Captive Portal Art Machine:
the importance of accessible technology for artistic use.

Author:
Alex Leitch
Supervisor:
Prof. Emma Westecott

A thesis submitted in partial fulfilment of the requirements
for the degree of Master of Design
in Digital Futures
in the Faculty of Design of OCAD University
April 10, 2014

Declaration of Authorship

I, Alex Leitch, declare that this thesis titled, 'Captive Portal Art Machine:
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:

Abstract

Ontario College of Art and Design University

Master of Design

Digital Futures

Captive Portal Art Machine:

the importance of accessible technology for artistic use.

by Alex Leitch

18th March 2014

This thesis consists of technical design project to, on one hand, make it easier for video artists to assemble web-based dual-screen branched narratives, and then to install and display them consistently in environments without consistent access to the internet. The project examines what it means for technology to be accessible, and how that accessible technology is commodified in support of the arts.

The installation of works based on the technology took place during a two-year period between 2013 and 2014 in a variety of exhibition contexts. This essay consists of technical details of installation as well as conceptual supports for what accessibility means, from a feminist and game-art perspective.

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, Cecily Carver, 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 for the duration of this work.

For Adina...

TABLE OF CONTENTS

Declaration of Authorship **ii**

Abstract **iii**

Acknowledgements **iv**

List of Figures **ix**

Abbreviations **x**

1 Introduction: A Better Time-Based Installation **1**

1.1 Introduction 1

1.2 Designing Software to Power Experience 3

1.3 Interaction and Presentation in Game Design 5

1.4 Initial Approach 11

1.4.1 Federal Development Grant, game:play Lab, and collaborative artistic practices 11

1.4.2 Code and Theory 13

1.4.3 Game Design Research 15

1.4.4 Feminism, Cybernetics, and System Controls 18

2 Background, Theory, and the State of the Art **22**

2.1 Game Engines 22

2.1.1 Twine 23

2.1.2 Multiscreen Video Technology 23

2.2 Theory and Politics 24

2.3 Cixous, Embodiment, and the Game Jam 26

3 Software Design, Industry Engagement, and Hardware Design **28**

3.1 Software Design 28

3.1.1 screenPerfect Engine—Interface Layout 30

3.2 Design Research . . . 32
3.2.1 Artist Collaboration . . . 32
3.3 Development Methods . . . 32
3.3.1 Agile Development with an Artist Partner . . . 32
3.3.2 GitHub and Open Source Software . . . 35
3.3.3 Licencing . . . 36
3.3.4 Science Fiction Inputs . . . 36
3.4 Industry Engagement . . . 37
3.4.1 Game Jams, A Design Method . . . 37
3.4.2 Dames Making Games and Game Jams . . . 38
3.4.3 Bento Miso and Bento Box . . . 39
3.4.4 NoJam 2: Video Video . . . 40

4 Hardware Deployment in Limited Local Space 46
4.1 Public Installations . . . 47
4.2 Hardware Design . . . 53
4.3 Physical Deployment on the Raspberry Pi . . . 56
4.3.1 Problems and Complications in Display . . . 58
4.3.2 Subnod.es and Public Private Space . . . 59
4.4 Distribution of Work . . . 60

5 Conclusion 62
5.1 Conclusion . . . 62

A screenPerfect Installation Guide 67
A.1 screenPerfect Code and Documentation . . . 67
A.2 Device List . . . 67
A.3 Software List . . . 68
A.4 Installation and Site Construction . . . 68
A.4.1 Before Leaving For Site . . . 68
A.5 Troubleshooting . . . 69
A.5.1 On Start, Google Chrome Cannot Find Control Screen . . . 69
A.5.2 The Control Device is frozen, or the videos are not changing. . . . 69
A.6 Tidbits and Tech Notes . . . 70

B Annotated Literature Review 71
B.1 Literature Review . . . 71

C Game Jam Documentation 77

C.1 Interview Files . . . 77
C.2 Questions To Ask Game Jammers . . . 77
C.2.1 Games List . . . 77
C.2.2 Bug Discovery . . . 78
C.2.3 Features Requested by Game Jammers . . . 78
C.2.4 Notes from committed jammers about screenPerfect . . . 78

D Raspberry Pi Setup Documentation 79
D.1 Materials and Supplies . . . 79
D.2 Background for Linux Commands . . . 79
D.3 Setting Up The Raspberry Pi . . . 79
D.3.1 Windows 7 SD Card setup and first boot . . . 79
D.3.2 Configuring Raspbian . . . 80
D.4 Software Setup for External WiFi Access . . . 80
D.5 Installing Node.JS . . . 83
D.5.1 Why Node? . . . 83
D.5.2 Installation Instructions for Node.JS . . . 83
D.6 Testing Node . . . 84
D.6.1 Selecting Monitoring Software . . . 84
D.6.2 Installation of Node Modules . . . 84
D.6.3 Troubleshooting NPM installations . . . 85
D.7 SSH via Direct Ethernet Connection
and WiFi Internet Access . . . 85
D.8 Backing Up the Raspberry Pi . . . 86
D.9 Mount Your USB Flash Memory Stick To the Raspberry Pi . . . 87
D.9.1 Configuring Your Mount Drive . . . 87
D.9.2 How to Boot Mount External Memory . . . 87
D.10 Set Up a wiFi Hotspot . . . 88
D.11 Configuring HostAPD . . . 88
D.12 Configuring DNS access via `dnsmasq` . . . 89

E Appendix E: MIT Licence and Research Ethics Approval 91
E.1 MIT Licence . . . 91

Bibliography 94

LIST OF FIGURES

1.1 SNES Game Cartridge, 1995 12
1.2 Seizuredome, S. Kochavi, A. Shulz, 2013. A tent with video installation. 13

3.1 screenPerfect software communication model 29
3.2 screenPerfect NoJam editor, Asset Population tab. 31
3.3 screenPerfect NoJam editor, Exit Population tab. 31
3.4 A branched tree of rooms from Twine. 43

4.1 Hannah Epstein psXXYborg at VideoFag, 1995 46
4.2 Hannah Epstein, Alex Leitch, Sagan Yee psXXYborg at VideoFag, 1995 47
4.3 Hannah Epstein psXXYborg at FAC 2014 49
4.4 Hannah Epstein psXXYborg at FAC 2014, playthrough view 50
4.5 Alex Leitch Concept for collapsible projection space 2013 51
4.6 Alex Leitch Concept for portable arcade box, 2013 52
4.7 Abe Shulz and Sage Kochavi Seizuredome at FireFly 2013 57

5.1 Una Lee, Difference Engine Initiative Visualization, 2012 65

D.1 Raspberry Pi with functioning wiFi antenna 82
D.2 Raspberry Pi in Thingiverse case 82

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.

CHAPTER

ONE

INTRODUCTION: A BETTER TIME-BASED INSTALLATION

1.1 Introduction

What constitutes good design? As described by Don Norman in the book "The Design of Everyday Things," (Norman, 2002) design can be sympathetic to their users, or have psychopathy coded directly into the construction of design artefacts. In response to this, I have designed a web application called screenPerfect to display synchronous multi-video screens and branched video narratives, and then a hardware solution that relies on the Raspberry Pi hardware architecture for installation.

ScreenPerfect is distinct from pre-existing engines, written to encourage video artists to use their own skillset to explore what is possible in an interactive experience. ScreenPerfect's game-side interaction mechanics emphasize a classic branched narrative structure similar to that of FMV games, which can then be extended by any programmer to encompass new features. The editing and interaction mechanic are both 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 – in order to do so.

This project has several parts which will be explained separately. The first part of the project is an application to build branched narratives out of video files, coded in javascript on the Node.JS framework for distribution via common internet technologies. This portion of the research has to do with the idea of structural political resistance through technology - implicit control systems - and how human abilities can be expanded by using contemporary technology, which is made by trained technicians but then released to be used without a proscriptive definition of the final use case. Although the intention is to make games, the software does not prevent, and even encourages, other uses. Users make use of the engine, developed through one specific user's design practice, to design new things. The first section of this paper presents how such an engine might be applied to video works.

The second part of this paper addresses how to perform user testing via a game jam format, a type of design charette in which users are given access to tools and asked to produce art with them. In that section, I document how No Jam 2: VideoVideo worked, the industry paired advantages to new engine development, and how this expands community ties for local artists working in a medium that frequently demands more collaboration than traditional, less dynamic art works.

In the third part of this paper, I address the issue of new media art and display. New media, particularly time-based media, is very challenging to display and support. Even low-end computers are bulky and expensive to deploy, so I have asked how we might circumvent traditional white-cube galleries while restricted by the requirements of internet technologies. In this part I detail how to build a server on the raspberry pi platform, a microcomputer, to deploy web-based applications to localized, internet-restricted spaces. In conclusion, I argue that we need technology to belong to its users, instead of pursuing an exclusive reliance on mass network technologies. I feel that there has always been room for the technical in art, and while good

technology fades into the background to leave only the artwork on display, the technology we choose to use provides the frame of what can be pursued.

1.2 Designing Software to Power Experience

The arts and the humanities are the technical name for the fields of work that produce both culture and its record of its culture. We use computers to do most white-collar office work. Machines automate and extend our ability to speak in repeatable patterns (Glanville, 2014). Repetition is a key component of mass production. Looked at as a tool, a computer is not so much a hammer as it is an ongoing negotiation - the user must decide what the black box means (Glanville, 2014).

The use of computers as tools requires a specific skill set that is as unique as the skill set of using a paintbrush. A key aspect of computer skill is a comfort with curiosity: digital tools change all the time, and many of them are not well written. Software is frequently unreliable, and hardware more so. Almost all software tools require time invested in skill acquisition before content - art - can be produced, and this time is expensive. The construction of the tools is an art, because when manufacturing a tool, it needs to be easily used, but it also needs to do something in a predictable, reliable way. A good digital tool should encourage rather than impede expression.

Artists, as a rule, have their own working methods and vision for their work in advance of picking up any new tool. For the purposes of this work, it is assumed that users have their own vision independent of the tool itself, which can be expressed, expanded, or extended by the use of a new tool. Ideally a tool vanishes in its use. It is a multiplier of force, where force can also be construed as capital, which could be the capital of knowledge and cultural membership, an availability of time to develop knowledge and then use the knowledge to operate in a space, or

a more conventional type of currency: money. Television and video are a format traditionally requiring capital to access. Culturally, video works - film and television - express mass ideals to mass audiences. YouTube, Vimeo, and visual FX production continue to drive technological innovation in systems development. Here I conflate instant film produced on a mobile phone with years of work in film production because the internet has flattened the effort it takes to consume media, if not its production. Most large-budget films in 2014 movies have at least some visual effects post-production. Some, such as "Life of Pi," are almost entirely composed of vFX and compositing, practice that requires years of skilled labour (Netter, Lee, Womark & Zemeckis, 2012). Acquiring editing and image-design skills to work with moving pictures is an expensive and time-consuming pursuit, but putting any video at all on the internet takes almost no effort in centers with access to broadband internet. The advent of Vine and YouTube has democratized video to the point where it can be traded as words once were – or pirated, as the popular works of Dickens commonly were in early America (Castillo, 2008). Streaming media sites present an awkward effort to access larger works, but not by much, and Netflix's distribution model means that a year of filming work – in the case of House of Cards ('House of Cards', 2013), more than twenty hours of original television – has been redesigned to be consumed at the convenience of the audience. Spoilers now encourage watching the series inside a week, rather than over a month, or a year.

This moves the value of a given video experience from the control of the video producer to the audience. Rather than being restricted in a viewing experience to a theatre, the audience now decides where and how to consume popular works. Video is consumable on every kind of screen, especially on the smartphone screen. This permits acquisition of a broad audience, even as it shifts the context of the work.

A single smartphone in 2014 is more than powerful enough to supply most of the serving needs of previous video works, including branched narratives that once required many VCRs and

multiple screens. As things become more accessible, according to Walter Benjamin (Benjamin & Arendt, 1968) they lose their aura, their singular magic, which means that accessing that magic becomes more challenging as the distribution of the work becomes easier. Partly, this seems to be a process of decontextualization: no matter how good a given film, it may be better in a theatre, en masse, because the theatre, its particular context, makes the experience of the film singular, even as the film itself is limitlessly reproducible.

The question in this context becomes: how to best retain the monetary and chronological capital of artistic tool use, vision, and creative practice, while restoring the aura of presentation required to engage with art on its own terms? How to expand artistic tool use into new means of expression? How to make use of contemporary methods in a way that may remain accessible and on display for years to come?

1.3 Interaction and Presentation in Game Design

In the context of this paper, my frame for the term "art" is digital games as interactive experience. In this case, technology design and game design must be considered together, in that the design of an experience is dependent on its technology.

That games are art, or can be art, has been popularly contested. Famously, in 2005, Roger Ebert took the position that games could never be art (Ebert, 2005). He recanted this statement in 2010, with a public admission of bullheadedness and a confession that he simply did not wish to engage with games as a form (Ebert, 2010). The debate has been reasonably settled, to my mind, with the rise of conferences such as Different Games NYC and Indiecade, with experiences at all investment levels to explore a variety of human experience. Manufacturing those experiences within a game is a challenging task, not least because game production can be expensive, and thus inaccessible.

Games, particularly the subset of video games known as triple-A or AAA, are expensive to make and require a team of people to produce. This has been seen as restricting the degree to which the stories these experiences communicate can be personalized. A large budget requires a large payback. While there are counter-examples, many triple-A games need to be able to make back their large production budget, which restricts their intended audience to those who can pay to play. Games such as 2013's Saint's Row 4, ('Saint's Row Four', 2013) are few and far between. SR4 is known for its witty in-jokes and careful presentation of balanced gender roles and skin colours, while simultaneously emphasizing violence and super-powers as the main interaction system of its world. The joke is that no matter how well a major company cooperates with the window dressing - the presentation of character models, a finale in which one rescues Jane Austen - games are about violence. This is still only one story, dictated by its engine and the necessity of selling enough copies to cover the budget.

The games themselves utilize engines which are generally private or closed-source. An engine is the software that delivers the physics and scripting that creates a game world to be manipulated. Many are closed-source and privatized, and therefore expensive. All of the engines as they presently exist require not only the skill of framing and lighting an engaging experience, but also many other skills - character modeling, animation, colour theory, programming or scripting, and sound. This creates a design challenge: How to best reduce the barriers to entry in game-making to encourage new voices?

A new type of game engine is one possible approach. An engine is the software that drives all interactions in-game. Assets, such as artwork, music, animations, and scripts to dictate how these assets are integrated, are all added to an engine that provides a framework to drive any given game. Some engines encourage more experimentation in design approaches than others. The Twine engine, for example, is designed to provide branched, highly-stylized text narrative to a web browser, and it has been adopted by a user base interested in telling detailed stories that

are highly personal - the sort of work that cannot always be addressed by games with a bigger budget. Twines are limited in form but not in scope. The Unity engine provides traditional assets and scripting, and has been used by independent developers to produce games such as Gone Home ('Gone Home', 2013), a work about a missing family mainly told through examining objects and listening to music. Twine takes advantage of an author's skill at pacing and writing to divide a narrative into a choose-your-own-adventure work, paced through timed links and designed to take advantage of the detailed design possibilities of text in the browser.

