Ontario College of Art and Design University

screenPerfect: the importance of accessible technology for artistic use.

Author:
Alex Leitch

Supervisor:
Dr. Emma Westecott

A thesis submitted in fulfilment of the requirements for the degree of Masters of Design

in

Digital Futures

in the Faculty of Design

13th December 2013

Declaration of Authorship

I, Alex Leitch, declare that this thesis titled, 'screenPerfect: the importance of accessible technology for artistic use.' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a master's degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ____

Date: ____

Acknowledgements

I wish to express my appreciation of my thesis advisors, Emma Westecott and Simone Jones, without whose generous contribution of time and sensible advice this would be a much weaker paper. I would also like to express gratitude to the Site 3 coLaboratory community, who provided me space to work and a community to work with, and Bento Miso - Dann Toliver, Jennie and Henry Faber expressly - for their technical advice and their help in hosting No Jam 2. I would also like to extend my thanks to Evan of the Digital Futures tech desk for a key tip when I was just starting my research. Hannah Epstein deserves my thanks as well for being a fantastic collaborator.

To my personal support community, many thanks for your patience with me. It's okay, it's done now. Now we can go back to conquering the world.

For Adina. . . .

Contents

Abbreviations

CYOA	**C**hoose **Y**our **O**wn **A**dventure A popular format for game narrative, describing a branched, choice-based structure, as versus a linear story.
FiG	**F**eminists **i**n **G**ames A SSHRC-funded association of digital researchers interested in disrupting gender bias in game development.
DMG	**D**ames **M**aking **G**ames A non-profit community organization based in Toronto dedicated to supporting dames interested in making, playing, and changing games.
JS	**J**ava**S**cript A scripting language, traditionally used for client-side programming on the web.
Indie	**I**ndependent Developer Indie developers are game developers not associated with tradtional, well-funded development houses.

1 Introduction: Researching A Better Game Engine

This is a thesis about art and technology.

The arts and the humanities are the technical name for the fields of work that produce both North America's culture and its record of its culture. We use computers to do the work, the same as people use computers to do most other kinds of well-paid brain work in 2013. Computers are about as far from a straightforward tool as it is possible to be: in 2013, a computer is not so much a hammer as it is an ongoing negotiation between software that works pretty well most of the time and hardware that is generally reliable as long as it is not presently wet.

The use of computers as tools is a specific skill set, which is as unique as the skill set of using a paintbrush. The key element of computer skills is a comfort with curiosity: digital tools change all the time, and many of them are not well written. Almost all take a vast period of time invested in skill acquisition before content - art - can be produced, and this time is expensive. People who have little time cannot afford to acquire the skills to use a new and complex tool. Due to these restrictions, the number of people for whom computer use in and of itself will be a delight is a limited population. This is problematic, because art production is difficult and time-consuming even without the boundaries raised by software challenges. The more limited and specific the skill set required to use contemporary software tools, the more difficult it is to include a diversity of voices in the cultural production of genuinely contemporary work. When artists are excluded from technology, culture splits on lines of privilege. There are artists who make art, and technologists, who make technology, but do not see themselves as particularly responsible for the ideas encoded in their work.

Technology is not neutral. It is authored, and where there is authorship, there is a responsibility for ideas. When large groups are left out of communication media, particularly those tasked with producing the language with which culture speaks to itself, there comes a disconnect in the public representation of our sense of self.

Video games are the newest of the cultural production engines. A good game, as described by the MDA (Hunicke, LeBlanc and Zubek, 2004), first involves engaging mechanics – the loop systems of reward and scoring – then graphics to create a world, then sound to fill out that world. The controls are weighted this way and that, but the most profitable games - referred to as AAA or triple-A properties - are presently power fantasies. There are other fantasies out there than the ones at the end of a gun. The tools have been created to design gun-centric worlds, and therefore, gun-centric worlds are the ones that get built. People who are not interested in games about violence do not have ready access to other metaphors, and this restricts the voices we have allowed to produce new cultural content.

The value of a broad range of voices in any cultural practice should be self-evident, but video games are presently unique in their high-level uniformity. Unlike any other discipline, the basic tools for games tend to be expensive, and the toolsets that exist to generate games support specific mechanics in similar-seeming worlds. This means that art games are unfairly compared to commercial games produced with the same media. The commercial games set the ground of the conversation, which denies the subjectivity of the producer of the second-stage tools. These producers - developers and game designers - are disappeared into the system, their work only revealed by which elements an artist can or cannot subvert. Without cooperative tools, it is challenging to make new things.

For many years, the most important thing about new games was their graphics. We are now topping out the graphics field. New pictures are available, and will always have their audience, but as Galloway has asserted, games are not pictures. They are not films. Games are software that exists when action is taken within their rulesets, and therefore, the newest and most interesting research in games is not how to shoot prettier things, but to find new ways to shoot them. It is not the best graphics that are most important, but the finest actions, the most interesting experience for the investment.

The problem of interesting experiences is not trivial. Video games offer an economically advantageous distraction engine, a way to enact an artificial life during a period of declining general wealth. Allowing a diverse range of voices easy access to portray their own games, their own alternate or idealized modes of being, is a way of making those voices more real, of offering an alternate human experience to the “asshole simulator” (Bissell, 2013) genres manufactured at much higher budgets.

1.1 Research Question

My research question is to ask which conditions make software tools more accessible, as versus more limited, to people who are new to making new media art. I am specifically limiting my research to the idea of a game engine. A game engine is a software system that allows content producers to place assets in context with programmed scripts to generate different systems of interaction. My research is in how the use of a radically simple game engine - a piece of software which is fundamentally limited - can enable a polyphony of creative expression by reducing the friction of learning new software tools.

In this paper, I ask questions of how collaborative work affects artistic production, how much influence a tool can have on an artist's practice, what sort of assumptions are made apparent to producers via the tool, and what the position of a technologist is in engendering this work.

1.2 Initial Approach

1.2.1 Collaborative Practice

Collaborative practice is somewhat difficult, as code practices around collaboration rely on open-source techniques as well as pair programming. Pair programming is a coding practice that involves partnering a more experienced and less experienced coder in order to communicate code experience and improve the legibility of the completed software.

This thesis assumes that the definition of gaming as an art form dependent on action (Galloway, 2006) is accurate, and therefore, after developing an initial round of software, the main body of feedback and development in this software package has been via user collaboration. To that end, I organized a game jam in concert with a local coworking facility, to host an event where volunteers can use the tool to develop their video works. The volunteers can then give feedback to both the software developers and to each other on what is possible with the tool. At that point, I will incorporate their feedback into future versions of the tool, hopefully leading to a stable platform for interactive experience development. This is a dynamic type of user-centered design, relying heavily on version tracking software and rapid updates, in line with Agile development ideals.

The initial software underlying screenPerfect has been developed in concert with an artist who laid out an idea for how a video interaction might work, which has then been created and refined, released to more artists, and then revised again. The hope is that each new version of the tool will generate a useful echo chamber, amplifying new ideas even as it makes advanced technology easily accessible to content producers.

The toolset can then be released and left for artists to use and analyse.

1.2.2 Code and Theory

The initial software of this project was developed as a response to the lack of privacy and control of various shared media sources online. Rather than developing for a mass audience, I began with the idea that this should be developed privately, for small audiences using disconnected technology.

The code of screenPerfect was written in Javascript on the server and client side, using Node.JS. Node is a server environment library and framework that is intended to permit web developers familiar with JS to write their code to the server. During the process of the thesis, the idea of screenPerfect as a game engine was taken up by my industry partners, Bento Miso, who are interested in the idea of new game engines as a use case for their language, Daimio (**daimio**). In the case of ScreenPerfect, I decided to write the initial application in Node.JS and javascript to take advantage of the speed of the Google V8 code engine. When that application was done, I turned it over to Miso, who refactored the code into something that could be attached to the Daimio language system.

The critical theory that underlies this practice is a combination of French poststructuralism - Helene Cixous in particular - and contemporary writing on video games and the history of women in technology. By producing the software and content with the input of a local feminist collective, Dames Making Games, I have grounded the work in a social justice driven practice which encourages women to take part in their own lives by learning how to interact with machines and communicate with the broader world.

1.2.3 Game Research

The Twine engine is a branching narrative engine for authoring text narratives. I have taken the idea of Twine and transformed it to use video and still images to expand the possibilities of an author experience.Twine is affordable and requires very little training to access, but it is also limiting, in that it undermines skillsets in imagemaking which may already be highly realized. Cinematographers making Twine games may find it lacking.

In addition, Twine does not take advantage of the new multi-monitor systems, which are both important and difficult to work with. ScreenPerfect is useful to both people who wish to make straightforward games, and to installers who would prefer to use multiple clients to explore the possibilities of multiple-projection screen surfaces.

1.2.4 Cyborgs, Women, Game Design (needs edits and actual arguments)

My initial approach and research method to this work is based on the Agile Manifesto, a software development methodology that opposes siloed, top-down software development. I have paired Agile development with the work of Helene Cixous, whose "Laugh of the Medusa" provided a template for *écriture féminine*, an argument for women writing of their own experience in order to be made visible.

This work is related to various texts of feminist or woman-oriented critical theory that have appeared in the years since: Haraway's Cyborg Manifesto, TIQQUN's Preliminary Materials Towards A Theory of the Young-Girl. These are academic constructions of femininity as it is seen in relation to technology: they are feminist in the formal sense of the word. There are other senses of the term, which I will not be examining within the paper. Rather than expressing this work in context with Cixous as *écriture feminine*, I will be using Cixous as a reference for the idea of the alien perspective as a position of resistance within a means of expression controlled by a neutral-to-hostile majority.

