Obfuscation
Frederik Thielke Mølgaard

Perhaps the most common and accustomed way of understanding code, and the aesthetics it contains, is through computer programs that compute in a expected way, following clear and clarifying coded instructions written by a programmer. These programs can be read and understood by other programmers as an intelligible and logical set of instructions. This paper will function as an overview of a less conventional, although, still widely established practice of programming and software development. Examining the history of obfuscation, unveils a span of practices, each with accommodating purpose and aesthetic style; This paper will undertake an examination of the similarities and diversities of this evolving and esoteric concept. It will span from early practices of coding and philosophical speculation, to contemporary, protesting software art. With this as a foundation, this paper will discuss the appliance of obfuscation as a contemporary tool for expressing political and ethical issues, and the potential for a future field of study.

Before the exploration of obfuscation as a concept, a short dictionary reference to the term from where obfuscation has its origin, is required. Obfuscation derives from the verb to obfuscate which has several, but similar meanings; To render obscure, to darken, or to confuse.1 This definition has clear parallels to the coming documentations of obfuscation in the discourse of software studies, and more broadly, Computer and Information Sciences.

In the field of software studies, the concept of obfuscation has so far been conceptualised to have emerged in the digital revolution around the 1980’s, where amateur programmers began programming one-line programs, primarily for training and for sports. This way of obfuscating the programming language itself is more commonly seen under the terms obfuscated code or obfuscated programming.2

Since the 80’s, competitions in several different programming languages has emerged. One of the oldest, yet still recognised of these contests is the International Obfuscated C Code Contest, more prominently known as IOCCC. The goal of this competition, is to write a short, yet original program, wherein the contests judges admire programs that illustrate some of the subleties of the C language, new aesthetics of obfuscation and generally showing off different programming styles in C.3 Following the first IOCCC contest in 1984, several other examples of obfuscated code and programming has been adopted i.e “naming obfuscation”. This term points toward the practice of experimenting with making code obscure, hard to read and understand, to point out that programmers aren’t forced to use signs and symbols in a static way, but rather in a dynamic way closer to the semantics of our natural languages, as it is also practiced in the kindred practice of writing weird languages.4 To exemplify the use of naming obfuscation, one can imagine how switching between letters with respectively lower- and uppercasing like “o” and “O”, instead of zero (0), can be useful for the purpose of making both the code itself more confusing and more likely to be misinterpreted. Although I only exemplified obfuscated code contest in the language of C, it’s relevant to point out here that much of the practices and contents in the field of obfuscated code and programming, are naturally also challenging the boundaries of and syntactic levels of code that is language dependent.

Another more technical example of obfuscated programming, although not obfuscated through naming obfuscation, is yet another one of Michael Matea’s and Nick Montfort’s
documents. This example is done in the programming language C, and is readable even though you don’t have much knowledge regarding the language C itself; 5

main()
{
write(0,"hello, world!
",14);
}

In this program, the function write is called and contains 3 arguments; First 0 is defining that this following writing will be written with default output; The next argument represents the characters “hello, world!”, with quotation marks to border them; The third argument 14, determines the length of the string. Subsequent an arguably transparent code, this following code will execute the exact same visual output, yet contains, as is seen below, a more complex and perhaps more confusing code compared to the previous example;

int i;

main()
{
for( i = 0; i < 14; i++) {
write_one_letter("hello, world!\n" + i);
}
write_one_letter(letter) {
write(0,letter,1);
}

In this piece of code, the visual output is equally the same as the previous code, but the difference here is that the programmer have increased the complexity within the code, simply by using a for-loop to obfuscate the code itself.

As visualised above, the aesthetic style of obfuscated code is identified by the practice of coding unconventionally, making the code harder to read by humans, yet is executed correspondingly by the computer.

