

Unveiling Interfaces:

A Software Studies Board Game for Algorithmic Literacy

Master Thesis

Handed in to

School of Communication, Art & Technology Aalborg University

Head of Department

Tom Nyvang

and the

Media Arts Cultures Consortium

Course

Media Arts Cultures

Supervisor

MA, PhD. Associate Professor Morten Søndergaard

By

Karla Victoria Zavala Barreda

Willem Adriaan Odendaal

Date of delivery

August 2018

Abstract


There has been growing concern over the role algorithmic systems have come to play in our increasingly digitized lives. For not only are algorithms obscured from public attention, blackboxed behind the software interfaces through which we interact with them, but so too are the ways in which they mediate, moderate, and augment the world hidden from public scrutiny. Consequently, there has been a wide-spread call from civic organisations, academics, and media critics for initiatives that can foster public algorithmic literacy. Such a literacy would allow those who engage with algorithmic systems in their daily lives to become more aware of, critical, and knowledgeable about how, when, and to what ends these automated systems impact their lives. Working with a practice-led research method we responded to this call through developing a critical board game designed to contribute to such a public algorithmic literacy. The board game, *Unveiling Interfaces*, was designed through codifying game elements and mechanisms with theoretical insights from the field of software studies. Software studies, we argue, can contribute to a public algorithmic literacy through its digital materialist approach, offering an understanding of how algorithms have a material relation to the world through their socio-technical assemblage. Read from a critical theory perspective, software studies moreover, offers a critique of the computational ideology propagated by these blackboxed algorithms, and thus address the underlying social, political, and economic effects these invisible systems have on the world. In conjunction with software studies, this thesis draws on approaches from game design, critical play, critical pedagogy, as well as critical theory throughout an iterative process of design and research as we aim to offer a practical and theoretical response to the call for algorithmic literacy.

Keywords: *board game, critical play, software studies, algorithmic literacy, euro-games, practice-led research, critical pedagogy, digital materialism*

Declaration of Authorship

We Karla Victoria Zavala Barreda and Willem Adriaan Odendaal hereby certify that the thesis we are submitting is entirely our own original work except where otherwise indicated. We are aware of the University's regulations concerning plagiarism, including those regulations concerning disciplinary actions that may result from plagiarism. Any use of the works of any other author, in any form, is properly acknowledge at their point of use.

8, August, 2018



Signatures:

List of Content

List of Figures	1
List of Abbreviations	2
Foreword	3
INTRODUCTION	5
CHAPTER 1	9
1. From Algorithmic Culture Towards an Algorithmic Literacy	9
2. Defining Algorithmic Literacy	12
3. Algorithmic Blackboxing, Transparency and Literacy	14
4. A Source of Critical Perspective on Algorithms	18
5. Research Question	21
CHAPTER 2	22
1. Methodology	22
1.1. Black Box Discussion of Algorithms	22
1.2. Literacy as Competences	24
1.3. Critical Theory	25
1.3.1. From Critical Theory to a Software Critique	26
1.3.2. A Literacy Model for Critical Theory	30
2. Method	33
2.1. Practice-led research.....	33
2.2. Selecting the Creative Practice Design Space	33
2.3. Critical Board Games	36
2.4. Design and Research Process	41
CHAPTER 3	45
1. Critical Play Goals.....	45
1.1. Game Design Goals	45
1.2. Value Goals	46
2. Framework of Play	48
2.1. Object of The Game	53
2.2. Setup	54
3. Game Mechanics: Operational Rules and Game Elements	55
3.1. Phase 1:	56
3.1.1. Action 1: Collect Software Tiles:	57
3.1.1.1. Software as algorithms.....	58
3.1.1.2. The Constituency of an Algorithm	61
3.1.1.3. Code and the materiality of algorithms	62
3.1.2. Action 2: Draw a new Developer Brief	65
3.1.3. Action 3: Develop and Publish an App.....	67
3.1.3.1. Interfaces: APIs, Code, and GUI	70
3.1.3.2. Software Tile: User Interface Side.....	75
3.1.3.3. User Interface: Blackboxing as Commodity	78
3.2. Phase 2: Event Cards OR Collecting Revenue	83
3.2.1. Event 1: Get a notification to ‘check your code’	84
3.2.1.1. Software as Socio-technical Assemblage	86
3.2.1.2. Grammars of Action	87

3.2.1.3. Ideology and Technocracies	89
3.2.1.4. Unveiling Interfaces Through ‘Checking Your Code’	90
3.2.2. Event 2: Collect Revenue	95
3.3. Resolution and end-game scoring	96
4. Critical Play in <i>Unveiling Interfaces</i>	96
4.1. Discomfort Design	97
4.2. Codification	98
CHAPTER 4	101
1. Evaluation Design	101
2. Playtests	101
2.1. Playtests as Internal Design Review	101
2.2 Playtest with Different Audiences as Project Evaluation	104
2.2.1 Sampling	105
2.2.2 Data Gathering	106
2.2.3 Data Analysis	108
3. Evaluating Critical Play Goals	108
3.1. Revising Game Design Goals	108
3.1.1. Revising Meaningful Play.....	111
3.2. Verifying Value Goals	114
4. Suggested Solutions	117
CHAPTER 5	124
1. Limitations	124
2. Reflections	126
3. Conclusions	130
BIBLIOGRAPHY	135

List of Figures

Figure 1.1. A 2018 screenshot of Google autocomplete search

Figure 2.1. The iterative cyclic web of practice-led research and research-led practice

Figure 2.2. Flanagan's critical play design model

Figure 3.1. Illustration of the game setup for two players

Figure 3.2. The front-side and back-side of a red 'Recommendation' app

Figure 3.3. The two code-sides of the Newsfeed Software Tile

Figure 3.4. Part of the spreadsheet we used to develop our Software Tiles

Figure 3.5. Examples of notation of data structures in Python

Figure 3.6. A Developer Brief's color combination published

Figure 3.7. Playing board design with an app published onto it

Figure 3.8. Software Tile's UI side

Figure 3.9. Week Marker on the board's Calendar

Figure 3.10. An example of an Event Notification Card

Figure 3.11. Peppa Pig Event Card

Figure 3.12. A special AppMarket related Event Card

Figure 4.1. Types of playtesters appropriate for each stage of prototyping

Figure 4.2. Complete second version of prototype 1 after Internal Design Review 1

Figure 4.3. MVP: Complete version 2 of prototype 2 after the third internal design review

Figure 4.4. Playtest Sessions 1

Figure 4.5. Playtest Sessions 2

Figure 4.6. Playtest Sessions 3

Figure 4.7. Event Notification Card "#1 Tool for Terrorist Recruitment!"

Figure 4.8. Terror Bull's 'Our Sonofabitch'

Figure 4.9. First wireframes for Prototype 2 of the code-side of the Software Tile

List of Abbreviations

API - Application Programming Interface

APK - Android Application Package

DPA - Data-Pop Alliance

GAFA - Google Apple Facebook Amazon

GUI - Graphic User Interface

HCI - Human-Computer Interaction

IAAWGMOOH - I'm an Agricultural Worker, Get Me Out of Here

IPA - iPhone Application Archive

MVP - Minimum Viable Product

PARC - Palo Alto Research Center

SDK - Software Development Kit

SOAS - The School of African & Oriental Studies (University of London)

TUI - Tangible User Interfaces

UI - User Interface

VUI - Voice User Interfaces

WIMP - Windows Icons Menus and Pointers

WWWF - World Wide Web Foundation

Foreword

We would first and foremost like to thank our thesis supervisor Morten Søndergaard for the enthusiasm he has shown for this research project from the first moment we approached him with our prospective topic. From there on out he has given us continued encouragement, constructive feedback, and have often prompted us to painfully (but appreciatively) reevaluate our academic approach and assumptions. This process has been an inspiring learning experience for both authors thanks to his mentorship.

Furthermore, we would like to thank our lecturers at Aalborg University and the University of Łódź who has given us feedback and advice during our research process, especially Elizabeth Jochum, Palle Dahlstedt and Katarzyna Prajzner. We would likewise extend our thanks to our lecturers at Donau University Krems as well as the Erasmus Mundus Joint Master of Arts degree in Media Arts Cultures consortium for the opportunity to write this thesis.

Moreover, we would like to thank our playtest participants who not only gave us their time and attention, but also extremely encouraging and insightful critique and feedback. It is in playing our board game with real players that made all the research we have been conducting feel relevant. We would also be remiss to not acknowledge the authors whom we drew on throughout this research. It was the work of Berry, Flanagan, Pold and Anderson, Manovich, Zimmerman and Salen, Kitchin, and others that gave us so many exciting and inspiring research to draw from and made the process of writing this thesis an extremely stimulating experience.

To my mother, I would like to say thank you for all the phonecalls which were welcome distractions from the thesis. To my father, I would like to say thank you for all the advice and encouragement you have given over the last 6 months. To both of my parents, thank you for the continued support regardless of what challenge I undertake in my life. I would also like to thank my co-author for her diligence, hard work, and extraordinary insights throughout the research process (despite our occasional disagreements).

Adriaan Odendaal

I would like to thank my mother Julia, who brought into my life the first computer I ever saw and encouraged me to explore it. To her I now say: here are my findings. To my aunt Elena, that after her departure left me full of ideas on how to make her proud. To my sister Maria Gracia, for her relieving conversations during the thesis writing. And to Adriaan, my thesis co-author, without your shared enthusiasm and hard work, this project wouldn't be as exciting as it is.

Karla Zavala Barreda

INTRODUCTION

In recent years there has been an increased concern over how the unseen algorithmic systems underlying digital technology are influencing our lives—mitigating, mediating, abstracting, and augmenting everything from our personal shopping habits to our political systems. While academics have coined the terms ‘algorithmic culture’ to describe this changing techno-cultural context, and ‘critical algorithmic studies’ to reconcile the “growing critical literature on algorithms as social concerns” (Gillespie & Seavers, 2016, n.p.), beyond academia there has also been growing concern on this topic. Recent non-fiction books such as Cathy O’Neil’s *Weapons of Math Destruction* (2016) and Safiya Noble’s *Algorithms of Oppression: How Search Engines Reinforce Racism* (2018) have popularized the discussion amongst a non-technical public, while investigative journalists have increasingly turned their attention to addressing and exposing problematic algorithmic systems such as found in *ProPublica*’s ‘Machine Bias’ investigative journalism series¹. It is thus not surprising that there has been a growing call from different corners of society for the development of a public algorithmic literacy that could empower a largely non-technical and academically inundated public that engages with these increasingly complex, problematic, and inherently invisible systems on a daily basis. The consequent aim of this thesis is thus not primarily to make an academic contribution to ‘algorithmic culture’ discussions or ‘critical algorithmic studies’, but rather to respond to the need for such critical discussions to take place outside of the domain of academia as well and contribute to a public algorithmic literacy.

The term ‘algorithmic literacy’ as related to social empowerment (rather than as an ostensive outcome of computer science pedagogy) has only come into perceptible use in recent years. Yet, despite the underdevelopment of the concept, in this thesis we argue that it is nonetheless possible to make a legitimate contribution to a public algorithmic literacy through framing it within the theoretical framework of software studies. Thus, rather than striving to offer a definitive definition of, or conclusive solution to, public algorithmic literacy—in its limited scope our research

¹ Veteran tech reporter Erin Griffith writes in *Wired Magazine* that tech journalism has in recent years turned away from an optimistic celebration of Silicon Valley, towards the perception that “[a]nything that doesn’t address the thorny questions facing the tech industry feels beside the point” (2017 December 16, n.p.).

aims to investigate and develop one possible way in which a contribution can be made to a public algorithmic literacy. We argue that software studies, through its methodological orientation towards digital materialism, offer a unique theoretical position from which to develop an algorithmic literacy. Thus, to answer our research question “How can software studies contribute to algorithmic literacy?” we ask the following two questions as well:

RQ1: How does software studies foster a critical stance towards algorithms materialized in cultural software?

RQ2: How can insights from software studies be implemented in an algorithmic literacy initiative?

To answer our second research question, however, it becomes necessary to go beyond the academic and theoretical domain of software studies – and beyond the disciplinary domain of this thesis as situated within a humanistic research tradition. Following, we chose to employ a practice-led research methodology, allowing us to initiate an academic investigation through which we could also create a practical project aimed at contributing to a public algorithmic literacy.

It is within this transformative orientation that we decided to create a critical board game called *Unveiling Interfaces*. With game mechanics and content derived from software studies insights and key-concepts and working with critical play and board game design principles, we considered that we could create a dialogical space in which a wider non-expert public could develop a critical stance towards and awareness of algorithmic systems in their daily lives. Thus, we worked with the following hypothesis:

Through developing a critical board game that applies software studies theory, software studies can contribute to a public algorithmic literacy.

After a series of playtests, we concluded that while in its current iteration there are some flaws in *Unveiling Interfaces*, the game could be considered as a prototype still in the game development cycle rather than a final publicly promotable product. While results from the evaluation of this board game were thus not conclusive, it did offer fruitful insights that could allow us to prove our hypothesis if implemented in a

successive iteration. In this, the research delivered some results that were not only insightful to us as researchers and practitioners but would hopefully also be considered a modest contribution to the fields of game design and software studies, as well as to future algorithmic literacy initiatives.

For the benefit of the reader, the thesis is structured in the following way: **Chapter 1** starts with a literature review in which we give attention to the academic interest in what has been called ‘algorithmic culture’, before covering the correlative attention that media critics, think tanks, and civic organizations have given to the problematics of this techno-cultural context. From there, we discuss the necessity for practical responses to this context, with a central focus on the most underdeveloped area of response: public algorithmic literacy. As part of the literature review, we outline a working definition of algorithmic literacy, possible initiatives that can be seen to respond to this definition, as well as the perceptible contribution software studies as a theoretical field can make to an algorithmic literacy project. Following this we formulate our research question.

Chapter 2 comprises our theoretical and methodological approach to our research question. We start with outlining the need of a humanistic approach towards what is usually the prerogative of technical domains such as computer science. We argue that this also extends to the conception of ‘literacy’ which can be argued to necessitate socio-cultural competencies. For us, this implicates a critical theoretical orientation towards our research question, as it allows us to develop a critical cultural reading of algorithmic technology. It is through this that we argue that software studies can be incorporated into our theoretical approach. An orientation towards critical theory also allows us to draw on critical pedagogy as a model for literacy situated within cultural criticism.

The second half of Chapter 2 is devoted to outlining our research method. As we are working with algorithmic literacy as practical concern, we argue for a practice-led research methodology that will allow us to work rhizomatically between theoretical research and the application of that research into a practical project. Within this method we then outline the criteria for choosing a project, and the consequent format of the project: a critical European-style board game. The chapter ends with our rationale for choosing this approach, consisting of a brief discussion of board games and critical play principles, as well as how we will implement this

approach through a critical play game design process in conjunction with software studies academic research.

The primary research of this thesis is conducted in **Chapter 3**. The chapter starts with an outline of the design goals and play values for the project. These are given as a reiteration of our definition of algorithmic literacy from Chapter 1, reformulated within our board game design method. We then continue to set out the frame of our board game – which comprises the overarching theme of the game. It is here that we start developing our primary theoretical research as well, drawing on software studies texts and insights. We continue this chapter by discussing the development of the game mechanics, and how they employ software studies theory. Systematically we work through how we designed the game elements, mechanics, and progression of play as we conduct our research, ending with a discussion of how our game is designed to evoke meaningful and critical play.

