9 Central Thesis Argument

9.1 Artists Need Accessible Technology

Artists need accessible technology, because techology is how we control mass speech in the West. Without granting straightforward access to the communication devices on which we rely, we risk developing a cultural context split between haves and have-nots. Technology is often developed by and for the most privileged, which means that the software developed to run that technology is preferenced to communicate and re-establish that privilege. Without accessible tech tools, the contemporary art we create is either restricted to outdated media, or becomes the de facto property of large advertising companies, such as Google.

Work that relies on the always-on internet to exist is dependent on external support structures no less than any work displayed by a major gallery. The advantage here of using contemporary tools is to make work accessible outside of the context of this dependence. This means that work can be embodied for collection or display in any context where an artist has the batteries, or owns the technology to display it, rather than relying on an external party for the ability to showcase one's own work. This repositions the value of artistic creation with the artist, rather than leaving the value to be held by groups that will benefit without rewarding the original producer. Collection and display is not without its difficulties, but it remains important as a financial remuneration for work done.

A simple, straightforward tool that can be distributed to isolated systems for display is a counter-argument to the idea that technological advancement means that one must give up the scarcity value of one's work in order to have an audience. It also means that video artists can access the kind of transparently direct interface used for internet distribution, which is simpler and cleaner than forcing already talented people to learn a new creative practice just to use new tools.

Artists should be connected to what is possible, so that our society can continue to understand how to talk to itself. Technology is not useful if it is not humane. What defines the humane is strongly related to feminist traditions of language, and the cyborg investigation of the possibilities between technology and the body. How we embody technology - how we make art, which is an idea - bound to the physical, this is an interesting question. One of the answers is to make the tools more accessible.

Artists have a long history of hacking together work out of whatever comes to hand. This is useful for resistance, but less good for strained gallery systems. Even a few pieces that take advantage of the power of contemporary - or affordably almost-contemporary - systems can make their metier more accessible, and with that accessibility, more able to locate a diverse audience.

9.1.1 Software tools as creative practice

Software is itself a form of creative practice and problem-solving through symbolic logic. Code is writing that designs the way that a machine should work. It is a direct control system, often described as if this then that. Art is the translation of the symbolic logic of ideas to objects, which becomes complex when the ideas no longer need to reside in a specific object to express themselves. Art relies, mainly, on context to be understood by its audience. Sometimes the context is provided by a reliance on the audience being well-read, sometimes not.

Software relies instead on libraries to function. When we include external libraries, we preserve their licencing, which frequently includes their authorship, which expresses the inheritance of the ideas our own code has evolved from. Technology is not neutral, software is not neutral even when written fresh. Technology is built on itself all the way down, and these inherited libraries encode many assumtions within the construction of even simple programs. These assumptions then control what aspects of that software are accessible to any end user. This means that it takes a specialized, always-shifting skill set to see what is possible within a given technology, even before you decide what you would like the result of the tool to be. The assumptions buried within a library will dictate some of what is possible. There will be complications, bugs, exposed assumptions. Things break. The joy of debugging code is that when repaired, the work vanishes. It is a prestige, a complex magic trick. The work vanishes, leaving only the art behind.

This prestige is dependent on pre-existing structures of authority. Being a creative work of structured language, code is reliant on many libraries, themselves code, written by previous programmers with their own expectations. The work is delicate: it depends on hardware systems and software systems to be both consistent and well-articulated in order to function, to do what it has been written to do. In this, code is not so different than an essay, or any highly conceptual creative practice, all of which rely on some degree of education to communicate their themes and ideas. This means that code, as a formalized expression of writing, is already an expression of a system of privilege. Technology is often created with an eye to a limited cultural outlook, because the system of privilege that produces software is exclusive to a broader set of creative voices. Therefore, there is a space open for technology designed to be subverted.