Nick Montfort depicts the relation between obfuscated programming as a practice, the aesthetic pleasures it reveals to the programmers, and in a way, the motivation behind the writing of these esoteric programs:

“Writers of obfuscated code strive to achieve latter, crafting programs so that the gap between human meaning and program semantics gives aesthetic pleasure.” 6

In the community of programmers and related practitioners, Donald Knuth was arguably one of the first to articulate the pleasures and amusing aspects of programming in 1974. During a talk he noted how he enjoyed writing software that could fit on one single punched card in the programming language called APL. 7 The constraint of using only a limited amount of characters when writing one-line programs was in itself an obscure way of programming, since programming languages allow unlimited amounts of code, only limited by the hardware; the corpus of the computer, and the competence of the programmer.

To sum up, all the previously mentioned documentations of obfuscation, obfuscated code and its subcategories, has merged on the foundation and motivation of fun and exploration, based on Donald Knuth’s, Michael Mateas’ and Nick Montfort’s statements.

Obfuscation
These early examples also indicate that the aesthetic value derives from the code itself; This is also supported by the code examples in 10Print. Although the examples in 10Print aren’t obfuscated code, the visual output of their one-line programs arguably rely’s on aesthetic pleasure through writing and watching the program execute. Reviewing the example of the one-line program “10 PRINT CHR$(205.5+RND(1)); : GOTO 10” that produces a visualisation of a maze, using only the characters “/“ and “\”, has clear similarities to the purpose of writing program both in the sense of aesthetic pleasure and through the explorative approach to the semantic gap between the interpretation of the human and the computer that seems to be the motivation behind these programs.

In more recent papers, new theories adopt obfuscation as a tool for resisting and providing ambiguous data as a resistance to digital surveillance, including data-mining, aggregation and analysis. Finn Brunton and Helen Nissenbaum theories on the use of obfuscation as a tactical tool for the purpose of, as the title express: “Vernacular Resistance to data collection and analysis”. As a framework for their argument, Brunton and Nissenbaum presents a factual scenario of a community that obfuscate the data of their local grocery chains’ “loyalty cards”, which was intentionally designed to collect data based on what their customer’s bought, to personalise advertisements and special offers. The motivation behind the use of obfuscation in this scenario, to produce ambiguous data collections with the intention of creating misleading and invalid data, because the community felt a loss of privacy. The anxiety in this community was clear and transparent, looking at their obfuscation protest now. The fact that a loss of privacy regarding what one bought, and the unidentifiable purpose of the larger data-collections aggregated was undeniable. One could easily imagine how this lack of transparency made the community anxious. This metaphor can, be interpreted as an analogy to a wide range of other societal transaction happening on the digital super-medium. From the essentials services of checking emails to social networking, it is safe to assume that every major business has adopted one or several ubiquitous data-collection and analysis tools, hidden behind a user-friendly interface.

Succeeding this examination from early to contemporary documentations of obfuscation, the following section will be an analysis of 3 selected examples of software that express a lack of privacy, in a growing, digital surveillance culture. Through the use of obfuscation, but in a manner yet to be defined. In connection to the assumption of how software can contain a representation of ideology, described by Wendy Hui Kyong Chun, the following pieces of software will be interpreted on this foundation. To outline the metaphysical assumption of what ideology really is, the definition in this paper, relies on the describe by Louis Althusser as: “a representation of the imaginary relation of individuals to their real conditions of existence.”

As previously described, obfuscation already has a documented history in both the analog and digital realm; From the dictionary reference to obfuscate, to early obfuscated code contests. But in relation to the use of obfuscation as a tool for obscuring data, Daniel C. Howe argues that obfuscation was first used, in a direct application towards the monetisation of human activity on the internet, by a small browser-extension, written by himself and Helen Nissenbaum, in 2006, named TrackMeNot. The telos of this piece of software is to help protect users from surveillance and data-profiling when using popular search engines such as Google, Bing and Baidu. The way TrackMeNot functions is through the use of what he calls ghost queries - one can imagine these ghost queries as decoys for search engines, making aggregating less accurate. So whenever a user uses her preferable search engine, TrackMeNot creates a trail of decoys that differs from the original search. This makes identifying a specific user, and aggregating their data much harder to do in an accurate way. Although the user is still being monetised, the purpose of TrackMeNot is, according to the creator’s own description:
“Amplifying users’ discontent with advertising networks that not only disregard privacy, but also facilitate the bulk surveillance agendas of corporate and government agencies….”