This approach addresses women as an alien construct to the more conventional world of technology, which has been recently associated with a masculinist performance that is unnecessary for the pure structure of good rules and the development, through that, of good software. This construction is relatively recent, as Nathan Ensmenger presents within his work on the systematic exclusion of women from programming as a trade (**ensmenger**).

Accessibility for artists is a major contemporary concern, related to the development of languages like Processing by Ben Fry and Casey Reas (**processing**), microcontrollers like the Arduino platform by Massimo Banzi co. (**arduino**), and frameworks like Scratch from the MIT Lifelong Kindergarten Group (**scratch**).

People who write code to develop an idea are engaging in a conceptual creative practice themselves, and they then display that creative practice through the artists who use their work. Purposefully producing small tools to support creative practice is a different model than that of major software development houses, where the work is designed in isolation, iterated on a strict schedule - 18 months in the case of Adobe **adobe** - and released. There are documented design methods for software that follows this model - I have listed Agile as the most well-known - but for the most part, this is simply considered an interative working practice which aims to a functional, finished result.

Although artists are the central agents of production of the invisible yet tangible value of the culture industry, they are not the prime beneficiaries of the financial system that backs, stores, and distributes the results of that capital. Artists were not always expected to be broke, but they are generally broke now, thanks to the triumph of the ideal of the starving artist starvingartist. Therefore, software for artists needs to be inexpensive, and set up to be almost trivially easy to use. This reserves the value of scarcity to the *ability* of the artist, rather than applying the majority value to the role of the engineer. This kind of invisibility is the invisibility of good management, of any type of good administration. Like housekeeping, code recedes until something goes wrong.

To test this idea, I have approached people to produce video—games with the screenPerfect software in the context of a voluntary game jam – a type of collaborative space where participants work with digital tools to generate new, raw games in a limited window – and then compiling the results into an arcade machine for presentation.

1.3 Structure of the Remainder of this Document

1.3.1 Chapters

The remainder of this document is structured as follows. Chapter 2 covers various methods used in my research to examine the conceptual importance of accessible-technology artist tools, including the Agile Manifesto. Chapter 2 also sets out the restrictions and main theoretical texts that underly my premise of what constitutes an accessible tool. It also addresses a list of tools used to produce the thesis proper, including git revisions, LaTeX, and their shortcomings.

Chapter 3 addresses theoretical documents, and how they relate to the process of crafting solid software from an artistic/conceptual work perspective. This is where I have placed works by Cixous as well as an examination of Galloway's essays on gaming as algorithmic culture. I will also delimit which texts I consider useful for this work, and how to ground a video——game in the broader context of recent Canadian art history.

Chapter 4, Design Research, is the chapter where I work through the process of developing the game engine with reference to the benefits of open, closed, and ideologically-driven software. There is some strategic foresighting here through Doctorow's Pirate Cinema, the text I have used as a How Might We for what a collaborative theatre might be. I have also included reference to the Brechtian active audience, which is useful for gaming. Chapter 4 also contains the bulk of my research on collaborative practice within communities.

Chapter 5, The Trouble With Amateur, addresses some of the questions raised by Galloway in his *Gaming: Essays on a Algorithmic Culture*, which includes an examination of why games built in resistance to gaming tropes are largely unsuccessful and unpopular to play. This chapter is also where I will examine some of the problems of making free content easier to provide to the internet, which includes an economic examination of how the value curve of creative practice goes flat as accessibility increases.

Chapter 6 is my conclusion, a restatement of my arguments, and the source of some optimism.

Chapter 7 is a list of works cited.

1.3.2 Appendices

Appendix A is my annotated literature review.

Appendix B is a compilaton of github commit comments over the course of the writing of the game, which detail the direction of how someone reasonably confident with computers and programming learns a new language.

Appendix C is the code record of screenPerfect proper.

Appendix D is a list of the games made to date with screenPerfect and their installation sites, along with links to where they might be found for future installation.

2 Background

2.1 Existing Software

ScreenPerfect does not exist in a void. Although written fresh in Javascript, it is dependent on many frameworks and libraries in order to work. The code was developed using the Node framework for Javascript on the server, which is supported by Google. It mimics functionality produced by the Dataton Watchout system, which provides simultaneous video windows using a custom, private hardware platform. The software that screenPerfect interacts best with is Google Chrome.

2.1.1 Game Engines: What Are They?

A game engine is a collection of software designed to make it possible for a team of artists, developers, musicians, and producers to work together to produce a complete product. Traditionally, game engines are used to produce 2D or 3D experiences with clear ”assets” such as 2D sprites or 3D player character/interaction models, backgrounds, interaction assets - crates, for example - music, and scripts in a programming language to tie all of these together into a play experience.

Some popular professional engines at the time of writing are Unity3D, which features native mobile integration and ease of scripting in both Javascript and C, Crytek, which comes with many high-end 3D resources preloaded for high definition graphic support, the Unreal Engine, which is quite stable and useful to experienced teams that prefer more control over their work.

There are popular hobby engines that de-emphasise programming as well, such as GameMaker, which is prized for PC compatibility, Game Salad for OSX, and Construct 2, which is PC-only but has a powerful engine to manage game physics and interactions.

These engines all assume a certain type of player interaction: they are designed to enable designers to produce specific types of games, such as a ”shooter” or a ”platformer”, similar in style to the Call of Duty franchise or Nintendo’s Mario series. The interactions available are easily understood as a language of action by their players, provided players have previous experience with video game play.

ScreenPerfect is distinct from pre-existing engines. It is a piece of software custom written to encourage artists to use their own skillset in image and video creation to explore what is possible in an interactive experience. Screenperfect has a set of play mechanics that have been pre-written. While they can be *extended* by anyone who knows the **daimio** language, the mechanics are straightforward to use and not designed to be altered by artists. This means that artists have a consistent environment in which to place their work, which will reliably showcase that work without them needing to learn how to program - an entirely new creative skillset - to do so.

2.1.2 Twine

Twine is the closest game engine to screenPerfect at the time of writing. Twine is a locally-installed hypertext-based branching narrative platform, which produces an interactive narrative that can be accessed through any web browser. It encourages expressive type styling and elements of multimedia, including music, coloured type, and well-designed game screens, but does not require them. Twine does not yet support video narratives, and is not entirely stored online as of yet.

The Twine engine was popularized by indie gaming celebrity Anna Anthropy in her 2012 book Rise of the Videogame Zinesters Anthropy, 2012, p. 2012. Since then, hundreds of Twines have gone live.

Twine is most notably popular with the queer indie gaming community. It has been used in wide popular release by Anna Anthropy, Merrit Kopas - author of *Lim*, Porpentine, Zöe Quinn, and games critic Soha El-Sabaawi, among others.

2.2 Multiscreen Video Technology

Dual screen technology, or more accurately, multi-screen synchronous web technology, is one of the big new ideas being heavily backed by Google in 2013. As a consequence, its Chrome browser has been designed to support software developed with a specific suite of frameworks, many of which are wholly supported by Google. This is an example of how software is not free: we cannot guarantee *what* is being done with the whole of our software installations, or whether there is a security flaw in a system written to be dependent on development tools from major software houses.

That being said, Google supports Node and Chrome both, so multi-screen technology using web browsers is accessible to people for no more investment than a new language, for the moment. Google, like many future-facing organizations, has a bad history of dismissing old technologies for the benefit of the new, but Chrome seems stable enough for now. ScreenPerfect relies on Node.JS, which is based on Google's V8 engine, and MongoDB for databasing. In addition, although the engine was originally authored independently using exclusively Javascript, later versions have been reauthored using the **daimio** dataflow language to describe connections between game files. Daimio has been released under the MIT licence by Bento Miso in Toronto.

The architecture of screenPerfect is wholly new, but the concept is based on the Dataton Watchout system, which encourages producers to develop large multi-screen *single* video experiences on custom hardware. Dataton Watchout costs approximately 40, 000*perinstallation*, *whichr*

Neither of these software packages provide any kind of support for a branching video experience natively, nor do they provide the ability to use existing hardware with same. ScreenPerfect provides this ability, because it has evolved out of the independent games community, rather than from the perspective of people who primarily consume television as a media habit. It is predicated on a comprehension of gaming and interaction that includes the ability to direct one's narrative, where the appeal of media is the appeal of *engaging* with media, rather than simply absorbing what an author—director has to say.

2.3 Theory

2.3.1 Helene Cixous and the *Écriture Féminine*

Cixous' *Laugh of the Medusa* predates the computer age, but perfectly and predictably describes the trouble with programming - which is a form of writing - within *Laugh of the Medusa*:

> And why don't you write? Write! Writing is for you, you are for you; your body is yours, take it. I know why you haven't written. (And why I didn't write before the age of twenty-seven.) Because writing is at once too high, too great for you, it's reserved for the great-that is, for "great men"; and it's "silly." Besides, you've written a little, but in secret. And it

> wasn't good, because it was in secret, and because you punished yourself for writing, because you didn't go all the way; or because you wrote, irresistibly, as when we would masturbate in secret, not to go further, but to attenuate the tension a bit, just enough to take the edge off. And then as soon as we come, we go and make ourselves feel guilty-so as to be forgiven; or to forget, to bury it until the next time. (**cixous**)

"Write, let no one hold you back, let nothing stop you: not man; not the imbecilic capitalist machinery, in which publishing houses are the crafty, obsequious relayers of imperatives handed down by an economy that works against us and off our backs; and not yourself." (**cixous**)

In this passage, Cixous chides her readers for not giving themselves the permission to write freely, or to be creative. French, Cixous worked with Lacanian theory, with many disciplines related to sex, which can be distilled to reproduction if one chooses. I do not so choose: Cixous wrote just as the pill was becoming available. Sex suddenly freed of the commitment of children by the first cyborgs, creativity - the act of creation - can now mean so many different things.

