM on Friday, October 31. He shared a link to the new website with a few friends, and by 10pm that evening, 450 people had visited the site and cast 22,000 votes [202]. Zuckerberg pulled the site down two days later, on Sunday evening, amidst outrage on campus at the usurping of student ID photos. The next day, the College's Administrative Board summoned Zuckerberg to answer to accusations of "breaching security, violating copyrights and violating individual privacy" [202]. The disciplinary action was ultimately dismissed and Zuckerberg was allowed to continue his studies at Harvard.

Four months later, in February 2004, Zuckerberg launched thefacebook.com. The Crimson reported on the new site's launch under the headline "Hundreds Register for New Facebook Website" [203]. In his interview for the article, Zuckerberg stated that "Facemash was a joke, it was funny, but at its root it had its problems—not only the idea, but the implementation. It was distributing materials that were Harvard's. I was very careful with [thefacebook.com] to make sure that people don't upload copyrighted material" [203].

Facemash taught Zuckerberg that, while Harvard students were interested in ogling and comparing photos of their classmates, a web application serving that need would have to at least appear to respect intellectual property rights to avoid reprimand. Zuckerberg's first sinful foray into social media proved that people desired a product that would let them stalk and rate their friends and friends of friends. His decision to take the site down and redesign the social contract with his users proved prescient. To his credit, Zuckerberg used his observations of the successes and failures of Facemash to inform the development of an entirely different product with dissimilar user experience and product design, which nevertheless engaged the same human urges. His subsequent creation was so successful that, even today, after countless privacy scandals, dishonest data collection policies, and an immeasurable impact on the global psyche, Facebook continues to garner billions of daily active users and remains a popular brand amongst people who appreciate the free communications services it delivers.

For these reasons, Facebook is an example of a sinful prototype that inspired a more lawful successor network.

YouTube: Piracy

YouTube grew to network scale for technical and non-technical reasons. It is exemplary of the multidisciplinary performance required to launch successful network software. YouTube became popular because it made use of a newly available technology for distributing video on the web, it was easier to use than its competitors, it made available intellectual property (IP) it did not own, and through acquisition, it was infused with enough money and legal talent to fend off IP lawsuits long enough to grow to network scale.

Jawed Karim, Steve Chen, and Chad Hurley started YouTube to make it easier to share and discover videos. In his commencement speech at the University of Illinois, co-founder Jawed Karim claimed that he and his cofounders created YouTube after having identified an unserved need online. While working at Paypal in the early 2000s, the three co-founders recognized a need for easier video sharing and discovery. According to Karim, the Indian Ocean Tsunami, which hit the West coast of Indonesia in December 2004, was the first natural disaster captured with mobile phone cameras. Videos appeared on online, but were scattered across many websites. These video files were too large to send by email, and to play them on websites, each user first had to install a video player browser plug-in, such as QuickTime. Having observed this need, Karim, Chen, and Hurley decided to create a website to make it easier to upload and share videos by hyperlink. On February 14, 2005, the three co-founders registered youtube.com. The first video was uploaded to the site only two months later, on April 23, 2005 [204].

YouTube's success can be partly attributed to the company's application of a cresting technology: Flash Video. In 2002, Macromedia released Flash MX, a new version of its Flash software. Flash was a multimedia authoring tool. The software was initially used for animation and rich media websites. To view a Flash project, web surfers would first have to install Flash Player. Adobe, which acquired Flash parent company Macromedia in 2005, boasted in PR releases and marketing material that 99% of web-enabled computers had Flash installed, although that figure has since been partially discredited [205]. Although specific figures are difficult to obtain, many games and multimedia websites were built in Flash at the time, and so many people had Flash Player installed on their computers. The 2002 update introduced a new feature that allowed Flash authors to include videos in their projects.

Until this point, it had been difficult to distribute video on the web as there were many mutually incompatible video codecs, or encoding formats, and players. Macs and PCs required different software to play subsets of the available codecs. Watching video on the web before YouTube was not a good experience. The QuickTime and Windows Media Player plugins each required frequent updates and would often spawn a software update notification at precisely the moment the user tried to watch a video. Often, videos did not work. The players were also not customizable, making it difficult for anyone to build a video rich website. QuickTime proudly displayed its clock-like Q icon on first load and whenever buffering the video, and it always showed its own distinctive grey timeline scrubber and play/pause button at the bottom of the video frame. Prior to Flash MX, there was no simple way to distribute video on the web with a custom player.

When it launched in April 2004, YouTube's founding team was uncertain how to garner interest in their service. Although the three founders and growing team had figured out how to stream video on the web, they were not sure what the site was for. At first, they positioned the service as a video dating website. "We didn't even know how to describe our new product. To generate interest, we just said it was a new kind of dating site… We even had a slogan for it: Tune In, Hook Up." The founders posted ads on Craigslist offering $20 to any woman who uploaded a video to the site.[74] Nobody replied to their offer. In desperation, Karim uploaded videos of 747's taking off and landing at a nearby airport [206]. The mixed

---

[74] Karim continued, "We were so desperate for actual dating videos, whatever that even means… So we spammed Craigslist in Los Angeles and Las Vegas encouraging women to upload videos of themselves. In exchange we offered to send them $20 for every video uploaded… We didn't get a single reply… It didn't even matter. Our users were already one step ahead of us. They began using YouTube to share videos of all kinds… We found this very interesting… By June we had completely revamped the website making it more open and more intuitive."

signals sent to early users must have been bizarre. In its earliest incarnation, the site did not even have a search function. Instead, the site showed only a random selection of its small, eclectic collection of clips.

During its first year of operation, YouTube faced huge amount of competition. According to Julie Supan, the company's first marketing director, a 2005 report claimed there were approximately 280 competing web video companies in business when YouTube was founded [207]. In January of that year, Google entered the fray with its Google Video product [207]. Google Video featured higher quality video streaming and searchable video transcripts for some videos. On paper, it was a superior product.

Despite its awkward initial positioning and stiff competition, YouTube grew quickly. In July 2006, only eighteen months after its founding, YouTube was streaming 100 million videos per day and had 20 million visitors per month [208].

Company insiders attribute their success to superior usability and uptime, compared to its competitors. Gideon Yu, the company's first Chief Financial Officer, described the difference between YouTube and its most dangerous competitor as follows: "With Google Video, you needed to know what codec your video and what was the dimension of the video frame, [whereas] with YouTube it was all in Flash and when you wanted to upload a video to YouTube, the only thing you needed to do was push a button that said 'upload'" [207]. Although usability was important, YouTube's killer feature was most likely the television clips it hosted without the copyright-holders' permission.

In 2005, YouTube became popular as a place to watch viral clips of American television shows. Clips of news and comedy shows were particularly popular on YouTube in its first year. The Daily Show with Jon Stewart aired on Comedy Central at 11PM on Monday through Thursday nights, and it was available for free the next morning on YouTube [209]. The Colbert Report, which debuted in October 2005, grew popular in part thanks to clips of host Stephen Colbert's speech mocking George Bush at the White House Correspondents Dinner, before it was pulled down by rights holder C-Span [210]. Perhaps most famously, on December 17, *Saturday Night Live* broadcast "Lazy Sunday," a satirical song starring Andy Samberg and Chris Parnell. A video clip of the television sketch was uploaded to YouTube, and within ten days it had been watched 1.2 million times [211]. Two months later, on February 17, 2006, NBC sent a cease and desist letter to YouTube demanding that they remove the video from the site [212], [213].

During its first year online, the co-founders considered how they could allow copyrighted material to stay on YouTube to foster growth, without being sued into oblivion. Court filings from Viacom's 2007 lawsuit against Google revealed that the YouTube founders and early employees were well aware of what they were doing all along [214, p. 120], [215].[75] In one of many similar instant messenger conversations, a product manager told co-founder Steve Chen that, “[I] went through all the most viewed/most discussed/top favorites/top rated to try and figure out what percentage is or has copyrighted material. it was over 70% [sic]” [215, p. 25].

By August 2005, the three YouTube cofounders had decided it would be best to initially allow copyrighted material to be illegally circulated on their website to build its popularity. Hurley emailed Chen and Karim. “[W]e need to start being diligent about rejecting copyrighted/inappropriate content. we are getting serious traffic and attention now, I don't want this to be killed” [215, p. 11]. “I really don't see what will happen. what if someone from cnn sees it? he happens to be someone with power? he happens to want to take it down right away. he get in touch with cnn legal. 2 weeks later, we get a cease & desist letter. we take the video down," Chen replied. Karim concluded the transaction with what would become YouTube’s policy: “lets remove stuff like movies/tv shows. lets keep short news clips for now. We can become stricter over time, just not overnight. like the CNN space shuttle clip, I like. we can remove it once we're bigger and better known, but for now that clip is fine.” The email trail shows that the co-founders had decided to prioritize growing till they could completely eclipse their competitors, before proactively seeking out and pulling down copyright infringing content.

On October 9, 2006, Google acquired YouTube for $1.65 billion [216]. After a heated competition between the two sites, and many others, Google decided that it would be safer to buy YouTube before anyone else. Market research firm Hitwise estimated that YouTube was responsible for 46% of the internet video streaming market when the deal was announced [217]. Unbeknownst to its users, the same Viacom court documents revealed years later that the founders had planned a sell their business quickly, from the start [215, p. 11]. Two months after the acquisition, Time Magazine published its memorable 2006 Person of The Year issue. The magazine’s cover showed an iMac and keyboard—the screen obscured by a shiny metallic rectangle and unmistakably YouTube-like video playback controls. Across the metallic video screen was printed one word: “You.” [218]

Once acquired, YouTube benefitted from the deep cash reserves and stable of Google-retained lawyers to delay legal consequences until the network was large enough to be irreplaceable. With Google’s engineering and infrastructure might, YouTube was able to cover ballooning bandwidth costs—reportedly

---

[75] Viacom, the parent company of CBS and Comedy Central, sued Google, which had by then acquired YouTube, for $1 billion on March 13, 2007. [214, p. 120]

200 terabytes of traffic per day at a cost of around $2 million per month [219]. Meanwhile, Google's lawyers stalled the incoming lawsuits, allowing YouTube to build a digital video destination of incomparable scale.

YouTube was made by a multidisciplinary performance in technology, copyright infringement, paid legal counsel, and showmanship. YouTube separated itself from the pack of competitors with a brazen and illegal approach to copyright law. With sufficient funding from its parent company, YouTube was able to weather legal and financial burdens beyond the means provided by its revenue streams. Once it had grown to network scale, YouTube's viewership became an attractive advertising surface for intellectual property owners like TV networks, movie studios, and music labels. In a handful of years, YouTube transitioned from "the Napster of Video" into a partner and service provider to copyright holders all over the world. By dint of its network scale, YouTube (a.k.a. Google) has positioned itself to have very favorable relationships for global streaming rights, because it gate keeps access to a growing audience of over a billion viewers.

In this section, I have argued that successful networks often break through to a wide audience by enabling a socially transgressive behaviour that they cannot achieve with existing services. In the marketplace of attention (Chapter 3: Fashion and Attention), a fledgling network must fulfill a participant's needs or wants to establish a meaningful place in their lives and the opportunity for recurring participation. I introduced the cold start problem with the example of Color, and I used the examples of Facebook and YouTube to show how more successful networks use socially unacceptable behaviour to boot new networks, which they ultimately reign in to relatively more acceptable legal and social practices. I proposed that Facebook is an example of a network that began with a particularly sinful prototype, which was reformed in its second iteration in response to public backlash. I used YouTube as an example of networks that boot with a sinful approach, in this case intellectual property piracy, then reform once they have reached indomitable network scale. In all of these cases we have observed that yet again the network organisms that foster the development of the software and the growth of the network that uses them are themselves multidisciplinary performances spanning law, fashion, software, networking, and public relations, and perhaps still other parts of life.

Chapter 5: Supply Chain

The *supply chain* is the network of raw resource extraction, design, manufacture, sales, and shipping that delivers everything in our lives that comes from away. It is "the chain of processes involved in the production and distribution" of commodities [220]. Everything that is made in one place and consumed somewhere else is implicated in the supply chain. Every process of manufacture and cultivation that requires materials from elsewhere is part of a particular supply chain, and the broader ecosystem of trade processes called *the* supply chain.

In this chapter, I will explore the relationship between computation, the supply chain, and the making of meaning. The chapter will unfold in two parts.

In the first section, I contend that, despite manufacturers' claims of ever increasing performance, general purpose computers are and have always been essentially disposable. Apple's integrated product designs are emblematic of this acutely disposable era of computing machines. Through a close study of Apple's product design lineage, I observe a design ethic that prioritizes usability and aesthetic appeals to the individual consumer over durability and environmental friendliness. I call this *Cupertino Design*.

In the second section, I argue that meaning is waste. In this portion of the text, I claim that disposable goods are parts of speech. Although their negative impact upon the environment deserves attention, it is important to understand that people are drawn to purchase AirPods and clothes that fall apart after one wash for the very same reason. These products are means of expression. To ensure a prosperous environmental future, humanity must undoubtedly address the dangerous consequences of unbridled consumerism; however, to move beyond our current moment, I assert that we must also acknowledge that human meaning is necessarily bound up with activities that can be criticized as wasteful and superfluous from a disconnected vantage point. Instead of rejecting computers as wasteful totems of transient fascination, I believe we must appreciate the natural human inclination to use the material around us as an extension of our embodied consciousness. Solutions to ecologically unfriendly patterns of living are more likely to come from a deep understanding of human meaning making and the intrinsic motivation to express oneself than overly simple castigation of humans as near-sighted hedonists.

Section 1: Cupertino Design

Apple is the most influential design organization in the world today [221], [222]. It is unparalleled in its ability to divine and execute category defining products that aspire to seamlessly integrate software, hardware, and services. For decades, the company has turned out a series of smash hit products that customers love and competitors love to imitate.

For all its strengths, the company may well be remembered for a lineage of beautiful but disposable products designed to last not more than five or six years of regular use.[76] Apple's hardware designs symbolize and perhaps catalyzed an industry-wide shift away from customizable, modular consumer electronics, towards sealed-shut plastic-and-glue assemblies that are a repair and recycling nightmare. In this section, I argue that Apple is a pioneer of fashionable disposable computers. In the pages that follow, I will explain how Apple's business model motivates the company to engage in disposable computer design, which I call *Cupertino Design*.

Apple is well known for setting trends in product design. From its signature Macintosh computers to its recognizable white earbuds, Apple has blazed a trail of influence since its debut in the mid-1970s. After a period of mismanagement and diminishing sales in the 1990s, Apple burst back onto the scene and quickly cemented its reputation for bold looks and innovative all-in-one products with its colorful 1998 iMac and the launch of the iPod in 2001 [224]. Since the iPod's debut, Apple has taken a dominant position within the fields of industrial, product, and software services design. The company's memorable advertising campaigns, vertical-defining products, and brand connotation with premium luxury computing experiences have made it a touchstone for anyone studying design in recent decades. When talking about product design today, Apple is the company on everyone's mind.

The iPod was a pivotal moment in Apple's design history. Its portable, fully integrated design proved that the company could achieve its greatest commercial and cultural successes with products discouraging of user modification and repair. In the 1990s, it was common for consumer electronics to require AA, AAA, or D size alkaline batteries [225, p. 39].[77] The iPod, which became Apple's bestselling product until the iPhone, kick-started the trend of mass market consumer electronics with non-user replaceable batteries

---

[76] Analyst Horace Dedieu estimated in 2018 that Apple devices remain active for 4.25 years on average [223]. He came to this conclusion by subtracting Apple's public active device figures from the total number of devices sold. I have added a couple of years in the text to compensate for people's penchant to discard older devices before they are completely unusable. Anecdotally, I observe that their wearables work well for two to four years, their phones work well for three or four years, and their laptops and desktop computers work well for four to six years. Of course, hobbyists and enthusiasts manage to use much older devices for decades, but this niche is not representative of most consumers.

[77] Meador estimated that Americans at the time purchased between one and three pounds of batteries per year [225, p. 39].

[226].[78] The music player's all-in-one design echoed the 1984 Macintosh, and took the approach even further by integrating the power source and input device into the case. At launch, the omission of a user-replaceable battery was contentious. Critics of the iPod's integrated lithium-ion battery received millions of views online, but the sealed package design ultimately prevailed [228]. It has since spread to phones and proximate consumer electronics categories.

From the manufacturer's perspective, non-modular products have at least three advantages over modular equivalents: they can be smaller, simpler, and harder for non-experts to repair, thus reducing upgrade cycle time.

First, integrated products can be packaged in smaller enclosures because they do not require multiple use modular affordances. Modular hardware require affordances for attaching and detaching pieces, which fully sealed devices can omit [229]. For instance, products with user replaceable batteries, like TV remotes and inexpensive electronics, often feature multiple-use cantilever snap-fit battery compartment covers.[79] These mechanical features increase packaging volume and are more likely to break than non-moving parts. Integrated designs reduce the number of parts, and so they can be more robust than modular equivalents.[80]

Industrial designs that eschew modularity and reparability can also reduce overall package size. Traditionally, computers and electronics were assembled using sliding rails and screws. Latter day Apple products replace these fixtures with glue to reduce the space costs incurred by screws [231]. This design choice makes repairs more difficult for the inexperienced, but allows for thinner enclosures [229].[81] The size of a device, and especially a personal mobile device, is a simple metric for annual iterative improvement. It is easy for consumers to compare products based on their outer dimensions, either with

---

[78] iPod sold 300 million units between 2001 and 2011 [227].

[79] Snap-fits are an engineering design pattern for assembling flexible interlocking parts [230].

[80] Integrated designs can be more robust, if the case material and enclosure are designed for robustness. Phones today are designed to optimize thinness over robustness, however, and so the edges of their glass screens are exposed, making them vulnerable to drops. For this reason, the robustness argument is only so meaningful in the current context.

[81] Not all glue assemblies are designed equal. Some glue assemblies, like iPhones, remain repairable at the systems level. If you smash your iPhone screen, a person with experience and the right tools can swap that component out. To repair the screen component itself would be much harder, as it is composed of sheets of glass, screen, and light emitting materials that are fused together during manufacturing. Due to their size and glue-to-components ratio, AirPods are also much harder to fix than an iPhone.

specific numerical product specifications, or simply by feel. Market performance indicates that thinner, sleeker devices appeal to customers more than larger modular alternatives.

Second, fully-integrated products allow designers to simplify enclosures and create eye catching minimalist designs that distinguish their products from competitors. The original iPod's white plastic face and unbroken shiny metal back are memorable designs in part because they featured no battery compartment or user-facing screws. For decades, Apple has cultivated a brand association with the concept of simplicity. An ideal Apple product "just works." Spare enclosures wordlessly communicate this product and user experience philosophy to prospective customers. Minimalist enclosures photograph well and lend themselves easily to the company's marketing strategy of making the product the gorgeous hero.[82]

Third, integrated products are more difficult to repair, which drives consumers to upgrade instead. Apple's choice to use hidden fasteners and glue makes their devices difficult for non-experts to repair. To disassemble an Apple device often requires that the repair person heat the enclosure to melt and loosen the glue inside, then delicately pry the parts open with plastic spudgers [233], [234]. These operations are intimidating to non-experts and require special tools.

Apple's behavior indicates the company is, in fact, actively hostile toward customers repairing their own devices. The company does not provide disassembly and repair manuals to customers [235]. Since 2009, Apple has used obscure pentalobe screws to make hardware repair more difficult [236]. The company tightly controls the sale and distribution of authorized replacement parts to certified repair partners only [237]. To become an authorized or independent repair partner, repair shops must sign contracts that limit what repairs they can execute. These contracts also stipulate Apple may perform unannounced in-person audits to check if independent repair shops are using unofficial replacement parts [238]. These measures, combined with industrial designs that make disassembly intimidating, and the looming threat of voided warranties, discourage users form repairing their devices. Apple claims these efforts are intended to keep repair quality high, but they are undeniably also motivated by a desire to shorten upgrade cycle time.

---

[82] Peter Belanger has photographed many Apple products for promotional material. Although he rarely speaks publicly about that work, in one interview he mentions in passing that some shoots require complex setups that enable him to "control each highlight and shadow independently" [232].

Apple's business model is very different from Google's, and these divergent incentive schemes push them to behave differently. Google is, of course, the other major software player in the smartphone supply chain.[83] Smartphone buyers today—which is to say nearly everyone—are forced to pick between their divergent economic tactics. Either one signs up for Apple's planned obsolescence or Google's privacy invading surveillance. In practice, one often gets both.

The primary alternative to iPhones is Android. In the Android ecosystem, Google provides handset manufacturers with an open source operating system called Android Open Source Project [239]. AOSP is a bare bones operating system. Google puts all of its software, including its Play Store, Maps, Chrome browser, and superior camera app, into a separate software package called Google Play Services [240], [241]. To ship phones with Google's high quality software layer, which sits atop AOSP, handset manufacturers like Samsung must agree to follow Google's implementation rules [125], [242]–[244]. These contracts allow Google to collect the personal data of every Android user, no matter which brand of phone they are using.

To provide users with free services, Google sells user attention to their real customers: advertisers.[84] In recent years, the meme that "if you're not the customer, you're the product" has gain renewed steam [29]. If a company is providing a free service, like YouTube or GMail, then it is recouping those costs somewhere else, most likely at your expense. The idea has been circulating with regard to television advertising since at least the 1970s [245].

Apple, meanwhile, earns the vast majority of its revenues from products and services sold to the people that use them. In Apple's business model, the customer is almost always the user. The company has recently decided to lean into this contrast. In January 2019, Apple launched an advertising campaign with a simple tagline "Privacy. That's iPhone" [246]. Still, this model has its own drawbacks.

According to its business model, Apple is motivated to engage in planned obsolescence as much as Google is motivated to collect information about its users. Despite Apple's claims that its business model incentivizes the company to respect customer privacy more than its competitors, Apple's sources of revenue also drive the company to engage in aggressive planned obsolescence.

---

[83] Outside of China.

[84] I discuss advertising based organisms at greater length in Chapter 4: Making Networks.

Apple generates record sales and profits thanks to its software-infused approach to planned obsolescence. Anyone who has held off updating their device for fear of hurting its performance is familiar with Apple's tactics. In 2017, the company was caught red-handed, after a software update it introduced deliberately slowed phones with older batteries, reducing compute performance by as much as half [247], [248]. The company claimed this software was introduced to "smooth" device performance and avoid sudden unexpected behavior, but the changes resulted in exactly the opposite effect. iPhones with batteries deemed worn by the operating system would suddenly shut down with 30-40% charge still remaining [247]. The so-called "Batterygate" revelation confirmed customers' long held suspicions and sowed lasting distrust in the company's commitment to delivering quality products. The strategy could be called *OTA Obsolescence*, because it harms performance with an over-the-air software update.[85]

OTA obsolescence is complementary to the electronics industry practice of *software throttling*, in which manufacturers use software to limit hardware performance. For instance, Panasonic limits the performance of their prosumer camcorders and sells a serial key to update the firmware and unlock the hardware's full potential for $200 [249]. Tesla, too, artificially limits the performance of its car batteries to create three price tiers for a given model, despite only manufacturing two different hardware specifications [250]. In the lead up to Hurricane Irma's Florida landfall in 2017, Tesla removed the software throttling of their mid-range Model 3 cars, which are sold as 60KWh cars, but actually sport 72KWh batteries inside [251]. In software throttling, a manufacturer uses software to differentiate hardware into different SKUs, despite it being cheaper to manufacture fewer, higher power models. These business motivated restrictions further indicate that hardware can be made more or less performant with OTA updates.

The efficacy of iPhone throttling and high priced battery replacements were laid bare in Apple's subsequent financial performance. In response to the "Batterygate" software throttling revelations, Apple reduced its replacement battery price from $79 USD to $29 USD in late 2017 [252]. These price changes affected the Christmas 2017 sales season and all of 2018. In January 2019, CEO Tim Cook announced that this price drop had caused many of their customers to not upgrade their phones [253]. This change contributed to the company earning approximately $9 billion less revenue than expected [254]. From this

---

[85] The acronym OTA. stands for "over-the-air." The term is used to describe software updates delivered to devices directly, especially over Wi-Fi. Before OTA updates, a device would have to be tethered to a computer to update.

event, we can confidently surmise that raising barriers to repair and releasing software that artificially throttles performance increases average revenue per user (ARPU).

I propose that we group Apple's product designs under the banner of Cupertino Design. First, in product marketing, sleek cases and all-in-one designs attract customers with their distinguished look and luxury prices. These integrated product designs communicate Apple's "just works" consumer electronics brand. To keep margins high and repeat purchases frequent, glued-in batteries, tamper-proof screws, and warranties that prohibit disassembly discourage product owners from replacing defective or worn parts on their own. Expensive first-party repairs encourage customers to upgrade devices when a simple part swap would satisfy their needs. When the hardware does not fail fast enough, Apple performs software sabotage, which I have called OTA obsolescence, to accelerate the upgrade cycle. These products work well, but at the expense of longevity and reparability. To maintain their high margins and user demand for new products, Apple flexes its refined approach to design to create best-in-class products that nevertheless need to be replaced within a handful of years.

AirPods are the culmination of Apple's Cupertino Design style. AirPods are a tiny, delightful wireless compute product made of batteries, plastic, and integrated circuits. Although they will work with any Bluetooth-capable device, they work best with other Apple products. Launched in 2016, the first AirPods iteration featured long, conspicuous white stems, which contain the slender cylindrical battery that is glued in place [255].[86] Their unique silhouette and signature white plastic material are recognizable at a distance, and their luxury price and unmissable look make them a mark of wealth [257], [258].

AirPods epitomize Jony Ive's design legacy. AirPods are tightly packaged disposable computers made of plastic, batteries, and glue. In the constrained set of use-cases for which they are precisely designed, they work wonderfully. When removing one AirPod, the photodiode sensors located on the portion of the earbud that goes inside the ear detect that they have been removed and pause music or media playback immediately. Upon reinsertion into the ear, playback starts up automatically. These delightful touches of user experience attention to detail elevate AirPods to the highest ranks of consumer electronics product design. These are a 'great product' in the 2017 sense of the phrase: they provide best-in-class user

---

[86] In fact, Apple launched a very similar Bluetooth earbud device in 2008 called the Apple iPhone Bluetooth Headset. It was discontinued shortly thereafter [256].

experience for individuals, however their impact on the environment, and thus the collective, may be less favourable.

Apple claims that it works hard to protect the environment, but its Cupertino style product designs make recycling unlikely. Every year, during their iconic keynote addresses, Apple executives take a few moments of each product announcement to emphasize the company's dedication to responsible sourcing, safe material use, and recycling [259]. In 2016, the company took this one step further with the debut of the first in a series of sizzle reels for its custom-built recycling robots. The company claims robots like its 2016 "Liam" machine, can disassemble 200 iPhones in an hour [260]. Unfortunately, the recycling reality is not so rosy. Apple devices are frequently sold and resold in the used, refurbished, and stolen gray markets [261]. Devices regularly reach end of life thousands of kilometres away from where they were originally purchased—far beyond the reach of Apple's recycling programs [262]. Realistically, only a small fraction of the devices the company manufactures will ever make it to an Apple recycling plant. The company's aforementioned product packaging decisions and bare minimum recycling documentation policies make their green robotics programme appear more marketing than truth.

The prognosis is even worse for AirPods. Judging by product teardowns, it is difficult to imagine anybody bothering to slice open the battery compartment of a glue-laden earbud to separate the handful of cheap components stuck inside its diminutive plastic casing. "Gluing products together, hiding the batteries away—that all makes recycling more difficult, less profitable, and more dangerous," remarked the CEO of a large electronics recycler, shortly after AirPods launched [263]. If one wants to reduce the impact of consumer electronics on the environment, then one must surely prioritize reparability, recyclability, and long-lived product designs. Every pair of AirPods is destined to be e-waste. Like all lithium-ion devices, AirPods batteries show wear after 500 to 1000 charges. Their packaging make them impossible to repair and, because their batteries are glued in place, it is even dangerous to put them in a trash compactor. In Jonathan Sterne's words, "they are designed to be trash" [264, p. 19].

Section 2: Meaning is Waste

Since beginning my journey into speculative industrial design in 2014, the moral problem of waste has repeatedly stymied my creative practice. During the first few years creating research prototypes, I have frequently been delayed by existential uncertainty about working with electronics. I know that these materials are the ones that fascinate me most, but their origins and end are abhorrent. How can I reconcile the urgent feeling that I must personally address computation, including hardware, with a moral disdain for the provenance and ultimate destination of the physical substrate?

At the outset of my experimental research-creation projects, I naively presumed that *real* industrial designers and engineers would have better strategies for object-making ready at hand. Someone with adequate training, I figured, would be able to work with superior materials, select better components, and know how to 3D model clever assemblies with ease. In fact, as I have shown in Section 1: Cupertino Design, the most widely respected industrial design professionals embrace disposability in most of their work. Although they have greater resources to dedicate to recycling campaigns and public relations, ultimately their most celebrated designs marry plastic, glue, batteries, and integrated circuits into often irreparable packages.

There are at least three problems working with computers as a medium. Computers are made of conflict minerals, they are often manufactured in terrible labor conditions, and they become obsolete in short order [265]–[271].

For empathic people, these worldly problems become personal. How could I make and iterate upon objects made of environmentally unfriendly materials? How could I ethically use as art supplies components made of conflict minerals that are assembled by children [272]–[274]? Pondering my own research, I imagine the materials traveling along an arcing trajectory, from mine extraction to garbage dump. My research prototypes achieve their purpose only during the fleeting moment at the top of the curve when they pass in front of a camera to be documented for this dissertation.

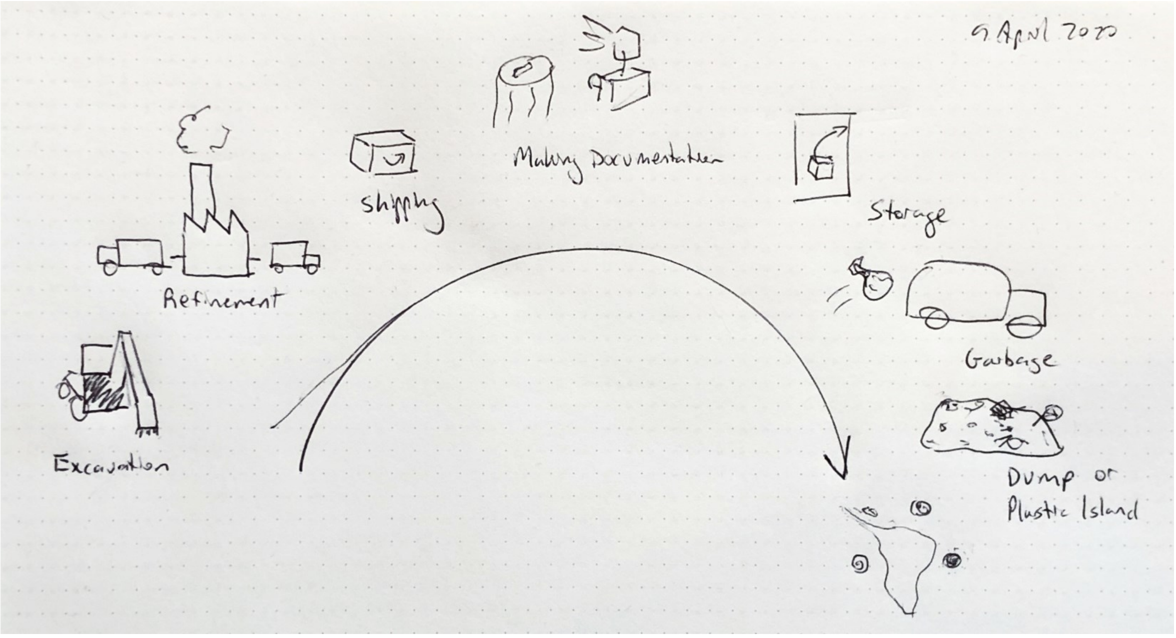


Figure 31 Trajectory of compute art supplies.

I have no good answers to these questions, but I have made some discoveries about the concept of waste that I would like to share, and it begins with a simple question: *what is waste?*

There are two pertinent definitions of waste that are sometimes confused. As a noun, waste is the material left over after manufacturing or use [275]. Waste (n.) refers to the refuse at the end of a process. As a verb, to waste means to squander [276]. Wasting (v.) something means to consume or expend it uselessly. I believe that this latter version, waste as a verb, is in need of rescue.

So often today, we are bombarded with conflicting messages that waste is bad, but that the choices we make as consumers are at once our most powerful political tool [277, p. 162]. It is news to no one that consumer capitalism traps us in an impossible quandary, were we are enjoined to support the local economy and simultaneously reproached for patterns of consumption that threaten the planet. We are at once incited to spend and blamed for the consequences of our spending.

One common refrain amongst anti-consumerist advocates is the return to earlier patterns of living where mass production and consumption were less present. I observe this tendency in what contemporary radical Marxists call "degrowth," or the effort to live according to patterns other than perpetual economic expansion [278]–[280]. Degrowth and related anti-consumer capitalist movements argue that the ecosystems that sustain human, animal and plant life are fragile and about to reach an irreversible tipping point. In a world of finite resources, it is unrealistic to build our economies and communities upon faith in unlimited growth. In popular culture, degrowth inspires people to pursue collective agriculture projects and collective living arrangements, sometimes outside of city-centres and the trappings of techno-capitalist modern life.

The appeal to reduce waste is perfectly logical. Economic pressures drive companies to increase revenue at all costs. As a result, we find ourselves consuming without cause and generating waste without reason. Why use a Swiffer™ when I could use a mop? Why package vegetables, liquids, and candies in their own individual plastic wrappers, when alternatives could surely exist? Disposable products and superfluous packaging are undoubtedly unsustainable. The same could be said of individual shipments from ecommerce retailers, the bi-annual smartphone upgrade cycle, vehicle ownership, and so many other modes of consumption, too.

The logical conclusion of anti-waste rhetoric is to justify all activity in terms of efficiency. Sensitive people are the first to feel the weight of this line of thinking. It goes without saying that one should avoid certain especially wasteful products, but where can one possibly fix the limit? The justifiable argument to reduce waste naturally leads sympathetic people to question the value of all their actions in terms of consumption and waste. Of course, single-use cleaning supplies are wasteful, but so too is everything else one does. Burning gasoline in a car is wasteful, but so is ordering something online that will have to be shipped. Raising animals for meat requires more water and feed than just vegetables, so that is undoubtedly wasteful, but large scale agriculture, too, can be damaging to the flora and fauna. If every material extracted and every consumable created is perceived as impinging on the sanctity of nature, then every human activity becomes marked with the blood of our ecosystem. Sustaining ourselves with food, clothes, and comforts has undeniable environmental impact. Is having children wasteful, then, too? Concluding that the world would be better off without humans is not far off.

This line of thinking is rational. If human life requires so many resources to sustain, and humans appear destined to multiply in number and wasteful desires until we exceed the carrying capacity of the Earth, then how can any of it be justified in terms of efficiency and worthwhileness? From this perspective, every action humans take and every desire we indulge may well be evaluated as wasteful, unethical, and contributing to the destruction of life as we know it on the planet. The natural conclusion is to worry about every action one takes, and to fear deep down that perhaps human existence itself is a blight upon our heretofore pristine planet. All forms of expression and exploration above and beyond the maintenance of life are up for critical analysis, and further human propagation appears deserving of scrutiny. However well-intentioned this ecologically-minded ethic was originally, it accidentally flattens some subtler distinctions that deserves a closer look.

Anti-waste reasoning confuses the serious and imminent threat to the biosphere with an efficiency-based valuation of human behavior. By beginning down the road of dividing life into wasteful and justified activities, we accidentally smuggle an engineering-like efficiency mindset into our assessments of human practices. Instead of focusing specifically on the damaging ecological impact of certain patterns of consumption, the mental category of "waste" asserts value judgments about what is worthwhile and what is unnecessary human activity. This approach inevitably discounts new forms of expression as wasteful indulgences, as we shall see.

In contrast to anti-waste rhetoric, I would like to argue that waste is a subjective category that bears further investigation. A deeper understanding of what counts as waste, and what does not, will help us to flesh out new solutions to the systemic problems causing ecological destruction. What's more, new forms of human expression often start off looking like waste, so I suggest that we should be wary of judging too quickly.

From the position of efficiency, the visual arts, for instance, can look like an exorbitant excess. How can one possibly justify the use of paint and canvas for painting, or of printing for photography, or physical materials refined through chemical processes for sculpture, in light of the impending ecological apocalypse? From a strict anti-waste logic, in which only survival and biological continuity is easily justified, perhaps art is a luxurious excess. The case for art is still harder to argue, for the outcome of artistic expression cannot be guaranteed at the outset. It would be bad enough to make one painting, but what if it is not even any good? Perhaps poetry, cinema, and music, too, are not sufficiently productive activities to merit resource allocation. If measured by the yard stick of radical conservationism it is unclear which human arts are indubitably redeemable.

Even language can look like waste, from a sufficiently critical angle. When we talk, joke, and write, we are using surplus energy to express ourselves and make new forays into the fields of ideas and emotion. To pre-human creatures that could not express themselves, talking undoubtedly looked like waste. Why bother with all that yipping and yammering? Yet today, humans spend nearly all of our time simulating in language matters entirely disconnected from our immediate physical circumstances. Ecologically unfriendly behavior threatens our biosphere, but there is something too imprecise in waste as an evaluative and decision-making criterion.

I believe that the concept of waste is in need of rescue, because *meaning is waste*. Contrary to the now prevalent view that waste will be our downfall, I believe that waste is the material basis of creativity. If life can be said to have purpose or value, it undoubtedly rests in those parts of living that are above and beyond mere efficient survival. Indeed, the origins of human expression are found in activity that can easily be mistaken for waste. The fight to protect the environment that makes human life possible is about protecting our capacity to express ourselves and relate, not to create maximally efficient living conditions in which expression is permitted only if it can be justified by its forecasted outcomes. Consumption, waste, and excess are the very basis of human expression, in language and any other medium.

New forms of expression come from our ability to afford to waste, from language to computing. In 1993, in an interview with Kevin Kelly for *Wired Magazine*, computer prognosticator George Gilder argued that "Over the last 30 years, we've seen transistors (or switching power) move from being expensive, crafted vacuum tubes to being virtually free. So today, the prime rule of thrift in business is 'waste transistors.' We 'waste' them to correct our spelling, to play solitaire, to do anything. As a matter of fact, you've got to waste transistors in order to succeed in business these days" [281]. Is this really waste then, after all?

When new technologies turn something expensive into something very cheap, the significance of that thing changes, and so too does the society around it. This is Marshall McLuhan's core observation in *Understanding Media: The Extensions of Man* [78]. When some human activity, like travel, goes from being very slow, expensive, and dangerous, to being fast, affordable, and safe, then its meaning changes. "The railway did not introduce movement or transportation or wheel or road into human society, but it accelerated and enlarged the scale of previous human functions, creating totally new kinds of cities and new kinds of work and leisure" [78, p. 10]. The locomotive did not invent long-distance travel, but its invention introduced a step change in accessibility that transformed human relations by reducing the time and difficulty of travelling between distant places. While a long distance trip just to see someone might have looked wasteful before the train, such a trip became more understandable once it was invented. This change in technology spurred unforeseen change in human relations.

If travel were to experience another step change in accessibility, making it significantly cheaper, faster, and safer, then its nature would change yet again, and human life around it. Let us imagine a scenario to make the point clearer. Today, the environmental costs associated with flying are making short trips increasingly socially unacceptable [282]–[285]. If tomorrow, a company were to make available a means of inexpensive, safe, and ecologically less harmful teleportation, it would undoubtedly be used in ways that would be considered wasteful by today's measures. The knock-on effects of such a step change in the medium of life are hard to predict. If teleportation were suddenly abundant, it might no longer be unusual to send a loved one a hug in person. If we could teleport at will, then perhaps it would be normal to grab a local ingredient from the other side of the world. If traveling were so transformed, its use might more closely resemble texting than commercial flight. Popping in and out of spaces for altogether frivolous reasons could become the norm, or contribute to new social practices that would seem bizarre today. Of course, such a technology might also cause new forms of damage to the ecosystem or others' quality of life—externalities, as economists say. Nevertheless, with these historical and hypothetical examples, we can observe that step changes in access can cause social norms to depart radically from those that dominated during the antecedent era. Notions of waste depend upon assumptions of technological availability and access. As these variables change over time, what is considered wasteful changes, too.

What one person calls waste, another might call a form of communication. When the value of human activity is measured against the yardstick of efficiency, new forms of expression outside of the norm are cast aside. Underground cultures and human interests that do not conform to the dominant ideology can easily attract the ire of those for whom they have no personal meaning or value. Fashion, for instance, is a very popular means of expression that is nevertheless misunderstood by many as a sheer deleterious vanity. Although many people enjoy to shop and dress up, the domain of fashion is an obvious target for environmentally-conscious disdain, in part because it is not considered worthwhile. The expressive potential of feminized, racialized, and otherwise belittled patterns of living are not wasteful, just because they exist outside of mainstream productivity narratives. A closer reading of fashion reveals why people are attracted to the outputs of ecologically damaging industrial practices, and gives hints at practical alternatives.

Fast fashion is perhaps the most derided branch of the fashion industry today. Zara and H&M are two retailers synonymous with this fashion industry shift towards selling bargain basement versions of ripped-from-the-runway looks [286]. These retailers borrow inspiration from haute couture shows with a heavy hand and release their new SKUs to stores within a matter of weeks. Traditional fashion retailers produce new items in a six to nine month production process. By contrast, Zara produces 20 collections per year and sends new items to stores twice per week [287]. Fast fashion brands like these are regularly pilloried for their low quality and relative disposability compared to more familiar fashion retailers and the luxury brands whose styles they ape [288]–[290]. These clothes cost very little, are made poorly, and end up in the trash quicker than ever before. Their low cost and cheap construction convince customers that they are not worth hanging onto or repairing.

New online-only retailers have begun to threaten these market leaders with even quicker *ultrafast* fashion cycles. Like social media platforms, ultrafast fashion retailers test market demand online before ordering production en masse. Their manufacturing processes are also sped up. ASOS takes products from the drawing board to its online store in six weeks, on average, and Boohoo does the same in just two weeks [287]. Missguided claims to bring products from concept to its webstore in under a week [287]. Internet analytics allow ultrafast fashion retailers to integrate market demand much more closely into the production schedule. Rather than set the trends, these fashion retailers offer various alternatives of quickly put together clothing visuals, and customer response on social media and online stores informs which styles survive. Ultrafast fashion accelerates tailoring and sewing so that clothes look more like memes than ever before. Still the question lingers: why are so many people inclined to treat clothes like visual culture?

Basic clothes are for warmth, but fashion is a means of expression. Garments can be more than just ways to keep warm and hide one's genitals. Like language and computers, clothes can be a symbolic system for creating, storing, and communicating information. Clothes can permit a person to express where they are from, what they believe in, how they perceive themselves, and how they want to be perceived. They can reveal things about the wearer that they did not intend to communicate, and they can lead viewers astray in their suppositions, too. To the sartorially-inclined, *clothes are parts of speech*. In sophisticated social contexts, what one wears can express just as much as it can provide protection from the elements. A pair of skinny jeans or one's choice of shoes can communicate even more than a charming sentence. It is easier to understand the place apparel has in human life if it is regarded as a representational scheme, or means of communication, rather than mere vanity.

People are drawn to fast fashion because it expands their vocabulary. Viewed as parts of speech, the human inclination to consume and discard clothes and other commodities becomes less repulsive, and easier to understand. People do not go shopping at H&M and Fashion Nova simply because they are uncaring robots of consumption. It is perfectly human to want to buy and wear clothes that fall apart quickly when clothes are a means of expression. Fast and ultrafast fashion put the eloquence of haute couture within reach of the masses. Brands like Boohoo and ASOS give less wealthy people access to the premium memes of luxury at an affordable price. If I buy a dress for $12 and wear it twice, I am using the clothing exactly like I might a sentence or a joke. In the present industrial moment, clothes are so abundantly available that some people can afford to discard them after use.

Buying single use clothes is a form of symbiosis with the supply chain. In a socio-economic moment of abundance and low wage manufacturing labour, clothing is more available than ever before. A person in a relatively well-off country does not need their clothes to last a long time, if the supply chain will remain reliably available to fulfill their demand in the future. Putting aside the environmental impacts of this lifestyle for just a moment, it is easy to perceive yet another synchrony with network computational living. If Google and Wikipedia are available, then memorizing becomes less important. So long as the fast fashion supply chain is so financially incentivized, there will always be more clothes to come.

If we focus on fashion's expressive potential, more environmentally friendly alternatives come into view. Digital clothes are one hint at a future of an ecologically friendlier style of supply chain-backed human expression.

Despite the growing number of people playing video games, the concept of purchasing and coveting digital clothes continues to baffle the masses. For all those who are not yet invested in a particular digital world, the idea of spending "real world" money on "virtual clothes" for an avatar is ridiculous. To me this connection makes perfect sense and speaks to the true nature of clothes in the first place.

Freemium games demonstrate that people already invest in virtual fashion as a means of communication. In freemium games, which are also called free-to-play games, players can participate without paying any upfront fee. While some games sell upgrades that advance a player through the story more quickly, recent hits have preferred a fairer business model. Games like Valve's DOTA 2, Riot Games' League of Legends, and Epic's Fortnite only sell items that provide players purely cosmetic upgrades. In these games, there is no competitive advantage to spending money. Nevertheless, players who spend hundreds of hours in-game develop personal relationships with other players, and establish a sense of self identity in that proprietary space. Once invested in a game's world, players are more inclined to spend money to look unique. Although cosmetic microtransaction games provide no way to pay to tip the game's mechanics in one's favour, players nevertheless shell out an extraordinary amount of money on these popular titles. League of Legends generated $1.7 billion of revenue in 2018, while Fortnite earned $2.4 billion in that same year [291], [292]. These games sell exclusively cosmetic upgrades. For these reasons, I prefer to think of LoL, DOTA, and Fortnite as fashion brands that happen to also create the virtual worlds in which their collections are available for sale. These wildly successful virtual apparel retailers redefine what it means to be vertically integrated.

Fashion houses like DOTA and Fortnite show that the same impulse to fashion expression can be enabled with an entirely different supply chain. As life becomes increasingly entwined with online experience, it makes perfect sense that the proclivity to speak with what one wears would also emerge in digital form. Clothes are a way of expressing oneself, and that principle applies just as much in a virtual context as offline. As our lives become increasingly connected with avatars and other digital representations, the underlying virtual and symbolic dimension to human consumption is laid bare. Fashion is a form of representation bound up with identity and social perception. Freemium games show that socially sidelined subcultures may contain valuable insights into seemingly unrelated domains of human expression.

Meaning emerges from parts of life that can look like waste to the cynical. In this chapter, I have argued that waste is a subjective category that is perhaps too imprecise a tool for evaluating what is worth pursuing in a human life. Like network software applications, the supply chain is involved in identity and memetic gestation. The supply chain produces parts of speech that people use to express themselves. The abundant availability of new memes of expression enables sometimes unforeseeable changes in culture and values. Until we can accept that the supply chain is a symbol factory, as much as an object factory, it will be impossible to imagine new interfaces and manufacturing techniques that reduce pollution while leaving open the doors of human expression.

Conclusion

In the introduction, I introduced this research-creation dissertation project with a few guiding questions:

> *How do technology, media, and abstraction relate? How are technologies extensions of language? Are new interfaces designed or discovered? How do computer and human labor relate in the context of an interface? How does symbiosis with network computation feel? How does one summon a network or grow an ecosystem? How do novel interfaces and product forms become socially accepted? How do networks change the meaning of art and design practices? Is network making an art form?* [87]

Network computing shapes our understanding of the world and of ourselves. How are we meant to address this medium, however we choose to define ourselves professionally?

To answer these questions, I have attempted to build a world where it is natural to see in terms of networks. This is a world grounded in the incentive mechanisms that shape the institutions around us, in which we must know the science and magic by which networks are born, strive, and thrive. The goal of this dissertation has been to present network computing as a subject of philosophical inquiry, artistic experimentation, and subjective experience that is open to creative intervention.

To build this discursive world, the dissertation's five chapters and appendix provide six analogical perspectives on networks, computing, and experience. Chapter 1 provides a technical and philosophical foundation for the text. This chapter argues that computation is a medium of abstraction, inside and out. Chapter 2 reframes software making for network audiences as a performance art, where artists can transform attention into resources to summon new society-shaping network organisms. Chapter 3 argues that memes are networks and networks are memes. Like memes, network organisms must compete in fashion games to become popular, and like networks, popular memes provide context for communication between strangers. Chapter 4 discusses the making of networks. Networks become more valuable as they become more populated. To get growth started, history shows it helps to enable some transgressive behaviour. Chapter 5 connects the supply chain to network computation. This chapter posits that meaning is grounded in waste and argues that people treat mass produced physical goods like parts of speech whose value is primarily symbolic—and so we return to abstraction, where we began. Finally, the

---

[87] See page 14.

appendix charts my personal research-creation journey working at the edge of fashion and camera computing, a new frontier in abstraction design.

Each of the aforementioned sections contributes to a worldview. Our lives, beliefs, opportunities, and suffering are increasingly informed by organisms much larger than ourselves. Networks unfold according to the self-perpetuation mechanisms these organisms discover. Our lives, values, and dreams are affected in turn. I hope that the ideas presented above are interesting, entertaining, and possibly inspiring in future work.

Works Cited

[1] S. Kemp, ‘Digital 2020: Global Digital Overview’, *DataReportal – Global Digital Insights*, Jan. 30, 2020. https://datareportal.com/reports/digital-2020-global-digital-overview (accessed Aug. 07, 2020).

[2] J. Stryjak and M. Sivakumaran, ‘The Mobile Economy 2019’, GSMA Intelligence, 2019. Accessed: Mar. 01, 2020. [Online]. Available: https://www.gsma.com/mobileeconomy/.

[3] ‘World Population Prospects - Population Division - United Nations’. https://population.un.org/wpp/Download/Standard/Population/ (accessed Sep. 07, 2020).

[4] A. Kay and A. Goldberg, ‘Personal Dynamic Media’, *Computer*, vol. 10, no. 3, pp. 31–41, Mar. 1977, doi: 10.1109/C-M.1977.217672.

[5] V. Bush, ‘As We May Think’, *The Atlantic*, Jul. 01, 1945. https://www.theatlantic.com/magazine/archive/1945/07/as-we-may-think/303881/ (accessed Sep. 10, 2020).

[6] D. C. Engelbart, ‘Augmenting human intellect: a conceptual framework’, Stanford Research Institute, Menlo Park, California, Summary Report AFOSR-3223, Oct. 1962. Accessed: Sep. 10, 2020. [Online].

[7] S. Levy, ‘The Downside of Policing Violent Videos | Backchannel’, *Wired*, May 07, 2017.

[8] G. Collins, ‘Future shock: Why there’ll be no flying cars’, *Google News Archive*, Charleston, South Carolina, Dec. 12, 1992.

[9] A. Witze, ‘How a small nuclear war would transform the entire planet’, *Nature*, vol. 579, no. 7800, Art. no. 7800, Mar. 2020, doi: 10.1038/d41586-020-00794-y.

[10] B. J. Cardinale *et al.*, ‘Biodiversity loss and its impact on humanity’, *Nature*, vol. 486, no. 7401, Art. no. 7401, Jun. 2012, doi: 10.1038/nature11148.

[11] S. Díaz, J. Fargione, F. S. C. Iii, and D. Tilman, ‘Biodiversity Loss Threatens Human Well-Being’, *PLOS Biol.*, vol. 4, no. 8, p. e277, Aug. 2006, doi: 10.1371/journal.pbio.0040277.

[12] R. McLeman, ‘Thresholds in climate migration’, *Popul. Environ.*, vol. 39, no. 4, pp. 319–338, Jun. 2018, doi: 10.1007/s11111-017-0290-2.

[13] ‘A Degree of Concern: Why Global Temperatures Matter – Climate Change: Vital Signs of the Planet’. https://climate.nasa.gov/news/2865/a-degree-of-concern-why-global-temperatures-matter/ (accessed Sep. 10, 2020).

[14] ‘List of Websites Blocked in Mainland China’, *Wikipedia*. Feb. 18, 2020, Accessed: Aug. 07, 2020. [Online]. Available:

https://en.wikipedia.org/w/index.php?title=List_of_websites_blocked_in_mainland_China&oldid=971021292.

[15] 'Chinese holiday island to unlock Facebook, Twitter for foreigners', *South China Morning Post*, Jun. 22, 2018. https://www.scmp.com/news/china/policies-politics/article/2152102/chinese-holiday-island-unlock-facebook-twitter-and (accessed Sep. 10, 2020).

[16] 'Iran Bars Social Media Again After a Day - The New York Times'. https://www.nytimes.com/2013/09/18/world/middleeast/facebook-and-twitter-blocked-again-in-iran-after-respite.html (accessed Sep. 10, 2020).

[17] S. Yasir and H. Kumar, 'India Bans 118 Chinese Apps as Indian Soldier Is Killed on Disputed Border', *The New York Times*, Sep. 02, 2020.

[18] ET Bureau, 'Chinese apps banned in India: India bans 59 Chinese apps including TikTok, WeChat, Helo - The Economic Times', *The Economic Times*, Jul. 29, 2020.

[19] A. Swanson, M. Isaac, and P. Mozur, 'Trump Targets WeChat and TikTok, in Sharp Escalation With China', *The New York Times*, Aug. 06, 2020.

[20] D. Tafazoli, M. E. G. Parra, and C. A. H. Abril, 'Computer Literacy: Sine Qua Non for Digital Age of Language Learning & Teaching', *Theory Pract. Lang. Stud.*, vol. 7, no. 9, Art. no. 9, Sep. 2017, doi: 10.17507/tpls.0709.02.

[21] 'About Us', *ICDL Europe*. https://icdleurope.org/about-us/ (accessed Sep. 08, 2020).

[22] 'CCC | Fingerprint Biometrics hacked again'. https://www.ccc.de/en/updates/2014/ursel (accessed Sep. 08, 2020).

[23] 'Japan researchers warn of fingerprint theft from "peace" sign'. https://phys.org/news/2017-01-japan-fingerprint-theft-peace.html (accessed Sep. 08, 2020).

[24] I. Echizen and T. Ogane, 'BiometricJammer: Use of Pseudo Fingerprint to Prevent Fingerprint Extraction from Camera Images without Inconveniencing Users', in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct. 2018, pp. 2825–2831, doi: 10.1109/SMC.2018.00481.

[25] 'Chinese "gait recognition" tech IDs people by how they walk', *AP NEWS*, Nov. 06, 2018. https://apnews.com/bf75dd1c26c947b7826d270a16e2658a (accessed Sep. 08, 2020).