Even though one might argue that TrackMeNot doesn’t present any definitive solution to the fact that users are still being monetised while browsing, it’s simply not the solely purpose of this singular piece of software. Instead TrackMeNot functions as a tool for protecting privacy, while being a part of a series of expressive software artefacts and art-projects that manifest the lack of privacy when browsing the web through artistic and computational methods. Just like the role of the arts, Florian Cramer argues that to understand software art and its critiques we must; “Pay attention to how and by whom programs were written.” By this, Cramer states that the output of an executed code is not the only thing worth studying. Instead Cramer suggests that both the code itself, and/or the creator of a specific program also needs to be taken in account for analysing a particular program. This postulate is supported by observations of how black box interfaces is really creating a seductive mirage for its perceiver, while its true processing is opaque to the one interacting with it.

From a philosophical perspective, Ben Schneiderman coined a term in 1980’s to describe exactly these kinds of interfaces as a form of “Direct manipulation”. Direct manipulation states that many popular interfaces are conventionally made to manipulate the user to think that they are in control of the computer. Turning complex processes within the specific software to a simplified and understandable interface that utilises the engagement of positive emotions for the human user. Direct manipulation is therefore a term one can use in this context to understand how engaging interfaces utilises human emotion, clarifying software’s purpose and use by making software understandable through human analogies, embedded in its interface. Though direct manipulation is intended to clarify the purpose and the use of software, the purpose of contemporary obfuscation-based tools indicate a critical relation between interfaces and the intended purpose of obfuscation as tool and tactic.

Daniel C. Howe presents a model of obfuscation goals. To this model, Howe identifies 3 major goals; Protection, expression and subversion. (see figure 1. below) The first goal, protection, refers to what extent the chosen tool protects the user, and in the case of TrackMeNot, against data-mining and data-profiling. The second goal, expression, refers to the amount in which the tool allows, and/or articulates, the amplification of the user. This makes the obfuscation-based tools an instrument that facilitates more than just functionality; social, cultural, political and ethical perspectives are being articulated through the use of these tools in the sense that businesses, like Google, can analyse the amount of obfuscated tools being used as a countermeasure to avoid surveillance and data gathering.
The third and last goal, *subversion*, refers to what extent the obfuscation tool tries to subvert the particular system the tool has been created to obscure. In a sense, this third goal can be understood through the analogy of free speech in the sense that these tools can be understood as a critique of a specific system or ideology a system represents, based on Chun’s assumption that software has become a metaphor for ideologies, among others. As illustrated by the arrows in this model, the goal of an obfuscation tool or tactic isn’t necessarily a single one.

To exemplify this, another browser extension named Adnauseam uses a similar tactic to TrackMeNot with the purpose of making surveillance and user-profiling inaccurate. Adnauseam obfuscates data while browsing, by silently and invisible to the user, clicking every advertisement to encounter throughout the web. By clicking endlessly on every ad a user encounter throughout daily web-searches, Adnauseam creates a metaphorical protective sphere that encapsulates vast amount of data of various advertisement databases, making aggregating the user’s data for the purpose of profiling with an outcome of a non-useful result.