The guilt remains, though. Creative practice is difficult, and every new creative practice - programming, video art, game design - must go through the same flailing critique of its status as art, or as real at all, as the last new thing. The critique then works to isolate new creative workers, making them unsure as to whether what they produce can be considered work at all.

"Today, a software program or platform, once written and deployed, relegates its user to simple read/write tasks, with little use for changing the structure of the platform, and no ability or rights to do so." (**straddler**)

"Simultaneously, the coordination of immensely esoteric skillsets are required to design and implement such platforms, consolidating power and capital with a small class of systems builders who may manifest their control in virtually any industry." (**straddler**)

"Servers, broadband, hardware... the infrastructure of the digital economy is still closely guarded and accumulated by a shrinking roster of private interests." (**straddler**)

2.3.2 History of Women in Technology

Ada Lovelace, daughter of Lord Byron, was the first programmer. She was excellent at rules, and beat herself up for it routinely **plant** The ability to put rules in order, to work backwards and forwards from a desired result all along the path of the machines, is a characteristic much sought in both programmers and game designers. Both roles are responsible for rule systems that will dictate a predictable result. Despite a historical involvement, women have been recently and quite comprehensively written out of technological roles.

The reasons for the write-out are clear. Partly, it is patriarchy. In a straighforward way, ladies may not possess uncomplicated positions of economic advantage within a patriarchy, and it is against the interests of the system to permit a polyphony of input at the rules-setting level. Computers have quickly become a good job with a good chance to better one's life. It is presently popular to assert that in the future, there will be two types of lives:

> "... those who tell computers what to do, and those who're told by computers what to do." - Marc Andreesen, Andreesen Horrowitz.

This seems broadly true, but something about the sentiment rings solutionist. Perhaps it is just that I do not personally like to think of a world where technology, and not humanism, drives society forward. I believe this is a swing state, and I believe it should be set aside, where possible. Anyone can learn to code. Learning to express oneself clearly in a creative medium is something harder.

screenPerfect is a tiny, didactic piece of software that only permits works within a specific framework. Like a gesture drawing, what is drawn is absolutely not implied, only the form. With that being true, it is astonishing how many works from the game jam ended up addressing questions of embodiment and stress situated within the body.

2.3.3 Lev Manovich, Alexander Galloway, and Software Takes Command

Alongside feminist written history, this thesis falls into the frameworks described by Lev Manovich in his 2013 book Software Takes Command. This book emphasises what Manovich sees as a gap in the academy's examination of *media* as the central component of art and literature, and seeks instead to directly address questions of how *software* can be itself analysed as possessing a direct impact on the systems of production with which it interacts.

I believe Manovich is overly aggressive in discounting the value and input of actual producers - I do not agree that individual forms of media are dead any more than I believe that print is dead - but I do think that his writing is directly related to my central research questions as to what impact a simple software tool might have on artistic prouction.

3 State of the Art

4 State of the Art

4.1 Twine and Accessible Tools

- Twines are super accessible but still fettered by presentation
- CYOA is a basic interaction
- This is often described as ”not a game”

4.1.1 Unity

4.2 Game Theory: Galloway, Anna Anthropy

- Games should be personal
- Chapter on artistic mods
- Mention Merrit Kopas’ work on LIM and HUGPUNX and etc.

4.3 Cyborg Technology: Haraway

- Emma’s paper at DiGRA

4.4 What About All The Gays And The Internet Sex

4.4.1 Cut this, even though it's clearly pretty relevant and f*cking everyone just turned out porn or violence games.

In a more complex circumstance, however, the elimination of women from the winning populace of the creative computer lexicon has an uncomfortable resonance with the minimization of the careers of female creatives everywhere. The Guerilla Girl's work through the late 1980s and the 1990s to today expresses the gender gap nicely: ladies just don't get featured for anything in art magazines except messy bedrooms (**emin**). This sexualization and obsession with the female as an object was extremely popular during the late 1990s, eventually resulting in Sex in the City, a counter-argument to Riot Grrl if ever there was one.

This is addressed most fiercely by the French anonymous collective TIQQUN in their *Preliminary Materials Toward a Theory of the Young-Girl* (**tiqqun**), released in 1999, just as the internet was really getting started. The Viridian manifesto by Sterling was released in the same year. All of these materials point towards a terror of the perfect image torrent that the internet unleashed, particularly the pornography made suddenly available on demand at any time, turned from a rare shame to a public utility pumped into one's house, like water or electricity, overnight.

The thing is, messy bedrooms hold attention like clean ones never will. The cult of youth carries on, because youth is when the majority population are still able to ask questions, and when they still have both disposable income, and time (**economyterrible**). The hardest-core gamers - and doesn't that description just have too much in common with pornography already - are stereotyped as single males between the ages of 18 and 35, although they skew older. Mainstream games consoles are dedicated to murdering things over and over and over again, in different varieties of the same acts, which bear a strong resemblance to mainstream pornography. At the same time, this depiction is complicated by the stereotype of a gamer as a male someone who does not, to put it indelicately, get laid very often.

Unfortunately, nothing whatsoever about the stereotype is true. While major gaming systems put on increasingly masculinized trade shows designed to turn out remarkably similar entertainment properties - the triple-A games that bring in hundreds of millions of dollars (**valueofaaa**) - the queers are hiding out elsewhere, and they are taking the art with them. Anna Anthropy's crew of trans indie game-makers are shaking things up by producing low-budget things which take advantage of easy to use tools to produce games that express a personal narrative, and it is this tradition that screenPerfect is intended to work with.

5 Industry Engagement and Design Research: Community Collaboration in Software Development

5.1 Industry Engagement

5.1.1 Bento Miso and Bento Box

In order to run a game jam, one requires space, and people interested in working on games that are in line with any themes you might want to test. In order to access that space, I worked with the Bento Miso coworking facility here in Toronto.

Miso is a not-for-profit bricks and mortar site that serves as home for both Bento Box, a local development hub, and the Dames Making Games, a feminist initiative to introduce women to digital game development processes. As a board member of DMG, I have repeatedly witnessed the limitations of extant gamemaking tools. The software has bugs, runs on only a few systems, or relies heavily on metaphors and software constructs that are understood to those who already play a range of commercial digital games, but which are not clear to those of us who are new to gamemaking practice.

As a not-for-profit, Miso also serves as the hub for a great deal of Toronto's indie - independent - game development community. They offer professional support and development advice, and I felt there was a good match between their professional skillset and my research interests. The DMG traditionally run a jam in November, and felt that screenPerfect - a new software designed to be accessible in a short time frame to people with extant skills - would be a good match for the audience associated with the organization.

Miso, and Bento Box, offered to help me with coding a more accessible front end to the screenPerfect engine in time for the jam, so that I could get feedback on the system mechanics rather than just the interface.

5.1.2 Dames Making Games and Game Jams

> Dames Making Games (or DMG Toronto) is a non-profit community organization based in Toronto dedicated to supporting dames interested in making, playing, and changing games. In short, we want to build an **inclusive** and **engaged** local community of game-makers. Our community isn't women only, but it is women-driven. *from the DMG.to website, accessed* **dmgto** *November 27 2013*

The Dames Making Games are a society within Toronto that work to promote women in video games. Initially funded in part by **FiG** and part of an SSHRC grant, the DMG are now entirely self-funded. They work by using the game jam method to introduce women and allies to simple game development tools. This provides a straightforward introduction to concepts of computer logic and programming problems for some people, to video game art development for others, and video game sound production for still others. Some develop system mechanics, some design whole levels or game narratives.

The point of the DMG is to promote access to this field to people other than the 18-to-35 year old males who form the primary demographic for the video game industry, in the hopes that a diverse population of game makers will produce a diverse population of games.

5.1.3 Game Jams, A Design Method

A game jam is a variant on the hackathon, which is a type of prolonged effort at taking an idea from concept to finished product in a limited period of time. They are related to design charettes or *parallel prototyping* (**charette**), a method whereby participants rapidly prototype a design idea over a short, intense period of time. A jam - or hackathon - gives registered participants a common area and space to set up their own supplies, and a theme. The group members come to the event with an idea and possibly some resources - video files, sound capability and so on - and use the jam time to assemble a game.

Generally, a game jam will produce a panoply of small game ideas with fleshed mechanics but simple art and sound design in order to demonstrate a possible path forward for a device or piece of software, which will then be polished at a later date, and presented to the indie community either online or at a social event. Sometimes these works will then go on to be finished commercial products, or intended for further consumption at major conferences such as Indiecade or GDC. These conferences ideally further the careers of the developers by providing access to funding bodies: publishing houses, or in Ontario, the Ontario Media Development Corporation.

Game jams can be time consuming to prepare, as they involve a great deal of communication on the part of the show-runners. In order to run a jam, one must open the application period well enough in advance to ensure a large available population of skilled users who are likely to be interested in producing content with the available tools, or interested in exploring new tools on offer. Typically, jam members have a theme suggested - ”Mother May I” or ”Snacktember” being a few run in 2013 by the Dames Making Games - and then participants bring their own preferred technology to knock out a fast prototype over a weekend.