Chapter 4 starts with the evaluation of our project which consists of playtesting our game to assess whether it meets our design goals and fulfils our play values. This is still part of the design process, but formulated within the context of academic research as an evaluation. Following the description of the evaluation is a discussion of the results and whether they can be related to the development of an algorithmic literacy in players. This discussion includes an investigation into how the project should be improved.

The thesis then concludes with **Chapter 5** our research limitations of and final reflections on both our practical and academic research. Though the formal conclusion of the thesis closes this chapter, as part of the practice-led research we also include the board game itself in terms of the player rule book and printable game elements (amongst other things) as appendixes.

CHAPTER 1

Literature Review

1. From Algorithmic Culture Towards an Algorithmic Literacy

A growing field of scholarship is describing contemporary—especially Western—culture as constituting of ‘algorithmic culture’ (Dourish, 2016; Gillespie, 2016; Kitchin, 2017; Striphas, 2012). Ted Striphas, appropriating the concept from Alexander Galloway’s writing, was the first to employ the term ‘algorithmic culture’ in reference to the techno-cultural context “in which computers, running complex mathematical formulae, engage in what’s often considered to be the traditional work of culture: the sorting, classifying, and hierarchizing of people, places, objects, and ideas” (cited in Granieri, 2014, n.p). Although algorithms have been part of society since the invention of mathematic, the rise of a veritable ‘algorithmic culture’ can be ascribed to the ubiquitous cultural rise of computers over the last few decades. Within this context, throughout this thesis we thus understand algorithms as the basic instructions or steps a computer software follows to produce its outputs (Kitchin, 2017, p. 14). As Roberge and Seyfert write in the introduction to the 2016 research compendium *Algorithmic Cultures*²:

Algorithms have expanded and woven their logic into the very fabric of all social processes, interactions and experiences that increasingly hinge on computation to unfold; they now populate our everyday life, from the sorting of information in search engines and news feeds, to the prediction of personal preferences and desires for online retailers, to the encryption of personal information in credit cards [...] In fact, the list of things they can accomplish is rapidly growing, to the point where no area of human experience is untouched by them. (2016b, p. 1)

Yet it is not merely the academics of ‘algorithmic culture’ that are responding to this perceptible paradigm shift. In 2017 the Pew Research Center and Elon University published an expansive survey of 1302 “technology experts, scholars, corporate practitioners and government leaders” reflecting on—as suggested by the

² *Algorithmic Cultures: Essays on Meaning, Performance, and New Technology* (2016) is a “cutting edge” research compendium providing “in-depth and wide-ranging analyses of the emergence, and subsequent ubiquity, of algorithms in diverse realms of social life” (Roberge & Seyfert, 2016, n.p.).

title of the survey—the *Pros and Cons of the Algorithm Age* (Rainie & Anderson, 2017, p. 4). In their reporting, while finding “agreement that the abundant positives of accelerating code-dependency will continue to drive the spread of algorithms” in society, sub-themes such as the following also emerged: “algorithms are primarily written to optimize efficiency and profitability without much thought about [their] possible societal impacts”; “algorithm creators (code writers), even if they strive for inclusiveness, objectivity and neutrality, build into their creations their own perspectives and values”; and “the datasets to which algorithms are applied have their own limits and deficiencies”—all of which works towards propagating what is being called “algorithmic biases” in society (ibid., pp. 9, 12, 65). In a similar recent research paper for the Rand Corporation public policy think-tank, Osoba and Welser IV cites instances of such algorithmic biases, ranging from “search engine autocompletion routines [that] learn to make incorrect defamatory or bigoted associations about people or groups of people” (Fig. 1.1.) to the “extreme systematic bias” propagated by “a criminal risk assessment algorithm” used in US sentencing hearings (2017, pp. 10–11)³.

Likewise, a 2017 white-paper titled *Algorithmic Accountability* from the digital advocacy research group World Wide Web Foundation (WWWF) draws on literature research, expert interviews, and topic workshops to investigate the growing concern over how “algorithms are controlling the inclusion—and exclusion—of people and information in an increasing number of settings”, granting “algorithms the power to perpetuate, reinforce or even create new forms of injustice” (2017, p. 3). This concern over how algorithms increasingly exercise power in society is reflected in ‘algorithmic culture’ scholarship—what Gillespie and Seavers calls the new emerging field of ‘critical algorithmic studies’ (2016, n.p.). As Rob Kitchin writes in *Thinking*

³ The list of demonstrative examples of algorithmic biases, injustices, and ‘misbehaviour’ are ever growing. Looking at recent journalistic reporting on technology will offer up a myriad of other identified instances of algorithmic biases, such as: *Facebook (Still) Letting Housing Advertisers Exclude Users by Race* (Angwin, Tobin & Varner, 2017 October 28) and *How We Examined Racial Discrimination in Auto Insurance Prices* (Larson, Angwin, Kirchner & Mattu, 2017 April 5) from ProPublica’s extensive ‘Machine Bias’ series; *What happens when an algorithm cuts your health care* (Lecher, 2018 March 21) from The Verge; or *Machines Taught by Photos Learn a Sexist View of Women* from Wired Magazine (Simonite, 2017 August 21) to name just a few. One can also turn to Osoba and Welser IV (2017), Kitchin (2017), O’Neil (2016), or Gillespie and Seaver (2016) for more comprehensive lists.

Critically About and Researching Algorithms (2017), a meta-research paper that has been taken up in this emerging field:

We are now entering an era of widespread algorithmic governance, wherein algorithms will play an ever-increasing role in the exercise of power, a means through which to automate the disciplining and controlling of societies and to increase the efficiency of capital accumulation. (p. 15)

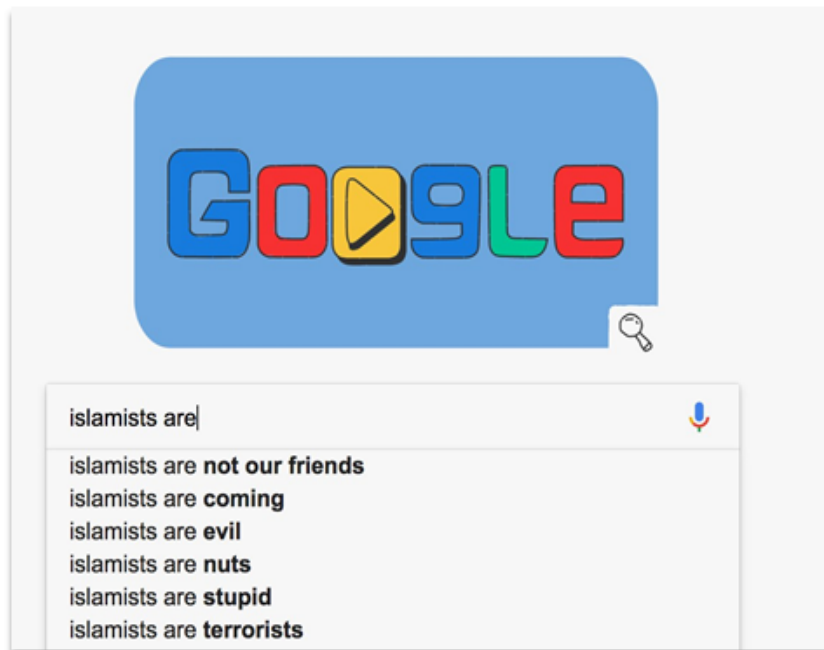


Figure 1.1. A 2018 screenshot of Google autocomplete search: The suggestions shown an example of how search engine algorithms many web-users use every day can unwittingly perpetuate societal prejudices. Reprinted from *Wired Magazine*, by I. Lapowsky, 2018. Retrieved from <https://www.wired.com/story/google-autocomplete-vile-suggestions/>.

For many, this techno-cultural context thus demonstrates a need to move *beyond* academic investigation towards actionable responses or the exercise of social ‘algorithmic justice’ (Diakopoulos, 2014; Osoba & Welser IV, 2017; Rainie & Anderson, 2017; WWF, 2017). As the WWF writes:

Until now, much of the debate on how to accurately identify potential algorithmic bias and harms has occurred either within internal corporate research labs or within the academic research world, and there has been a lack of consensus amongst the broader community regarding what a ‘solutions toolkit’ would look like. (2017, p. 5).

The overarching areas for necessary action that can be identified in these calls for a ‘solutions toolkit’ seems to primarily focus on **algorithmic transparency**, “enabling citizens, consumers, data journalists, watchdog organisations and others to verify and understand the inputs, processes and outputs of a complex algorithmic system to identify evidence of harms as a first step for redress”, and **algorithmic accountability**, “the responsibility of algorithm designers to provide evidence of potential or realised harms” (WWWF, 2017, p. 5, 10). However, a third response can also be identified, although literature on it is comparatively scarce, seldom extending beyond a singular paragraph framed in a larger discussion of algorithmic accountability, namely: **public algorithmic literacy** (Diakopoulos, 2014; Kitchin, 2017; Osoba & Welser IV, 2017; Rainie & Anderson, 2017; WWWF, 2017).

2. Defining Algorithmic Literacy

While the distinct calls for algorithmic literacy are frustratingly brief in their definitions, it is possible to arrive at a provisional definition of what an algorithmic literacy can mean through synthesizing the calls from various of the already cited publications, as well as others who make arguments that can be seen as analogous or supplementary to these call for a public algorithmic literacy⁴. Algorithmic literacy should thus:

- a. **Necessarily be citizen-centric or oriented towards the public:** Central to the conception of algorithmic literacy is that it should constitute a ‘public education’ (Diakopoulos, 2014; Osoba & Welser, 2017; Rainie & Anderson, 2017), or that it should be definitively oriented towards empowering “citizens and citizen-centric groups” (WWWF, 2017, p. 13). Algorithmic literacy initiatives should thus extend beyond privileged access to formal education.
- b. **Foster awareness or consciousness of algorithms in everyday life:** Under the algorithmic literacy theme, one respondent from the Pew Centre survey emphasizes that the “first and most important step” in response to the ‘algorithmic age’ is to develop “better social awareness of who, how, and where

⁴ We explicitly define algorithmic literacy as contextualized by these publications, as opposed to the occasional use of the phrase in computer science-based pedagogy with its explicit focus on technical competencies.

[algorithms are] being applied” (Kutarna in Rainie & Anderson, 2017, p. 16). While also a primary theme in analogous calls for ‘algorithmic awareness’ (European Commission, 2017; Hamilton Karahalios, Sandvig, & Eslami, 2014), algorithmic literacy should “instil literacy about how algorithms function in the general public” (Rainie & Anderson, 2017, p. 15) and allow members of a public “to understand their own participation in algorithmic systems” (WWWF 2017, p. 13).

- c. **Not be contingent on technical competencies:** Some respondents of the Pew Centre survey emphasize a necessary understanding of the “techniques” and “methods” (Rainie & Anderson, 2017, pp. 19, 75) behind algorithmic systems, while Caplan and Reed for the Data & Society research institution draws wholly on a definition of algorithmic literacy “in terms of reading, writing and making algorithms” (2016, p. 8). However, given the socio-technological problematics already discussed, at the same time they hold that such a technical definition “would most likely have limited effects as knowledge about manipulation doesn’t necessarily ‘empower’ better decisions” and can even engender a “a false sense of security” (ibid.). Moreover, while technical comprehension can assuredly be instrumental to an algorithmic literacy as discussed here, Osoba and Welser IV argue that an over-emphasis on technical competencies would make initiatives oriented towards a non-expert public unfeasible. The authors thus argue that a public algorithmic literacy “is not the same as requiring that users understand the inner workings of all algorithms” (2017, p. 23).
- d. **Foster a critical stance towards algorithmic technology:** Given the preceding point, Osoba and Welser IV argue that the impetus of an algorithmic literacy should be to develop public “scepticism” of algorithms and contribute to an awareness that “algorithms can lead to inequitable outcomes” and in that counteract the public’s “uncritical reliance on algorithms” (2017, pp. 2, 23). Diakopolous, cited by the WWWF as well as Rob Kitchin, more poignantly argues that “what we generally lack as a public is clarity about how algorithms exercise their power over us”, resulting in a consequent need for a “critical stance towards algorithms” (2014, p. 2). Given the social implications of algorithmic biases, injustices, and ‘misbehaviours’ that prompted these calls, the

WWWF likewise draws on the Data & Society institute's discussion of algorithmic literacy programs as necessarily enabling citizens to “perceive when or if they or others are being marginalized”, emphasizing the importance of a critical stance towards algorithmic technology's social—as opposed to a merely their technical—dimensions (Caplan & Reed, 2016, p. 8).

Encapsulated in these criteria we see an algorithmic literacy as enabling members of a public to critically reflect on their relationship with the invisible algorithmic technologies increasingly moderating and mediating their daily lives. While none of the proponents of algorithmic literacy give clear recourse to a definition of ‘literacy’ itself, as we will argue in the following chapter, algorithmic literacy as defined above can consequently be located within a specific theoretical conception of ‘new literacies’. Moreover, it should be conclusively stated that it is with some authorial discretion, even though derived from an overview of existing literature, that we have set out the defining criteria of algorithmic literacy above. We thus do not hold that these criteria are exhaustive or authoritative in any way, although we would argue that a project, initiative, or program that fulfils the above criteria could be considered a definite—though not necessarily *definitive*—response to the distinctly identified calls for an algorithmic literacy. In the same vein, existing approaches and initiatives that fulfil some of the above criteria can rightfully be argued to *contribute to* a public algorithmic literacy, as will be discussed below. For importantly, the call for algorithmic literacy is not primarily an academic discussion, but rather a ‘call to action’—a part of the actionable ‘solution toolkit’ espoused by the WWWF.

3. Algorithmic Blackboxing, Transparency and Literacy

From the literature review there are two identifiable problems arising from a perceptible algorithmic culture that might prevent the development of public algorithmic literacy. The first problem is based on the ever-increasing technical complexity of algorithm which “limit[s] the ability to query their processes and decisions, putting citizens and consumers at a high level of risk” (WWWF, 2017, p. 12). This is especially a problem given the increasing ubiquity of machine learning algorithms which often make automated decisions using complex and inscrutable

neural networks that even their engineers have difficulty comprehending (Koh & Liang, 2017, p. 8; Knight, 2017).

The second problem stems from the fact that algorithms are actively and purposely hidden from the public at the code and human computer interaction levels by software producers (Hamilton et al., 2014, Raine & Anderson, 2017). Resulting from the conjunction of these problematics, algorithms are often “referred to as ‘black box’ constructs” (Raine & Anderson, 2017, p. 74). This is a term historically popularized through cybernetics, and notably explained by science and technology studies theorist Bruno Latour as “the way scientific and technical work is made invisible by its own success” so that “[w]hen a machine runs efficiently [...] one need focus only on its inputs and outputs and not on its internal complexity” (1999, p. 304). It is in inhibiting the development of an algorithmic literacy that opening these black boxes consequently become integral for algorithmic literacy. As Hamilton et al. contend, “many see opacity in technologies as a call for inquiry into what processes of debate and concern have been arrested or settled behind the opaque surface of ‘black boxes’” (2014, p.3).