[26] K. Bashir, T. Xiang, and S. Gong, 'Gait recognition without subject cooperation', *Pattern Recognit. Lett.*, vol. 31, no. 13, pp. 2052–2060, Oct. 2010, doi: 10.1016/j.patrec.2010.05.027.

[27] S. Krishnamoorthy, L. Rueda, S. Saad, and H. Elmiligi, 'Identification of User Behavioral Biometrics for Authentication Using Keystroke Dynamics and Machine Learning', in *Proceedings of the 2018*

2nd International Conference on Biometric Engineering and Applications, New York, NY, USA, May 2018, pp. 50–57, doi: 10.1145/3230820.3230829.

[28] 'AncestryDNA® | DNA Tests for Ethnicity & Genealogy DNA Test'. https://www.ancestry.ca/dna/ (accessed Sep. 10, 2020).

[29] J. Naughton, '"The goal is to automate us": welcome to the age of surveillance capitalism', *The Observer*, Jan. 20, 2019.

[30] C. Darwin and P. C. Darwin, *On the Origin of Species: A Facsimile of the First Edition*. Harvard University Press, 1964.

[31] J. Ma, 'Alibaba Founder Jack Ma: Ideas & Technology Can Change the World', *YouTube* -, Jun. 19, 2013. https://www.youtube.com/watch?v=euxJhgYZXL8&feature=youtu.be&t=30m31s (accessed Aug. 29, 2020).

[32] G. M. Weinberg, *An Introduction to General Systems Thinking*. .

[33] C. Staff, 'Computer Science Is Not a Science', *Communications of the ACM*, Jan. 2013. https://cacm.acm.org/magazines/2013/1/158757-computer-science-is-not-a-science/fulltext (accessed Aug. 31, 2020).

[34] J. A. Stone, D. P. Kitlan, M. E. Hoffman, and D. R. Vance, 'Cultural, sociological, and experiential challenges for CIS education', *J. Comput. Sci. Coll.*, vol. 23, no. 5, pp. 212–216, May 2008.

[35] Ubiquity Staff and P. J. Denning, 'Ubiquity: An Interview with Peter Denning on the great principles of computing', *Ubiquity*. https://ubiquity.acm.org/article.cfm?id=1276163 (accessed Aug. 31, 2020).

[36] V. G. Cerf, 'Where Is the Science in Computer Science?', *Communications of the ACM*, Oct. 2012. https://cacm.acm.org/magazines/2012/10/155530-where-is-the-science-in-computer-science/fulltext (accessed Aug. 31, 2020).

[37] T. Nelson, *Ted Nelson on Pernicious Computer Traditions*. 2008.

[38] T. Nelson, *Computers for Cynics 0 - The Myth of Technology*. 2012.

[39] G. Deleuze, *Foucault*. A&C Black, 2006.

[40] A. A. Lovelace and L. F. Menabrea, 'Sketch of the analytical engine invented by Charles Babbage, by LF Menabrea, officer of the military engineers, with notes upon the memoir by the translator', *Taylors Sci. Mem.*, vol. 3, pp. 666–731, 1842.

[41] N. Nisan and S. Schocken, *The elements of computing systems: building a modern computer from first principles*, 1. MIT Press pbk. ed. Cambridge, Mass.: MIT Press, 2008.

[42] *Transistors, How do they work ?* Learn Engineering, 2016.

[43] B. Brungard, *Registers and RAM: Crash Course Computer Science #6*. 2017.

[44] John Walker, 'The Analytical Engine: Is the Emulator Authentic?' https://www.fourmilab.ch/babbage/authentic.html (accessed Aug. 07, 2020).

[45] J. G. Brainerd and T. K. Sharpless, 'The ENIAC', *Electr. Eng.*, vol. 67, no. 2, pp. 163–172, Feb. 1948, doi: 10.1109/EE.1948.6443970.

[46] I. James, 'Claude Elwood Shannon 30 April 1916 — 24 February 2001', *Biogr. Mem. Fellows R. Soc.*, vol. 55, pp. 257–265, Dec. 2009, doi: 10.1098/rsbm.2009.0015.

[47] C. E. Shannon, 'A symbolic analysis of relay and switching circuits', *Electr. Eng.*, vol. 57, no. 12, pp. 713–723, Dec. 1938, doi: 10.1109/EE.1938.6431064.

[48] 'constants - Arduino Reference', *The Arduino Reference*, Aug. 07, 2020. https://www.arduino.cc/reference/en/language/variables/constants/constants/ (accessed Aug. 07, 2020).

[49] B. Brungard, *Boolean Logic & Logic Gates: Crash Course Computer Science #3*. 2017.

[50] C. R. Nave, 'AND Gate', *HyperPhysics*. http://hyperphysics.phy-astr.gsu.edu/hbase/Electronic/and.html (accessed Aug. 07, 2020).

[51] B. Brungard, *How Computers Calculate - the ALU: Crash Course Computer Science #5*. 2017.

[52] R. White and T. E. Downs, *How Computers Work*, 10 edition. Indianapolis, IN: Que Publishing, 2014.

[53] P. Falstad, 'Full Adder', *Circuit Simulator*. https://www.falstad.com/circuit/e-fulladd.html (accessed Aug. 07, 2020).

[54] W. Toomey, 'Logic Gates - Building an ALU', 2013. http://www.csc.villanova.edu/~mdamian/Past/csc2400fa13/assign/ALU.html (accessed Aug. 07, 2020).

[55] A. Clements, *Principles of computer hardware*, 4th ed. Oxford ; New York: Oxford University Press, 2006.

[56] J. C. Scott, *But how do it know?: the basic principles of computers for everyone*. Oldsmar, FL: John C. Scott, 2009.

[57] *The Central Processing Unit (CPU): Crash Course Computer Science #7*. 2017.

[58] 'No Bugs' Hare, 'Modified Harvard Architecture: Clarifying Confusion', *IT Hare on Soft.ware*, Sep. 21, 2015. http://ithare.com/modified-harvard-architecture-clarifying-confusion/ (accessed Aug. 07, 2020).

[59] Oxford English Dictionary, ‘machine, n.’, *OED Online*. Oxford University Press, Accessed: Aug. 07, 2020. [Online]. Available: https://www.oed.com/viewdictionaryentry/Entry/111850.

[60] Oxford English Dictionary, ‘algorithm, n.’, *OED Online*. Oxford University Press, Accessed: Aug. 07, 2020. [Online]. Available: https://www.oed.com/view/Entry/4959.

[61] M. Davis, ‘Logic and the development of the computer’, in *Handbook of the History of Logic*, vol. 9, Elsevier, 2014, pp. 31–38.

[62] J. Ferreirós, ‘The crisis in the foundations of mathematics’, *Princet. Companion Math.*, pp. 142–156, 2008.

[63] A. Hodges, *Alan Turing: The Enigma: The Enigma*. Random House, 2012.

[64] J. Montelius, ‘Elixir, functional programming and the Lambda calculus.’ 2018.

[65] A. M. Turing, ‘On Computable Numbers, with an Application to the Entscheidungsproblem. A Correction’, *Proc. Lond. Math. Soc.*, vol. s2-43, no. 1, pp. 544–546, 1938, doi: 10.1112/plms/s2-43.6.544.

[66] L. De Mol, ‘Turing Machines’, in *The Stanford Encyclopedia of Philosophy*, Winter 2019., E. N. Zalta, Ed. Metaphysics Research Lab, Stanford University, 2019.

[67] C. Arthur, ‘Apps more popular than the mobile web, data shows’, *the Guardian*, Apr. 02, 2014. http://www.theguardian.com/technology/appsblog/2014/apr/02/apps-more-popular-than-the-mobile-web-data-shows (accessed Sep. 02, 2020).

[68] ‘There’s an App for That’, 1043881, Jan. 24, 2012.

[69] E. Johnson, ‘Full transcript: Political consultant Bradley Tusk on Recode Decode’, *Vox*, Oct. 10, 2016. https://www.vox.com/2016/10/10/13231894/bradley-tusk-ventures-uber-politics-recode-decode-podcast-transcript (accessed Aug. 07, 2020).

[70] N. Scheiber, ‘Growth in the “Gig Economy” Fuels Work Force Anxieties’, *The New York Times*, Jul. 12, 2015.

[71] N. Eyal, *Hooked: How to Build Habit-Forming Products*. Penguin Publishing Group, 2014.

[72] J. Berger, *Ways of Seeing*. Peter Smith Pub Incorporated, 1990.

[73] E. S. Herman and N. Chomsky, *Manufacturing Consent: The Political Economy of the Mass Media*. Knopf Doubleday Publishing Group, 2011.

[74] C. Cadwalladr, ‘“I made Steve Bannon’s psychological warfare tool”: meet the data war whistleblower’, *The Guardian*, Mar. 18, 2018.

[75] Quote Investigator, ‘We Shape Our Tools, and Thereafter Our Tools Shape Us’, *Quote Investigator*. https://quoteinvestigator.com/2016/06/26/shape/ (accessed Aug. 07, 2020).

[76] J. Tolentino, ‘The Age of Instagram Face’, *The New Yorker*. https://www.newyorker.com/culture/decade-in-review/the-age-of-instagram-face (accessed Aug. 07, 2020).

[77] Quote Investigator, ‘If Your Only Tool Is a Hammer Then Every Problem Looks Like a Nail’, *Quote Investigator*. https://quoteinvestigator.com/2014/05/08/hammer-nail/ (accessed Aug. 07, 2020).

[78] M. McLuhan, *Understanding Media: The Extensions of Man*, 4th Printing edition. Signet Books, 1964.

[79] L. M. Adleman, ‘Computing with DNA’, *Sci. Am.*, vol. 279, no. 2, pp. 54–61, 1998.

[80] P. W. K. Rothemund, ‘A DNA and restriction enzyme implementation of Turing Machines.’, *DNA Based Comput.*, vol. 27, pp. 75–119, 1996.

[81] A. Churchill, S. Biderman, and A. Herrick, ‘Magic: The Gathering is Turing Complete’, *ArXiv190409828 Cs*, Apr. 2019, Accessed: Aug. 07, 2020. [Online]. Available: http://arxiv.org/abs/1904.09828.

[82] R. Kaye, ‘Infinite versions of minesweeper are Turing-complete’, *Manuscript*, Aug. 2000, [Online]. Available: http://web.mat.bham.ac.uk/R.W.Kaye/minesw/infmsw.pdf.

[83] N. T., ‘Is Minecraft Turing-Complete?’, *StackExchange*, Apr. 17, 2011. https://gaming.stackexchange.com/questions/20219/is-minecraft-turing-complete (accessed Aug. 07, 2020).

[84] G. Branwen, ‘Surprisingly Turing-Complete’, Dec. 2012, Accessed: Aug. 07, 2020. [Online]. Available: https://www.gwern.net/Turing-complete.

[85] M. Hoff, ‘Always Already Programming’, *Github*, Feb. 03, 2020. https://gist.github.com/melaniehoff/95ca90df7ca47761dc3d3d58fead22d4 (accessed Aug. 07, 2020).

[86] S. Krishnamurthi, ‘Can MS Excel be considered a functional programming language?’, *Quora*. https://www.quora.com/Can-MS-Excel-be-considered-a-functional-programming-language/answer/Shriram-Krishnamurthi (accessed Aug. 07, 2020).

[87] K. L. Yandoli, ‘Kim Kardashian West Has Her Own Emoijs And They’re Pretty Amazing’, *BuzzFeed*. https://www.buzzfeed.com/krystieyandoli/kim-kardashian-has-her-own-emoijs-and-theyre-pretty-amazin (accessed Aug. 07, 2020).

[88] J. I. Wong, ‘Apple’s creating a new sticker economy, and Kim Kardashian is going to dominate it’, *Quartz*. https://qz.com/727610/apples-creating-a-new-sticker-economy-and-kim-kardashian-is-going-to-dominate-it/ (accessed Aug. 07, 2020).

[89] D. Karwatka, ‘Joseph Marie Jacquard and the punched card textile loom’, *Tech Dir.*, vol. 59, no. 4, Nov. 1999, Accessed: Aug. 14, 2020. [Online]. Available: https://search.proquest.com/openview/3a427277248c3ffbe07af61e42d2df2f/1.pdf?pq-origsite=gscholar&cbl=182.

[90] B. Randell, Ed., *The Origins of Digital Computers: Selected Papers*, 2nd ed. Berlin Heidelberg: Springer-Verlag, 2013.

[91] P. 01 J. 2019 | 19:44 GMT, ‘The Jacquard Loom: A Driver of the Industrial Revolution - IEEE Spectrum’, *IEEE Spectrum: Technology, Engineering, and Science News*, Jan. 01, 2019. https://spectrum.ieee.org/the-institute/ieee-history/the-jacquard-loom-a-driver-of-the-industrial-revolution (accessed Aug. 14, 2020).

[92] A. Taylor, ‘The Automation Charade’, *Logic Magazine*. https://logicmag.io/failure/the-automation-charade/ (accessed Aug. 07, 2020).

[93] L. von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum, ‘reCAPTCHA: Human-Based Character Recognition via Web Security Measures’, *Science*, vol. 321, no. 5895, pp. 1465–1468, Sep. 2008, doi: 10.1126/science.1160379.

[94] J. Lung, ‘Ethical and legal considerations of reCAPTCHA’, in *2012 Tenth Annual International Conference on Privacy, Security and Trust*, Jul. 2012, pp. 211–216, doi: 10.1109/PST.2012.6297942.

[95] ‘reCAPTCHA’, *reCAPTCHA*. https://www.google.com/recaptcha/about/ (accessed Sep. 02, 2020).

[96] ‘Think it. Jott it. Do it.’ https://jotts.typepad.com/jott/ (accessed Sep. 02, 2020).

[97] ‘Jott Leaves Beta, Continues To Do One Thing Awesome’, *TechCrunch*. https://social.techcrunch.com/2008/08/21/jott-leaves-beta-continues-to-do-one-thing-awesome/ (accessed Sep. 02, 2020).

[98] K. Hara, A. Adams, K. Milland, S. Savage, C. Callison-Burch, and J. Bigham, ‘A Data-Driven Analysis of Workers’ Earnings on Amazon Mechanical Turk’, *ArXiv171205796 Cs*, Dec. 2017, Accessed: Aug. 07, 2020. [Online]. Available: http://arxiv.org/abs/1712.05796.

[99] ‘Amazon Mechanical Turk’. https://www.mturk.com/ (accessed Sep. 02, 2020).

[100] A. Semuels, ‘The Internet Is Enabling a New Kind of Poorly Paid Hell’, *The Atlantic*, Jan. 23, 2018. https://www.theatlantic.com/business/archive/2018/01/amazon-mechanical-turk/551192/ (accessed Sep. 02, 2020).

[101] B. Price, ‘Frank and Lillian Gilbreth and the motion study controversy, 1907-1930’, in *A Mental Revolution: Scientific Management Since Taylor*, Ohio State University Press, 1992.

[102] D. S. Ferguson, ‘Don’t call it “time and motion study.”’, *IIE Solut.*, vol. 29, no. 5, pp. 22–24, May 1997.

[103] M. J. Nadworny, ‘Frederick Taylor and Frank Gilbreth: Competition in Scientific Management’, *Bus. Hist. Rev.*, vol. 31, no. 1, pp. 23–34, 1957, doi: 10.2307/3111727.

[104] A. McKinlay and J. Wilson, ‘“All they lose is the scream”: Foucault, Ford and mass production’, *Manag. Organ. Hist.*, vol. 7, no. 1, pp. 45–60, Feb. 2012, doi: 10.1177/1744935911427219.

[105] J. Paxton, ‘Mr. Taylor, Mr. Ford, and the Advent of High-Volume Mass Production: 1900-1912’, *Wayne State Coll. Econ. Bus. J. Inq. Perspect.*, vol. 4, no. 1, p. 17, 2012.

[106] B. Eater, ‘8-bit Computer Update’, Mar. 09, 2016. https://www.youtube.com/watch?v=HyznrdDSSGM&list=PLowKtXNTBypGqImE405J2565dvjafglHU (accessed Aug. 07, 2020).

[107] ‘Scratch - Imagine, Program, Share’. https://scratch.mit.edu/ (accessed Sep. 02, 2020).

[108] ‘What is Max? | Cycling ’74’. https://cycling74.com/products/max (accessed Sep. 02, 2020).

[109] ‘Introduction to Blueprints’. https://docs.unrealengine.com/en-US/Engine/Blueprints/GettingStarted/index.html (accessed Sep. 02, 2020).

[110] ‘Project Bloks - Project Bloks research into tangible programming is complete’, *Project Bloks*. https://projectbloks.withgoogle.com/ (accessed Sep. 02, 2020).

[111] ‘The complete littleBits library of STEAM and Coding lessons’. https://classroom.littlebits.com/welcome (accessed Sep. 02, 2020).

[112] ‘Dynamicland’. https://dynamicland.org/ (accessed Sep. 02, 2020).

[113] A. Holovarty, ‘Chrome’s web audio change is bad news | Holovaty.com’, May 09, 2018. http://www.holovaty.com/writing/chrome-web-audio-change/ (accessed Aug. 07, 2020).

[114] M. Arrington, ‘New Facebook Redesign More Than Aesthetic’, *TechCrunch*, Sep. 05, 2006. https://social.techcrunch.com/2006/09/05/new-facebook-redesign-more-than-just-aesthetics/ (accessed Aug. 07, 2020).

[115] S. Biddle, P. V. Ribeiro, and T. Dias, ‘Invisible Censorship: TikTok Told Moderators to Suppress Posts by “Ugly” People and the Poor to Attract New Users’, *The Intercept*, Mar. 16, 2020. https://theintercept.com/2020/03/16/tiktok-app-moderators-users-discrimination/ (accessed Aug. 07, 2020).

[116] ‘How TikTok recommends videos #ForYou - Newsroom | TikTok’, *TikTok Newsroom*. https://newsroom.tiktok.com/en-us/how-tiktok-recommends-videos-for-you (accessed Aug. 07, 2020).

[117] B. Laurel, *Computers as Theatre*, 2 edition. Upper Saddle River, NJ: Addison-Wesley Professional, 2013.

[118] E. M. Rogers, *Diffusion of innovations*. Simon and Schuster, 2010.

[119] 'Elon Musk on Twitter: "Traffic is driving me nuts. Am going to build a tunnel boring machine and just start digging..." / Twitter', *Twitter*. https://twitter.com/elonmusk/status/810108760010043392 (accessed Aug. 07, 2020).

[120] 'FAQ', *The Boring Company*. https://www.boringcompany.com/faq (accessed Aug. 07, 2020).

[121] 'The Boring Company - Funding, Financials, Valuation & Investors', *Crunchbase*. https://www.crunchbase.com/organization/the-boring-company/company_financials (accessed Aug. 07, 2020).

[122] R. Copeland, 'Elon Musk's New Boring Co. Faced Questions Over SpaceX Financial Ties', *Wall Street Journal*, Dec. 17, 2018.

[123] J. Horowitz, 'Elon Musk's Boring Company is now raising money by selling flamethrowers', *CNNMoney*, Jan. 28, 2018. https://money.cnn.com/2018/01/28/news/musk-boring-company-flamethrowers/index.html (accessed Aug. 07, 2020).

[124] E. Musk, 'Elon Musk on Twitter: "After 50k hats, we will start selling The Boring Company flamethrower" / Twitter', *Twitter*. https://twitter.com/elonmusk/status/940056523304181762 (accessed Aug. 07, 2020).

[125] R. Amadeo, 'New Android OEM licensing terms leak; "Open" comes with a lot of restrictions', *Ars Technica*, Feb. 13, 2014. https://arstechnica.com/gadgets/2014/02/new-android-oem-licensing-terms-leak-open-comes-with-restrictions/ (accessed Aug. 11, 2020).

[126] 'CSI S.T.A.R. XR-5 FG-1508 Advanced Battle Rifle (Color: Trooper White)', *Evike.com*. https://www.evike.com/products/68300/ (accessed Aug. 07, 2020).

[127] E. Musk, 'Elon Musk on Twitter: "The rumor that I'm secretly creating a zombie apocalypse to generate demand for flamethrowers is completely false" / Twitter', *Twitter*. https://twitter.com/elonmusk/status/957494364028071936 (accessed Aug. 07, 2020).

[128] L. Matney, 'Elon Musk's Boring Co. raises $113 million to chase a pipe dream', *TechCrunch*, Apr. 16, 2018. https://social.techcrunch.com/2018/04/16/elon-musks-boring-co-raises-113-million-to-chase-a-pipe-dream/ (accessed Aug. 07, 2020).

[129] 'SEC FORM D: The Boring Company', *Securities and Exchange Commission*, Apr. 16, 2018. https://www.sec.gov/Archives/edgar/data/1737630/000173763018000001/xslFormDX01/primary_doc.xml (accessed Aug. 07, 2020).

[130] A. L. Hoffmann, N. Proferes, and M. Zimmer, '"Making the world more open and connected": Mark Zuckerberg and the discursive construction of Facebook and its users', *New Media Soc.*, vol. 20, no. 1, pp. 199–218, Jan. 2018, doi: 10.1177/1461444816660784.

[131] 'Facebook Expansion Enables More People to Connect with Friends in a Trusted Environment', *About Facebook*, Sep. 26, 2006. https://about.fb.com/news/2006/09/facebook-expansion-enables-more-people-to-connect-with-friends-in-a-trusted-environment/ (accessed Aug. 09, 2020).

[132] A. Gawer, 'Bridging differing perspectives on technological platforms: Toward an integrative framework', *Res. Policy*, vol. 43, no. 7, pp. 1239–1249, Sep. 2014, doi: 10.1016/j.respol.2014.03.006.

[133] T. Gillespie, 'The politics of "platforms"', *New Media Soc.*, vol. 12, no. 3, pp. 347–364, 2010.

[134] J. Gans, 'The Rise of Content Platforms', *Harvard Business Review*, Oct. 13, 2011.

[135] E. Goldman, 'The Complicated Story of FOSTA and Section 230', *First Amend. Law Rev.*, vol. 17, pp. 279–294, Apr. 2019.

[136] L. Vittoria, '28 Famous People You Could Meet on Raya, The Dating App for Celebrities', *Nylon*, Mar. 16, 2016. https://www.nylon.com/articles/celebrities-on-raya-app (accessed Aug. 07, 2020).

[137] A. Pardes, 'What Is Clubhouse, and Why Does Silicon Valley Care?', *Wired*, May 22, 2020.

[138] R. Ma and Y.-Y. Lu, 'Ep. 55: Kuaishou: The Anti-Douyin / TikTok?', *Tech Buzz China*, Nov. 08, 2019. https://www.techbuzzchina.com/episodes/ep-55-kuaishou-the-anti-douyin-tiktok (accessed Aug. 07, 2020).

[139] J. Elman, *Musical.ly's Alex Zhu on Igniting Viral Growth and Building a User Community | #ProductSF 2016*. Greylock Partners, 2016.

[140] B. Gilbert and D. J. Rosenthal, 'Season 5, Episode 8: TikTok', *Acquired*, Dec. 08, 2019. https://www.acquired.fm/episodes/tiktok (accessed Aug. 07, 2020).

[141] J. Livingston, *Founders at Work: Stories of Startups' Early Days*, 1st Corrected ed., Corr. 4th printing edition. Berkeley, Calif: Apress, 2009.

[142] S. Jurvetson and T. Draper, 'Viral Marketing', *Draper Fisher Jurvetson*, Mar. 21, 2007. https://web.archive.org/web/20070321005608/http://www.dfj.com:80/cgi-bin/artman/publish/steve_tim_may97.shtml (accessed Aug. 07, 2020).

[143] H. McCracken, 'How Gmail Happened: The Inside Story of Its Launch 10 Years Ago', *Time*, Apr. 01, 2014. https://time.com/43263/gmail-10th-anniversary/ (accessed Aug. 07, 2020).

[144] D. A. Vise and M. Malseed, *The Google Story*. New York: Delacorte Press, 2005.

[145] ‘Google’s New Email Service, Gmail, Under Fire for Privacy Concerns, Possible Wiretap Law Violations | Privacy Rights Clearinghouse’, *Privacy Rights Clearinghouse*, Jun. 01, 2004. https://privacyrights.org/resources/googles-new-email-service-gmail-under-fire-privacy-concerns-possible-wiretap-law (accessed Aug. 07, 2020).

[146] ‘Gmail Invitation Prices Crash’, *Wired*, Jun. 10, 2004.

[147] S. Ludwig, ‘Gmail finally blows past Hotmail to become the world’s largest email service’, *VentureBeat*, Jun. 29, 2012. https://venturebeat.com/2012/06/28/gmail-hotmail-yahoo-email-users/ (accessed Aug. 07, 2020).

[148] J. Elias and M. Petrova, ‘Google’s rocky path to email domination’, *CNBC*, Oct. 26, 2019. https://www.cnbc.com/2019/10/26/gmail-dominates-consumer-email-with-1point5-billion-users.html (accessed Aug. 07, 2020).

[149] Oxford English Dictionary, ‘network, n. and adj.’, *OED Online*. Oxford University Press, Accessed: Aug. 07, 2020. [Online]. Available: https://www.oed.com/view/Entry/126342.

[150] R. Dawkins, *The Selfish Gene: 30th Anniversary Edition*. OUP Oxford, 2006.

[151] C. James, ‘Fame in the Twentieth Century’, *Fame in the Twentieth Century*, 1993.

[152] M. Brooke, ‘Come Along, Do! (1898)’, *BFI Screenonline*. http://www.screenonline.org.uk/film/id/444430/ (accessed Aug. 07, 2020).

[153] T. de Vries, ‘The “Cinématographe Lumière” a Myth?’, Dec. 19, 2017. https://wichm.home.xs4all.nl/myth.html (accessed Aug. 07, 2020).

[154] M. Heidenry, ‘Introducing Florence Lawrence, Hollywood’s Forgotten First Movie Star’, *Vanity Fair*, May 25, 2018. https://www.vanityfair.com/hollywood/2018/05/florence-lawrence-first-movie-star-old-hollywood (accessed Aug. 07, 2020).

[155] A. Knight and A. Knight, *The Liveliest Art: A Panoramic History of the Movies*. Macmillan, 1978.

[156] B. Morris, ‘Diana Vreeland, Editor, Dies; Voice of Fashion for Decades’, *The New York Times*, Aug. 23, 1989.

[157] D. Vreeland, *D.V.* Harper Collins, 2011.

[158] L. I. Vreeland and L. Immordino-Vreeland, *Diana Vreeland: The Eye Has to Travel*. Koch.

[159] E. Dwight, ‘The Divine Mrs. V’, *New York Magazine*, Nov. 04, 2002. https://nymag.com/nymetro/shopping/fashion/features/n_7930/ (accessed Aug. 07, 2020).

[160] R. E. Nisbett and T. D. Wilson, ‘The halo effect: Evidence for unconscious alteration of judgments.’, *J. Pers. Soc. Psychol.*, vol. 35, no. 4, pp. 250–256, 1977, doi: 10.1037/0022-3514.35.4.250.

[161] D. Sacks, ‘Troy Carter: Fired By Lady Gaga And Loving It’, *Fast Company*, Jan. 13, 2014. https://www.fastcompany.com/3024171/step-up-troy-carter (accessed Aug. 07, 2020).

[162] M. Vakulenko, ‘To understand Beats you need to understand Lady Gaga’, *Medium*, Jun. 15, 2014. https://medium.com/@mvakulenko/to-understand-beats-you-need-to-understand-lady-gaga-e334de3da6d2 (accessed Aug. 07, 2020).

[163] A. Elberse and M. Christensen, ‘Lady Gaga (B)’, *Harv. Bus. Sch. Suppl.*, pp. 512–017, Aug. 2011.

[164] J. Jurgensen, ‘The Lessons of Lady Gaga’, *Wall Street Journal*, Jan. 29, 2010.

[165] M. H. Goldhaber, ‘The attention economy and the Net’, *First Monday*, Apr. 1997, doi: 10.5210/fm.v2i4.519.

[166] A. G. Martínez, ‘No, Data Is Not the New Oil’, *Wired*, Feb. 26, 2019.

[167] T. Hale, ‘Data is not the new oil’, *Financial Times*. http://ftalphaville.ft.com/2019/05/08/1557318291000/Data-is-not-the-new-oil/ (accessed Aug. 07, 2020).

[168] C. Delo, ‘Facebook to Partner With Acxiom, Epsilon to Match Store Purchases With User Profiles’, *AdAge*, Feb. 22, 2013. https://adage.com/article/digital/facebook-partner-acxiom-epsilon-match-store-purchases-user-profiles/239967 (accessed Aug. 07, 2020).

[169] K. Leetaru, ‘The Data Brokers So Powerful Even Facebook Bought Their Data - But They Got Me Wildly Wrong’, *Forbes*, Apr. 05, 2018. https://www.forbes.com/sites/kalevleetaru/2018/04/05/the-data-brokers-so-powerful-even-facebook-bought-their-data-but-they-got-me-wildly-wrong/ (accessed Aug. 07, 2020).

[170] J. Angwin, T. Parris Jr., and S. Mattu, ‘Facebook is quietly buying information from data brokers about its users’ offline lives’, *Business Insider*, Dec. 30, 2016.

[171] N. Singer, ‘Mapping, and Sharing, the Consumer Genome’, *The New York Times*, Jun. 16, 2012.

[172] T. Speicher *et al.*, ‘Potential for Discrimination in Online Targeted Advertising’, in *FAT 2018 - Conference on Fairness, Accountability, and Transparency*, New-York, United States, Feb. 2018, vol. 81, pp. 1–15, Accessed: Aug. 07, 2020. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01955343.

[173] O. Rask, ‘What is Programmatic Advertising? The Ultimate Guide’, *Match2One*, Jan. 25, 2019. https://www.match2one.com/blog/what-is-programmatic-advertising/ (accessed Aug. 07, 2020).

[174] ‘About Ad Auctions’, *Facebook Business Help Center*. https://www.facebook.com/business/help/430291176997542 (accessed Aug. 07, 2020).

[175] J. Nicas, ‘Apple News’s Radical Approach: Humans Over Machines’, *The New York Times*, Oct. 25, 2018.

[176] P. M. Napoli, ‘Special Issue Introduction: Big Data and Media Management’, *Int. J. Media Manag.*, vol. 18, no. 1, pp. 1–7, Jan. 2016, doi: 10.1080/14241277.2016.1185888.

[177] M. D. Smith and R. Telang, ‘Data Can Enhance Creative Projects — Just Look at Netflix’, *Harvard Business Review*, Jan. 23, 2018.

[178] T. Havens, ‘Media Programming in an Era of Big Data’, *Media Ind. J.*, vol. 1, no. 2, 2014, doi: http://dx.doi.org/10.3998/mij.15031809.0001.202.

[179] ‘Privacy Statement’, *Netflix*, Jul. 31, 2020. https://help.netflix.com/legal/privacy (accessed Mar. 19, 2020).

[180] D. Harris, ‘Netflix analyzes a lot of data about your viewing habits’, *GigaOm*, Jun. 14, 2012. https://gigaom.com/2012/06/14/netflix-analyzes-a-lot-of-data-about-your-viewing-habits/ (accessed Aug. 07, 2020).

[181] M. Ball, ‘“Quality” Is a Distraction – If It Exists at All (Netflix Misunderstandings, Pt. 6)’. https://redef.com/original/quality-is-a-distraction-if-it-exists-at-all-netflix-misunderstandings-pt-6 (accessed Aug. 09, 2020).

[182] *Tiger King (TV Mini-Series 2020) - IMDb*. .

[183] *Tidying Up with Marie Kondo*. Netflix, The Jackal Group, 2019.

[184] *Black Mirror*. Zeppotron, Channel 4 Television Corporation, Gran Babieka, 2011.

[185] *Indian Matchmaking*. 2020.

[186] ‘“attention optimizing design” - Google Scholar’. https://scholar.google.ca/scholar?hl=en&as_sdt=0%2C5&q=%22attention+optimizing+design%22&btnG (accessed Aug. 07, 2020).

[187] ‘“attention optimizing design” - Google Search’, *Google Search*. https://www.google.com/search?client=firefox-b-d&q=%22attention+optimizing+design%22 (accessed Aug. 07, 2020).

[188] B. Briscoe, A. Odlyzko, and B. Tilly, ‘Metcalfe’s Law is Wrong’, *IEEE Spectrum*, Jul. 01, 2006. https://spectrum.ieee.org/computing/networks/metcalfes-law-is-wrong (accessed Aug. 07, 2020).

[189] G. Gilder, ‘Metcalfe’s law and legacy’, *Forbes ASAP*, vol. 13, p. 12, 1993.

[190] W. Oremus, ‘Snapchat: Why teens’ favorite app makes the Facebook generation feel old.’, *Slate*, Jan. 29, 2015. https://slate.com/technology/2015/01/snapchat-why-teens-favorite-app-makes-the-facebook-generation-feel-old.html (accessed Feb. 19, 2020).

[191] A. Heath, ‘Teens still love Snapchat more than Facebook, but Instagram isn’t far behind’, *Business Insider*, Oct. 14, 2016. https://www.businessinsider.com/teens-prefer-snapchat-and-instagram-over-facebook-2016-10 (accessed Aug. 09, 2020).

[192] R. Molla, ‘Nearly half of U.S. teens prefer Snapchat over other social media’, *Vox*, Oct. 14, 2017. https://www.vox.com/2017/10/14/16471688/us-teens-use-snapchat-snap-social-media-facebook-twitter-instagram (accessed Aug. 09, 2020).

[193] J. E. Solsman, ‘YouTube has 1.8 billion users logged in and watching every month’, *CNET*, May 03, 2018. https://www.cnet.com/news/youtube-has-1-8-billion-users-logged-in-and-watching-every-month/ (accessed Aug. 07, 2020).

[194] M. Singh, ‘Douyin, TikTok app in China, hits 400 million daily active users’, *TechCrunch*, Jan. 07, 2020. https://social.techcrunch.com/2020/01/06/douyin-tiktok-app-in-china-hits-400-million-daily-active-users/ (accessed Aug. 07, 2020).

[195] G. Marvin, ‘TikTok tops 500 million users, says VP, as brands flock to the video platform’, *Marketing Land*, Sep. 24, 2019. https://marketingland.com/tiktok-tops-500-million-users-says-vp-as-brands-flock-to-the-video-platform-267970 (accessed Aug. 07, 2020).

[196] D. P. Reed, ‘The Law of the Pack’, *Harvard Business Review*, no. February 2001, Feb. 01, 2001.

[197] ‘About Grace Hopper’, *CHIPS Magazine*, Jun. 27, 2011. https://www.doncio.navy.mil/chips/ArticleDetails.aspx?ID=2265 (accessed Aug. 07, 2020).

[198] P. Graham, ‘Do Things that Don’t Scale’, *Paul Graham*, Jul. 2013. http://paulgraham.com/ds.html (accessed Aug. 07, 2020).

[199] A. Kay, *Founder School Session: The Future Doesn’t Have to Be Incremental*. 2014.

[200] D. Brown, *Human Universals*, 1st ed. New York: McGraw-Hill, 1991.

[201] B. M. Schwartz, ‘Hot or Not? Website Briefly Judges Looks’, *The Harvard Crimson*, Nov. 04, 2003. https://www.thecrimson.com/article/2003/11/4/hot-or-not-website-briefly-judges/ (accessed Aug. 07, 2020).

[202] K. A. Kaplan, ‘Facemash Creator Survives Ad Board’, *The Harvard Crimson*, Nov. 19, 2003. https://www.thecrimson.com/article/2003/11/19/facemash-creator-survives-ad-board-the/ (accessed Aug. 07, 2020).

[203] A. J. Tabak, ‘Hundreds Register for New Facebook Website’, *The Harvard Crimson*, Feb. 09, 2004. https://www.thecrimson.com/article/2004/2/9/hundreds-register-for-new-facebook-website/ (accessed Aug. 07, 2020).

[204] R. Alleyne, ‘YouTube: Overnight success has sparked a backlash’, *The Telegraph*, Jul. 31, 2008.

[205] T. Arah, ‘99% Flash Player Penetration – Too Good to be True?’, *Alphr*, Feb. 20, 2009. https://www.alphr.com/blogs/2009/02/20/99-percent-flash-player-penetration/ (accessed Aug. 07, 2020).

[206] *Me at the zoo*. 2005.

[207] T. Wasserman, ‘The revolution wasn’t televised: The early days of YouTube’, *Mashable*, Feb. 14, 2015. https://mashable.com/2015/02/14/youtube-history/ (accessed Aug. 07, 2020).

[208] T. Arango, ‘Titans of Old & New Media Mix & Match’, *New York Post*, Jul. 14, 2006. https://nypost.com/2006/07/14/titans-of-old-new-media-mix-match/ (accessed Aug. 07, 2020).

[209] B. Watson, *Jon Stewart: Beyond the Moments of Zen*. New Word City, 2014.

[210] N. Cohen, ‘YouTube Is Purging Copyrighted Clips’, *The New York Times*, Oct. 30, 2006. https://www.nytimes.com/2006/10/30/technology/30youtube.html (accessed Aug. 07, 2020).

[211] D. Itzkoff, ‘Nerds in the Hood, Stars on the Web’, *The New York Times*, Dec. 27, 2005.

[212] X. Jardin, ‘NBC nastygrams YouTube over “Lazy Sunday”’, *Boing Boing*, Feb. 17, 2006. https://boingboing.net/2006/02/17/nbc-nastygrams-youtu.html (accessed Aug. 07, 2020).

[213] A. M. Kerwin, ‘NBC Doesn’t Believe in Viral’, Feb. 27, 2006. https://adage.com/article/news/nbc-viral/108459 (accessed Aug. 07, 2020).

[214] H. Travis, *Cyberspace Law: Censorship and Regulation of the Internet*. Routledge, 2013.

[215] S. J. Baskin *et al.*, ‘VIACOM’S STATEMENT OF UNDISPUTED FACTS IN SUPPORT OF ITS MOTION FOR PARTIAL SUMMARY JUDGMENT ON LIABILITY AND INAPPLICABILITY OF THE DIGITAL MILLENNIUM COPYRIGHT ACT SAFE HARBOR DEFENSE’, p. 86, Mar. 2010.

[216] ‘Google To Acquire YouTube for $1.65 Billion in Stock’, *Google Press Center*, Oct. 09, 2006. https://web.archive.org/web/20070326192143/http://www.google.com/press/pressrel/google_youtube.html (accessed Aug. 07, 2020).

[217] P. R. La Monica, ‘Google buying YouTube - Oct. 9, 2006’, *CNN Money*. https://money.cnn.com/2006/10/09/technology/googleyoutube_deal/index.htm?cnn=yes (accessed Aug. 07, 2020).

[218] ‘You (Time Person of the Year)’, *Wikipedia*. Aug. 03, 2020, Accessed: Aug. 07, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=You_(Time_Person_of_the_Year)&oldid=970965357.

[219] E. Bangeman, ‘YouTube’s future (or lack thereof)’, *Ars Technica*, Oct. 03, 2006. https://arstechnica.com/information-technology/2006/10/7892/ (accessed Aug. 07, 2020).

[220] ‘supply, n.’, *OED Online*. Oxford University Press, Accessed: Aug. 11, 2020. [Online]. Available: http://www.oed.com/view/Entry/194665.

[221] O. Hopkins, ‘Jony Ive’s legacy as the most important designer of the last two decades is assured’, *Dezeen*, Jun. 28, 2019. https://www.dezeen.com/2019/06/28/jony-ive-legacy-apple-head-designer/ (accessed Sep. 02, 2020).

[222] L. E. Howorth Matt Peckham, Alex Fitzpatrick, John Patrick Pullen, Victor Luckerson, Claire, ‘The 50 Most Influential Gadgets of All Time’, *Time*, May 03, 2016. https://time.com/4309573/most-influential-gadgets/ (accessed Sep. 02, 2020).

[223] H. Dediu, ‘Determining The Average Apple Device Lifespan’, *Asymco*, Mar. 01, 2018. http://www.asymco.com/2018/03/01/determining-the-average-apple-device-lifespan/ (accessed Aug. 10, 2020).

[224] ‘Timeline of Apple Inc. products’, *Wikipedia*. Aug. 05, 2020, Accessed: Aug. 10, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Timeline_of_Apple_Inc._products&oldid=971371154.

[225] R. L. Lankey and F. C. McMichael, ‘Life-Cycle Methods for Comparing Primary and Rechargeable Batteries’, *Environ. Sci. Technol.*, vol. 34, no. 11, pp. 2299–2304, Jun. 2000, doi: 10.1021/es990526n.

[226] H. Stuever, ‘Battery And Assault’, *Washington Post*, Dec. 20, 2003.

[227] ‘Apple’s iPod: the best-selling digital music player in history’, *Financial Times*, Feb. 12, 2016.

[228] *iPod’s Dirty Secret*. 2003.

[229] M. Proske, J. Winzer, M. Marwede, N. F. Nissen, and K.-D. Lang, ‘Obsolescence of electronics - The Example of Smartphones’, in *2016 Electronics Goes Green 2016+ (EGG)*, Sep. 2016, pp. 1–8, doi: 10.1109/EGG.2016.7829852.

[230] ‘Snap-fit’, *Wikipedia*. Feb. 22, 2020, Accessed: Aug. 10, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Snap-fit&oldid=942086834.

[231] International Conference on Product Lifetimes and the Environment, T. Cooper, N. Braithwaite, M. Moreno, and G. Salvia, Eds., *Product Lifetimes and the Environment (PLATE) conference proceedings, Nottingham Trent University, Nottingham, 17-19 June 2015*. 2015.

[232] M. Shane, ‘The illusion of simplicity: photographer Peter Belanger on shooting for Apple’, *The Verge*, May 08, 2013. https://www.theverge.com/2013/5/8/4311868/the-illusion-of-simplicity-photographer-peter-belanger-on-shooting (accessed Aug. 10, 2020).

[233] *iphone6 Remove iCloud*. G-Lon, 2016.

[234] ‘How do I pronounce “Spudger”?’, *iFixit*, Feb. 21, 2017. https://help.ifixit.com/article/144-spudger-pronounce (accessed Aug. 11, 2020).

[235] K. Wiens, ‘Using copyright to keep repair manuals secret undermines circular economy’, *The Guardian*, Dec. 20, 2013. http://www.theguardian.com/sustainable-business/copyright-law-repair-manuals-circular-economy (accessed Aug. 10, 2020).

[236] ‘Apple’s Diabolical Plan to Screw Your iPhone’, *iFixit*, Jan. 20, 2011. https://www.ifixit.com/News/14279/apples-diabolical-plan-to-screw-your-iphone (accessed Aug. 10, 2020).

[237] J. Koebler, ‘Internal Documents Show Apple Is Capable of Implementing Right to Repair Legislation’, *Vice News*, Mar. 28, 2019. https://www.vice.com/en_us/article/d3mqna/internal-documents-show-apple-is-capable-of-implementing-right-to-repair-legislation (accessed Aug. 10, 2020).

[238] M. Stone, ‘Apple’s Independent Repair Program Is Invasive to Shops and Their Customers, Contract Shows’, *Vice News*, Feb. 06, 2020. https://www.vice.com/en_us/article/qjdjnv/apples-independent-repair-program-is-invasive-to-shops-and-their-customers-contract-shows (accessed Aug. 10, 2020).

[239] ‘Android Open Source Project’, *Android Open Source Project*, Aug. 11, 2020. https://source.android.com/ (accessed Aug. 11, 2020).

[240] ‘Overview of Google Play Services | Google APIs for Android’, *Google Developers*, Jul. 29, 2020. https://developers.google.com/android/guides/overview (accessed Aug. 11, 2020).

[241] ‘Google Play services’, *Google Play*, Aug. 10, 2020. https://play.google.com/store/apps/details?id=com.google.android.gms&hl=en_CA (accessed Aug. 11, 2020).

[242] J. Kastrenakes, ‘Google will start charging Android device makers a fee for using its apps in Europe’, *The Verge*, Oct. 16, 2018. https://www.theverge.com/2018/10/16/17984074/google-eu-android-licensing-bundle-chrome-search (accessed Aug. 11, 2020).

[243] H. Lockheimer, ‘Complying with the EC’s Android decision’, *Google*, Oct. 16, 2018. https://blog.google/around-the-globe/google-europe/complying-ecs-android-decision/ (accessed Aug. 11, 2020).

[244] C. Warzel and A. Ngu, ‘Google’s 4,000-Word Privacy Policy Is a Secret History of the Internet’, *The New York Times*, Jul. 10, 2019.

[245] ‘You’re Not the Customer; You’re the Product’, *Quote Investigator*, Jul. 16, 2017. https://quoteinvestigator.com/2017/07/16/product/ (accessed Aug. 10, 2020).

[246] M. Wuerthele, ‘“Privacy. That’s iPhone” ad campaign launches, highlights Apple’s stance on user protection’, *AppleInsider*, Mar. 14, 2019. https://appleinsider.com/articles/19/03/14/privacy-thats-iphone-ad-campaign-launches-highlights-apples-stance-on-user-protection (accessed Aug. 10, 2020).

[247] S. Gibbs, ‘Apple admits slowing older iPhones because of ageing batteries’, *The Guardian*, Dec. 21, 2017. http://www.theguardian.com/technology/2017/dec/21/apple-admits-slowing-older-iphones-because-of-flagging-batteries (accessed Aug. 10, 2020).

[248] S. Gibbs, ‘Apple reduces speed of iPhones as batteries wear out, report suggests’, *The Guardian*, Dec. 19, 2017. http://www.theguardian.com/technology/2017/dec/19/apple-iphone-reduce-speed-old-batteries (accessed Aug. 10, 2020).

[249] L. Powell, ‘What’s The Deal With The Panasonic Filmmaker Upgrade?’, *Focus Camera*, May 31, 2019. https://www.focuscamera.com/wavelength/panasonic-filmmaker-upgrade-s1/ (accessed Aug. 11, 2020).

[250] F. Lambert, ‘Tesla discontinues Model 3 Mid Range battery pack’, *Electrek*, Mar. 17, 2019. https://electrek.co/2019/03/17/tesla-discontinues-model-3-mid-range-battery-pack/ (accessed Aug. 11, 2020).

[251] M. Zetlin, ‘Elon Musk (Temporarily) Stops Throttling Florida Tesla Drivers’ Batteries So They Can Escape Irma’, *Inc.com*, Sep. 11, 2017. https://www.inc.com/minda-zetlin/elon-musk-removes-software-throttling-from-florida.html (accessed Aug. 11, 2020).

[252] ‘Apple apologises for iPhone slowdowns’, *BBC News*, Dec. 29, 2017.

[253] T. Cook, ‘Letter from Tim Cook to Apple investors’, *Apple Newsroom*, Jan. 02, 2019. https://www.apple.com/newsroom/2019/01/letter-from-tim-cook-to-apple-investors/ (accessed Aug. 10, 2020).

[254] C. Gartenberg, ‘Apple says it could miss $9 billion in iPhone sales due to weak demand’, *The Verge*, Jan. 02, 2019. https://www.theverge.com/2019/1/2/18165804/apple-iphone-sales-weak-demand-tim-cook-letter-revised-q1-estimate (accessed Aug. 10, 2020).

[255] ‘AirPods Teardown’, *iFixit*, Dec. 20, 2016. https://www.ifixit.com/Teardown/AirPods+Teardown/75578 (accessed Aug. 10, 2020).

[256] D. E. Dilger, ‘Review: Apple iPhone Bluetooth Headset’, *AppleInsider*, Jul. 30, 2007. https://appleinsider.com/articles/07/07/30/review_apple_iphone_bluetooth_headset (accessed Aug. 10, 2020).

[257] ‘Urban Dictionary: I don’t speak broke.’, *Urban Dictionary*. https://www.urbandictionary.com/define.php?term=I%20don%27t%20speak%20broke. (accessed Aug. 11, 2020).

[258] R. Tarbet, ‘Sorry, I Don’t Speak Broke’, Feb. 11, 2019. https://www.tohsthelancer.org/entertainment/2019/02/11/sorry-i-dont-speak-broke/ (accessed Aug. 11, 2020).

[259] ‘Apple Events - Apple Special Events’, *Apple (CA)*. https://www.apple.com/ca/apple-events/ (accessed Aug. 11, 2020).

[260] C. Rujanavech, J. Lessard, S. Chandler, S. Shannon, J. Dahmus, and R. Guzzo, ‘Liam - An Innovation Story’, p. 8, Sep. 2016.

[261] ‘Secondary iPhone Market a Boon for AT&T, Verizon — and Apple, Too’, *AllThingsD*. http://allthingsd.com/20120117/secondary-iphone-market-a-boon-for-att-verizon-and-apple-too/ (accessed Aug. 14, 2020).

[262] M. Stopera, ‘I Followed My Stolen iPhone Across The World, Became A Celebrity In China, And Found A Friend For Life’, *BuzzFeed*, Mar. 31, 2015. https://www.buzzfeed.com/mjs538/i-followed-my-stolen-iphone-across-the-world-became-a-celebr (accessed Aug. 10, 2020).

[263] K. Wiens, ‘Try to Dissect Apple’s New AirPods and You’ll Shed Blood’, *Wired*, Dec. 21, 2016.

[264] J. Sterne, ‘Out with the Trash: On the Future of New Media’, in *Residual Media*, C. R. Acland, Ed. U of Minnesota Press, 2007, pp. 16–31.

[265] S. B. Young and G. Dias, ‘LCM of Metals Supply to Electronics: Tracking and Tracing “Conflict Minerals”’, Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 1875976, Apr. 2011. doi: 10.2139/ssrn.1875976.

[266] S. Taffel, ‘Towards an Ethical Electronics? Ecologies of Congolese Conflict Minerals’, *Westminst. Pap. Commun. Cult.*, vol. 10, no. 1, Art. no. 1, Sep. 2015, doi: 10.16997/wpcc.210.

[267] ‘Analyzing labor conditions of Pegatron and Foxconn: Apple’s low-cost reality’, *China Labour Watch*, Feb. 2015, [Online]. Available: https://digitalcommons.ilr.cornell.edu/cgi/viewcontent.cgi?article=3965&context=globaldocs.

[268] J. Chan and N. Pun, ‘Suicide as protest for the new generation of Chinese migrant workers: Foxconn, global capital, and the state’, *Asia-Pac. J.*, vol. 37, no. 2, pp. 1–50, 2010.

[269] W. Müller, ‘Foxconn economics: how much room for better pay and working conditions?’, *Flex. Workforce Low Profit Margins Electron. Assem. Eur. China*, p. 155, 2016.

[270] ‘Beyond Foxconn : Deplorable Working Conditions Characterize Apple’s Entire Supply Chain’, *China Labour Watch*, Jun. 2012, Accessed: Aug. 11, 2020. [Online]. Available: http://www.chinalaborwatch.org/report/62.

[271] B. Lüthje and F. Butollo, ‘Why the Foxconn Model Does Not Die: Production Networks and Labour Relations in the IT Industry in South China’, *Globalizations*, vol. 14, no. 2, pp. 216–231, Feb. 2017, doi: 10.1080/14747731.2016.1203132.

[272] J. Chan, N. Pun, and M. Selden, ‘Interns or workers? China’s student labor regime’, *Asian Stud.*, vol. 1, no. 1, pp. 69–98, 2015.

[273] J. Chan, M. Selden, and N. Pun, *Dying for an iPhone: Apple, Foxconn, and The Lives of China’s Workers*. Haymarket Books, 2020.

[274] C. Duhigg and D. Barboza, ‘In China, Human Costs Are Built Into an iPad’, *The New York Times*, Jan. 25, 2012.

[275] ‘waste, n.’, *OED Online*. Oxford University Press, Accessed: Aug. 10, 2020. [Online]. Available: http://www.oed.com/view/Entry/226027.

[276] ‘waste, v.’, *OED Online*. Oxford University Press, Accessed: Aug. 10, 2020. [Online]. Available: http://www.oed.com/view/Entry/226029.

[277] C. L. Wrenn, ‘How to help when it hurts? Think systemic’, *Anim. Stud. J.*, vol. 7, no. 1, pp. 149–179, 2018.

[278] G. Kallis, ‘In defence of degrowth’, *Ecol. Econ.*, vol. 70, no. 5, pp. 873–880, 2011.

[279] F. Sekulova, G. Kallis, B. Rodríguez-Labajos, and F. Schneider, ‘Degrowth: from theory to practice’, *J. Clean. Prod.*, vol. 38, pp. 1–6, 2013.

[280] G. D’Alisa, F. Demaria, and G. Kallis, *Degrowth: a vocabulary for a new era*. Routledge, 2014.

[281] K. Kelly, ‘George Gilder: When Bandwidth Is Free’, *Wired*, Apr. 01, 1993.

[282] M. Joselow, ‘TRANSPORTATION: Experts push back on growing “flight shame” movement’, Sep. 05, 2019. https://www.eenews.net/stories/1061110965 (accessed Aug. 13, 2020).

[283] P. Kalmus, ‘A climate scientist who decided not to fly’, *Grist*, Feb. 21, 2016. https://grist.org/climate-energy/a-climate-scientist-who-decided-not-to-fly/ (accessed Aug. 13, 2020).

[284] G. Topham, '"Flight-shaming" could slow growth of airline industry, says Iata', *The Guardian*, Oct. 17, 2019. http://www.theguardian.com/business/2019/oct/17/flight-shaming-could-slow-growth-of-airline-industry-says-iata (accessed Aug. 13, 2020).

[285] S. Randles and S. Mander, 'Aviation, consumption and the climate change debate: "Are you going to tell me off for flying?"', *Technol. Anal. Strateg. Manag.*, vol. 21, no. 1, pp. 93–113, Jan. 2009, doi: 10.1080/09537320802557350.

[286] C. Research, 'Fast Fashion Speeding Toward Ultrafast Fashion', *Coresight Research*, May 20, 2017. https://coresight.com/research/fast-fashion-speeding-toward-ultrafast-fashion/ (accessed Aug. 12, 2020).

[287] 'Report: "Ultra-fast" fashion players gain on Zara, H&M', *Retail Dive*. https://www.retaildive.com/news/report-ultra-fast-fashion-players-gain-on-zara-hm/443250/ (accessed Aug. 12, 2020).

[288] H.-M. Joung, 'Fast-fashion consumers' post-purchase behaviours', *Int. J. Retail Distrib. Manag.*, vol. 42, no. 8, pp. 688–697, Jan. 2014, doi: 10.1108/IJRDM-03-2013-0055.

[289] X. Long and J. Nasiry, 'Sustainability in the Fast Fashion Industry', Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 3486502, Nov. 2019. doi: 10.2139/ssrn.3486502.

[290] M. Collett, B. Cluver, and H.-L. Chen, 'Consumer Perceptions the Limited Lifespan of Fast Fashion Apparel', *Res. J. Text. Appar.*, vol. 17, no. 2, pp. 61–68, Jan. 2013, doi: 10.1108/RJTA-17-02-2013-B009.