5.1.4 NoJam 2: Video Video

The DMG have a great deal of experience running jams, and therefore, I partnered with DMG/Miso to get access to a group of skilled animators, filmmakers, and gamemakers. By partnering with them, I gained ready access to their community population, and they gained access to my software. One of the most common difficulties with game jams is that the short timeframe can cause a lot of frustration to new non-programmers: they spend a lot of time wrestling with tools, rather than generating the content of their games. The DMG would like to make it more straightforward for their membership to generate games and interactive narratives in a short period of time.

No Jam is a two-week jam scheduled by the DMG in November. In order to prepare screenPerfect for the jam, I handed over the basic engine to Bento Box - the production arm of Miso - who cleaned the interface elements and released a web-based version of the software for users. This was a win for them, as they were able to refactor my local code base to take advantage of a new language they have produced, called Daimio. Daimio, being a dataflow language, is ideal for describing choice patterns as they relate to a database. ScreenPerfect is a good engine match for types of games that rely on interactive choices.

As a pair, Dann Toliver - architect of Daimio - and I worked together to clean up the javascript elements of screenPerfect for speaking to the Daimio dataflow language. The group then released a refactored version of the code in time for No Jam, so that our participants could get a clean version of the software to work with. This was challenging for me, as it involved a great deal of trust, and moved the software away from how I had initially envisioned the UI. In particular, we needed to scrap an early idea for a branched narrative "tree" display, which was not included, although it had been planned all along.

After we received No Jam applications, we went through to choose participants who seemed interested the theme and the software restrictions, sent out acceptances, ordered food, and generally set the dates. Applicants were provided diaries to record their working process over the course of the week. The first weekend of the jam consisted of workshops from a variety of specialists to provide direction in how to think about the software and the jam process as research. My presentation is included in Appendix C, consisting of how to work with the screenPerfect software, how to think about multi-screen video, and how to think about technology as a form of creative practice which is both limiting and freeing.

The applicants were then sent home for a week to work on their video projects, and asked to document their ongoing process with one another on a private Google Group. Most participants ignored this request, which left us with relatively little promotional material.

On the actual weekend, we asked that participants arrive with the majority of their video content and design prepared. There were vastly uneven responses to this request, which strongly affected the ability of participants to produce a finished game by the end of the weekend. I interviewed each group early in the process, and then later polled them with informal questions regarding their experience with the software.

The group experience with the software proved interesting. Accomplished filmmakers had a better time with it, but the most suprising response was from young, self-identified gamemakers, who rather than exploring what was possible within the context of the software tools, decided instead to try to use them to reproduce existing game types, many of which were totally incompatible with the software's design. Of particular interest was the group who tried to reproduce a classic Japenese roleplaying game within the context of video: this did not work so well, and they continued to work at it even after it became apparent it was unlikely to go well. The game itself remains unfinished, but deserves mention as the most unique and possibly stubborn effort. Used to working with uncooperative tools, the participants seemed unsure how to cooperate with a tool clearly designed to a single end.

Despite this surprising result, No Jam was a success, with nine groups producing diverse works on ideas such as how to express a practice of mindfulness, how to work with pornography in a way that forces the viewer to interact with what's happening on screen, exploring systematic violence against women, exploring narratives of imprisonment, magic, and in one unique case, permitting a puppet to escape a toy box.

In setting up No Jam, we did present at least one workshop on the importance of the personal narrative in producing creative work, which may have influenced the results. Game jammers mostly described their interest in producing work that was finished, and one jammer explicitly stated that she was pleased to have had a finished work at the end of the jam, this being an uncommon result for her when she had to learn the usual round of new software each time.

No Jam resulted in at least five "finished" works, which have since been included in several exhibitions around the city, including the December and January Toronto Long Winter series.

5.2 Software Design

5.2.1 Interfaces, engines, and interactions

A software interface is the part of the software that a person interacts with directly **interactiondestext** where a software engine is the part of the code that detects and defines what a computer can do with that interaction. The engine that I coded responds to interactions sourced from users. Interface interaction is what appears to to define the majority of user experience, but the response of the engine underlying that interface is just as important. A key part of the design of screenPerfect is that it was laid out to handle things like auto-saving invisibly, so that a user's work would not be lost.

The interface of the software is just as important as the engine, however, because a poorly designed interface will confuse a user, thereby rendering the experience of using the engine figuratively opaque. screenPerfect's roots are as a software engine, which takes user interaction and then does things with it. The user interacts with the interface, which speaks to the engine, which then returns values to whichever interface the user has selected.

FIGURE 5.1: screenPerfect software communication model

5.2.2 Initial screenPerfect Engine—Interface Layout

In the case of screenPerfect, the interface is laid out in three parts. The first part is the setup screen, which is where game designers load their media (both videos and static files) and lay out the links between those files. This is the essence of a game made in screenPerfect: which choice will a player make to navigate the system as designed by the artist?

The further screens are the client and control screens. screenPerfect supports up to ten client screens and ten control screens, although the interface only exposes a polyphony of client windows, while restricting artists to a single control set for simplicity's sake.

Figure 5.2: screenPerfect initial screen layout. Client, Control, Setup.

5.2.3 screenPerfect layout: NoJam

As the chief author of the screenPerfect software, it is very easy for me to understand where video files should go, and I come to understand how the software works implicitly. This is a common problem in software authorship, much as text requires editing and paintings require critique. I brought the software to Bento Box for refinement for reasons already stated: they needed a tool to express use cases for Daimio, and I needed help designing a UX that was actually useful to users.

The final layout of screenPerfect is much cleaner than the one with which I had been working. Rather than hidden tabs, everything is laid out clearly, and allows users to see where their files are going. It is open-sourced, and available on GitHub for evolution by advanced users - the DMG has already forked a version, iV (**iv**), for further development.

What follows are screencaptures of the pre-fork game jam variant of screenPerfect.

Figure 5.3: screenPerfect NoJam screen layout. Client, Control, Setup.

5.3 Design Research: What Goes Into Starting A Software Project

5.3.1 Artist Collaboration

Beginning development from a first position within the arts is unusual, even for an Agile workflow, but in the case of this software, it has been wholly driven by artistic collaboration. I firmly believe that simple tools to relieve the friction points of the artistic process will lead to better art which is more widely available. This is to say, although the developer is themselves a creative who will decide *how* to solve problems, the problems to solve may be better handled by an outside party. This is down to an issue of demand.

In order for software to exist, and to be seen to exist, it requires an interaction. Unlike a hammer, which takes up space on a shelf, software is essentially a text document until it is used. As Gallowaysays, a game is defined by action 2006. Galloway, 2006

5.4 Development Methods

5.4.1 Agile

The Agile methodology is based on a manifesto, as is so much else of this work. Agile is a response to previous software design practices, called "Waterfall," where software frameworks are laid out and heavily documented in advance of production. Waterfall methods are popular in major software companies, which rely on extensive documentation to communicate between business units. They emphasize planning over software production or delivery deadlines.

Agile, described in 2001 by a group of software developers, reflects a less top-down approach to the software development practice. Rather than pre-planning every element of software, screenPerfect was designed by discussion with key stakeholders as to how it should come together, and what the final product of the development process should be. This is a hacker-oriented means of development, reflecting Plant's statement that reverse engineering - "starting at the end, and then engaging in a process that simultaneously dismantles the route back to the start" is how hackers work **plant** Agile, released four years after Plant's assertion, engages the process of iterative development via reverse engineering in a more formal sense. Agile specifically emphasizes **individuals and interactions** over processes and tools, **working software** over comprehensive documentation, **customer collaboration** over contract negotiation, and **responding to change** over following a plan.

In the case of my development process, this worked as follows: The initial project was laid out by Hannah Epstein, who described how the game processes for psXXYborg should work using a series of YouTube videos linked through their annotation technique. In development conversations, it became clear that Youtube, in addition to having many distracting advertisements, also did not have a great way to make annotation invisible, and was very slow to load. This is a problem with reliance on external networks: they cannot be as fast as locally served files. Hannah specifically emphasized speed, smooth loading, and using video based in static rather than streaming or live files. These needed to be served within a closed environment to an attentive audience.

From that point, I began to research languages that emphasized speed over structure. Scripting languages, without strong classes or inheritance, are especially good at this type of development work. I reached out to other developers and asked how they would solve this problem, and they came back to me with a variety of answers - some used PHP, some used Python, all of them relied on JS for their front end. In researching different ways to solve the basic problem - passing a variable back and forth through wireless technology to select two on-screen videos at almost the same time - I discovered the Node framework. I explained how this would work - minimum installation time and expense, reliance on a straightforward machine installation on-site - and got approval from my partners.

At that point, I went home and wrote a server using Express.JS, socket.io, and node to write an application intended for the Safari browser, in line with my partner's habit of exporting video in the restricted H.264 format. Due to conflicts relating to codec patenting, one of the many secret things that underly the "free" internet, Safari supports H.264 where Chrome does not. Chrome supports Webm via the V8 engine, the same engine that supports the Node framework. Webm is also a more compact video format, which results in smaller file sizes and lower bandwidth costs, which eventually affects both load time and playback lag on client machines.

Having worked extensively to serve H.264 video and discovered that loading many videos in H.264 will quickly overload the Node-native server, I reencoded the video works into the Webm format and converted my development process to pursue the Chrome browser. This also made psXXYborg available on mobile browsers on the Android platform, which is advantageous, as Android is much more readily available to low or no-budget projects than Apple devices, even old ones.