Independent games are typically distributed through the internet, or rely on the internet for their entire lifecycle. This is problematic, not only because the shape of the engine dictates the shape of the experiences that can be produced. The internet is owned, mainly, by very few extremely wealthy people, and the technology is frAgile, in that it is reliant on many external factors to survive, and on machinery with decreasing life-cycles (Paul, Leeson, Manovich, Sawon & Simon, 2013). There are many ways to lose access to the internet, from legal means such as France's HADOPI laws, by which whole households can have their access cut off, to a simple lack of bandwidth in an installation space. This causes problems for both exhibitions and archives, as art based on access to the external web can vanish with no warning. This is also unacceptable for institutional collection. The availability of web art combined with its unreliability devalues the work of the artists who have created it. Art that relies on the network is simultaneously omnipresent and vanishing, capable of accessing a mass audience and disappearing at the moment when a local audience is available.

Audience definition becomes important in this context. A browser-based or internet- distributed game has the possibility of reaching a very broad audience - millions of players. The artist has no guarantee of the context of their work in the view of the audience, beyond that it is likely to be screen-based, viewed on a personal or work machine. Perhaps this works: Internet artists such as Cory Arcangel – exhibited at the Whitney in 2011 - and installation works

backed by major museums, such as The New Museum's Rhizome, have seen success and popular uptake. Whether or not screen-based art as it presently exists is effective is outside the scope of this paper, however. Within the scope of this work is that digital work is difficult to display outside the context of this mass market. It is also difficult to record the things that make more impressive offerings of net art – the massive collective performance art piece "Twitch Plays Pokemon" ('Twitch Plays Pokemon', 2014) for example – special, outside of being part of the specific moment in which they happen. If electronic art is to be included in large collections, or displayed privately, or reproduced such that it benefits mainly the artist rather than the distributor, there needs to be a means to display it that does not rely on external resource providers. This includes the easily-considered difficulty of network providers as well as the more challenging to contextualize power grid. Both are unreliable in the permanent sense, where the relatively small amounts of power and information (which are sometimes the same things) can actually be supplied locally.

The localization of a broad audience - how to supply a thousand or ten thousand people simultaneously with a single experience - is outside the scope of this paper. My design work instead addresses the question of how to bring a work built for broad distribution into a narrow context for better engagement via a system of resilient display. This system uses local resources rather than relying on the constant availability of a global supply.

A local supply of exhibition resources makes sense from the point of view of customised presentation of work. By controlling the media server and internet protocols for service directly, an artist can install their work where they like – including in contexts that would otherwise not have access to the broader internet at all. The idea that the internet is watching us as we watch the internet has gained potency in recent years, particularly with the revelation of data harvesting by major first world spy agencies such as the NSA and the GCHQ (Ackerman & Ball, 2014). The rise of a true panopticon system leads to questions about the implementation

of technology. Where Foucault's Discipline and Punish was based on theoretical constructs, and the ever-popular 1984 posited a dark totalitarianism, 2014 saw instead the Ukranian government text message protesters that they were being watched for their part in civil disobedience (Merchant, 2014). These are no longer theoretical problems – they are pragmatic systems, a part of the landscape of fact.

As such, art which wishes to make use of the internet has the opportunity to provide a matter-of-fact system of resistance, by refusing to permanently link to the main channels. The privacy of the user can only be assured by permitting them the ability to reliably control distribution and display of their work. Because the internet is written in privileged, private languages – code – it becomes interesting to examine this idea, of privacy and resistance in the public sphere, through critical theory concerned with the play of language. The most interesting of these for the purpose of this work is Helène Cixous, whose most famous writing plays on language itself as a site of resistance and desire.

There is a sense of play in code: there are many, many ways to achieve a topically identical personal experience using software. The software choices that underly those experiences are largely invisible: they are the good housekeeping that allows the experience to happen at all. In order to learn the language, it must be revealed not in a compiled state – the state desktop language occupies – but rather in an uncompiled, or "interpreted" format. Web technology relies almost wholly on interpreted code, rather than compiled code. Viewing source code in a browser has been possible for twenty years, since NCSA Mosaic 2.0a3 [6]'s April 6th 1994 release ('In the beginning there was NCSA Mosaic....' 1994).

This ability – to see the code, change it, post, and immediately view the effect of changes, without paying extra for a compiler and an assembly system, represents a sea change in how developers could learn about writing software. It meant the difference between a purposeful

investment and the ability to pursue learning on a more personal schedule, and from there, an expansion to self-expression to the web we see today. With the advent of developer tools included standard in browsers, it has become even more straightforward to build and test software ideas quickly in an environment provided on every operating system.

This relative accessibility on an independent system is key to the broader accessibility of code as a language for a wide audience, and from there, to a diversity of cultural production within this new creative sphere. The cognitive load of the inheritance of computer science as a discipline is much higher than the cognitive load of simply writing a program that works. It blocks access by requiring funding and time to learn something new. People who have little time cannot afford to acquire the skills to use a new and complex tool. 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 pro- duction 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. The bulk of technical authorship acknowledged through formal means – peer review at large universities, high positions in large corporations – has been heavily restricted in a systemic fashion that recreates the society that first permitted these organizations to exist. This makes small resistances and large capture of technological means important. 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. The public representation of self becomes limited, and the experiences on offer follow these limitations, becoming narrower, and ultimately, perhaps less interesting.

The problem of interesting experiences is not trivial. Video games, especially video games featuring strong narrative and a lot of player agency, 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 make 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.

1.4 Initial Approach

1.4.1 Federal Development Grant, game:play Lab, and collaborative artistic practices

The initial code of screenPerfect came about as part of a collaborative research and development project in OCADu's game:play lab to produce a vision of how dual-screen game artworks might work going forward. The original software powered a game called psXXYborg (Epstein, Leitch & Yee, 2013), made by Hannah Epstein under the supervision of Emma Westecott. From there, I became curious as to how we could transform the engine software to include game-editing tools, to encourage a wider range of video artists to use the software. This became the basis of the initial portion of my thesis work, the screenPerfect engine. In order to generate sufficient games to demonstrate the software and its potential, and to figure out where the software could be improved, we then partnered with Bento Miso co-working space and Dames Making Games in a mutually beneficial game jam called No Jam 2.

No Jam 2 featured both an editing segment and a re-architected version of screenPerfect that uses Bento's new language, Daimio, designed mainly for open use on the internet. After the

jam, the games were collected, with their resources, and screenPerfect was forked to become two separate engines. The original engine was retained for displaying works to that point, and a new engine called iV was created by Bento from the idea of screenPerfect's operation to promote ease of access for Dames Making Games, a feminist community group run at Miso by the same developers who worked on the engine.

In my development of screenPerfect, I have made use of the Agile method of software development, which is explained in Chapter 3. The Agile method of software development is based on the idea of delivering working code in advance of documentation, and putting the user ahead of the planner in software design. I have included a summary of design method in Appendix A, The Agile Manifesto.

FIGURE 1.1: SNES Game Cartridge, 1995

As such, my method has been to focus on user-centered design to overcome the difficulties of migrating skills involved in one creative practice, video design, to a second practice – game design, or user-centered narratives. ScreenPerfect's initial layout and idea of operation was designed with a video artist – Hannah Epstein - who laid out an idea for how an interaction might work. The interaction was duly written, then released, refined, and redeveloped for a broader audience via the inclusion of editing tools, who then created games of their own. The hope is that each new version of the tool will generate a useful echo chamber, amplifying and

iterating new ideas and tools even as it makes advanced technology easily accessible to content producers.

The toolset can then be released and left for artists to use and analyze, and can be expected to run privately on optimized systems similar to game cartridges from the 1990s (Figure 1.1). Rather than actual cartridges, these installation kits take the form of locked SD cards or USB sticks, which can be relied upon to hold their data in a reliable, mobile format. This means that artists will be able to install and display their own games independent of any central server, free of what the technician might decide is the context of the work. This should permit reliable installation of completed works even in remote contexts – a forest, for example, or a desert (Figure 1.2). This is distinct from other systems in that it is built using contemporary technologies, but also in that it is built with an eye to permanent, disconnected installations that rely on contained frameworks and simple, contemporary script languages, rather than on translations of preexisting software such as Java.

FIGURE 1.2: Seizuredome, S. Kochavi, A. Shulz, 2013.
A tent with video installation.

1.4.2 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. The project emerged from pairing with Hannah Epstein,

who had an idea for a game featuring dual screens, but access to only commercial tools with limited potential for extensibility. Rather than producing exclusively an engine for a single production, it seemed reasonable to produce an entire editing environment, which would place the control of the final experience into the hands of the artist. This would in turn produce an engine that could be relied upon for public performance, but also an opportunity to easily make more than one game using this particular series of interactions. Video artists are already skilled technicians with a grasp of how to set a scene, so it seemed positive to extend their ability to easily piece together branched full-motion-video game works that would then display on the internet.

Part of the motivation for this engine is that commercial engines tend to prioritise commercial distribution and mass experience, whereas artist installations tend to prioritise the direct experience of a specific work at a specific time. Although there are commercial FMV engines, they do not permit easy access to multi-screen synchronisation, and cannot be accessed offline. This meant that presenting the works developed using these systems meant agreeing to advertising, or to sign up for an account with the company, rather than being able to turn on an appliance and serve an art work.

The engine that drove psXXYborg worked well in practice, but the installation of multi-screen technology caused many issues. The technology, designed for multiple screens, required a great deal of technical support, as well as a very specific server that could be easily damaged - the mac mini - and keyboards, mice, as well as various power supplies. The total installation process for an early build of psXXYborg/screenPerfect has been detailed in Appendix A, "screenPerfect Installation Guide." The technical limitations of repeated equipment rental included tablets with permanent Google logins attached, inconsistent software environments, and web standards unevenly implemented across environments. Mobile computers in particular do not have any kind of consistent support for HTML5 video standards, the most common problem being that

video will not play on load in some, but not other, HTML5 environments. This meant that in addition to custom software, consistent hardware was required for our installations.

The code of screenPerfect is written wholly in javascript via the Node.JS software framework. Node is a server environment intended to permit developers familiar with javascript to write code for both the browser client and the server without switching computer languages, as has been common practice until the release of Node. ScreenPerfect's design concept has proven popular, and in order to simplify the interface of the editing tools, it was forked by my industry partners, Bento Box—Miso – the business hosts of Dames Making Games - who are interested in the idea of new game engines as a use case for their language, Daimio (Box, 2013). Bento Box produced a clean variant of my editing interface in order to help me run a game jam to gather samples of what an FMV game might look like. In return, I gave them full permission to convert screenPerfect to their own language and to extend that engine into a new application called "iV."

The critical theory that underlies my 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 (http://www.dmg.to), and as part of a wider feminist research network (SSHRC-funded Feminists in Games (http://www.feministsingames.com)), 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.

1.4.3 Game Design Research

FMV games are an old format, relatively speaking. The FMV began almost as soon as chapter selection became available on the laser disc systems of the 1980s, with 1983's game "Dragon's

Lair" by Rick Dyer and Don Bluth the first entry in what would become a strange sub-genre. FMV was successful through 1984, but quickly failed due to the expense of laser disc systems and the relative cost of game development, with the 1985 Halcyon system costing $2000 - adjusted for inflation, $4,347.88 in 2014 - and offering only two games. The most well-known FMVs outside of Dragon's Lair are Night Trap, released in 1992, and Phantasmagoria, from 1996 - Night Trap went on to become part of the congressional hearings of offensive video game material, along with Mortal Kombat, the first widely popular fighting game to allow people to rip out one another's spines. Phantasmagoria was better known as the first adult-oriented game released by Roberta Williams, famed for her involvement in Sierra's King's Quest point-and-click adventure game series (Wolf, 2012).

FMV and branched narrative games differ from cartridge-based action games in that they do not typically feature the same immediate feedback of a score going up and the instant player controls of a more typical 2D or 3D action game. Instead, players select what will happen next at key intervals. Older games are easy to display, so long as working hardware can be found, because they rely on consistent materiel for installation. New media interactive forms, particularly those on the internet, live in a more malleable format. They can change, or be taken down, at any time. The gameplay experience of a console game can be had even when disconnected from the internet – in fact, the Nintendo 3DS, a pocket console, outsold every other system on the market in 2013. It is speculated that its success is due largely to the fact that the 3DS is a portable system that does not connect to the broader internet unsupervised. This reliability is something that is rare to find in more complex computers: sometimes, as argued by Don Norman in "The Design of Everyday Things" (Norman, 2002), it is best to have a single thing do one thing really well.

FMV games remain interesting enough to engage fans. They have been recreated using Youtube and preserved from laser discs and DVDs ('LASERDISC ARCADE PROJECT AKA

CLASSIC FMV GAMES 2.0', 2014). Phantasmagoria can easily be found in complete playthrough on Youtube, where the annotation system makes recreating a point-and-click environment trivial (https://www.youtube.com/watch?v=oAXC-MwfpHA). Nonetheless, they were heavily systems-dependent and are tricky to develop, given the difficulties surrounding copyrighted video works and public distribution - how to transmit something that contains so many different pieces of video information?

Another example of portable electronic interaction is the smartphone. Mobile is a huge segment of the market, excellent for text messaging, talking, and playing games that separate an audience from each other. Initially this technology seems not so great for bringing people together in the same space. The privacy of the phone has been seen as undermining or distracting, but might be seen to have instead a lot of potential: people examine things one-on-one with their devices, and then will share them with their peers.

Smartphones are inherently private systems used in public places. This leads to a set of assumptions on behalf of interaction developers: an application is a private thing, paid for, and downloaded to a private space, whereas a web page is a public resource that can be viewed in private on a phone. A smartphone is also a single encapsulated controller, with all necessary inputs provided by its touch surface. For interactive artwork, this means that some assumptions can be made.

The first is that the audience of an interactive art piece is likely to be familiar with how to interact with a touchscreen, but also may be distracted. It cannot be assumed that they will download or pay for an application sight unseen for the sake of art, because that would constitute an expenditure of resources without reference, but they can be asked to go to a web page. People commonly use their phones in public, and therefore it seems reasonable to ask

them for the minimal engagement of looking at something specific, this time art served only within the gallery.

There are already games that aim to subvert this separation, and systems have been built to take advantage of the power of pocket computers. A notable effort is "Spaceteam," (Smith, 2013) a 'Simon-Says' game for teams of up to four players. The application pairs to itself across phones by using a common network connection, and players in the same physical space cooperate to pilot a star ship. This allows players to make use of a device with which they are already comfortable to cooperate and share an experience.

This shared experience makes it possible to privately host a public space. screenPerfect, a web application served locally, takes advantage of an assumed set of smartphone users in galleries having access to the internet in their pockets. The internet is both bigger and smaller than the wider network – the internet that includes Google and other multi-national internet corporations. Rather than relying on the external resources of remote servers, screenPerfect provides a private wiFi point and what is called a "captive portal" to let players pair with one another and the server, control a large screen, and interact with a piece of video art in a localized area. This means that an artist can control the exhibition space for their work, design the experience of the work, and ensure that their audience will experience the work in a context that makes sense. It also ensures that technicians can access the underlying engine should something go wrong during the installation.

1.4.4 Feminism, Cybernetics, and System Controls

This work is related to various texts of feminist or woman-oriented critical theory: Haraway's "A Cyborg Manifesto," TIQQUN's "Preliminary Materials Towards A Theory of the Young-Girl." Haraway's "Cyborg Manifesto," from 1989, reads in relation to the world before the internet,

where TIQQUN's commodification of women ten years later – from 1999, shortly before the first of a series of devastating economic crashes – more neatly quantifies the value of young woman as commodity (Cixous, 1976)(TIQQUN, 1999). Cixous' work seems an interesting way to look through the construction of self that the internet provides to people who would like to package and produce an image of themselves. Where Haraway would rather be a cyborg than a goddess, and TIQQUN insists that young women—perfect stereotypes are the ultimate product of empire, Cixous insists on an individual self-determination and presentation, even when such seems impossible in a world where the tools for communication are totally controlled from a distance. Cixous insists on rebellion with a sense of play, and on this being the responsibility of the individual even when it seems impossible.