9.1.2 Large audiences are not always beneficial

This is reflected in arguments about book copyright, as much as about film copyright. Google's efforts to digitize whole libraries have been met with resistance by the Author's Guild of America and others, because the digitization of the work elides the value of the author at the nominal benefit of a broader audience for their work. In truth, this publication gives Google more space to place their advertisements for no fee to the content producer, the artist who wrote their idea to begin with. The same is true for free games, but because games require action - a degree of concentration and involvement - they cannot be "stolen." Their experience is crafted via interaction, where the experience of writing designs instead a memory that lives on inside the mind of the user.

Therefore, the benefits of cooperation with advanced corporations are not unmixed. These are not companies that pay out artists after using their work. This reduces the most obvious means of value, the money, to nothing. This means that it is important for software workers who wish to work with artists collaboratively to consider the question of remuneration and audience, and further, the true value of privacy. Privacy can substitute for materiality, because in both cases, it is the sensation of exclusivity that is on offer, rather than the object itself being of discrete worth. This is important for new media, because new media is famously difficult to collect.

The best new media, like most of the best new tools, makes it easier for artists to work and to get paid for their work, while also being straightforward to maintain, repair, and distribute publicly.

9.1.3 Gendering Code

The best new systems were, for a period, likely to be produced by individuals with a superfluity of time and access - capital formats that are associated with luck and conventional constructions of social privilege. This means that many hackers - a word here used in the MIT sense of "constructive re-users of technology" - came from an exclusive position. One might even think all of them were destined to do this. Unfortunately, this is not the case: in the 1970s, programming was advertised to women as a reasonable career even in Cosmopolitan magazine **ensmenger1** It was not until later that programming came to be associated, through a system of hypercompetitive standardized tests, with masculine traits related to the ideal of math and science as the height of rational - manly - work.

Code is not necessarily manly, or unmanly, or gendered at all, until someone puts a gendered pronoun into their documentation, and thus demonstrates their assumptions about their users. Presenting gender as visible in code is unfortunate: it is an avoidable, simple grammatic change to shift a library from being a nominally more-neutral territory to one that is passively armoured against outsiders, as demonstrated by Miller and James in their paper "Is the generic pronoun he still comprehended as excluding women?" **pronounscience** In this case, the outsider would be someone who would prefer a neutral pronoun, or a female one: this is the sort of thing that matters a great deal in writing, particularly in writing with heavily gendered languages such as French, but which becomes elided when it is expressed in the apparently neutral world of technical production.

I would argue, elsewhere, that this is because the technical world has been so coherently reconstructed around competitive performances of masculinity. This argument is too broad for the scope of this particular paper, however: arguments around masculinity and its constructions compose entire faculties. This is intended to be a small paper, about the importance of access and privacy, and how software tools can permit artists to work more freely with more of both. A simple tool can articulate things inarticulable by something more complex.

9.1.4 Simple, elegant design is important for accessibility

In English and in the world of industrial design, which code most certainly is, the best arguments for design simplicity are made in "The Design of Everyday Things" by DA Norman. Norman argues that "far too many items in the world are designed, constructed, and foisted upon us with no understanding - or even care - for how we will use them" **everydaythings** While Norman is there referring to objects, it is clear from the complexity and complaints around any major software package that there are similar upsets to be found within software. The more difficult it is to set up a new interaction, the less time there is for designing what the content of that interaction should be. Good design also recedes. It is the bad that comes forward. In this case, the bad which comes forward is comprised almost entirely of linguistic assumptions in design: things that disappear as neutral when they should appear as distinct, and problematic.

The easiest writing to absorb on this topic is not apparently about design at all. It is, instead, about desire - but the design of objects is the design of that which is desired, within a capitalist market-driven system, because one must design, and then spend vast amounts of capital to build, that which others might buy. The resonance of writing, design, and desire to poststructural feminism is made stronger when one considers the works of previous feminists. The suffragettes who got ladies the vote also sold them their soap via advertising firms at the turn of the century **consumerbeauty** This cycle carries forward and forward and forward: women are people, people are people, there is a troubling disparity in which work is assigned to whom when, which carries forward invisibly into an understanding of who is allowed to use which tools to do what work, also known as privilege.