In their white-paper, the WWWF usefully identifies four main areas to be addressed to counteract the blackboxing of algorithms in promoting ‘algorithmic justice’: technical, ethical, policy, and knowledge gaps. Yet, the WWWF also acknowledges that in order to implement any solution it is important to recognize that there is a “shared responsibility of all stakeholders: algorithmic system designers, legal and regulatory authorities, public interest groups and users” (2017, p. 17).

Given these points, we have identified two provisional criteria to map initiatives that can be seen as possible responses to a lack of algorithmic literacy in the public: on the one hand the **area of interest** of an initiative and on the other hand its **stakeholder group**. Taking these criteria into consideration, we have determined a spectrum of actions that can facilitate a public algorithmic literacy. For example, algorithmic accountability initiatives can generally be seen to work within technical, ethical, and policy interests, focusing in the producers’ responsibilities and involving the algorithmic system designers, legal and regulatory authors as target group. However, while liability for algorithmic biases or harms fall to the institutions, Diakopolous argues that computationally literate journalists doing “algorithmic accountability reporting” can act as interpreters that “frame, contextualize, and explain” these increasingly complex and problematic algorithms to a non-technical

public (2014, pp. 5, 29). Outlining a conceptual model “that captures different investigative scenarios based on reverse engineering algorithms’ input-output relationships”, Diakopolous demonstrates how journalism can interrogate algorithmic black boxes and engender a critical public stance towards algorithmic technology (ibid., p. 2). Thus, through the ‘algorithmic accountability reporting’ initiative having public interest groups (journalists) as the stakeholder group and technical, ethical, and policy interests, it can also function as a public algorithmic literacy initiative⁵.

Conversely, contributions to a knowledge gap can also be made with the system designers as stakeholder group, as some researchers from the field of Human-Computer Interaction (HCI) might argue (Hamilton et. al., 2014). While seamless software design in which the presence of algorithms are invisible is often seen as “indicative of successful design”, Hamilton et al. inverts this presupposition in their focus on the algorithmic curation of information sources on Social Network Sites. Reporting on a consequent ‘seamful’ re-design of the Facebook newsfeed user-interface, in which the underlying algorithmic system is exposed to users, researcher Karrie Karahalios states that some participants in their study “were shocked to learn that their feed was manipulated” and “felt empowered” by the re-design (2014, n.p.). This issue is also commented on by Rainie & Anderson when they state that: “[t]he solution is design. The process should not be a black box into which we feed data and outcomes an answer, but a transparent process designed not just to produce a result, but to explain how it came up with that result” (2017, p. 23).

Perhaps closer in pedagogical spirit to an ‘algorithmic literacy’ is the longstanding data literacy movement that has emerged as a mainstream term in relation to the ongoing ‘data revolution’ that started in the early 2010s (Data-Pop Alliance, 2015, p. 11). While data literacy includes “the ability to read, work with, analyse and argue with data”, it also emphasizes empowerment through encouraging “critical thinking and consciousness” towards data-centric technology and emphasizing the “social processes and ethical questions that surround them” (D’Ignazio & Bhargava, 2015, pp. 1,2,3). An example of this can be seen in the MIT

⁵ Such algorithmic accountability journalism can be seen most actively applied in ProPublica’s ‘Machine Bias’ investigative journalism series already cited. Their reporting includes almost didactical articles, such as *Breaking the Black Box: How Machines Learn to Be Racist* (Larson, Angwin, Parris, 2016 October 19), as well as articles explaining their computational investigative processes, such as *How We Analyzed Amazon’s Shopping Algorithm* (Angwin & Mattu, 2016 September 20).

‘Center for Civic Media’ ‘Data Murals’ project that worked against “technology-centered interventions” towards fostering a more critical public engagement with data⁶, answering a “growing demand for greater transparency and more ways to build data literacy among stakeholders” (Bhargava et al. 2016, p. 197,198).

Given that “algorithms are critical enablers of the data revolutions that is taking place”, data literacy approaches can be seen to imply a degree of algorithmic literacy as well (WWWF, 2017, p. 4). D’Ignazio & Bhargava writes that data literacy thus includes “[u]nderstanding the algorithmic manipulations performed on large sets of data to identify patterns” (2015, p. 3).

Lastly, it can also be argued that increasingly popular code and computational literacy, or computational thinking, initiatives can foster an algorithmic literacy in students since these pedagogical curriculums generally covers subjects such as algorithmic expressions, algorithm design, as well as executing algorithms through computer programming (Chester, 2016; ECDL Foundation, 2015; Vee, 2017). While many may argue, such as Osoba and Welser IV, that such technical competency does not necessarily equate the much-needed public algorithmic literacy, it can assuredly be instrumental to it. Vee writes that in a context where computers have become ubiquitous in society one of the “most dominant current motivation for coding literacy is that of individual empowerment” as “the ability to program [computers] gives a person access to more avenues of control” (2017, p. 77). An example of a computational thinking initiatives aimed towards empowerment can be seen in the ‘Black Girls Code’ project that claims to “give underprivileged girls a chance to become the masters of their technological worlds” (cited in Vee, 2017, p. 78). Yet though these initiatives are increasingly oriented towards the public user instead of necessarily the producers, its area of interests is predominantly technical.

These last two areas of action are focused in different degrees on an engagement with knowledge gaps as interest, and users as stakeholders. In this they differ from the first two examples mapped and display an important distinction in terms of thinking about algorithmic literacy. Raine & Anderson argue that the obscuring and blackboxing of algorithms essentially mean that “they are not evident

⁶ The ‘Data Mural’ team implemented a workshop grounded on Paulo Freire’s critical pedagogy approach, building on the concept of generative themes in Plug Minas school in Belo Horizonte, Brazil. The approach was to build “participatory and impactful data literacy using a set of visual art activities” in order to increase the participants’ critical relation with data in their everyday lives (Bhargava et. al., 2016. p. 197).

in user interfaces and their code is usually not made public” which means that “most people who use them daily are in the dark about how they work and why they can be a threat” (2017, p. 74). In data literacy and computational literacy initiatives, there is thus a more direct possible contribution to a public algorithmic literacy as they engage the users of algorithmic technology. In addition, the WWF report argues that “algorithmic literacy efforts would need to be context and industry-specific” if they are to be effective (2017, p. 13). Thus, if there is a knowledge gap where users are in need of algorithmic literacy, a necessary question to ask in developing a successful algorithmic literacy initiative is thus: how, where, and in which contexts does the general public engage with algorithmic technology and their user-interfaces?

4. A Source of Critical Perspective on Algorithms

Starting in the early 2000s, Lev Manovich’s *The Language of New Media* (2001) engendered the field of study formally known as software studies (Truscello, 2003). Truscello writes on the emergence of software studies that this field followed a digital materialist turn which was the outcome of importing “computer science into cultural studies” (2003, n.p.). Consequently, by focussing on the ontological effects of the computer in society Manovich argued that the influence worked in both ways: software is a cultural product inasmuch as it is used to create culture. As opposition to technological determinism, which presupposes that digital phenomena exist independently from its materiality (van den Boomen, Lammes, Lehmann, Raessens and Schäfer, 2009), software studies consequently investigate the functional and material components of digital technology from a social and cultural perspective. As Kitchin wrote during the time he was himself involved in formalizing the field of software studies: “we know very little about the ways in which software is socially created; the nature of software itself; how discourse, practices, and knowledge get translated into algorithms and code” (2011, p. 946).

In later writing focused on algorithms, Kitchin outlines multiple ways in which algorithms can be studied and argues that researchers can look at algorithms “technically, computationally, mathematically, politically, culturally, economically, contextually, materially, philosophically, ethically and so on” (2017, p. 16). There is thus no singular approach to studying algorithms, as for the author algorithms are

“contingent, ontogenetic, performative in nature and embedded in wider socio-technical assemblages” and thus “are not formulated or do not work in isolation, but form part of a technological stack that includes infrastructure/hardware, code platforms, data and interfaces, and are framed and conditions by forms of knowledge, legalities, governmentalities, institutions, marketplaces, finance” (ibid., p. 16, 25). Yet coming from software studies himself, Kitchin argues that as an approach to studying algorithms in society a “code/software studies’ perspective” is significant in that it allows researchers to study “the politics and power embedded in algorithms, their framing within a wider socio-technical assemblage and how they reshape particular domains” (ibid., p. 20). As he writes: “[algorithms] shape how we understand the world and they do work in and make the world through their execution as software, with profound consequences” for software is “fundamentally composed of algorithms” and thus constitute “algorithmic machines” (2017, pp. 14, 18).

In the same vein, writing on algorithmic culture Paul Dourish argues that the social significance of formalistically abstract algorithms only become clearer when we look at related phenomena that “emphasize different aspects of the sociotechnical assembly” (2016, p. 3). It is thus natural that one approach he cites, exemplified by software-studies’ Adrian Mackenzie, is to consider the significance of algorithms in their materialization “not just in a piece of code or even in a larger software system but in a specific instantiation—as a running system, running in a particular place, on a particular computer, connected to a particular network, with a particular hardware configuration” (ibid., p. 5). For it is such material socio-technical manifestation that “critically shape the effect that the algorithm has” (ibid.). Thus, from a software studies perspective, algorithmic culture can be seen to find socio-technical expression in the material software constituting what Lev Manovich termed ‘software culture’ (2013).

Predicating discussions of ‘algorithmic culture’, Lev Manovich argues that between the 1960s and 2010s a profound “softwarization” of culture took place (2013, p. 5). In his writing Manovich offers a useful macro-categorization of different software as either grey or cultural. For the author the former “runs all systems and processes in contemporary society”, exemplified by logistical or industrial automation software, while cultural software finds itself “in the center of the global economy, culture, social life, and, increasingly, politics,” exemplified by products such as

Google's Gmail, Microsoft Word, and Facebook (2013, p. 7). Furthermore, Manovich explains that cultural software is "cultural in a sense that it is directly used by hundreds of millions of people and that it carries 'atoms' of culture" (2013, p. 7). Following this argument, one can thus look to cultural software as one of the sites in which the public engage most ostensibly with algorithmic systems through their user-interfaces. A potential contribution to a public algorithmic literacy can thus be made through not only opening the black box of these cultural 'algorithmic machines', but by investigating algorithmic technology as materially contextualized within cultural software.

This follows the presupposition of Osoba and Welser IV (2017) and Caplan & Reed (2016) that an algorithmic literacy should importantly involve not only technical skills but also a critical stance towards the technology that enables it, namely software. Halavais points out that these efforts have tended to focus on code, stating that there is a need of an "educational effort" which "doesn't just teach kids to 'code,' but to think critically about how social and technological structures shape social change and opportunity" (cited in Raine & Anderson, 2017, p. 74). In the same vein, Steiner argues that the solutions focussing on teaching people how to code imply that "knowing how to produce algorithms protects oneself from their diverse and pernicious effects across multiple domains" (cited in Kitchin, 2017, p. 19). Instead he argues that "there is a need to focus more critical attention on the production, deployment and effects of algorithms in order to understand and contest the various ways that they can overtly and covertly shape life chances" (Steiner cited in Kitchin, 2017, p. 19). This view is supported by Vee, who suggests that "[a]pproaching software with the theoretical tools of the humanities and social sciences is a central project of the new field of software studies" (2014, p. 24). Thus, making her own contribution through an investigation of the roles coding literacy plays in society, the author argues that "the ubiquity and power of software demands that we use diverse theoretical tools to unravel what it means to live in an algorithmic culture" (Vee 2014, p. 42). This correlates with the needs identified for an algorithmic literacy, where as it is oriented towards the public in their consciousness for everyday encounters with algorithms, it is not contingent on technical competencies.

While assuredly, focusing on cultural software as the context of user engagement with algorithmic systems is consequently not the only way to contribute to their algorithmic literacy, we argue that it is nonetheless a significant one given the

role cultural software as ‘algorithmic machines’ have come to play in our lives. And it is thus that we argue insights from software studies can be useful as a critical perspective on cultural software in the formulization of a public algorithmic literacy initiative.

5. Research Question

Given the arguments made in this chapter, we contend that algorithmic literacy as a necessarily critical engagement with algorithmic systems can benefit from a software studies perspective as it is through software that algorithms find material expression as socio-technical assemblages, and through which users consequently engage with algorithms in everyday life. Software studies thus provide theoretical tools to understand not only the role algorithms play in culture, but also how they are black boxed by the user-interfaces of cultural software. Thus, we ask:

RQ: How can software studies contribute to algorithmic literacy?

RQ1: How does software studies understand the role algorithms play in culture?

RQ2: How can insights from software studies be implemented in an algorithmic literacy initiative?

CHAPTER 2

Methodology and Method

1. Methodology

1.1. Black Box Discussion of Algorithms

In his writing, critical theorist Jürgen Habermas addressed the rise of technocratic consciousness, outlining how the increasing redefinition of practical problems in society as mere technical concerns has had a “disintegrative effect on the public sphere” (Held, 1980, p. 254). Following this logic it can be argued that the current state of so-called algorithmic culture increasingly subordinates the control of society and culture to software, with algorithmic logic dictating cultural value-system. For as Habermas noted, the level of rationalization of technocratic agendas has created a “negative utopia of technical control over history” (ibid., p. 254). Habermas’ critique shows that ideologically science and technology reduce “practical questions about the good life to technical problems for experts” through which “contemporary elites eliminate the need for public, democratic discussion of values, thereby depoliticizing the population” (Bohman & Rehg, 2017, n.p.). To put it differently, by adding a technical layer to the public sphere such a techno-positivist worldview excludes non-technicians from debates about technology, its design, its effects, and its further development—while at the same time those technologies are increasingly being deployed as means of governance and cultural production (Pasquale, 2015; O’Neil, 2017; Striplas, 2015; Gillespie, 2014). This, according to Habermas, not only justifies a “particular class interest in domination, but also affects the very structure of human interests” (Held, 1980, p. 254).

This context raises concerns because authority in society becomes “increasingly expressed algorithmically,” and decisions which used to be “based on human reflection are now made automatically” (Pasquale, 2015, p. 8). Berry, van Dartel, Dieter, Kasprzak, Muller, O’Reilly, and de Vicente outlined the dangers in terms of transparency of the technological processes, which are currently completely in the dark. They argue that “as the digital increasingly structures the contemporary world, curiously, it also withdraws; it becomes harder and harder for us to focus

upon, as it becomes embedded, hidden, off-shored, or merely forgotten about” (Berry et al., 2012, p. 44).

Likewise, Vilém Flusser's (1983) ideas of freedom in a post-industrial context deals with very similar subjects as aforementioned. His concerns are centred in the public's uncritical attitude towards the 'apparatus' which he defines as “products of applied scientific texts” (1983, p. 14). This term, in the context of our thesis, can also be applied to digital devices such as smartphones and laptops that in their reliance on software can function to be described as 'algorithm machines' (Kitchin, 2017, p. 14). Moreover, we argue that Flusser's 'uncritical attitude' correlates to the 'critical stance' criteria reviewed on our literature review, since both terms aim to counteract the over reliance of technical apparatuses by empowering users with knowledge of their socio-technical dimensions. As Flusser explains “human decisions are now being made on the basis of apparatus decisions,” with the technological objectivity of apparatuses being fundamentally taken for granted (1983, p. 33).