Furthermore, the creator’s of Adnauseam describes on their webpage that the design of this obfuscation tool was also intended to function as a tool for criticising user’s disregard for advertising networks and surveillance culture.\(^{17}\)

Looking at Adnauseam in regard to the obfuscation goals, Adnauseam encompasses each goal. First of all, Adnauseam protects the user from being targeted and analysed successfully by surveillance agendas. Secondly, the software expresses critique and discontent for the contemporary tech-industry and the growing business of data aggregation and profiling of users. Thirdly, Adnauseam can be interpreted as a tool for amplifying the power of the user, based on the statements by the creator's themselves, with the intention of restoring the freedom to click wherever one desires without fearing a surveillance system that monetised and aggregate your every move.

Analysing obfuscated code and obfuscated programming with Howe’s model of obfuscation goals would seem irrelevant if one was to argue that these concepts had anything to do with countering digital surveillance, but several equalities emerges regarding its purpose. In the gap between human and computer semantics, the practice of obfuscated code conceals the true purpose of a program, whilst being a puzzle to read and understand for humans. In this regard, it’s almost like the practitioners of obfuscated code sought to protect or conceal the true meaning of a specific program. Secondly, the obfuscation goal, expression, also fits rather naturally into the practice of obfuscated code, because the emergence of this practice became a cultural and social, esoteric community that found pleasure in writing obfuscated code for fun. Thirdly, the notion that obfuscated coding as a practice, is a small and esoteric one in comparison to the conventional way of programming, emphasises the goal of subversion. It does so by breaking down the conventional way that code is understood and written, subverting the larger system of coding practices.

On the contrary, the present practice of using obfuscation tools indicate that because the practice of producing obfuscation tools doesn’t necessarily obfuscate its own code, the connection between it and the practice from which it originated, is too sought. Although a discussion could develop this notion further, these two practices appear related in connection to the purposes in which these practices were developed, as well as their aesthetics.

While these two previous examples of obfuscation-based tools express countermeasures and discontent to our surveillance culture, other software addresses this with an aspect closely related to the traditional obfuscated code; and that is through fun.
Vortex is another browser-extension that creates a game, which makes the player take control over how networks and surveillance agendas identify you. It does this by making players swap data to disguise identity and location. The data Vortex incorporate in its game is cookies, IP-addresses and geo-locations. To shortly elaborate on this, cookies is a bit of data that functions as a part of a standard HTTP protocol. It functions, similarly to IP-addresses and geo-locations in the sense that this data is perhaps the most conventional data that is recorded and stored by networks to identify who the user is, and to monetise activity. The game, Vortex, evolves around playing as other users, swapping cookies to experience how one can surf the web, while networks identifies you as another person. This measure of changing how a user is perceive by networks on the web, is relational to the techniques adopted by e.g. TrackMeNot and Adnauseam; Producing inaccurate and ambiguous data, which is of no value to surveillance networks because the data becomes obfuscated and misleading.

Although Vortex evolves around fun, and TrackMeNot and Adnauseam appears more or less expressive in a more grave sense, the production of obfuscated, inaccurate data is almost identical, if one compares the goals of each individual project to the model of obfuscation goals. The commonplace for these obfuscation-oriented, browser-extensions is found by the common purpose of creation. Through expressing the lack of privacy in a growing surveillance culture by the tactical use of obfuscated-based techniques, which creates misleading and ambiguous data, resisting communities utilises these tools for their own protecting; Building tools for protecting ones privacy and identity on the web; by users, for users.

This notion of users that tries to obscure information that is meant to be provided to a computational understanding and aggregation, indicates a contradictory relation to the aesthetic and semantic purpose of obfuscated code. While obfuscated code and obfuscated programming challenges the way humans understand the technical functioning of specific programming languages and its aesthetic style, obfuscation-based tools imply to be functioning the other way around. TrackMeNot, Adnauseam and Vortex reversely produces data that is obscure to data-collecting and analysing systems. This indicates that even though Obfuscated code and obfuscated programming as practices serves a different purpose than that of obfuscated-based tools, the techniques that obfuscated-based tools utilises are arguably inspired by the esoteric programming communities of the 80’s.