9.1.5 Linguistic visibility and the politicization of language in Canada specifically

There are lots of interesting threads in the disappearance of a language that drives the mechanics of a world. Computer programming language is the language that builds medical devices, cars, phones. That such language recedes when good - thus remaining unexamined for long stretches of time - and then appears when buggy is an interesting problem, one that is a challenge to overcome. This is where it becomes useful to look outside of computer science to begin to have an idea of how encoded ideals within language can influence the effect one can have one one's world. The best-known and most straightforward theorist on this topic is Hne Cixous, who wrote in 1964 an essay called the Laugh of the Medusa.

Cixous spent a lot of the Laugh of the Medusa on the idea of direct, physical desire. This has some interesting resonance with the popularization of code-communication, because pornography sprouted in the comfortable anonymity of the internet like mushrooms sprout in forests after the rain, so clearly someone's desires were not being met by the extant system of expression. This resonance is outside the scope of my thesis project, although I believe it is interesting to note that the majority of young women, given a dual-screen game, made games about dating, directly about gender, or in the case of the finest producer - a transhuman - pornography itself. Embodiment is clearly very important to these people. The state of the personal narrative in a new medium is entirely too broad for me to address here, however.

Therefore, the part of Cixous that is interesting for understanding the idea of code as a creative practice of resistance is her insistance that women write, and that women write to resist the language in which they are cast. French is a strongly gendered language. It places a gender on each noun, and a degree of familiarity on the use of the term ”you” that we cannot yet articulate within English: the best we can do is ”they,” which is pluralized in a fashion that ”vous” is not. This is particularly interesting in Canada, a country of dual languages, where schoolchildren are trained from a young age to have at least a passing fluency with both worlds. We are expected to learn between two countries: it is illegal to exclude the exoticized European other from our signage.

Language is a politicized site of visibility within Canada. This forced inclusivity, which itself still elides much Canadian experience, is important to my understanding of code as a practice of linguistic structure. Cixous' particular articulation of the value of the minority writing their own world, even when forced to use a dominant language - that itself is constructed to preclude their existence - is a valueable view of how resistance to invisibility might be realized.

In code, this comes because code languages, and the libraries that make them work, are released by major corporations on a regular basis. Innovations become dependent on an ability to fall in line with the way of thinking that hundreds or thousands of previous workers have made happen, while still preserving a sense of creativity that permits one to figure out basic problems. These problems are largely technical: how to make best use of a second screen, how to force a specific video to play back in that framework. Everything inside a computer is numbers, however, and the way those numbers line up has been dictated by politics. ScreenPerfect, as psXXYborg, was written to take advantage of the Safari browser - backed by Apple - but webM video files are backed by Google, and are more compact. Due to complex negotiations around copyright, the Apple's webkit solution will not use webM, and Chrome would not play back H.264.

These differences are often put down to simple ”technical problems,” but this idea is inaccurate, and elides the code that underlies the systems built by large companies to transmit information. Video files are a contested ground at the moment, because in addition to being a convenient communication medium, some video files are incredibly protected forms of information. The litigation surrounding piracy of television and conventional first-run media has been publicly associated with theft: a type of theft where an object's value is removed even while the object itself remains present and resellable. Patent fights about video format, about technical standards, and about open or closed systems, dictate the terms of how media may be distributed and displayed.

9.1.6 Effectively resisting authority involves access to privacy and the ability to realize one's own visions

screenPerfect gets around many of the restrictions on video by coding a new system for display and interaction, which values a short, engaged experience over the longer-form systems that are already in place. Rather than pursuing the television experience, SP turns video to a system more in line with video games, specifically riffing on an engine popular with independent game-makers. Independent game-makers are concerned with a new medium, the video game, which is made up of the interaction of people with the game system. As Galloway argues in his essay on gaming, the software alone does not count.