Flusser also points out that apparatus cannot have an owner because there is an industry that programs it for the “sake of the industrial complex, which in turn functions for the sake of the socio-economic complex” (ibid., p. 12). It is thus “not he who owns the hard objects, but who controls the software, who in the end holds the value” (ibid.). The same could be argued in terms of computer software⁷, where the user gets access to a product, but the control relies on the owner through updates, and the proprietary nature of source-code. Thus, for Flusser the area of action is “in the realm of the automated, programmed and programming apparatus” (ibid., 37). Given the premise of this research project, it is perhaps necessary to add that Flusser held that despite the nature of the 'apparatus' “it is possible to create room for freedom” (ibid., p. 36).

While Flusser wrote on creating a critical theory of the technology of photography, Lev Manovich similarly suggests expanding the understanding of the apparatus of the computational from a “distribution and exhibition” tool to one of that produces media and acts as a “media storage device” (2001, p. 43). For as Truscello writes, Manovich poses that “[s]oftware studies must not only investigate the ways in

⁷ As Manovich writes in *Software Takes Command*: “In the beginning of the 1990s, the most famous global brands were the companies that were in the business of producing materials or goods, or processing physical matter. Today, however, the lists of best-recognized global brands are topped with the names such as Google, Facebook, and Microsoft” producing not consumer electronics but rather software (2013, pp. 6-7).

which the computer's ontology shapes culture, it must also analyse the culture that shapes computer programming” (2003, n.p.). This entails a transdisciplinary gaze which can support both the technical and cultural dimensions of software, for as Kitchen and Dodge argue: software studies do not focus only on the technical aspects, but instead

fuses the technical with the philosophical to raise questions about what software is, how it comes to be, its technicity, how it does work in the world, how the world does work on it, why it makes a difference to everyday life, the ethics of its work, and its supporting discourses. Software studies then tries to prise open the black boxes of algorithms, executable files, database structures, and information protocols to understand software as a new media that augments and automates society. (2011, p. 246)

Thus, it becomes clear that developing a critical stance towards digital technologies implies a critical understanding of both the technical and social reality of software and algorithmic technologies. This understanding offers a counter-discourse to how technologies are perceived to function seamlessly in Habermasian technocracy. As Gillespie, Boczkowski, and Foot emphasize, it becomes necessary to shift to a perspective which “requires a concerted effort to look beneath the technology at the human underbelly of the sociotechnical system” (ibid., p. 13).

1.2. Literacy as Competences

In the same way as technological discourse can easily preclude any non-technical perspectives, anything termed ‘algorithmic literacy’ can easily be argued to be the purview of computer science pedagogy. Within a perceptible ‘software culture’, traditional conceptions of literacy as a demonstrable skill-set for ‘reading and writing’ can thus be seen to extend to computer science competencies, which include reading and writing computer code as algorithmic expressions (UNESCO 2005, p. 149; Vee 2017)⁸. Yet given our preliminary definition of algorithmic literacy in the previous chapter, we argue that it rather fits within a paradigm of literacy as constituting socio-cultural competencies.

This definition arose from the conception of literacy understood as the ability to not only read written texts, but also the “‘non-print’ bits” of culture “like values and gestures, context and meaning, actions and objects, talk and interaction, tools and

⁸ Given the importance software had come to play in contemporary societies, the conception of ‘reading and writing’ skills for the 21st century are argued by some to include the ability to write computer code and create software (Vee, 2017, p.22).

spaces” (Lankshear & Knobel, 2003, p. 8). This shift gave rise to a plethora of ‘new literacies’ such as ‘visual literacy’, ‘environmental literacy’, ‘media literacy’, and more as literacy increasingly became understood as connoting “the idea of being able to find one’s way around some kind of system, and to ‘know its language’ well enough to be able to make sense of it” (ibid., p. 15). This conception fundamentally reflects a “broader, metaphorical” extension of literacy to include skills for “social awareness and critical reflection” (UNESCO 2005, pp. 147,150). For example, information literacy can refer to the technical ability to “access and use a variety of information sources” but “can also be defined as the development of a complex set of critical skills that allow people to express, explore, question, communicate and understand the flow of ideas among individuals and groups in quickly changing technological environments” (ibid., p. 150).

While such an approach applied to ‘algorithmic literacy’ does not necessarily negate or exclude a technical computer science pedagogical approach, it does release it from the sole prerogative of computer science. Following, working against the technocratic consciousness criticised by Habermas opens room for alternative or complementary conceptions of an algorithmic literacy from a critical cultural studies perspective. It is from this position that we orient our own research, without the presumption that our approach will constitute a full algorithmic literacy theory or program. Rather, we aim to contribute to the conception and operationalization of algorithmic literacy from a Humanities disciplinary point of view that emphasize the necessary inclusion of socio-cultural competencies that can constitute a critical stance towards algorithmic technology, reflecting the approach of software studies to computer science as well.

1.3. Critical Theory

Given the epistemological orientation discussed above, this research project proceeds from a critical theory approach. Kincheloe & McLaren explain that even though none of the critical theorists that composed the original Frankfurt school of critical theory “ever claimed to have developed a unified approach to cultural criticism,” its ontology can be understood as a “dialectical concern with the social construction of experience” (2002, pp. 87-88). The Frankfurt school members viewed “their disciplines as manifestations of the discourse and power relations of the historical context that produce them” (ibid.). Accordingly, Bronner explains that this

approach is influenced by Marxism, especially in terms of “the ensemble of social relations’, at the center of historical materialism” with its objective to “understand a fact within the value-laden context wherein it assumes meaning” (2011, p. 25).

Critical theory however, proposed a different variety of Marxism wherein the critical method is highlighted and its concerns are centred on the processes of alienation through and reification of hegemonic ideologies. Therefore, as an object of study, cultural objects become spaces of power struggles within critical theory. This approach describes how by focusing on everyday products and objects, one can “illuminate the character of society and [its] cultural trends” (Bronner, 2011, p. 26). By doing so, critical theorists focus their studies on how forms of dominations mutate under capitalism. Moreover, Bronner states that “critical theory was intended as a general theory of society fueled by the desire for liberation” and it is between that objective of liberation that the “substance of emancipation” defined the character of its critical method (2011, p. 24).

Consequently, the research within critical theory is characterized by its purpose of inquiry, which aims to understand the relationship between social structures and ideological patterns of thought in order to confront and change unjust social systems (Schofield, 2014). It has also been argued that this approach provides a “new poststructuralist conceptualization of human agency” (Kincheloe & McLaren 2002, p. 89). Put differently, the research done within critical theory aims to explain social change, as it strives for a transformative outcome (Schofield, 2014; Mertens, 2015). Accordingly, as critical theory “explicitly addresses issues of power and justice” (Mertens, 2015, p. 23) its transformative aim is connected to the concerns of the Frankfurt school where they “agreed on the need for increased education to counteract authoritarian trends” (Bronner, 2011, p. 5). Given our research questions, within this theoretical orientation we thus need to approach our research in two ways, namely: understanding algorithms in their material relation to social structures and ideology; and framing this critical understanding within a literacy model oriented towards transformative social emancipation.

1.3.1. From Critical Theory to a Software Critique

The rise of ‘cultural computers’, through the lenses of critical theory, entails understanding of the cultural industry as an instrument of control of capitalism as

Giroux (2005) explains that critical theory as a methodology enables the unveiling of hidden systems of technological and media oligopolies. Horkheimer and Adorno described that culture “is infecting everything with sameness. Film radio, and magazine from a system. Each branch of culture is unanimous within itself and all are unanimous together” (2002, p. 94). The authors posit that culture produced under a monopoly becomes identical, and these standardized forms are claimed as “derived from the needs of the consumers: that is why are accepted with so little resistance” (ibid., p. 95). Moreover, Jameson, while introducing new elements from postmodernism to the Frankfurt school’s late capitalism (Franco, 2015), explains that this includes “new forms of business organization (multinationals, transnationals) beyond the monopoly stage but, above all, the vision of a world capitalist system fundamentally distinct from the older imperialism” (1991, pp. xviii-xix). This entails that this kind of businesses are expanding globally and that they “are not tied to any one country but represent a form of power and influence greater than any one nation” (cited in Franco, 2015, p. 613). Jameson also states that among the characteristics of the logic of late capitalism includes “the new international division of labor, a vertiginous new dynamic in international banking and the stock exchanges [...] and new forms of media interrelationship [...], computers and automation” (1991, p. xix).

Furthermore, Jameson considers that “[p]ostmodernism is what you have when the modernization process is complete and nature is gone for good. It is a more fully human world than the older one, but one in which ‘culture’ has become a veritable ‘second nature’” (1991, p. ix). Therefore, if this reasoning is applied to the development of computation, it could be argued that it is concomitant to the digitalization of government, economics, and culture, the ‘realization’ of the ‘techno-positivist ideal’ of potentially translate every ‘bit’ of the natural and social world into digits. As Berry et. al. points out: “the software that is now widely used is part of a wider constellation of software ecologies made possible by a plethora of computational devices that facilitate the colonization of code into the life world” (2012, p. 49). It is within this ‘colonization’ that algorithms—materialized in software— “codify through their outputs a specific, increasingly ubiquitous texture of reality, a skin that’s being overlaid in buildings, fashion, cars, jewelry, print publications, and chairs” (ibid., p. 14). On this matter, Kittler argues that “physical hardware with the algorithms forged for its computation, has finally gotten rid of

hardware itself. As a result, software has successfully occupied the empty place and profited from its obscurity” (1997, p. 151). For Kittler, this entails that the algorithm is hidden from its outputs. Furthermore, he posits that “because software does not exist as a machine-independent faculty, software as a commercial or American medium insists on its status as property all the more” (ibid.).

This understanding of software as a black box of hardware and algorithms can be understood in terms of critical theory as attributable to the economic system and copyright revenues from which the techno-oligopolies benefit. Matthew Fuller points out that “software’s legal reordering as a separate kind of entity, [...] became a commodity, an entity the prime or sole motive for the production of which is to generate a monetary profit for those who own the entities” (2008, p. 3). Thus, it can be argued that the cultural computer has become an object of commodification where the designing of both the hardware and software that compose the apparatus have been increasingly dominated by few corporations worldwide, often referred to as ‘Big Tech’ or GAFA⁹ (Foer, 2017; Moore & Tambini, 2018). This has intensified the cultural hegemony of highly technological developed countries and have accentuated the gap between producers and consumers. Nevertheless, as Manovich points out: “software as a theoretical category is still invisible to most academics, artists, and cultural professionals interested in IT and its cultural and social effects” (2013, p. 9).

As critical theory informs our basic assumptions that we as researchers have “about the nature of the field” we aim to investigate (Gustavsson, 2007, p.3), it likewise shapes our understanding of algorithmic technology as socially construct software, as proposed in the academic work of software studies (Kitchin & Dodge, 2011; Kitchin, 2017; Goffey, 2008; Manovich 2011). Although software studies and critical theory are not formally related, software studies can clearly be located in the traditions of critique established by critical theory. As David M. Berry, one of the early writers of software studies, states:

The digital is simultaneously technical and social, material and symbolic, but it is also a historically located concept, as are its instantiations in concrete computational devices. This is not a new problem, but it does make the digital challenging to investigate and, I believe, it puts critical theory in a unique position to problematize. (2011, p. 50)

⁹ Acronym that stands for Google, Apple, Facebook, Amazon.

It is not incidental that Berry as software studies proponent writes extensively about software from a critical theory perspective, as demonstrated in his book *Critical Theory and the Digital* (2014). While in their reading list for what they call ‘critical algorithmic studies’ Gillespie and Seaver categorize software studies texts such as those by Fuller and Goffey, or Manovich’s pivotal software studies text *Software Takes Command* (2013), under the “critical theory approach” as they offer a perspective on how “algorithms fit with, and help advance, specific ideological worldviews” (2016, n.p.).

Like critical theory, software studies has been argued to draw on Marxian historical materialism (Kitchin & Dodge, 2011, p. 246; Truscello, 2003, n.p.) in its proposition of a ‘digital materialism’ methodology (Casemajor, 2015; van der Bomen et al., 2008). As Casemajor writes, ‘digital materialism’, as a ‘field of interest’ started gaining momentum in the 2000s as “[m]aterialism as a theoretical paradigm assumes that all things in the world, including things of the mind and digital stuff, are tied to (and in some cases, determined by) physical processes and matter” (2015, pp. 4–5). This focus on software can be seen to respond to what Jameson explained as being Marx’s call “for a history of technology, a materialist history” when he remarked at several points: ‘a critical history of technology [...] does not exist’ (2011, p. 55). For Jameson, the call has been associated traditionally with the era of nineteenth-century industrialization whereas the “heavy industry today itself [is] displaced by cybernetics and information technology” (2011, p. 56). The change of focus on the ‘technological design’ —that corresponds to the methodology of critical theory and Marxism—helps to inform our research question.

Manovich writes that “Software Studies has to investigate the role of software in contemporary culture and the cultural and social forces that are shaping the development of software itself” (2013, p. 10). This approach balances the discussion of digital technologies and serves as a counterargument to techno-positivist statements where computer science is perceived “as a kind of absolute truth, a given which can explain to us how culture works in software society” when indeed “computer science is itself part of culture” (ibid.). By focusing on the software, a new version of historical materialism is being applied to understand the technological landscape and the action of *blackboxing* algorithms through software interfaces. This approach allows us to unpack the black box of cultural software.

The theoretical work done by software studies thus enables us to arrive at a critical conception of algorithmic technology, due to the fact that it provides a “set of ideas for thinking about software and its relations—how it is constitutive of, situated in, and does work in, the world” (Kitchin & Dodge, 2011, p. 245). Finally, as Berry proposes, with software studies “[t]he challenge [...] [is] to bring software back into visibility so that we can pay attention to both what it is (ontology), where it has come from (through media archeology and genealogy) but also what it is doing” (2011, p. 17). Thus, software studies as theoretical framework investigates software as culture, and through its digital materialist approach affords an understanding of how algorithms not only function in the world but are *blackboxed* by the socio-technical context of software productions. We thus turn to software studies as a field of research that can be situated within a critical theory approach in order to answer our inquiry.

1.3.2. A Literacy Model for Critical Theory

The term ‘critical’ has not only “become one of the most complex words we deploy in scholarship,” but is further complicated by its myriad of denotative meanings (Harvey, 2018, p. 36). It is thus important and necessary to distinguish what we mean by ‘critical’ – especially as ‘critical thinking’ has a fundamentally different meaning when put in the context of pedagogy as opposed to how we use ‘critical’ in relation to algorithmic literacy. Collin Griffin writes that ‘critical thinking’ as a concept “is derived directly from such influential figures as Rogers and Maslow, and familiar psychological constructs of personal growth, authenticity, self-actualisation, self-direction, peak experiences and so on” (1988, n.p.). Moreover, Griffin writes that “although critical thinking and perspective transformation through adult learning have been attributed to critical social theory in reality this is only a new manifestation of personal growth psychology” (ibid.) Contrastingly, we are employing ‘critical’ as used in ‘critical theory’ – where thinking critically is a reflection on the “relation between individual social transformation, personal and political emancipation” (ibid.). This definition informs how we will answer our research question, as we read the ‘critical stance’ within our algorithmic literacy criteria to be congruous with how ‘critical’ is conceptualized within critical theory.