[291] M. Perez, 'Is There Life After "League Of Legends"? Riot Bets Big On Its First New Game In 10 Years.', *Forbes*, Oct. 15, 2019. https://www.forbes.com/sites/mattperez/2019/10/15/is-there-life-after-league-of-legends-riot-bets-big-on-its-first-new-game-in-10-years/ (accessed Aug. 13, 2020).

[292] M. Perez, '"Fortnite" Creator Calls Analyst Revenue Reports "Wildly Inaccurate", But Still Won't Share Numbers', *Forbes*. https://www.forbes.com/sites/mattperez/2020/02/28/fortnite-creator-denies-revenue-decline-reported-by-nielsen-but-still-wont-share-numbers/ (accessed Aug. 13, 2020).

[293] B. Pauker, 'Epiphanies from Chris Anderson', *Foreign Policy*. https://foreignpolicy.com/2013/04/29/epiphanies-from-chris-anderson/ (accessed Aug. 27, 2020).

[294] 'US $0.79 |ZONBEMA Original Test Back Rear Camera With Flash Module Sensor Flex Cable For iPhone X XR XS 5 5S 5C SE 6 6S 7 8 Plus XS MAX|rear back camera|flex cameracamera flex cable - AliExpress', *aliexpress.com*.

//www.aliexpress.com/item/32834646273.html?src=ibdm_d03p0558e02r02&sk=&aff_platform=&aff_trace_key=&af=&cv=&cn=&dp= (accessed Aug. 18, 2020).

[295] M. Farnsworth, ‘Photos: Here’s what the new Amazon Go cashierless convenience store looks like’, *Vox*, Jan. 21, 2018. https://www.vox.com/2018/1/21/16913984/what-does-photos-amazon-jeff-bezos-seattle-new-no-cashier-line-grocery-story-amazon-go (accessed Aug. 18, 2020).

[296] Radio Times Staff, ‘Amazon Echo review - 3rd generation’, *Radio Times*, Aug. 12, 2020. https://www.radiotimes.com/technology/2020-08-12/amazon-echo-review-smart-speaker/ (accessed Aug. 18, 2020).

[297] A. Bacchus, ‘Microsoft HoloLens 2 Hands-On Review: The Future On Your Face’, *Digital Trends*, Nov. 08, 2019. https://www.digitaltrends.com/computing/microsoft-hololens-2-ar-hands-on-features-price-photos-video-release-date/ (accessed Aug. 18, 2020).

[298] S. Gibbs, ‘Google Glass review: useful – but overpriced and socially awkward’, *The Guardian*, Dec. 03, 2014.

[299] Facebook, ‘Oculus Connect 6’, *Oculus*. https://www.oculusconnect.com/ (accessed Aug. 18, 2020).

[300] M. Gurman, ‘Apple Is Ramping Up Work on AR Headset to Succeed iPhone’, *Bloomberg.com*, Nov. 08, 2017.

[301] M. Gurman, ‘Apple’s Secretive AR and VR Headset Plans Altered by Internal Differences’, *Bloomberg.com*, Jun. 19, 2020.

[302] J. Carpenter, *They Live*. Alive Films, Larry Franco Productions, 1988.

[303] S. Zizek, ‘What Rumsfeld Doesnt Know That He Knows About Abu Ghraib’, *In These Times*, May 21, 2004. https://inthesetimes.com/article/what-rumsfeld-doesn-know-that-he-knows-about-abu-ghraib (accessed Aug. 18, 2020).

[304] S. Fiennes, *The Pervert’s Guide to Ideology*. Blinder Films, British Film Institute (BFI), Film4, 2012.

[305] A. Mustafa, H. Kim, J.-Y. Guillemaut, and A. Hilton, ‘Temporally coherent 4D reconstruction of complex dynamic scenes’, *ArXiv160303381 Cs*, Mar. 2016, Accessed: Aug. 21, 2020. [Online]. Available: http://arxiv.org/abs/1603.03381.

[306] *Temporally coherent 4D reconstruction CVPR 2016(Oral)*. 2016.

[307] A. J. Champandard, ‘Neural Doodle’, *Github*, Mar. 05, 2016. https://github.com/alexjc/neural-doodle (accessed Aug. 27, 2020).

[308] ‘3D Print Your Own Glasses’, *Caret Dash Caret*, Jul. 22, 2014. https://caretdashcaret.com/2014/07/21/3d-print-your-own-glasses/ (accessed Aug. 23, 2020).

[309] ‘Cleric flags fatwa against Pokemon’, *BBC News*, Jul. 21, 2016.

[310] *Shenzhen: The Silicon Valley of Hardware (Full Documentary) | Future Cities | WIRED*. 2016.

[311] C. Shirky, *Little Rice: Smartphones, Xiaomi, and the Chinese Dream*. New York: Columbia Global Reports, 2015.

[312] *How to Find a Shipping agent in Huaqiangbei - Field Notes 31*. .

[313] *Dafen Oil Painting Village - Field Notes 34*. .

[314] *Shenzhen Music Instrument City Market - Field Notes 35*. .

[315] *Making Camera Jewelry in Shuibei and Huaqiangbei, Shenzhen - Field Notes*. .

[316] *Where to Buy Raspberry Pi and Arduino in Huaqiangbei, Shenzhen - Field Notes*. .

[317] *How to Manufafture Laser Cutters in Shenzhen - Field Notes*. .

[318] *Plastics Injection Molding: Step-By-Step at the Factory - Field Notes*. 2017.

[319] 'Chaihuo - HackerspaceWiki'. https://wiki.hackerspaces.org/Chaihuo (accessed Aug. 24, 2020).

[320] 'xfactory Home', *xfactory*. https://www.xfactory.io (accessed Aug. 24, 2020).

[321] B. Huang, *The Essential Guide to Electronics in Shenzhen by Bunnie Huang*, 1st ed. Singapore: Sutajio Ko-Usagi PTE LTD, 2016.

[322] A. B. Huang, *The Hardware Hacker: Adventures in Making and Breaking Hardware*, 1 edition. San Francisco, CA: No Starch Press, 2017.

[323] 'Radio in a Bag | Weil, Daniel | V&A Search the Collections', *V and A Collections*, Aug. 25, 2020. http://collections.vam.ac.uk/item/O85208 (accessed Aug. 25, 2020).

[324] A. Dunne, *Hertzian Tales: Electronic Products, Aesthetic Experience, and Critical Design*. MIT Press, 2005.

[325] 'Ancient jewelry headwear of Russian women', *Beauty will save*, Sep. 24, 2012. https://viola.bz/ancient-jewelry-headwear-of-russian-women/ (accessed Aug. 25, 2020).

[326] 'Upgrade-Ready Virtual Reality Backpack'. https://hackaday.io/project/167957-upgrade-ready-virtual-reality-backpack (accessed Aug. 26, 2020).

[327] D. Brennan, '5 VR Backpack PCs at a Glance', *Road to VR*, Jan. 29, 2018. https://www.roadtovr.com/vr-backpack-pc-at-a-glance/ (accessed Aug. 26, 2020).

[328] A. Robertson, 'I tried Magic Leap and saw a flawed glimpse of mixed reality's amazing potential', *The Verge*, Aug. 08, 2018. https://www.theverge.com/2018/8/8/17662040/magic-leap-one-creator-edition-preview-mixed-reality-glasses-launch (accessed Aug. 26, 2020).

[329] 'A Closer Look at ALYX's Kanye-Cosigned Chest Rig', *HYPEBEAST*. https://hypebeast.com/2018/4/alyx-chest-rig-release-details-hbx (accessed Aug. 26, 2020).

Appendix 1: Research-Creation

In the preceding chapters, I spelled out five analogies for understanding network computing. Each of these chapters addressed the subject from a distinct perspective: Abstraction, Performance, Fashion & Attention, Making, and the Supply Chain. In each, I engaged the subject through intellectual analysis.

In this appendix, I will present the research-creation projects with which the rest of the text has been in dialogue. Appendix 1 charts my research story from beginning to end. This appendix is split into short sections, each of which discusses a period of research experimentation. I will draw connections to prior chapters and provide new background information where necessary throughout this part of the text.

The purpose of this appendix is to document the dialogue between thinking, making, travel, and conversation that enabled me to write the prior chapters. This creative, multidisciplinary approach constitutes the sixth and final analogy for representing and thinking in the language of network computation.

Project Proposal

In my dissertation proposal, I laid out my reasoning for studying the design of interfaces to network computation through research-creation experiments.

First, new communication technologies change cultural norms in ways that cannot be predicted before their popularization. It is impossible to precisely predict how technologies will be adopted, because the hardware roadmap alone does not indicate how people will use and abuse a given bundle of scientific discoveries and human labour. For this reason, I chose to get my hands dirty and reflect upon what I discovered myself, in the hopes of finding something new.

Second, applications of new technologies are nevertheless grounded in the behaviours, tastes, and norms that precede their popularization. Successful applications provide comprehensible interfaces to unfamiliar technologies. To be legible, a software design must be visible to the present day while catalyzing the cultural change that invents tomorrow. With this idea in mind, I set out to use the memes around me to imagine new product forms and experimental directions.

Third, novel technology packages are popularized through collaboration between their creators, their early users, and the environment. A product or technology package may correctly foresee the inevitability of a new category of human behavior, but its cultural positioning can still fail to generate sufficient positive social interest to reach long term viability. To become parts of life, new product forms must build enough memetic momentum to become culturally relevant and socially acceptable. With this in mind, I conceived of my project as *network* experiments. The work would have to be a collaboration between me, the informal group of early users who play with my prototypes, and the unanticipated influences I encounter in my environment upon handling my own creations.

To expand and reflect upon these conceptual arguments, I proposed to design and iterate new network interfaces, in both hardware and software, and to document this firsthand experience in qualitative autoethnographic style. Although my experimentation process employed engineering materials, such as electronics components and 3D printing, my approach more closely resembled an art practice than engineering optimization. To avoid hewing too much to tradition, my project aimed to humbly find meaning through prototyping, rather than chase prematurely defined functional goals.

I chose to focus my research experimentation on network cameras. At the time I started the project, network cameras were poised to spawn the next generation of computing interfaces. In the wake of the smartphone wars, cameras, powerful SoCs, batteries, and networking hardware had all become incredibly affordable [293].[88] To wit, on August 18, 2020, an iPhone 5S 8-megapixel rear camera could be purchased for $0.78 on AliExpress—with no minimum order quantity [294]. When paired with recent advances in machine learning, these affordable, high resolution sensors were perfectly positioned to enable previously unimaginable applications. Our cultural associations with cameras are at present dominated by twentieth century media forms: the picture and the video. However, image sensor data is entirely unlike chemical photography processes. In a camera sensor, light is transformed into data. The ongoing renaissance in machine learning has begun to unlock the potential of this image sensor data, and researchers and product designers have only just begun to explore the plethora of interfaces these substrates enable.

With the contemporary supply chain as a backdrop, I set out to explore interface making and meme experimentation. Network cameras provided me with a well-balanced canvas. There was relatively little prior art for such interfaces in everyday life, and so my imagination and that of my audience was not yet polluted by expectations. At the same time, there were sufficient software and hardware resources for me to launch my experiments from the shoulders of giants.

---

[88] SoC is an acronym for system on a chip, a device that integrates the processor, RAM, graphics processing, I/O, and other functions into single integrated circuit. SoCs have several advantages over modular alternatives: they can be smaller, more power efficient, and largely prefabricated solutions for building new computing devices.

I planned to begin by creating and playing with modest functional prototypes (toys) to understand the material potential of new bundles of components. Where product and industrial designers might focus on demographics or known forms of computation, I intended to embrace personal interest and collaboration with the people around me with whom I shared my designs to find playful combinations that are meaningful to real people. Through play and sharing these toys with others, I believed I could quickly and intuitively orient subsequent iterations to address human behavior, rather than purely theoretical ambitions. If an iteration struck me as fun or interesting, I put more energy into it. An interface could be amusing and possibly even grow to be useful without attempting to solve present-day problems head-on.

With internet-connected cameras as my material, and open-minded design prototyping as my methodology, I embarked on my research-creation project. My goal was to see what I could learn by performing new interfaces and building new product forms. My process began with an observation about the present state of software development and a cinematic inspiration.

Ideological Lenses

Today, the most powerful technology corporations intend to use their large cash reserves and dominant market position to control the next wave of personal computing hardware, which they call virtual, augmented, and mixed reality—VR, AR, MR, or XR for short. Apple, Google, Facebook, Amazon, Alibaba, and Tencent have all released or are known to be developing XR software and hardware services. These projects span a wide range of physical scales. At the architectural scale, Amazon Go stores replace cash registers with machine vision cameras that monitor what each customer takes off the shelves and charge their Amazon accounts automatically. In living quarters-scale, devices such as Google Home and Amazon's Echo and Alexa products use microphones and machine learning to deliver software and product services in response to speech and household sounds [295], [296]. Google Glass, Microsoft HoloLens, Facebook Oculus, and the much rumoured Apple augmented reality products all point to the technology behemoths' intention to bring similar technologies to glasses-like head-mounted wearable devices [297]–[301]. These social, retail, software, and hardware powerhouses intend to use their advantageous positions in the smartphone-era to create popular devices and software that mediates or replaces the wearer's vision and hearing with layers of internet-infused proprietary software. If companies like Apple and Google successfully convert smartphone software services dominance to XR product dominance, then these large profit-optimizing corporations will be in a position to dictate how and what people perceive all of the time. While people already spend many hours a day on their phones already, the power to place proprietary software experiences directly in front of the eyes and in the ears marks an unprecedented level of influence over popular discourse and experience. If these corporations achieve their goals, they will have privatized at least two of the five senses.

At the outset of my research project, I felt there was an urgent need for independent designers and researchers take seriously wearable-scale hardware, cameras as input devices, and cutting edge machine learning software designs. Around me I observed many technology arts researchers still struggling to adapt to the smartphone as a locus of human attention and cognition, despite it being already ten years old at that time. I felt a sense of deep responsibility to conduct artistic and creative research experiments in network making with software powered internet cameras, because their presence in our lives was already becoming increasingly frequent, despite the general lack of public awareness.

Although wearable cameras challenge present day expectations of privacy, I imagined that the deployment of this technology by states and corporations was inevitable, and so perhaps the most ethical option amidst the incoming wave of surveillance technology would be to give access to such tools to as many people as possible. In 2016, as I prepared my first experiments, protests broke out in American cities over police unlawfully killing of black Americans. During the protests, it came to light that police that had once rejected body cameras as invasive of their workplace privacy, had begun to enthusiastically adopt the technology. This shift in police department policy came in large part because body camera providers had found ways to grant police departments and individual officers some amount of control over the recordings. Given the increasingly tense political climate, I wondered if it might not be important for protesters and political activists to have access to similar technology, so that a fuller version of conflicts between the two groups might be known to the public.

At the same time, I began to compare the technology oligopolies' desire to mediate all human perception of the world with Slavoj Zizek's classic analysis of the 1988 John Carpenter movie *They Live.* In *A Pervert's Guide to Ideology*, Zizek summarizes the film and provides an interesting analysis of his own. The movie follows the protagonist John Nada, a homeless blue collar worker in Los Angeles who discovers a pair of sunglasses that reveal the ideology underlying his surroundings. When he puts on the glasses, colourful media images and enticing marketing phrases are replaced by bold black text conveying more direct messages of conformity. A billboard for a computer console is replaced with the word "OBEY," a mural depicting a woman reclining on the beach becomes "MARRY AND REPRODUCE," and the pages of magazines read "STAY ASLEEP," "NO THOUGHT," and "DO NOT QUESTION AUTHORITY" [302]. Dollar bills in the newsstand attendant's hand say simply "THIS IS YOUR GOD" [302]. In the film, the glasses perform ideological critique for the wearer. The parallels to augmented reality applications are self-evident, with the exception that AR applications will no doubt *add* layers of ideologically laden media in accordance with their purveyors' vested interest in keeping consumers complacent and never questioning of their authority.

Zizek uses *They Live* glasses to explain ideologies, the unacknowledged assumptions embedded in our belief systems and worldviews. To borrow another Zizekian analogy, in May 2003, then US Secretary of Defense Donald Rumsfeld described the dangers of the unknown in the War on Terror. He explained that there are several categories of knowledge. "There are known knowns. These are things we know that we know. There are known unknowns. That is to say, there are things that we know we don't know. But there are also unknown unknowns. There are things we don't know we don't know" [303]. Zizek later pointed out that there is a fourth epistemological category that Rumsfeld's matrix implies, but which he personally failed to define. Unknown knows, Zizek claims, are the most dangerous category of knowledge of all. These are the things that operate within our thinking, but of which we are unaware [303]. These are our ideologies; the unacknowledged assumptions that underlie our approach to the world.

In Zizek's analysis, *They Live* glasses provide a model for understanding ideological critique. Our intuition might be that ideology is like the lens of a pair of glasses that obscures our clear vision of the world. In Zizek's words "this, precisely, is the ultimate illusion" [304]. Instead, the film suggests that "ideology is not simply imposed on ourselves. Ideology is our spontaneous relationship to our social world—how we perceive its meaning, and so on" [304]. The film's glasses reveal what we cannot see with our naked eyes.

I became fascinated with the juxtaposition of Zizek's take on *They Live* glasses and the forthcoming rise of XR. What would a pair of XR *They Live* glasses look like as a consumer product? A full-fledged XR system, with its sensing and 3D tracking displays, was well beyond my reach. I decided to carve out a meaningful chunk of the problem space. I began to experiment with cameras.

Prop Prototyping: Camera Props

Before deciding on the exact form my project would take, I began experimenting with my imagination alone. In my sketchbook, I took note of my intention to use objects that I already owned as if they performed new and different functions. I call this approach prop prototyping and I call these particular prototypes camera props. How might I hold my phone differently if it were constantly streaming camera sensor data? If cameras became wearable, how might photographic composition turn into more of a dance practice? If the device were even smaller, where might I perch it to observe a scene? How might I want to attach small cameras to my person or to the environment? To begin to answer these questions, I filmed with my phone as if I were wearing a sensor, and I swung its lens across surfaces as if it could capture textures. I kept a small USB battery in my pocket and pretended that it was a stand-alone internet camera that could fit in the palm of my hand. By using existing electronics and other physical objects as if they were sensing devices, I placed myself into an imaginary future where I could more easily dream of new applications.

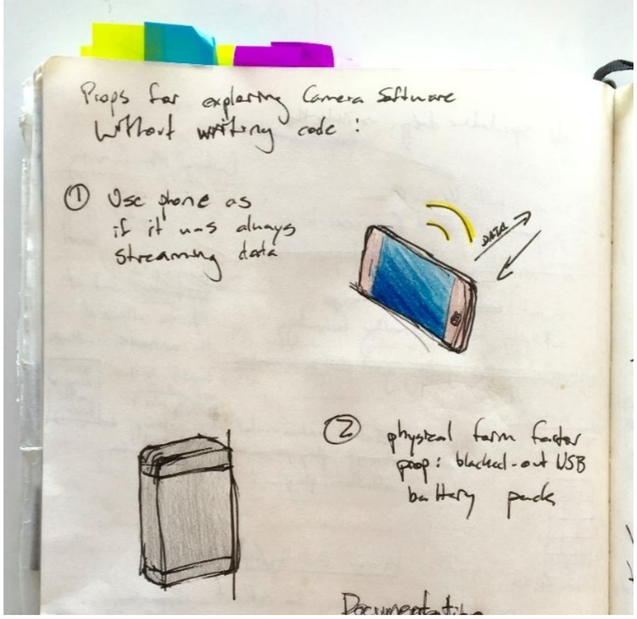


Figure 32 Sketchbook, June 17, 2016.

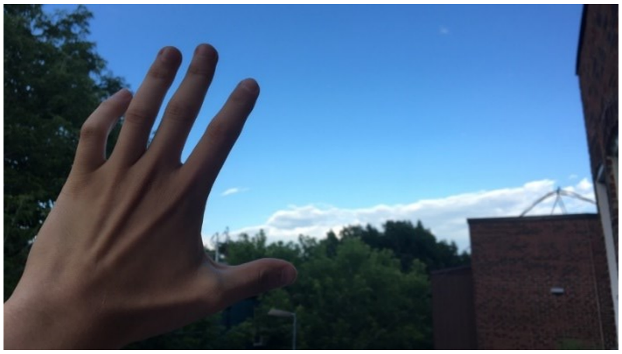
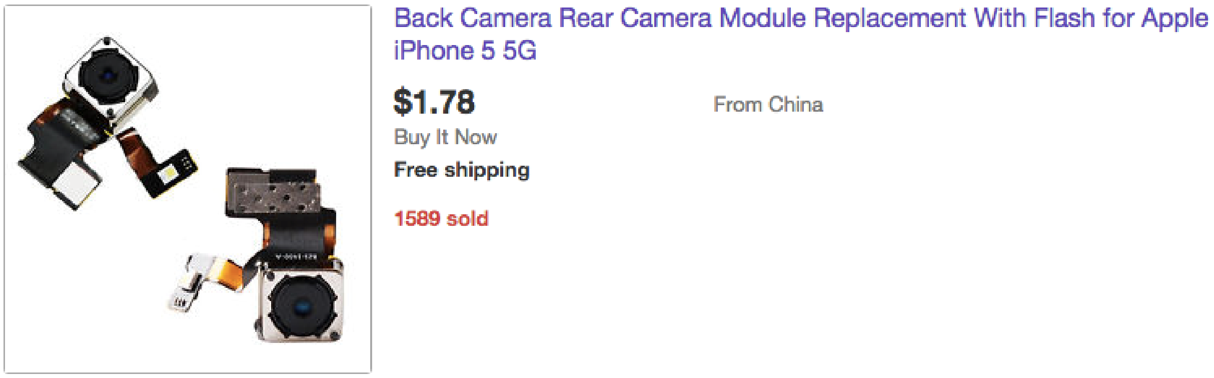


Figure 33 Still image holding the phone as if it were head- or chest-mounted, July 3, 2016.

Figure 34 Video still, June 9, 2016.

Through the gestural exercises, I recognized that the smartphone was perhaps the world's best intellectual property theft device. Although it had been in our pockets for years already, these portable ubiquitously connected cameras make it easy to capture surroundings—all that was missing was the software to turn those captures into something special. What would it mean to be able to paint with computationally enriched versions of the textures, images, and sounds I could grab with my phone? If I could capture the image of a person, thing, or any intellectual property, then surely sometime in the future, software would be able to reconstruct or synthesize variations on the original subject matter. I imagined architectural textures rendered in virtual spaces and physical material, too. Camera props helped me to ask new questions about data, software, and interfaces, and they lent me traction to begin imagining new applications.

To take the prop practice further, I began attaching non-functioning camera modules to objects all around me. First, I purchased dozens of inexpensive camera modules on AliExpress. When they arrived, I used glue and tape to attach them to everyday objects. The purpose of these experiments was to ask "What will life be like once cameras are essentially free?" If cameras can be integrated into objects for roughly no cost, how will our relationship to these objects and image capture change? I dangled cameras from a baseball cap and affixed a pair of cameras to the eyes of an Alexander Calder cat sculpture with a rubber band. With these experiments I created my first Camera Hat and Camera Cat, which recurred in my imagination and sketches for years thereafter. They exemplified two research trajectories that would emerge with greater clarity over time: non-functional cameras that are aesthetic ornaments emblematic of the supply chain in our time, and functional cameras that refactor our relationship to familiar objects. I also attached a camera to the bridge of a pair of protective glasses and created my first – very rough – camera glasses. The awkward apprehension I felt about wearing these decorative camera glasses around others lent a visceral quality to the intuition that people would not appreciate being looked at by a camera all the time.



Back Camera Rear Camera Module Replacement With Flash for Apple iPhone 5 5G

$1.78

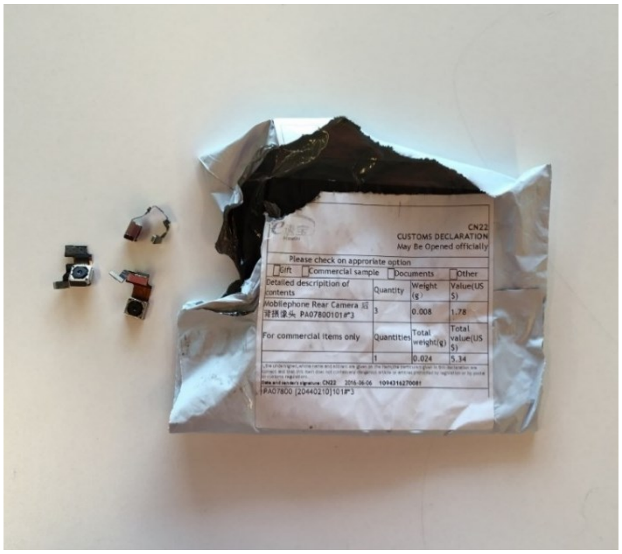
From China

Buy It Now

Free shipping

1589 sold

Figure 35 Screenshot of AliExpress listing, June 17, 2016.



CN22
CUSTOMS DECLARATION
May Be Opened officially

Please check on approriate option

☐Gift ☐Commercial sample ☐Documents ☐Other

Detailed descripition of contents	Quantity	Weight (g)	Value(US $)
Mobilephone Rear Camera 后背摄像头 PA07800101#*3	3	0.008	1.78
For commercial items only	Quantities	Total weight(g)	Total value(US $)
	1	0.024	5.34

Date and sender's signature: CN22 2016-06-06 1094316270081

PA07800 [20440210]101#*3

Figure 36 iPhone 5 camera modules and shipping package, June 29, 2016.

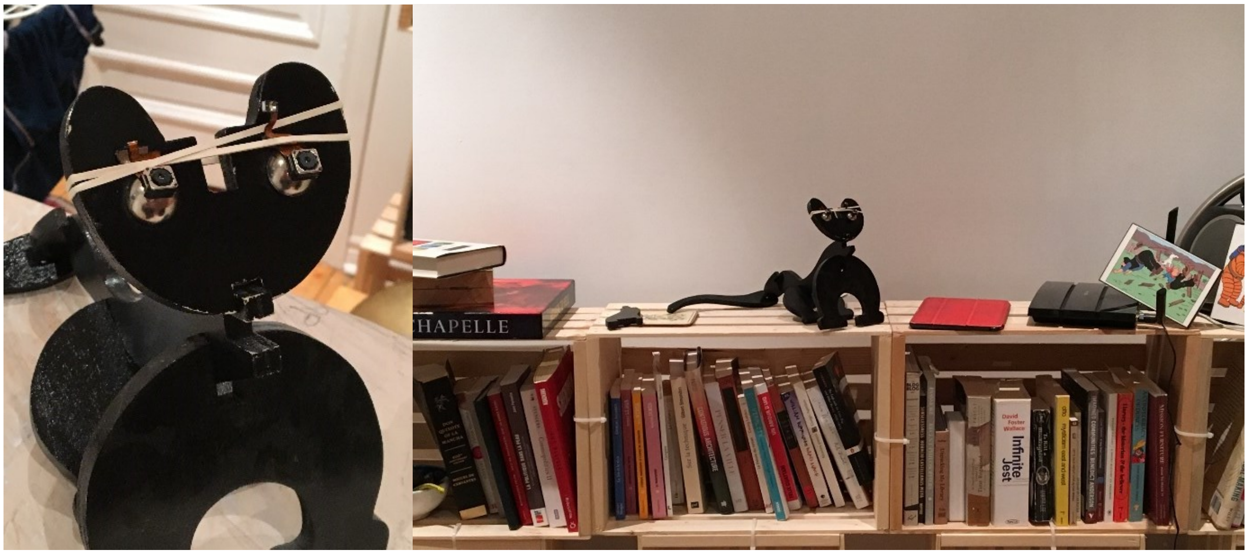


Figure 37 Camera Props, July 13, 2016.

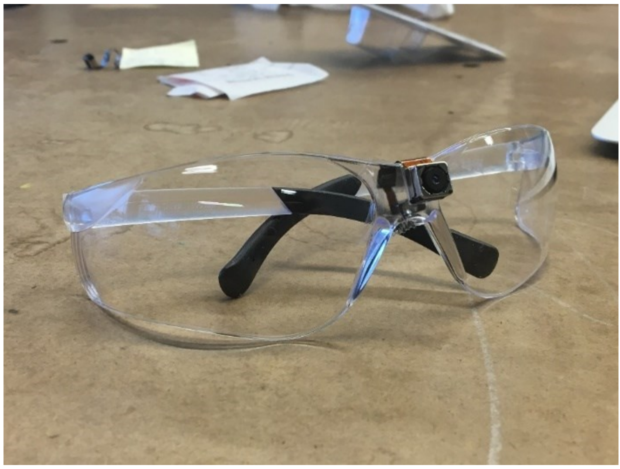


Figure 39 Camera Glasses, July 8, 2016.

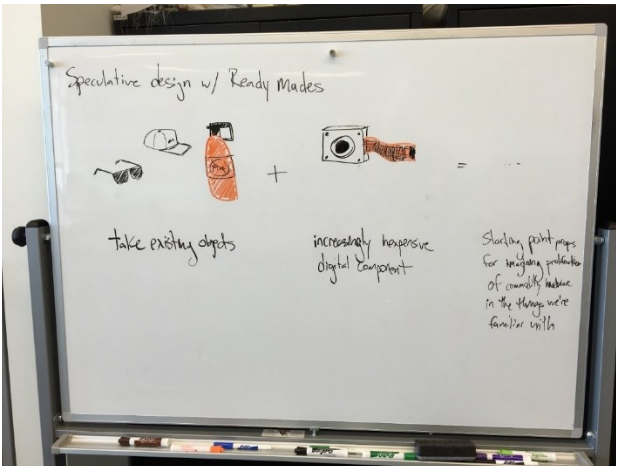

Speculative design w/ Ready Mades
+
=
take existing objects
increasingly inexpensive digital component
Starting point props for imagining proliferation of commodity hardware in the things we're familiar with

Figure 38 Camera Cat, July 7, 2016.

Through prop prototyping and imaginative play, I developed deeper ideas about how cameras and machine vision would give new life to existing objects. The fire extinguisher was one particularly interesting camera prop. I began by affixing cameras to a medium-size fire extinguisher. The surfaces of the plastic nozzle cover and trigger made it easy to attach two cameras, one facing the target (the fire) and one pointed up in the direction of the firefighter (the user). Even in this early state, the rough prototype revealed that objects could record data about their application and their user at the same time. The Camera Fire Extinguisher could capture data about its functional application, such as where it was pointed, the firefighter's aim and accuracy, and what it was pointed at when not in use to control a blaze. Meanwhile, the selfie camera could record the user's affect, by capturing data about their facial expression, posture, and the environment around them. The same ideas could be applied to any object. Could a set of cameras in a ring around a person's water bottle or coffee cup detect what emotional or auditory stimuli cause them to take a sip? The camera prototypes had unlocked a greater degree of sophistication in my speculative design thinking.

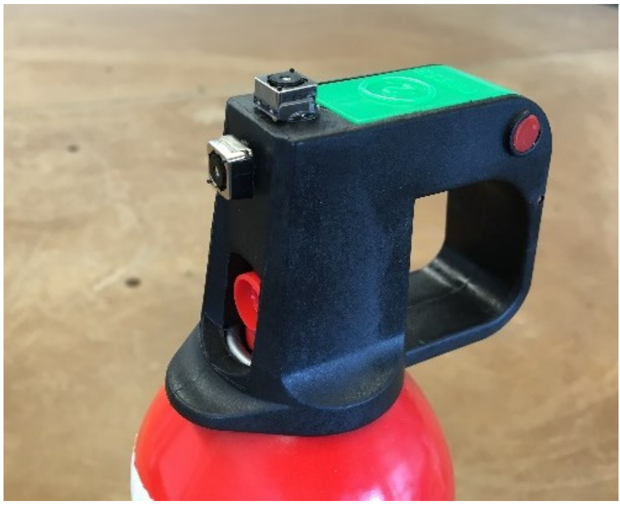


Figure 40 Camera Fire Extinguisher, July 13, 2016.

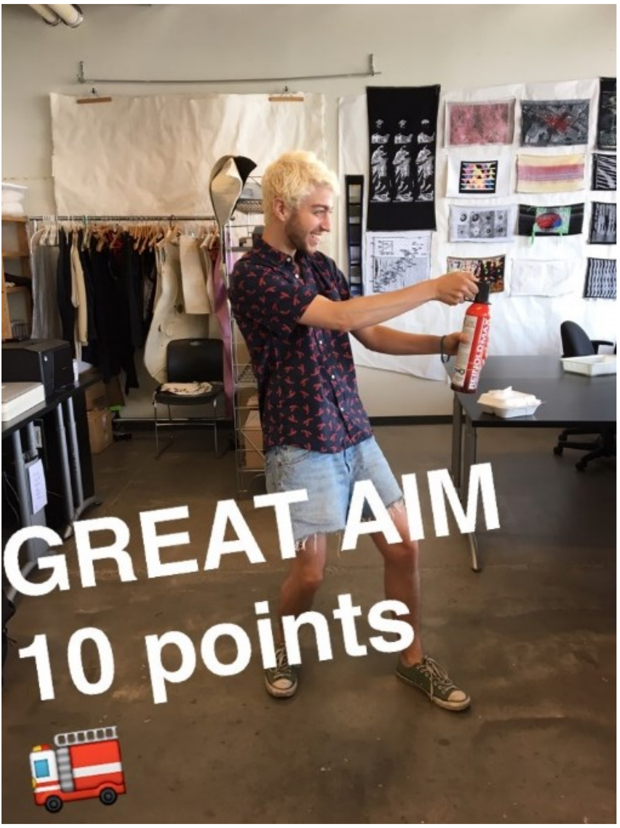
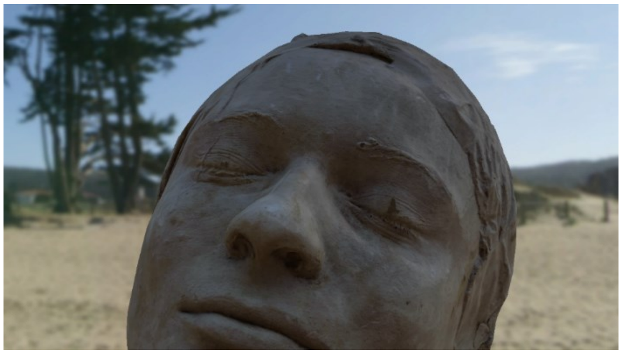


Figure 41 Speculative gamification of fire extinguishing or training software, July 13, 2016.

While I pursued dreams with prop designs, I simultaneously immersed myself in testing new camera software applications. I played with 3D photogrammetry software, which reconstructs a 3D scene from still images, video, or LIDAR data. I researched how Tesla presents drivers data from their suite of cameras, which had only rudimentary self-driving abilities at the time. I also used and misused a free plant identification iPhone app to see how popular machine vision software degrades in adverse situations, and to explore opportunities for memetic intervention.



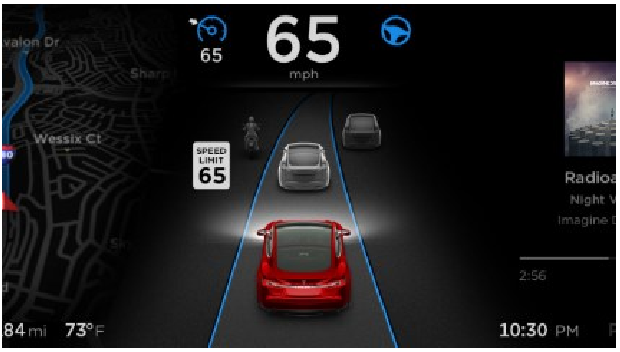
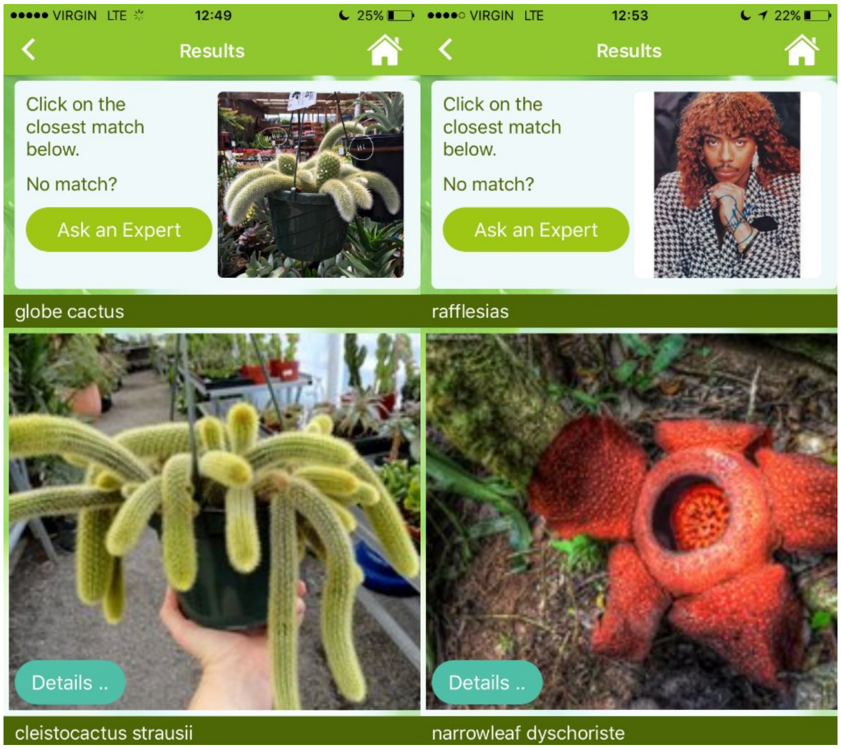


Figure 43 Screenshot of photogrammetry software displaying a scanned model, July 2, 2016.

Figure 42 Screenshot of Tesla main console software, June 30, 2016.



Click on the closest match below.

No match?

Ask an Expert

globe cactus

Details ..

cleistocactus strausii

Click on the closest match below.

No match?

Ask an Expert

rafflesias

Details ..

narrowleaf dyschoriste

Figure 44 Screenshots of plant identification software output given two inputs: a meme of a cactus with faces drawn on it and a picture of Rick James, July 4, 2016.

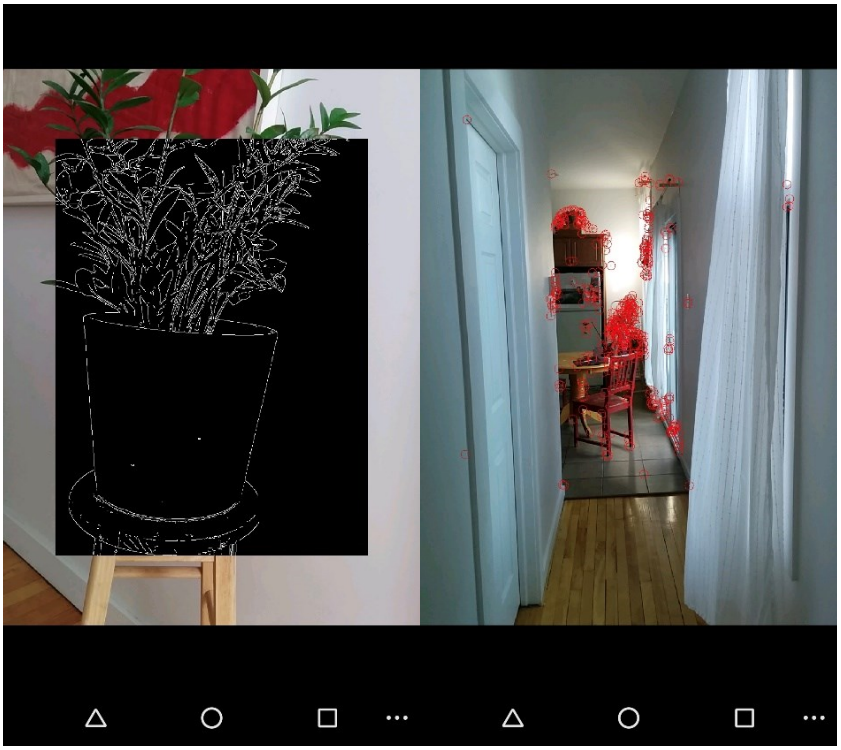
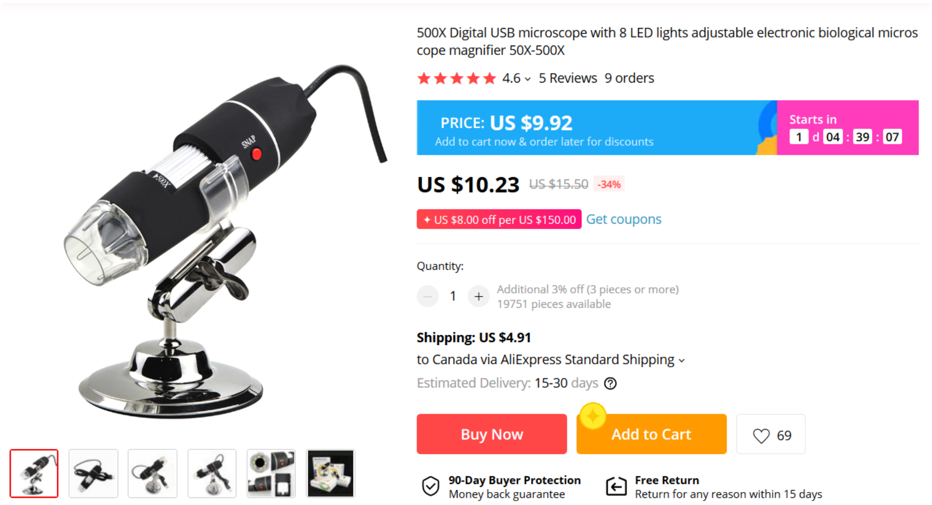


Figure 45 OpenCV edge detection on Android, August 9, 2016.

While searching for interesting camera art supplies on AliExpress, I came across a $14 50x optical zoom USB microscope. Despite its modest zoom factor, I was amazed that I could purchase a digital microscope so inexpensively. I bought the microscope and began investigating the barely visible world. I looked at plants, LCD displays, anodized aluminum, my bleached hair, skin, tattoos, bugs, and fabrics. The microscope's camera was less than two megapixels, but its portable form factor and magnifying optics made it completely unlike the more expensive and higher megapixel-count cameras in my phone. I took most of the following images with the microscope attached to my computer, but I later discovered adapters that enabled me to use it in conjunction with an Android phone. By playing with the device, I learned and internalized that microscopes are network cameras, too. Zoom factor is one more variable available to the network camera maker.



500X Digital USB microscope with 8 LED lights adjustable electronic biological microscope magnifier 50X-500X

★★★★★ 4.6 5 Reviews 9 orders

PRICE: US $9.92
Add to cart now & order later for discounts

Starts in 1 d 04 : 39 : 07

US $10.23 ~~US $15.50~~ -34%

US $8.00 off per US $150.00 Get coupons

Quantity:
1 Additional 3% off (3 pieces or more)
19751 pieces available

Shipping: US $4.91
to Canada via AliExpress Standard Shipping
Estimated Delivery: 15-30 days

90-Day Buyer Protection
Money back guarantee

Free Return
Return for any reason within 15 days

Figure 46Screenshot of an AliExpress listing for the same microscope from another provider, August 22, 2020.

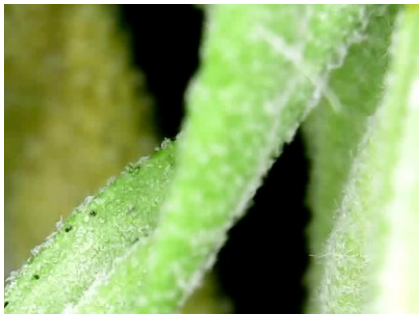


Figure 47 House plant leaf 200x digital zoom, August 5, 2016.

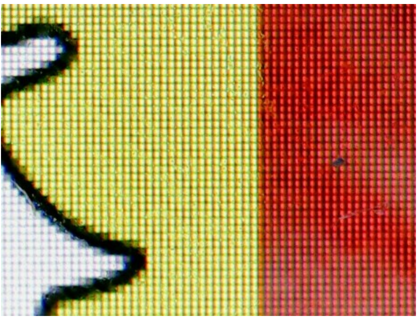


Figure 48 iPhone display 50x optical zoom, August 5, 2016.

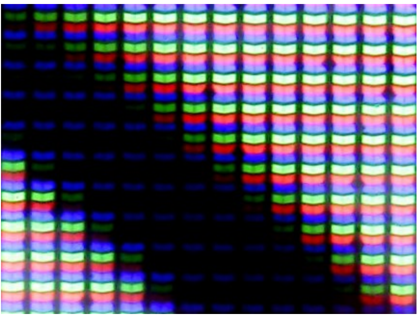


Figure 49 iPhone display 200x digital zoom, August 5, 2016.

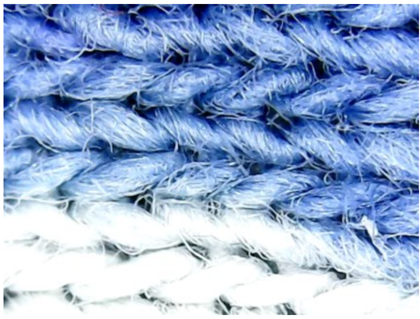


Figure 52 Tie-dye t-shirt 200x digital zoom, August 5, 2016.

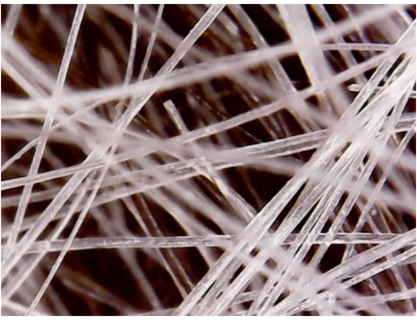


Figure 50 Bleached hair 50x optical zoom, August 17, 2016.

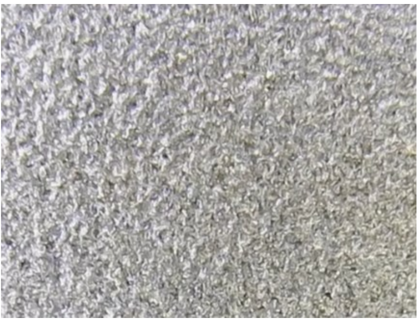


Figure 51 Anodized aluminium iPhone 200x digital zoom, August 5, 2016.

My experiences with the small camera props and ideas about texture capture pushed my thinking in the direction of Chicklet Cameras, or very small battery powered cameras. During my research, I discovered computer vision journal articles about 4D reconstruction from multiple camera angles [305], [306]. This early technology research indicated that soon, one might be able to recreate a living 3D scene from a handful of cameras in a room or outdoors. Not only could these cameras be used to create a 3D model of the space, they could also reconstruct the space and its contents over time—thus the fourth dimension. Although the technology was ultimately out of reach, I repeatedly imagined tossing a handful of Chicklet-sized cameras into a space to capture its contents in 4D. I presumed that such 4D cameras could do their reconstruction in the cloud, or on a beefier device on the local network, however I wondered if the camera hardware itself might need a means for pointing down at the scene, not only up from the floor.

I sketched variations on the Chicklet Camera concept with different fasteners. I imagined one version with elasticized rubber bands embedded in the backside, which was inspired by the UE Roll Bluetooth speaker design. Another variation had a hair clip style snap clip on its back. Others featured suction cups, Velcro backings, magnets, ball joints, and kickstands, like a bike. I ultimately decided that the chicklet form factor was too ambitious for a first prototype, but I found the sketching process inspiring and productive, especially as I began to imagine new form factors that would make sense when paired with forthcoming machine vision software capabilities.

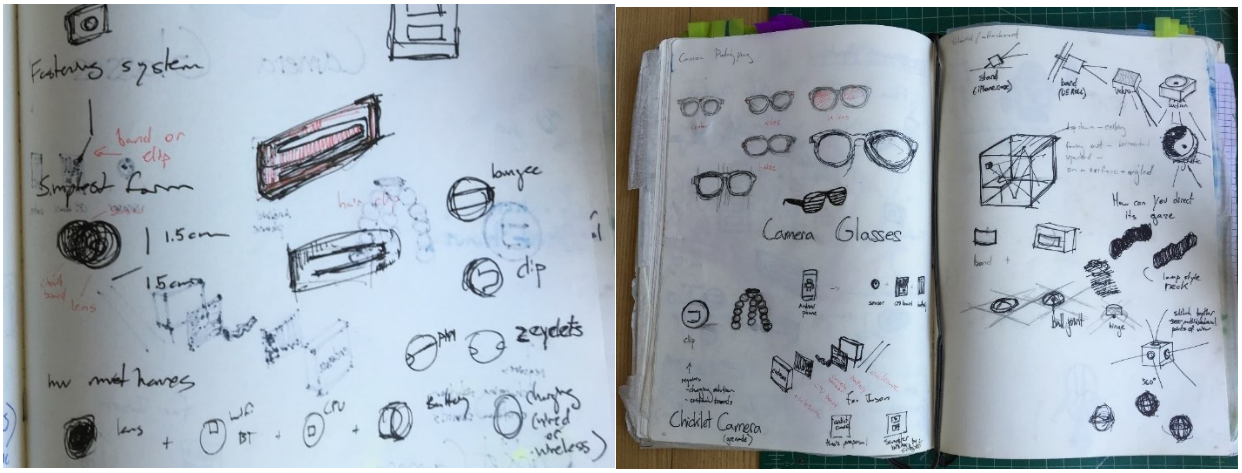
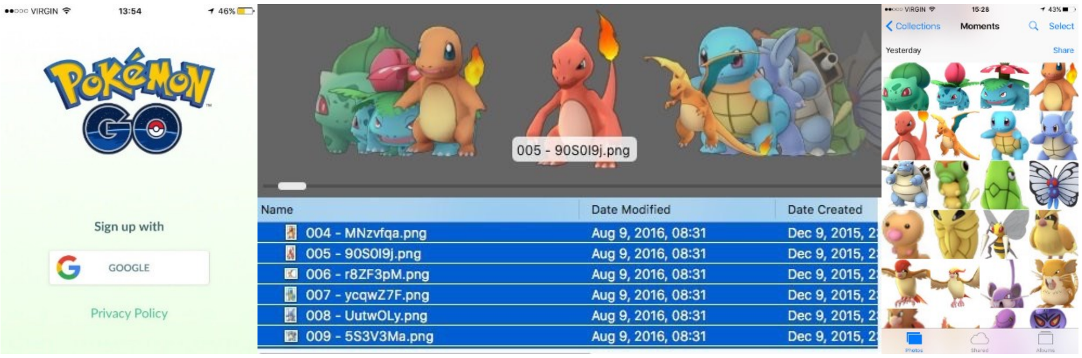


Figure 53 Sketchbook fasteners, July 16, 2016.

While I sketched my first ideas during the summer of 2016, Niantic Labs launched Pokémon GO, the hugely popular augmented reality smartphone game. A friend became obsessed with the game late that summer and played constantly while taking breaks from other tasks. I found the game disrespectful of my time; its attention optimizing design drove players to grind endlessly for opportunities to “catch ‘em all.” Instead of playing the game, I decided to cynically beat it by other means. I downloaded the game’s Android APK onto my computer, and delved into its guts.[89] Since Pokémon GO players were essentially devoting huge portions of their time to unlocking 3D models of Pokémon, I decided to extract the whole set of available characters as image files on my computer. I imported the images to my phone's Camera Roll. In some way, I was able to catch every single Pokémon without ever playing the game. Of course, the fun of the game comes from the tension and anticipation of slowly earning each fantasy animal. While catching a Pokémon technically means changing a single value in a cloud database somewhere, the emotional involvement in playing requires that one not cheat the system as profoundly as I had. Again, I made progress understanding the value of imagery, data, and interfaces through a practical exploit.



Name	Date Modified	Date Created
004 - MNzvfqa.png	Aug 9, 2016, 08:31	Dec 9, 2015, 2
005 - 90S0I9j.png	Aug 9, 2016, 08:31	Dec 9, 2015, 2
006 - r8ZF3pM.png	Aug 9, 2016, 08:31	Dec 9, 2015, 2
007 - ycqwZ7F.png	Aug 9, 2016, 08:31	Dec 9, 2015, 2
008 - UutwOLy.png	Aug 9, 2016, 08:31	Dec 9, 2015, 2
009 - 5S3V3Ma.png	Aug 9, 2016, 08:31	Dec 9, 2015, 2

Figure 54 Screenshots of Pokemon GO, character assets exfiltrated from the Android APK, and imported to my iPhone Camera Roll, August 9, 2016.

Machine Learning Against Humanity

In the summer of 2016 I traveled to Vienna, Austria to research how machine learning would affect arts and design practices. I attended the Nucl.ai machine learning arts conference and interviewed a few machine learning practitioners about their presentations and craft. The research project was funded in part by the Milieux's Textiles and Materiality Independent Project Grant.

Although machine learning was already an old discipline, the latest crop of deep learning techniques, which drew upon larger data sets and more powerful graphics hardware, had begun to show promising results. Alex J. Champandard's Neural Doodle project, for instance, demonstrated that simple drawings could be transformed into Impressionist paintings with style transfer techniques [307]. In Neural Doodle, the user paints a rough scene with a handful of colours in an interface reminiscent of Microsoft Paint. The

---

[89] APK is an Android Application Package, or Android app executable. APKs are equivalent to EXE on Windows.

ML software then transforms the rudimentary work into an image in the style of Renoir, or another painter.

Upon my return to Montreal, I found myself wondering how best to dream about new machine learning applications while the tools remained difficult to appropriate. Anyone paying attention could see that these technologies would reconfigure our ideas about computation and media. How could I get an imaginative handle on a technology so difficult to grapple with directly? To begin to dream in machine learning, I made myself a thinking instrument.

Machine Learning Against Humanity (MLAH) is a toy designed to help players imagine new applications for machine learning. MLAH is composed of a set of laser cut tiles. Each tile is emblazoned with a word or phrase. There are verb tiles, which represent high level machine learning capabilities. These are words and phrases such as recognize, synthesize, transfer style, and automate. There are media tiles, with names like text, speech, gesture, and face. There are hardware tiles, such as camera, vehicle, projector, and earpiece. There are also adjectives, such as immersive, ambient, and the rather wordy phrase “inexpensive enough to waste.” I later created a booster pack of tiles for materials and organic life, such as glass, ceramic, tree, and ecosystem.

MLAH is more of a toy than a game. There are no rules, points, or win states in playing with the tile set. MLAH can be played with alone or in a group. In experiments playing with friends, I have found that it is fruitful to place around 17 tiles on a flat surface like a table or the floor. Players can do as they please: match tiles into funny or interesting chains, create two dimensional figures (like crosses or abutted words in scrabble), or stack them together according to some taxonomy. Although I created the tiles according to a few categories, there are no differentiating marks on them, and so thinking flows fluidly without constraint.