Throughout this process, I would meet with Hannah, who would provide me with updated feature requests. This is normal for the agile process, which places an emphasis on completing software delivery by deadlines, rather than on documentation and a firm attachment to original plans.

In order to keep track of all of the changes to the software, I used the GitHub source control system. GitHub allows users to store and update their code base while keeping track of changes in what are called "commits." Other GitHub users can then "fork" or copy a specific version of the existing codebase, and from there make their own changes. A "fork" is considered a new project. A "branch" is a "fork" within a project that contains changes not yet passed to the "master" branch. My GitHub commit log for the screenPerfect fork of psXXYborg can be seen in Appendix C, which documents the changes made over months - bug fixes, design shifts, changes in the layout of the program, and new ideas for the setup and control files, including the late inclusion of how branching narrative would actually work without a database.

Overall, Agile worked for this process by allowing me to respond to user requests for code changes and information rather than forcing me to work to a standard pre-set from above. A Waterfall process would have discouraged me from even attempting to take on the work. By working in small steps back from a pre-set destination with total freedom as to how the code actually came together, Agile allowed me to demonstrate different working parts of the software as they came together. The documentation for the project is tied into the code commits, and inseparable from the actual written code within its archive.

5.4.2 GitHub and Open Source Software

The development of screenPerfect is dependent on a variety of external technologies. Although relatives and derivatives of Google's V8 system are foremost among these, there is also a dependence on the licencing and mindset of the open-source movement, and the GitHub software repository system.

Open source software is not the same as free software, as provided by structures such as the GNU General Public Licence . Open source is the peer reviewing system of software. What open source means is that even if a given piece of software compiles to a single program which can then be distributed for use on the desktop - as screenPerfect does not - the code that goes into the executable file is freely available on the internet, to be changed, supported, and developed by the population of software workers who exist in the broader world. These developers may work on closed or open source projects in their usual working time. They may be very skilled or quite new to development work. What matters is that the software's code is then shared publicly, where it can be reviewed and compiled and extended and changed by anyone at all.

The intent of open source is that anyone may learn from such freely-shared information, and anyone may contribute to the collective knowledge base. In this, GitHub is not unlike JSTOR. The stark difference is that GitHub is costly to the user who is sharing information, rather than to the user who is seeking information. A public GitHub account costs nothing, a private one with a limit on projects is less than 10*CADpermonth, andanypublicprojectscanbefoundonlinebyanyone.Thisisusefulforreference, ason*

There are some obvious problems with open source. One of the clearest is that with all that intellectual property out there for free, it is a challenge to make any money on an open project. The other is that there is no way to guarantee quality: one takes what one can get, although it is assumed that contributions to projects are made in good faith, and major project contributions are checked by trusted individuals before they are published. For example, the Mozilla project relies on contributors whose code is applied to the codebase after approval by certified reviewers **mozillacontribute**

5.4.3 Licencing

One of the ways these problems are dealt with is through licencing. The Creative Commons at creativecommons.org expresses their mission as follows: "Creative Commons develops, supports, and stewards legal and technical infrastructure that maximizes digital creativity, sharing, and innovation." It is therefore an appropriate open standard licence for *creative practice*. A preferred licence for software development is the MIT Licence, which is closer to the Gnu Public Licence, but does not preclude making money from one's open source work.

5.4.4 Science Fiction Inputs

My own idea for how this project would work is taken from Cory Doctorow's *Pirate Cinema*, which features a scene wherein characters climb trees, and using pico projectors already built into their phones, assemble a movie theatre from nothing more than sheets and ropes in the trees. I felt this sort of mesh-networked sharing is much more likely than a continued reliance on the surveilled internet for sharing copyrighted and copyrighted-material derived works. Since I could not find a system that would permit this type of sharing on the internet, I felt that this project would provide a good chance to build one.

6 Complications and Problems

6.1 Problems and Complications

In this section, I address issues with the format of ScreenPerfect games. I also address the problems of accessibility in tools, including the issue of avocatonal versus vocational creative practice.

6.1.1 Problems with CYOA as a format

* People want the full achievement * There is a huge pressure on creators for consistency in order to bring out catharsis and etc. * This is super tough in a CYOA because by definition, it's not a one-playthrough thing with accomplishments.

6.1.2 Avocational Work versus Professionalisation

* We currently live in a money-based system and need to earn it via our work. * The easier it is to make a game, does this reduce the percieved value of the game itself * Puts weight on the object that distributes the experience.

6.2 Good Branching is Expensive and Difficult

7 Conclusion

7.1 Conclusion

The work that has gone into the production and release of screenPerfect is not inconsiderable. As a creative project based in grounded theory and reflective practice, code is a tricky thing to pin down. It must be declarative, yet it reveals the internal architecture of the people who write it. To write code is to be a craftsperson, and to be a craftsperson is to reveal some of yourself with every line. Programming solo is as rewarding as any other solitary occupation, yet it leaves many loose ends. An excellent piece of software is likely to require input from a wide array of specialists in graphic design, interface development, and logic. There is an inevitability to induced flaws - bugs - that cause the program to fail. Once complete, it is likely that finished software will fall out of fashion: software is the unfinished symphony of the 21st century. Just as there is no way to call a piece of writing finished, because another word can always be added or cut loose, code is subject to scope creep. Code changes. It is not a silkscreen, once pulled and forever finished. It is not a painting, which, dried and delivered, is safe until the conservators come for it. Code lives, like writing, in context and within an ecosystem. Unlike writing, code answers to its context; without the machines to whom it speaks, it is without consequence. Within those machines, it may have a concrete effect on the world around it, and for that reason it continues to be valued; this is the craftsmanship that, unlike art, continues to make a living. Code cannot be set aside; although it will work as intended, it will break without permission. To code is to attempt to write a coherent world into being, to attempt to factor in all the diverse chances within. In Cixous' Laugh of the MedusaCixous, 1976, she expresses that to be considered real, women must write for themselves. In my thesis, I have extended this to the world of code: one must write after one's own interests, because to take only that work which is assigned is to fall behind. Writing a projection/presentation/game system has been work designed to address the problem of what, exactly, a game is, or what is a valuable piece of work. screenPerfect is designed to evaporate, leaving only the experience of its content to represent itself. It does not care what your content is, or to whom you're serving it, or where. The emphasis is on allowing your audience to experience things as quickly and easily as possible. Works produced using screenPerfect can be displayed anywhere in the privileged world; anywhere a series of screens and a single server can be set up. This emphasis on experience moves the interaction sphere into the world. The display of video-art and collaborative gameplay made possible through screenPerfect can be anywhere at all, and indeed works best at night, outdoors, in temporary installations. These are the new/old/new exhibits, the unmissable one-time-only parties, the experience that happens in a hard to access place but leaves no marks for future visitors to interpret.

The reflective portion of this research has been to address the question of use and pragmatism, as well as what constitutes research within an artistic context. The answer is difficult to quantify; research is pursuing a read, book-learning, critical examination of a practice, as all art is practice. Art is fundamentally blue collar, as is coding; both are works of craftsmanship, both can be helped along by automation, but ultimately, their declaration within a finished state is dependent upon the person who has produced them. The blue-collar trade of art has been sold out by the academy, linked tightly to the idea that thinking about a thing is more valuable than working on a thing oneself. Coders are legendarily difficult to organize, resistant to unions or to the thought that they are themselves labourers.

The newest video games try to dream worlds past being human, and most fall far short. Rather than permitting a wide exploration of possibilities, many possibilities narrow to the point of a gun, to the same forearms for twenty years (twitter). At the heart of screenPerfect is the idea that we can pull away from artificial distance and have instead on-site participation, unique experiences that project real, contemporary art into real, contemporary spaces. We can have events anywhere, and these events can bring anyone together, with even terrible technology. Underlying the architecture of this code is the idea that all the different flavours of classism should be undone with the opportunity to see amazing things, no matter who you are. Space should belong to the people who occupy it most often, not only the people who pay for it at a distance.

Learning new programming languages is always a challenge, as is enacting a pragmatic device from the perspective of art theory. There is the concern that these works are not for anything, not for a job or an application or a visible piece of content, a series of products released to do something in the world. Many applications, games and devices are released into the world that simply consume resources for the sake of their own consumption, simply to show that the people involved have the resources to spend. A new digital toy is frequently out of reach for the vast group of people who depend on their existing technology to work for as long as they can make it do so, and therefore, these tools are designed in the same way as the first university mainframes: they live somewhere else, and can be accessed by even the most unfortunate smartphones, five years out of date and slow. The idea is to permit this sort of advanced media to get to places that I do not expect it to turn up, in places that are specifically not shiny. The idea is to use existing technology to permit exploration: use what there is, and then make it greater, rather than interject an external device that will require people to spend resources to use.

Ultimately, this sort of software development is about permission. Permission for people to do what they like in the spaces they need to occupy and use, including the digital space. This is not about developing a single app or a platform that will be easily marketed, but is instead about focusing on the value of the exclusive experience. The Game Jam, which is a located, people-in-the-room bit of enthusiasm, and the value of a small community which helps one another to develop, is reflected in the variety of games produced for the system. This is about exploring the possibilities of a space designed to share an experience through a personal connection.