As the concept of obfuscation has evolved through the last 35 years, a wide variety of practices has put it to use; From early practices of writing obfuscated code as a explorative practice, to the use of obfuscation as tool and tactic for expressing political and ethical issues. As this eluding concept differs in terms of its purpose and use, a relation is arguably present. Based on the historical research documented throughout the last 3 decades, this paper suggest that the relation between the historical development of obfuscation and obfuscation-based tools is found in their common practicing of exploring uncommon techniques for the purpose of gaining new knowledge concerning the gap between respectively computers understanding of semantic logic in code, and a humans understanding of it. Although the documentations on obfuscated code and programming presented in this paper suggest that the art of writing obfuscated code is motivated by fun and exploration, which seems contra the grave purpose of obfuscation-based tools, Vortex contradicts this argument, as the browser extension evolves around play, while enlightening users about the underlying data monitoring processes that is now a standardised aspect of web-surfing.

In addition to the connection between the purposes of the practices of obfuscation, another similarity appears in the relation of the writers of obfuscation practices. In the case of practicing obfuscated code, practitioners formed coding contests to centralise this
esoteric practice and explore cooperatively. And it is by the relation of cooperative practices that relations between the practice of obfuscated code and obfuscation-based tools is founded.

As a foundation for discussing contemporary obfuscation-based tools and its representation of an ideology that oppresses surveillance culture in the new media, drawing upon Chun’s argument that GUI’s (graphical user interfaces) and software function as a analogy to ideology, will be one of the major metaphysical assumptions in this discussion.\textsuperscript{20} Combining this description of ideology, represented in interfaces, with Ben Schneiderman’s term of direct manipulation, a clear correlation appears between the illusion of being in direct control of the computer and the mantra that encapsulates the creation of interfaces that seek to clarify the functioning of computers by relatable representations.

As interfaces are arguably good and very well praised by users, interfaces are undeniably creating a black box mirage, hiding all the processes behind a simplified interface which goal is to please the user, and give them a sense of control - while truly, they are not in control. However the obfuscation-based browser extensions TrackMeNot, Adnauseam and Vortex represent another ideology in software development. These browser extensions ideology is to give the control back to the users by creating awareness of how data gathering processes are invasive while browsing the web, and to obfuscate the systems that identify and analyse us without warning.

Including the creator's of these obfuscation-based software tools correlates to the ideology represented in their software if one lays emphasis on Florian Cramer’s notion of software as an art. What these software project articulates is what the creator's interpret as dysfunctional and political issues in the material medium in which this software serves to exist in; In the abundant mass of quantified data.

Change is inevitable, and the way we understand the world is changing as well. Since year 2000, only one-quarter of all the worlds accumulated knowledge and information was stored digitally. In 2013, only less than 2 percent of the worlds information is stored non-digitally. This is a benchmark that makes Kenneth Neil Cukier and Viktor Mayer-Schöenberger wonder what we can do with these quantified amounts of \textit{big data}.\textsuperscript{21} Many indications are made in this article, yet the primary message is that datafication will undeniably change the way we think about the world. First, big data opens up to the use of massive volumes of data, it will change the way we approach the use of it. Secondly, big data is not only defined as data-chunks with massive scale, it is just as well characterised by the growing need to make everything datafied; our geographical location, words and poetry become data, identity and behaviour becomes patterns, and our relationships are datafied through monitoring of our likes on Facebook, and other social networks.