The purpose of MLAH was to spur my own thinking and to perhaps help liberate others to imagine new and unexpected collisions of machine learning with the rest of life. Rather than resort immediately to the sometimes dry and alienating mathematical and computational abstractions, MLAH encourages players to dream of applications first. Anecdotally, I found the most exciting moments playing with the set were when one person added a tile to another person’s idea chain, thus creating a jolt of surprise.

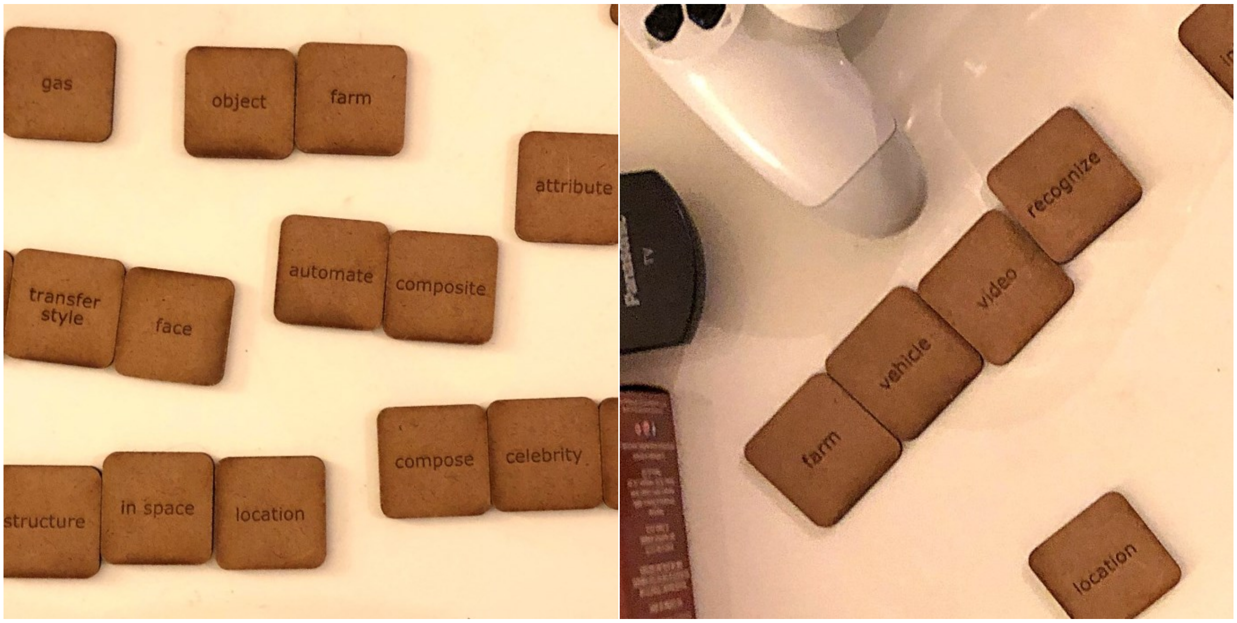


Figure 55 Machine Learning Against Humanity in use, June 15, 2020.

Camera Glasses

As Fall turned to Winter, I narrowed my focus to building a pair of camera glasses. I remained convinced that it was important for creators outside of the major technology companies to address the impending XR boom. Augmented reality glasses were still a major scientific research project at large companies, so I decided to focus my energy on camera glasses, instead, as they were a more approachable piece of the larger XR problem space.

I surveyed the market for materials to prototype the glasses, and was surprised to discover that there were no great options for prototyping camera hardware. I did not feel personally equipped to address the problem myself, but it seemed to me then, as it does now, that there ought to be a great camera prototyping platform, like Arduino is to physical computing projects. Nevertheless, I chose to focus on the camera glasses and use the best available hardware solutions to begin thinking up new product, industrial, and software designs.

Once I had chosen to make camera glasses, I was able to decide on the specific hardware components I would use to build the first prototypes. I considered using low cost second-hand Android phones as a hardware stack, but I ultimately opted against this option because Android was a less approachable ecosystem to learn for the type of projects I intended to pursue, and perhaps even more importantly, the phones' rigid PCBs meant that I would be stuck with phone form factor prototypes. In the end, I chose to use a hardware stack consisting of a Raspberry Pi, the 8 megapixel Pi Camera v2, a 500 mAh 3.7V Li-Po battery, and an AdaFruit PowerBoost 1000. The Pi and Pi Camera were the heart of the project, the battery would make it mobile, and the PowerBoost boosted the 3.7V battery to 5V and provided easy charging over Micro-USB. I used a Raspberry Pi 3 Model B+ as a software development kit, because it is a faster computer, then I transitioned to the Pi Zero W for the final production version, because of its smaller form factor. I chose to use this hardware stack because it provided a well-supported easy entry point to camera prototyping, which was important given my limited experience at the time.

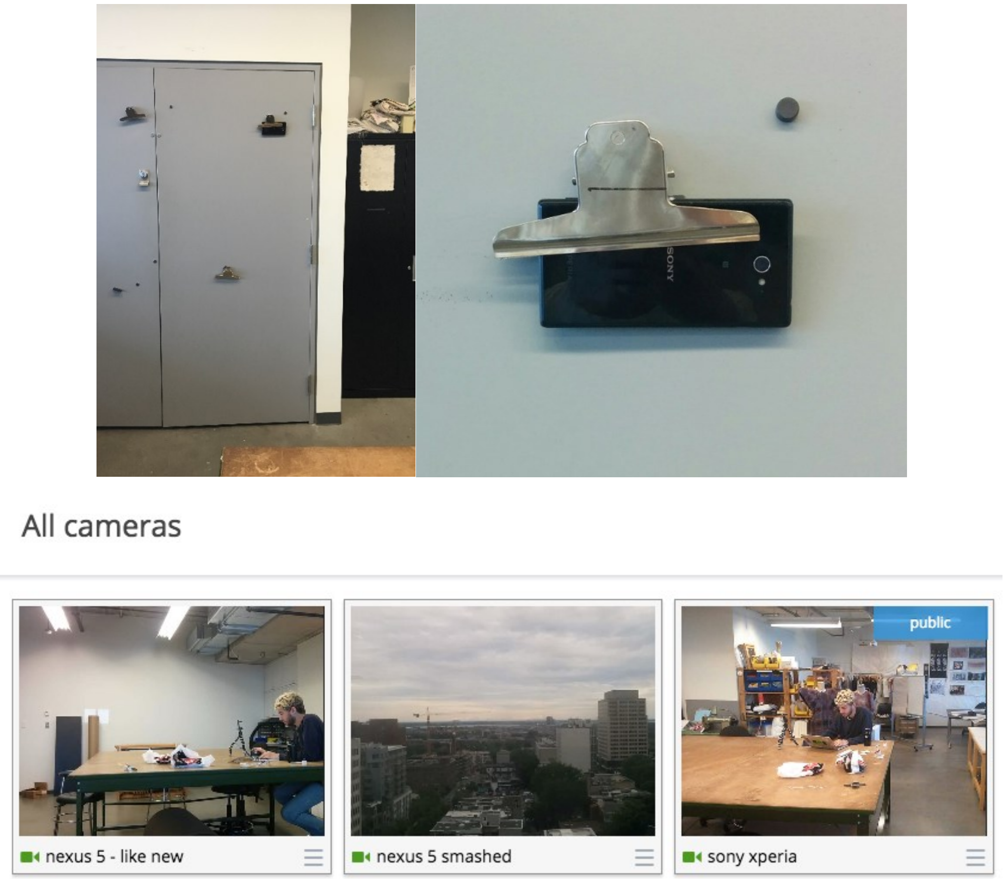


Figure 56 Testing streaming applications with second hand Android handsets, July 8, 2016.

I built my first version of the prototype very quickly. I had an inexplicable fascination with the idea of putting the camera on a pair of shutter shades first. I was interested in putting the relatively high tech components into a frivolous, party-like package, and the irony of doing so with frames called “shutter shades” was amusing, too. As soon as I had put them together, it became obvious that they obscured the wearer’s vision too much. I quickly iterated to more understated metal frames, to which I affixed the camera with tape.

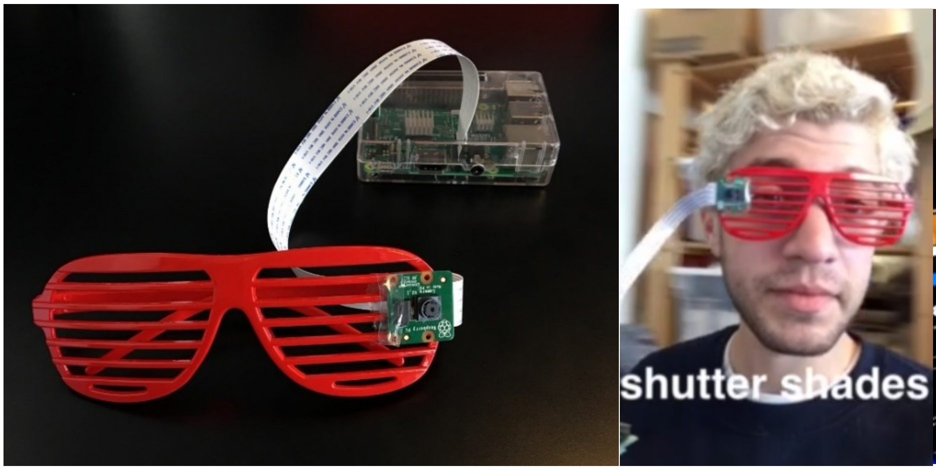


Figure 57 Shutter Shades, October 3, 2016.

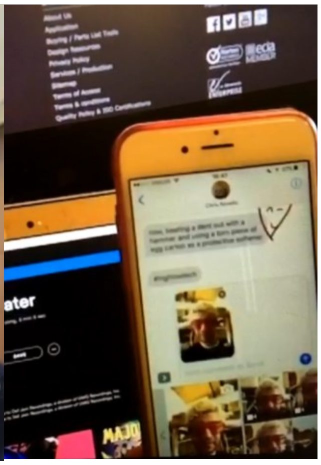


Figure 58 Image of iPhone selfie camera feed in iMessage, captured from the Shutter Shades prototype, October 3, 2016.

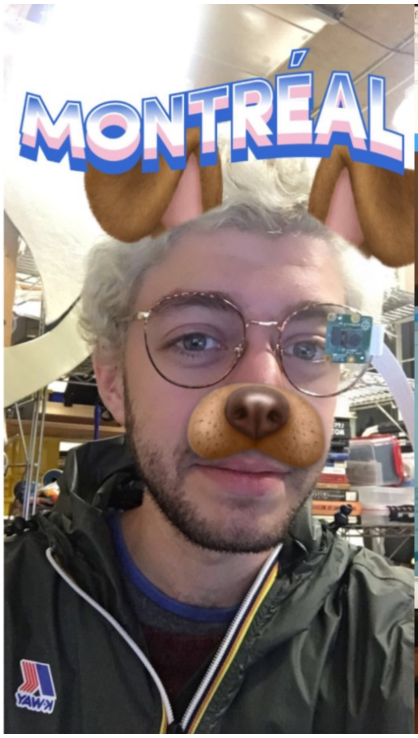
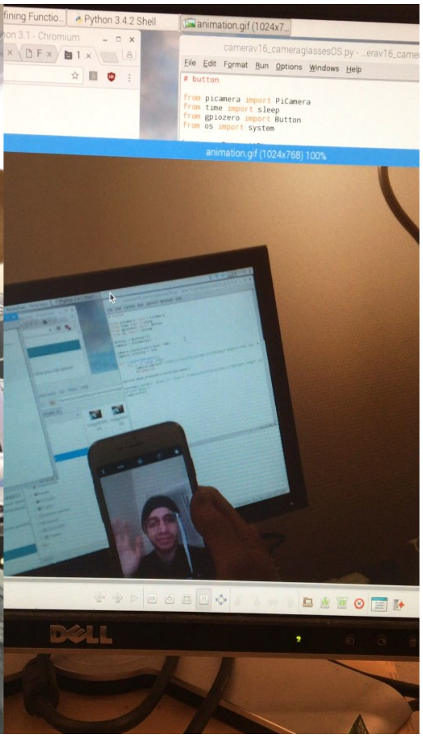


Figure 60 Camera glasses with metal frames, October 6, 2016.


Python 3.4.2 Shell
animation.gif (1024x7...
camerav16_cameraglassesOS.py - ...erav16_camer
File Edit Format Run Options Windows Help
animation.gif (1024x768) 100%
DELL

```
# button

from picamera import PiCamera
from time import sleep
from gpiozero import Button
from os import system
```

Figure 59 Image of camera glasses video playing back on Raspbian on the Raspberry Pi 3B+, November 6, 2016.

Camera prototyping projects exhibit the unusual property of documenting themselves. Naturally, the first images I captured with the Shutter Shades and subsequent camera glasses were pictures and videos of the glasses output, itself. As soon as a prototype was working, I would take a picture of the screen on which I had programmed the device. I also made the obvious observation that, when one walks around the world with camera glasses, every mirror becomes a camera. I got into the habit of taking my working prototypes on a walk to the large mirror in the bathroom down the hall, or pulling out my phone and switching it into selfie mode to capture a satisfying picture or video to document the progress. It felt new and a little bit magical, because it both augmented the experience of using my phone's selfie camera, and it exceeded the phone's capabilities in some respects.

As soon as I got a working version of the basic camera glasses, I began showing it to friends. In its earliest iteration, I rigged the glasses up to simply broadcast the live video stream to a screen. At a small house party, new initiates had fun plugging the live camera output into a projector, then playing with video feedback. They pointed it at their phone cameras, and took pictures of the results. Others spontaneously suggested adding pink eyelashes to the glasses—a more adventurous aesthetic compared to what I had previously considered. Encounters with the public showed that the glasses could engage people's interest when the design leaned into fun and playfulness, both aesthetically and functionally.

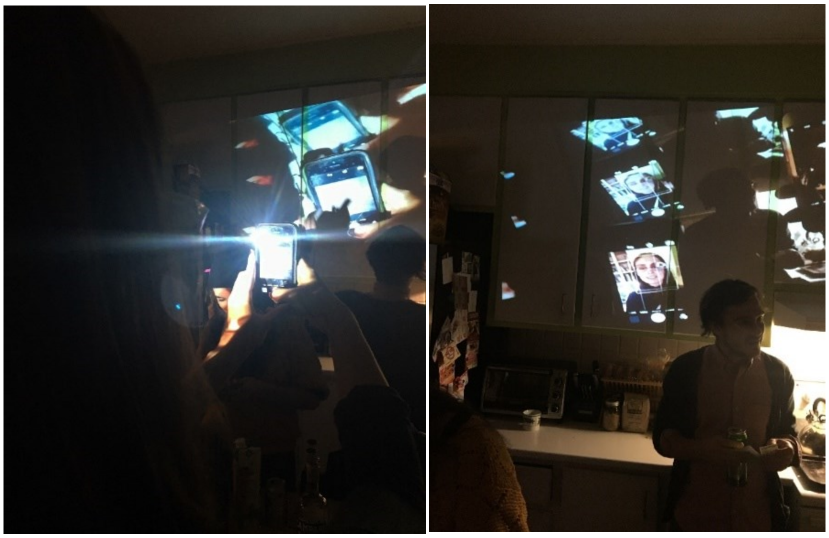


Figure 61 Still images of a person wearing the camera glasses, looking at their phone, with the projected live image of the camera glasses feeding back in the background, October 27, 2016.

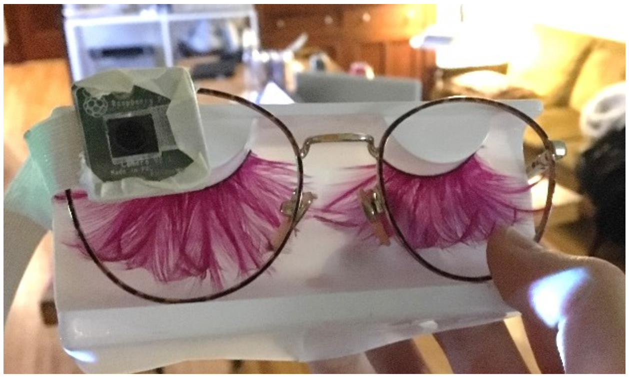


Figure 62 Early camera glasses prototype with pink eyelashes, October 27, 2020.

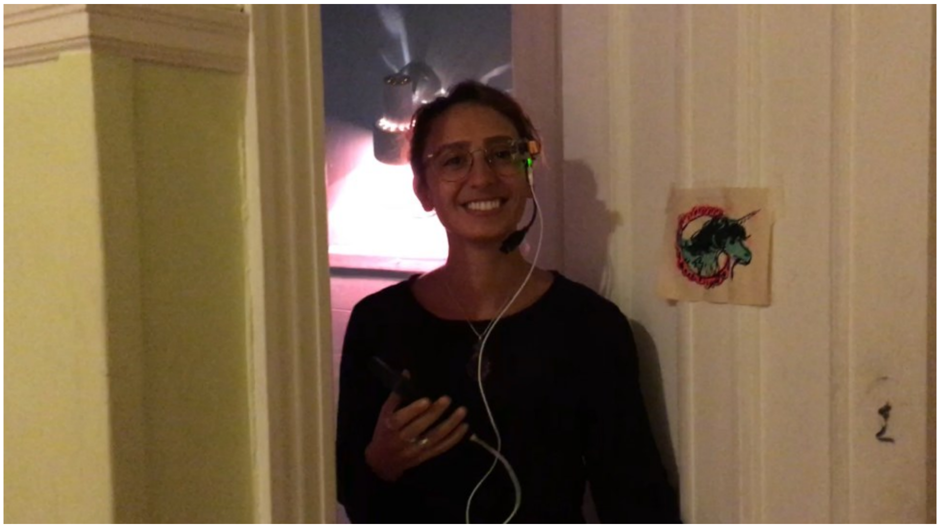


Figure 63 Early battery powered camera glasses prototype test with Leila, January 23, 2017.

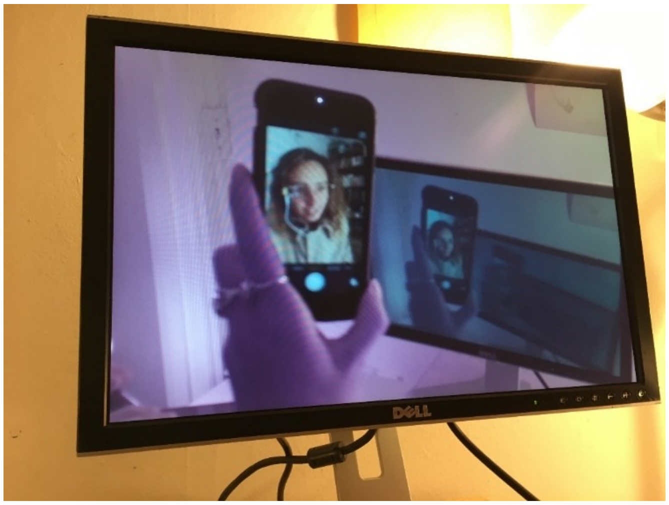


Figure 64 Early battery powered camera glasses prototype test with Leila, January 23, 2017.

In November 2016, Snap, the parent company of Snapchat, released the first version of its Spectacles camera glasses to the market. Despite their winter launch, Spectacles exuded a fun summer feeling. Instead of making high end wear-all-day glasses, Snap had opted to create sunglasses only. They sported a single button for taking pictures and videos, and an outward-facing ring of LEDs in the frames alerted interlocutors when the wearer captured images.

Although they failed to become a popular product, Spectacles marked a significant shift in hardware product marketing. Instead of presenting their glasses as an Apple-like platonic ideal of face worn computation, as most competitors had done to that point, Snap wisely opted to pitch Spectacles as a nearly disposable computer device fit for chill trips to the beach. Spectacles product marketing embraced the parent brand's youthful sense of frivolity and play. The product's design took into account the smartphone ecosystem's deflationary effect on the price of electronics components such as cameras and systems on a chip. Spectacles launched with a brilliant vending machine buzz marketing campaign that made the glasses a scarce object of desire.

Despite these prescient product marketing decisions, Snap's glasses had several dire software drawbacks. These charming wearable computers worked exclusively with the Snapchat app. There was no way, whatsoever, to execute anything but the predefined functions built into the glasses, and no way to interface them with software other than Snapchat. This meant that wearers were limited to taking pictures and ten second clips, and that they would be obliged to open their phone before syncing them and sharing them at all. These were not just camera glasses, they were Snapchat glasses, and that meant opening Snapchat all of the time.

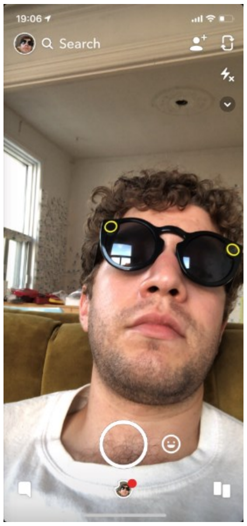


Figure 65 Screenshot of Snapchat showing me wearing a pair of Snap Spectacles, July 31, 2019.

Snap Spectacles emboldened me in my vision of building a reference design for a more-or-less open source pair of camera glasses. I imagined putting together a set of 3D models for printing the glasses frames, a bill of materials with the relevant Raspberry Pi parts and accessories, and an instructional guide for assembling and programming the device. Although I never completed this aspect of the project, I continue to believe that it would be relevant, interesting, and possibly useful to anyone wanting to film from their own perspective. There are a handful of similar builds on Instructables and YouTube, but none is sufficiently fleshed out or well designed to make a decent pair of glasses that are both easy to fabricate and comfortable to use. A thoroughly thought through open design could make DIY camera glasses a fun weekend project by simplifying the many decisions and software integration challenges into a prepackaged guide. If enough people found the design compelling and built it themselves, then the DIY hardware community would be better positioned to create a layer of free and open source applications atop said camera glasses. A simple reference design would transform readily available hardware components into a promising hardware platform for software application prototyping. It would be very interesting to see what changes and updates people in the Raspberry wearables community would invent, once a starting point is laid out clearly in an instruction manual and video guide.

Working on the camera glasses taught me that other people may need to see specific use-case examples to understand the value of a new technology product form. When I mentioned that I was working on camera glasses to non-technologists, I was sometimes met with smirks and questions asking if I had untoward intentions for the project. I was amused and a little uncomfortable with the association between camera glasses and surreptitious filming. One of the few clear use-cases that emerged through these conversations was that a pair of camera glasses would allow artists and makers to document their work while their hands were full. This explanation clicked with creators.

I began prototyping the glasses frames at the same time that I began interning at the Fab Lab du PEC. Raphaël Demers, the lab manager, introduced me to the laser cutter, and showed me how to use its cutting and engraving functions in tandem to create a basic acrylic enclosure for the Raspberry Pi Camera. To draw my first pair of glasses, I found an open source add-on for Blender that generates frames based on a handful of dimensional parameters [308].Unfortunately, the add-on generated frames did not include the temple arms. I was not yet sufficiently skilled at 3D modeling to modify the frames to add arms or a slot for holding the Pi Camera. Nevertheless, I decided to proceed with printing the half-finished design in order to move from pure ideas to something I could hold and evaluate with my senses.

I used the add-on to generate a model, then printed it on a 3D printer. Raphaël also showed me how to use the CNC mill to mill a version of the glasses frames out of a block of hardwood and a thick chunk of clear acrylic. The 3D printed version was the easiest and quickest to fabricate, but it did not look very attractive. I was very impressed by the definition of the CNC milled wooden frames. The clear acrylic frames looked interesting, but were not as cleanly cut as the wooden pair. The CNC versions also required supports to hold the frames during the milling process, which meant they would need to pass through a lengthy and potentially challenging hand finishing process.

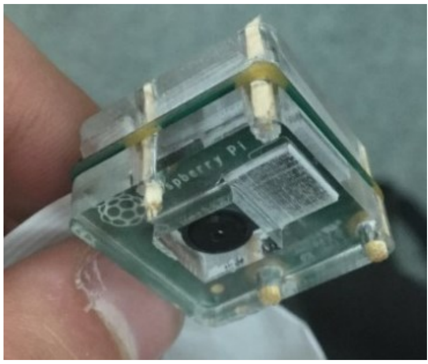


Figure 66 Laser Cut Acrylic Pi Camera enclosure, October 6, 2016.

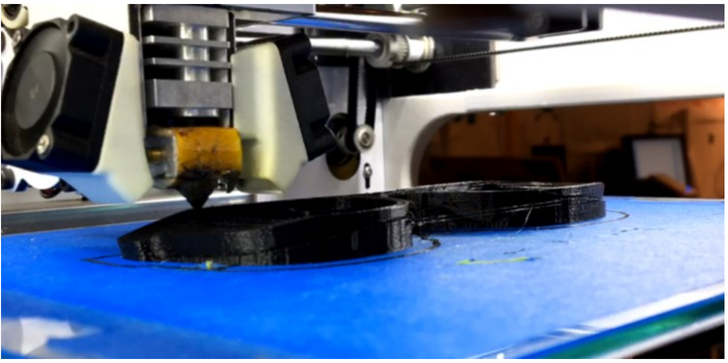


Figure 67 3D printing glasses frames prototype, October 20, 2016.

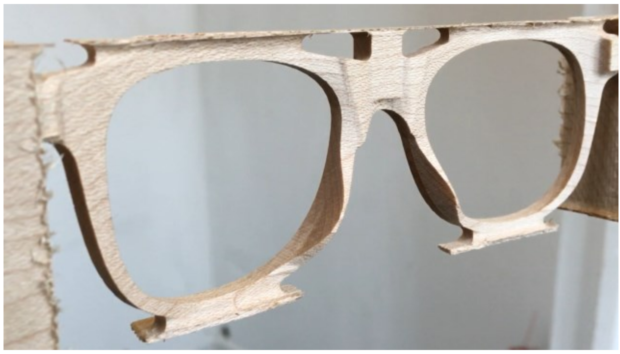


Figure 68 Wood and acrylic CNC glasses frames prototypes, October 26, 2016.

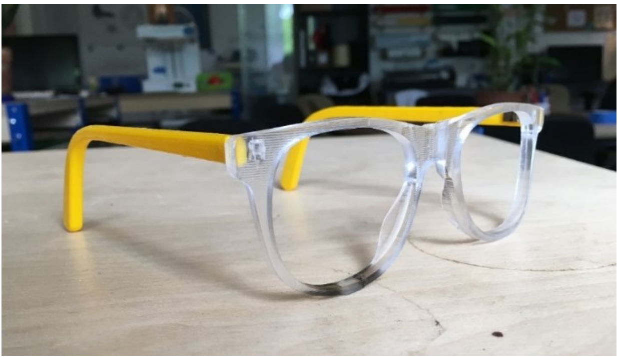


Figure 69 Acrylic CNC glasses frames prototypes with 3D printed arms added later, May 25, 2017.

I added buttons and software to allow the user to interact with the camera glasses without plugging the Raspberry Pi into a display. I created a Python camera glasses operating system, of sorts, and set up the Raspberry Pi’s Linux OS to automatically start my program on boot. I configured the computer to launch in headless mode to save power. To add the buttons, I first plugged them into a breadboard, and soon switched to soldering them to solid core wires, which I soldered to the Raspberry Pi Zero W (the smaller version of the Raspberry Pi).

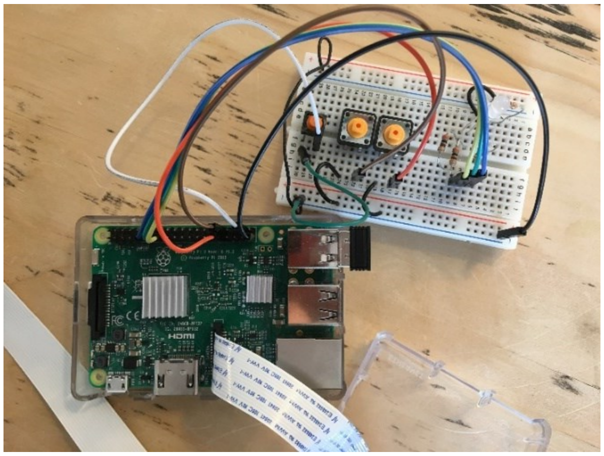


Figure 70 Breadboard prototype, March 13, 2017.

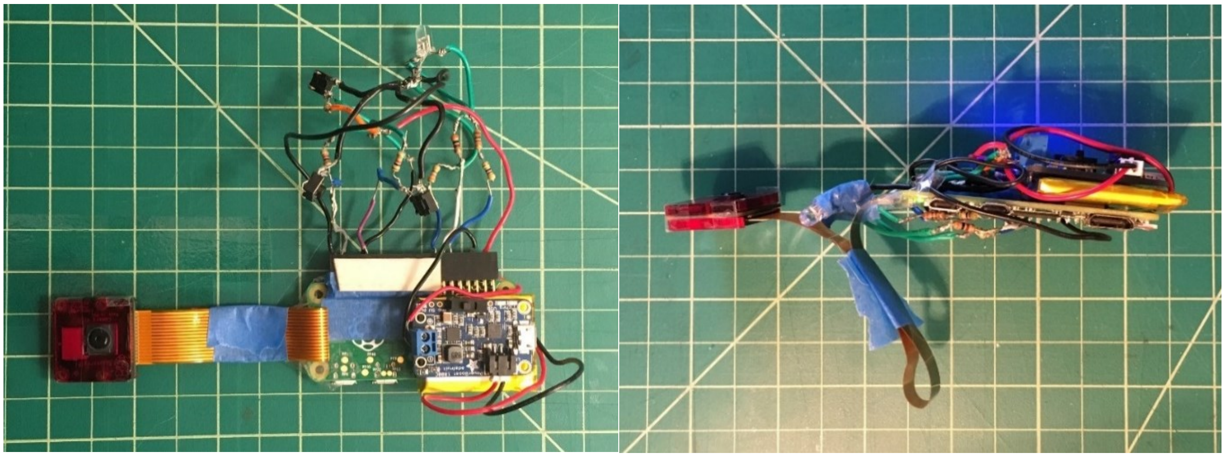


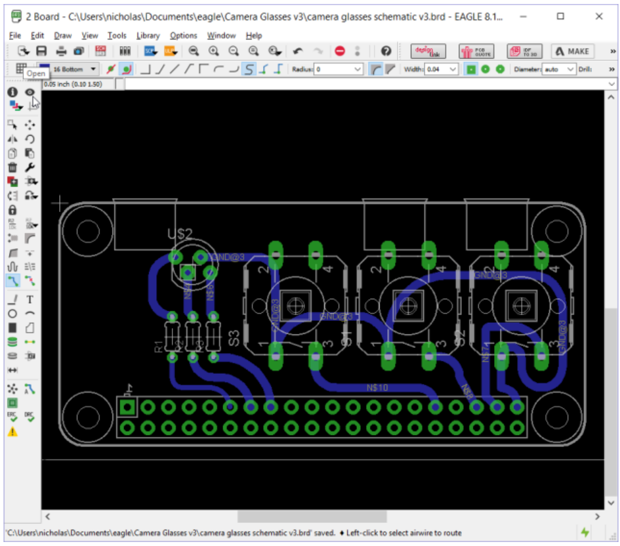
Figure 71 Camera hardware with interactive buttons, LED status indicator, Li-Po battery, PowerBoost, red acrylic camera PCB protector, March 15, 2017.

Raphaël, my colleague at the Fab Lab, suggested that I move from hand soldered wires to cutting my own circuit board, to which I could solder the buttons and additional accessories. While the solid core wire design worked, it was unwieldy and fragile. Instead, Raphaël suggested that I create an extension circuit board of my own that would fit perfectly on the backside of the Raspberry Pi Zero.[90] Raphaël showed me how to use Autodesk Eagle and Inkscape to create layouts for circuit boards. He also helped me engrave my first circuit board with the CNC mill. Although it is tempting to call them PCBs, these are not printed boards. Instead, the CNC drill head mills away a thin layer of copper that coats the top layer of a fiber glass

---

[90] This is the same approach as the many first and third party Arduino and Raspberry Pi extension boards called Hats.

board. By removing the copper, one creates breaks in the conductive layer, which become the negative space between circuit traces. I started calling the camera glasses NicoCam at this point.


2 Board - C:\Users\nicholas\Documents\eagle\Camera Glasses v3\camera glasses schematic v3.brd - EAGLE 8.1...

16 Bottom

Radius: 0

Width: 0.04

Diameter: auto

Drill:

0.05 inch (0.10 1.50)

'C:\Users\nicholas\Documents\eagle\Camera Glasses v3\camera glasses schematic v3.brd' saved. ♦ Left-click to select airwire to route

Figure 72 NicoCam circuit board design, May 4, 2017.

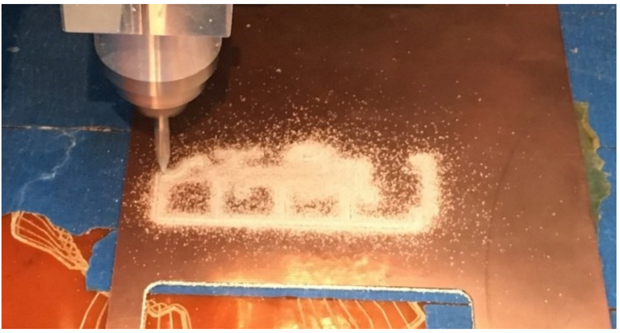


Figure 73 CNC milling the NicoCam circuit board, May 20, 2017.

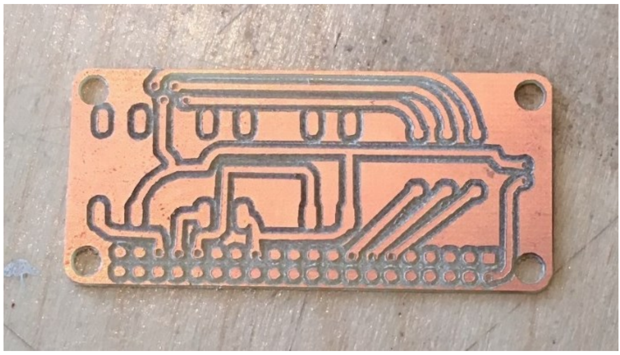


Figure 74 NicoCam v1 circuit board, May 6, 2017.

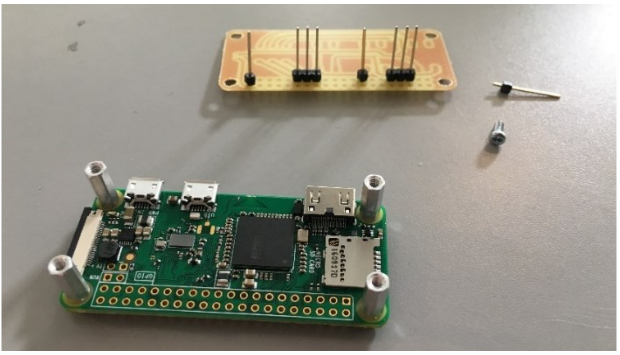


Figure 75 NicoCam v1 circuit board and Raspberry Pi Zero W with spacers attached, May 7, 2017.

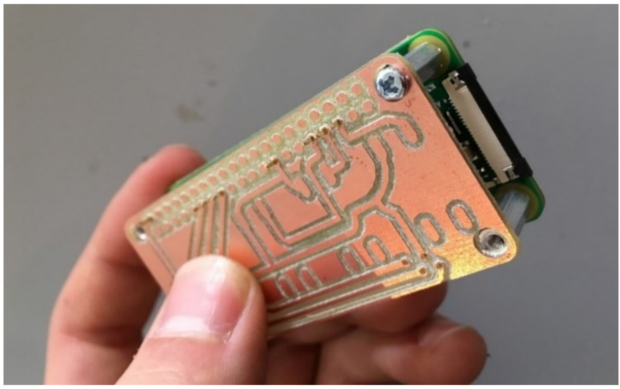


Figure 76 NicoCam v1 circuit board attached with spacers to Raspberry Pi Zero W, May 7, 2017.

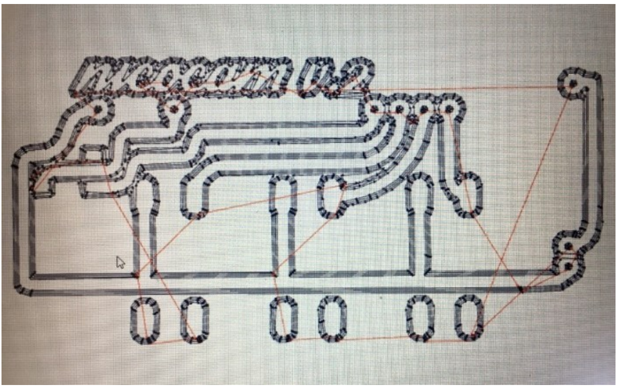


Figure 77 CNC cutting path for NicoCam v2 circuit board, May 20, 2017.

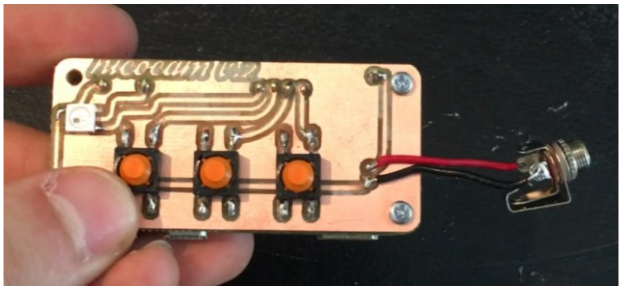


Figure 78 NicoCam v2 circuit board with SMT LED and 3mm audio jack as power source, May 20, 2017.

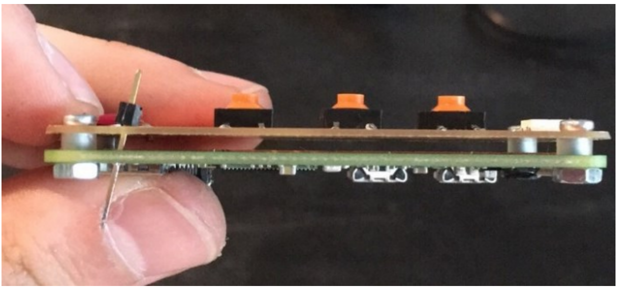


Figure 79 NicoCam v2 circuit board, assembled, May 19, 2017.

With Raphaël's help, I also devised of a system for splitting the computer and the power source between the two sides of the glasses. I intended to put the Pi and my custom board on the left temple arm of the frames and the battery and PowerBoost the other side. While Snap and other professionally designed high end glasses connected the two sides of their glasses with wires through the frames' nose bridge, for simplicity sake, I decided to attach the two sides of my glasses with a wire located behind the neck. The sunglasses neck strap was partly inspired by my father, who happily wears his glasses with the neck strap, so that they are never misplaced. Although they appear uncool, I thought that recontextualizing the neck strap in a pair of camera glasses was irreverent and even aesthetically interesting solution to the power problem. Raphaël suggested using a simple 3mm audio style cable to connect the Pi to the PowerBoost (see jack above).

Next, I began work on the glasses frame arms, which would enclose both the computer and battery. I decided to model my enclosure with Fusion360. I created many versions of the arm design, which I differentiated by PLA plastic colour. In addition to using Fusion360 for the first time, this development process taught me to use the 3D printer like a render output device. I could visualize the design quickly on the screen, or I could 3D print it and hold the physical object in my hand one to three hours later.

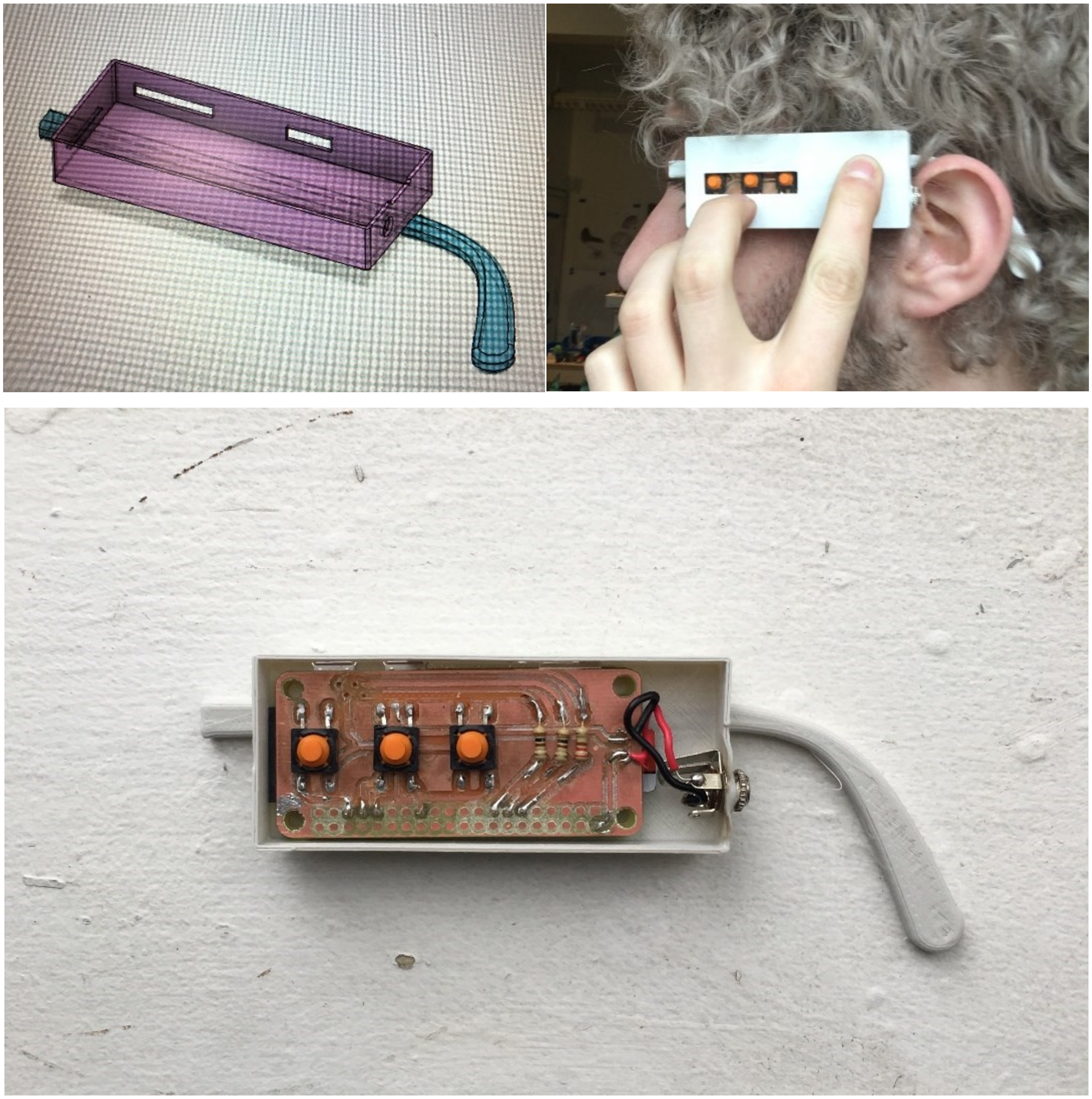


Figure 80 NicoCam frames left arm v1, May 14, 2017.

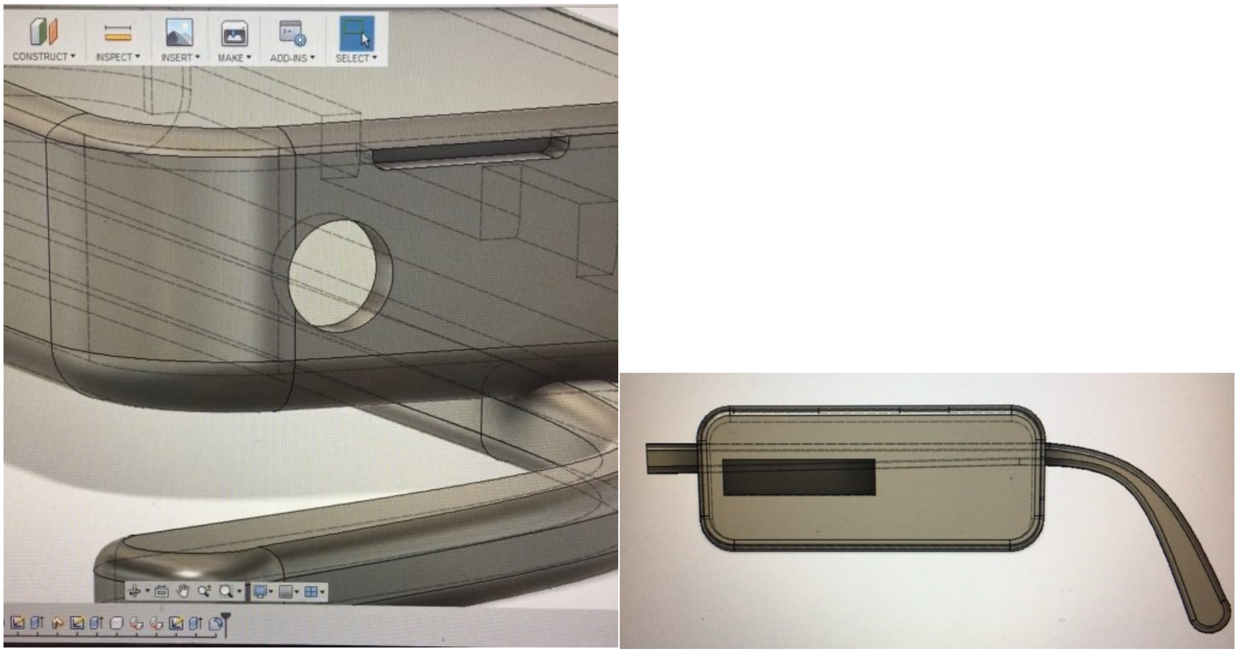


Figure 81 Fusion360 NicoCam frames 3D model, May 16, 2017.

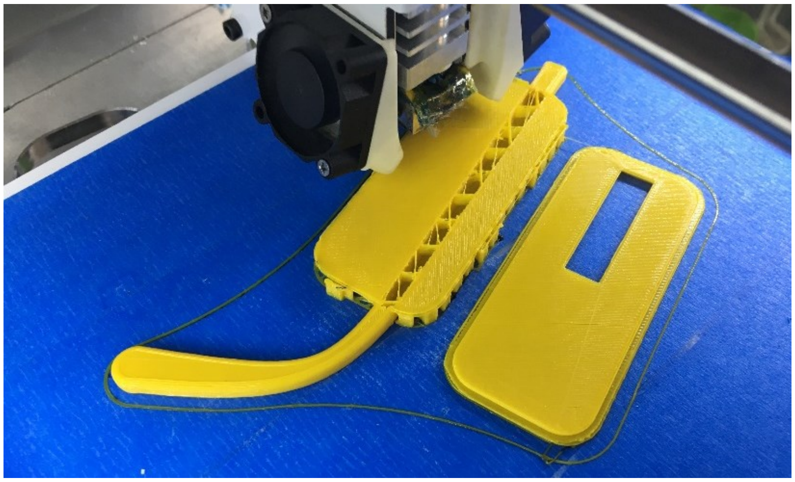


Figure 82 3D printing NicoCam frames v3, May 16, 2017.

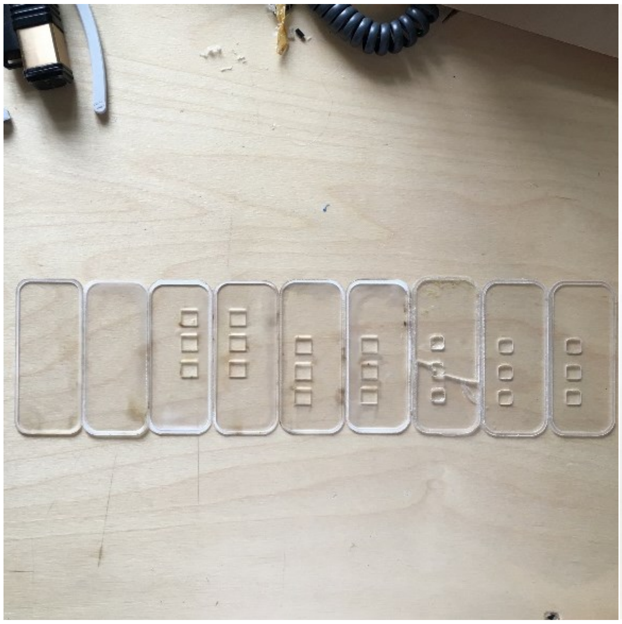


Figure 83 Laser cut acrylic case cover iterations, May 25, 2017.

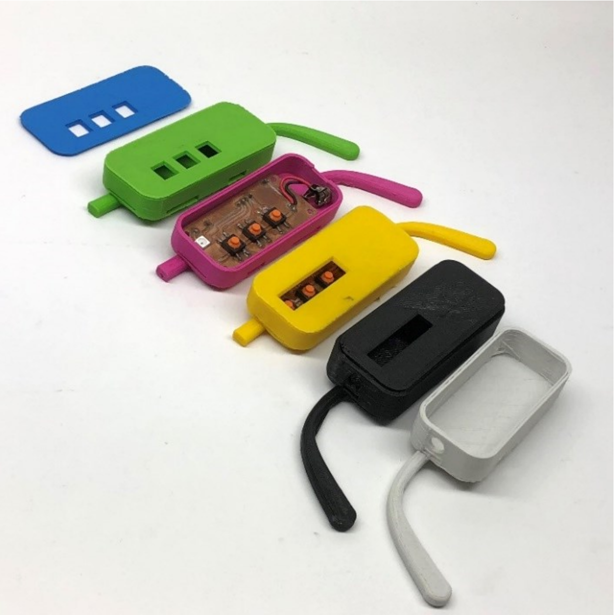


Figure 84 Selection of NicoCam frames prototypes coded by PLA colour, June 28, 2019.

Raphaël also helped me imagine a smart solution for providing the wearer of the glasses with visual feedback. Like Snap's glasses, I wished to include an LED indicator in the corner of the wearer's vision, to present simple feedback and information about the system in the absence of a rich display technology. Raphaël suggested that I use a surface mount (SMT) LED instead of the through-hole LED I had used in the earlier prototype. He suggested that I mount a light refracting prism on the surface of the outward-facing SMT LED, then redirect that light to the wearer's peripheral vision with a short length of fiber optic cable. I included affordances for this design approach in the final blue version of the left arm frames.

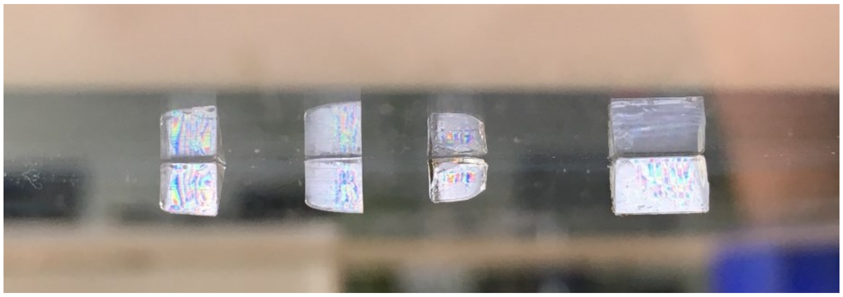


Figure 85 Light refracting prisms test, May 24, 2017.

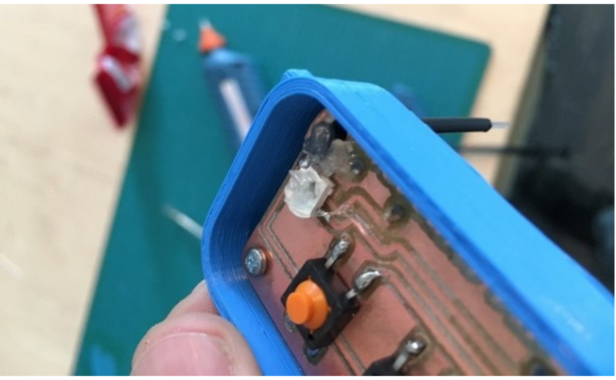


Figure 87 SMT LED with light refracting prism and fiber optic cable, May 24, 2017.

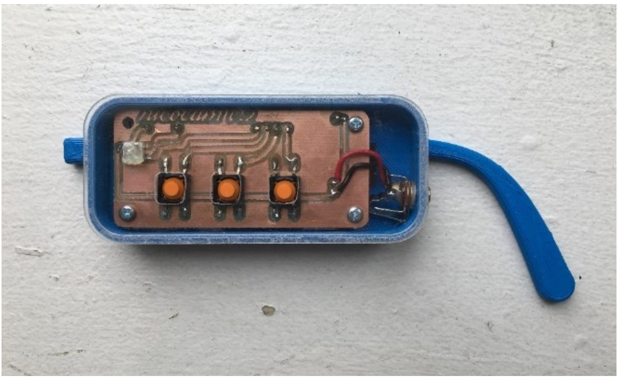


Figure 86 Final version of NicoCam v2 left temple arm, May 24, 2017.

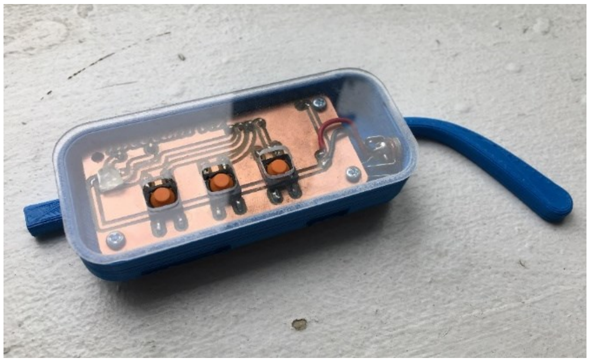


Figure 88 Final iteration of NicoCam v2 left temple arm, May 24, 2017.

The software for the NicoCam v2 was rudimentary but functional. The three buttons on the left arm allowed the user to record a video, take a picture, or shut down the device safely. The fiber optic LED worked in principle, but I did not complete the design of this aspect of the device to a high degree of finish.

As the camera glasses prototype progressed, I became increasingly frustrated with the size of the Raspberry Pi Zero W and PiCamera in a glasses product form. No matter what designs I sketched, the PiCamera's needlessly large PCB made it impossible to embed it in an elegant pair of glasses. Likewise, the Pi Zero created an unavoidably large shape along the glasses frame arms. My frustration at the size of the computer and camera boards pushed me to pursue other product forms, beyond glasses. I stopped work on the glasses at this stage. Some part of me thinks that it would have been wise to complete the project, even in a rough form, because it was quite close to being a decent-looking functional prototype. Nevertheless, breaking from this project opened new doors that I might not have had time to explore had I stuck to the original idea.

ewaste club

Looking back at it now, I can see that my experimental interest in network cameras split into two parts in early- to mid-2017. First, I continued work on trash computers. This half of my research experiments collects all of the functioning computer apparel and camera apparel designs that I explored through subsequent sketching and making. I call these trash computers because their designs embrace the commodification of computational hardware components. If Apple and Mattel can treat computers like disposable material, then so can I.[91]

Second, I began to push harder on making more traditional fashion objects, such as clothes, jewelry, and bags, with computational supply chain aesthetics. This second category of experiments captures my work embedding non-functional computer components and technical terminology as ornaments in a fashion style that I call *ewaste aesthetics*.