Given that our research concerns not only generating a theoretically critical stance towards technology, but responding to the need for a public algorithmic literacy, it is important to place our research within an explicitly pedagogical frame that fits within a critical theory approach¹⁰. It is consequently that we draw on the literacy promotion model of Paulo Freire, the ‘father’ of critical pedagogy, within the well-established reading of “critical pedagogy as a form of critical theory” (Morrow & Torres, 2002, p. 4).

Following Antonio Gramsci conception of literacy as being fundamentally ideological and that it “may have less to do with the task of teaching people how to read and write than with producing and legitimating oppressive and exploitative social relations” (Giroux, 2005, pp. 1,2), Freire identified what he called the traditional ‘banking model’ of education. In such a model “the teacher issues communiques” instead of communicating, and effectively re-produces and legitimizes ruling-class ideology within the context in which the pedagogical exchange occurs (Freire, 2005, p. 72). For Freire, a critical “dialogical or problem-posing education” conversely allow for the humanization of learners through what he called conscientization (*conscientização*): “The awakening of critical consciousness [that] leads the way to the expression of social discontents” (Freire & Macedo, 2005, p. 36, Freire 2005, p. 40). In Freire’s practiced four-stage literacy model, learning to read thus becomes not a question of technical competencies but rather of social empowerment. For such a critical literacy “always involves critical perception, interpretation, and rewriting of what is read”, which leads to a “a critical reading of reality” (Freire & Macedo, 2005, pp. 23,24). Importantly, it is the prominence of Freire’s work that contributed to the historic expansion of literacy to include socio-cultural competencies discussed at the outset of this chapter (UNESCO 2005Lankshear & Knobel, 2003).

However, despite Freire’s demonstrative Marxist orientation, there is no formal connection between his transformative pedagogical model and critical theory (Morrow & Torres, 2002, p. 5). Nonetheless, as Morrows and Torres writes in their comprehensive *Reading Freire and Habermas: Critical Pedagogy and Transformative Social Change* (2002), it is especially with Frankfurt school’s Jürgen Habermas and his theory of ‘communicative action’ framed by a discussion of the

¹⁰ Especially given the justifiable critique of literacy promotion historically being aligned with the vested interests of industry and capital (Vee, 2017, p. 55, Levi-Strauss, 1962, p. 293).

public sphere that Freire shares implicit similarities (ibid., pp. 5,16). Drawing on Morrow and Torres' work it is thus in reading "Freire through Habermas and Habermas through Freire" that one can "avoid falling into the trap of evaluating [Freire's] work in isolation as a comprehensive theory of society" while simultaneously "fleshing out the more abstract notion of [Habermas'] theory of communicative action in relation to a [Freire's] theory of dialogue and practical pedagogy" (ibid., p. 14).

Such a dialectic between these two authors works well within the guiding principle of critical theory, framed within a transformative research approach, of not simply aiming to "describe a situation from a particular vantage point or set of values" but to "[try] to change the situation" itself (Cohen & Crabtree, 2006, n.p.). As Morrow and Torres writes: "Reference to the methodology of critical theory (here including Freire) is closely connected to slogans invoking the 'unity of theory and practice'" (2002, p. 53). It is thus drawing on Freire's model for critical literacy promotion, read in relation to critical theory, that we can approach our research question and the problematic addressed in it practically as well as academically¹¹. Moreover, this approach will also allow us to address a transversal problem that naturally circumscribes our research as disciplinarily situated in the humanities: Humanities is often seen as a discrete domain with little demonstrated social relevance (Harpham, 2005, p.22). As Gretchen Busl from the Texas Woman's University writes in her pithily titled article *Humanities research is groundbreaking, life-changing... and ignored* for *The Guardian*:

The inward-focused nature of scholarship has left the public with no choice but to respond to our work with indifference and even disdain, because we have made little effort to demonstrate what purpose our work may have beyond the lecture hall or academic journal (2015 October 19, n.p.).

¹¹ It is not incidental that many data literacy initiatives, which we regard as auxiliary or interconnected to what could be considered algorithmic literacy, likewise draw on Freire in their conception and operationalization of data literacy (Tygel & Kirsch, 2015; Bhargava & D'Ignazio, 2015).

2. Method

2.1. Practice-led research

Morrow and Torres write that both Habermas and Freire emphasized the need for methodologies that “gives priority to knowledge that potentially can be appropriated by subordinated agents to enhance their critical consciousness,” highlighting “the ideal of linking theory and practice that also recognizes the strategic importance of other forms of knowledge” (2002, p. 64). Given this, although our research will predominantly draw on the theoretical work of software studies, we propose orienting our research to the transformative concerns of critical theory and critical pedagogy. Therefore, we have chosen a practice-led methodology.

Like critical theory, practice-led research is “concerned with the relationship between theory and practice,” though with an emphasis on the significance of creative work as a “form of research [that] generates detectable research outputs” (Smith & Dean, 2009, pp. 4-5). Research using this method is “interwoven in an iterative cyclic web” of practice and research, consisting of “numerous points of entry, exit, cross-referencing and cross-transit within the practice-research cycle” (ibid., pp. 2, 8). This process of iteration is central to the methodology, allowing a researcher to move “spider-like” within the practice-led research’s cyclical model (see Fig. 2.1.) instead of constructing the research along a teleological structure (ibid., p. 19).

2.2. Selecting the Creative Practice Design Space

Within a practice-led research method, we needed to first identify the creative practice that will function as the design space for our research. To do this we proceeded by benchmarking examples of practical project that could be evaluated according to criteria derived from the already discussed call for algorithmic literacy (as well as our theoretical and methodological concerns). These ad hoc criteria can be summarized as:

1. The project format should be able to facilitate a public education program that can fit within a critical pedagogy model.
2. The project should be contextualizable to the concrete experience of the participants with algorithmic technologies in the form of cultural software.

3. The content of the project should not be contingent on technical competencies.
4. The medium of the project should afford space for a critical stance towards software.

Given these points, we reviewed numerous projects that could answer some of these criteria and provide a practical frame for our research. Among the examples explored were: *Auto-Illustrator* (2001), a software artwork by Adrian Ward that draws attention to the algorithmic affordances in design applications¹²; *Textbook* (2017) by Benjamin Grosser, a web-browser extension designed as critique of Facebook's user-interface by removing all images from the platform; the already discussed 'Data Murals' (2016) workshop from the MIT Center for Civic Media by Bhargava, et al.; Mozilla's web literacy platform (2018), which includes open-source educational tools for remixing webpages' HTML code, amongst others; *Bad News* (2018) by DROG research group from the University of Cambridge, a web-based game designed to foster awareness of how online platforms propagate fake news; and *The Altar of the Algorithm* (2017) by Noothout, a critical design project done at the Eindhoven University of Technology which investigates the ideological dimensions of software algorithms.

In addition to the above, we also arrived at examples of games such as: *Keep Cool* (2017), a board game by Gerhard Petschel-Held and Klaus Eisenack¹³ in which players interact with the complex socio-economic and political factors of climate change; *I'm an Agricultural Worker, Get Me Out of Here (IAAWGMOOH)* (2014), a board game designed by TerrorBull Games as a commission from The School of African & Oriental Studies (SOAS) at University of London to help students understand the complex relationship between fair-trade initiatives and its effects on agricultural workers; and *The Landlord's Game* (1904) by Elizabeth Magie, a didactic precursor to *Monopoly* (2018) in which players negotiate the adverse effects of US property laws in the early 20th century. Though the perceptible lack of software critique in these examples, evaluating such games and their creative affordances allowed us to identify an ideal design space for our project, namely: critical board game design.

¹² This project was awarded an honorary mention at the 2001 Prix Ars Electronica.

¹³ Made with the support of German Federal Ministry for The Environment, Nature Conservations and Nuclear Safety (BMU), Global Climate Forum (GCF), and Potsdam Institute for Climate Impact Research (PIK) among other institutions.

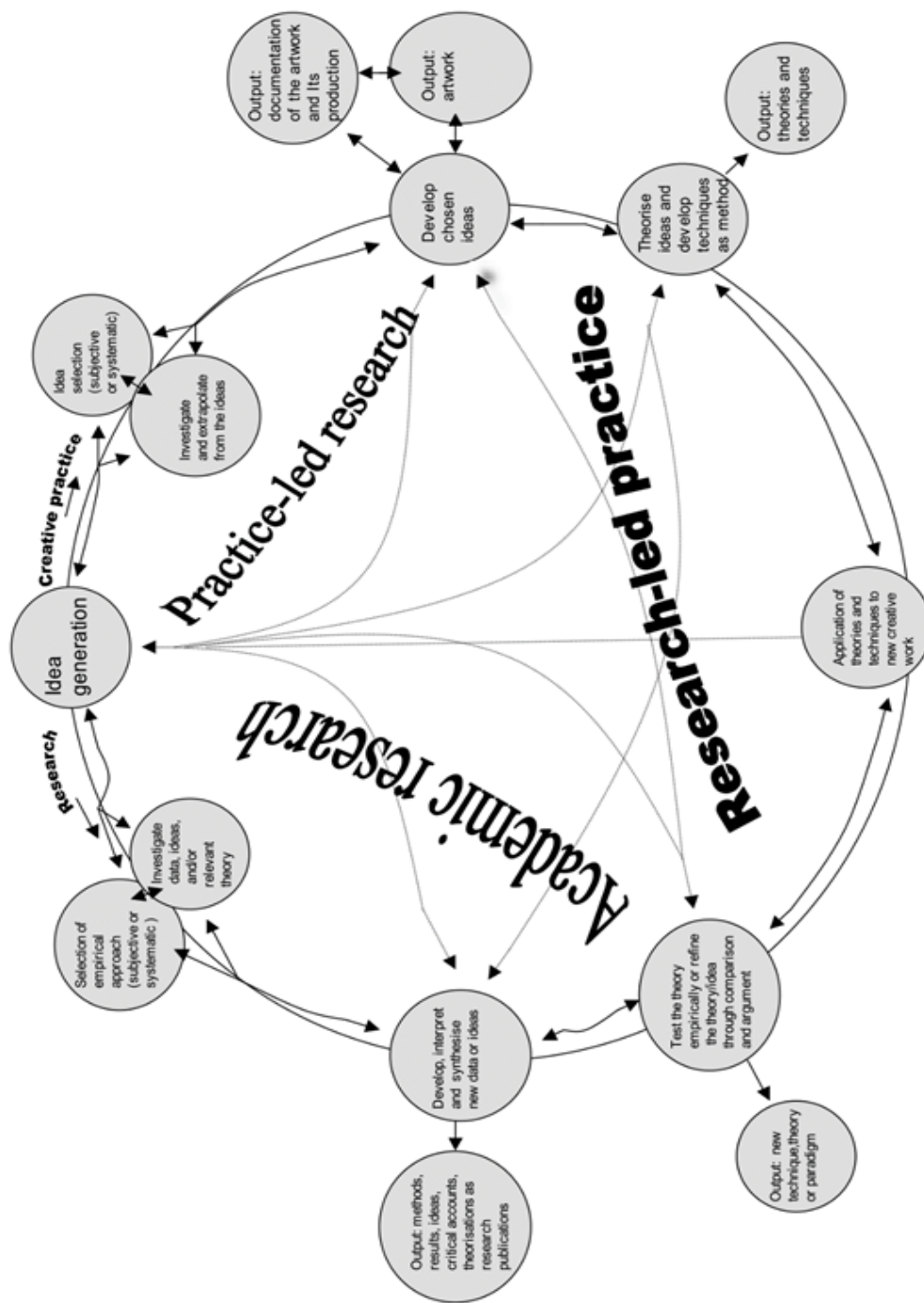


Figure 2.1. The 'iterative cyclical web' practice-research model: Researchers should move through the model 'spider-like' according to Smith and Dean. They describe it's flow as follows: "The outer circle of the diagram consists of various stages in the cycle of practice-led research and research-led practice, and the smaller circles indicate the way in which any stage in the process involves iteration. The right-hand side of the circle is more concerned with practice-led research, the left-hand side with research-led practice, and it is possible to traverse the cycle clockwise or anti-clockwise as well as to pass transversely" (2009, p. 19). Reprinted from *Practice-led Research, Research-led Practice in the Creative Arts: Research Methods for the Arts and Humanities* (p. 19), by Smith, H., & Dean, R.T., 2009. Edinburgh: Edinburgh University Press.

2.3. Critical Board Games

In order to define ‘critical board game’ it is useful to look at its two interrelated components. Firstly, the ‘critical’ component draws on Mary Flanagan’s conception of ‘critical play’ (2009). In her writing Flanagan proposes a “theory of avant-garde game” (2009, p. 1) in which alternative or radical game design perspectives are deployed “for artistic, political, and social critique or intervention, in order to propose ways of understanding larger cultural issues as well as the games themselves” (ibid., p. 2). Flanagan argues that games, like any other media, “carry beliefs within their representation system and mechanics” (ibid., p. 4). This understanding of games entails that artists can use games “as a medium of expression” (ibid., p. 4), where the game elements and mechanics become tools for artistic creation. Furthermore, following Flanagan’s writing, Lindsay Grace posits that “[c]ritical games use critique as their game design premise. They have one goal—critical commentary through gameplay” (2014, p. 2). This reference to critical commentary and social critique in both Grace and Flanagan can be interpreted as being consistent within the understanding of ‘critical’ as used in critical theory.

Secondly, ‘board game’ can be fundamentally defined, according to game designer Greg Costikyan, as “a form of art in which participants, termed players, make decisions in order to manage resources through game tokens in the pursuit of a goal” (cited in Phillies & Vasel, 2006, p. 45). The popularization of board games as a medium can however be ascribed to “the mass printing techniques of the industrial revolution [which] enabled the huge variety of board games we know today” (Flanagan, 2009, p. 63). This led to two primary genres of board games emerging: American and European-style board games (Mayer & Harris, 2010, p. 3). The former involves more luck than strategy, whereas in European-style¹⁴ board games the gameplay relies less on luck than on a number of other shared characteristics such as information-rich environment, open-ended decision, and end-game scoring. Between critical play, and Euro-style board games, there are certain affordances that we created an ideal space for our research project. These will be discussed below.

The premise for selecting board game design as the practical component for our research project stems heavily from the perceptible trend starting in the 1980s to use tabletop games such as board games as pedagogical tools for computer and

¹⁴ Also known as ‘Eurogames’ or ‘designer games’.

code literacy, as well as computational thinking, curriculums¹⁵ (Bogost, 2005, p. 32). The effectiveness of these are perhaps best explained in terms of Ian Bogost's writing on procedural literacy, in which these games follow a classical "pedagogical structure of grammar, logic, and rhetoric" (ibid.). The appeal of games to teach coding stems from the fact that they constitute, like software itself, procedural systems dictated by logical "rules of interaction" that can thus employ classical pedagogical structures (ibid., p. 33). Yet importantly, "procedural literacy can be cultured not only through authorship, such as learning to program, but also through the consumption or enactment of procedural artifacts" such as games (ibid. p. 34).