This part is about the different values permitted to different classes of entertainment. At a high level, art can afford to be alienating, and indeed is frequently valued more highly for its power of alienation than for any other thing. This is a distinction made especially true in video games, where the power of “fun” tends to be valued highly for its commercial properties. If a game is not “fun,” all elements of its interactive powers of storytelling cease. This means that games which are not “fun” are widely called by other names - interactive new media art, for example.

screenPerfect is designed to permit the development of games that are not only not necessarily fun, but which may be narratively incomplete, or nonsensical, while still being absorbing. These are games that do not require reading, but permit themselves to be experienced as deeply as a given group of readers wish to experience them. The tool can be perverted: perverse use is built in, with video copyright being at such a premium. It can be used to host experiences in galleries or in warehouses, by people with minimal technical knowledge and little ability to mask their normal online activities. This is a device to let people make maximum use of the devices they already have, to host a dance party on a subway or a massive art tour through a gallery. The demonstration content may be upsetting, but the access permitted is broad. This is about sharing things, for the better.

8 Endnotes, References, Works Cited

8.1 Works Cited

Anna Anthropy, Rise of the Videogame Zinesters, http://www.auntiepixelante.com/games/
Merrit Kopas, http://mkopas.net/ Zoe Quinn, http://www.beesgo.biz/ Zach Blas, http://www.queertec
bombs/ Porpentine, http://aliendovecote.com/ Soha El-Sabaawi, http://www.el-sabaawi.com/

8.1.1 Software

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.

A Agile Manifesto

A.1 Agile Manifesto

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more. © 2001, agilemanifesto.com

A.1.1 What Is Agile?

* Put in what is pair programming as well.

Agile is an ideal based on a manifesto released at the turn of the century. It is not specific enough to serve as a development framework, but instead serves to structure a way of thinking about software development in collaboration between developers and their user population. The original Agile Manifesto was released in 2001, and has a vast number of signatories to date.

Agile is intended to address the gap between planned software, which is typically fixed at release, and reality, which is that software breaks routinely and needs regular maintenance and iterative upgrades. The manifesto suggests that developers should focus on results over process, software that works over software that's well-documented, collaboration over contract specifics, and responsiveness to change over following a specific plan.

Agile has proven useful for this project because its emphasis on responsiveness and deliverables has ensured that the working software of screenPerfect can move forward to permit more games to be made, rather than locking the software to a concern for itself as an object. This is a valuable way to move through a development process which spares us from needing to commit to a given method, and instead emphasizes results from practice.

This practice must be documented, which is where the code-commit process appears. Agile deprivileges documentation for itself, preferring to include the necessary as the code itself is logged. Therefore, I have chosen to use the Git toolset to retain my version control on both this document itself and the code developed to run the main game engine.

B Annotated Literature Review

B.1 Literature Review

http://kotaku.com/the-weird-escapism-of-life-sims-730629952 Leigh Alexander on aspiring to pay a mortgage in real life, which is important because people won't be able to do that, a lot, in North America.

http://agilemanifesto.org/principles.html The Agile Manifesto, which backbones my actual software development style: working software is what counts. Important, as Agile is in direct defiance of corporate control structures.

Maly, T. (2013). We Have Always Coded. Medium.com. https://medium.com/weird-future/2acc5ba75929 This article investigates gender essentialism from the perspective of biological essentialist arguments frequently used to say women can't or shouldn't code because of various, entirely specious, evolutionary problems. This is an argument that happens on top of a perceived resource scarcity; the scarcity is in this case employment of women in technology, so opportunity.

Fashion article about hegemony of fashion writing and the death of exclusivity, with notes on shows being locked down to resist the blog invasion. http://thenewinquiry.com/essays/cool-fronthot-mess/[I need some work on how basic economics works with supply and demand in the absence of a good money supply.

bell hooks. (1992). Is Paris Burning?. Black looks: race and representation (pp. 145-156). Boston, MA: South End Press. bell hooks is always useful in concert with Derrida to explain that you can't actually know what anyone else is actually thinking; having sympathy for other people is not the same as understanding their direct experience. Useful because it underlies a lot of feminist practice. This is an article about exclusivity,

which is an issue of privilege, which is an issue of perceived resource scarcity, in this case access.

Bizzocchi, J. Tanenbaum, J. (2011) Well Read: Applying Close Reading Techniques to Gameplay Experiences. In Well-Played 3.0, Drew Davidson Eds., Etc press www.etc.cmu.edu—well-read-jim-bizzocchi-joshua-tanenbaum Something to explain game design processes in the technical section of the document.

Buxton, W. (2007) Sketching User Experiences: Getting the Design Right and the Right Design. Morgan Kaufmann Publishers. Something to explain user interface design practices in the technical section.

Chun, W. H. (2011). Invisibly Visible; Visibly Invisible and On Sourcery and Source Code. Programmed visions software and memory (pp. 1-54). Cambridge, Mass.: MIT Press. Still need to read this, but the title is pretty relevant to the central research question of my thesis, which is on the value of code and invisibility as a permission.

Cixous, H., Cohen, K., Cohen, P. (1976). The Laugh Of The Medusa. Signs: Journal of Women in Culture and Society, 1(4), 875. Woman must write her own desires into being in order to be seen. This is a paper that reinforces arguments about scarcity of perception, and issues of control, specifically of women and women's opinions.

Deleuze, G., Guattari, F. (1987). Introduction:Rhizome. A thousand plateaus: capitalism and schizophrenia (pp. 3-4). Minneapolis: University of Minnesota Press. Everyone else is doing it [including them]. May as well. I gather they're popular right now.

Deleuze, G. (1992). Postscript on the Societies of Control. October, Winter(59), 3-7. Surveillance culture is bad for people, yet inevitable as it becomes automated. This is an issue of control, which is subverted by taking possession of even a single means of production; to refuse to code is to be illiterate of the systems by which production is governed. To refuse to acknowledge the implicit control of a system is to lie to oneself; if other people have designed the system, the system operates[Alex Leitch, 2013-10-01 2:18 PM There are no complete systems, particularly in computers, because Godel's incompleteness theorem says so. This is a joke that is also true and I am not sure how to cite it, but it holds for most systems of even informal logic, which computers are composed of. This is a purely theoretical way to look at a computer system that is both true and not true; the incompleteness theorem concerns math systems, not people, specifically. You can't test people for their incompleteness. But people are still incomplete. If they weren't incomplete, they wouldn't have religion.] as it is intended. The way out of this is to read the system, then find the gap within it.

Dell, K. (1998). Contract with the skin: masochism, performance art, and the 1970's. Minneapolis:University of Minnesota Press. Pain or revulsion is another way to escape a system, which is to make it so dear - dear as in price - that it is difficult to reproduce the work because it costs too much. Abjection, or the ability to pay for something with revulsion, is one of the less efficient but more effective systems of resistance. The liminal space represented by a willingness to publicly maim oneself is reserved for those who do not fit well within a system that relies on completion: broken skin stands in for a refusal to submit to hierarchy. This collapses in various ways over time - it's unlikely that even facial tattoos will keep people out of the workplace for long - but still represents a real way that people refuse to be included in a more perfect/uniform work. This is an article on how masochism, the revealing of the inside, has a recent history in art and what role that sort of display performance contributes to feminism.

Doctorow, C. (2008). Little Brother. New York: Tom Doherty Associates. A detailed look at how surveillance culture breaks down social contracts, and a how-to guide on resistance via action disguised as a novel. Also has a variety of excellent, accurate examples of ways to disguise data so that it cannot be confirmed by a rogue authority. Connected to Deleuze on Societies of Control, and specifically addresses homeland security spy tactics.

Doctorow, C. (2012). Pirate Cinema. New York: Tom Doherty Associates, LLC. Contains a scene with multiple tiny projectors used to set up a cinema in a park from pockets, which is more or less what the software itself is supposed to do when it runs. The entire book is about resistance to copyright authority. Contains a quite didactic passage on how even hardware control chips do not actually control or prevent smart-enough people from using controlled software, and in doing so, presents a vision of the world where the most privileged are no longer privileged with money alone, but also with knowledge or access, which is a type of prestige - which is a type of magic trick. Concerningly libertarian.

Gleick, J. (2011). The information: a history, a theory, a flood. New York: Pantheon Books. A survey text of the history and development of data-centric information technology. Explains a little of the context for how tools like screenperfect can be expected, themselves, to proliferate to the point of uselessness. This is useful because it prevents needing to look up each paper about completeness theories and map-rename signal-to-noise mathematics independently. Signal-to-noise mathematics are important because they provide a way to think about how to privilege information in the learning process, or the internet search process; people passively look up how to find what they need. Downside: The noise often contains trace characters that allow further exploration. Upside: there really isn't that much signal out there, no matter how much noise is happening.

Gram, S. (2013, March 1). Textual Relations: The Young-Girl and the Selfie. Textual Relations. Retrieved April 12, 2013, from http://text-relations.blogspot.ca/2013/03/the-young-girl-and-selfie.html Young women's bodies are not only super-powerful, but can be the stereotype vehicles for all of consumer culture. This is important because young women are disproportionately discouraged from tech culture, even as they control the scarcity that is approved sexual relations within North America, a scarcity that cannot be overcome with money alone - you can't buy love, they say. So this is valuable because it is an excellent analysis of how stereotypically correct bodies benefit from fitting into a system of action, which is related to code practice because only stereotypically correct bodies are encouraged to participate. Per masochism, however, there are no correct bodies, only correct images of bodies. Resist the system in a predictable direction, and you become a new format of marketed body, with a new predictability. This predictability can be coded, but the closer one gets to perfect, the harder it becomes to occupy the role while remaining human. This is a deeply misogynist text, but is a perfect text for examining the role of image on the internet, and things which are seen but hard to describe, as code is.