From this point forward, I referred to the broader project, which encompasses both functional and non-functional computer designs, under the banner of *ewaste club*.[92] ewaste club is the main project output from my research-creation dissertation. It is composed of hundreds of experiments that explore what it means to use computers like a fashion material. We live in a moment of ubiquitous network connectivity and inescapable symbiosis with the supply chain. For a period of four years, I imagined what it would mean for fashion and culture to integrate the messy, technical, capitalist aesthetics of electronics manufacturing as beautiful embellishments and inspiration for clothing and objects.

Part of the reasoning for approaching computers in this way is the overwhelming oligopolistic power that network technology corporations lord over us. In order to appeal to the network illiterate masses, corporations like Apple create sealed packages of predefined software-hardware-service integrated applications. This has advantages and disadvantages, but the primary danger is that these powerful corporations are incentivized to keep the population computationally illiterate, because they are better consumers if they do not understand the medium with which they commune every day.[93]

---

[91] See Chapter 5: Supply Chain for more about disposable computers and their relationship to meaning.

[92] The website ewaste.club served as a testbed for the internet-connected prototypes I subsequently created, and will most likely be repurposed to document the sum of these projects subsequent to the publication of this dissertation.

[93] See Chapter 5: Supply Chain, for more on this subject.

Not only is it extremely difficult to compete with such a large organizations and their talented employees, but it is also not very aesthetically interesting. Apple’s design style is bound up with a belief that it is best not to see too much of the internal workings of computing machines.[94] The power of the independent artist and designer working at the edge of network fashion is to zig where the behemoths zag. For this reason, I chose to embrace raw electronics as decorative elements, and to seek inspiration from their material forms (colour, line, texture) for experiments in embroidery and garment making. The ewaste club projects, including the aforementioned camera glasses, ask what it would mean if visual and fashion culture embraced the weirdness of living with computers, rather than papering over it with the tired twentieth century memes that encourage disempowered users to embrace their creativity in media forms that computers are bound to colonize, refactor, and possibly render obsolete.

Pivot to Fashion and Accessories

3D Printed Jewelry

In my disenchantment with the camera glasses, I began to make jewelry. First, I used the Form One SLA printer to make a few very basic rings. The results were moderately interesting, but it was a good exercise to print something then wear it on one’s person for the next few days to get started in this new direction. Next, I went to jewelry craft shops and looked into the fixtures and chains required for making necklaces. I then fired up Fusion360 and made a new enclosure for the camera glasses computer. It was a heart shaped black box, which I mounted on a chain and wore around my neck. The images it recorded would no doubt have been awfully shaky, but it signalled a shift to more aggressive aesthetic explorations.

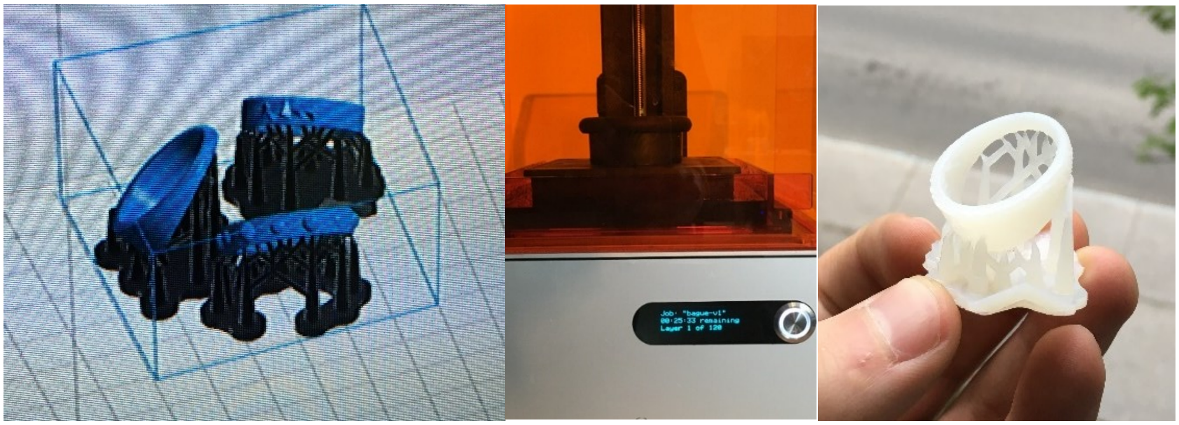


Figure 89 SLA printed ring, June 4, 2017

---

[94] Despite the Bondi blue iMac era of see-through Macintosh devices.

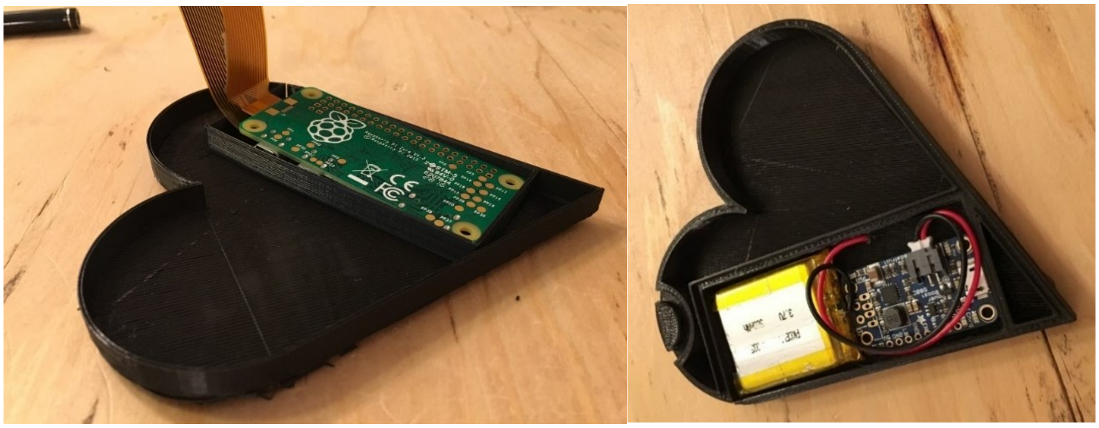


Figure 90 Heart Camera Necklace 3D printed prototype, June 7, 2017.

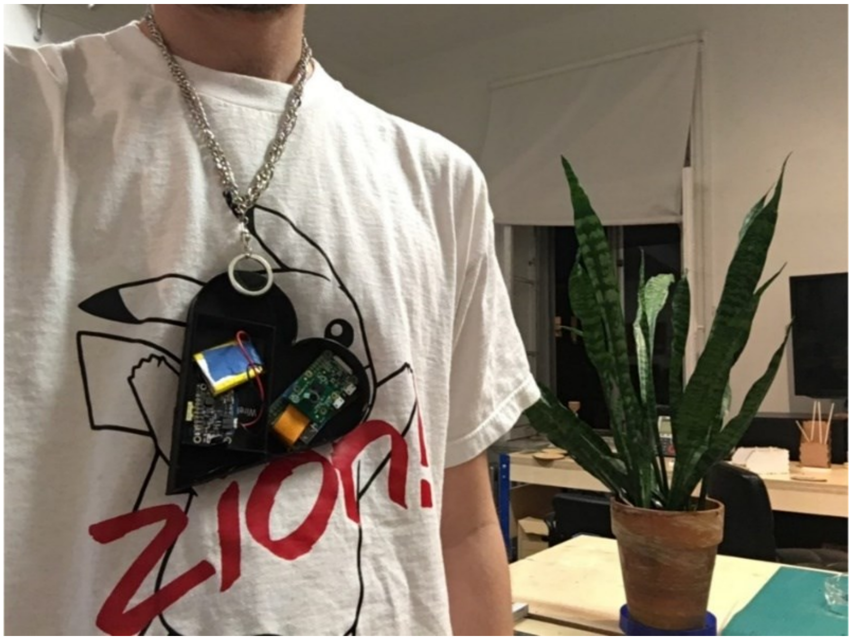


Figure 91 Heart Camera Necklace 3D printed prototype, June 9, 2017.

Graphic T-Shirts

At the same time, I began making graphic t-shirts with thermo-adhesive vinyl and the Fab Lab du PEC's vinyl cutter. I embraced lab manager Raphaël's technique of drawing with felt markers, taking pictures of the drawings, vectorizing them with Inkscape, then cutting the resulting files out with the vinyl cutter. The first shirt I made was a little drawing of a shoe. The second was a red shirt with black hearts drawn in the same manner. Later, I vectorized a microscopic picture of my beard, then transferred that to a shirt. I took inspiration from baseball uniforms and created a shirt to symbolize my research-creation experiments. The back of this shirt was emblazoned with the phrase "E-WASTE 17"—I had not yet decided to drop the hyphen. When I learned that Saudi Arabian clerics had declared a fatwa on Pokémon and Pokémon Go because they believe the game is Zionist, its imagery promotes polytheism, and that its mechanics support Darwin’s theory of evolution, I created a shirt with Pikachu obscured by the word “zion” in large red letters [309]. I worked through these ideas, testing which sources of inspiration yielded the best results and felt the most in line with my style.

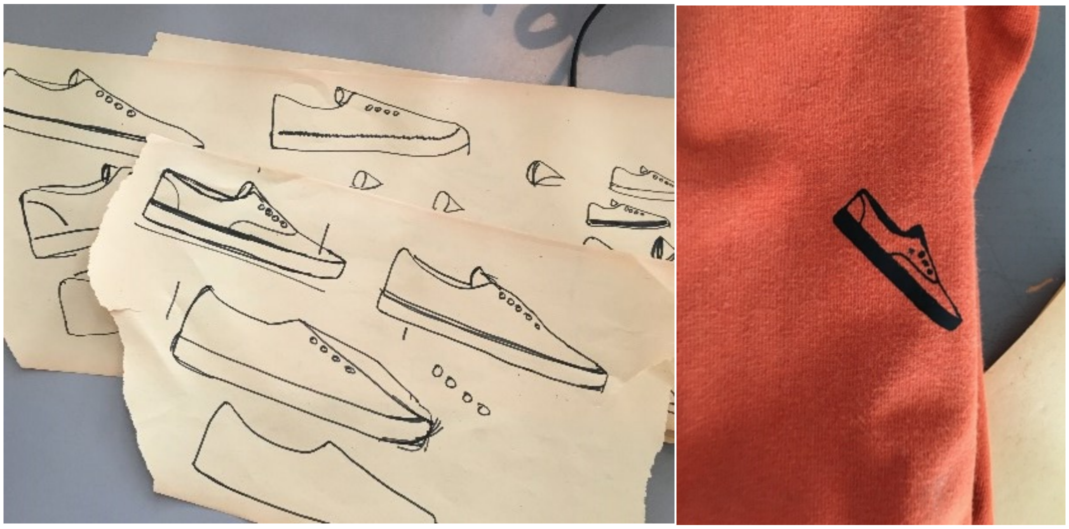


Figure 92 Shoe shirt, March 10, 2017.

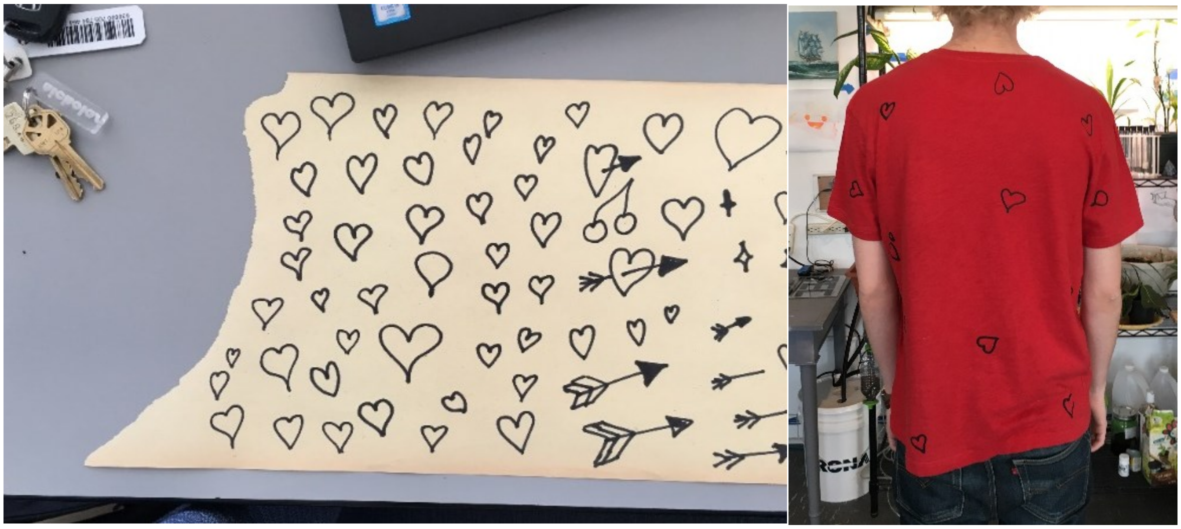


Figure 93 Heart shirt, March 19, 2017.

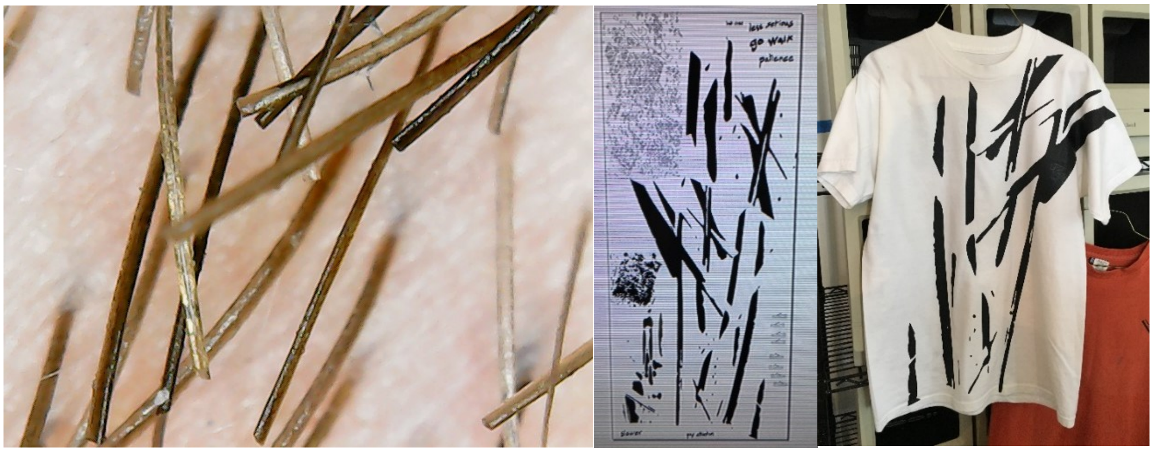


Figure 94 Microscopic beard image and vector, March 16, 2017.

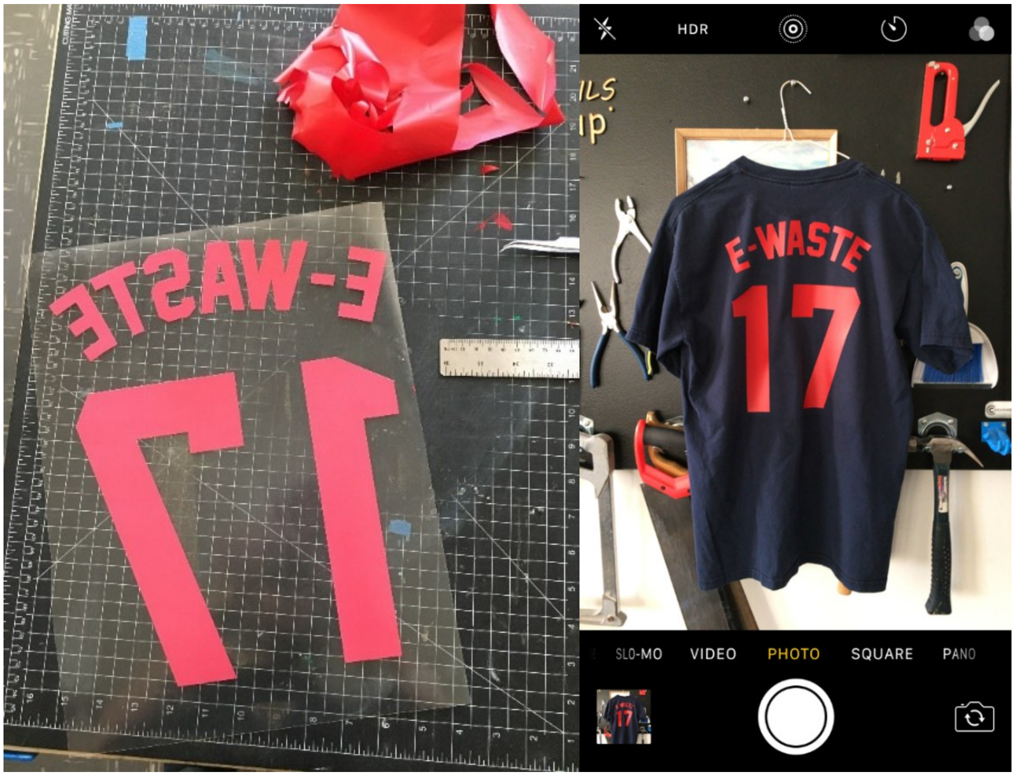


Figure 95 Thermo-adhesive vinyl, March 25, 2017.

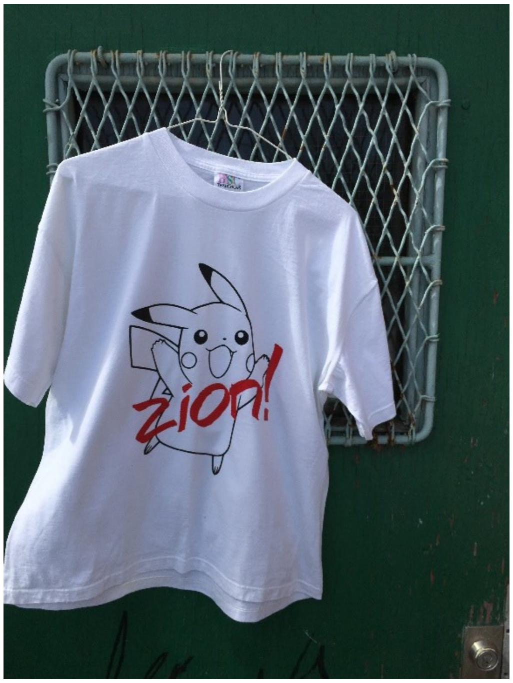


Figure 96 Pikachu Zion shirt, March 26, 2017.

Laser Material Experiments

I also experimented with laser engraving imagery onto unusual substrates. I engraved screenshots from my camera roll onto thin pieces of transparent acrylic, which I also cut to smartphone proportions. I attempted to melt and roll them into a screenshot bouquet, but it looked terrible at the scale that I tried. I also burned pieces of denim with drawings. I put Velcro on the back of one of the denim drawings and attached it to complementary Velcro sewn to my backpack. One day, I found some discarded toys at the Fab Lab. Something struck me, and I put the dress on a stray brick and engraved a Disney princess-like face onto it. I called it Brickerella and it became a de facto mascot for the Lab. These experiments did not go anywhere in particular, but I still find them intriguing material explorations that may become useful at a future date.

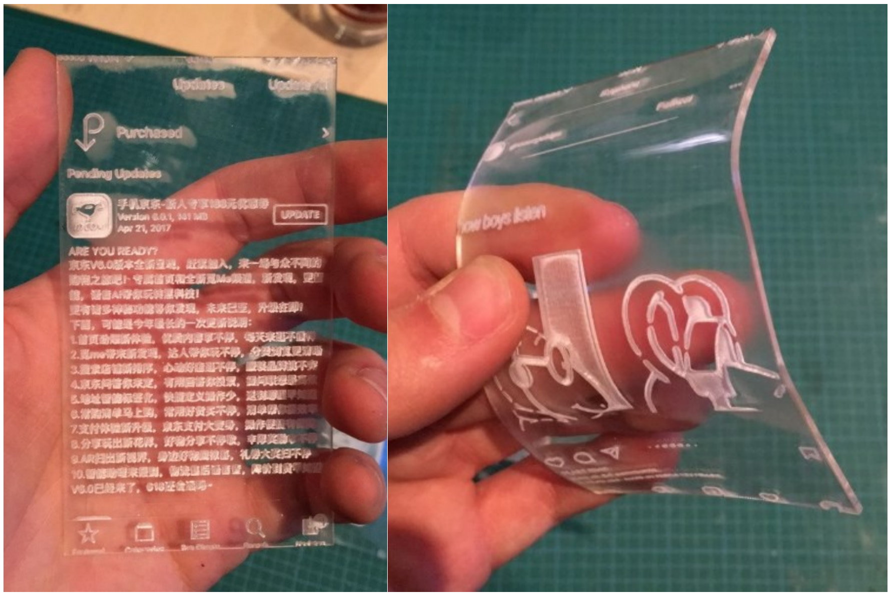


Figure 97 Laser engraved iOS App Store update screenshot and Instagram meme page, May 5, 2017.

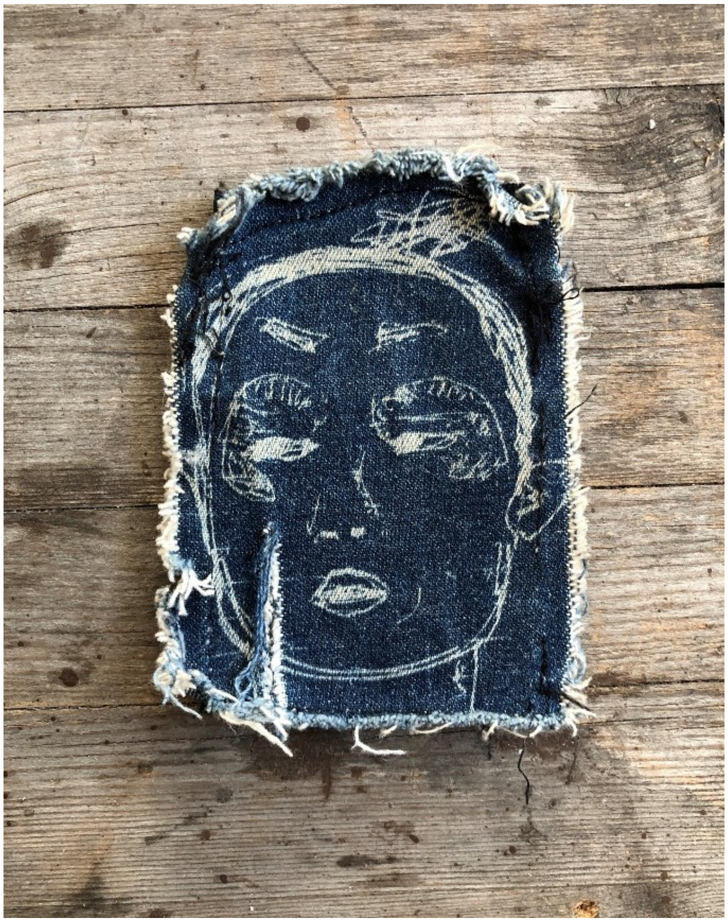


Figure 98 Laser burned illustration on denim, August 24, 2020.

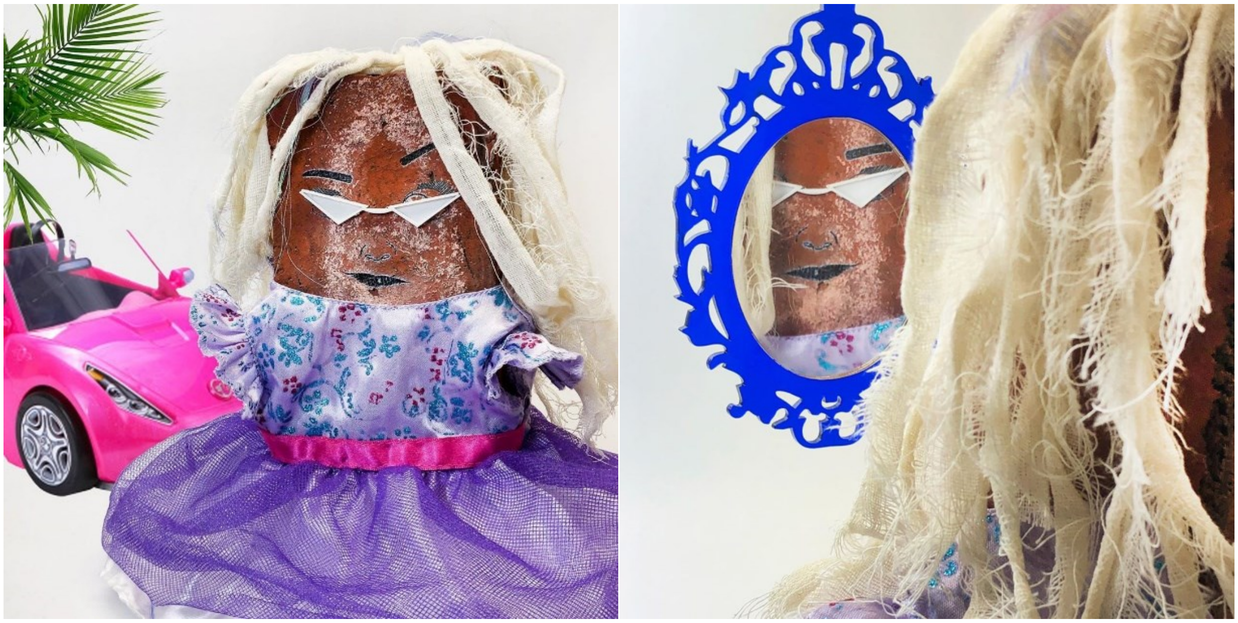


Figure 99 Brickerella promotional photos, June 9, 2019.

Shenzhen, China

In July 2017, I traveled to China's Pearl River Delta to research how a designer-artist's practice might change with direct access to the Chinese manufacturing supply chain. The Pearl River Delta is an area of South East China and nearby territories including Shenzhen, Hong Kong, Macau, and a handful of other metropoles. This area, and especially the city of Shenzhen, are popularly understood as the centre of worldwide electronics manufacturing and trade [310], [311]. The focus of my research trip was to create a series of vlogs documenting my fieldwork researching the region's resources, and to conduct makerly experiments using the city of Shenzhen as if it were one great big fabrication laboratory. The research was funded by Concordia University's Interdisciplinary Programme's Fine Arts Travel Grant and Milieux's Textiles and Materiality Individual Project Grant. The research I conducted in China was inspired by my research prior to departure, but it remains a separate project that I documented primarily in a series of vlogs called Field Notes, which is available on YouTube [312]–[318]. Although the research trip to Shenzhen was a separate project from my dissertation, my fieldwork and findings played a role in the projects I pursued upon my return to Montreal, and so I will summarize the relevant highlights.

Exploring Factories and Markets

During the first couple of weeks I spent in China, I accompanied a former colleague while she visited factories to source products for future distribution in America. We visited a factory that produces toy drones and another factory that produces both toys and industrial agricultural drones for watering crops.

We visited an underwear factory in a rural area far outside of the city-centre, and a toy factory that made remote controlled cars, water bottles, and fidget spinners. We also visited a dollar store product distributor, which is a multi-story showroom that worldwide dollar store retailers visit to select which items to add to their inventory.

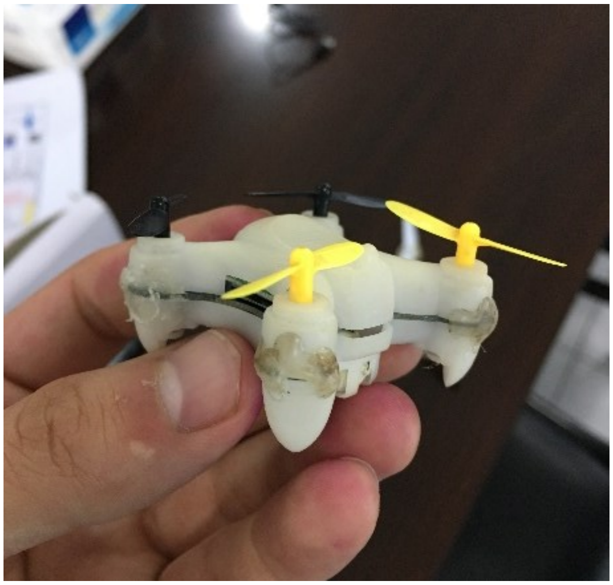


Figure 100 Toy drone prototype, July 7, 2017.

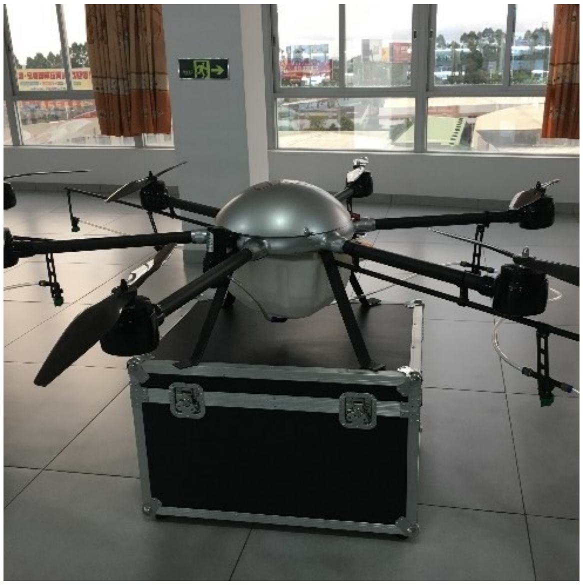


Figure 101 Agricultural drone with water tank, July 7, 2017.

Figure 102 Agricultural crop watering drone just before a live, rooftop demonstration, July 7, 2017.

Figure 103 One size fits all underwear factory, July 8, 2017.

Figure 104 An aisle in the dollar store distributor's model store, July 7, 2017.

I visited a few more factories with members of a local maker space called Chaihuo x.factory [319], [320]. We toured the full production line at a Shenzhen plastics injection molding factory that produces parts for name brand electronics makers like Lenovo and Samsung. We also visited an upstart laser cutter factory in Donguan. Finally, we visited a small and very messy CNC prototyping shop in the outskirts of Shenzhen.

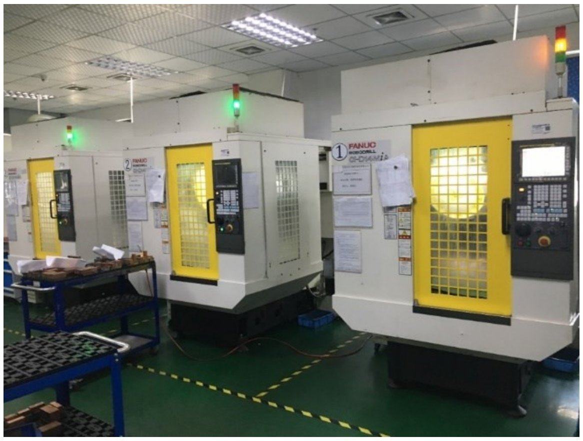


Figure 105 CNC machines for milling molds in the injection molding factory, July 29, 2017.

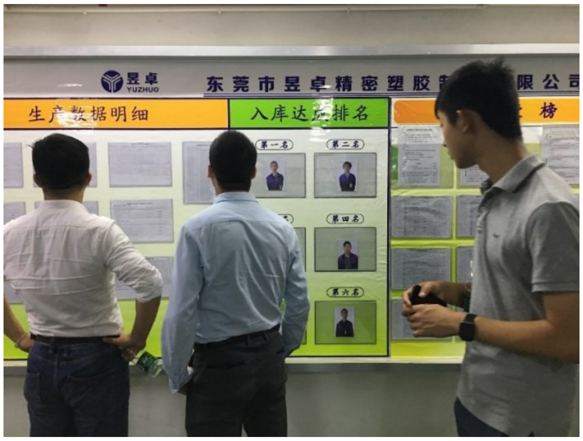


Figure 106 Employee information boards in the injection molding factory, July 29, 2017.

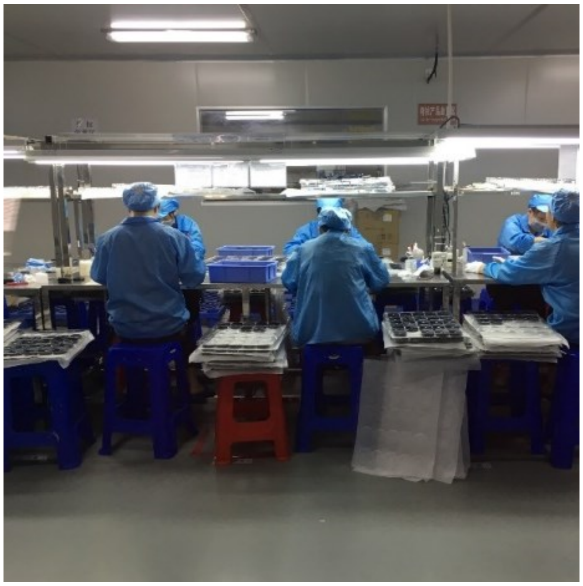


Figure 107 Workers in a clean area finishing the injection molded parts, July 29, 2017.

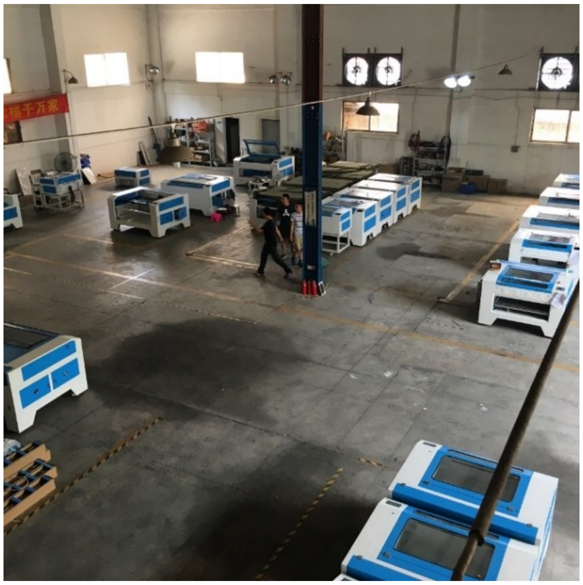


Figure 108 Laser cutter factory floor, July 29, 2017.

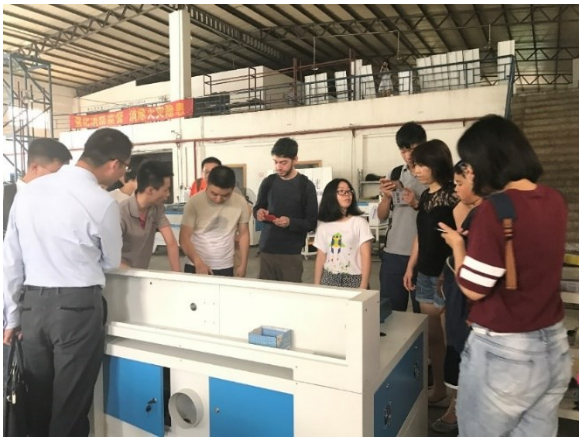


Figure 109 Inspecting a nearly completed laser cutter, July 29, 2017.

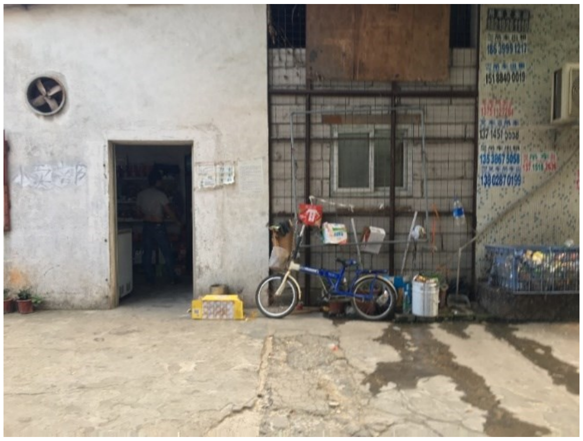


Figure 110 Entrance to the CNC milling shop, July 29, 2017.

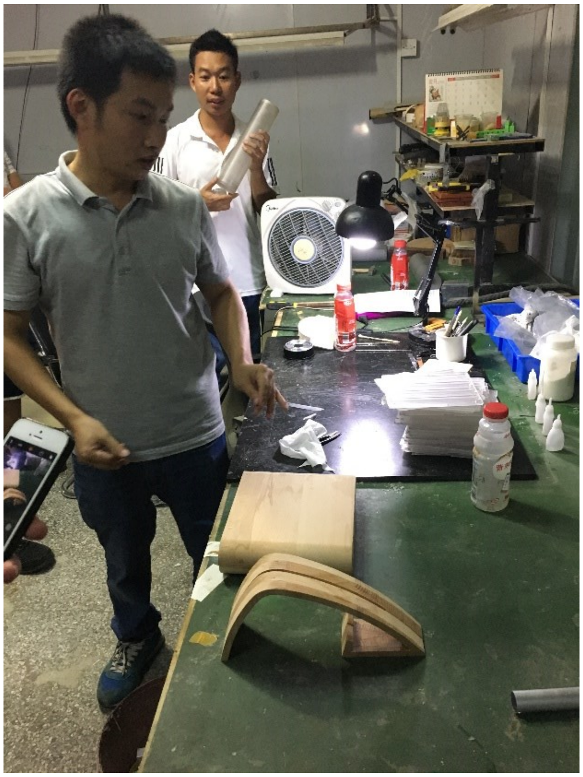


Figure 111 Demonstration of the CNC prototyping process, July 29, 2017.

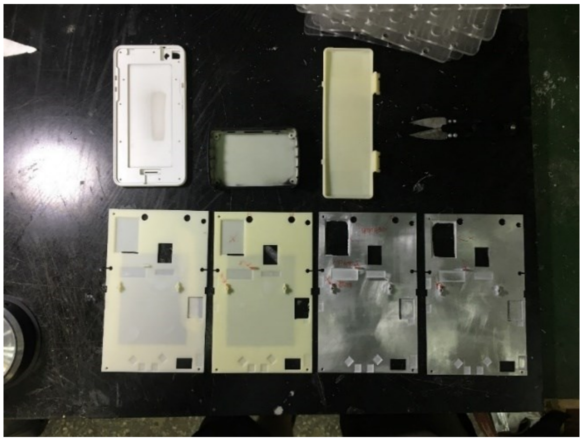


Figure 112 Iterations in milling a CNC part, July 29, 2017.

I spent a large part of my time in China visiting Shenzhen's wholesale markets. Shenzhen is best known for its electronics market area, called Huaqiangbei, but like all large cities in China, Shenzhen has wholesale markets dedicated to many product categories [311], [321], [322]. Typically, the wholesale markets are large single or multi-story buildings. Long and winding hallways are divided into stalls, which range from around two to ten meters wide. Each stall is occupied by a storefront that represents a distributor located somewhere near Shenzhen. Some keep stock in the stall while others show only samples and can deliver larger quantities upon order. Each market is dedicated to a product category. During my stay, I visited a combination toys and stationary market, a jade and jewelry market, a jewelry supplies market, a musical instrument market, a phone accessory market, a variety of electronics markets, an iPhone parts market, an Android parts market, and a few of fabric markets.

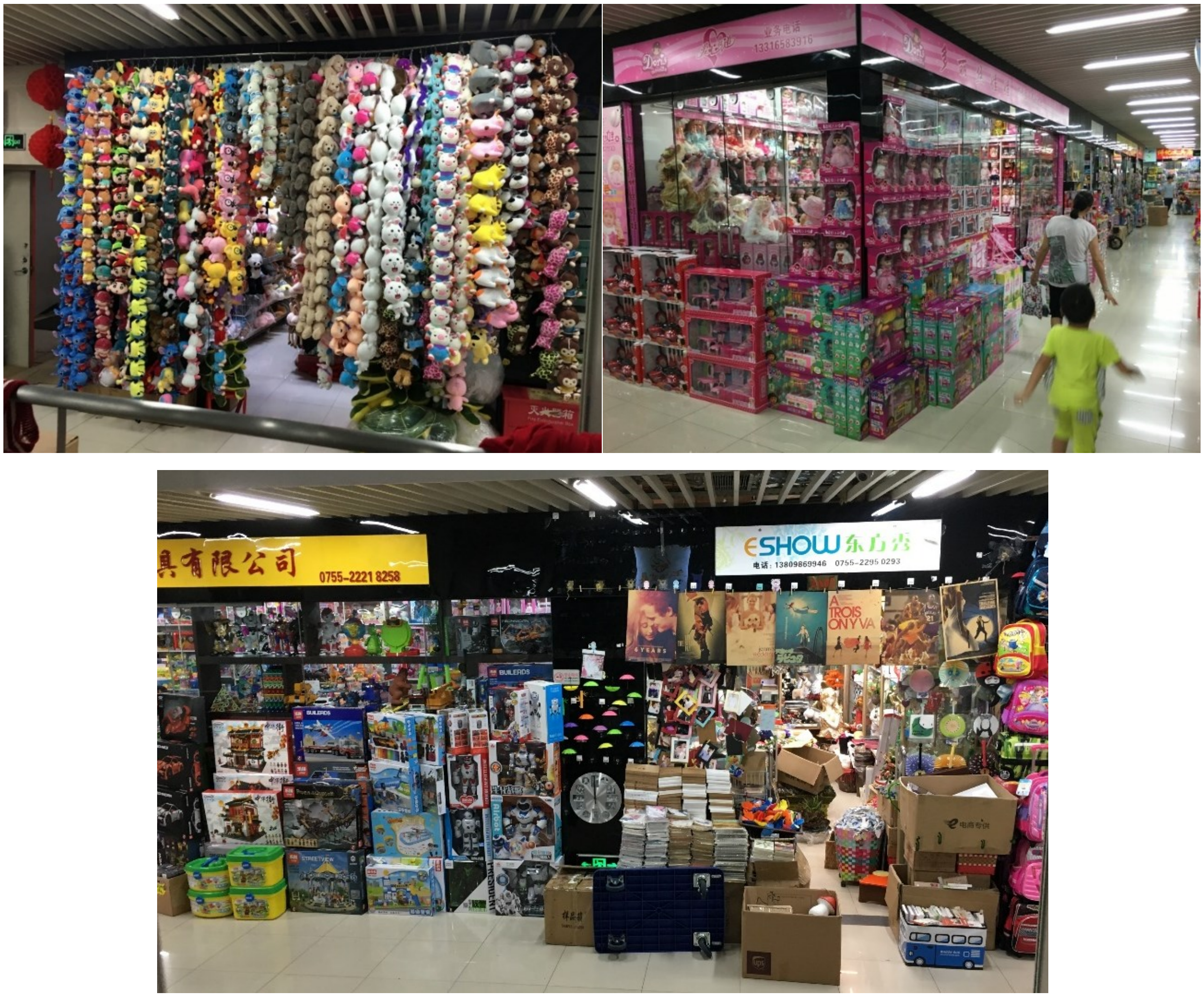


Figure 113 Toy and stationary wholesale market, July 14, 2017.

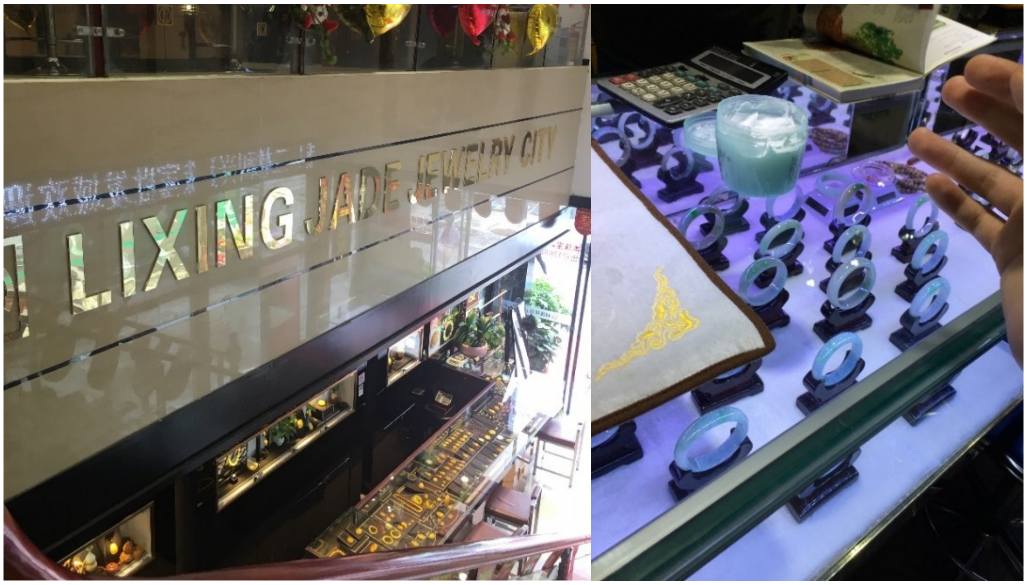


Figure 114 Lixing Jade Jewelry City, Shenzhen, July 14, 2017.

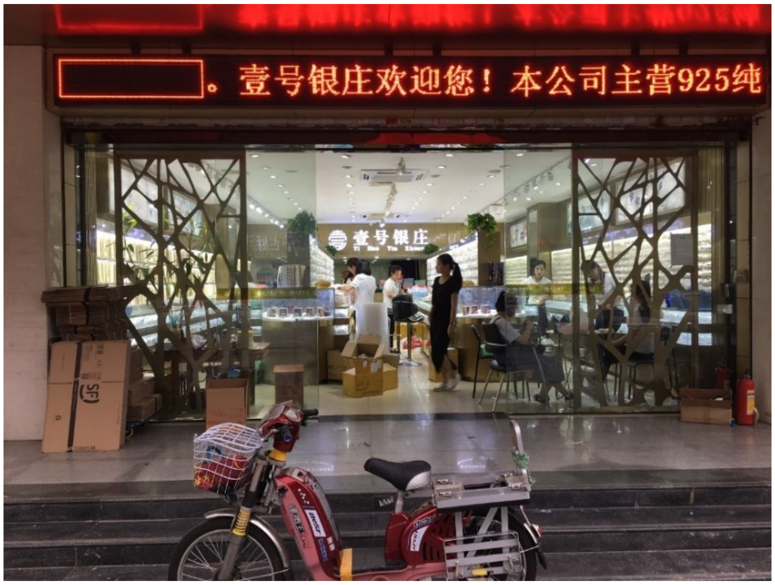


Figure 115 Wholesale jewelry supplies store, July 19, 2017.

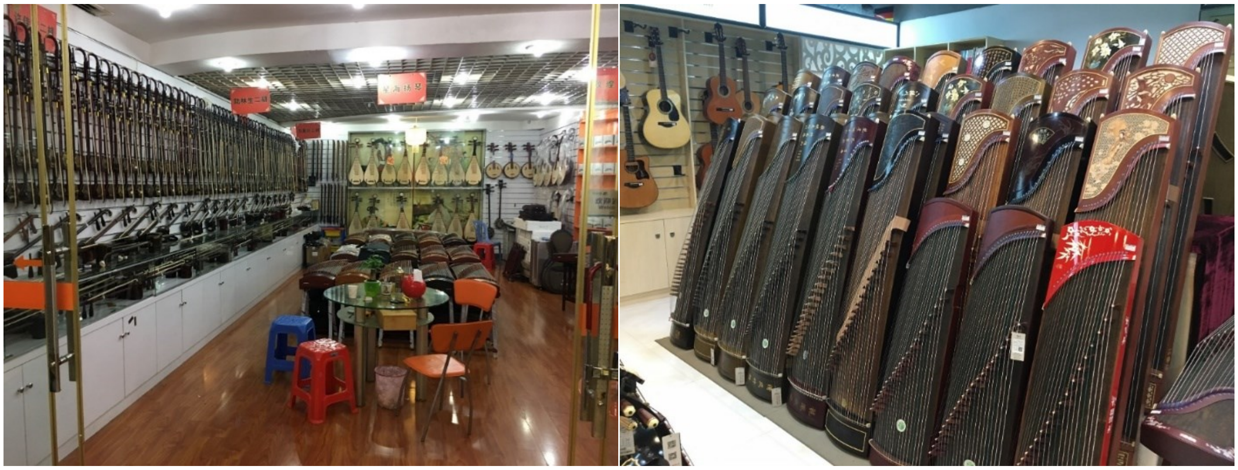


Figure 116 Store in the wholesale music instrument market, July 18, 2017.

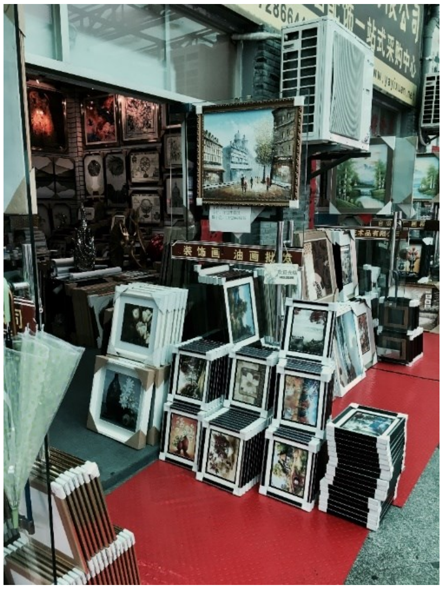


Figure 117 Gallery in Dafen painting village, Shenzhen, July 16, 2017.

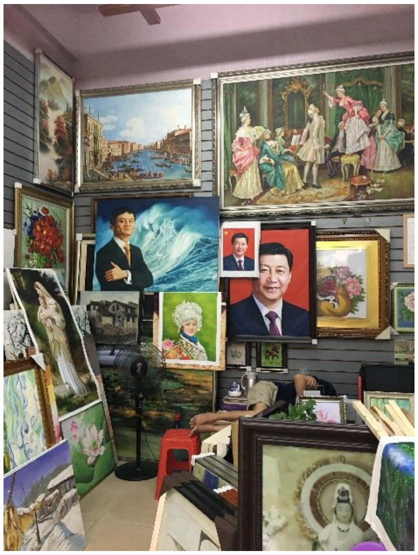


Figure 118 A clerk asleep in a painting gallery in Dafen painting village, Shenzhen, July 16, 2017.

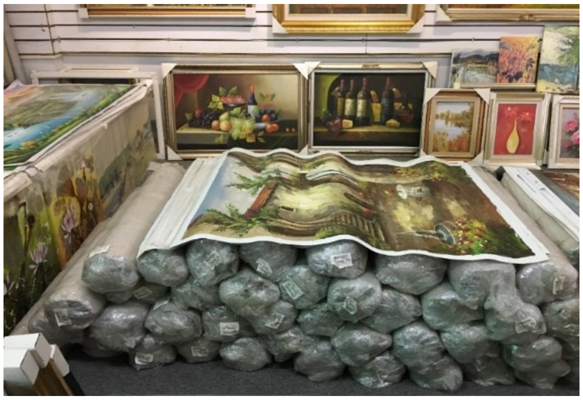


Figure 119 Painted canvases are rolled together and stored on the floor in Dafen painting village, Shenzhen, July 16, 2017.

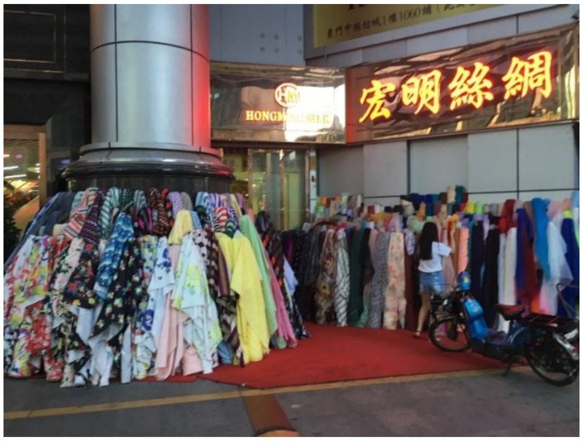


Figure 120 Entrance to a fabric market in Shenzhen, July 13, 2017.

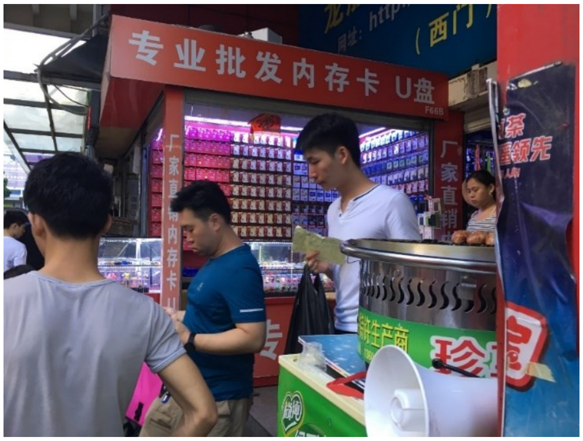


Figure 121 Entrance to a Huaqiangbei wholesale market, July 12, 2017.

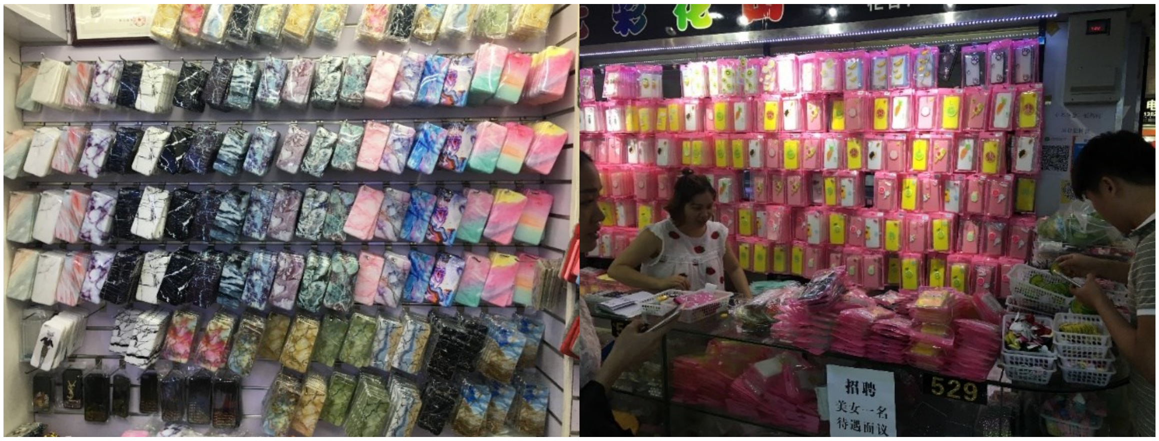


Figure 122 Stores in the multi-story smartphone accessory market in Huaqiangbei, July 14, 2017.