All of these texts have complicated thoughts about embodiment, which I have chosen not to address, as I am more interested in embodiment via the idea of a piece of writing having a perceptible effect. Code has this ability. When you touch a screen, writing – interpreted or compiled – controls what happens next. In this context, the *écriture féminine* can mean literal social breakdown, or a very physical change in the environment. Cixous' concept describes a space where women required to write, even using imperfect language or tools, least they be written out by the dominant voices that surround them. Her work, produced in the 1960s, describes a deeply embodied form of resistance to a language that describes what women want purely in context with what society wants for their bodies. The ideas are vividly expressed in relationship mainly to sexuality, but desire runs deeper than that: desire can include the desire for agency, for authority over oneself and one's life. Cixous states that to write is to write oneself into history, or into being (Cixous, 1976), while playing all the while with the idea of flight versus theft. Piracy, associated with copyright protections which have done nothing to limit distribution of video over the internet, has been characterized as theft - *voler*, in French. Women who code, similarly, are designing works that must simultaneously be pragmatically

functional, and may express totally different priorities than those implied by the authoritative structures present in the language.

In a practical sense, when one produces a work via a creative practice, that work expresses something of how one thinks. Cixous provides a ripping manifesto in *Laugh of the Medusa*: those who are different should produce work that reflects and presents what they want to see in the world. This is a position of resistance through joy.

This approach addresses woman as an alien construct to the more conventional world of technology, which has become associated with a masculinist performance. This is unnecessary for the pure structure of good rules and the development, through that, of good software. This construct is relatively recent, as Nathan Ensmenger discusses in his work on the systematic exclusion of women from programming as a trade (Ensmenger, 2010). Although artists are central agents of production of the invisible yet tangible value of the culture industry, they are not always the prime beneficiaries of the financial system that backs, stores, and distributes the results of that capital. This is capital as both skill and capital as resource distribution: computers are expensive.

In this instance, the artist is a specific person: video artists and cinematographers, who have a wide array of highly-trained skills that produce incredible images, which can then translate into FMV games. The software developer is both an administrator - a designer of forms, processes, and workflows - and a creative collaborator in their own right, as their code will dictate how the audience engages with video works that respond according to programming. A good engineer codes an experience that is engaging without being visible, a body of work that recedes into the background to better present video works in new contexts. Paired together, new works can be produced that take advantage of both sets of technical facility to enable new, different stories to be produced and displayed.

To test this idea, I have approached people to produce videogames 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 games in a limited window – and then the results are compiled for display into an arcade machine for presentation.

CHAPTER

TWO

BACKGROUND, THEORY, AND THE STATE OF THE ART

2.1 Game Engines

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 digital experience.. Traditionally, game engines are used to produce 2D or 3D experiences using 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 gameplay.

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-emphasize 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 digital gameplay.

2.1.1 Twine

Twine is a game engine that allows designers to build HTML5 text narratives that branch into a choose-your-own-adventure game. screenPerfect was inspired by the popularity of both Twine and video on the internet. Twine encourages expressive type styling and elements of multimedia, including music, and game screens, but does not require these elements for a complete interactive narrative. Twine did not yet support video narratives in 2013.

The Twine engine was popularized by DIY gaming celebrity Anna Anthropy in her 2012 book Rise of the Videogame Zinesters (Anthropy, 2012). Since then, hundreds of Twines - the adopted term for narratives built in Twine - have seen release.

2.1.2 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 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. screenPerfect relies on Node.JS, which is based on Google's V8 engine.

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 forty thousand dollars per installation, which makes an inexpensive alternative appealing from a creative standpoint. screenPerfect permits people to use existing hardware to synch multiple videos to one set of controls. This is also distinct from another related tool, ChromeCast, that allows people to wirelessly pair a television with a touchscreen for control and consumption of the touchscreen at a larger size.

2.2 Theory and Politics

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

> "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, 1976)*

In this passage, Cixous chides her readers for not giving themselves the permission to write, because writing is reserved for those who might be published. This is similar to game-makers who might not produce, merely because the engines are inaccessible, or distribution unlikely. Women have had a long history in technology. Ada Lovelace, daughter of Lord Byron, has been identified as the first programmer (Plant, 1997). 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 for in both programmers and game designers. Both roles are responsible for rule systems that will dictate a predictable result.

In a straightforward way, ladies may not possess uncomplicated positions of economic advantage within a patriarchy: to do so would be to "have it all," a famously complex desire which is reanalyzed every year in popular press. The balancing factor is housework and the social expectation of family, which still places a gendered burden on women to produce both children and home. This system of expectations re-creates itself in each new trade as it arrives, in fields as far apart as World War 2 factories (Summerfield, 1984), Victorian mills (Baskerville, 1999), and lately, code factories in Manhattan and San Francisco versus one's own kitchen, canning (Schulte, 2014). 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 is broadly accurate, but not specifically. Computers are a tool, and the way that the tool is presented, held, and used, dictates the results. A computer - a machine for automating an interaction - can be made simpler or more complex to use. The device may be changed to an amplifier of force, made more opaque, or made clearer for those who choose not to learn to code, but can still understand systems of logic. If this then that is not a complex instruction set. The complex instruction sets should, rather than being encouraged to control people, be developed to be under the control of people.

screenPerfect has been designed to present the idea that a simpler system will result in a more diverse body of work. It dictates nothing whatsoever about content, presenting instead a simple system of switches that permits the author the broadest possible control over simplified

interaction sets. It is implied that these interactions will lead to a coherent narrative, but it does not dictate what content an artist might use. The voice of the artist is brought to the forefront of the work, rather than the voice of technology.

By simplifying the process of game design and displacing its nexus from the computer to other design tools, technology is repurposed to be one tool among many. This displaces technology's primary position and refocuses the work on the intent of the artist. This is an implicit system of resistance to the narrative of technology: people can once again tell computers what to do, and extend themselves via their tools.

2.3 Cixous, Embodiment, and the Game Jam

Many of the participants of the game jam discussed in Chapter 4 produced, with the simple yet powerful tools provided, narratives centered on their own embodied or disembodied experience. The only group to resist doing this were the younger OCADu students who came to the jam via a game design practice, rather than a filmmaking, animation, photographic or other new media experience.

Of the games produced and finished, PornGame by Max Lander is directly about the experience of sexuality as it applies to a machine. Grimoire is about the loss of personal control following the finding of a "grimoire" or textbook. Kill Fuck Marry was about bad decisions when it came to dating and sex, OM about a practice of embodied mindfulness, and Cyborg Goddess went straight to Donna Haraway's Cyborg Manifesto in a literal interpretation. Glitch.95, though mainly interested in the glitch aesthetic, is a depiction of the beginnings of an internet relationship. Puppet story is about a genderless creature coming to life, Pinocchio-esque - or perhaps more Galatea, though that is projection.

This is simply an observation, but the most personal and body-centric stories came from this jam, which was nominally about only the use of a tool to express a new format of interaction software. This seems tied to Cixous' assertion that 'You can't talk about *a* female sexuality' and her immediate followup that 'Time and again I ... could burst forth with forms much more beautiful than those which are put up in frames and sold for a stinking fortune' (Cixous, 1976). This is the core of what games critics are discussing when independent games, or new art forms of any type, are being spoken about publicly. Cixous's conception of masturbation and writing as a unified activity practiced in secret can be easily associated to the idea of imposter syndrome, the thought that one could not possibly be "good enough" to break into a creative or high-level industry Chance and O'Toole, 1987.

Imposter syndrome is a major problem in the technology industry, and in games even moreso: because code is a high-level creative practice, with a great deal of material reward when done properly, and games are similar, the stakes are high. I have conflated games and code because both are systems of organized rules: game design is a type of code, where one purses conditioned response and engagement through good user experience design. It is easy to back off from both practices or to pursue them exclusively as a hobby, rather than believe in them as a trade. Does one wish to join the commercial order? Is it necessary to be validated, to have commercial success, or is it simply another demand on the work, that the work sell to prove that it is fundamentally worthwhile? These questions are outside the scope of this paper, although my personal opinion is in line with Cixous: whether the money follows or not, a plurality of voices in any creative practice is important for its own sake.

Cixous' throwaway line, of "arid millenial ground to break" seems especially poignant in light of the generational nickname of the jam participants.

CHAPTER

THREE

SOFTWARE DESIGN, INDUSTRY ENGAGEMENT, AND HARDWARE DESIGN

3.1 Software Design

A software interface is the part of the software that a person interacts with directly where a software engine is the part of the code that detects and defines what a computer can do with that interaction. The interface of 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 potentially opaque. screenPerfect's roots are as a software engine, which takes user action 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.

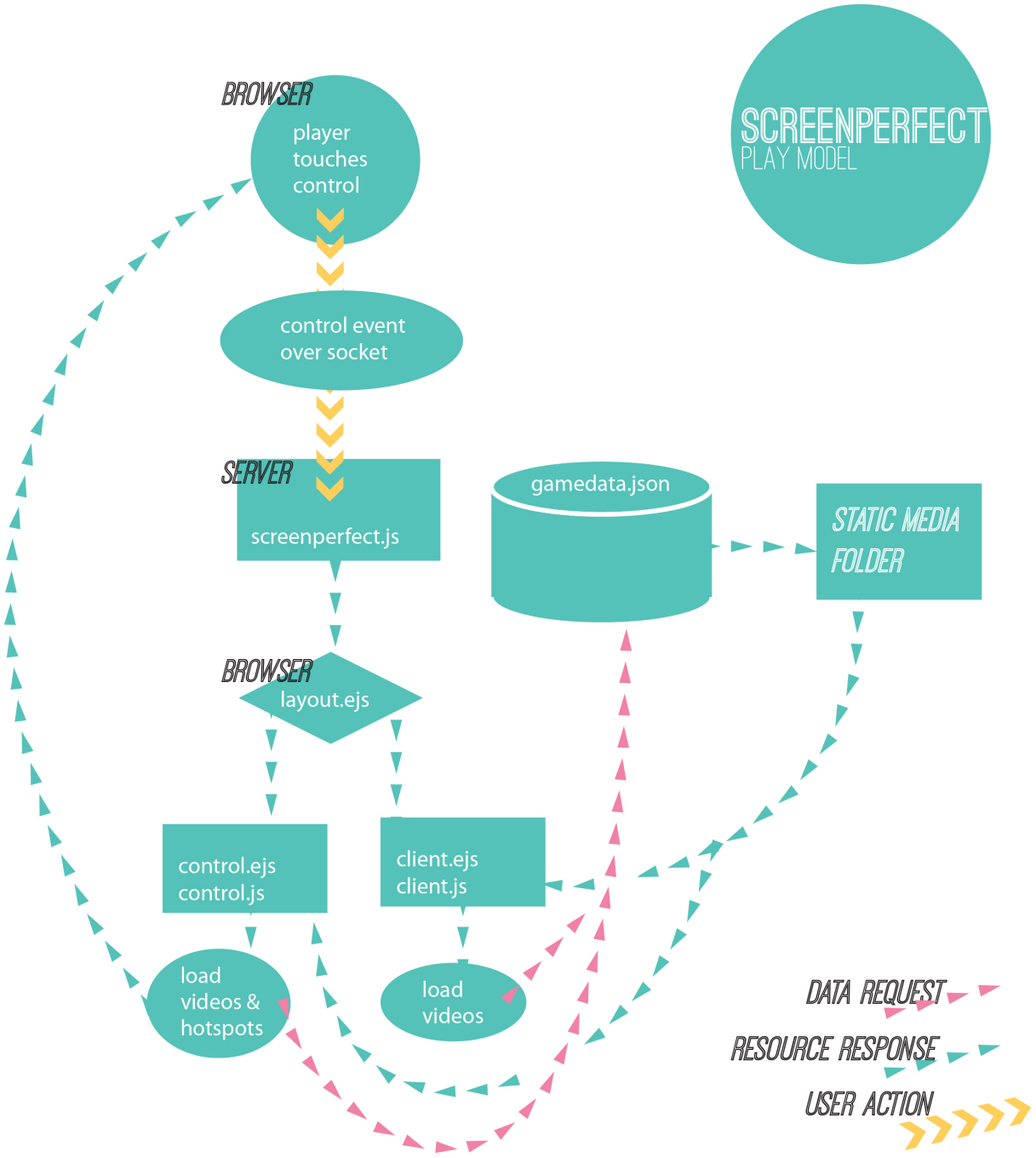


FIGURE 3.1: screenPerfect software communication model

3.1.1 screenPerfect Engine—Interface Layout

In the case of screenPerfect, the interface is laid out in three parts. The first part is the setup screen, 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.

The layout of screenPerfect's editing tools did not work very well for authors who were not already part of the prototyping process. Therefore, as part of the extension of the software for NoJam, Bento Box——Miso reauthored the editing interface of the software. The final editing layout is clean, though less expressive than the original design. Rather than hidden tabs, everything is displayed on launch. This permits authors to see their video files during the entire game editing process, which greatly speeds game creation. What follows are screencaptures of the pre-fork game jam variant of screenPerfect.

FIGURE 3.2: screenPerfect NoJam editor, Asset Population tab.

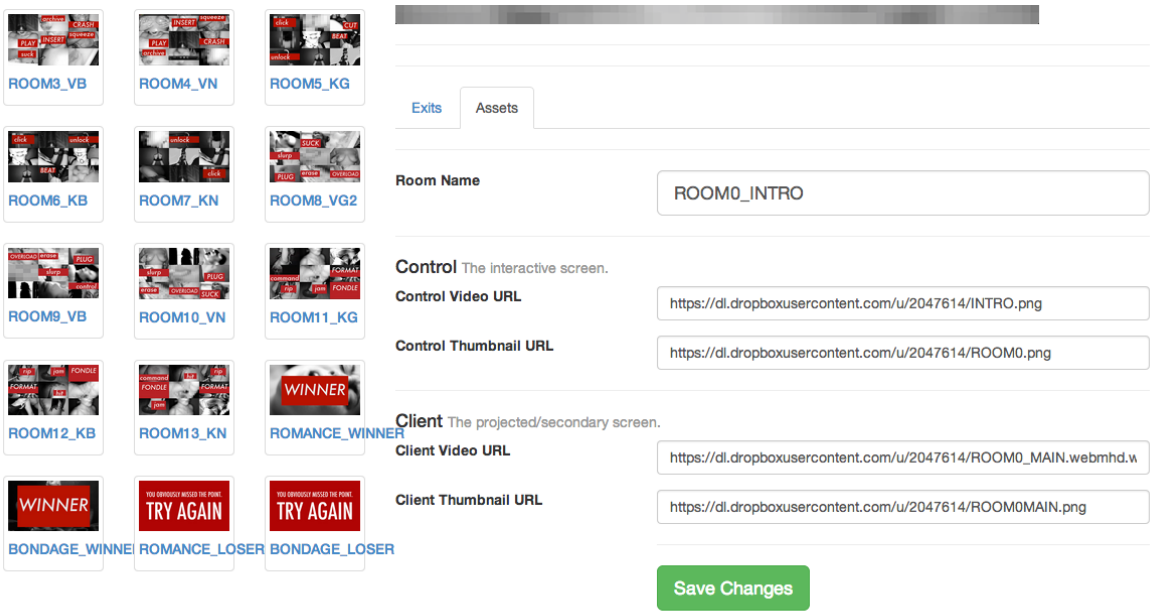
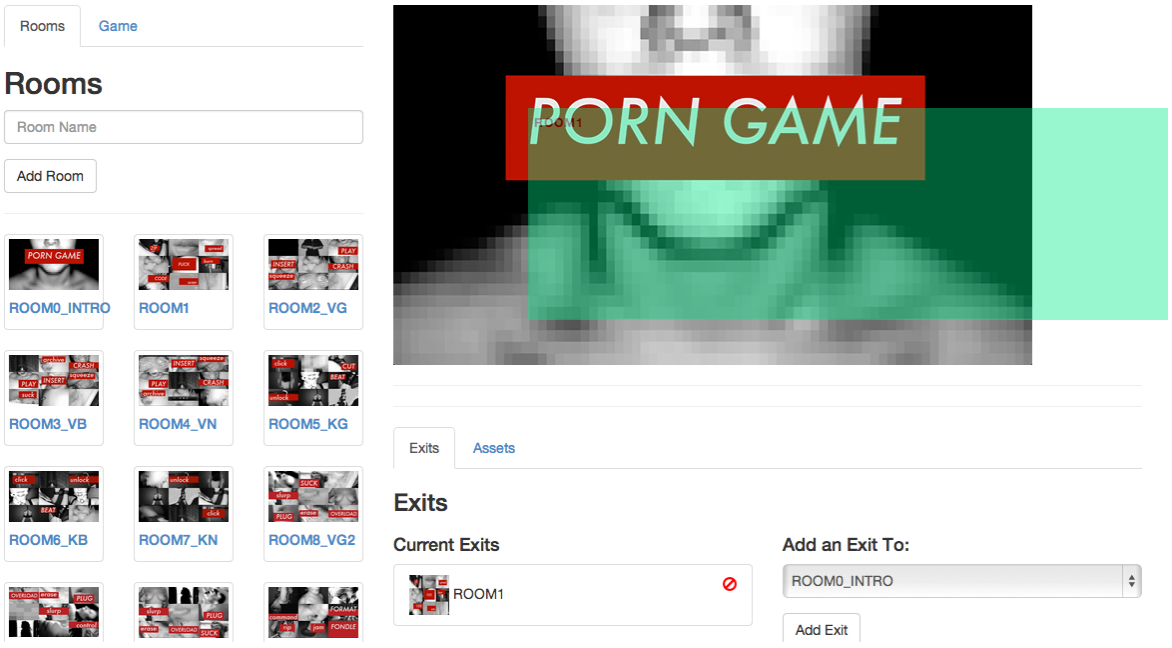


FIGURE 3.3: screenPerfect NoJam editor, Exit Population tab.