In this perspective obfuscation serves as a tool to deflect computational intrusion in our lives, or at least articulates the growing discontent for involuntary surveillance. Edward Snowden, arguably the most renowned whistleblower, argues that our privacy doesn’t have to become inflate for security agendas to succeed;

\textit{“The people looking at this data are looking for criminals. You could be the most innocent person in the world, but if somebody programmed to see patterns of criminality looks at your data, they’re not going to find you – they’re going to find a criminal.”}\textsuperscript{22}

In the previously documented software projects, the issues of today concerns the question of how users can avoid being monetised and identified, while browsing the web. Furthermore, the lack of transparency in the development of new techniques that serves as tools for collection and analysing data, is another issue these tools serves to enlighten the public to. While these political issues are likely be resolved under political agendas, these projects can serve as educational tools for empowering digital literacy to society. However,
this paper has been primarily concerned towards the examining of the relation between different kinds of obfuscation, which is critical to note as the political and ethical issues obfuscation-based tools addresses calls for further examination. As this paper theorises on the success and failure of the use of obfuscation, it worth noting here at the end that no political standpoints should be carelessly drawn from here.

To conclude on obfuscation, its historical development, and the contemporary obfuscation-based tools, 3 major points has been identified throughout this examination. First, early documentations on obfuscated code and obfuscation-based tools has a relation in their common search for new way of using and understanding the different semantic interpretation of data, respectively by computers and humans. However, Obfuscation-based tools utilises obfuscation by manipulating data to mislead and create ambiguous results for data-collecting and analysing systems, while the practice of obfuscated code and programming works the other way around; writing code that seeks to obfuscate code for the purpose of exploring the gap between the human and the computers semantic interpretation of code.

Secondly, this paper suggest that obfuscation-based software tools function intentionally to their creator’s goal; Addressing issues of digital surveillance, while adopting an ideology of giving the control back to the users by presenting tools for resisting data-collecting and analysing systems through the use of obfuscation techniques that produces misleading data. Thirdly, obfuscation as a field of study, adds new perspectives and tools for empowering user control and digital literacy while addressing issues through software that is multifunctional; It functions both as software art that criticises the conventional use of data collecting and analysing systems, and functions as well as a tool for hiding ones identity with obfuscation techniques, to privately explore the web.

While obfuscation might appear weak and futile compared to e.g. encryption, obfuscation is nonetheless a well documented technique that serves a variety of goals. As this paper indicates, obfuscation is a concept derived from small communities working with digital arts and related academic studies, which by definition encompasses several purposes and aesthetic character. While they all rely on the aesthetic value of its execution, obfuscation is contrary e.g. encryption, a tool that has great potential for obfuscating data in a datafied world. Finally, it appears that obfuscation-based tools have correlations to its predecessor obfuscated code and programming, as it have successfully functioned to address the political and ethical issues of a growing digital universe.
Notes

1. Based on the explanation by the included Oxford Dictionary in every macintosh.

2. The definitions of these concept are partly adopted by Michael Mateas’ and Nick Montfort’s article: “A Box, Darkly: Obfuscation, Weird Languages and Code Aesthetics.”


4. For more on weird languages check Michael Mateas’ and Nick Montfort’s “A Box, Darkly: Obfuscation, Weird Languages and Code Aesthetics.”

5. The two code examples are written in the programming language C, and is extracted from the Matthew Fuller’s, “Software Studies - a lexicon” edited by Matthew Fuller (The MIT Press Cambridge, Massachusetts London, England)

6. Ibid. p.194.


8. Referring to the first example in “10Print”, p.3.


10. The primary metaphysical assumption of this paper is based on Chun’s description of how interfaces and software can be interpreted as a metaphor for ideology, “Programmed Visions”, Daemonic Interfaces, Empowering Obfuscations, Pp.59-95.

11. Ibid.


16. Model used is from the article found at: http://www.aprja.net/?p=2510, and inspired explanation drawn from here as well.

17. Description available at: http://adnauseam.io/

18. For more on the functioning of Vortex, full documentation is available at: http://static.squarespace.com/static/5138079de4b0908a3a9799a1/t/525c8956e4b015de6e623e55/1381796182891/vortex.pdf


22. http://www.amnesty.org.uk/mass-surveillance-us-nsa-edward-snowden-gchq#VXVsOeeKG6o