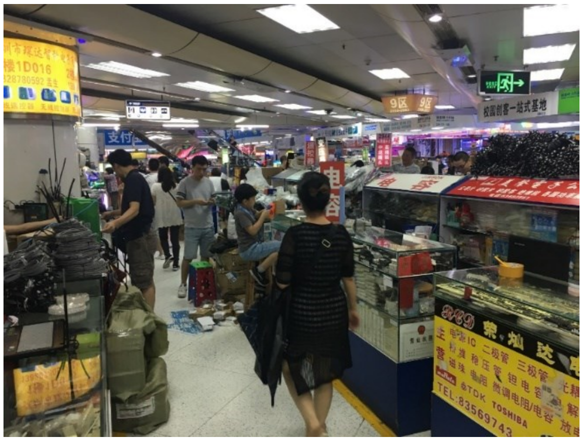


Figure 123 Huaqiangbei electronics market floor, July 18, 2017.

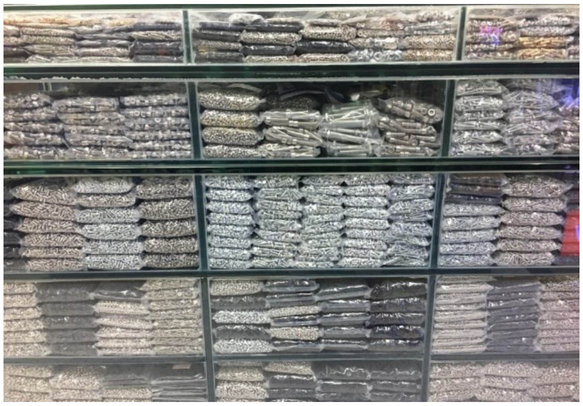


Figure 124 Screws and fastener store in Huaqiangbei, July 18, 2017.

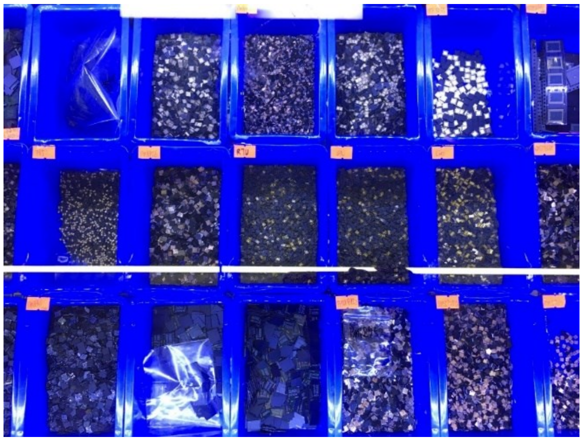


Figure 125 Smartphone components bins, July 25, 2017.

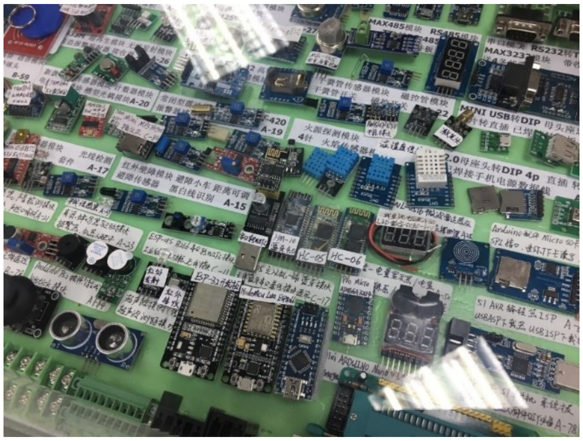


Figure 126 Arduino accessory store, July 18, 2017.

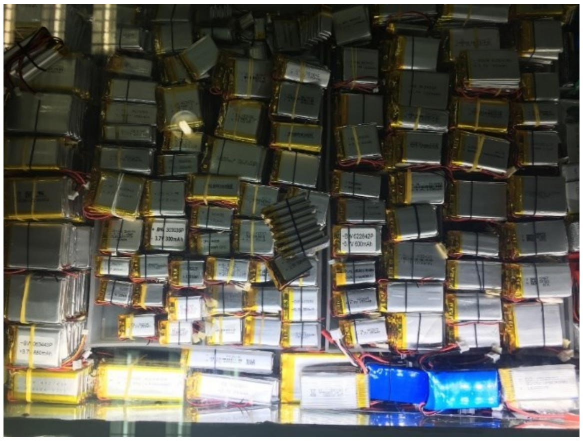


Figure 127 Lithium polymer battery store, July 18, 2017.

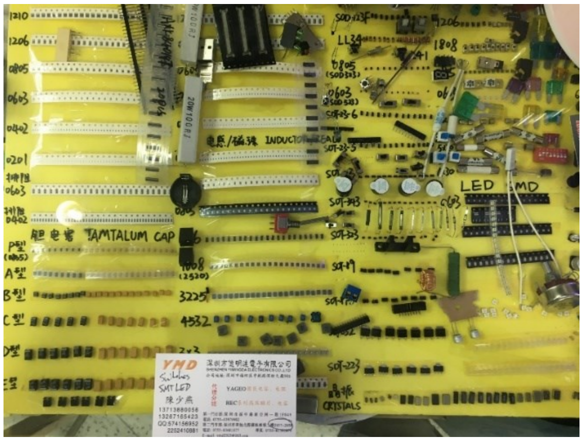


Figure 128 SMT LED store, July 18, 2017.

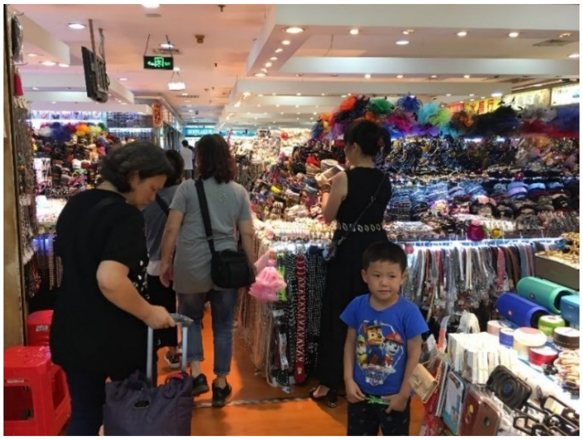


Figure 129 Luohu fabric market, July 17, 2017.

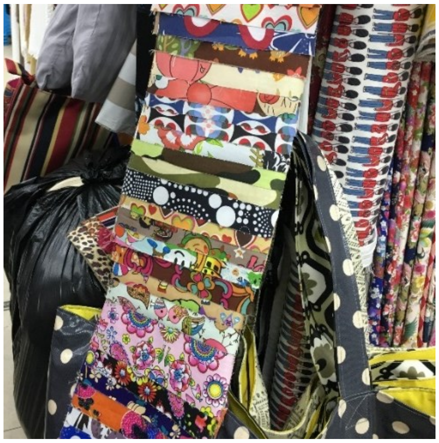


Figure 130 Fabric samples in Luohu market, July 17, 2017.

Camera Jacket

While in Shenzhen, I decided to try to use the surrounding resources to make two new projects. The first project I embarked upon was one that I had dreamed about in Montreal, but never had the resources to complete. I had long imagined that a Camera Jacket would be an interesting type of garment to produce, but the cost of custom tailoring in Montreal had made that an intimidating prospect. After visiting fabric markets in a few different parts of the city, I eventually landed on the fabric market on the top floor of the Luohu shopping center. Luohu is the first train stop in Shenzhen after crossing the border from Hong Kong. It is known to locals and tourists as a counterfeit shopping center, where one can easily find fake versions of designer labels.

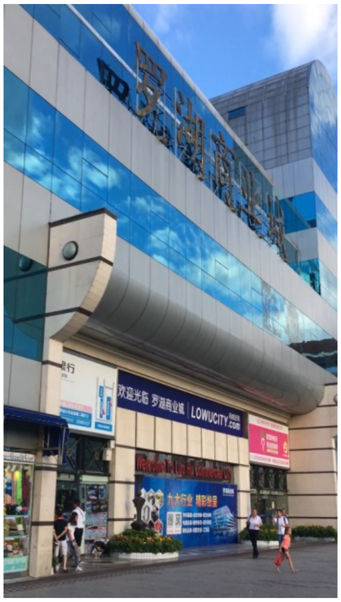


Figure 131 Luohu shopping center, July 24, 2017.

Luohu is also known as a pit stop for people visiting Hong Kong and the Pearl River Delta who are in the market for custom clothes. The top floor of the shopping center is shared between a huge wholesale fabric market, which is home to around 200 different vendors, and several dozen tailors. Customers pick a fabric and bring it to a tailor who will turn it into a custom suit, dress, or jacket.

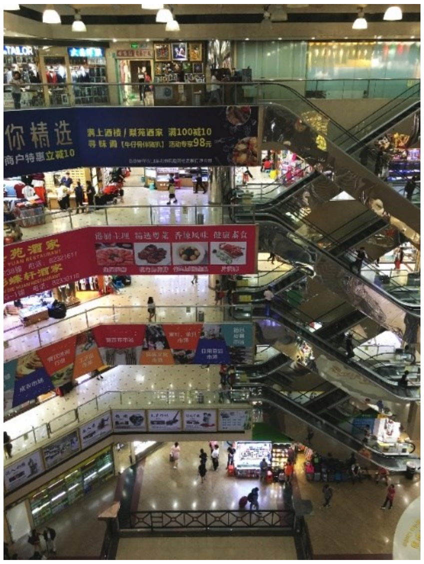


Figure 132 Luohu shopping center, July 17, 2017.

To practice making with the supply chain, I decided to collaboratively construct a camera jacket with one of the tailors in the market. Anxious to choose wisely, I took my time, and eventually found a tailor shop with an assistant who spoke English. I explained that I wanted a jacket in the style of a green K-Way brand rain slicker, which I had brought with me. I asked the shop assistant how we could go about putting pockets to hold a Raspberry Pi and Pi Cameras. Initially, I wanted to install several pockets, in the front, back, arms, and jacket sides. She told me it was too complicated and attempted to turn me down, but I persisted. In the end, I convinced her to sit with me for a few moments and do a very rough prototype with some scrap cloth she had lying around. The head tailor scoffed but her smirking belied a certain amount of amusement at the project. In general, these tailors are used to performing exact copies of existing garments, or making very standard suit jackets and pants. We settled on a pair of designs: one sewn in and the other glued. In all I contracted the tailor shop to make three jackets: pink, black, and red. I paid the bill and she told me that the tailors and seamstresses at the factory (I do not know where) would have it finished in a few days. We communicated over WeChat and a week later I returned to the store to collect the results. The glued in version was weakly constructed, and it failed to hold together the water repellant material I had chosen. The sewn in pocket, however fared better. It was an appropriate size to hold the Pi Zero.

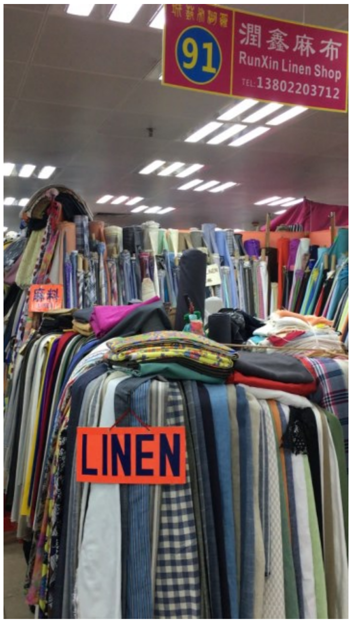
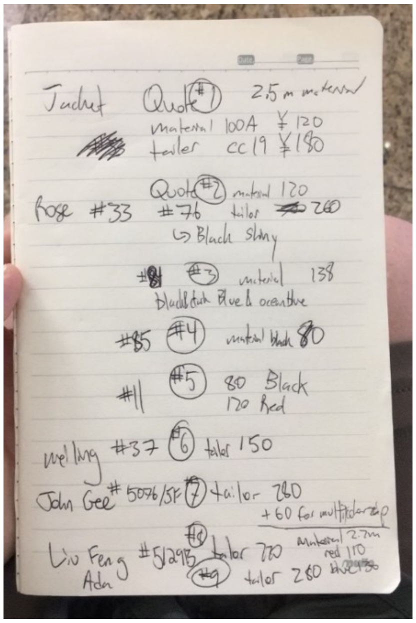


Figure 133 Picking cloth in the fabric market at Luohu shopping center, July 24, 2017.



Jacket Quote #1 2.5 m material
material 100A ¥120
tailor cc19 ¥180

Quote #2 material 120
Rose #33 #76 tailor 260
↳ Black shiny

#8 #3 material 138
black dark Blue & ocean blue

#85 #4 material black 80

#11 #5 80 Black
120 Red

Welling #37 #6 tailor 150

John Gee #5076/5F #7 tailor 280
+60 for multifolder zip

Liu Feng #5129B #8 tailor 220 material 2.7m
red 110
Ada #9 tailor 280 blue 130

Figure 134 Comparison shopping notes, July 24, 2017.

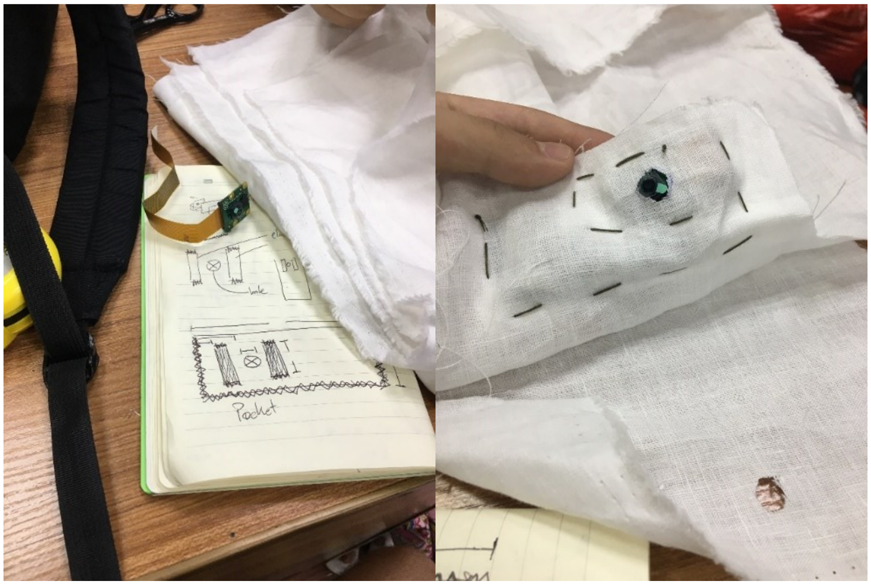


Figure 135 The tailor's assistant and I worked out a design to hold the camera in place some spare cloth, July 24, 2017.

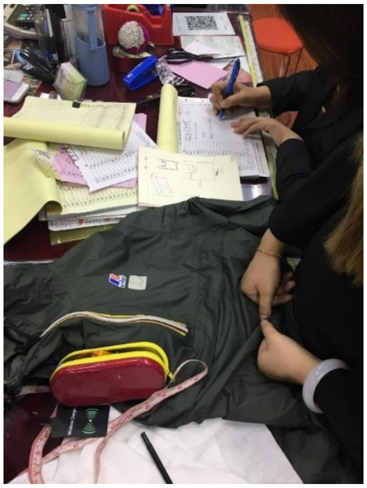


Figure 136 The tailor's assistant measures the reference jacket, July 24, 2017.

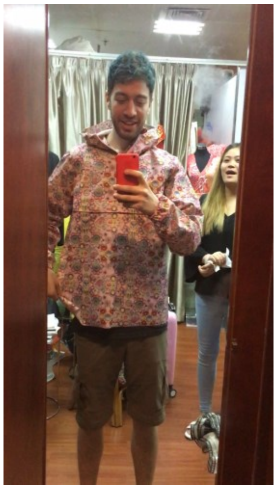


Figure 137 Trying on the finished product in the tailor's shop, July 24, 2017.

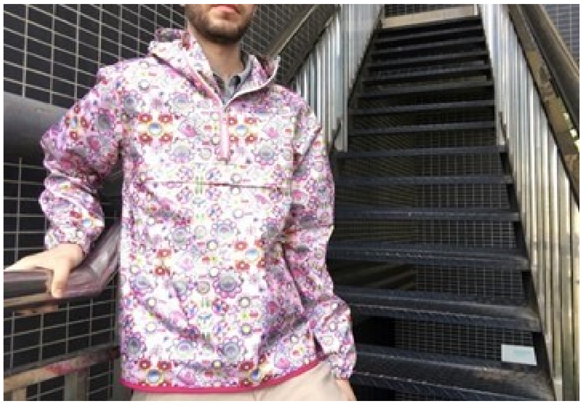


Figure 138 Finished jacket with metal eyelet on left breast for camera, July 25, 2017.

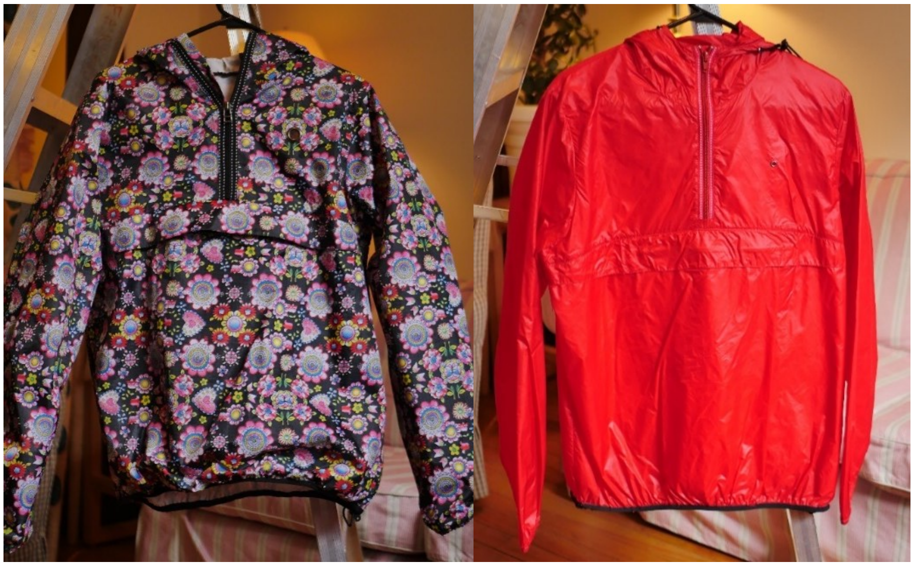


Figure 139 Black and red versions of the jacket, August 27, 2020.

Camera Collier

The second project I attempted while in Shenzhen was jewelry making. Before leaving, I had 3D printed a few rings and a heart-shaped Camera Pendant, but I was not yet satisfied. In Shenzhen, access to components was abundant and inexpensive. In the marketplace stores, minimum order quantity and shipping fees were not typically a problem, if the store was interested in my business at all. I must note that some vendors waved me away when it was clear from my look that I had no idea what I was doing.

I decided to create ewaste earrings and a waterfall necklace with cameras and flex cables I found in the market. These golden and shiny components screamed precious jewelry to me, and I wondered if it would be possible to translate them into something one might want to wear. I purchased components in the market and found jewelry making supplies in a neighbourhood not too far away.

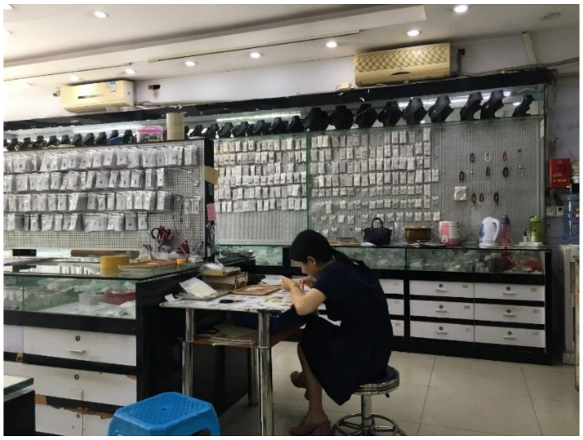


Figure 140 Jewelry supply store where I bought silver-plated chains and fixtures, July 31, 2017.

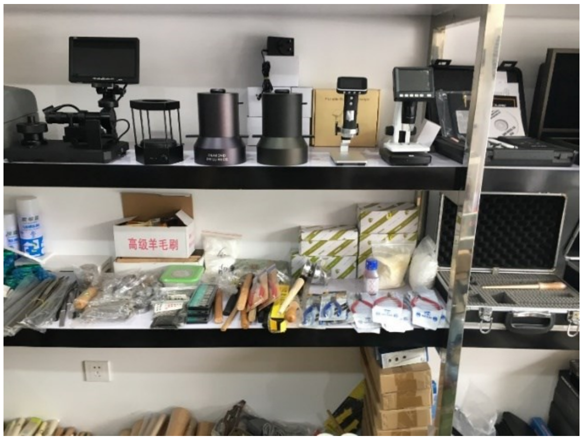


Figure 141 A jewelry tools and supply store in Shenzhen where I found the three essential pliers (bottom right, red), July 31, 2017.

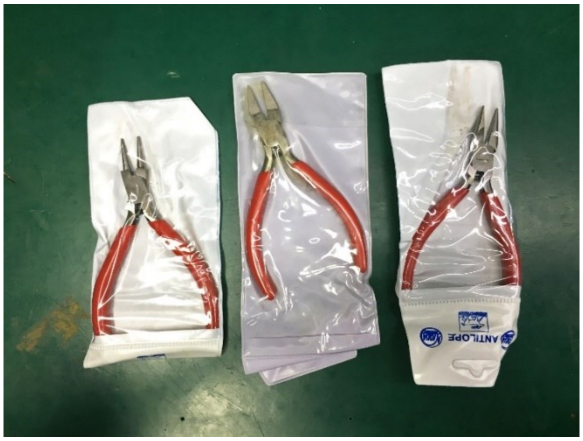


Figure 142 Three jewelry pliers, July 31, 2017.

With the help of local engineer Henk Werner, I was permitted to work on the project at Trouble Maker, a maker space on the top floor of a building in the heart of Huaqiangbei. I broke a few camera modules open, and pierced holes in some of the more aesthetically pleasing flex cables I had found. I attached them to necklace chains and earring fixtures.

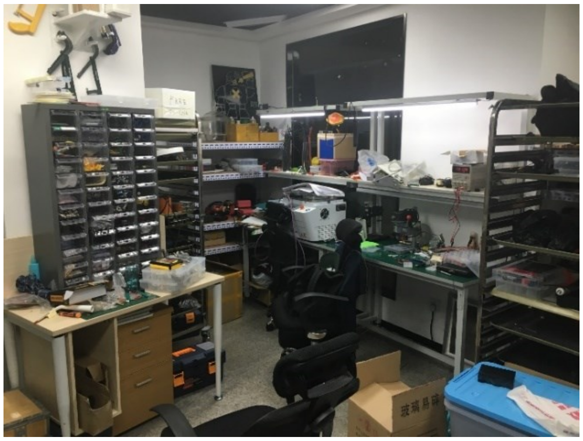


Figure 143 My work station at Trouble Maker, July 25, 2017.

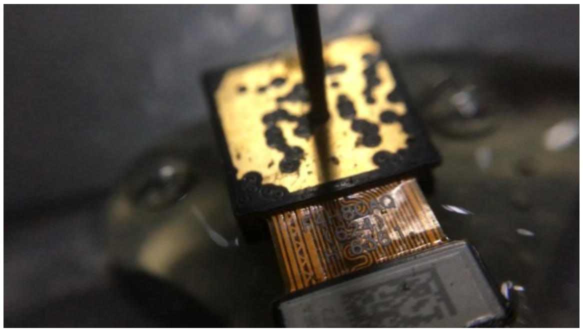


Figure 144 Drilling a hole through a camera close-up, July 25, 2017.

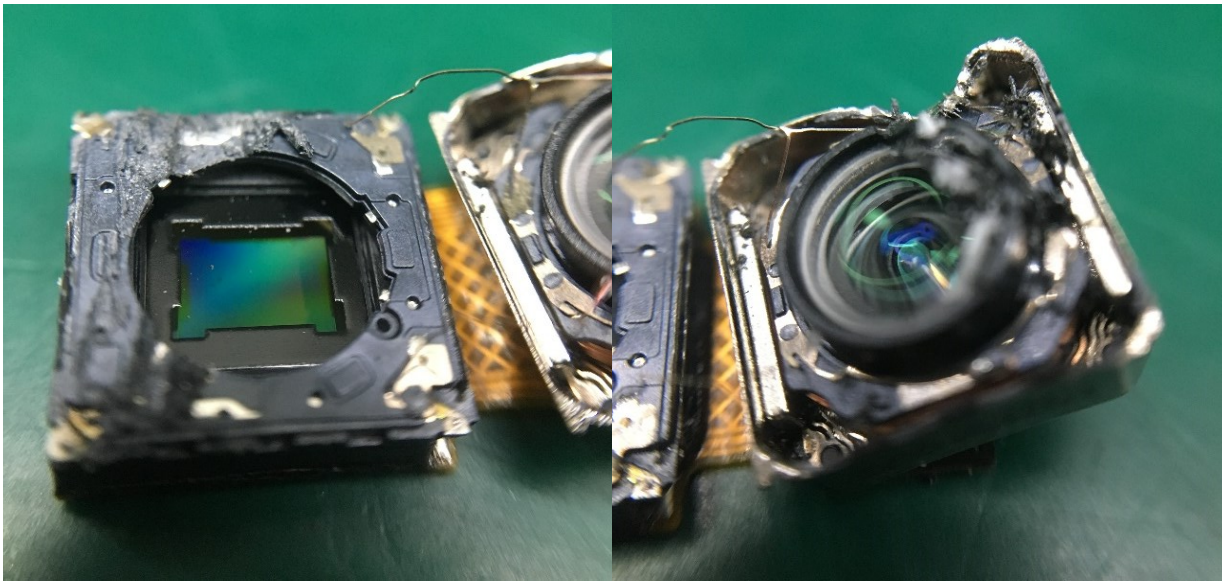


Figure 145 Inside the camera module are the lenses and image sensor, July 25, 2017.

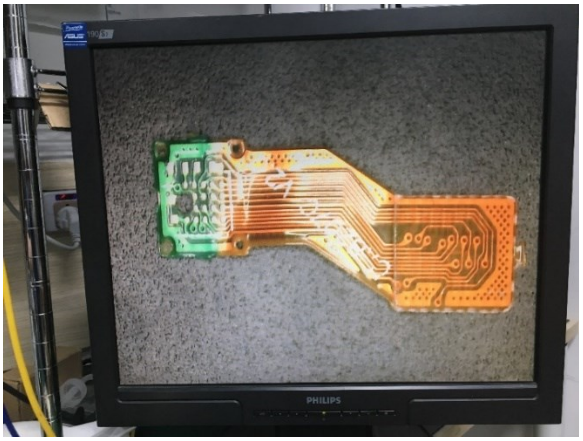


Figure 146 Microscope view of experiments drilling holes in flex cables with the drill press, July 25, 2017.

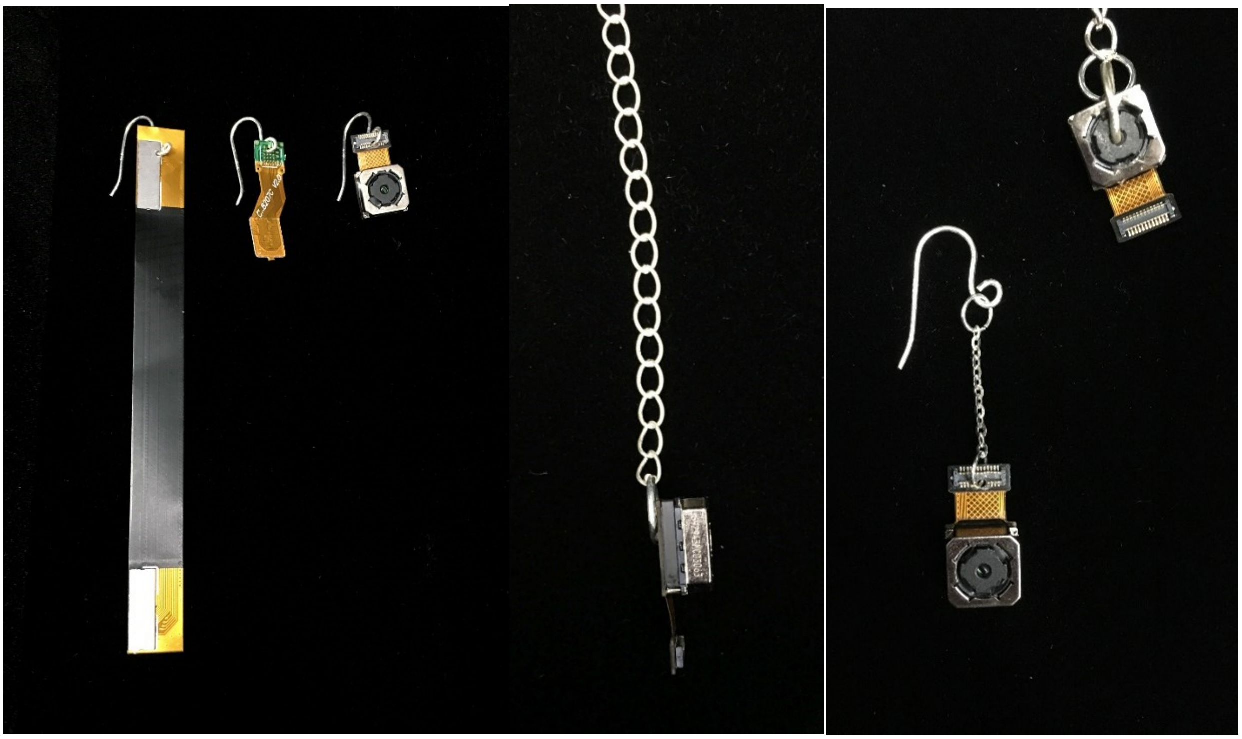


Figure 147 First earring experiments, July 25, 2017.

I had trouble making satisfying jewelry in Shenzhen. I was nervous, in a country I had never visited, without the local language, and still uncertain about being an artist working with electronics. I packed up the prototypes that I had made and my tools in a red jewelry box that I bought in the market and prepared for the return voyage. I picked the project up again in November 2019. With some new experience and confidence gained along the way, I completed the waterfall necklace design that had first inspired me to make jewelry with cameras. I call the finished piece Camera Collier.

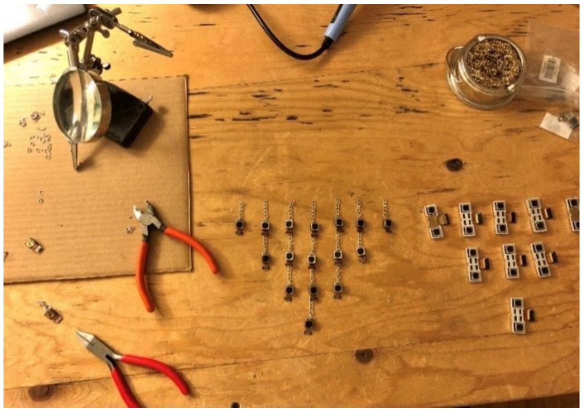


Figure 148 Camera Collier workspace, November 28, 2019.

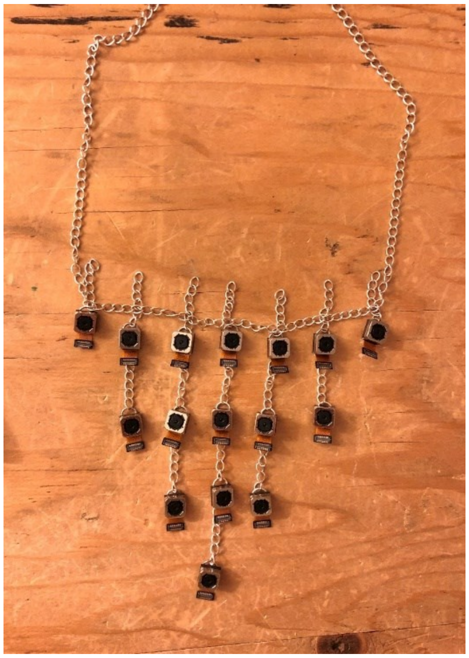


Figure 149 Camera Collier waterfall necklace, November 28, 2019.

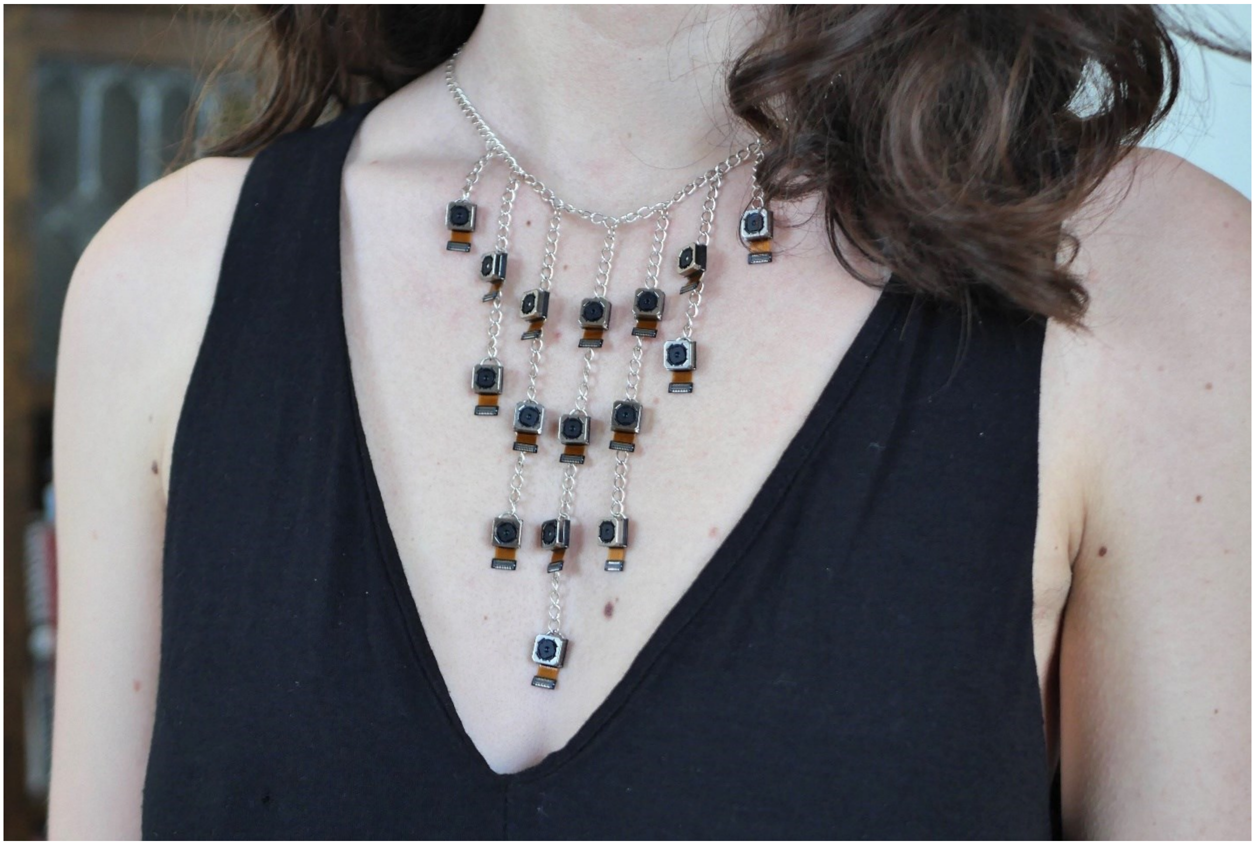


Figure 150 Alice wearing the Camera Collier, November 28, 2019.

The main lesson I drew from both of these projects was actually more about confidence and the artistic process than the supply chain or making. In both projects, I hesitated profoundly because I felt out of my depth. Couture and jewelry making were practices I had never attempted before, and I chose to take them up in a foreign land where I was on my own. I took to them bravely, but when faced with challenges in making, I doubted myself. I watched YouTube videos to pick jewelry making tools and to learn the basic techniques of attaching pieces together. I wasted some time doubting whether it was responsible to try something new while on a funded trip. Upon my return, I put the projects aside and didn't return to them for over a year. In retrospect, the projects were good, and I *was* able to produce an aesthetically interesting piece without any extra equipment or training—only calmness. In future, I hope to put more trust in myself, even if I do not have qualifications or people supporting the specifics of my project, because I learned that I can turn out a good result even in such circumstances. For the necklace, I worried that soldering jump loops to the backs of the cameras was not a good approach, because one would not want solder against one's skin in real jewelry. When I picked the project up again in 2019, I knew that it was okay to make a prototype that looks correct, but might not meet the full standards professional jewelry makers achieve. If I create a convincing looks-like prototype, then I will have something physical in hand to show a professional, who will then have an easier time seeing the purpose of the piece—and who will be able to offer expert advice to solve whatever questions I cannot answer alone. Next time, if a project becomes too daunting, I hope I will have the courage to return to it in shorter order, and to exhibit my work online or in person, no matter what I fear critics might say.

Textiles

Upon my return to Montreal in late summer 2017, I leapt headfirst into fashion. Inspired by my garment and jewelry experiments in China, I decided to dive into the details and learn firsthand how to work with the materials of apparel design. In the following months, I redoubled my efforts to become conversant with digital embroidery, and I took my first steps towards designing and assembling my own garments from scratch with an immersive experience in Concordia's Theatre Department Costume Shop.

Embroidery

I learned to machine embroider both at the Fab Lab du PEC and at Milieux's Textiles & Materiality cluster (T&M), where I am a research member. At the Fab Lab I had access to a small Janome embroidery machine. I had learned the basics of embroidery at the Lab when I started working there in the fall of 2016, but difficulty using the machine had frustrated my efforts.

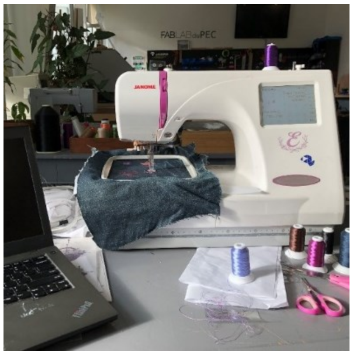


Figure 151 Embroidery machine at the Fab Lab du PEC, December 10, 2017.

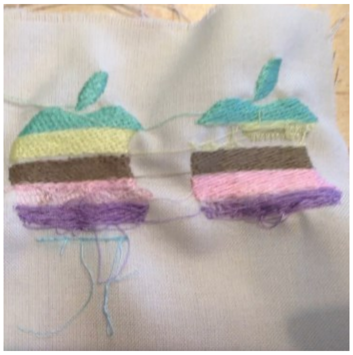


Figure 152 Failed first embroidery test on Janome machine, November 5, 2016.

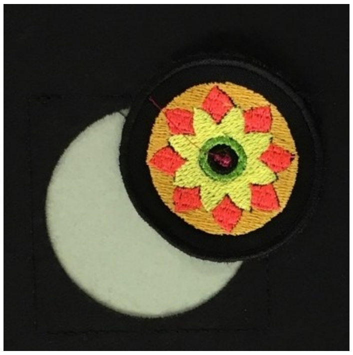


Figure 153 The first patch I made with Geneviève's tutelage, June 14, 2017.

With the help of digital embroidery technician Geneviève Moisan at Milieux, I learned superior techniques on more finely tuned machines. In a series of workshops and a few 1-on-1 sessions, Geneviève taught me and a few other members of the cluster how to make patches and do machine laying. At T&M I had occasional access to their fabulous twelve colour industrial Tajima embroidery and laying machine and more regular access to an excellent six colour Brother embroidery machine. I created a number of tests to get a feel for both of them.

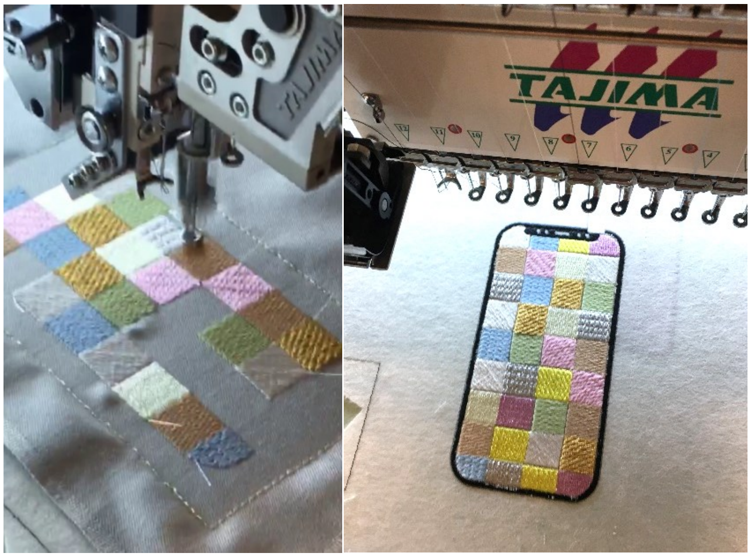


Figure 154 iPhone App embroidery on Tajima machine September 5, 2017.

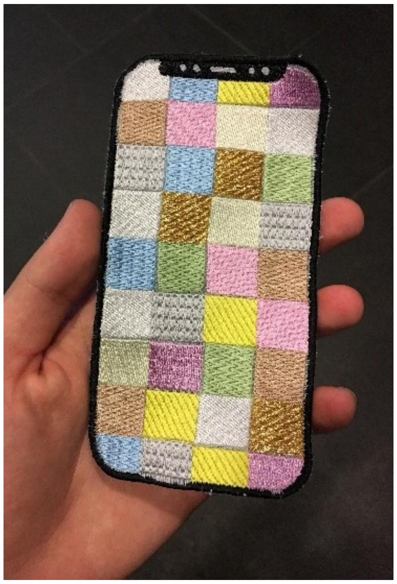


Figure 155 iPhone App embroidery test, September 13, 2017.

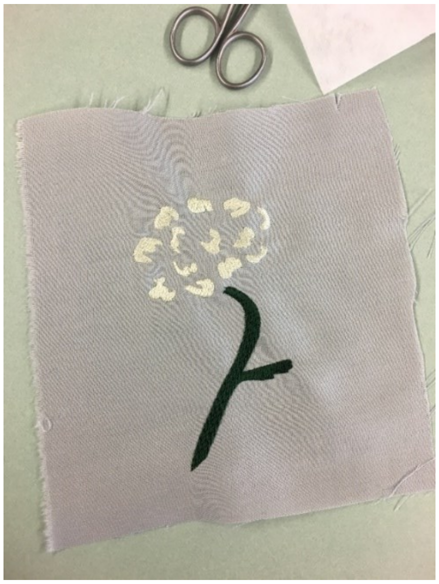


Figure 156 Embroidery of an oil pastel illustration, September 7, 2017.

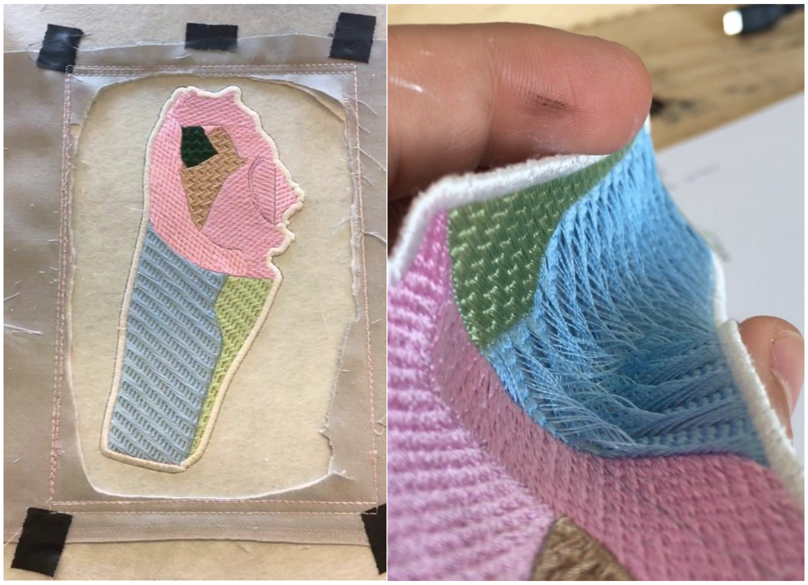


Figure 157 Pastel Falafel, September 7, 2017.

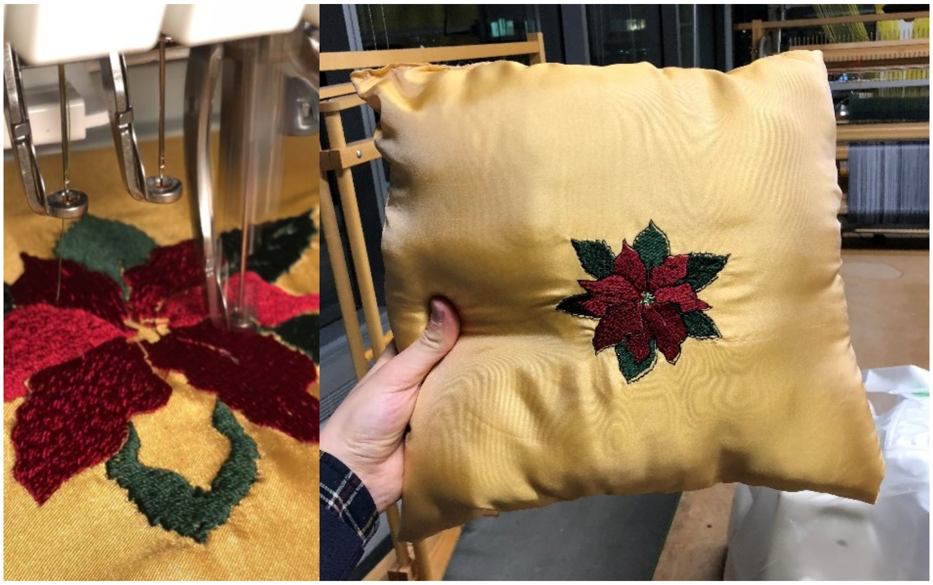


Figure 158 Poinsettia cushion embroidery, December 25, 2017.

During this learning process, I became much more intimately acquainted with the material properties of fabric and thread. I learned the basics: that one should not embroider twice over the same area, how to create a Steil border so that a patch does not fray, and how to use different kinds of backing material in different situations. I also observed subtleties in working with embroidery that I would never have learned from a textbook or YouTube video alone. While making the Pastel Burrito, I was happily surprised to see that the advanced embroidery fill patterns in the Tajima software created an area of embroidery that had a 3D effect when bent inward upon itself. Thanks to their angle and embroidery style, the blue threads that are merely textural when laid flat jump to life when the fabric is curled inward upon itself. I have not yet used this technique elsewhere, but I suspect that I might in the future. I was also very happy to discover the appliqué technique, in which one traps a piece of cloth under a stitch, then cuts away the excess. It echoes collage in some ways, and although I have not yet used it in another project, either, I think it is a fascinating way to bring diverse patterns and textures to a work. Having just returned from China, I was also amused to see that the rolls of sequins for the laying machine, which I have not yet tried, are stored in the same type of tape-and-reel device as SMT components are stored for robotic pick-and-place machines.

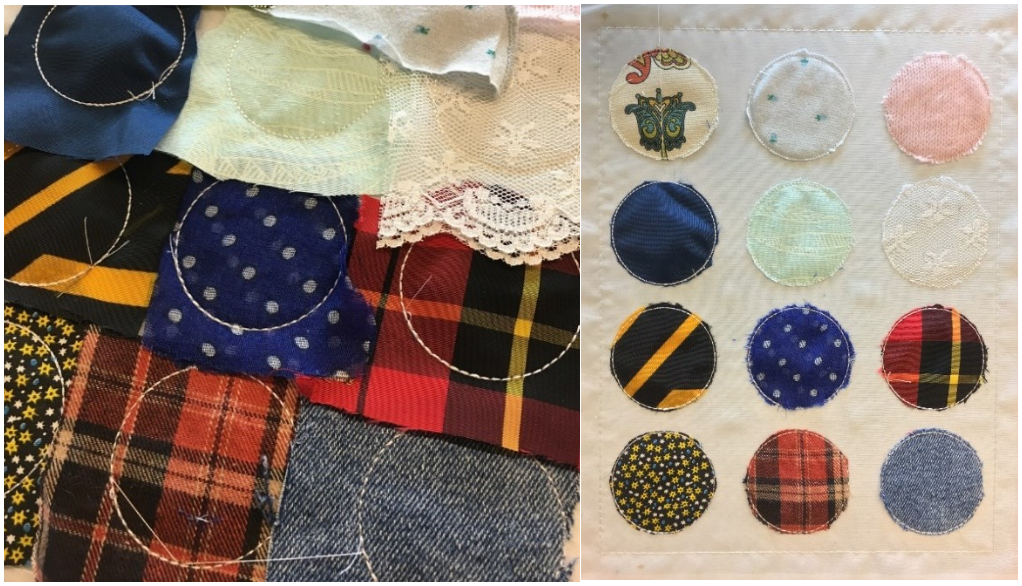


Figure 159 Applique test, September 17, 2017.

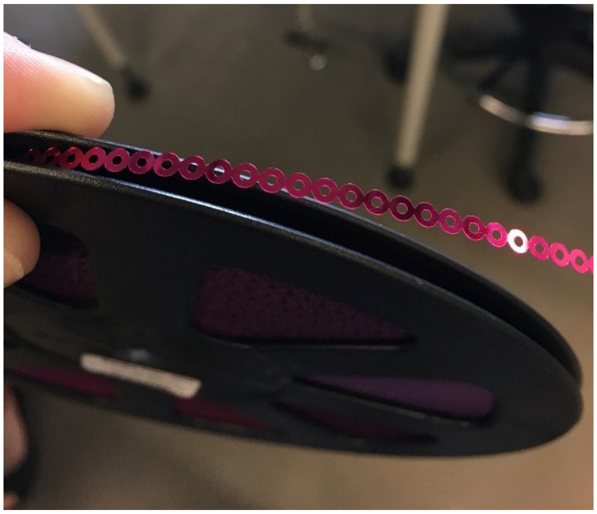


Figure 160 Sequins stored in a reel, July 13, 2017.

Over the following months and years, I integrated embroidery into my making practice. In one such project, I drew inspiration from media nail art tutorials and created a baseball cap embroidery with a hand and a vibrant coloured nails. In another, I reflected upon the feeling of distraction and disarray in my research project by embroidering a vectorized version of the handwritten word "Focus" across the chest of a white t-shirt.

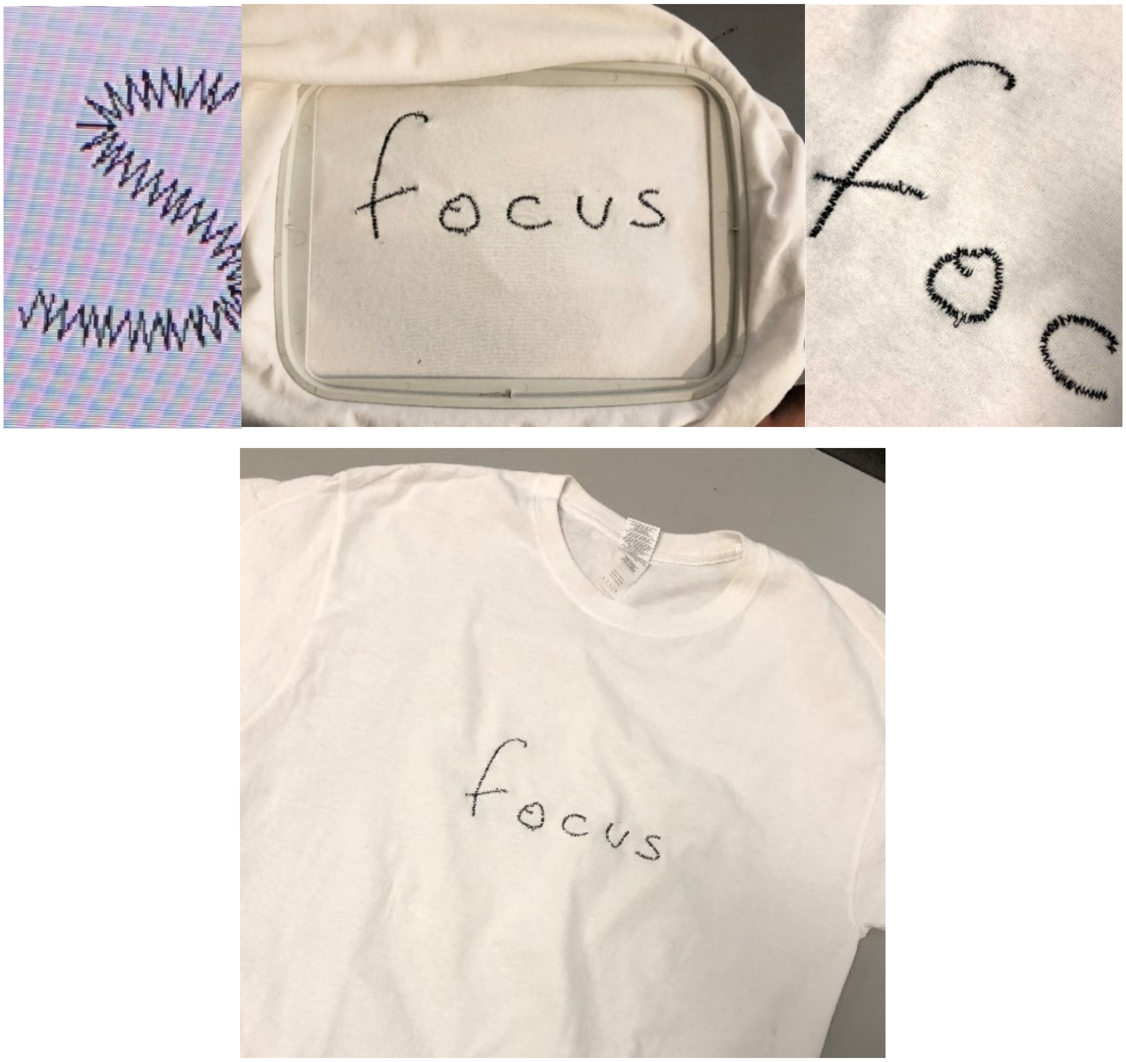


Figure 161 Focus shirt, December 29, 2019.