3.2 Design Research

3.2.1 Artist Collaboration

screenPerfect began with an artistic collaboration, where as a programmer, I worked closely with an artist to reproduce the technical elements of a working practice in order to make it available to other artists in a similar field. This specificity allowed us to develop a very simple tool that solves a minimal set of problems in a tidy fashion. As a developer, it can be difficult to tie work to a given set of problems, or to ensure it has value to an audience outside oneself. Therefore, collaboration gives access to a set of problems that may seem easy, but are frequently technically complex and challenging - and therefore rewarding.

3.3 Development Methods

3.3.1 Agile Development with an Artist Partner

The agile method of software development is based on the Agile Manifesto, much as the underlying feminist elements of this project are based on the Cyborg Manifesto, and Cixous' manifesto for *écriture féminine*. 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. Waterfall emphasizes planning over software production or delivery deadlines.

The idea of agile was described in 2001 by a group of software developers ('The Agile Manifesto', 2001). By using an agile practice of responsive, user-centered design, screenPerfect's interaction model was designed through a series of discussions with key stakeholders, followed by iterative

code revisions to a rough first prototype. This can be seen as 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" (Plant, 1997). 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 ('The Agile Manifesto', 2001).

The screenPerfect development process emerged from a series of linked videos on YouTube, as laid out by Hannah Epstein. We then reviewed strengths and weaknesses of this model: the ability for a large audience with public interlinked video files, but the downside of long load times and ads on pages detracting from the video content. In addition, this required uploading films at low quality to a remote server. From the initial prototype, we asked how the process could be improved, particularly for an exhibition context.

The game processes laid out in Youtube were converted to a "how might we" - a series of static files presented as interactions in still film. Hannah Epstein laid out an idea of how the video screens should work together, and I confirmed that this system was theoretically possible using websockets - a communication protocol - loaded into a Node.JS application. From there, I wrote a Node app that served basic video files to multiple browser screens simultaneously. This started as a chat application, serving text to three screens simultaneously over websockets. We then replaced the text with video files, layered a control structure over the videos using plaintext JSON files to replace a reliance on a database structure.

A database was not originally required for psXXYborg or later games, because installing a database is an additional step that nontechnical end users cannot be relied upon to find straightforward. Every step of the development process was intended to result in code that is legible to

anyone who can read javascript, while being absolutely straightforward to use for a nontechnical video author.

In development conversations, it became clear that YouTube, in addition to having many distracting advertisements, 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 video based in static rather than streaming or live files. These needed to be served within a closed environment to an attentive audience.

Scripting languages 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.JS software framework, a software package designed to permit developers to use Javascript on both the server and client side of a web application. From here, I designed a client-server-control model - Figure 3.1, the screenPerfect Software Communication Model. This relied on an interlinked system of hardware and software, and was designed to respond to what resources it found available in its load space.

Hannah relied on the h.264 format for her video production, which necessitated an early reliance on the Safari web browser, as HTML5 video does not yet have a settled public codec at the time of writing. Due to conflicts relating to codec patenting, one of many such conflicts that underly the "free" internet, Safari supports H.264 where Chrome supports webm via the V8 engine, the same engine that supports the Node framework. Webm is a 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.

Overall, Agile worked for this process by allowing a response to user requests for code changes and information rather than forcing work to fit a standard pre-set from above. A waterfall process would have simply not been flexible enough. 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.

3.3.2 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. Open source means 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 code is then shared publicly, where it can be reviewed, compiled, 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. There are some obvious problems with

open source. One of the clearest is that with intellectual property out there for free, it is a challenge to make any money on an open project. Another 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.

3.3.3 Licencing

One of the ways these problems are dealt with is through licensing. 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 license for creative practice. A preferred license 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.

3.3.4 Science Fiction Inputs

My own idea for how this project would work is derived from Cory Doctorow's Pirate Cinema, which features a scene wherein characters climb trees, and using projectors already built into their phones, assemble a movie theatre from nothing more than sheets and ropes in the trees (Doctorow, 2012). I feel 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.

3.4 Industry Engagement

3.4.1 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. Game jams and hackathons are both derived from the design charette or parallel prototyping process (Martin, 2012), a method in which 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 equipment and 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-out 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 are intended for further consumption at major conferences such as Indiecade or GDC. These conferences can further the careers of the developers by providing access to funding bodies and publishing houses (whether traditional or online), or in Ontario, the Ontario Media Development Corporation. Other funding sources can include research groups, via research funding bodies. By framing game jam development as research, participants can be released from the need to make commercially appealing works. In the case of Dames Making Games and screenPerfect, funding came from the Feminists in Games project, headed by Jen Jenson of York University, and GRAND FRAGG, a research project dedicated to expanding the diversity of voices represented in gaming. This research funding supported the research goals of this collaboration led by Professor Emma Westecott at the game::play Lab.

Game jams can be time consuming to prepare, as they involve a great deal of communication on the part of the organizers. In order to run a jam, one must open the application period well enough in advance to ensure a large cohort of skilled participants who are likely to be interested in producing content with the available tools, or interested in exploring new tools on offer. Typically, jam attendees have a theme suggested - for example, "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 produce a fast prototype over a weekend.

3.4.2 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 November 27, 2013*

The Dames Making Games are a community group in Toronto that work to promote women in video games. They have been funded in part by FiG (http://www.feministsingames.com) and in part by member donations. I am a founding director and advising director with the organization, which has given me ready access to a test audience for my ideas with regards to development tools. The Dames Making Games use 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 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 ('Game Developer Demographics: An Exploration of Workforce Diversity', 2014)('Essential Facts About The Computer Game Industry', 2014), in the hopes that a diverse population of game makers will produce a diverse population of games.

The Dames Making Games are interested in screenPerfect as it provides an underlying template for a game-making system that might be easier for newcomers to use than other freely available game engines. The other two members of the Board of Directors of DMG are Cecily Carver and Jennie Faber, who, in exchange for development work as members of Bento Box, have since forked screenPerfect to become a more elaborate engine, called iV in honour of the idea of a Twine engine that created branched narratives from Vine videos (Klimas, 2009).

3.4.3 Bento Miso and Bento Box

Game jams require both space and people who are interested in working on games. A themed game jam, such as No Jam 2, which was designed to test specific software, requires a specific audience and support. In order to access that space, I worked with the Bento Miso co-working facility in Toronto, with OCADu's game:play lab and Emma Westecott, and with Bento Box, a development company that runs Bento Miso as a not for profit co-working facility.

Miso is a not-for-profit community coworking facility that serves as home for both Bento Box, a local development hub, and the Dames Making Games. It is also the hub of a great deal of Toronto's independent game development community. Miso/Bento Box offer professional support and development advice to game developers, and I felt there was a good match between their professional skillset and my research interests. The Dames Making Games group regularly run a jam in November, and felt that screenPerfect - 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.

Bento Box was also at the time seeking an engine that could display the capabilities of their private computer language, Daimio, which offers users the ability to reprogram work on the fly in the browser without being a trusted network source. Therefore, I accepted their help and their offer of hosting the jam in return for giving them permission to fork - copy, reproduce, and extend - my engine under their name.

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

3.4.4 NoJam 2: Video Video

DMG have a great deal of experience running jams, and therefore, I partnered with DMG/Miso to access to a group of skilled animators, filmmakers, and gamemakers. In addition to a skilled community, the ongoing results of research in the community are dependent on participatory research. The premise of DMG is that people learn well by doing something new in a supportive environment, and that they can carry the experience of a supportive learning environment forward to effect real change in the game development landscape. game::play Lab is researching the outcomes and process of this type of development practice. 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. This is a theme that came up again and again in

research interviews, most notably and clearly presented in Appendix A's DVD support by Team Grimoire - Katie Foster and Mikayla Carson:

"I've come to jams three times now, and I never do it very well... the new tool caused me to change my concept and execution, but not by much. It's learning on the first day if I can make my idea work with that tool and then teaching myself that tool, because usually I have no idea how to use it.... I like the constraints that have been put on the project, since it's made me pursue new things I'd never pursue. ... It's like five rooms, they all hook straight to the next one, I'm not intimidated about putting it into the tool. ... that's what's so awesome, I'm so used to having to learn ALL new skills, every time I jam, and then I get really frustrated and I don't want to do it any more, so this time I'm like 'I know how to do all this stuff already.'"

The reduction in technical barriers to entry resulted in five complete games coming out of the No Jam process, one of which was exhibited at the Art Gallery of Ontario.

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 jam participants. 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 similar to that of the Twine engine, which was not included, although it had been planned all along.

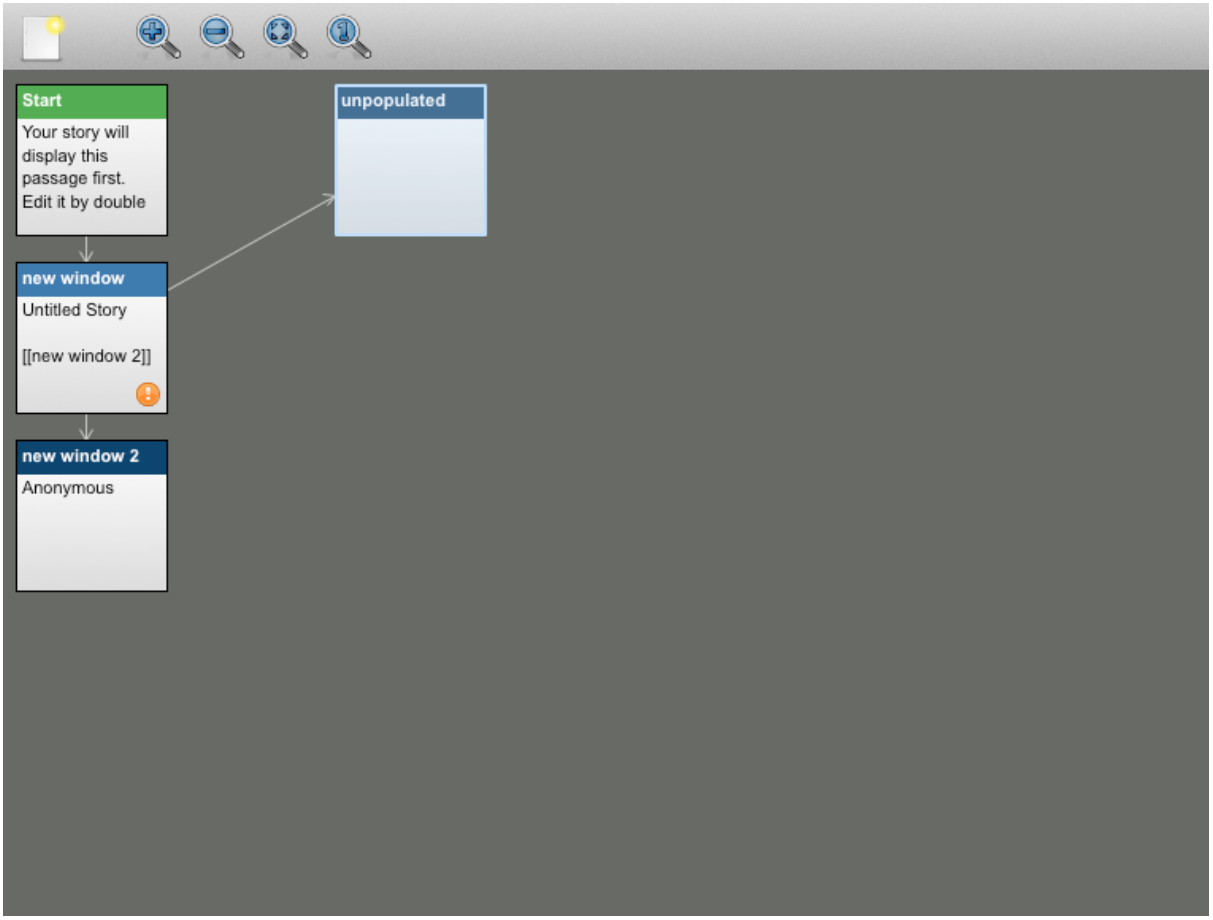


FIGURE 3.4: A branched tree of rooms from Twine.

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 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.

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 online material.

On the actual weekend, we asked that participants arrive with the majority of their video content and design prepared. There were 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 interviews are documented in my electronic support materials in named files. During interviews, I asked participants about their background, what they expected from a game jam, their previous experience in game or media design, and whether they found the restrictions of working with the screenPerfect software package useful, damaging, easy, or difficult. The main responses of interest were not about the software at all, although Mikayla Carson clearly stated that as a filmmaker she found screenPerfect much less frustrating than traditional game engines. The responses of most interest were how people think about producing media, their frustrations with traditional computer work, and what their experience with collaborative game practice is like.

I asked each group to name themselves, talk about their background, and tell me what they expected out of a game jam. Then, if appropriate, I asked what their experience with the software was, and how it compared to other game-making engines and software.

The group experience with the software proved interesting. Accomplished filmmakers had a better time with it, but the most surprising 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 design. Of particular interest was the group who tried to reproduce a classic Japanese 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.

Despite this, 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 personal narrative in producing creative work, which may have influenced the results. Game jammers mostly described their interest in producing work that was finished. 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. The finished works can be found on the supporting materials DVD under the folder "screenperfect games."