Looking at the pedagogical potential of board games within this trend, Mayer & Harris argue that designer games have a "rich variety of game mechanics to provide skills practice and the opportunity to explore new ideas through thematic elements" (2009, p. 11). This, they argue, supports learning in four fundamental ways: authentic experiences, student engagement, social and life skills, and high-order thinking. Moreover, board games as pedagogical tools also offer a conduit for Freire's dialogical literacy model. Francesco Crocco argues that "the rich interactivity offered by games naturally reroutes learning away from the traditional 'banking model' of education that reduces students to passive recipients of institutionalized, rote knowledge" (ibid., p. 27). However, this is not to say such games inherently escape ideological scrutiny, as it is only "[w]hen the valuable learning principles embodied by games (digital or otherwise) are used to promote critical thinking about hegemonic ideas and institutions rather than to propagate them" that they can be called "critical gaming pedagogy" according to Crocco (ibid., p. 29). Hence, it can be inferred that critical gaming pedagogy can work within a Freirian approach, which can help foster algorithmic literacy. As such, this approach to game design naturally aligns with Flanagan's conception of critical play, as critical play games "also explore notions of agency and education" through making particular social or political issues relevant to players (2010, p. 3).

It is particularly in the creation of a space for social interaction between players that games create an ideal scenario not only for learning, but also for critical reflection. For Flanagan, games as cultural artefacts can be regarded as *playculture*, as they constitute a "site for production and consumption of culture,

¹⁵ See for example: *Code Monkey Island* (2014), *Code Cards* (2014), *Potato Pirates* (2017).

community, language, commerce, work, and leisure” (ibid., p. 254). By allowing players to interact and/or collaborate socially (Mayer & Harris, 2009), board games can also emulate Habermas’ conception of the public sphere—a necessary democratic space circumscribed by “the idea of inclusive critical discussion, free of social and economic pressures,” which analogously then creates a dialogical space for Freirian pedagogical conscientization to occur (Bohman & Rehg, 2017, n.p.). An example of how a board game might facilitate this can be seen in TerrorBull’s “global domination liberation” board game *War on Terror* (2016) (TerrorBull, 2015e, n.p.). This game is meant to prompt critical debate about geo- politics through letting players “starts with a fledgling empire and the best intentions” to ‘liberate’ the world (ibid.) before they are confronted with having to make morally compromising choices such as “funding terrorism” to stay competitive and win the game (Sheerin in TerrorBullGames, 2010 January 7, n.p.)¹⁶. *War on Terror* demonstrates another unique affordance of board games as procedural systems, which in that players need not have the technical diplomatic skills necessary for global politics to engage with it as a game system. Therefore, it can be argued that people who are not familiarized with the technicity of software and algorithms can access engage in an exploration of them through the abstracted tokens and information-rich play environment of a board game.

Equally important in this is the fact that games as a medium offer a space to convey a social critique, as understood within critical theory. This is closely related to the perception of games as procedural artefacts. For as Lindsay Grace argues, games can engender social critique as they might “remind us that our daily lives may be better understood as games [...] or that some political systems are in themselves absurdist games with no winners” through inverting “the relationship of games to life” (2014, p. 2). “[G]ames are ways of understanding the world”, writes Grace, and critical games consequently provide alternative ways to understand it (ibid., p. 4). Flanagan also points out that critical play can provide a “viewpoint or an analytical framework”, which can be achieved by “[creating] a platform of rules by which to examine a specific issue—rules that would be somehow relevant to the issue itself” (2009, p. 6).

¹⁶ Importantly, as its creators describe this game: “[I]t’s not like a didactic game, it’s not set up to say: look here’s our politics in the form of a board game - play it and listen. It’s more interactive” (Sheerin in TerrorBull Games, January 7, 2010, n.p.).

Huizinga famously argued that the playing of games occur in a ‘magic circle’, as *playculture* create spaces both separated from but relatable to reality (Flanagan, 2009, p. 9). Flanagan suggests that critical play is a subversive practice since it takes advantage of the magic circle as a space for experimentation. Following Negri’s definition of subversive practice, Flanagan notes that “when working against pervasive systems of power [...] subversive practices still have the power to trigger a social change when used on the right scale and with the right tools” (ibid., p. 11). For the author, critical games can thus become activist games. These types of games “can be characterized by their emphasis on social issues, education, and occasionally, intervention,” as they transcend conceptual exercises and “engage in social issues” (ibid., p. 13). In other words, activist games pursue a transformative outcome. This correlates with the transformative paradigm proposed by critical theory—not only in terms of social critique but also in providing a space for dialogue within the public sphere.

A board game that can be categorized as ‘activist game’ is *The Landlord’s Game* (1904) by Lizzie Magie. The game was used as a medium to support the ideas behind economist Henry George’s *Progress and Poverty* (1879), in which he postulated that land monopoly was the principal reason of poverty. Magie was inspired by George’s proposal for the Single Tax Movement as an alternative, where a “high, uniform tax be applied on all land, raw or developed” (Flanagan, 2009, p. 87). Magie converted these ideas into a board game that had two modes of play, one with the single tax solution and the other without it. The dual modal design allowed the game to be used as a subversive space to show the benefits of George’s proposal. In an ironic twist of fate, however, the second mode became the only mode of play after the patent of the game was acquired by Parker Brothers and rebranded as the board game known worldwide today as *Monopoly*¹⁷.

Flanagan also argues that critical game design offers an artistic mode of expression, as the idea of understanding “games as frameworks for thinking about culture” (Flanagan 2010, p. 1) came from artists such as Yoko Ono and Alberto Giacometti. Likewise, the artist Brenda Romero proposes that “games are capable of a higher form of communication, one which actively engages the participant and

¹⁷ It is not of not that Francesco Crocco develops his conception of ‘critical game pedagogy’ through reverting *Monopoly* back into a critical game to recaptures its original subversive critique of capitalism.

makes them a part of the experience rather than a passive observer” (2018, n.p). Moreover, Flanagan draws on Antonio Negri’s conception of subversion to explain that the critical play design is meant as a “creative act rather than a destructive act” (2009, p. 11). Thus, for the author, while artists introduce their interventions as “artistic objects into public spaces”, they also appropriate “the cognitive space of public space, of everyday space, and functions in an interventionist fashion” to effect social change (ibid.). This can be observed in her experimental board game *Train* (2009) in which the objective of the game is to load as many people, in the form of tokens, into the train’s wagons and transport them from one point to the other one. It has a roll-and-play mechanics combined with a deck of cards that players can draw in order to carry out the actions described in them. Nonetheless, within the gameplay it is revealed that Romero used the rules of the game as a medium to express the ruthless logistic system and operations behind the Jewish Holocaust, and explores questions of compliance with and complicity in this system through player participation in this game. In *Train* the rules of the game force players to mistreat their tokens in the quest to make their ‘trains’ reach its destination first, as the game mechanics “instantly become military orders” (Bogost, 2009 June 30, n.p).

Conclusively, as it can be shown in the examples already mentioned, board games can be used to express complex themes¹⁸. For Grace, the reflective characteristic of games such as these follow the theoretical approaches of Critical Design (Dune & Rabby, 2001), Critical Gameplay (Grace, 2012), as well as Bogost’s Procedural Rhetoric (2007). Thus, as “critical design takes as its subject the ways in which design reflects specific social characteristics”, games as design objects also “serves as gauge of social anxiety, bias and value” (Grace, 2014, p. 5). For Grace critical games “rely on reflection” and as any other form of social critique it needs a complex theme as subject (ibid.). The board game *IAAWGMOOH* by TerrorBull embodies this approach as their brief was to design a game that would allow students from SOAS “to get to grips with the complex relationship between fair trade initiatives and how these affect the actual workers in rural settings” (TerrorBull, 2015d, n.p). The game developers drew on a four-year academic study and through their game aimed to break the main components down so that the student could

¹⁸ Flanagan on the matter points out that critical play as an explicit game design approach is a relatively recent phenomenon and there is thus not a “great mass of games used in this way yet” (2010, p. 3).

understand how these socio-political and economic components interrelate. This particular affordance to express complex themes is one of the foremost reasons why we have chosen board games as our design space: they allow a theoretically developed software critique, as found in the works of software studies scholars, to be channeled into an accessible medium for public consumption. However, the challenge, as Flanagan states in this regard, is “to find ways to make compelling, complex play environments [...] to offer novel possibilities in games, and for a wide range of players” (2009, p. 6).

2.4. Design and Research Process

Just as game designers “follow an overall scheme of investigation or research” within the game design process (Flanagan, 2009, p. 252), within a practice-led research method the academic research we conduct will be guided by the same game design outcomes and processes. This offers a unique structure for academic investigation as “[g]ames and play environments are particularly useful frameworks for structuring systemic and conceptual concerns due to their multifaceted and dynamic, rule-based nature” (Flanagan 2010, p.3). Thus, following a critical play game design model as outlined by Flanagan (Fig. 2.2.), complemented by the work of Grace and Crocco, allows us to conduct our academic investigation as we systematically apply software studies theory to the design process. For this we will draw on key texts from software studies from authors such as Manovich, Berry, Kitchin and Dodge, Fuller, Cramer, Pold. Going back to Smith and Dean’s circular model, this will then allow us to move “through a series of processes” anti-clockwise “towards academic research”, counterbalancing the “right-hand side” of creative practice (2009, p. 21).

Working within a game design model, the first step in our research would thus be **setting the goal and values** of the game. These will be “necessary for the project to create meaningful play,” through which critical play can be engendered (Flanagan, 2009, p. 257). The design goals and values will be informed by our research question and theoretical orientation, predicating the core theoretical arguments developed through the game design process. This aligns with practice-led research methods which can be goal-oriented “consisting of an initial plan and a clear idea of an ultimate objective or target outcome” (Smith & Dean, 2009, p. 23).

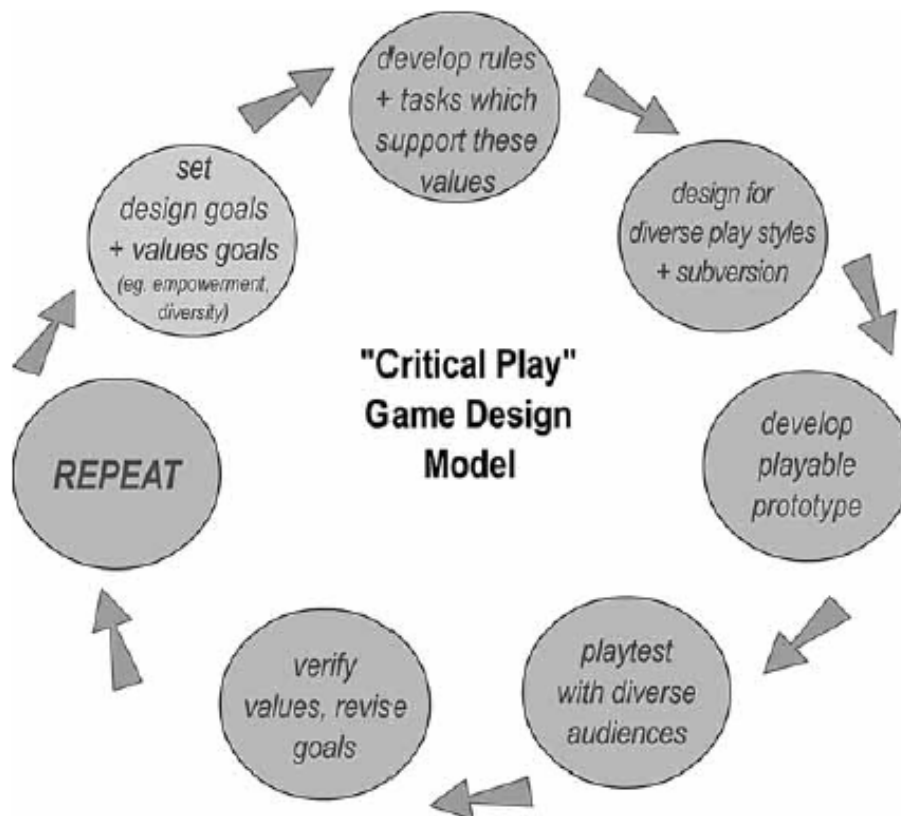


Figure 2.2. **Flanagan's critical play design model.** Reprinted from *Critical Play: Radical Game Design* (p. 257, by Flanagan, M., 2009. Cambridge, Massachusetts: MIT Press).

Following is the primary design phase of **developing rules that support the set values and designing for subversive play**. This phase consists of developing the game itself, the “framework of play” as Flanagan calls it, which comprises the actual game (2009, p. 257). For this we will draw on design approaches from Euro-games and traditional game design, as outlined by authors such as Mayer & Harris (2009), and as found in the canonical *Rules of Play: Game Design Fundamentals* (2004) by Katie Salen & Eric Zimmerman. However, oriented towards critical play, “instead of compliance to a pattern whereby the usual designers develop the usual ideas through the usual stages” we will seek to promote alterations to the “design process in a way that addresses intervention, disruption, and social issues and goals alongside of, or even as, design goals embedded into the mechanic and game elements” (Flanagan, 2009, pp. 256-7). It is consequently here that we will develop and implement our theoretical arguments as drawn from software studies literature to fulfil our play values.

The last series of steps consist of **prototyping, play-testing, and verifying or revising goals**. Through building a working prototype we will be able to “evaluate the game through the play tests and player comments” (Flanagan, 2009, p.258). The purpose of this is to then “verify that the values goals emerge through play, and revise goals and add or drop options based on feedback to ensure an engaging game and support the project values” (Flanagan, 2009, p. 258). For the playtest design we will draw on Fullerton, Swain, and Hoffman’s (2004) methodological writing on the subject. This stage will then comprise our project evaluation. Yet within practice-led research the evaluation of creative practices can “usefully borrow from the scientific model,” as Smith and Dean argues (2009, p. 27). Thus we will draw on practices from social sciences regarding qualitative data gathering and analysis to frame the playtesting as a source for academic insights as well.

The last design step is to **repeat** the process. As with practice-led research, iteration is integral to game-design. This element of practice is especially useful for the academic research as well, as Smith and Dean writes: “although research workers in both the humanities and the sciences usually have clear goals, engaging with processes along the way which allow for emergence, and permitting the project to shift in relation to them, is quite common and is often the secret of success” (2009, p. 23).

Conclusively, it is through the design process outlined above that we aim to develop not merely a practical project, but through it academic research as well. For if we pursue the practice-led model “idea generation leads to experiments, gathering of data and/or analysis of theory or criticism” which “may be followed by the development or synthesis of material [...] with outputs at a number of possible stages” (ibid., p. 21). Yet although the output will include formal academic research, practice-led research has an important “unstated implication” that defies how knowledge is produced and disseminated commonly by verbal and/or numerical means and which implies that research “needs to be treated, not monolithically, but as an activity which can appear in a variety of guises across the spectrum of practice and research” (ibid., p. 3). This resonates with the transformative concerns of critical theory and pedagogy and the overarching challenge of this thesis, which is to contribute to a public algorithmic literacy from the standpoint of humanities, yet go beyond the confines of its disciplinary domain. As Brown and Sorensen writes of their own practice-led research: “We believe that the impact of this rich diversity of

output is significant as it allows us to reach audiences outside traditional academic forums, to engage with a broader range of disciplines and to evoke greater public scrutiny and comment” (2009, p. 161). Conclusively then, by applying a practice-led research model we aim to answer our research question through the interrelated design of a critical board game and academic research drawing on software studies literature. Consequently we posit the following hypothesis:

Through developing a critical board game that applies software studies theory, we can contribute to public algorithmic literacy.