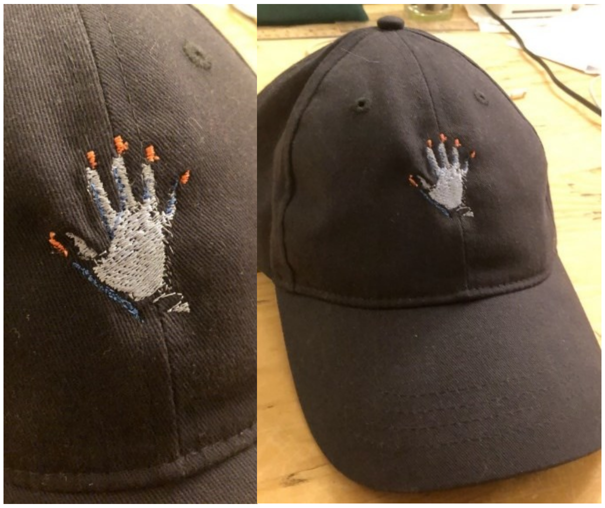


Figure 162 Nails hat, March 23, 2019.

The embroidery techniques that I learned also fed back into my ewaste club design project. Inspired by the golden Kapton-covered flex cables of the Raspberry Pi Camera and those that I had found in China, I attempted to embroider my own flex cable-like accessory. I clamped a plastic snap onto it, and I imagined it might snap onto other aspects of the project. The embroidery had a beautiful lustre on the golden side, but I had great difficulty aligning the black and yellow lines on the reverse side, which were meant to evoke the conductive traces in a flex cable. Each time the machine stopped and changed thread colours, it lost alignment. Ultimately the Textiles & Materiality cluster's embroidery fees were a bit too expensive for such dense embroidery, so I put the experiment aside. Nevertheless, the resulting sketch has a fun honeybee quality.

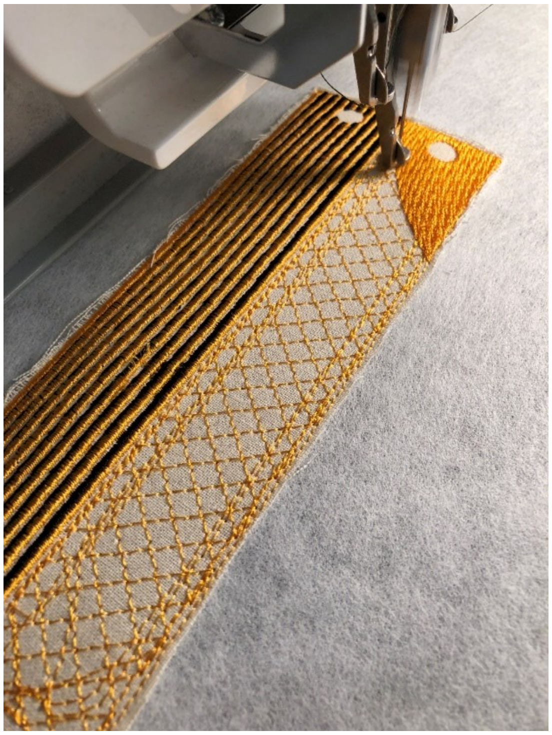


Figure 163 Flex Cable embroidery for ewaste, February 17, 2018.

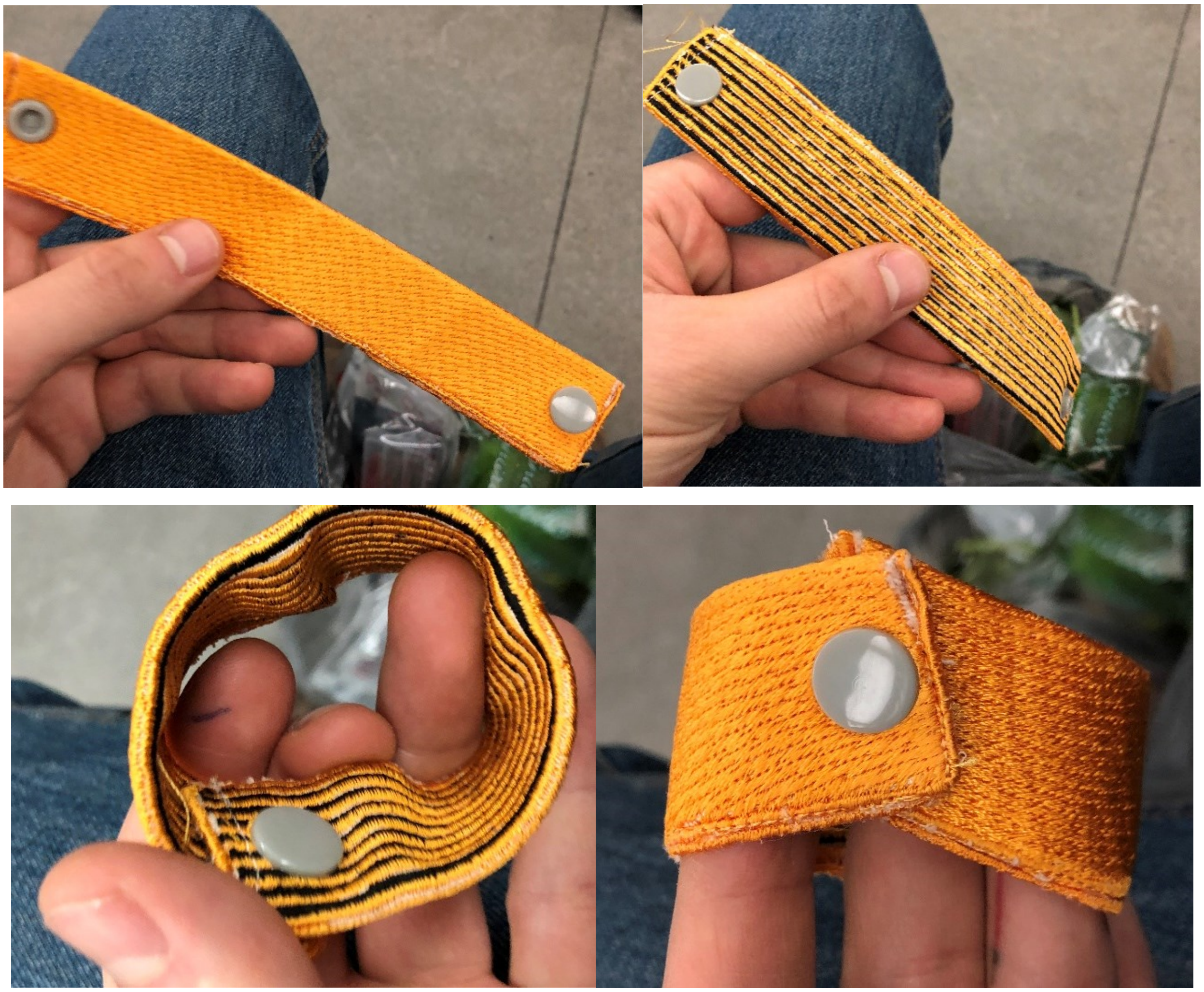


Figure 164 Flex Cable embroidery for ewaste, February 17, 2018.

Clothes Making

In late 2017, I began learning to design and construct my own garments. In November, my research supervisor Ana Cappelluto introduced me to Laura Acosta, the director of the Concordia Theatre Department's costume shop. In the costume shop, Laura and her assistants worked with student theatre directors and costume designers to construct or adapt garments for the stage. Laura generously taught me to sew with an industrial sewing machine and how to use an industrial serger to finish seams. She also showed me the basics of measuring, marking, and constructing garments. To get started, Laura suggested that I first replicate a children's shirt. I set to work and replicated the original garment—albeit roughly.

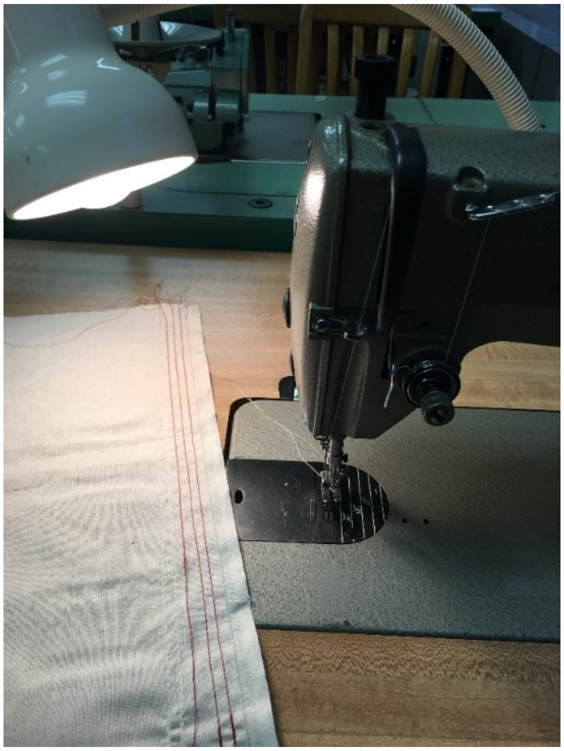


Figure 165 Industrial sewing machine, November 16, 2017.

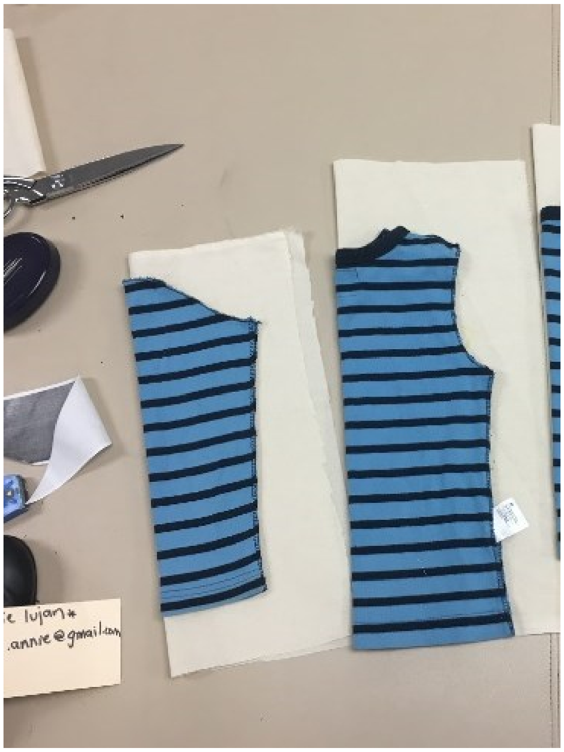


Figure 166 A children's shirt split into four pieces and prepared for tracing onto muslin, November 16, 2017.

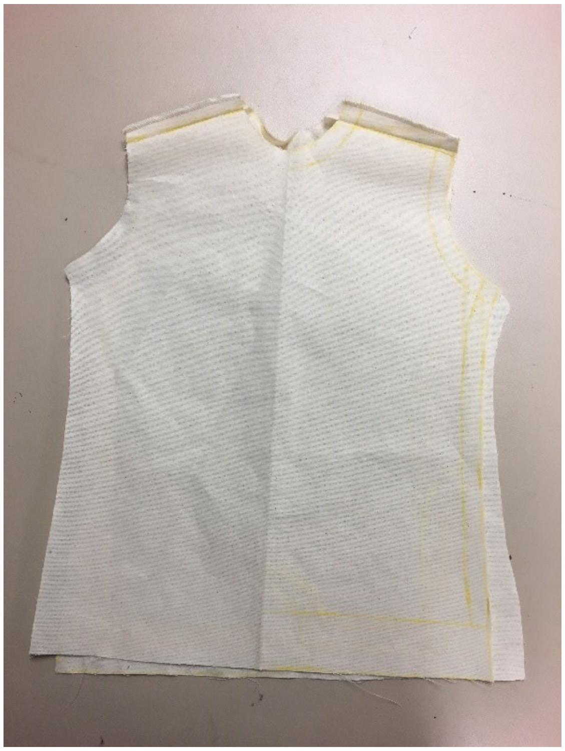


Figure 167 Torso of the children's shirt, November 16, 2017.

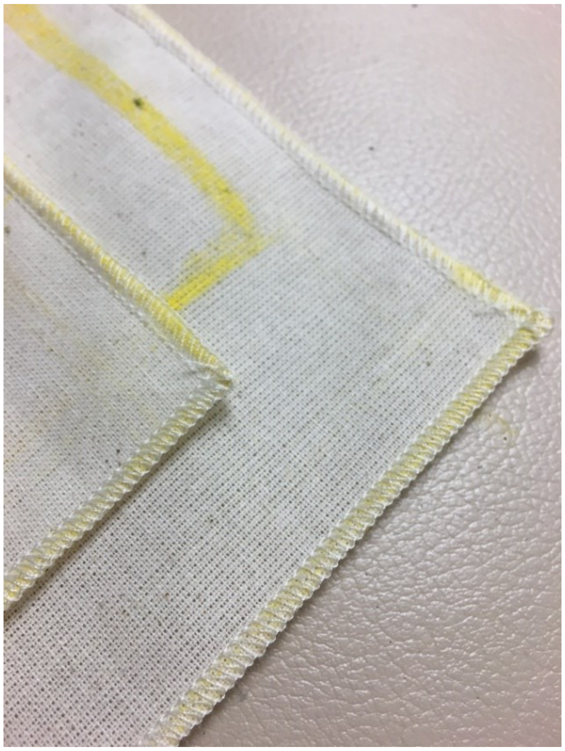


Figure 168 Detail of serged pieces, Novemeber 16, 2017.

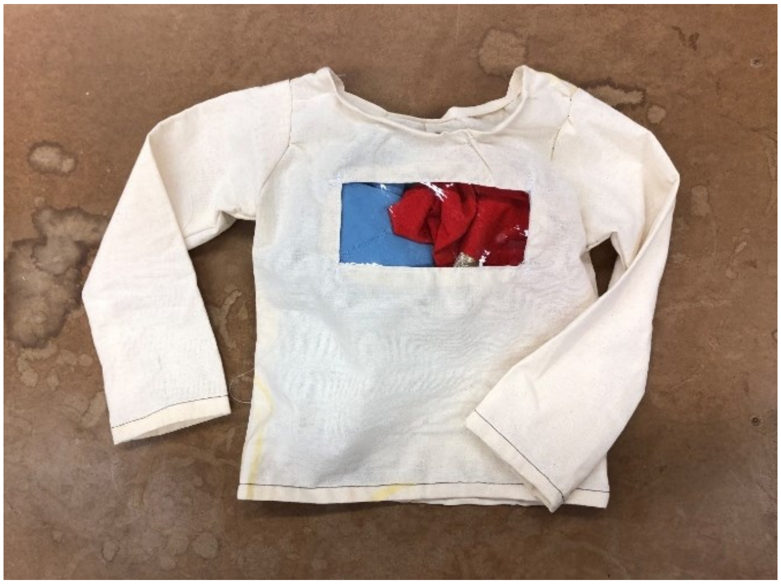


Figure 169 Replica children's shirt, which I later cut a hole in to prototype a "shirt with a window," February 20, 2018.

Once I had a handle on the basics, Laura pushed me to make a garment from scratch. I decided to make a garment to house the same Raspberry Pi camera platform that I had already put in the Camera Glasses and Camera Jacket. I found initial inspiration in one of the garment's the costume team had prepared for an upcoming production of The Tempest. I found myself inclined to drape the garment, rather than start with a drawing. Having never worked with cloth at human scale, I felt unable to conceive in drawings of how the cloth would fall. I draped the cloth over a manikin and iterated over the design a few times, folding and pinning as I went.

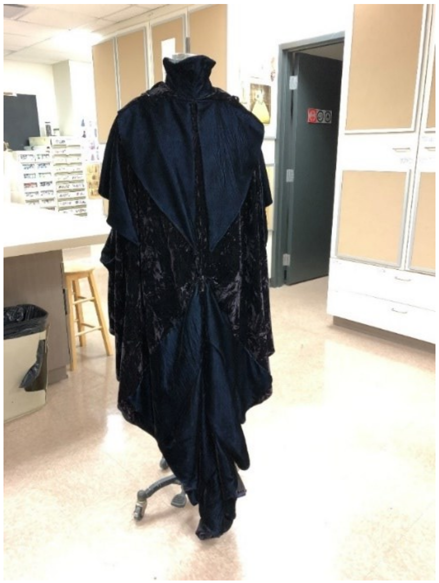


Figure 170 Inspirational garment from The Tempest, December 5, 2017.

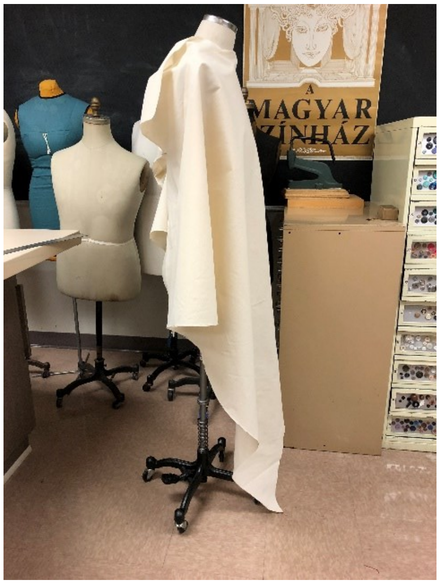


Figure 171 Draped fabric, December 5, 2017.

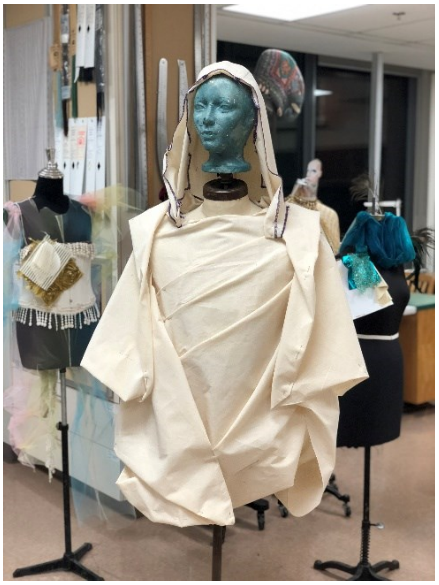


Figure 172 Draping experiment, December 5, 2017.

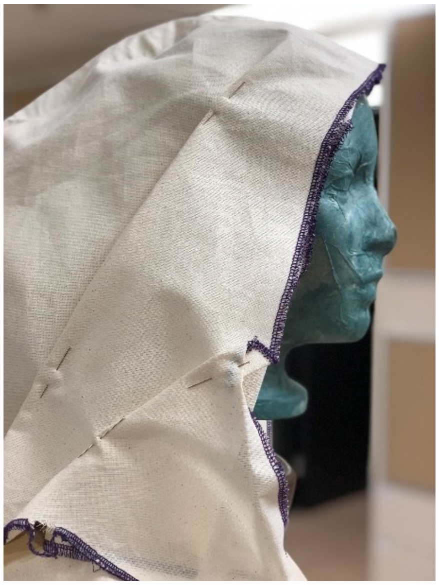


Figure 173 Hood experiment, December 5, 2017.

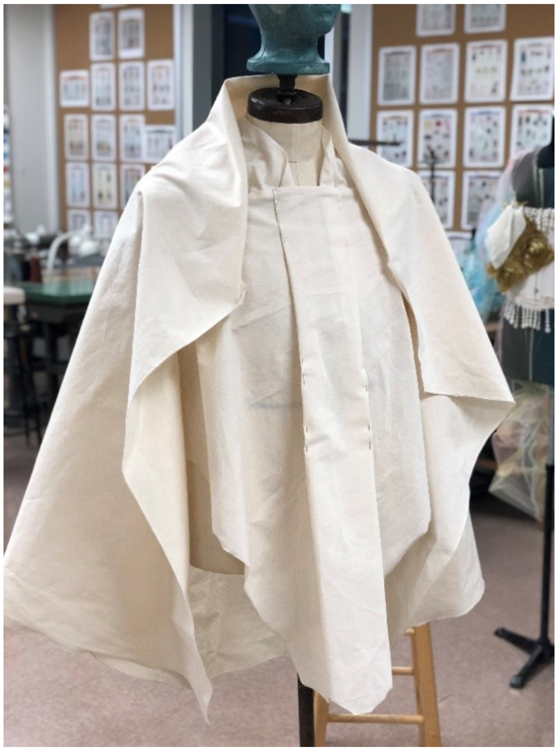


Figure 174 Draping experiments, December 8, 2017.

I purchased special waterproof fabric at a store on Rue Chabanel in Montreal, and with an encouraging kick from Laura, I took a risk, cut into the material, and began sewing my first garment together.

Figure 175 Cloth shopping in garment districts of Rue Chabanel and Rue Saint-Hubert, December 14, 2017.

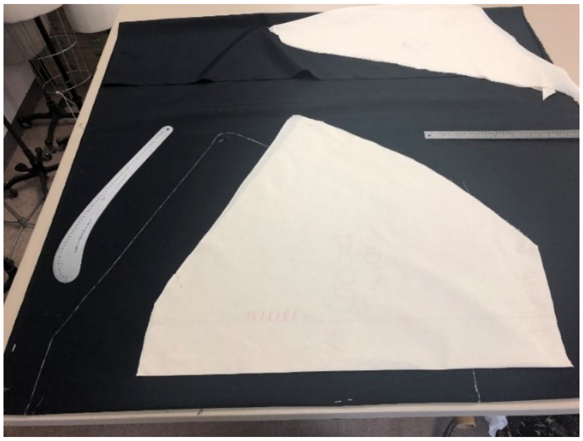


Figure 176 Cutting the waterproof material according to a muslin pattern, December 13, 2017.

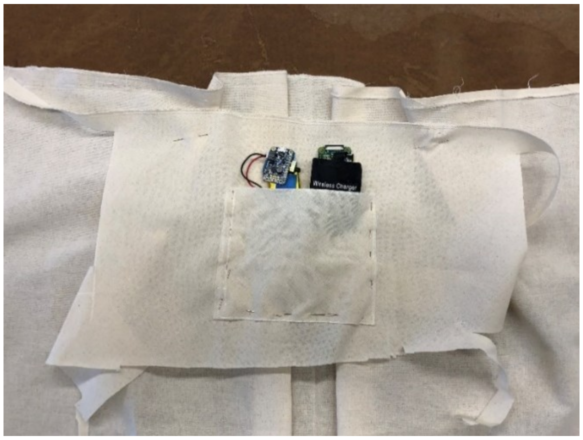


Figure 177 Pocket sketches, December 19, 2017.

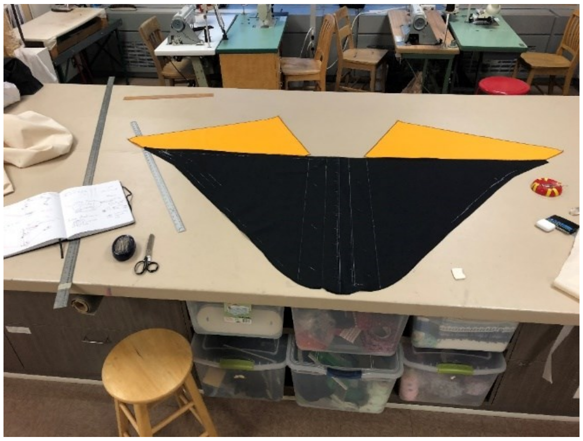


Figure 178 Measuring and assembling the pieces, December 13, 2017.

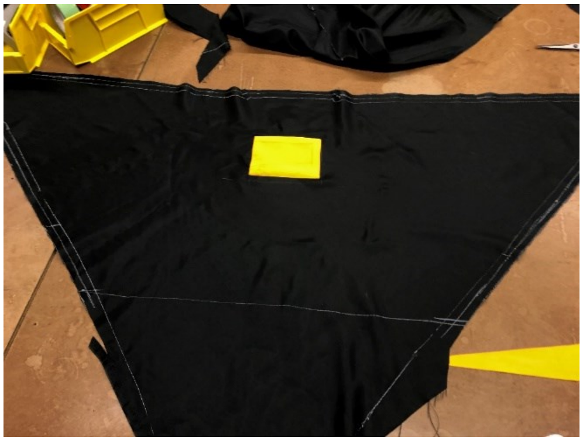


Figure 179 Computer pocket, December 19, 2017.

The final garment is a two-piece waterproof cape with an interior pocket for a computer near the chest. Both pieces are lined with black cloth, and the shoulders of the front piece are made of the same waterproof material in yellow. The garment is worn by buttoning the front piece first, then draping the rear cape over the shoulders.

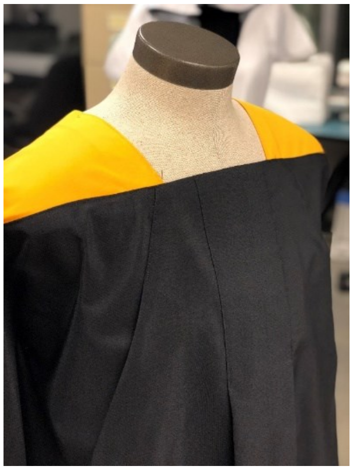


Figure 180 Front piece, detail, December 20, 2017.

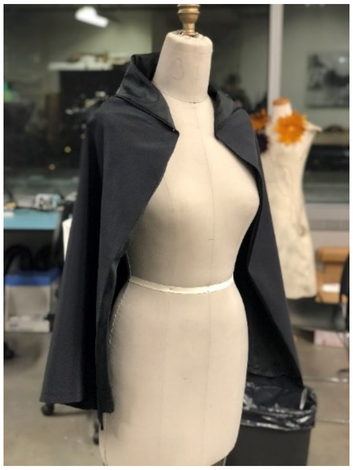


Figure 181 Rear piece, December 15, 2017.

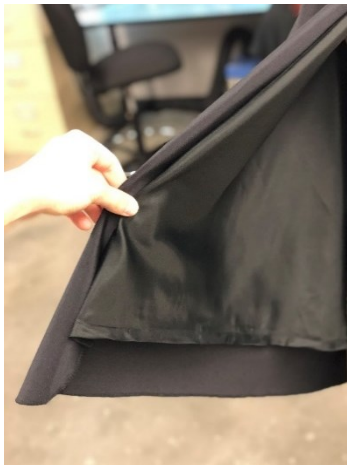


Figure 182 Inner lining, December 15, 2017.

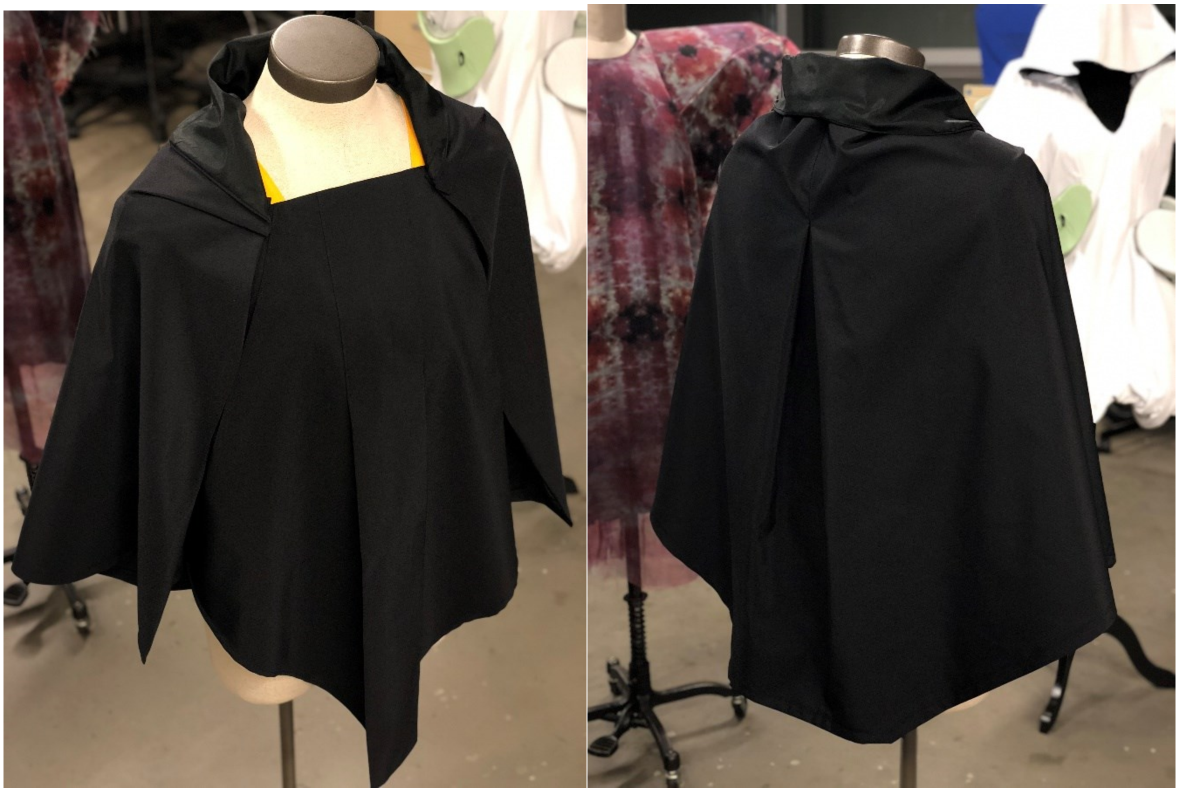


Figure 183 Finished garment, December 20, 2017.

Making my first article of clothing allowed me to break through the initial fear of making a mistake and opened the world of fashion to me as a viable route for expressing my design ideas. Although the final Computer Cape is a bit melodramatic, it nevertheless is a real working piece of clothing that I constructed from start to finish with my two hands. I think the choice of colour is perhaps the most striking aspect of its design, and the excessive use of pleats is suitable for a first work, but something to temper in future endeavors. The camera pocket is not particularly integrated into the design, but its presence maintains the connection between this garment and the ewaste project.

While working in the costume shop, I pursued a number of side experiments. While fabric shopping, I stumbled upon a transparent vinyl, which I managed to sew with some difficulty. I loved the look of the colourful thread piercing through the pliable clear vinyl. I did some sewing samples and began to experiment with stuffing the clear vinyl with synthetic padding material. I created structural triangles, which I imagined could be connected together to create a polygonal jacket reminiscent of 3D computer representations, whose surfaces are often broken into triangles. I created a playful variation on this same concept when I made a cloud toy, composed of clear vinyl, black thread, and synthetic padding. A relative loved the look and gave it to her young daughter, who I am told loves it very much. I introduced my burgeoning embroidery practice into the fold when I experimented with creating embroidered pockets, possibly for holding small computers. Finally, I used my new sewing skills to make a handful of denim patches. I hand drew a few flowers, vectorized and embroidered them onto denim, then cut them apart and sewed them onto Velcro. I sewed matching Velcro pieces onto my backpack, and stuck them in place. I left the denim edges rough intentionally to play with giving the pieces a feeling of imperfect vitality.

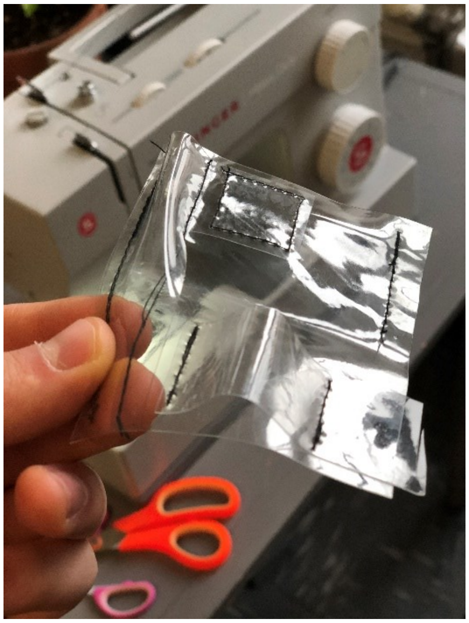


Figure 184 Experiments sewing clear vinyl inspired a handful of other projects, November 25, 2017.

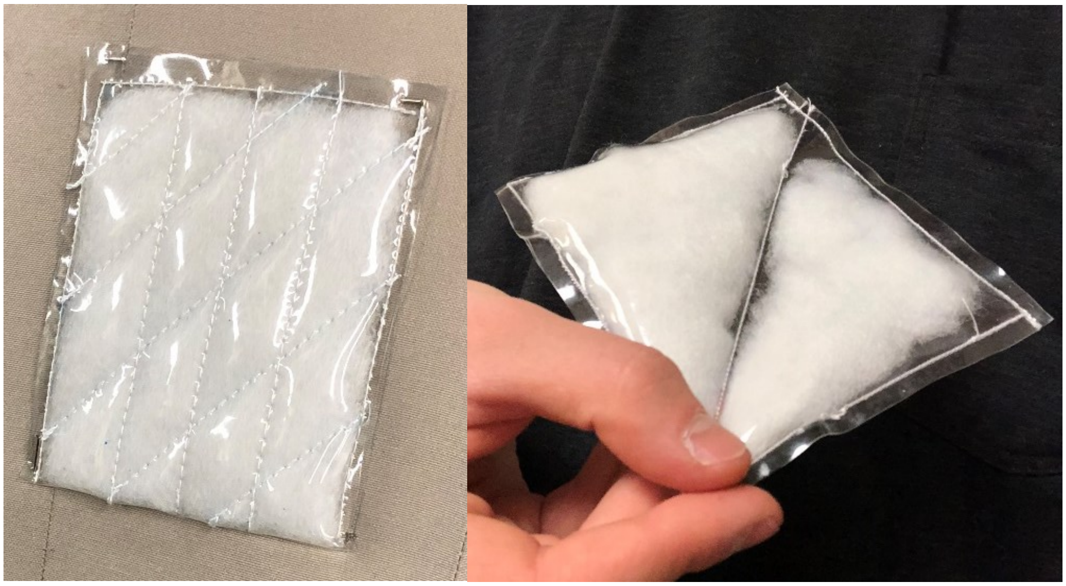


Figure 185 Padding experiments, November 25, 2017.

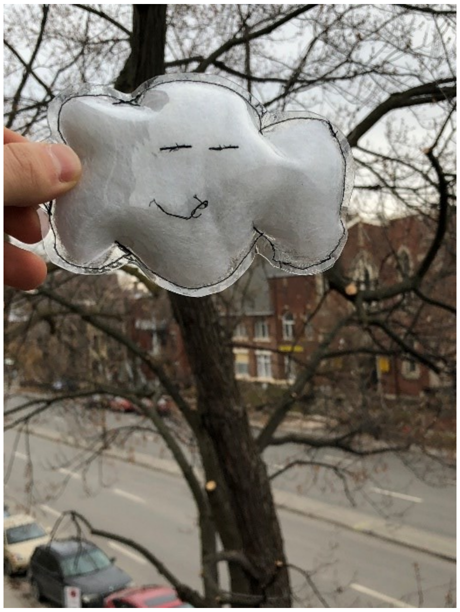
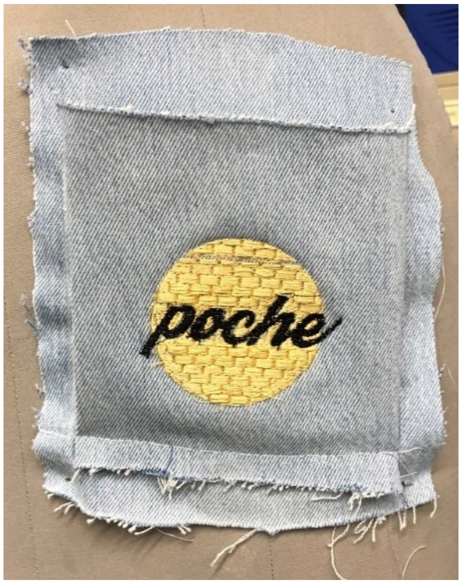


Figure 186 Cloud toy, December 2, 2017.



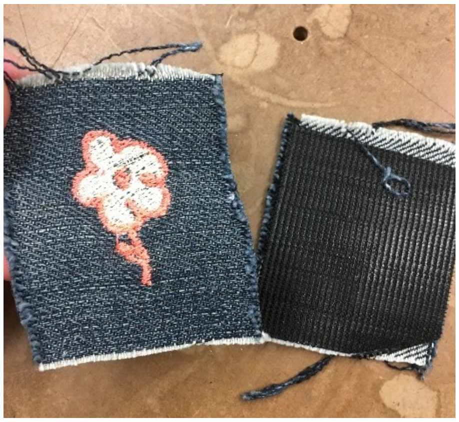


Figure 188 Flower patch, front and back, December 18, 2017.

Figure 187 Embroidered computer pocket sketch, December 5, 2017.

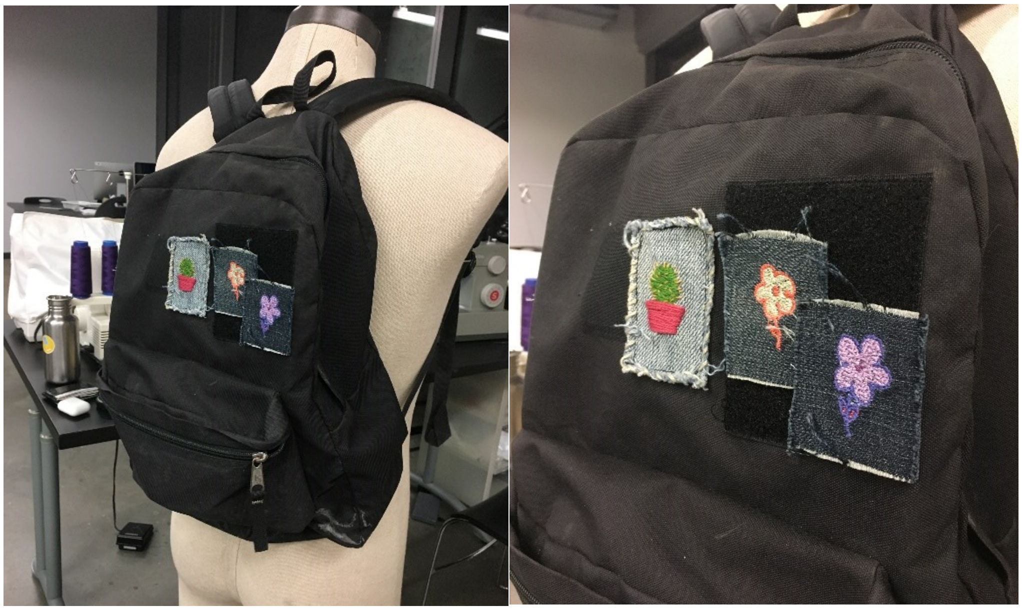


Figure 189 Flower patches on backpack, December 18, 2017.

Camera Patch

In the course of finishing the cape garment, I accidentally experimented my way into the design for Camera Patch, the spiritual successor to my Camera Glasses. Once I had created the pocket in the cape, I made a few rough sketch pouches to think through how I might hold the Pi computer, battery, and accessories together inside the cape's pocket. Playing with these pouch designs one night in the T&M studio, I realized that the pouch, alone, might be an interesting design direction.

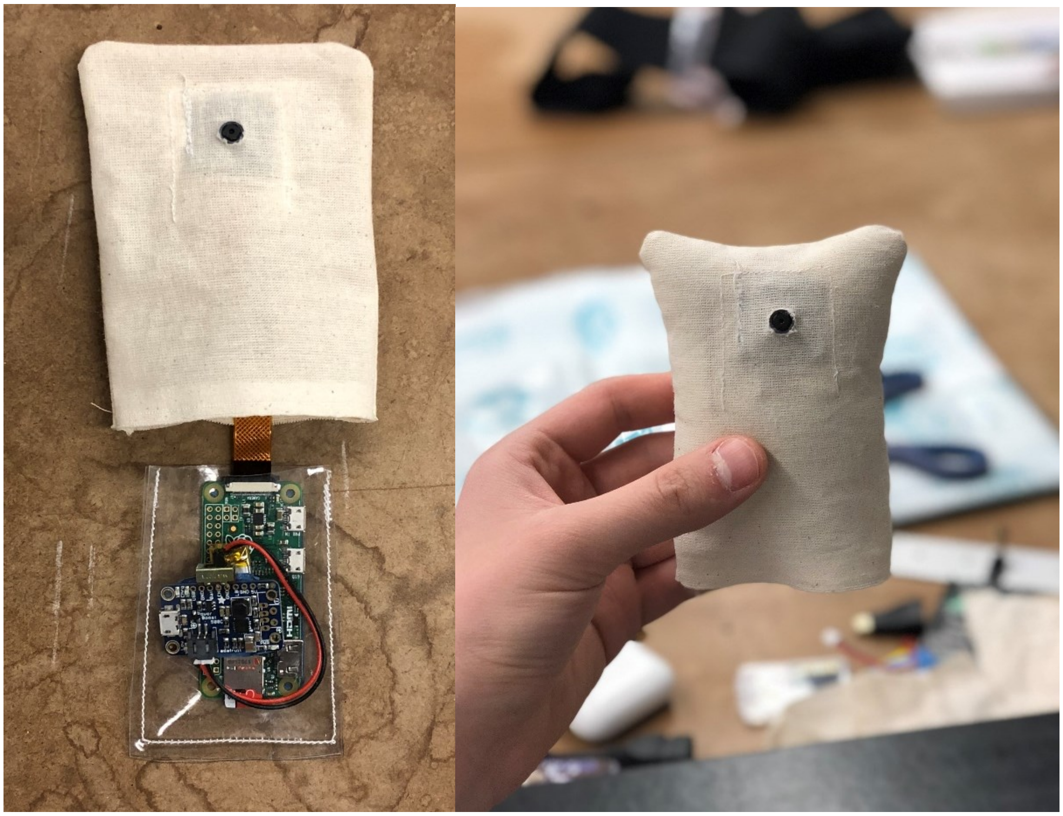


Figure 190 Camera pouch sketches, December 26, 2017.

The initial version of Camera Patch was a simple looks-like prototype. I placed the components for the camera hardware stack I had developed during the Camera Glasses project into a sewn-shut transparent vinyl pouch. Camera Patch's see-through electronics design recalls Daniel Weil's 1981 piece Radio in a Bag, in which the artist placed the components of a working transistor radio in a transparent PVC bag [323], [324, p. 38]. The choice of transparent vinyl drew from the prior experiments with structural triangles filled with padding material. To this, I added another recent inspiration to complete the idea. Having created the flower patches only two weeks prior, I was quick to draw upon Velcro as a fixing material. I attached strips of adhesive Velcro to the back of the package and placed it on my knapsack. Immediately, I knew that I had found something special.

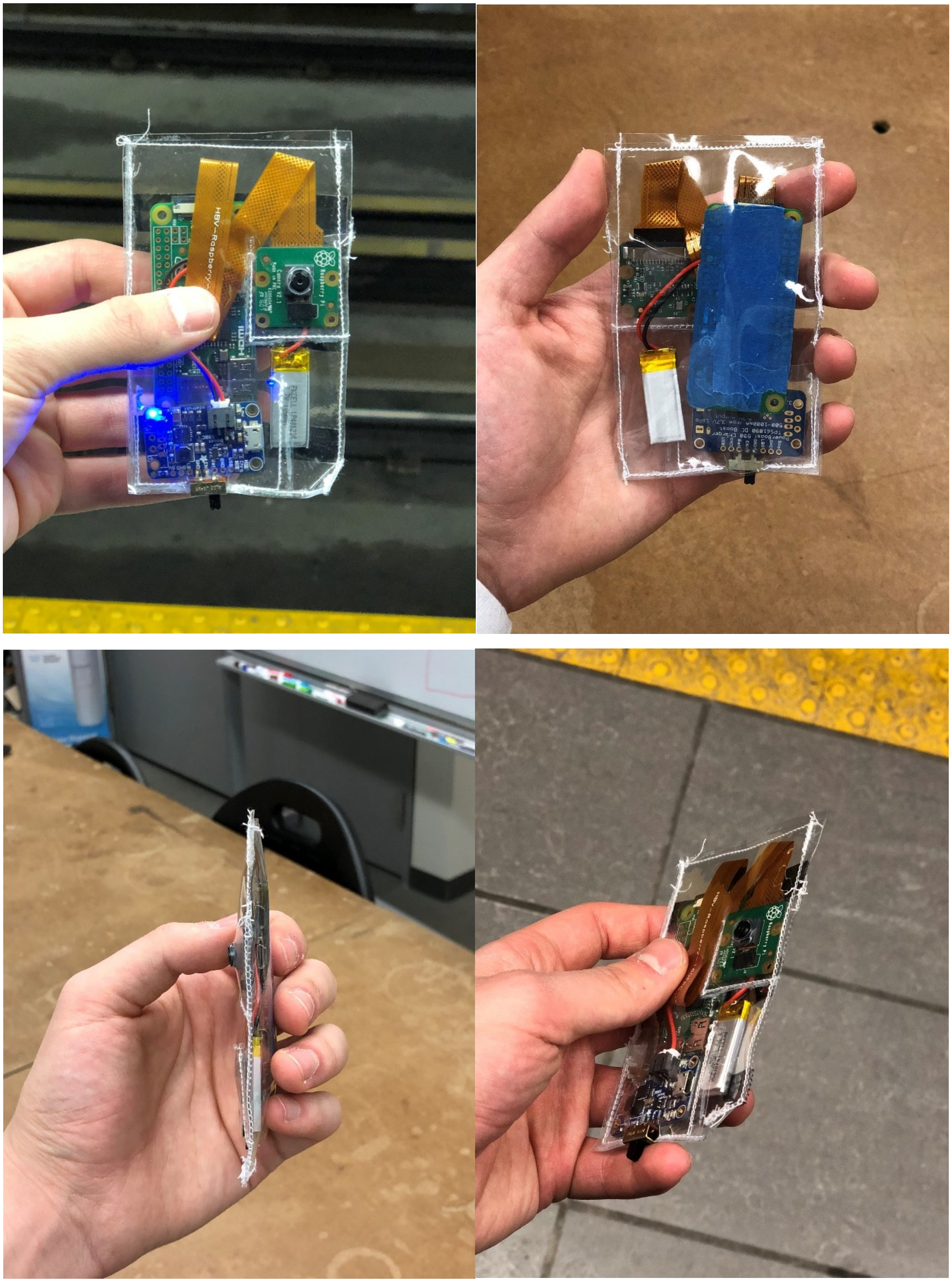


Figure 191 Camera patch looks-like prototype, December 26, 2017.

I was excited by this new design direction because it indirectly solved the form factor frustrations I had had with the glasses-style enclosure. Instead of suffering the relative bulkiness of the Pi Zero in the context of glasses frames, the Camera Patch could flex the thinness of the total package (the z dimension), at the acceptable expense of the vertical and horizontal area (x and y dimensions). Suddenly the camera looked like a supply chain trading card. Although it was only a looks-like prototype, I could see the path forward to the intersection of the ewaste aesthetics I had been exploring through fashion objects and the more traditional functional prototypes I had made to this point.

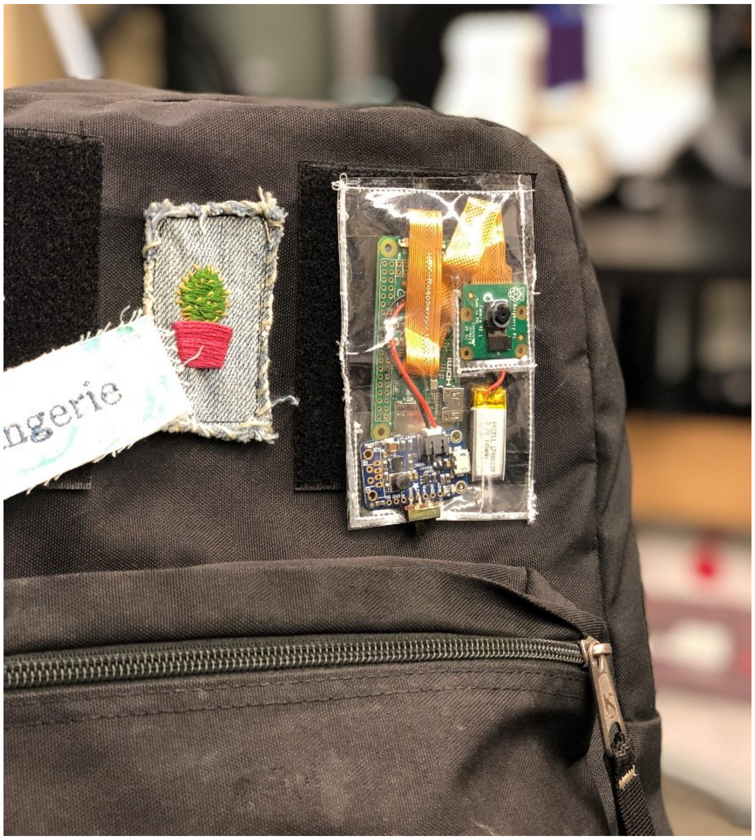


Figure 192 Camera Patch looks-like prototype on knapsack, December 26, 2017.

I set to work developing a basic application for the Camera Patch. I envisioned this first application as a proof of concept. Given that it would be affixed to one's backpack, for example, I decided to write an application that would push a photo from the Pi to a website every five seconds, over a tethered Wi-Fi connection to my phone's hotspot. In effect it was like an ambient live stream; less information than a video feed and without audio, but enough visual stimulus to possibly expand upon later.

To write the Camera Patch software, I taught myself to use Google Cloud Shell and Functions, as well as to write and execute all of this code in Python and Shell. At the given interval, the camera would snap a picture and push it to a Google Cloud Function, which would add it to a Firebase database and key value pair store. When a client loaded the webpage, they would connect to a NodeJS server on a VPS, which would, once loaded on the client's browser, fetch the image stream from Firebase. Once I had it running at on the stationary Pi 3, I transferred the code to the Pi Zero and tested it with friends and during a visit to the nearby grocery store. It worked well.

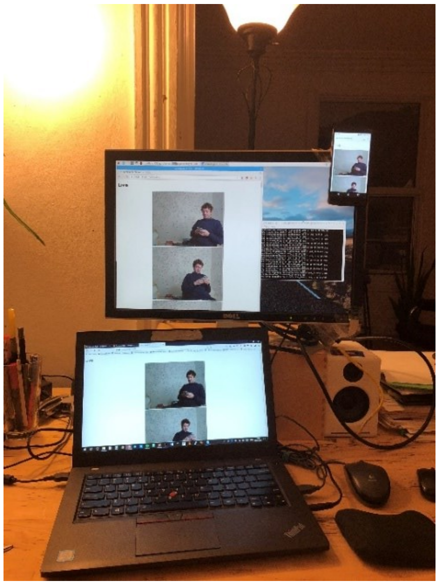


Figure 193 Development setup, January 8, 2018.

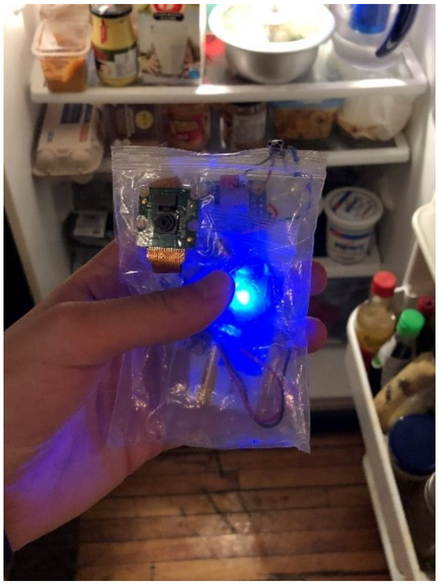


Figure 194 First Camera Patch working wireless test with Patch in a plastic baggie, January 12, 2018.

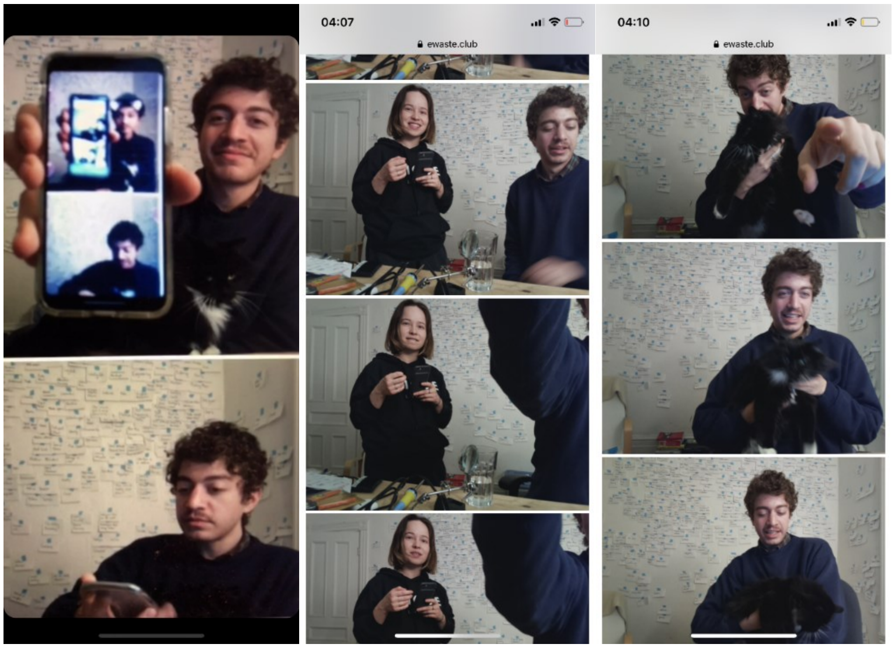


Figure 195 First images of the working application loaded in a mobile browser, January 12, 2018.

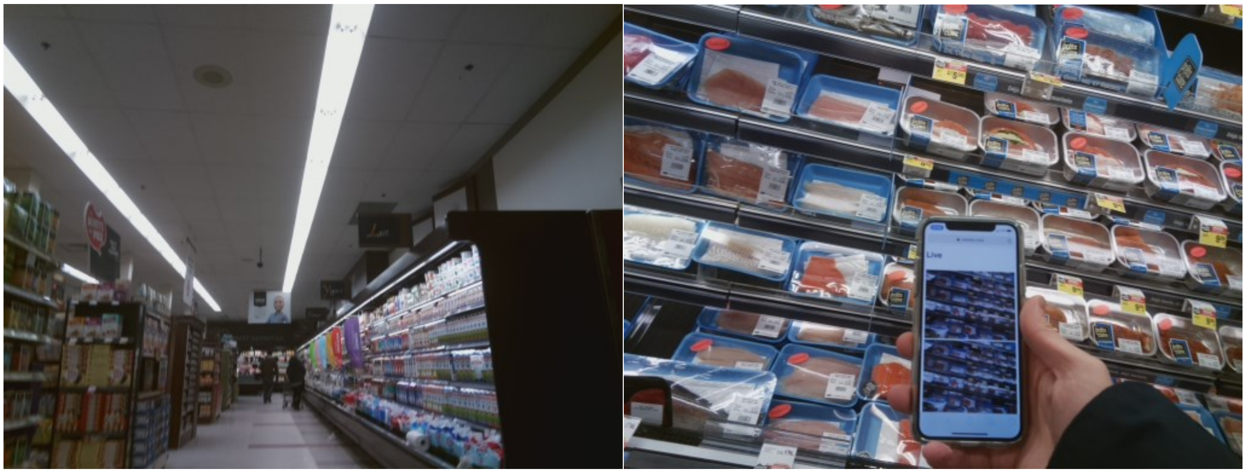


Figure 196 Outdoor testing: visit to the grocery store, January 18, 2018.