CHAPTER

FOUR

HARDWARE DEPLOYMENT IN LIMITED LOCAL SPACE

FIGURE 4.1: Hannah Epstein
psXXYborg at VideoFag, 1995

FIGURE 4.2: Hannah Epstein, Alex Leitch, Sagan Yee
psXXYborg at VideoFag, 1995

4.1 Public Installations

During the course of this project, games made with ScreenPerfect had many public outings. We installed psXXYborg specifically in a variety of spaces, and there were a number of design approaches to the construction of those spaces, governed mainly by Hannah Epstein. Hannah and I decided through a series of conversations on a number of designs for display spaces for psXXYborg, including a straight projection on mylar, a whole-room space that would encompass the user in different projected, linked screens, and ultimately a portable "confessional" booth. The first variant of this is displayed in Figure 4.1, a custom-painted cargo van which we displayed at Videofag in Kensington Market as part of the Queer Arcade with Vector Art—Game Festival as part of their 2013 show (http://www.vectorfestival.org/).

More recent exhibitions of psXXYborg have been built into a tent, which can be more easily displayed indoors. The psXXYborg tent has been exhibited at Long Winter in December 2013 and at the Feminist Art Collective's annual conference at OCADu in 2014 (Figure 4.3, 4.4).

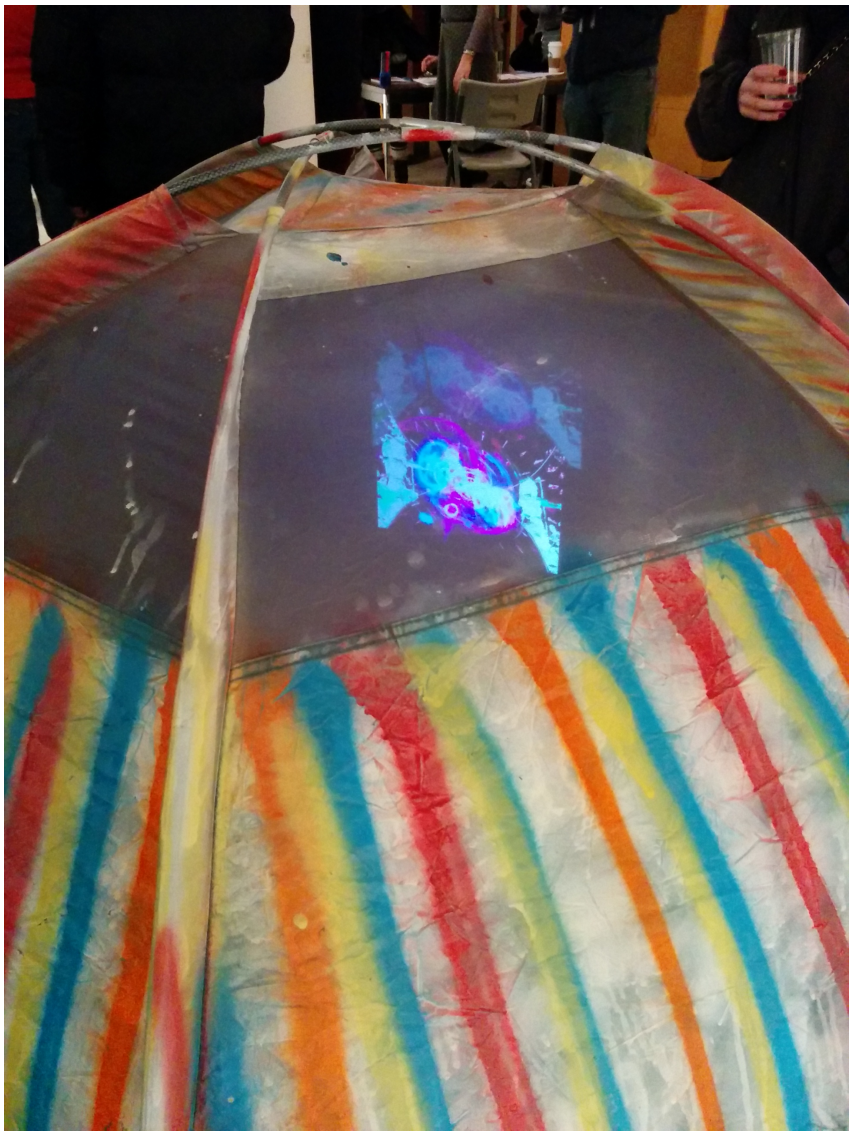


FIGURE 4.3: Hannah Epstein
psXXYborg at FAC 2014

FIGURE 4.4: Hannah Epstein
psXXYborg at FAC 2014, playthrough view

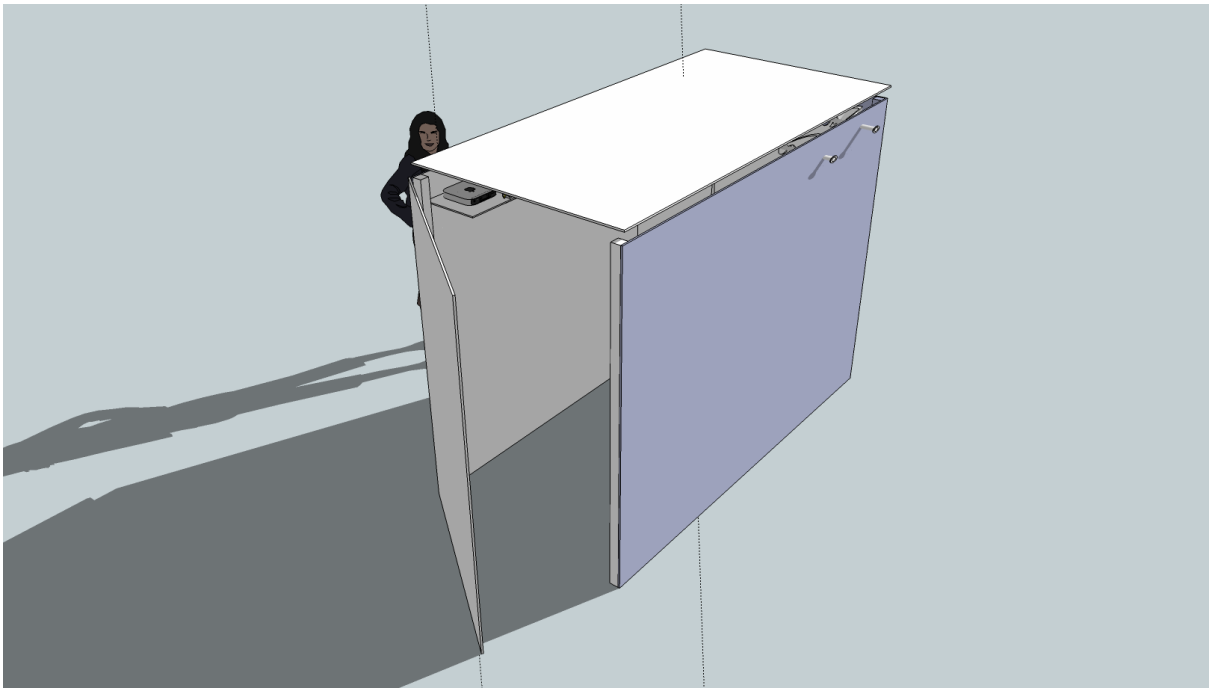


FIGURE 4.5: Alex Leitch
Concept for collapsible projection space 2013

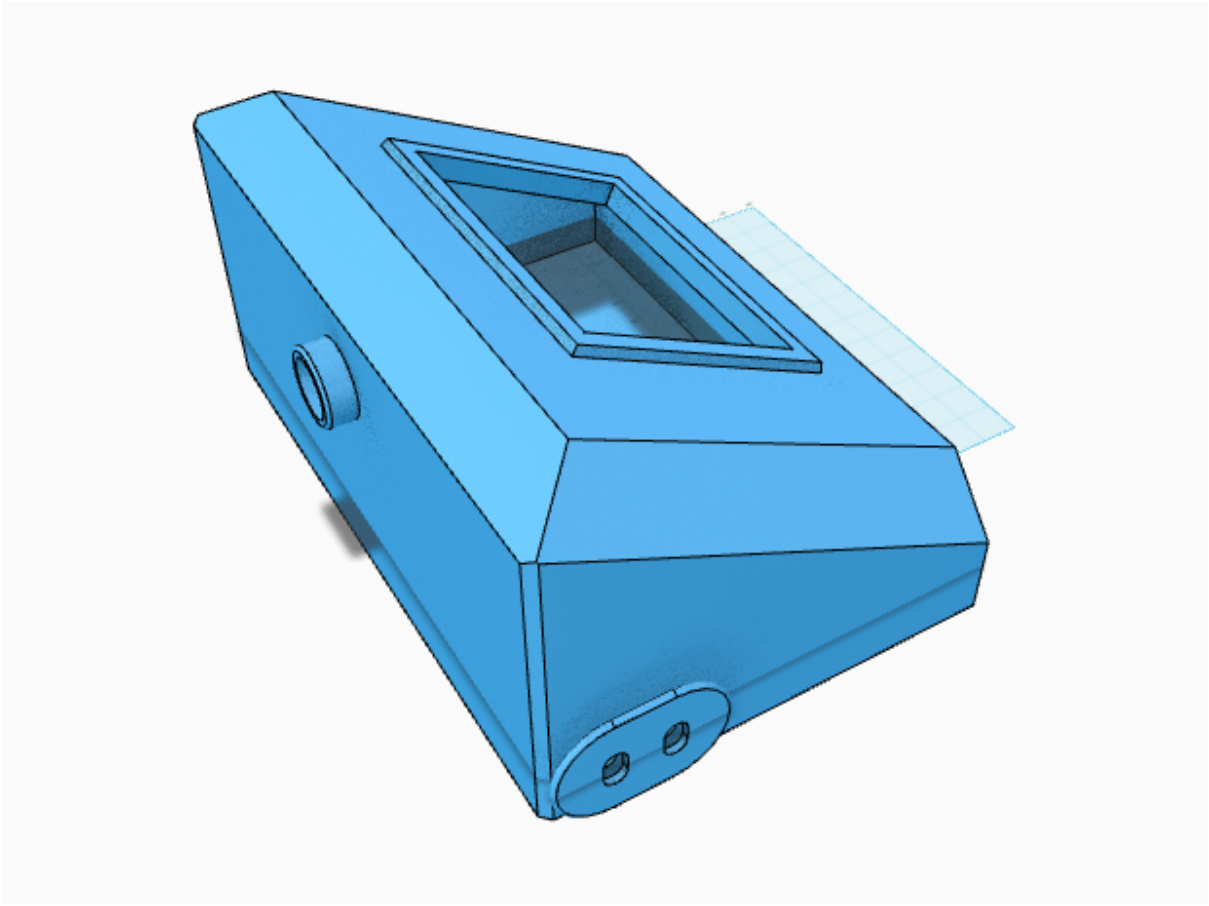


FIGURE 4.6: Alex Leitch
Concept for portable arcade box, 2013

4.2 Hardware Design

The challenge of new media installations is huge. The success of a project like screenPerfect is dependent not only on its software, or on its ease of use, but on how straightforward it is to install on-site. The software is dependent on open wiFi and high bandwidth, a stable computing system, and a variety of computer interfaces. Idealistically, the software is open. In reality, it is incredibly difficult to build a new software system to work on broad platforms, and this ended up being an unrealistic goal.

Because the computing system needs to be stable and specfic, but does not need to be used for development, I began to look into appliance-appropriate hardware. There are a number of low-power ARM-controller based hardware platforms on the market at the time of writing, including the Arduino, the Beagle Bone, and the Raspberry Pi. All of these systems are designed to help artists make interactive works that take advantage of computing power for expression. The Arduino is a popular system for learning to build electronic interactions, the Beagle Bone a full-featured Linux computer, and the Raspberry Pi has come out of an idealistic foundation that hopes to encourage teenagers to learn to program and work with ultra-simplified computing systems.

The Raspberry Pi is my choice for building a system to support ScreenPerfect. It is simplified, plugs into a television set for a monitor, and uses a lightweight distribution of Debian linux as an operating system. Debian is famous because it has excellent package management. This means that when one installs software on Debian, the software tends to maintain its own dependencies, which reduces the amount of time a programmer spends adding and removing libraries to get something simple to work. Where an Arduino would require an entire secondary shield to access the internet, the Raspi is a full computer out of the box, able to access the internet while still being useful from the command line.

Raspberry Pi as a platform is also affordable, at $35 for a Model B with ethernet port and 512Mb of RAM. Its operating system loads straight to an SD Card, which means it can be readily backed up and re-imaged should something happen to the hardware. In addition to this, the computer is the size of a credit card, which makes it straightforward to install in an artistic location such as a van or tent. The Raspberry Pi foundation is also a not-for-profit, which supports the opening of technical education to a broad range of children in the UK and overseas. This fit with the ideological stance of game::play Lab and the DMG, as well as my personal politics: that people should be able to use their tools as they see fit, and that technology should serve its users, rather than requiring the user to serve technology.

screenPerfect's technical display problems have been embedded in limited systems, such as a reliance on institutional wireless, designed to block filesharing, which also block peer-to-peer experiments such as screenPerfect's server from connecting to its client computers. This resulted in a need for a portable wireless hotspot to serve the application reliably, as in addition to blocking institutional routers, too many users would overwhelm the service. This provided a direction for my primary development to take: a computer that served both its own web application and provided its own infrastructure, from being turned on to when it was turned off, which did not require any kind of specialized setup for display. In addition, I began to consider how screenPerfect might exist in public space without the requirement of a "smart" screen or any kind of specialized interaction equipment.

People are willing to use their smartphones publicly, but mainly to access the external internet, or messaging services while they are in public. This led me to consider how people interact with the internet publicly, and to consider topics of privacy and public space, and how these problems have already been solved by galleries and coffee shops wishing to offer their clientele data services to promote engagement.

In public spaces, internet is supplied by wiFi, which comes through a specific type of router known as a "captive portal." A user will walk into a shop, attach to a network, and "sign" an agreement to make use of the wiFi within that space.

Normally, the wiFi will then give them access to the external internet - the internet as supplied by a major ISP. The direction I have taken with the captive portal supplying itself with wiFi and a server was pioneered in 2013 by the Eyebeam project Subnod.es, in which users pair to a captive portal which is also a server, supplying access to an entirely private chat room, which is available only to users on the network supplied by the captive portal itself.

The first issue addressed by this approach is that there is an inherent contradiction between downloading a site-specific piece to a personal device when the installed context is a core component of the aesthetic experience. Downloading applications to a smartphone seems invasive, particularly if those applications are experimental or site-specific, as - post-psXXYborg - I think that screenPerfect games are when they are at their best. The next is that web applications are very much not user specific - they can be experienced anywhere while they are on the open web, even if their content is intended to be restricted to a specific type of installation, or requires it for best use.

A small, portable piece of resilient hardware seems to offer a solution. By serving the application locally, there is no reliance on an outside pipe. A copy of the game can be sold, customised, and stored in a collection, if such is desired, or installed in any kind of specific cabinet for later use.

I decided on the Pi specifically because it is a cheap, accessible, reproducible system with a broad community of access and support, with the ability to include hardware controls where necessary. This type of system could also be built out on any leftover PC using a build of the Debian Linux system.

4.3 Physical Deployment on the Raspberry Pi

One of the earliest problems inherent to screenPerfect has been compromises in how the software is served to players and artists. Reliance on public internet is difficult, because the internet is not always available. WiFi is taken for granted in most institutions, but it is not always reliable, and the most interesting installation zones, such as the forest, may not have internet available at all.

There are other challenges to public deployment, such as leaving a valuable production environment out in public, or requiring a technician to look in on specialized equipment. Both of these restrict the venues permitted for public display of work. Also limiting are instances where work should be deployed near people consuming alcohol, which is notoriously bad for electronics.