CHAPTER 3

Critical Board Game: Unveiling Interfaces

1. Critical Play Goals

As stated in the method section, Flanagan explains that it is necessary to set design and value goals for the game. Both of these are important in order to follow the critical play design method, and should consequently be weighed equally (2009, p. 257).

1.1. Game Design Goals

Our aim is to develop a board game that can be played by a wide range of players. Consequently, we intend to design what is called a 'gateway game'. Gateway games, write Mayer and Harris, can appeal to a wide audience as they are meant "to introduce those unfamiliar with designer games with the genre" (2009, p. 113). The authors explain that the appeal of these games is that they develop in less than an hour, have relatively few rules, but still offer "a level of strategy and interaction beyond expectations" (ibid, p. 113).

In terms of **formal game design** the goal is to achieve meaningful play. Salen and Zimmerman explain that "creating meaningful play is the goal of successful game design" (2004, p. 34), which indeed is the prerequisite if we aim to communicate our values through a board game that engenders critical play. Likewise, Flanagan draws on the definition of 'meaningful play' offered by Salen and Zimmerman when they write that "[m]eaningful play occurs when the relationships between actions and outcomes in a game are both discernable and integrated into the larger context of the game" (cited in Flanagan 2009, p. 94). However, Flanagan suggests that a main component that makes play meaningful "is the social and cultural context" in which play takes place (2009, p. 94). And as critical games "looks outward to the society in which it is played, providing a game as the medium to critique the non-game system" (Grace, 2014, p. 5), critical play is thus contingent on meaningful play unfolding within the gameplay. In this sense Flanagan writes that critical play games seize the play space to "occupy play environments and activities

that represent one or more questions about aspects of human life” (2009, p. 9). This subversive act differs from games where its play values might not be connected to the type of inquiries that critical play does by focusing on social critique.

Like Salen and Zimmerman, Erlhoff and Marshall define game design as “a complex, multilayered design activity, whereby systems of meaning (games) are created through the design of rulesets resulting in play” (2008, p. 185). The authors contend that the core task of game design is to layout interactive systems, which afford “spaces of possibility for players to explore” (ibid). Drawing on this conception, it can be argued that it is in the area of possibilities that meaningful play can be engendered. For the authors, the meaning is derived from the objects which are part of the system. In this regard, Salen and Zimmerman offer a valuable account of four components involved in the designing the context of play through which 'meaningful play' may occur: “spaces, objects, narratives, and behaviors” (2004, p. 41). These then becomes areas for implementing systems of meanings imbued with values. By achieving meaningful play, we thus create a play environment to occupy and ask questions about the larger context, which in the case of this game will be centered in the blackboxed algorithms materialized in the cultural software players interact with in their real lives.

1.2. Value Goals

Zimmerman defines play values as “the abstract principles that the game design should embody” (in Fullerton et al., 2004, p. 16). Flanagan, however, states that in critical play design there is a necessity of “a constant reflection on the humanistic themes, or values during design” (2009, p. 255). We thus adapt our algorithmic literacy criteria as the following critical play value goals:

Firstly, **the board game should allow players to develop awareness of algorithms in everyday cultural software.** This entails that the frame of the game should have elements that are relatable to cultural software users. In this way, we aim to respond to Berry’s and Manovich’s challenge of bringing the socio-technical dimensions of software back into visibility. Moreover, this goal also involves contextually exposing how the algorithms are blackboxed through a myriad of interfaces in cultural software.

Secondly, **the board game should aim at to foster a critical stance towards algorithms in their contexts by making players realize how algorithms**

and software can be ideological. As it was argued in the methodology, we will draw on software studies' concept of the socio-technical production of software, which shapes the development of software itself. This also entails an understanding of algorithms are part of software ecosystems located within a certain economic system. For as Flusser pointed out, the *program* of the *apparatus* is dictated by its socio-economic context, wherein the power is not in the ownership of the device but on the control of the software. Additionally, by offering a digital materialism perspective that supports technical and cultural dimensions of software, we aim to develop a play environment where the players can explore how algorithms have biases, and dictate grammars of actions as they are materialized in software.

Thirdly, **the board game should aim at instilling a technical appreciation of algorithmic technologies without relying on the technical knowledge of players.** As it was argued in the previous chapter, the techno-positivist worldview adds a technical layer to the public sphere. Thus, taking into consideration the transformative paradigm from where this project has been conceptualized, our aim is to subvert the exclusion of non-technicians from the debate about technology. This will inform the communication strategy of the game, as well as the other game elements, such as the objects, the theme, and the rules.

These value goals tie back to our theoretical orientation towards a critical pedagogy: as literacy by Freire is not merely a technical act of reading and writing, but as Park writes, “can also mean the capacity to ‘read the world’ (Freire, 1987) and question the basic assumptions of society” through which “people can generate deeper and different understandings of texts; question social and economic realities; and reimagine the status quo” (Park, 2012, pp. 629-630). For Crocco, following Freirean pedagogy, the strategy behind creating a critical gameplay pedagogy is through codification of game elements (2011, p. 30). This serves to create a game “as an artifact to be critically examined for the ways it reifies hegemonic ideology” (ibid.). Crocco emphasizes that “the point is to resist empathy and passive identification with game content and point of view so that the player can maintain a critical perspective”¹⁹ (ibid.). For Freire, it is in the relation between codification and decodification that conscientization occurs, as he writes:

¹⁹ Crocco's example of codifying a board game is in terms of a “modified version of *Monopoly*” in which he “[codifies] the theme of social mobility under capitalism” through letting players “simulate the

While codified representation is the knowable object mediating between knowing subjects, decodification—dissolving the codification into its constituent elements—is the operation by which the knowing subjects perceive relationships between the codification’s elements and other facts presented by the real context—relationships which were formerly unperceived (cited in Morrows & Torres, 2002, p. 124).

It is useful to point out that for Morrow & Torres, reading Freire in relation to critical theory means that the process of ‘codification’ and ‘decodification’ can be understood theoretically as a “strategy of nondogmatic cultural criticism”—as found in the methodological approach of critical theory (2002, p. 130). The process of codification thus reflects our own methodological approach of using software studies, as a critical theory on software, to develop every element and mechanic in the game as will be discussed throughout this chapter. This thus creates a situation where, through players engaging critically with codified game elements, our value goals can potentially be achieved.

2. Framework of Play

In developing a game it was important to firstly conceptualize what the ‘frame’ of the game will be, for importantly the frame circumscribes the ‘reality’ of the game, communicating “the relationship between the artificial world of the game and the ‘real life’ contexts that it intersects” (Salen & Zimmerman, 2004, p. 94). This frame can also be called the theme of the game, or in Huizinga’s terms, the “play world”—the “temporary worlds within the ordinary world” that delineates the boundaries of the magic circle in which the game is played (ibid., p.96).

In a coding literacy board game such as *Code Monkey Island* (2014), for example, players learn “fundamental programming concepts” such as data structuring and Boolean logic within the following frame: “Each player becomes the wise leader of their own tribe of monkeys, and it’s their job to guide them safely around the island!” (Sidhu, 2014, n.p.). There is a myriad of such code literacy games that in their efforts to make learning more engaging, promote abstracted and decontextualized engagement with the structural components of software through

effects of class inequality by playing characters who begin the game with different amounts of money, land, and privilege, and compete to win using the game’s normal mechanics” (2011, p. 31).

fictional ‘play worlds’ or framing²⁰. In deliberate opposition to this, the framing of our game contextualizes player engagement with algorithms through the materiality of everyday software. The consequent frame, or ‘play world’, of our game is thus set out in the theme prologue of our rule book (Appendix 1):

You no doubt use countless apps on your phone everyday, not giving a second thought to what goes on behind those friendly flat user-interfaces. Unveiling Interfaces takes you inside the black box of these algorithmic machines as you play as an app developer, trying to succeed in a competitive tech-market by developing and publishing the most high-grossing apps possible. But along the way you'll have to decide: at what cost are you willing to win? (Appendix 1: Rule Book, p. 1).

The frame of the game is derived from software studies’ methodological orientation towards a digital materialism (Casemajor, 2015, pp. 4–5). For Casemajor, Manovich’s pivotal *Language of New Media* set this orientation of software studies through providing five principles of digital materiality: numerical representation, modularity, automation, variability and transcoding. Through these principles, she argues, “special attention” could be given “to the interface and usability of programs” that could be carried on by an analysis “observing the formal organisation of information on the screen, its interactive potential and the phenomenological aesthetic experience proposed to the user” (ibid., p. 8). Notably, Casemajor also observes how through this “[t]he social dimension of use and the cultural dimension of meaning are put forward to explain how media objects are ‘experienced, created, edited, remixed, organized and shared’” through software (ibid.). It is in extending this approach that Manovich later asserts: “Software Studies has to investigate the role of software in contemporary culture, and the cultural and social forces that are shaping the development of software itself” (2013, p. 10).

Likewise, as part of the formalization of software studies through a series of publications by MIT Press²¹, in their *Code/Space: Software and Everyday Live* (2011) Kitchin & Dodge write a manifesto of software studies stating it should focus

²⁰ A common practice in educational gamification is the use of a fictional theme to “[form] the basis for some underlying epic drama capable of taking gameplay from the doldrums of everyday life and [create] an entire alternate reality” in efforts to engages students (Niman, 2014, p.112).

²¹ In May 2008 *SoftWhere*, an international workshop in software studies, was held at UCSD during which the discussion was about “the meaning of studying software cultures and the direction and goals of Software Studies” (SoftWhere, 2008, n.p.). The most important outcome of the workshop was a collaborative project “to further develop the field including a new book series on Software Studies @ MIT Press” (ibid.). A month later MIT Press new book series was established with Fuller, Wardrip-Fruin and Manovich as editors (Fuller, Sep 13, 2008, n.p.).

its “analysis explicitly on the conceptual nature, and productive capacity of software, and its work in the world, from a critical social scientific and cultural perspective” (2011, p. 246). This perspective becomes intertwined with Manovich’s digital materialism when Kitchin and Dodge quote Manovich’s argument that “if we don’t address software itself, we are in danger of always dealing with effects rather than causes, the output that appears on a computer screen rather than the programs and social cultures that produce this output” (cited on Kitchin & Dodge, 2011, p. 246).

For the purpose of developing a ‘play world’ for our game that could convey our value goals, there was a consequent need to frame player’s engagement with algorithms, not as decontextualized and abstracted concepts communicated through the actions of monkeys on an island (as is the case of *Code Monkey Island*) but as materialized in software situated in their actual socio-cultural contexts. For, as Goffey argues in *Software Studies: A Lexicon* (2008) (another key text published in MIT’s initial software studies series), algorithms’ materiality as software becomes apparent in cases such as “when a commercial website uses data-mining techniques to predict your shopping preferences” (p. 15). It is thus imperative to the software studies methodology is the necessity of locating any kind of software critique in the specific instances of its materialization. For as Kitchin and Dodge write:

[S]oftware is diverse in nature. It takes form in the world through multiple means, including hard-coded applications with no or limited programmability (embedded on chips), specialized applications (banking software, traffic management systems), generic user applications (word processors, Web browsers, video games), and operating systems (Windows, Mac OS, Linux), that run on a variety of hardware platforms. (2011, pp. 4–5)

It is thus important to move from vague notions of ‘software’ as a macro-phenomenon to its specific instances. As Fuller writes in *Beyond the Blip*’s (2003) introduction: “Theorisations of software that are able to operate on the level of a particular version of a program, a particular file structure, protocol, sampling algorithm, colour-scheme, API, Request For Comment, and so on, are necessary” (pp. 17-18).

Given the above stated, it is not surprising that as in *Language of New Media*, in *Software Takes Command* Manovich firstly addresses the categorical instances of the software he aims to study. Manovich initiates a distinction between “visible” software used by consumers and “grey” software that runs the infrastructural

processes and systems of society through, for example, “industrial automation software” (2013, p. 21). The subset of visible software that Manovich focuses on, however, is what he calls “cultural software” (ibid.). The author then continues to outline sub-categories of what he considers cultural software, which itself falls under the general umbrella of “application software”, writing: “cultural software also includes tools and services that are specifically designed for communication and sharing of information and knowledge” (ibid., p. 26). This category of software he calls ‘social software’, which includes “search engines, web browsers, blog editors, email applications, instant messaging applications, wikis, social bookmarking, social networks [...] Facebook, the family of Google products (Google Web search, Gmail, Google Maps, Google+, etc.), Skype” (ibid.)²².

It is thus that within the frame of our game the central subject is the subcategory of ‘social software’, and specifically social networks as found on smartphone devices. This focus follows Manovich’s argument that the term ‘application software’ itself is changing in meaning as we move from discrete desktop applications to mobile- and web-based applications (ibid.). Moreover, as social software such as *Facebook Messenger*, *Whatsapp*, *Instagram*, *Youtube* (that effectively functions as both a media and social software), and Chinese social networks such as *WeChat*, *QQ*, and *Tik Tok* represent the most downloaded applications on *Android’s Google Play* and the *iPhone’s App Store* app marketplaces we thus argue that this is an ideal space for fostering algorithmic literacy as this is where the public prominently engage with algorithmic technology (Sensor Tower, 2018, n.p.). Through using these instances of software to frame our game, we importantly also work within the Freirean pedagogical model, as “education must be contextualized, i.e., it should arise from the concrete experience of the educands” in order to foster a social conscientization (Tygel & Kirsch, 2015, p. 115).

By framing the theme between the actions of developing social software, and publishing it to an app market accessible through a smartphone device, the game draws heavily on Berry’s work *Critical Theory and the Digital*. By drawing on this publication we can respond not only to a digital materialist critique of software but

²² Manovich however makes it clear that his categorizations are constructed in a flux, acknowledging that “[t]hese and all other categories of software shift over time” as was the case in the 2000s when social software and media software were intertwined through the integration of multi-media experiences in social networking platforms (2013, p. 28).

also to a cultural critique that works within our methodological orientation of software studies as a critical theory of software. As Berry argues:

The speed and iteration of innovation in this area of technology might be incredibly fast and accelerating, but software can be materialized so that we may think critically about it [...] The platform can offer a standpoint from which to study software and code, and hence the digital, but it is not sufficient without taking into account the broader political economic contexts. Indeed, one of the difficulties with studying software is that it requires a complete assemblage of technologies in order to work at all. (2014, p. 56)

Moreover, in *Critical Theory and the Digital* Berry introduces the concept of compactants (computational actants), which affords the engagement between the users and the machinery. The role of the compactants “can be understood as a dual surface/machinery structure,” and “are designed to function across the interactional and codal layers—as is much software in use today.” (ibid., p. 68). This parallel software engineering conventions of front end (the presentation layer) and back end (data access and code) development. The author argues that software as compactants comprises of “two faces”: the “commodity face” of software, “accessible via the interface/surface” which provides the stable commodity, service, or functional value of the software as consumer product; and inversely, the “mechanism face” of software that is found in its source-code and “contains the mechanisms and functions ‘hidden’ in the software” (ibid., p. 69).