I constructed a new vinyl enclosure for the working Camera Patch that isolated the components to avoid accidental shorting. I attempted to sew pockets for each component into the interior of the Patch. Unfortunately, this added to the package's thickness, and made it less aesthetically satisfying. The vinyl was difficult to work with in a sewing machine, especially for detailed work, because of its sticky surface caught on the machine's foot. I attempted to thermoform transparent PETG plastic to fit the components using a laser cut wooden mold, but I had trouble fitting the parts together and abandoned that approach.

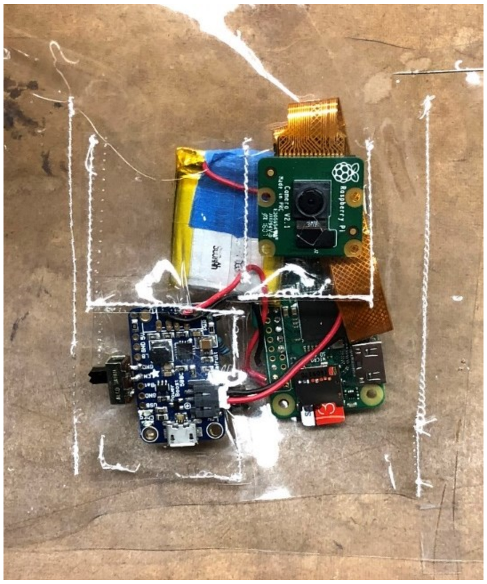


Figure 197 Development of the Camera Patch pocket system, January 12, 2018.

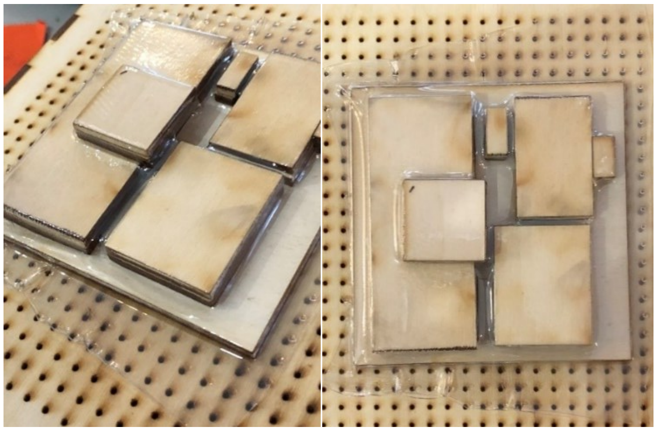


Figure 198 Thermoformed PETG tests with laser cut wooden mold, February 4, 2018.

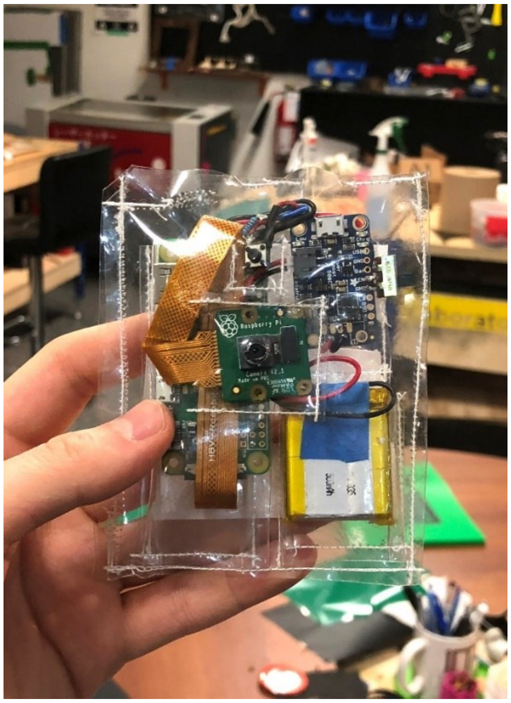


Figure 199 Assembled Camera Patch working prototype, January 21, 2018.

I celebrated the completion of the first working ewaste apparel product by embroidering an ewaste club patch. I designed the patch in Embird and embroidered it with the Brother machine in the T&M studio space. The patch features the phrase ewaste.club, which is the URL I used to host the prototypical Camera Patch photo feed. The ring of repeating text is divided by the Chinese Yuan symbol to represent the provenance of its constituent components as well as the thinking that inspired its creation. At the centre is a Camera patch shaped golden rectangle that represents the experimental project that I believe best represents the whole of this body of work.

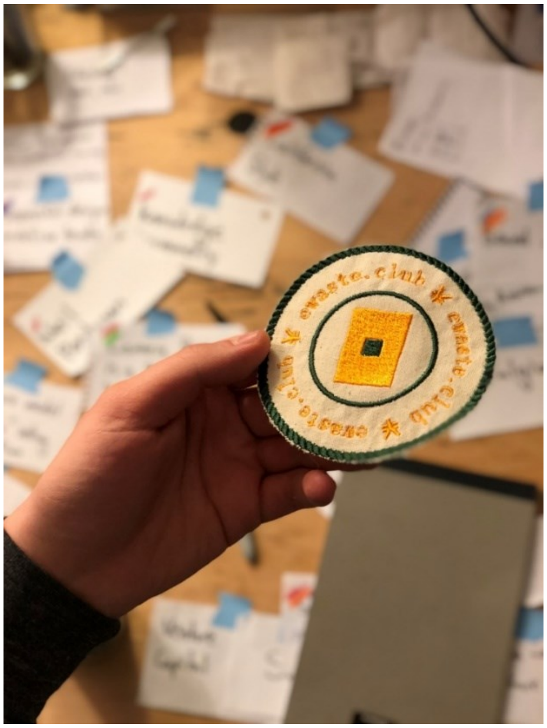


Figure 200 ewaste club patch photographed in front of the note cards I used to organize and write chapters one through five, February 16, 2018.

Camera Crown, Camera Tiara

Once liberated from the glasses product form, I began to explore alternative head mounted apparel that could better conceal the Raspberry Pi Zero and camera. In conversation with my supervisor Ana Cappelluto, I began to dream of a Live Streaming Camera Crown and Camera Tiara. I created a 3D printed model to test the idea of concealing a plastic skeleton underneath a cloth adornment. I sewed a denim headband that was inspired by Japanese apparel like the forehead protector (*hiitai-ate*) and the helmet scarf (*hachimaki*). I made two blue-and-gold cloth crowns that drew upon 11th and 12th century Slavic women's headwear [325]. I intended to embellish all of these designs with rich embroidery and accessories. Although I ultimately did not pursue these designs beyond a rudimentary non-functional 3D print and a handful of fabric sketches, I choose to include them here as I continue to believe it is an interesting form for computer apparel that I may return to in future work. The presence of these prototypes in the documentation enriches the reader's understanding of the breadth of my research exercises.

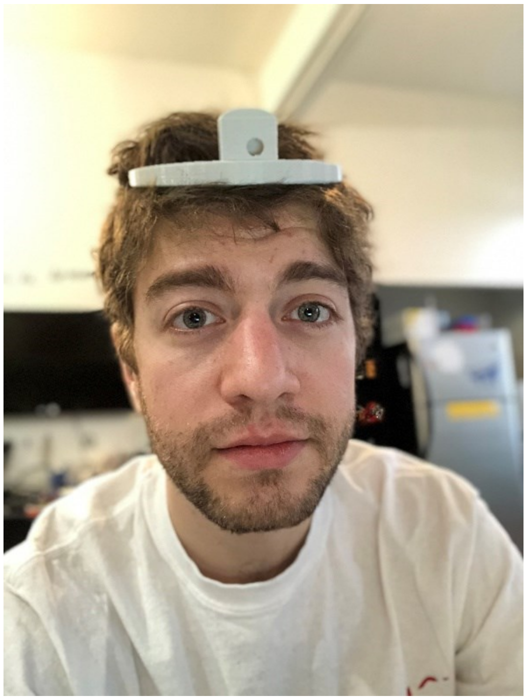


Figure 201 Camera Tiara 3D printed skeleton test, February 20, 2018.

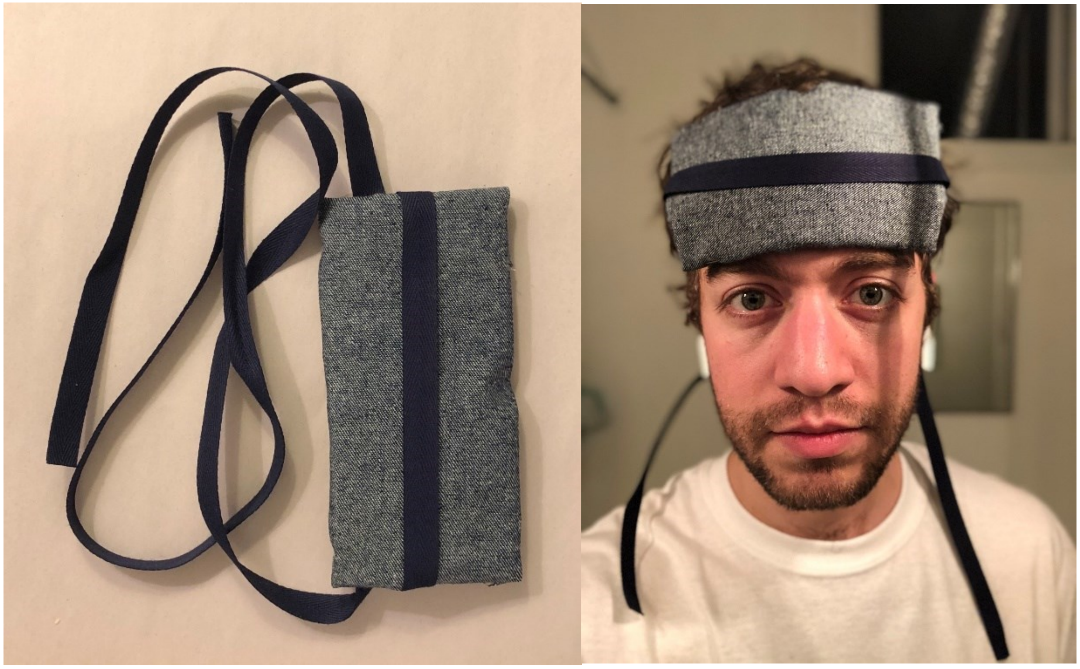


Figure 202 Denim headband, February 22 and 23, 2018.

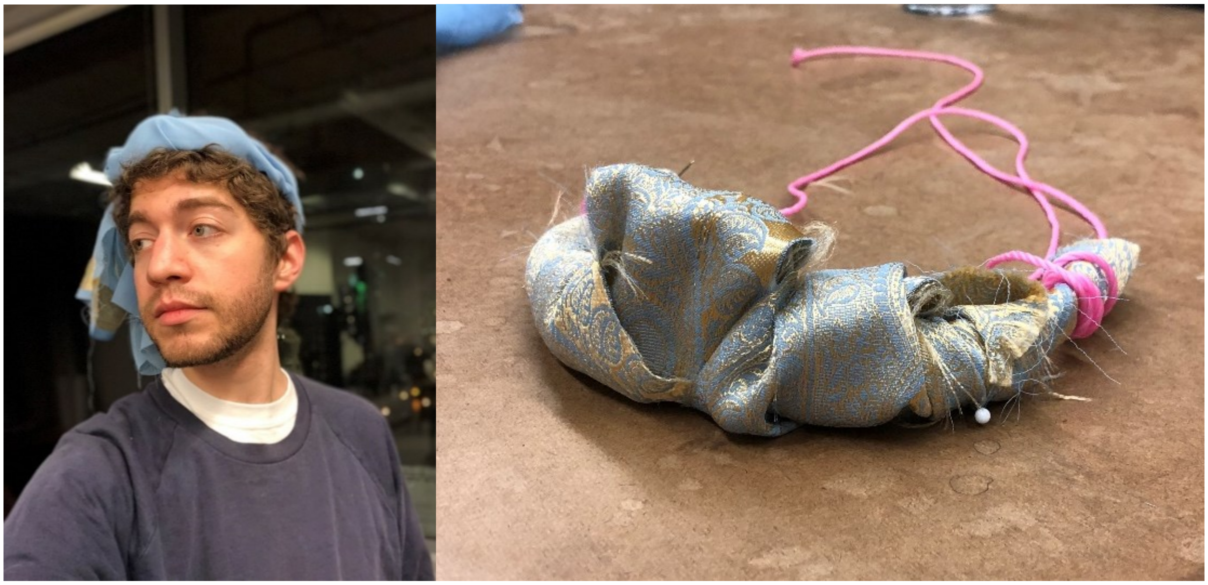


Figure 203 Fabric headwear tests inspired by Slavic women's headwear, February 20, 2018.

Bags and Pockets

The fabric and apparel work on the Cape and Camera Tiara inspired me to look further into computer bags. The first thing to come to mind when I think of a computer bag is, of course, the shoulder strap laptop bag. This symbol of early 2000s white collar work is not, however, the end point of wearable compute satchels. As I worked through my projects, products like HTC Vive and Magic Leap One inspired me to think about splitting compute and displays into distinct connected units. Online, I observed people putting VR-capable laptops and batteries into knapsacks [326]. Later, computer manufacturers introduced their own XR backpacks [327]. In Magic Leap's case, the company's AR development kit featured a puck-shaped compute unit connected by wire to the waveguide head-mounted device [328]. Inspired by conversations with my friend Christopher Novello, who is an artist and professor of computation, I began to imagine how computing devices might collide with wearable bags.

My aesthetic experiments explored how a person might wear a heftier computer at chest-height. I made chest-mounted bags, which, on Christopher's suggestion, were inspired by the then-popular Alyx chest rig [329]. I created paper prototypes and a version made of muslin. I wrapped them in other fabrics to see how more fanciful computer bags might feel. I wrote the word "BATTERIES" across the bag in thermo-adhesive vinyl to see how it would look calling out electronics components as if they were streetwear brand names. These designs emerged at the same time as I played with making a tight fitting pullover smock with a large chest pocket for storing computers. When filled with computers, the pocket created an unattractive silhouette. I felt compelled to finish the smock design in denim without the pocket. I returned to chest bags a few months later and made a pair of additional muslin bag sketches with zippers. These led to the development of a smaller shoulder bag for carrying a phone, wallet and keys. I made one prototype in muslin then a more aesthetically pleasing version in denim, which I used regularly for a few months.

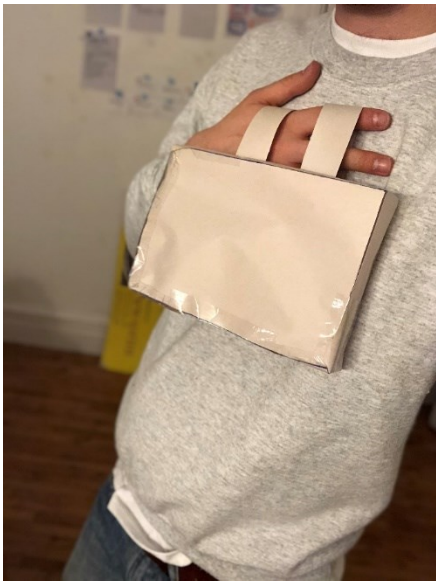


Figure 204 Paper chest bag sketches, February 23, 2018.

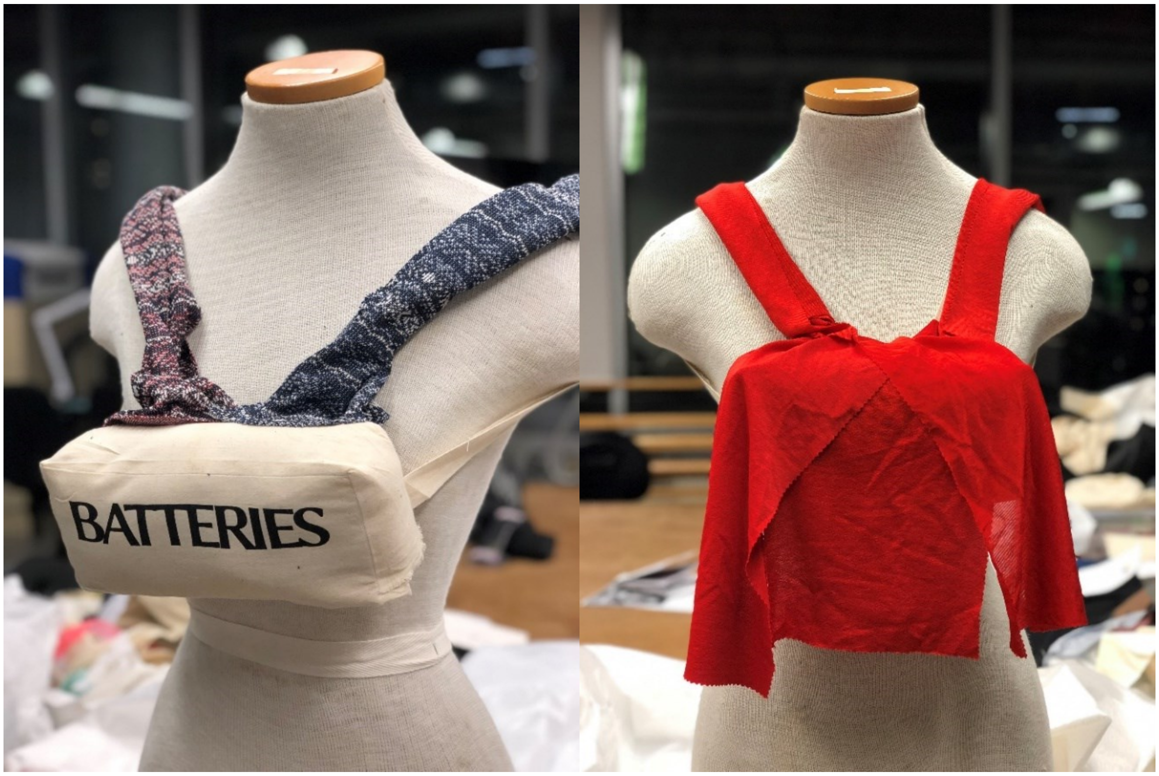


Figure 205 Chest bag tests with different fabrics, February 20, 2018.

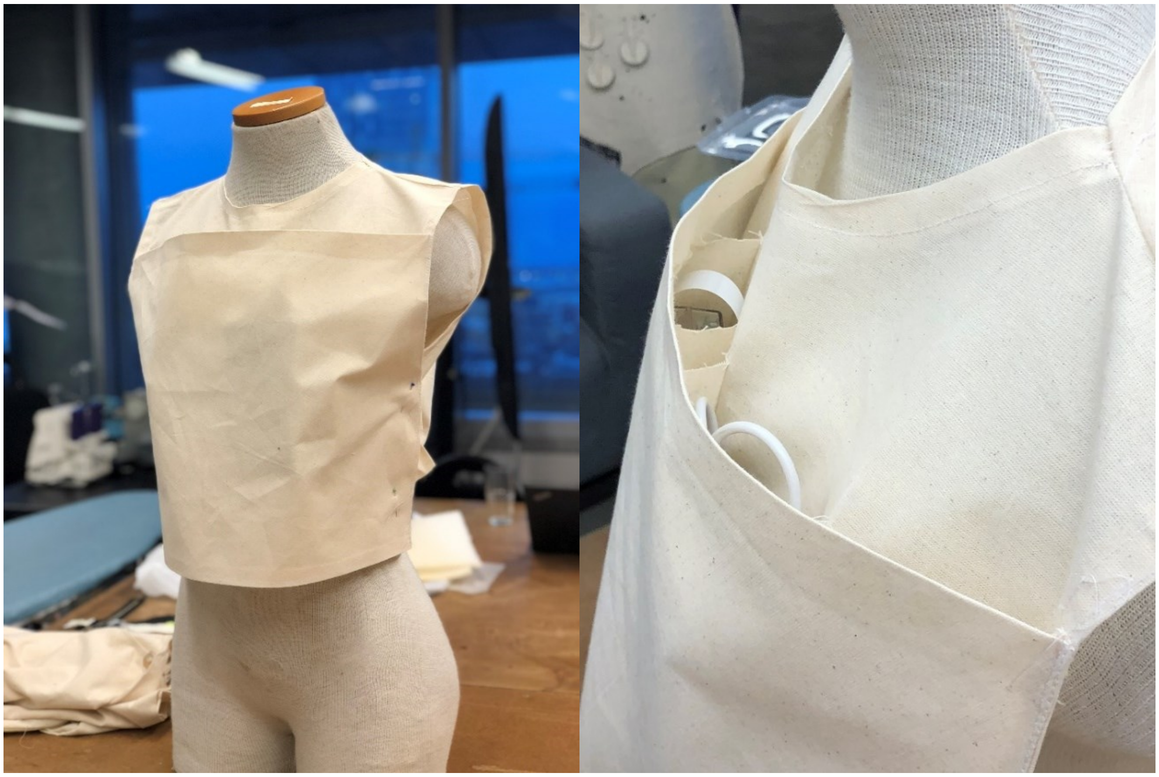


Figure 206 Pullover smock with chest pocket, February 21, 2018.

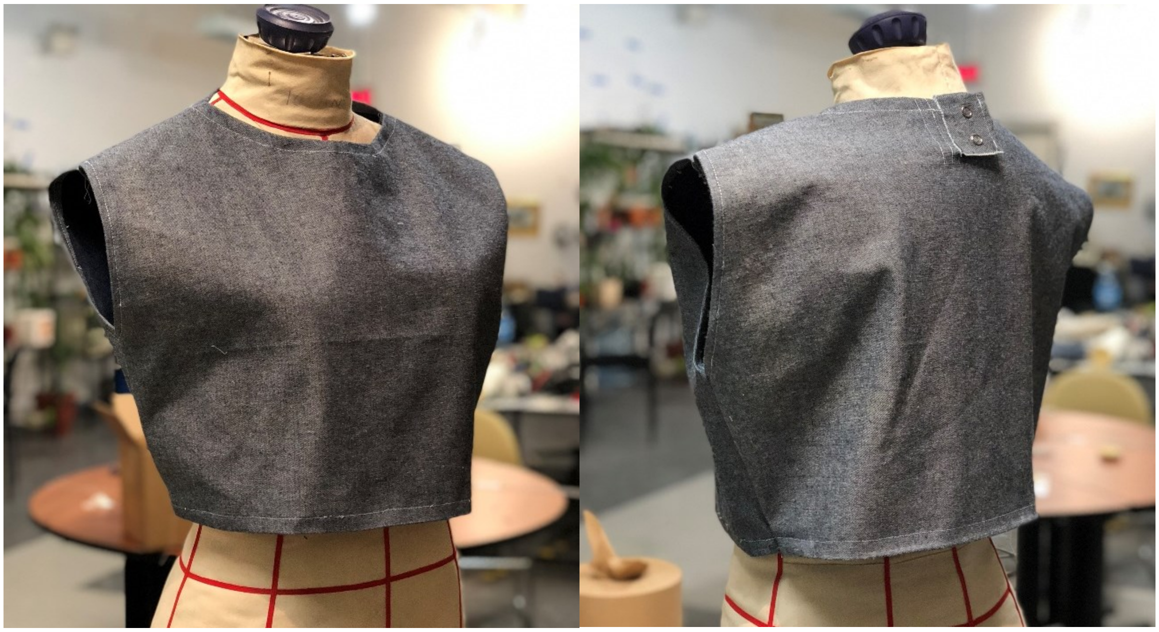


Figure 207 Denim pullover smock, February 24, 2018.

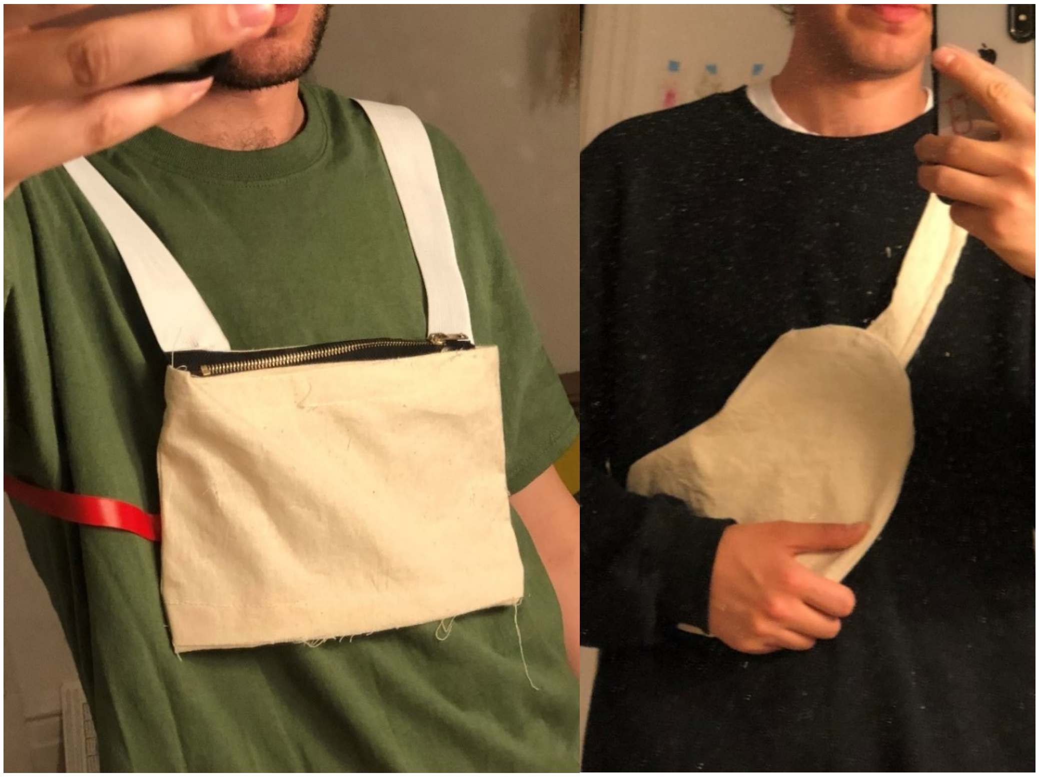


Figure 208 Zipper bag and kidney bag, June 23 and 24, 2018.

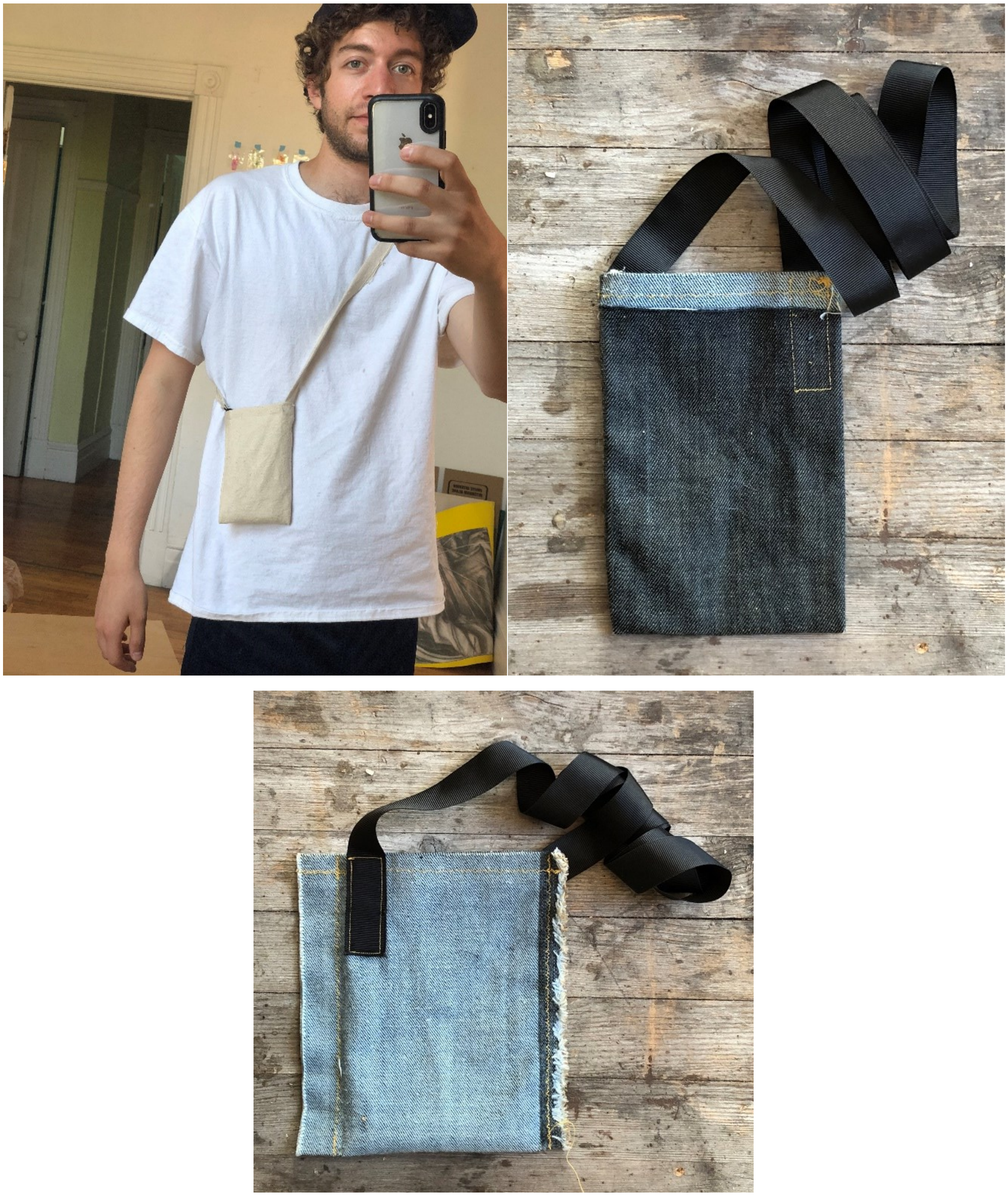


Figure 209 Shoulder bag for phone, wallet, and keys in muslin and denim, June 29, 2018 and August 24, 2020.

I also experimented making simple canvas shopping bags. I made a bag called Sac Sac that features illustrations of other bags on both sides. I also made the Alphabet Bag, which has Velcro-backed letters and Velcro patch, so the bag carrier can customize what it says.

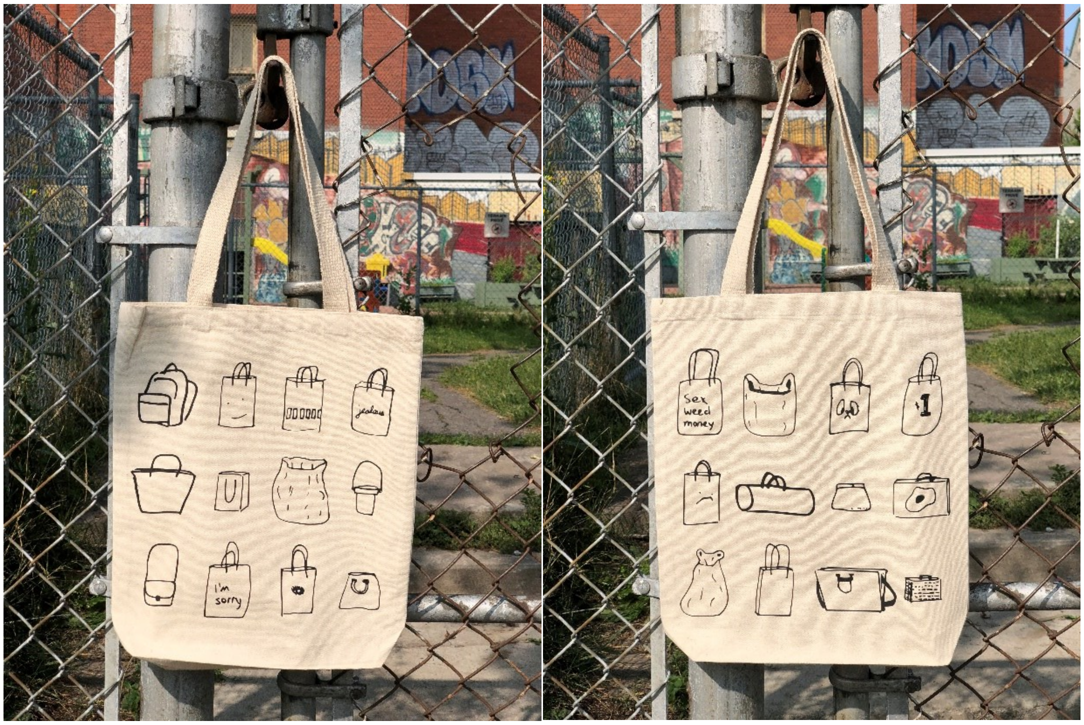


Figure 210 Sac Sac, August 18, 2018

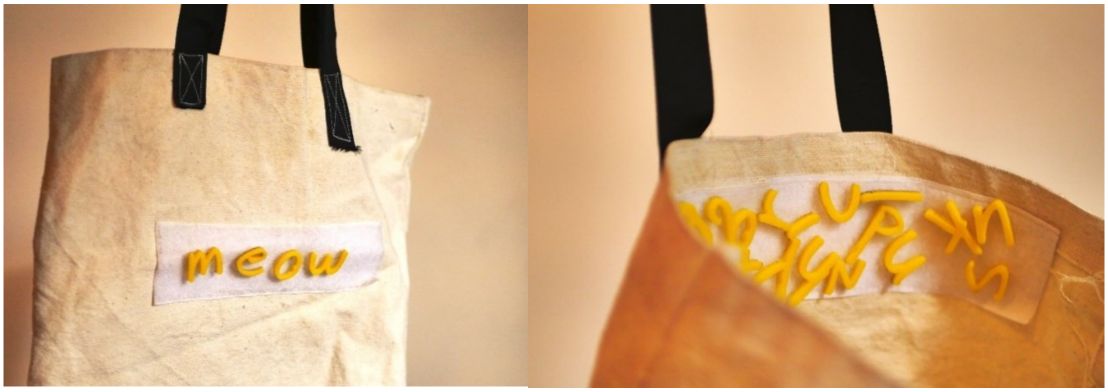


Figure 211 Alphabet Bag, August 27, 2020.

Plastic Jewels

I revisited jewelry fabrication later in 2018 with two final experiments. First, I made use of a piece of scrap iridescent acrylic that someone had left at the lab to create a quite interesting pendant sketch. Iridescent acrylic is regular clear acrylic coated with a special iridescent material on one side only. To test turning it into a piece of jewelry, I downloaded a grayscale depth map from the web, following a suggestion from my colleague Raphael Demers. This image, likely captured with an Xbox Kinect or similar structured light sensor, represents measurements of the distance between the sensor and the surfaces of the captured object on a scale from white to black, where white are the closest pixels and black the furthest away. I used the laser cutter's relief engraving mode, which varies the laser power as a function of the darkness of the pixels in the source image, to burn away material and roughly replicate the 3D form in the original. I did this engraving operation on the non-iridescent side of the material. The process left me with a handful of beautiful jewel-like pieces of plastic—fit for a pendant or earring. In tests, I found that the colour of the light is most beautiful when the object is placed between the sun and the viewer, with the iridescent side facing the sun. The colour of the light that passes through the plastic changes as the object is rotated.

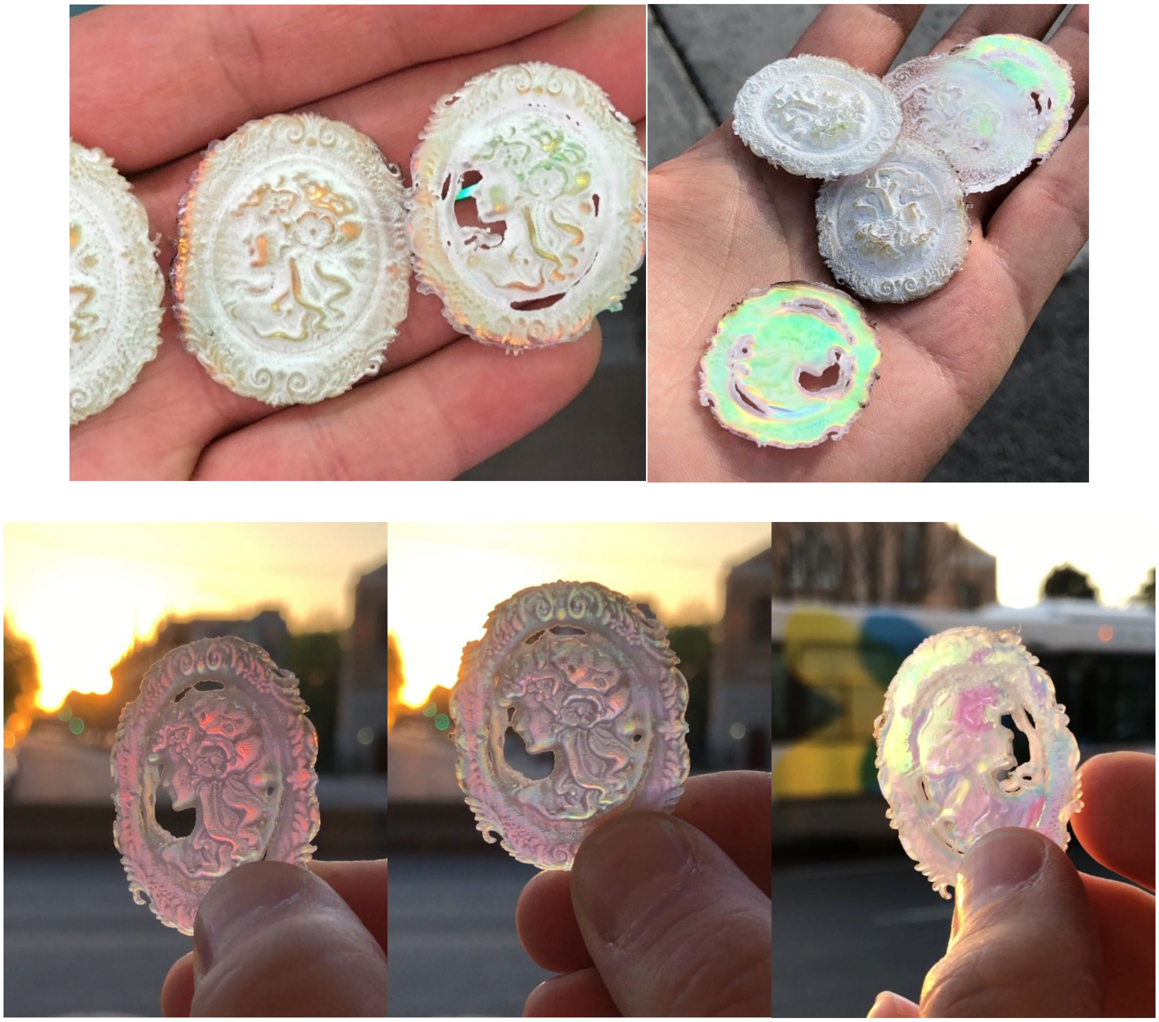


Figure 212 Iridescent jewelry tests, May 17, 2018

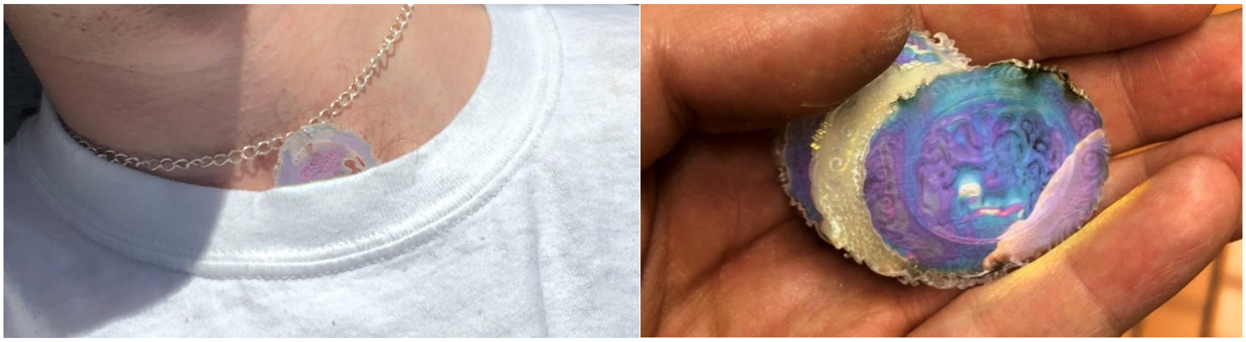


Figure 213 Iridescent jewelry tests, May 17, 2018.

Second, I created a few pendants out of acrylic, vinyl, and wood embedded in one another. In the first test, I laser cut teardrops out of a mirror-finish ⅛ inch thick acrylic and embedded them, back-to-back, in a piece of laser cut ¼ inch transparent acrylic. I then embedded this larger piece in a laser cut ring of MDF. I made another pendant with a mirror heart in place of teardrops. In the third and final experiment of this series, I cut two pieces of gold vinyl into the shape of a hot dog and its mirror image. I placed the vinyl back-to-back between two layers of transparent ⅛ inch acrylic, which I embedded in a ¼ inch piece of transparent acrylic. In both these and the iridescent pendant experiments, the illustrations were generated by scanning hand drawings executed with a felt tipped marker.

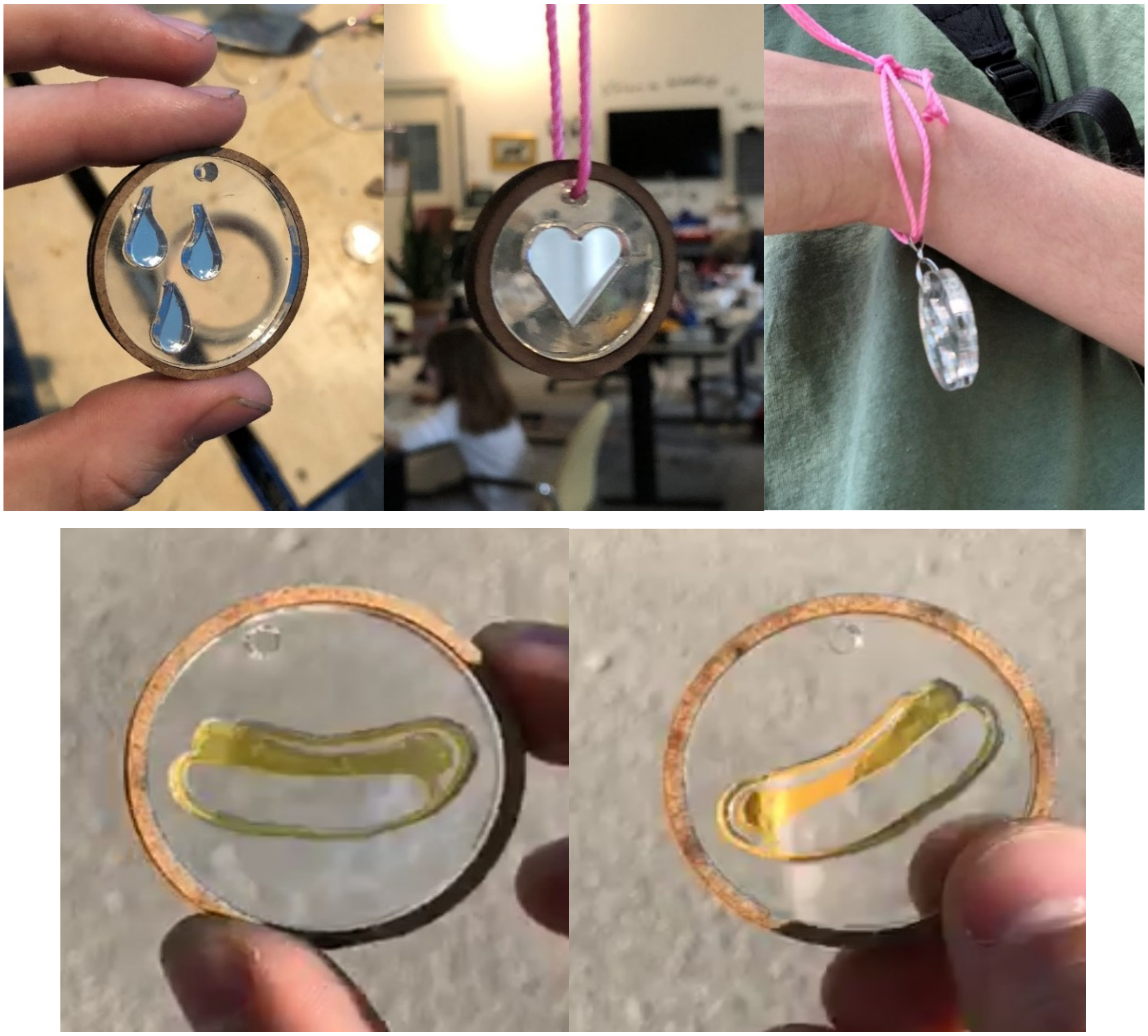


Figure 214 Embedded acrylic pendants, June 14, 2018.

Graphic Shirts II

I returned to the ewaste fashion experiments in 2018. For the first time, I used the laser cutter to cut pieces out of one fabric, then I attached them to a t-shirt made of another fabric. I created two shirts inspired by the physics theory of the heat death of the universe in this way. These were experiments at inventing a new streetwear brand based on a scientific concept. I sewed the laser cut material to the t-shirt in Heat Death I, and I used heat activated glue to fuse the fabrics in Heat Death II. I made digital sketches of a streetwear shirt about the JavaScript programming language. I also used the laser cutter to burn the words GPT-II into a piece of fabric which I sewed onto a shirt. GPT-2 is a powerful machine learning model for text synthesis created by OpenAI. I created a small collection of shirts called Poubelles in response to the Vidange art exhibit at gallery Gham & Dafe. The collection was made up of hand drawn illustrations which I transferred onto shirts with the vinyl cutter and thermo-adhesive vinyl. It included a shirt with a chicken, a broom, a garbage bin, and a jetski with the word Canicule—because it was a very hot summer.

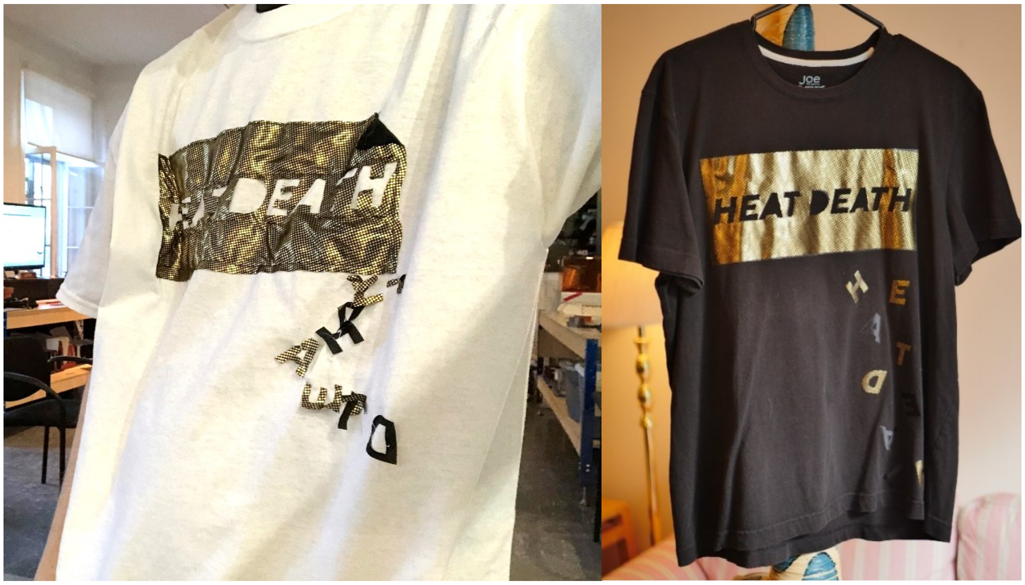


Figure 215 Heat Death I and II, August 27, 2020.

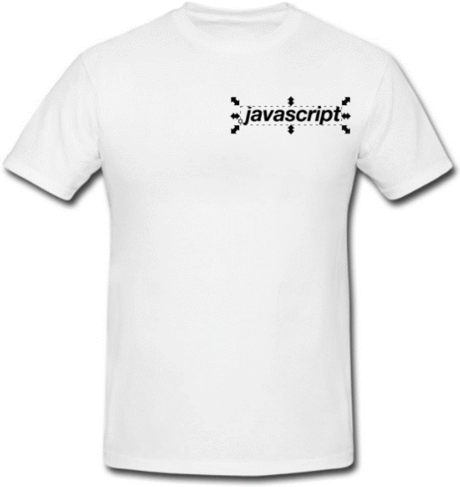


Figure 216 Javascript shirt sketch, September 1, 2019.

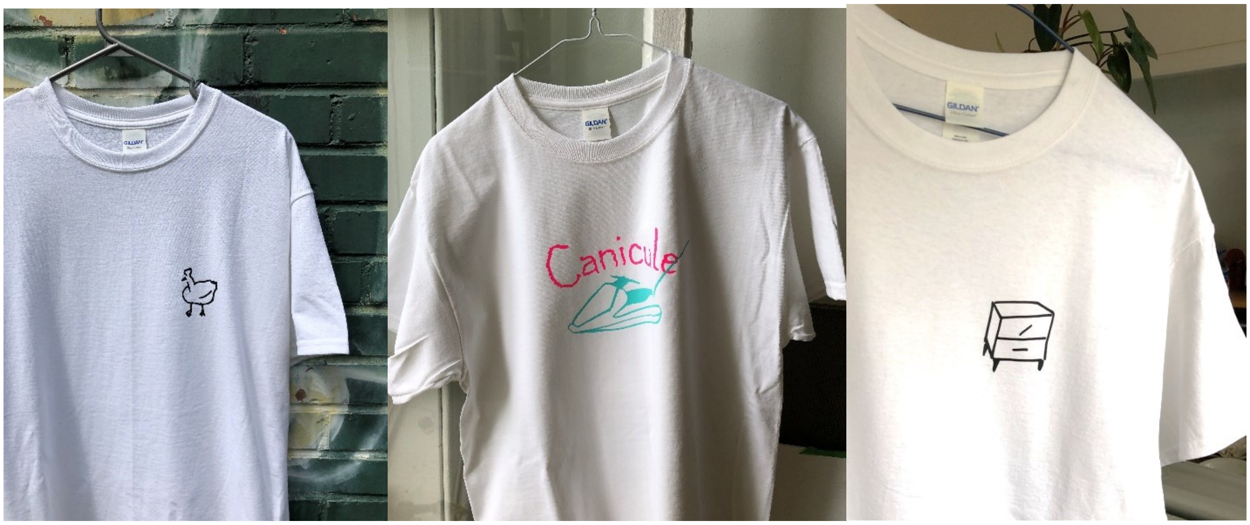


Figure 217 Poubelles t-shirts, July, 2019.

Conclusion

In this appendix, I have documented four years of research-creation experimentation and contextualized these projects in relation to the themes explored in the five chapters of the dissertation. There are three key takeaways from this section. First, loose experimental research projects take longer to complete but may lead to more meaningful research outputs. Second, when making, move quickly and perceive setbacks as opportunities for the research to take a new direction. Third, artistic research-creation interventions can productively inform more straight-forward research writing projects.

First, loose experimental research projects take longer to complete but may lead to more meaningful research outputs. Although the research and writing of this dissertation took *far* longer than I would have preferred, the additional time allowed the project to grow into a fuller examination of the physical, digital, and conceptual network arts. Research-creation experiments led me down paths of inquiry that I could not possibly have anticipated at the outset of the project.

At the beginning, for instance, the research programme was only to experiment with new applications of camera hardware. During the course of the experimentation, however, the project bifurcated into works-like camera hardware prototyping and aesthetic apparel experiments. These related but distinct research trajectories took me into different artistic disciplines, such as digital fabrication, clothes making, and jewelry making. They also encouraged me to travel around the world in search of new inspiration and studio-like creative spaces. In the end, I was able to group these two related experimental paths under the umbrella of ewaste. I would never have been able to predict the importance of the fashion path upfront, and I would certainly not have imagined having the time or mental space to take up techniques like digital embroidery and basic tailoring techniques under the auspices of the original project proposal. Had I been restricted to explore only what I could envisage from the start, the breadth of experiences and the rich variety of influences on the experimental process would have been greatly impoverished.

Second, when making, keeping a quick pace and attacking new research domains without procrastinating helps to quickly identify which challenges will require external support and which challenges are opportunities to adapt the project. One of the greatest delays in my research process were occasional bouts of motionlessness caused by the overwhelming nature of the wide spanning projects I undertook. Too often, my experimentation slowed because I began to procrastinate or lose focus due to apprehension at addressing a research task in a domain outside of my existing expertise. If I could correct these tendencies now, I would make sure to dive headfirst into the scariest aspects of the project as quickly as possible. This approach makes it easier to distinguish tasks that are far beyond the reach of my personal growth from those that are merely scary because they are unfamiliar. Quickly identifying members of the former class makes it easier to look for help from other parties. This preserves research momentum, which is important for morale. Identifying the cause for trepidation can also, surprisingly, lead to new research directions that embrace the constraints, rather than resist them.

My dissatisfaction with the form factor of the Raspberry Pi Zero W and Pi Camera, for instance, initially slowed my research, but ultimately led to new research experiments that were much *more* interesting than the initial design. When I left for China, the NicoCam v2 board was working and I had completed the bulk of the hard work making the camera glasses function. The interlude in China undoubtedly caused a context switching penalty, as I worked on different projects in China. Nevertheless, when I honestly faced the fact that I was unhappy with the glasses form factor, my supervisor Ana Cappelluto brilliantly suggested that I embrace the constraint and make something other than a pair of glasses in the vein of existing products. Ana initially suggested looking at masks, which led me to thinking about crowns. In the end, these ideas combined with the experience in the costume shop led me to create the Camera Patch, which I believe is the simplest and best form of the camera streaming hardware that I came up with during the whole research period. Although I became stuck on frustration with the glasses for a period of time, the constraint actually led me to a more interesting research prototype which aligned better with my ideas about ewaste and frivolous disposable computers. I would not have learned these lessons without experience in the trenches of first hand experimentation.

Third, artistic research-creation interventions can productively inform more straight-forward research writing projects. Making enclosures, clothes, software, bags, toys, and jewelry pushed me to transform ideas conceived of in conversation and in paper sketchbooks into physical prototypes. Testing these ideas against real world techniques created creative tension conducive to new idea generation. In other words, from thinking, to making, and back to thinking, the experimental process informed the writing process. Making has been essential to writing informed theoretical research.