Grosz, E. A. (2008). Chaos, Cosmos, Territory, Architecture. Chaos, territory, art: Deleuze and the framing of the earth (pp. 15-28). New York: Columbia University Press. Technology as sexual performance and definition of space. Not terribly well-realized but more academic than other sources on the same subject. Useful because code is a creative process, and creative processes - per Wilde - are useless ... like peacock feathers, or any other sort of look-at-me performance. Even things which do things are useless.

Haraway, D. (1987). A Manifesto For Cyborgs: Science, Technology, And Socialist Feminism In The 1980s. Australian Feminist Studies, 2(4), 1-42. Primary text on women restructuring their bodies, invisibly, to take over the world. See also Quinn Norton on IUDs (http://www.quinnnorton.com/said/?p=404) - this is useful because women can resist commodification, such as that described by TIQQUN, invisibly. Code is, in Agile practice, shifting from architecture to a sort of cooking; this library and that, all put together in a frame to pursue an idea, rather than to do a specific thing from the outset. This is a text about resisting control systems by allowing oneself to cooperate until there is a space to break free. Frequently, people don't even notice you have.

Haraway, D. (2009). The companion species manifesto: dogs, people and significant otherness. Chicago, Ill.: Prickly Paradigm Press. More Haraway. Now on cancer, not sex. I need to read this but I don't think it will be too useful, except that it articulates that humans, with their tool-use, are not actually special; we are part of a system of mammals. This may be useful elsewhere.

Hunicke, R., LeBlanc, M., Zubek, R. (2004) MDA: A Formal Approach to Game Design and Game Research. sakai.rutgers.edu—hunicke$_2$004.*pdfMoreongamedesigntechniquesforthetechnica*

Kristeva, J. (1982). Powers of horror: an essay on abjection. New York: Columbia University Press. (http://www.csus.edu/indiv/o/obriene/art206/readings/kristevaHorrifying things have a power that is more potent than any non-horrifying things could hope to possess. This paper details why that is. It goes very nicely with Cixous and discussions of the IUD, because it is about what happens when barriers truly break down. I think Kristeva's horrors are basically the key to the entire news cycle and Grand Theft Auto to boot. This paper is the original on how revolting things are fascinating but resist being part of a system, unless they're cleaned away and perfect. See above comments on masochism paper; the awesome attraction of the awful.

Krug, Steve (2000) Don't Make Me Think: A Common Sense Approach to Web Usability. Riders Publishers. This is another technical paper for arguing that software design should be totally invisible. Useful because it ties together the systems of control argument - control is implicit, presented as undefeatable, a smooth surface - with the idea that things should be useable, so that people can find their own uses for the tool beyond what is initially intended by the author.

Schafer, T. (1998). Grim Fandango (1.0) [Video Game]. USA:LucasArts. Classic adventure game with minimal interface and a fixed runthrough. One of the last great adventure games. An excellent exercise in game design where the game itself is preset, but the ideas the game displays, including an interest in a subculture that is not much popularly examined (Mexico), and a good narrative. Evidence that narrative is important in gameplay, which is key to the development of screenperfect as a narrative branching tool. Also remarkably and incredibly broken on contemporary systems, as the initial code was rendered directly by processor speed, rather than at a stage or two removed; the game was broken by Moore's Law, which is good evidence for why tools need to be considered unto themselves. The new narratives are temporary. This is a narrative about the temporary; death, and the waiting period before leaving - while being wound up in a longstanding celebration.

Luvaas, B. (2006). Re-producing pop: The aesthetics of ambivalence in a contemporary dance music.International Journal of Cultural Studies, 9(6), 167-187. Retrieved April 10, 2013, from the Scholar's Portal database. An interesting look at what ethnographic research can be, and the speed of cultural shift and recycle since the rise of the internet.

To be read in concert with various VICE mag articles about cocaine, new york. Used originally in article about Seapunk movement.

Moggridge, Bill (2006). Designing Interactions. MIT Press, Cambridge MA. More technical reading about how people interact with software, about how people can control interactions.

Moyer, J. (2012, September 14). Our Band Could Be Your Band: How the Brooklynization of culture killed regional music scenes - Washington City Paper. Washington City Paper - D.C. Arts, News, Food and Living. Retrieved April 22, 2013, from http://www.washingtoncitypaper.com/articles/43235/our-band-could-be-your-band-how-the-brooklynization-of/

Cultural uniformity because the internet makes things from different places seem the same, even though they're really not the same. Relates to the Young-Girl article about how if you are one perfect shape, that perfect shape will always sell at least a little, which obfuscates the truly beautiful and interestingly specific evolutions with things which have been data-optimized to be more popular. Popular isn't better, and neither is monoculture, but also no good is the sort of individuality that is itself a sort of monoculture.

Mulvey, L. (1975). Visual Pleasure and Narrative Cinema. Screen, 16(3), 6-18. On the male gaze, which is the central gaze in most videogames, particularly first-person shooters. This is important because the male gaze sets how most blockbuster video games are allowed to be perceived. Important because video games, like most software, are mainly compared to cinema, even though they have very little in common with cinema for elements beyond the technical. Core to arguments about how women are seen, which is essential to understand the TIQQUN readings in their slightly tongue-in-cheek misogyny.

One Laptop per Child. (n.d.). One Laptop per Child. Retrieved July 3, 2013, from http://one.laptop.org/ I was thinking about discussing how the OLPC project led to various other tech advances, including the rasPI - it made netbooks happen, then tablets happened. The OLPC was the project that said "wait, things don't need to be faster, they need to be better." Absolute disaster; in the countries it was intended for, it was already superceded by mobile phones. Classic example of condescending outsiders trying to Make A Difference rather than examining difference. Probably too broad a scope for this project.

Orlan: a hybrid body of artworks. (2010). London [u.a.: Routledge. Orlan led to Lady Gaga so directly that she has since sued her. Discussing the liminality and limits of flesh without Orlan's surgeries is a challenge; almost no other artist (burden? Shoot) has gone so far, but this distance is collapsed in film like Nip—tuck and the normalization of Hollywood surgery. Related to Kristeva and articles about the mortification of the flesh for the sake of appearances, which is what I am interested in with the arcade box. Although I want that to be subtly upsetting, not overtly upsetting.

Reines, A., TIQQUN. (2012). Preliminary materials for a theory of the young-girl. Los Angeles, CA: Semiotext(e) The new translation, which includes a feminist preface by Reines about the body of young women and how she almost was sick over the assertions of TIQQUN, which happened about the same time as everyone else was going bananas for Second Life, a game where you make an entirely new body that has since been abandoned by all but the most escapist. TIQQUN accurately observe that people are escaping into their own bodies, not those of the computer screen; the new presentation is that the brain and image on the internet reinforce the physical appearance through the phone, a piece of technology governed by the male gaze.

Stephenson, N. (1995). The diamond age, or, Young lady's illustrated primer. New York: Bantam Books. This is pretty well a perfect piece of fiction about cyborgs and universal education and China as an Oriental-escape paradise. Fun look at a post-scarcity economy that has simultaneously happened and can't happen. This is a book about an alternative resistance to the always-on personal presentation future, where books reflect their users. This is a fantasia, but an appealing one, with a lot to say about the subject of veterans, what abuse looks like from a perspective other than the dominant, and what recovery might look like. The main characters are all female, and all develop in different directions, including one who escapes by using the book to hack out a new life under direct supervision. Contains an unpleasant thesis about the value of personal matriarchal influence on future leadership.

Sternberg, M. (2012). They Bleed Pixels (1.0) [Video Game]. Toronto:SpookySquid Games. Excellent representation of a female lead game character in a genuinely challenging platformer. Useful because it exposes the programmer's preference for difficult-but-rewarding game mechanic loops, along with a conscious choice to show a young woman who has strong personal agency as a hero. A manifesto for better, simpler video games.

Swartz, A. (2013). Aaron Swartz's A programmable Web an unfinished work. San Rafael, Calif.: Morgan Claypool Publishers. The internet doesn't belong to us, but it could, and here are some technical guidelines to pursuing that as a worthy goal. This is the other way to approach technical development; something that should be extended rather than presented as complete in and of itself. Swartz rebels against societies of control by describing systems to expose information at a basic level rather than obfuscate them. This eventually led to his death.

Team Little Angels (2009). Bayonetta (1.0) [Video Game]. Japan:Sega. What a hilariously sexist but also perfect meta-narrative of female power while subject to the male gaze. The rudest, most violent fun game released to ever feature a lady protected, literally, by her hair. A game with a strong female lead in the hilariously Kate Beaton "strong female characters" mold, which is problematic in its presentation even as it is simultaneously winking. Has unfortunately fixed gameplay goals, but allows players the reward of working through the game on a basic mechanic of style rather than skill alone. Fun!

Toom, A. (2012). Considering the Artistry and Epistemology of Tacit Knowledge and Knowing. Educational Theory, 62, 621-640. Retrieved April 12, 2013, from the Scholar's Portal database. More technical information on how to design interfaces so that people understand, passively, what they're supposed to do with it. This is about passive learning, which is how most people learn software: through exposure and experience with previous systems, we understand the language that the developers no longer expose even though help systems. The way of using the software becomes implicit.