This chapter addresses my central response on how to reconcile the gap between an excellent idea for web-based deployment, and the physical reality of gallery spaces with sharply limited resources available for persistent software deployment.

FIGURE 4.7: Abe Shulz and Sage Kochavi
Seizuredome at FireFly 2013

4.3.1 Problems and Complications in Display

Some brainstorming resulted in the following scenarios that a reasonable arcade machine would need to address in order to present advanced work to the new, highly exclusive exhibit scenarios.

1. Data service to an external source cannot be assumed to be available.
2. The exhibit is assumed to be displayed in public
3. The environment is assumed to be meterologically hostile - hot or cold, wet or very dry, and to be hosting at least one party, such as an art opening, possibly with music
4. The exhibition is assumed to be supervised by technically untrained people.
5. The emphasis of the work should be on the work's display, rather than on a laptop screen.
6. The collectors of the work are assumed to have extremely limited resources for rugged workstations.
7. Any host-provided data carriage for external connection - wiFi - is assumed to be overloaded by default.

These are all very real constraints that impact display of new media art. We use computers for work and play, but we still separate our lives into periods when we pursue one or the other, and we still have boundaries between our personal and public lives. To use the same machines to display art as we do to build the work is to reduce the work from something approachable to any other tab in a computer. New media works especially must be seen within their exhibition context to be understood.

4.3.2 Subnod.es and Public Private Space

This project has a precursor using similar technology builtat Eyebeam in New York in 2013. Subnod.es uses a captive portal similar to my own, based on the inexpensive Raspberry Pi framework, to display a chat client to only the local environment. The differences are substantial, although mainly located within the code. Subnod.es relies on an external DNS being made available via the actual subnod.es software, and depends on a different collection of software to serve the portal proper. It is also built such that those library dependencies are inseparable from the main project script.

The chief concern of subnod.es I have not yet mentioned: subnod.es was built as a response to concerns about communications privacy in North America under the NSA. Specifically, the author is concerned that people behave differently when they are watched, a subset of the concerns generally associated with panoptica and totalitarianism. While I have not specifically structured screenPerfect's Art Portal to address these concerns, it has been built to be largely private. It serves an application to a limited selection of a public space.

The assumption of the Captive Portal Art Machine is that galleries have limited resources, but that people who go to art galleries almost certainly have access to a smart phone, which is a form of private space. Smart phones are people's own homes, and are built to assume that they will stay with their owners at all times. This means that to install an app is to ask a lot of a viewer: specifically, it is to ask someone to bring an application into their private space without getting to sample it first. To contrast, serving that same application on the broad internet is to entirely delimit the context the art may be experienced within, which reduces its scarcity value to almost nothing while simultaneously removing the curator's ability to set the context of an exhibition experience. This means that it's unlikely an artist can be compensated in any

conventional sense, despite their large audience, and also means that the curation of the exhibit is no different than the "curation" found on Tumblr.

A better outcome might be to make a limited public space available in a private context, and this is what we are doing when we ask that people open their phones and look at a website. The Internet is, famously, the new public space. By presenting a web application using public technology within the exclusive context of the gallery - or desert, or forest - we take control again over how our art is presented, and from there, how it can be consumed. A gallery or exhibit space can be set up very specifically for the benefit of an audience in a way that the internet in general cannot be, and web technologies are uniform and affordable in the way that more custom projection design software is not.

This sense of limited private space is key to the code-switching that human communication relies on. We are not the same people in public as we are in private, and we are again different people when we are in different publics, work to the street to school to the gallery. Technology that sensitively addresses these different code contexts seems likely to benefit its authors and its users both, by permitting the integration of the personal focus with a personal device with the context of a semi-public facility or "safe space" for engagement with the art work.

4.4 Distribution of Work

This work has failed from the point of view of distribution at its conclusion. There is no reasonable way to distribute these works that is not reliant on a Node.JS enabled system being available, which means that ultimately, they are impossible to display as intended. I have included a record of the finished works built during No Jam 2 in my appendices, where they can be installed and played from disc, but they still require at minimum an installation of Node.JS to be played. Without that, they remain hidden. The point of this work is not the internet,

so much as it is the possibilities present in the technology we rely upon for the internet to be accessible.

In the meantime, I have built individual variants of finished game jam games and shipped them in complete SD cards to their owners, so that, with a little effort, they can be run from any Model B Raspberry Pi just like a game cartridge.

CHAPTER

FIVE

CONCLUSION

5.1 Conclusion

The work that has gone into the production and release of screenPerfect is not inconsiderable. As a creative project, code is a tricky thing to pin down. It must say something structurally, yet it reveals the internal architecture of its authors. Programming leaves 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 inevitable tendency to produce flaws - bugs - that cause the program to fail. Once complete, it is likely that finished software will fall out of fashion. 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 lives, like writing, in context and within an ecosystem. Code answers to its context. As Alexander Galloway says, without the machines that run it, code is without consequence Galloway, 2006. Within those machines, code may have a concrete effect on the world around it, and for that reason it continues to be valued. To code is to attempt to write a way of addressing the world, a single way that must take into consideration all the assumptions of people who tried to address the world before, and, with future-proofing, the world after.

In Cixous' Laugh of the Medusa (Cixous, 1976) she states that to be considered real, women must write for themselves. In my thesis, I have extended this to the world of code, and through it to computers: one must write after one's own interests, represent a possibility for what one believes can be real. I have done this by writing a tool that permits an array of interested parties to produce games without a reliance on what a game-engine maker thinks of as resources, and then extended it with hardware to support what I consider to be a straightforward installation path using materials from a not-for-profit, then sharing them publicly.

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. The system does not judge content. It does not care where you serve your information, or to whom. The emphasis is on allowing an audience to experience things as quickly and easily as possible. Works produced using screenPerfect can be displayed anywhere a series of screens and a single server can be set up. This emphasis on experience moves the interaction sphere of art and gaming into the world. screenPerfect's impact is most felt at night, outdoors, or in temporary installations. 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 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 what constitutes accessibility. At minimum, this is software which is straightforward for non-technical people to use. In addition, this includes attention paid to hardware to reduce the reliance on specialized technicians to exhibit and install interactive works, which otherwise are not terribly collectible Paul et al., 2013. 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.

At the outset, I asked a series of questions about capital, monetary and otherwise, creative practice, and preservation in the digital context. In the effort of designing screenPerfect, I have put forth a physical idea, which is that a small tool optimized to do one thing well is better than a tool that can do all things but requires a manual. I have suggested that a piece of software that addresses the needs of a specific group of curious people is likely to permit them to express themselves in a new medium, and I have issued the idea that a piece of coherent software with its own operating system might have a better half-life in a public context than one which needs to be custom installed each time by hand.

I am optimistic that the systems optimized and built with screenPerfect - a system of small SD cards with an optimistic installation of the raspberry Pi operating system and a specific set of working software - will continue to operate when plugged in and turned on, presenting the opportunity to engage with their works going forward. For my own part, the project has not been optimised for distribution or further use by new artists. It has been presented as a prototype, an argument in favour of a way of thinking. Although I have already begun to customize a hardware system to expand on how the software might be installed, at the time of writing, the work is incomplete. There is always more to be written.

There are futher questions within my theory chapter, about the valuation of personal narratives as games, as economic or artistic practice. My conclusion to this is that it is not necessary to join the commercial order, but that if one does not display one's work, it is straightforward to regard the work as not having been done at all. This is the core of software. It hides until it breaks. A new direction could be to have software that obviously breaks, or that decays in an interesting fashion, and this leads to the glitch aesthetic, which is popular, and beautiful in its

way. Another could be simply to understand that things which cannot be seen or quantified have nonetheless value, and to quantify that value on a different plane - efforts at this direction include Una Lee's Difference Engine poster, about what the participants of the first women-only game incubator achieved from participation in that incubator.

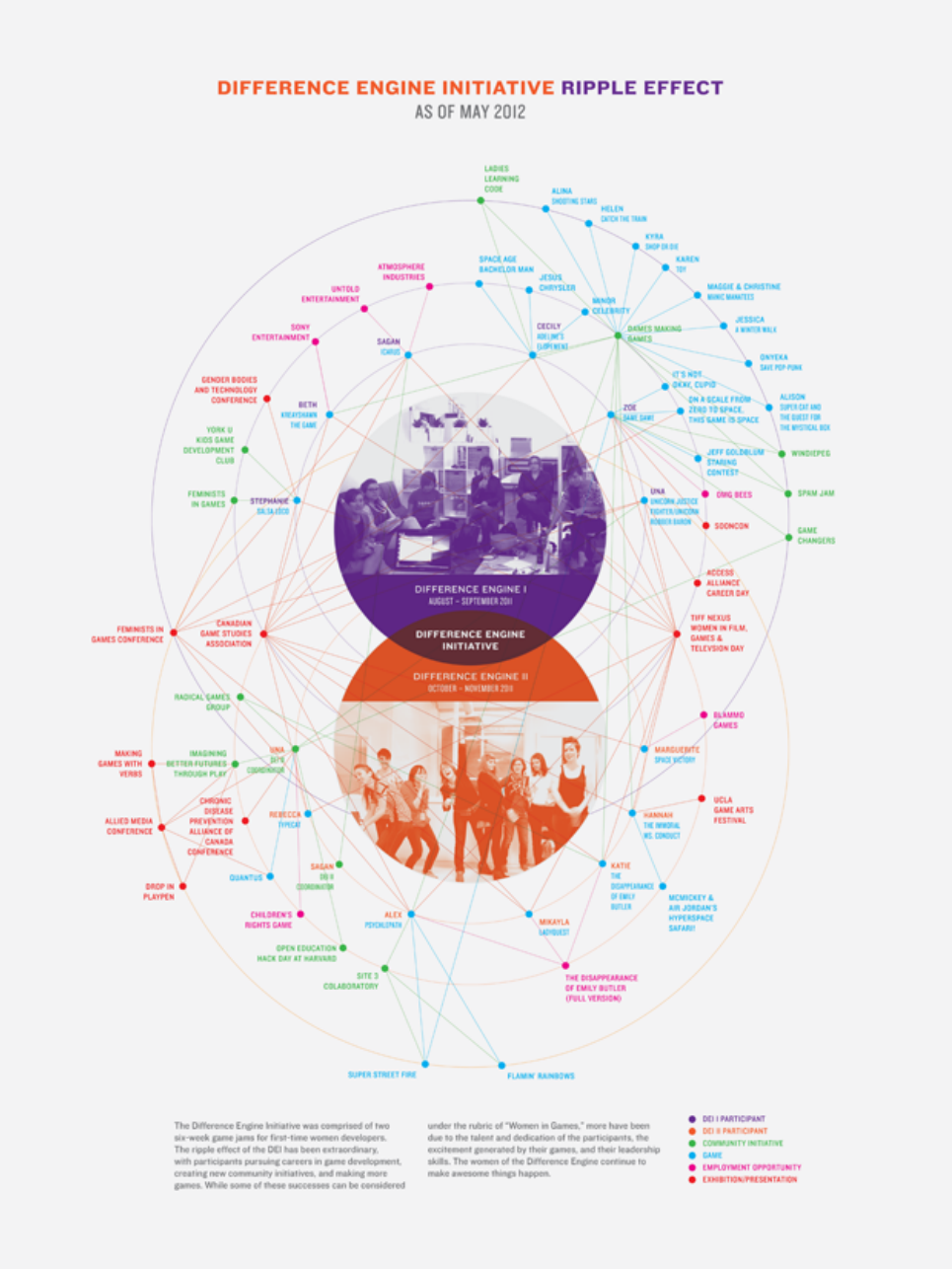


FIGURE 5.1: Una Lee, Difference Engine Initiative Visualization, 2012

Does one wish to join the commercial order? Is it necessary to be validated, to have commercial success , or is it simply another demand on the work, that the work sell to prove that it is

fundamentally worthwhile?

APPENDIX

A

SCREENPERFECT INSTALLATION GUIDE

A.1 screenPerfect Code and Documentation

The code of screenPerfect is located on the DVD included in this volume, and at http://www.github.com/pretentiousgit/screenperfect-dev. This DVD includes five games produced for the screenPerfect Engine: PornGame, psXXYborg, OM, Grimoire, and Cyborg Goddess.

A.2 Device List

1. Large powerbar with surge protection and power supply cable
2. Server with Node.JS installed
3. Projector or monitor
4. Portable wireless hotspot
5. Touchscreen device with browser software
6. Optional: Speakers with subwoofer
7. keyboard
8. mouse
9. power supply
10. hdmi or vga cables as appropriate

A.3 Software List

1. Google Chrome (preferred), on all devices likely to play.
2. screenPerfect will also run on Safari, but there are issues due to Google's refusal to support the MP4/h.264 video codec (large, but good quality) and Apple's similar refusal to support V8/webM (small, better, owned by Google).
3. NodeJS installed on screenPerfect's host computer.
4. git installed on screenPerfect's host computer.
5. Miro VideoConverter on a mac system

A.4 Installation and Site Construction

A.4.1 Before Leaving For Site

1. Ensure NodeJS is installed and functional on main server.
2. Install latest repository of screenPerfect from GitHub.
3. Ensure all video files are present and accounted for in all appropriate formats.
   (a) webM
   (b) MP4
4. Find surface to project on.
5. Lay out power cable and power bar.
   (a) power to wireless hotspot
   (b) power to main server
   (c) power to projector
   (d) power to support tablet interface
   (e) power to speakers
   (f) monitor/projector
   (g) keyboard
   (h) mouse
6. Connect devices to their relevant power supplies.
7. Plug in green jack of speakers to headphone jack of main server.
8. Turn on all devices.
9. Turn off all mobile data for wireless hotspot device. If necessary, remove SIM.
10. Turn on wireless hotspot.
11. Turn on your main server.

12. Connect to your wireless hotspot.
13. Open Terminal (Mac), navigate to screenPerfect's home directory, turn on screenPerfect
    (a) Navigate to the game folder you wish to run by typing "cd" and then the directory path at the prompt.
    (b) type node screenPerfect, press enter.
    (c) You will see a message saying "screenPerfect listening on port 3003".
14. Open a new tab in Terminal (command-T)
15. Type ifconfig to get the IP address of your wireless hotspot.
    (a) look for the value called "inet" - it will be a 192.168.x.x address if you are attached to the hotspot, and (probably) 10.x.x.x if you're attached to a wider network.
16. On the Server, Open Chrome and navigate to http://0.0.0.0:3003/client
17. Open Terminal, and in the tab where you typed "node screenPerfect" there will be a stream of messages about a heatbeat.
18. There should also be a video playing in your browser now.
19. On the Tablet Device, open Chrome. Replace the below values with the ip address located in step 14a
    (a) http://x.x.x.x:3003/control
20. If all goes well, the "Start" button will appear onscreen on your tablet.
21. Touch the start button, and the projected video will change.

A.5 Troubleshooting

A.5.1 On Start, Google Chrome Cannot Find Control Screen

1. Check that the Tablet is on the wireless network provided by the hotspot.
2. If not, repair the hotspot by disabling it, reenabling it, and reconnecting both the server and the tablet to the same network.

A.5.2 The Control Device is frozen, or the videos are not changing.

Just about anything else, really, including the above.

1. Open the Terminal.
2. Check the heartbeat stream for an error such as "Abort Trap," "Bus Trap," or "stream error in pipe."
    (a) All of these are normal. They are a result of someone tapping the screen quickly enough to overwhelm the server.

3. type node screenPerfect and press enter to restart the application.
4. Refresh the tablet browser and press the start graphic to make sure the video is being served.
5. Refresh the server browser to ensure everything is passed correctly.