9.1.7 Video Games are new, and therefore as-yet free for expression

Video games are new, and therefore they are as-yet unformalized: the best we get is a system of triple-A games, which are largely concerned with imposing men shooting things with imposing guns. The triple-A framework is not typically particularly feminist, with one or two startling exceptions - the end of Saint's Row 4, which is a feminist game through and through, features the rescue of Jane Austen by space aliens, for example. By and large, the popular and commercial nature of large games precludes the sort of deeply personal representation that Cixious refers to in her own work.

Games - code activated by interaction - are not restricted to only the blockbuster, however, any more than film is. Games are sometimes accused of Cinema Envy, and I believe this is a real problem: because they are not yet taken seriously, games are still free to explore new ideas, to be obscene or funny or reflective of their own culture. Jane Austen is peculiarly popularly persistent, after all: even post-Bridget Jones, the Austen canon continues to inspire new works of popular culture. The important part, however, is that game-makers are still relatively free artists. The code they use, based on massive information systems, still has the potential to make money and drive creativity in the use of and development of technology.

Innovation is not a worthwhile term here, because it has recently been devalued to refer primarily to systems that can be developed quickly and deployed to a broad audience. My interest is not in the broad audience, although it is simple good practice to write code that can be widely distributed. My interest is in the support and production of technology that permits new forms of private expression. Here, too, I do not mean private in the sense of overly limited. I mean private in the sense that something may be presented to a limited audience with the understanding that the complete experience is meant to be retained by that audience. At the same time, the technology is intended to take advantage of systems already in broad use, because these are systems accessible to artists, who must work with what is presently real to define what might become real.

9.1.8 the importance of collaborative practice and responsive development practice

Systems designers can build systems well, artists can make visuals well, this means that they should maybe collaborate because systems designers will otherwise just design incredibly beautiful systems, which are invisible to anyone but other systems designers.

In this, it is important that the privacy of a system not be restricted to exclusively those with the major capital to install and control a given band of technology. While developing the dual-screen technology to run psXXYborg, I was considering a scene from Cory Doctorow's Pirate Cinema (2012), in which a crowd of young film-makers, who make their work entirely from pirated media that has been remixed and repurposed, put up a movie theatre in a forested park by synchronizing the pico projectors on their phones. This was the chief inspiration for screenPerfect: a technology so lightweight that it would require no setup and no particular technical skill to use, which could then permit artists who already had clear vision the advantage of being able to screen their works anywhere, quickly, with nothing to lose.

Pico projectors have not yet taken off in popularity, because they have not yet begun to be built into smartphones and cell phones. That does not matter. We have a laptop on every table, and on those laptops are browsers. To write software that can be understood by the browser, one must interact with a system of capital that is dedicated to the co-option and devaluation of the author at the privilege of the corporation (TVO interview, Sawyer). The point, then, as a practitioner of a form of creativity that has not yet been completely co-opted - for the creativity of someone solving a problem within code, a specific problem especially, is still a creative practice - is that we may resist and open a door to further resistance.

Together, good art and good technology can make good experiences. What I mean by good experiences is experiences that ask questions or display well-rounded characters or use any of the many guidelines and theory that we have that drive the Humanities and the arts more broadly. If we pair people off, then we lose the part where the technologists are scared of the artists because the artists speak a mysterious language that the technologists do not understand, and the artists can learn that what looks like magic - a beautiful glass panel that just does what they say - actually takes a lot of work, it's not work that is unquantifiable either. It's built in code commits and checkins, you can see where the period goes, it isn't being a wizard. On either side. It's being a mechanic. It's assembling things well and putting them together and bolting them down and recording that so that it can be done again by another person. This is true on both sides of the equation. Feminism is equality. Art is technology.

By driving with art, rather than technology, the experience can be newer, and less expected. Together, they can make each other better and more engaging than anything with a sole motive of profit can be alone. Together, they can wish for something better.