Volition Inc. (2011). Saint's Row the Third (1.0) [Video Game]. USA:THQ. This and the followup, Saint's Row 4. Games that took the GTA pattern and subverted it to make a game that is cleverly and strongly and messily about playing video games and the fantasies of those games. Saint's Row is a sandbox video game about playing videogames and what a videogame means at its base. It allows people to play as whatever type of character they like, which exposes the fallacy that videogames are solidly about anything but mechanics; the art and design on top expose the code in their very mutability, but there are no ways to solve the game puzzles except violence. This is entertaining, because rather than being a game about traffic patterns and random mayhem specifically (GTA-V), it is a game about playing games, about false achievements and the ability to do anything at all as long as it's violent. Also notable because first lead female character is a hacker from the FBI. This is an important plot point. You can also play gay or with the robot AI you rescue … but not the vice president, who's a dude. Central to my argument that software development is a second-stage creative practice because with no fixed skins, the game itself is much more exposed.

C Game Jam Documentation

C.0.1 Questions To Ask Game Jammers

- What were you expecting when you came to the jam?
- What features did you immediately want in your software?
- How has your group process worked throughout the week?
- How is your group process going today?

C.0.2 Games List

- Porn Game by Maxwell Lander
- Grimoire by Katie Foster and Mikayla Carson
- Kill Fuck Marry by
- Mind Safe by Dann Toliver and Robby
- Glitch95 by Arielle, Rebecca and Bronwyn
- Omm by Brittany and Diana
- Cyborg Goddess by Cara and Kate McKnyte
- Empty Puppet by Danielle Hopkins and Dawn

C.0.3 Bug Discovery

- Room Zero must be the first room edited.
- Room Order cannot be altered in a meaningful way - ID is hidden from users

- WebM video is unplayable on Apple devices
- H.264 video is slow to unplayable on non-Apple devices

C.0.4 Features Requested by Game Jammers

- Sound effects on control input - requested by Arielle
- Timed hotspots which appear and disappear on specific video cues
- A game tracer that tracks which choices players make, and records their games
- Gesture controls - pinch, zoom, throw - on touchpoints.
- Tree View to visualize how a game is laid out
- Rooms cannot be deleted - delete and reorder rooms
- Games cannot be deleted - delete and reorder games
- Copy and paste room layouts so that one does not have to recreate grids - done.
- Hotspots that can move around the room.

C.0.5 Notes from committed jammers about screenPerfect

For Arielle, the most engaged of the jammers, the idea of turning any touch device into a custom console controller, with custom buttons, is engaging. The more traditional the game developer, the harder a time they had with the idea that they'd be showcasing content with the narrowest help from the new tool. The filmmakers were very impressed with the ability to not touch a darn thing and have considerable success.

D Software Dependencies, GitHub Records

List of software used to develop screenperfect.

E Transcriptions of Public Talks

E.1 Transcript of PsXXYborg presentation, August 29, 2013

Rapture of the Nerds. I haven't had any internet at home for about a week. I figured that out today. This is what the van looked like, and we were all super proud of the van, which has gone to its final resting place.

So... all of mine is in text. I hope you like reading.... because what I do is text. I was not heavily involved in any of the theory parts of this project, or much of the artistic development parts of the process. I heard Feminist Art Game and Donna Haraway and I thought...MMM! Excellent! These are things I have studied! I have written a lot about these things! I can probably do this. And when I was invited to the project, we didn't really have a person to make the dual-screen part ... happen. And then we tried to hire a couple of them, and that did not go well. So the net result was that I would go over to Hannah's place, and I would hang out, and they would say "How are we going to DO this?" and I would say "aaaaaaaauuhhhm, I've heard of this thing! Somebody told me about it last week, we'll use that."

This is not really an easy or sensible thing to do. Most people come to coding [a new project] already knowing a language, and they're comfortable with that language, and that language has conventions and assumptions, and ways it works. The language that psXXYborg is written in is Javascript, which until recently was not available for use on the server. It was only available for use in the browser. I'm sorry if none of this makes any sense to you. You can ask me later.

[Learning Javascript and writing psXXYborg in it] is exciting and very very new, and it was on the basis of a lot of money. Google has paid a lot of money for the technology that drives psXXYborg to exist, and it really wants us to use it. So I had heard of

Node, but I'd never really touched it, and I hadn't written more than about four lines of javascript in my life before we started this project. But I said "oh, we'll use Node and that will be fine." I think maybe Cecily and Jennie understand how much of a joke learning an entirely new computer language for a single project is.

psXXYborg is about new ideas, so it runs using new ideas. These are ideas about the direction of the web, and how to use it. Google has very specific ideas about how it would like the web used, and we used some of those ideas, which are about interactivity and the Chrome browser, to make this thing work. It works in ways that it shouldn't work. It works using a programming language that was not designed to build streaming web servers, and it uses the very newest ideas to do that, so a lot of psXXYborg was about research.

Coding in isolation is difficult, because code is about making ideas into actions. Code is a language that is about assumptions, and it's about driving machines to do what you want them to do, to make your own world. As a technologist, coming up with new ideas that are actually useful to people is almost impossible. It's just very, very difficult. People will tell you that is not true, that they have wonderful ideas: the problem is that everybody else has pretty much the same ideas, because tech builds on tech, the way that lanugage builds on language. English picks up bits and pieces of other words from other languages like a thief and technology advances upon itself to develop new things. So coming up with something that's genuinely new using a technological driver is really difficult, because the people who are already thinking about technology are not thinking about technology for what it might make [the humane], they're thinking about making more technology [more stuff].

Without a project goal, there's nothing to write. If you don't have a thing to make, you're not making a thing. You're arranging some ideas on a page, and that's all. And in isolation, that gets very lonely, because you do need to concentrate when you're coding, you need to concentrate for gigantic blocks of time - it's a creative practice that way. It's very hard to hold ideas in your head for a long period of time if you have to go off to meetings and do other things. So this is a creative practice that's about isolation that cannot be driven alone. HARD.

psXXYborg offered me an excellent opportunity to think about how new, expensive, "free" technologies might be used to make hard things easier. Technically hard things. The magic in code is that it disappears. You only see code when something goes wrong, because otherwise, it simply acts as though it is a glass panel. And that is mostly what technologists want code to do. We pretty much want it to retreat into the background, because when it's centre stage and when it's on display, people become frightened of it, sort of like how if you put a word problem in front of a 13-year-old, they screetch and run to the other room. Just a problem. It's a thing that happens. So until something goes wrong, code is invisible.

The joy of fixing things is that when they go right, the code disappears again. So that's the whole idea about the invisibility of code. I'm working on that further for my thesis, because I think it's a big idea, and it's a good one to write down in the context of the arts.

There are lots of interesting threads in the disappearance of a language that drives the mechanics of your world. These are languages that build your medical devices, your cars, your phone: having them be invisible is an interesting problem the same way the underlying gendered assumptions in language is an interesting problem to overcome from the point of view of the legal system. It's an interesting problem. What I mean by an interesting problem is a problem that's worth spending a long time on, teasing out all the details. It's like a math problem that's made up of words. Not a word problem, a problem with the words. So, language assumes culture, Javascript assumes callbacks. That means you write it in reverse. It's sort of like being Ginger, you dance backwards in heels the whole time.

As a coder, what you write becomes real in a tangible sense, making interesting new things is difficult, thinking about what they might be is hardest, particularly alone - so, how do you fix this? You find a gang. A girl gang! The arts have an audience, but their audience is limited by technology [**LISA notes** If you have a particularly brilliant painter or a particularly brilliant graphic designer or an awesomely amazing musician, that is very fine, however, if they cannot record their information, or transmit it to a broader audience, then they will be broke. We know this is a problem. The next problem is how can we line up the technology so that they don't go broke after they do have the audience. We're still working on that one. It's a scary one.

Technology has a profit margin, but technology, because it builds on itself in a recursive manner, after education and privilege has given people the ability to write it, often has a very limited cultural outlook. People burn out in the tech field all the time. They burn out constantly. Google hires people for the express purpose of burning them out building machines. Because that's what we do. We write documents that are machines. They're laws. They're just laws made of a different kind of language. And they are breakable. There's an entire body of people who work breaking the laws of the language that we write into our documents. We call them hackers. They're very valuable. We need people who can break the laws to show us where they go wrong.

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. This is because of the aforementioned recursiveness of technology. Tech builds on tech, almost unquestioned, but art builds on culture and culture has a way of slipping around. When you are not looking at it, it moves in the dark. You think that teddy bear is sitting on your shelf but you wake up and it's at the end of your bed and let me tell you, you will scream. Culture can be seen as nothing but questions, questions like "How might a person be?" I think the answer is better.

Bibliography

Anthropy, A. (2012). *Rise of the videogame zinesters: how freaks, normals, amateurs, artists, dreamers, dropouts, queers, housewives, and people like you are taking back an art form* (Seven Stories Press 1st ed.). New York, USA: Seven Stories Press.

Bissell, T. (2013, September 25). Poison tree: an open letter to nico bellic about 'grand theft auto v'. *Grantland*. Retrieved from http://www.grantland.com/story/_/id/9719678/tom-bissell-writes-letter-niko-bellic-grand-theft-auto-v

Cixous, H. (1976). Laugh of the medusa. *Signs: Journal of Women in Culture and Society*, *1*(4), 875.

Galloway, A. (2006). *Gaming: essays on algorithmic culture (electronic mediations)*. Minneapolis: University of Minnesota Press.

Hunicke, R., LeBlanc, M. & Zubek, R. (2004). Mda: a formal approach to game design and game research. *Rutgers*. http://www.cs.northwestern.edu/ hunicke/MDA.pdf.