A.6 Tidbits and Tech Notes

- People with little technical experience will sometimes have trouble determining if they have actually touched/interacted with the app appropriately. Some may accidentally exit the application on the tablet, and may be unsure of how to re-enter the experience.
- screenPerfect is most rewarding when played in isolated concentration somewhere quiet. It is challenging to concentrate on the video enough to get full effect if people are distracted.
- screenPerfect crashes approximately once per three hours of play time, more in high temperatures.
- screenPerfect games work best in Android, as Android supports the webM/V8 codec. Apple devices require mp4s, which are approximately 4x the size.
- screenPerfect was developed in JavaScript using ExpressJS for NodeJS. screenPerfect was developed on a MacBook Pro for use with Chrome, but will function almost as well with Safari.

APPENDIX

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. (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_2004.pdf More on game design techniques for the technical portions of the paper. Describes methods which will need to be addressed as part of the Agile/How Did I Make This Game methodology.

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 pre-set, 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/43
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.

APPENDIX

C

GAME JAM DOCUMENTATION

C.1 Interview Files

Interview sound files can be found on the Supporting Materials DVD within the "Game Jam Interviews" folder.

C.2 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.2.1 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.2.2 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.2.3 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.2.4 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.

APPENDIX

D

RASPBERRY PI SETUP DOCUMENTATION

D.1 Materials and Supplies

Raspberry Pi The Raspberry Pi is a full linux computer the size of a large credit card. A Raspi runs Debian linux off of a common SD card. **32Gb SD Card for Raspberry Pi** This is where we place the operating system and software for the Pi. **USB wiFi dongle** Edimax-based wiFi USB dongle, for serving wiFi hotspot on the Pi. **USB flash memory** For transferring or storing complete programs authored on external systems. **Keyboard and Mouse** For initial computer setup. **Ethernet Cable** Standard cat5 ethernet cable for programming remote. **HDMI TV and cable** Used as a monitor for the Raspberry Pi. **Micro USB and power supply** Power for the pi. **Mac or PC computer with USB ports, ethernet port, SD Card reader** Required for raspberry pi setup.

D.2 Background for Linux Commands

sudo means "do this now even if I appear to have insufficient user permissions" in Linux apt-get is an inherited "package manager" from Debian linux. "Dependencies" are the software your software requires to run, Debian uses apt-get to manage them. Things that follow sudo are commands.

D.3 Setting Up The Raspberry Pi

D.3.1 Windows 7 SD Card setup and first boot

This section is written for a Windows 7 environment, and is based on the common tutorial at http://learn.adafruit.com/adafruit-raspberry-pi-lesson-1-preparing-and-sd-card

1. Connect your main computer to the internet.

2. Download the most recent Raspbian distribution image from http://www.raspberrypi.org/down
3. Download Win32DiskImager from the greater internet. This is preferable because it allows you to write image backups to your harddrive.
4. Using Win32DiskImager, write your Raspbian distro to your SD card on your main computer.
5. Eject the microSD card and stick it into the Raspberry Pi.
6. Plug in your keyboard, and plug a mouse into your keyboard.
7. Plug in your HDMI cable and monitor. Turn them on.
8. Plug in the MicroUSB cable for power to the Raspberry Pi.

D.3.2 Configuring Raspbian

Once the RasPi is turning on, it needs to be set up to include all of its software. Turn the Pi on, and wait until the blue configuration screen comes up. Figure 4.3: Early RasPi Configuration Screen

1. expandrootfs Expand the boot system so that you will not run out of onboard memory for software.
2. memorysplit Reduce the GPU to minimum, because we will be using the raspi as a headless server from the command line.
3. changepass Change the password so that the Raspberry Pi will be less easy to hack.
4. ssh Enable SSH so that the pi will be accessible from an external computer.

When done, select finish to exit. Type sudo reboot to restart the raspi.

D.4 Software Setup for External WiFi Access

A wiFi antennae can be used for one purpose at a time: it can either be used to access the external internet, for acquiring software to install into the raspi, or it can be used for serving a hotspot. It cannot do both at the same time. To load the pi up requires external access, so we will be loading that first. You must configure your wiFi before plugging in your wiFi antenna.

In Linux, there is warning you if you mistype a folder name, say, adding an "s" to "network" to make it "networks." If you would like to confirm your folder name is correct, try typing "ls /etc/" to list the contents of that directory. Network is a default folder, and Interfaces is already present at first boot, so you can make sure your things are all there before you really get started. The way to tell you have done something wrong is if you type the below command and an empty new file opens. You are editing a file here, not creating one. At your console prompt, type the following:

```
sudo nano /etc/network/interfaces
```

This opens a text editor called 'nano.' Enter the following into it.

```
auto lo

iface lo inet loopback
iface eth0 inet dhcp

allow-hotplug wlan0
auto wlan0

iface wlan0 inet dhcp
wpa-ssid "network name, commonly called an ssid, goes here"
wpa-psk "password"
```

Then type CTRL-X and Y to save your file.

```
sudo halt
```

Plug in your wifi antennae, pull the Raspberry Pi's power cable, and plug it back in. This should make the raspi's antennae turn blue as it turns on. This little blue LED will frequently be the only way to tell something is going correctly or incorrectly, so it is an excellent tell that your machine is running.

Plug in your wifi antennae, pull the Raspberry Pi's power cable, and plug it back in. This should make the raspi's antennae turn blue as it turns on. This little blue LED will frequently be the only way to tell something is going correctly or incorrectly, so it is an excellent tell that your machine is running.

The case in the following figures is a modification of an open-source design supplied by Thingiverse.com user DrewTM. It is Thing #114244 and can be found at https://www.thingiverse.com/thing:114244.

FIGURE D.1: Raspberry Pi with functioning wiFi antenna

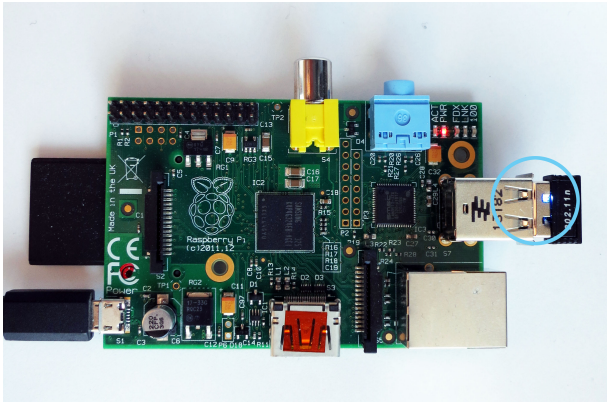
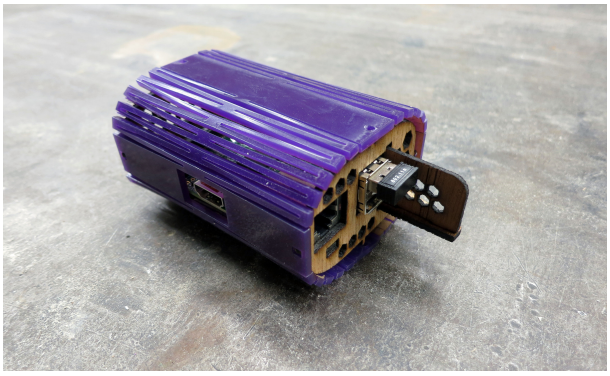


FIGURE D.2: Raspberry Pi in Thingiverse case



If all went well, you've now connected to your own supply of wireless internet. This will not work if you are using an 802.1x network, such as those within OCADu. On your own home network, however, type:

```
sudo apt-get upgrade; sudo apt-get update
```

This will upgrade your rasppi to whatever the latest agreed-upon package lists are, then update those packages to their most recent approved version.

D.5 Installing Node.JS

D.5.1 Why Node?

I've chosen to install Node because it is the software framework I selected to run the new game engine built in Part 1 of this thesis. Node is a new framework designed to get Javascript running on a server. There are advantages and disadvantages to this approach. The advantages are that JavaScript is a beautiful, minimal language that is relatively easy to learn. The disadvantages are that there is a heavy public bias against JS due to its years as a client-only language designed to manipulate what are known as Document Object Model (DOM) elements in-browser.

The brilliance of Node is that it replaces the need for a specific input-output window, replacing that definition requirement with any internet browser. Node, backed by Google's V8 engine, currently works best on Chrome, but it can interact with any browser.

Node is therefore easy to use, and easy to program for from the perspective of a mainly web based development chain.

D.5.2 Installation Instructions for Node.JS

Create a directory for Node to live in by typing the following at prompt.

```
sudo mkdir /opt/node
```

Acquire the node "tarball" - compressed framework files - via the internet.

```
wget http://nodejs.org/dist/v0.10.2/node-v0.10.2-linux-arm-pi.tar.gz
```

Unzip (desticky from tarball) it:

```
tar xvzf node-v0.10.2-linux-arm-pi.tar.gz
```

Copy the contents of the newly unzipped folder and paste them to your new directory. This leaves a copy of the tar and a copy of the unzipped tar at their original locations. You can probably remove them using sudo rm when you're sure everything is where it should be.

```
sudo cp -r node-v0.10.2-linux-arm-pi/* /opt/node
```

Edit - or create - a .bash_profile file, which is a type of script that runs when you turn on the pi. In this case, it runs and tells Node that it exists on your computer, so that typing node runthisprogram will do something. What is a .bash_profile?

From your root directory, to open a new nano text file:

```
sudo nano .bash_profile
```

Then add the following and save it to your new .bash_profile file...

```
PATH=$PATH:/opt/node/bin
export PATH
```

Control-X, Y to save it.

Node lives in the `/opt/node` directory you created above. This adds the commands "node" and "npm" to what are called "environment variables." If you are curious, and god knows you must be to play with a raspi, you can type `ls /opt/node/bin` and see the little programs sitting there in their bin.

D.6 Testing Node

Node will need to be able to fetch its own packages separately from the raspi from the internet in order to run some of the monitoring software I've chosen to use. Particularly, you will need the `forever` package.

D.6.1 Selecting Monitoring Software

`forever` has ultimately been the software I've decided on to monitor and run screenPerfect, because it is a node-native package that keeps things running even when they crash. There are other software packages used for broader deployment, such as Monit, which installs to your Debian parcel rather than to Node. Monit typically runs with what is called an HTTP Proxy, which can be written directly in Node or installed independently. In a full deployment build, Monit and HAProxy would be preferable to Node alone, because this follows the best practice of separating out different programming elements from one another in production. Monit and HAProxy can also deploy applications above and beyond Node itself, which is preferable for things written in Python, for example.

For this example, though, `forever` works well. It provides monitoring to tell us what the application is doing, and automatically restarts node applications when they crash. Were I deploying this such that it could keep an eye on the internet, which I am not, I would also include `nodemon`, as is recommended by the Subnod.es project. `nodemon` monitors your development code and pushes changes from a central server to your deployment automatically.

That is outside the scope of this paper at present.

D.6.2 Installation of Node Modules

To install a node package - or "module" - you type

```
npm install PACKAGENAME
```

To install one globally, type

```
npm install PACKAGENAME -g
```

To absolutely force install:

```
sudo su
PATH=/opt/node/bin/:$PATH
npm install PACKAGENAME -g
exit
```

To install `forever` and `nodemon`

```
npm install forever -g
npm install nodemon -g
```

To run `forever` and `nodemon` together....

```
forever start /usr/local/bin/nodemon /path/to/YOURAPP.js
```

D.6.3 Troubleshooting NPM installations

When I tried to install `forever` the first five times, it timed out, gave me a 404 error repeatedly, and declared I had insufficient permissions to do a global install. This is where computer science faith, confidence, and patience come in. When the install did not work for half an hour, I took a break, came back, and discovered that it installed the next day.

This process is heavily dependent on a massive network of computers and other people. In development, it is quite likely things beyond one's own control are going to go wrong. Going for a break will help you keep patient.

D.7 SSH via Direct Ethernet Connection and WiFi Internet Access

Eventually, you will need both of the powered USB slots on the raspi for a USB key and for your wiFi. In addition, the raspi doesn't have the power to drive a monitor and consistently serve wiFi out of its USB ports. To get around this, it is most convenient to be able to SSH in to your device. Although it appears to be best practice to use the wpa_supplicant file to store how you wish the Raspberry Pi to connect to the internet, I have had limited success with it, likely because I am not configuring a static IP for my raspi properly.

My `/etc/network/interfaces` file looks like this:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address [MY MAIN TERMINAL'S ETHERNET IP PLUS ONE]

auto wlan0
allow-hotplug wlan0
iface wlan0 inet dhcp
   wpa-ssid "network name here"
   wpa-psk "dubiously secure password"
```

```
sudo nano /etc/default/ifplugd

### MANY TALK, HOW COMMENT, SUCH WARNING ###
INTERFACES="eth0"
HOTPLUG_INTERFACES="eth0"
ARGS="-q -f -u0 -d10 -w -I"

SUSPEND_ACTION="stop"
```

This is an edit of the existing bits, and I can't tell if it will break everything long-term. Here is what your startup script should read. This ensures that your wiFi antenna turns on, which is likely not something it was doing when you plugged in your ethernet directly.

```
sudo nano /etc/rc.local
#!/bin/sh -e

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
 printf "My IP address is %s\n" "$_IP"
fi

# Disable the ifplugd eth0
sudo ifplugd eth0 --kill
sudo ifup wlan0

exit 0
```

CTRL-X and Y to save, then `sudo reboot` open a terminal on your main laptop. On your laptop, at the prompt, enter:

```
ssh pi@[the static ip address you entered under eth0 static above]
```

Your `pi@[static ip]` should appear in your terminal window, which means you can now talk to raspi. Per usual, to ensure your wifi is still working properly, try a `sudo apt-get update` or `ping google.com`, both should return you data.

D.8 Backing Up the Raspberry Pi

Now that everything has been configured for the first steps, type `sudo halt`, and when the Raspberry Pi turns off, remove the SD card from it. Place the SD card back in your main computer and reboot Win32DiskImager.

Create a new file folder somewhere within your Documents folder. I called mine Raspberry Pi Backups.

In the Write From section of the application, select your SD card, which is probably called boot. In the Write To section, select your new folder.

Write a copy of the kernel image from the boot card to the new backup directory. Then safely eject your SD Card and re-insert it in the RasPi. It is best practice to form these occasional backups as you proceed through set up. Many of these steps can cause the Raspberry Pi distro to break badly. A backup will save a great deal of time when the inevitable happens.

D.9 Mount Your USB Flash Memory Stick To the Raspberry Pi

D.9.1 Configuring Your Mount Drive

This bears some thinking about, because the `/media/` folder is for media, and you are instead choosing to run a program off of the drive. Subnod.es suggests making it your `www` drive, for world wide web. I picked `/mnt/`.

Find your USB memory by listing the the things plugged into dev:

```
sudo ls /dev/sd*
```

If you've been following along, yours is almost certainly named "/dev/sda1".

So make a directory for it to be addressed at:

```
sudo mkdir /mnt/USBSTICKNAME;
```

Then mount it to that directory

```
sudo mount -t vfat -o uid=pi,gid=pi /dev/sda1 /mnt/USBSTICKNAME/
sudo reboot
```

Rebooting will restart the raspi but also close your SSH session. Watch the lights on the raspi board until they're stable again, about two minutes, then:

```
ssh pi@[static ip]
```

Oh look. Your USB drive does not automatically mount at boot. Problem.

D.9.2 How to Boot Mount External Memory

Find out the actual name of your external memory card:

```
ls -l /dev/disk/by-uuid
```

Write down the UUID of your USB stick.

This is the most manual way to run this operation, and there is software that handles automatic drive mounting. It is called `usbmount` and was discarded during this process because it ended up being more convenient to rely on my Node application being loaded directly onto the SD card, rather than from boot.

```
sudo chmod 775 /mnt/USBSTICKNAME
sudo sp /etc/fstab /etc/fstab.bak
sudo nano /etc/fstab
```

Add the following to `/etct/fstab`

```
UUID=YOURUUID /mnt/USBSTICKNAME vfat rw,defaults 0 0
```

CTRL-X, Y to save, then

```
sudo reboot
ls /mnt/USBSTICKNAME
```

This command should display the contents of your USB key when you go looking for it.

At this point, I have taken a copy of my Node application and moved it to the SD card in a separate directory. Although I have optimistically tried to make this a headless - no keyboard or monitor - box, realistically, lots can go wrong with the SSHing process. You will probably eventually want a keyboard, and it is _much_ easier to store your access point as a single image per card, much like any other video game.

To store your games locally, rather than in the USB stick:

```
sudo cp -r /mnt/USBSTICKNAME /home/pi/YOURDIRECTORYNAME
```

D.10 Set Up a wiFi Hotspot

To get started, you will need some more software.

```
sudo apt-get install hostapd dnsmasq
```

When everything is done installing, you will be converting your `/etc/network/interfaces` file to serve a hotspot, rather than connect to the internet.

Here is what my final `/etc/network/interfaces` file looks like:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
   address 169.254.222.xx #xx is a stand-in for an actual address, not included.

allow hotplug wlan0

## wlan internet connect settings are commented out for easy swap.
#auto wlan0
#iface wlan0 inet dhcp
#   wpa-ssid "network name"
#   wpa-psk "network password"

iface wlan0 inet static
   address 192.168.42.1 #42 is a joke about Douglas Adams, in honour of my thesis
    advisor.
   netmask 255.255.255.0
```

D.11 Configuring HostAPD

`hostapd` is the software that provides the access point using the Raspberry Pi. It can be tricky, and in order to make it work, it needs to be compiled for one's specific model of wiFi antennae. For the purposes of this paper, we are using an antenna sold and supported by Adafruit. The appropriate compile of the hostapd software is included in the supplementary files to this paper, but can also be found at http://www.adafruit.com/downloads/adafruit_hostapd.zip.

To install a valid copy of hostapd:

```
wget http://www.adafruit.com/downloads/adafruit_hostapd.zip
unzip adafruit_hostapd.zip
sudo mv /usr/sbin/hostapd /usr/sbin/hostapd.ORIG
sudo mv hostapd /usr/sbin
sudo chmod 755 /usr/sbin/hostapd
```

Now set up a daemon - a piece of automatic system software - to run the hostapd configuration file on boot.

```
sudo nano /etc/default/hostapd
```

Uncomment (remove the hash mark in front of) `#DAEMON_CONF=""` and replace that line with `DAEMON_CONF="/etc/hostapd/hostapd.conf`. Then type CTRL-X and Y to save your file.

My hostapd file is listed below.

```
sudo nano /etc/hostapd/hostapd.conf

interface=wlan0
driver=rtl871xdrv
ssid=piebox
hw_mode=g
channel=6
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=berrybox
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

D.12 Configuring DNS access via `dnsmasq`

Configuring `dnsmasq` is straightforward. The installation package comes with an extensive config file, which lives at `/etc/dnsmasq.conf`, and includes all of the options necessary to turn on a DNS routing service.

To configure your dnsmasq installation, enter `sudo nano /etc/dnsmasq.conf` and then add the following lines to the top of the configuration file. The configuration file contains all these values commented out already, and may be worth a separate read.

```
interface=wlan0
dhcp-range=192.168.42.2, 192.168.42.50,255.255.255.0,12h
address=/#/192.168.42.1 #redirect all DNS requests to 192.168.42.1
server=/screenperfect/192.168.42.1#3003
address=/apple.com/0.0.0.0
```

What the above does is tell the raspi to listen on the wlan0 interface, to the dhcp range between 192.168.42.2 and 42.50, for twelve hours per time a client connects to the wiFi point. In addition, the portal is supposed to redirect all DNS requests - things like "google.com" - to the Pi's main address, which is - as we can see in/etc/network/interfaces - 192.168.42.1, and from there to the port 3003, on which my particular Node application listens.

In addition, the portal serves a spoof address to apple.com, which helps us to pop up the appropriate page on the captive portal when it is turned on.

To date, this portion has not proven totally effective. Getting a page to pop up on a captive portal requires a series of correct internet handshakes per device, so it has so far been easier to set the URL by hand on client devices to the Raspi.

APPENDIX

E

APPENDIX E: MIT LICENCE AND RESEARCH ETHICS APPROVAL

E.1 MIT Licence

screenPerfect

The MIT License (MIT)

Copyright (c) 2013 Alex Leitch

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Daimio

The MIT License (MIT)

Copyright (c) 2013 Bento Box Projects, Inc.

Research Ethics Board

November 5, 2013

Dear Emma Westecott,

1. RE: OCADU 143 “ScreenPerfect: an HTML5 video management project.”

The OCAD University Research Ethics Board has reviewed the above-named submission. The protocol and consent form dated November 5, 2013 are approved for use for the next 12 months. If the study is expected to continue beyond the expiry date (November 4, 2014) you are responsible for ensuring the study receives re-approval. Your final approval number is **2013-42.** Please note that this approval also covers the work of Graduate Student Alex Leitch.

Before proceeding with your project, compliance with other required University approvals/certifications, institutional requirements, or governmental authorizations may be required. It is your responsibility to ensure that the ethical guidelines and approvals of those facilities or institutions are obtained and filed with the OCAD U REB prior to the initiation of any research.

If, during the course of the research, there are any serious adverse events, changes in the approved protocol or consent form or any new information that must be considered with respect to the study, these should be brought to the immediate attention of the Board.

The REB must also be notified of the completion or termination of this study and a final report provided. The template is attached.

Best wishes for the successful completion of your project.

Yours sincerely,

Tony Kerr, Chair, OCAD U Research Ethics Board

OCAD U Research Ethics Board: rm 7520c, 205 Richmond Street W, Toronto, ON M5V 1V3
416.977.6000 x474

REFERENCES

A list of works cited within this text.

Ackerman, S. & Ball, J. (2014, February 29). Optic nerve: millions of yahoo webcam images intercepted by gchq. Retrieved February 28, 2014, from http://www.theguardian.com/world/2014/feb/27/gchq-nsa-webcam-images-internet-yahoo

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.

Baskerville, P. (1999). Women and investment in late-nineteenth-century urban canada: victoria and hamilton, 1880-1901. *Canadian Historical Review*, *80*, 2.

Benjamin, W. & Arendt, H. (1968). Work of art in the age of mechanical reproduction. *Illuminations*, 214–218.

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

Box, B. (2013, November 23). Iv. Retrieved from https://github.com/jennie/iV

Castillo, L. T. (2008). Natural authority in charles dicken's martin chuzzlewit and the copyright act of 1842. *Ninteeth-Century Literature*, *62*(4), 435–464.

Chance, P. R. & O'Toole, M. A. (1987). The imposter phenomenon: an internal barrier to empowerment and achievement. *Women and Therapy*, *6*, 51–64.

Cixous, H. (1976). Laugh of the medusa. *Signs: Journal of Women in Culture and Society*, *1*(4), 875. Retrieved from http://www.dwrl.utexas.edu/~davis/crs/e321/Cixous-Laugh.pdf

Doctorow, C. (2012). *Pirate cinema*. New York, USA: Tor Teen.

Dunne, A. & Raby, F. (2013). *Speculative everything*. Boston, USA: Massachussetts Institute of Technology.

Ebert, R. (2005). Vide games can never be art. Retrieved from www.rogerebert.com/rogers-journal/video-games-can-never-be-art

Ebert, R. (2010). Okay, kids, play on my lawn. Retrieved from www.rogerebert.com/rogers-journal/okay-kids-play-on-my-lawn

Ensmenger, N. (2010). Making programming masculine. In E. by Thomas J. Misa (Ed.), *Gender codes: why women are leaving computing* (pp. 115–141).

Epstein, H., Leitch, A. & Yee, S. (2013). Psxxyborg. video game. OCADu game::play lab. Canada:OCADu.

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

Glanville, R. (2014, January). *Cybernetics*. Lecture at OCADu.

Homans, M. (1986). *Bearing the word: Language and female experience in 19th century women's writing*. Chicago, USA: The University of Chicago Press.

Klimas, C. (2009). Twine. Retrieved from http://www.twinery.org

Martin, B. (2012). Design charettes. In B. Martin (Ed.), *Universal methods of design : 100 ways to research complex problems, develop innovative ideas, and design effective solutions* (pp. 58–59). Beverly, Ma.

Merchant, B. (2014, January 21). Maybe the most orwellian text message a government's ever sent. Retrieved January 21, 2014, from http://motherboard.vice.com/en_ca/blog/maybe-the-most-orwellian-text-message-ever-sent

Netter, G., Lee, A., Womark, D. & Zemeckis, R. (2012). Life of pi. Motion Picture. Fox 2000 Pictures. USA: Fox.

Norman, D. A. (2002). *The design of everyday things*. USA: Basic Books.

In the beginning there was NCSA Mosaic.... (1994, April 6). Retrieved January 13, 2014, from ftp://ftp.ncsa.uiuc.edu/Mosaic/Windows/Archive/MosaicHistory.html

The Agile Manifesto. (2001). Retrieved from http://www.agilemanifesto.org

Gone Home. (2013). video game. The Fullbright Company. USA.

House of Cards. (2013). Television. Netflix. USA: Netflix.

Saint's Row Four. (2013). video game. THQ, Volition. USA: THQ.

Essential Facts About The Computer Game Industry. (2014, February 24). Retrieved June 11, 2013, from http://www.theesa.com/facts/pdfs/esa_ef_2013.pdf

Game Developer Demographics: An Exploration of Workforce Diversity. (2014, January 12). Retrieved October 1, 2005, from http://archives.igda.org/diversity/IGDA_DeveloperDemographics_Oct05.pdf

LASERDISC ARCADE PROJECT AKA CLASSIC FMV GAMES 2.0. (2014, March 14). Retrieved from https://www.youtube.com/playlist?list=PL80FE4E85EF2A41EB

Twitch Plays Pokemon. (2014, February 14). Retrieved from http://www.twitch.tv/twitchplayspokemon

Paul, C., Leeson, L. H., Manovich, L., Sawon, M. & Simon, J. F. (Eds.). (2013, November 1). From virtual to real – successfully translating digital work to a collection context. Leaders in Software Art Conference. Retrieved from http://softwareandart.com/?page_id=1215

Plant, S. (1997). *Zeros + ones: digital women and technoculture*. London: Fourth Estate Ltd.

Schulte, B. (2014, March 14). The new domesticity vs. ambition. Retrieved July 22, 2013, from http://www.washingtonpost.com/blogs/she-the-people/wp/2013/07/22/the-new-domesticity-vs-ambitio

Smith, H. (2013). Spaceteam. Sleeping Beast Games. USA. retrieved from http://www.sleepingbeastgames.com/spaceteam/

Summerfield, P. (1984). *Women workers in the second world war: Production and patriarchy in conflict*. New York:USA: Routledge.

TIQQUN. (1999). *Preliminary materials towards a theory of the young-girl*. Boston, USA.: Semiotext(e).

Wolf, M. (2012). *Encyclopedia of video games: the culture, technology, and art of gaming*. Santa Barbara: USA: ABC-CLIO.

BIBLIOGRAPHY

A reading list of relevant works consumed during the production of this text, in whole or in part.

Ackerman, S. & Ball, J. (2014, February 29). Optic nerve: millions of yahoo webcam images intercepted by gchq. Retrieved February 28, 2014, from http://www.theguardian.com/world/2014/feb/27/gchq-nsa-webcam-images-internet-yahoo

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.

Baskerville, P. (1999). Women and investment in late-nineteenth-century urban canada: victoria and hamilton, 1880-1901. *Canadian Historical Review*, *80*, 2.

Benjamin, W. & Arendt, H. (1968). Work of art in the age of mechanical reproduction. *Illuminations*, 214–218.

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

Castillo, L. T. (2008). Natural authority in charles dicken's martin chuzzlewit and the copyright act of 1842. *Ninteeth-Century Literature*, *62*(4), 435–464.

Chance, P. R. & O'Toole, M. A. (1987). The imposter phenomenon: an internal barrier to empowerment and achievement. *Women and Therapy*, *6*, 51–64.

Cixous, H. (1976). Laugh of the medusa. *Signs: Journal of Women in Culture and Society*, *1*(4), 875. Retrieved from http://www.dwrl.utexas.edu/~davis/crs/e321/Cixous-Laugh.pdf

Doctorow, C. (2012). *Pirate cinema*. New York, USA: Tor Teen.

Dunne, A. & Raby, F. (2013). *Speculative everything*. Boston, USA: Massachussetts Institute of Technology.

Ebert, R. (2005). Vide games can never be art. Retrieved from www.rogerebert.com/rogers-journal/video-games-can-never-be-art

Ebert, R. (2010). Okay, kids, play on my lawn. Retrieved from www.rogerebert.com/rogers-journal/okay-kids-play-on-my-lawn

Ensmenger, N. (2010). Making programming masculine. In E. by Thomas J. Misa (Ed.), *Gender codes: why women are leaving computing* (pp. 115–141).

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

Homans, M. (1986). *Bearing the word: Language and female experience in 19th century women's writing*. Chicago, USA: The University of Chicago Press.

Klimas, C. (2009). Twine. Retrieved from http://www.twinery.org

Martin, B. (2012). Design charettes. In B. Martin (Ed.), *Universal methods of design : 100 ways to research complex problems, develop innovative ideas, and design effective solutions* (pp. 58–59). Beverly, Ma.

Merchant, B. (2014, January 21). Maybe the most orwellian text message a government's ever sent. Retrieved January 21, 2014, from http://motherboard.vice.com/en_ca/blog/maybe-the-most-orwellian-text-message-ever-sent

Norman, D. A. (2002). *The design of everyday things.* USA: Basic Books.

In the beginning there was NCSA Mosaic.... (1994, April 6). Retrieved January 13, 2014, from ftp://ftp.ncsa.uiuc.edu/Mosaic/Windows/Archive/MosaicHistory.html

The Agile Manifesto. (2001). Retrieved from http://www.agilemanifesto.org

Essential Facts About The Computer Game Industry. (2014, February 24). Retrieved June 11, 2013, from http://www.theesa.com/facts/pdfs/esa_ef_2013.pdf

Game Developer Demographics: An Exploration of Workforce Diversity. (2014, January 12). Retrieved October 1, 2005, from http://archives.igda.org/diversity/IGDA_DeveloperDemographics_Oct05.pdf

Twitch Plays Pokemon. (2014, February 14). Retrieved from http://www.twitch.tv/twitchplayspokemon

Plant, S. (1997). *Zeros + ones: digital women and technoculture.* London: Fourth Estate Ltd.

Schulte, B. (2014, March 14). The new domesticity vs. ambition. Retrieved July 22, 2013, from http://www.washingtonpost.com/blogs/she-the-people/wp/2013/07/22/the-new-domesticity-vs-ambitio

Summerfield, P. (1984). *Women workers in the second world war: Production and patriarchy in conflict.* New York:USA: Routledge.

TIQQUN. (1999). *Preliminary materials towards a theory of the young-girl.* Boston, USA.: Semiotext(e).

Wolf, M. (2012). *Encyclopedia of video games: the culture, technology, and art of gaming.* Santa Barbara: USA: ABC-CLIO.