This approach informs not only the theme, but also the core mechanics of our game which will be discussed throughout this chapter. By framing our game within the context of software production and consumption, we aim to provide a context to the afore described relation between the ‘two faces’ of software. This approach is central to our game as an understanding of software as constituting of hidden algorithms is inhibited by the *blackboxing* of software through its commodifying interfaces. Finally, we hold that a critical stance towards software can thus be achieved by ‘unveiling the interface’ and bringing the underlying mechanism—the algorithm—into view for players. Hence the chosen name of the board game: *Unveiling Interfaces*.

2.1. Object of The Game

Game historian David Parlett argues that games usually have a two-fold structure, consisting of an ‘ends’ and ‘means’. In terms of the ‘means’, a game is an “agreed set of equipment and of procedural ‘rules’ by which the equipment is manipulated to produce a winning situation”, while the ‘ends’ comprises that ‘winning situation’ as players compete to “achieve an objective” (cited in Salen & Zimmerman, 2004, p. 4). It is between these two-fold structure that “the unique enjoyment of game play” emerges (Salen & Zimmerman, 2004, p. 6). Hence the objective of *Unveiling Interfaces*, set within the frame of the game, is as follows:

As app developers each trying to launch their social media startup empire, players have to compete for revenue as they try and publish as many profitable apps as possible before the AppMarket becomes saturated with the products of competitors. This is how players win:

- 1. Players receive **Developer Briefs** which contain different possible software feature combinations according to which they can build a profitable app using **Software Tiles**. Each feature combination corresponds to different levels of revenue for their apps.*
- 2. Then, players have to develop and publish as many profitable apps as possible, using the **Software Tiles** they collect to complete their **Developer Briefs**.*
- 3. Players compete for limited spaces on the **AppMarket** playing board when publishing their apps in order to make the most profit. Because once there are no more spaces left on the board for new apps to be published, the game ends and the player with the most revenue earned wins and is crowned the next member of GAFA (Google, Amazon, Facebook, Apple). (Appendix 1: Rule Book, p. 1)*

Yet, in alignment with our value goals, to prompt critical play the objective of the game as set out in the rule book concludes:

But be careful: The algorithms inside of the apps players build might have unintended consequences for its users and for you too! (Appendix 1: Rule Book, p. 1)

2.2. Setup

Important for setting the frame of the game is the physical game components, for these set the boundaries of the magic circle (Salen & Zimmerman, 2004, p. 107).

The board game elements comprise the following:

1 AppMarket playing board
1 Week Marker
60 Revenue Tokens (15 square, 15 circle, 15 diamonds, 15 pentagon)
4 Bloatwares
12 Developer Briefs
36 Software Tiles (6 blue, 6 turquoise, 6 pink, 6 orange, 6 green, 6 yellow)
20 Event Notification Cards
Bring Your Own Device: Each player's personal smartphone device will be needed in the game
(Appendix 1: Rule Book, p. 1)

These are set up as follows to create the playing field (Fig. 3.1.):

Board and Pieces

1. Place the **AppMarket** board within easy reach of all players. Place the **Week Marker** on the first week of the calendar on the playing board.
2. Place **Bloatware** on the last open slots of the playing board, depending on the number of players:
 - 2 players = 3 Bloatware
 - 3 players = 1 Bloatware
 - 4 players = 0 Bloatware
3. Each player has to pick a set of **Revenue Tokens**, based on its shape, to keep.
4. To keep their game scores, players must take out their **personal smartphones**, open up the **calculator app** and set it to zero (0).

Cards

1. Shuffle the **Event Notification Cards** and place them facing down on their allocated spot on the board.
2. Shuffle the **Developer Briefs** and deal three (3) to each player. Place the remaining **Developer Briefs** face down next to the board.
3. Shuffle the **Software Tiles** and deal four (4) to each player. Place the remaining Tiles user-interface-side up on the first spot allocated to them. Then place the top 4 (four) **Software Tiles** on the four (4) remaining spots (Appendix 1: Rule Book, p. 2)

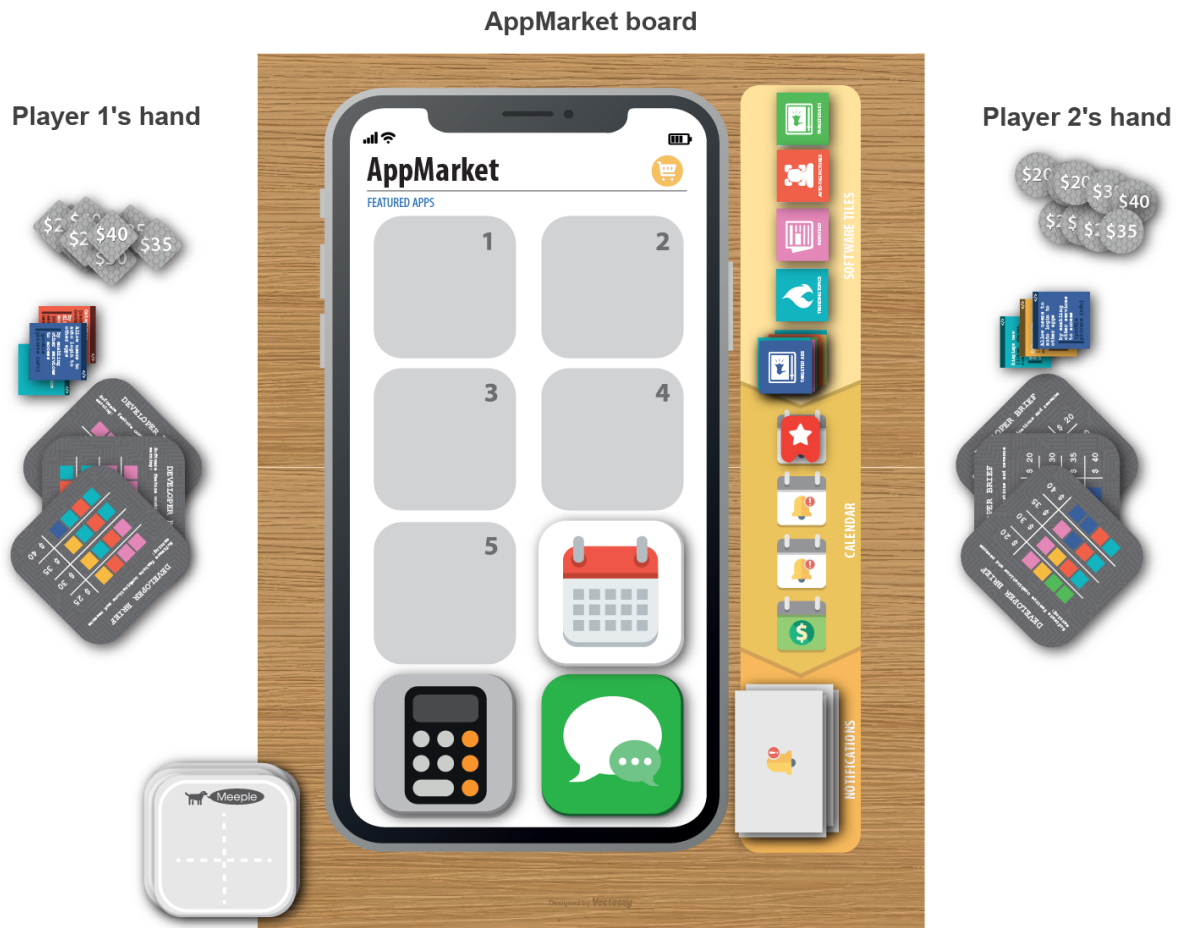


Figure 3.1. Illustration of the game setup for two players.

The way in which software studies theory is codified into the game elements set out on the board, and how they interrelate, will be explained in the rest of this chapter in terms of how these elements interact as game mechanics. For as Michael Erlhoff Tim Marshall writes: “Game designers must choose which components make up the game, and assign behaviors and relationships to each of these components” which are “simply kinds of rules that describe how an object can act” (2008, p. 186).

3. Game Mechanics: Operational Rules and Game Elements

Following Flanagan’s critical play method, after setting the design and value goals and the object of the game, the next step was to “develop rules and constraints that support [those] values” (2009, p. 257). As important as the frame of the game is “at the very heart of games, is RULES, the space of games framed as formal systems” (Salen & Zimmerman, 2004, p. 6). Accordingly, the aim of this section is to describe the dynamic system of the game (Flanagan, 2009; Salen & Zimmerman, 2004). As

Zimmerman elucidates, understanding games as participatory and dynamic systems helps to avoid focusing only on the “content, narrative, or aesthetics” (cited in Fullerton et. al., 2004, p. 330). Instead, Zimmerman urges game designers to ask “more fundamental questions”, given that games should “be understood not just as content but as action” (ibid.). It is in this that Salen & Zimmerman describe game systems as composed by rules which “constitute the inner, formal structure of the game” (2004, p. 125). The authors propose a classification of rules within a game that is related to different levels: operational, constitutive, and implicit rules. **Operational rules**, are also known as “rules of play” that “direct the player’s behavior” (ibid., p. 139). The **constitutive rules** are the “core mathematical rules of a game” (ibid.). Finally, the **implicit rules** are the “unwritten rules of etiquette and behaviour that usually go unstated when a game is played” (ibid.).

While the constitutive rules will only be referenced implicitly, the focus of this discussion is in the development of the operational rules. For it is in these rules that we codified the computer ontology that software studies propose. Importantly, following Bogost’s writing on procedural literacy (2005) it is also the rules of the game that enables an appreciation of algorithms and software without technical experience through turning the frame of the game into a procedural system.

Furthermore, as Flanagan explains, rule designing also includes the components of the game such as tokens, cards, design props, that are “necessary to support the game’s values and play” (ibid., p. 257). Henceforth, the operational rules and game components comprising those rules will be discussed in the order in which they are presented to the players in the Appendix 1: Rule Book. Each discussion will start with a description of relevant game mechanics, including a explanation of their attributes and internal relationship to other game components, so that the discussion of the operational rules and codifying theory relating to their design process them can be properly contextualized.

3.1. Phase 1:

There are two phases that structure each player’s turn during each round of play until the game concludes. As stated in the rule book:

*In the first **Phase**, players start their turn by completing one of the following **Actions**:*

- **ACTION 1: Collect Software Tiles**

- **ACTION 2: Draw a new Developer Brief**
 - **ACTION 3: Develop and publish an app**
- (Appendix 1: Rule Book, pp. 2-3)

Phase 1 constitutes the choices players enact during the game. For “[p]lay doesn’t just come from the game itself, but from the way that players interact with the game” (Sales & Zimmerman, 2004, p. 33). Importantly, in these interactions players “are making choices” (ibid.).

3.1.1. Action 1: Collect Software Tiles:

In this action, a player has to collect two (2) Software Tiles. They can either:

- *Draw two (2) **Software Tiles** from the top of the Software Tile deck (blind draw).*
- *Draw two (2) **Software Tiles** from the four tiles on the board with the code-side up. If the player does this, he needs to replace it on the board with the **Software Tile** from the top of the deck. (Appendix 1: Rule Book, p. 3)*

There are six different colour Software Tiles in the game, each colour corresponding to different content either on their fronts or their backs. The tiles are thus importantly double-sided. The front of the Tiles correspond to one of nine software features that a social software application might have, such as ‘Newsfeed’, ‘Recommendations’, or ‘Trending Topics’. These features are labelled and graphically represented by icons taken from social software conventions and would thus be familiar to habitual users of social software applications. On the back of the Tiles there are short statements that corresponds to the feature on front. A tile that reads “Trending Topics” on one side, might then read “Populate trending topics based on [user’s personal interests]” (Appendix: Game Database)²³ on the other.

Within *Unveiling Interfaces* these Tiles are conceived as part of a tile-placement mechanic, as popularized by designer board games such as *Carcassonne* (2000) and *Cacao* (2015) in which “players draw and lay down tiles to either advance the game or to establish a playing area within which other actions can be taken” (Mayer & Harris, 2008, p. 115). Such tile-placement often work according to rules of spatiality and pattern-building. In this game, players collect Software Tiles in each round with the purpose of completing the Developer Briefs that necessitate players

²³ A full list of software tiles’ content can be found in Appendix 2: Game Database and Appendix 3: Printable Prototype.

to complete certain Tile colour-combinations. In this sense, the Tiles can then be placed on the game board in order to occupy space and allow players to gain revenue necessary for winning. For facilitating critical play, however, the Software Tiles are crucial for understanding the critique of algorithms as materialized in software. Responding to our value goals, it is the Software Tiles that are meant to foster an ‘algorithmic awareness’ in players.

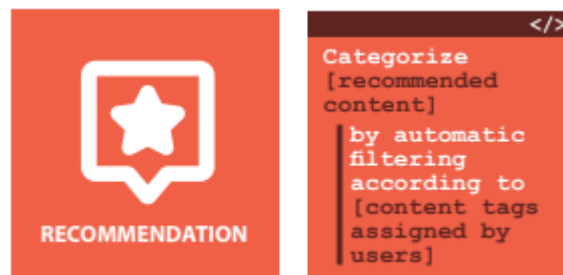


Figure 3.2. The front-side and back-side of a red ‘Recommendation’ app (Appendix 3: Printable Prototype).

3.1.1.1. Software as algorithms

In a 1958 article for *American Mathematical Monthly*, John W. Tukey wrote, in what is purportedly the first usage of the term ‘software’: “Today the ‘software’ comprising the carefully planned interpretive routines, compilers, and other aspects of automative programming are at least as important to the modern electronic calculator as its ‘hardware’” (cited in Fuller, 2008, p. 2). This conception is echoed in one of Manovich’s central principles of new media, a principle that he returns prominently in his book *Software Takes Command* (aptly subtitled *Extending the Language of New Media*) when he writes:

One of the key uses of digital computers from the start was automation. As long as a process can be defined as a finite set of simple steps (i.e. as an algorithm), a computer can be programmed to execute these steps without human input. (2013, p. 128)

It is for this reason that in *Unveiling Interfaces* the social software apps are composed of Software Tiles. These Tiles are not only constituted by the recognizable software features and accompanying icons contained on the front of the Tile, but by the content on the back as well: the software’s underlying automatizing algorithms. While the front of the tiles might correlate to what Berry calls the ‘commodity face’ of software as compactants, the back becomes the

‘mechanism face’. In *Unveiling Interfaces*, we have designed the Software Tiles to be defined by their explicitly stated algorithmic content instead of their correlative ‘commodity face’. With this design, we aim to encourage players to appreciate the software features they might be familiar with in real life as constituting of algorithms in the game system. Moreover, in the game’s constitutive rules we have designed a colour system where each software feature has two possible colours (a ‘Newsfeed’ tile can be either pink or turquoise); however, there are two different newsfeed-algorithms in the game, each exclusive to only one of the two colours (See Fig. 3.3).



Figure 3.3. The two code-sides of the ‘Newsfeed’ Software Tile. The newsfeed-algorithm “Prioritization [uploaded content] in newsfeed that has been shared by user’s friends” is only pink, while “Prioritization [uploaded content] in newsfeed boosted by advertiser spending” is only turquoise (Appendix 2; Appendix 3).

For the player then, it is the algorithm rather than just the feature on the user interface (UI) side that needs to be considered if one is to understand the actions of the software on a specific colour of Tile. As Goffey and Fuller write: “Designing a piece of software [...] of any kind entails a process of abstraction: capturing the logic of what an application, a gadget, a system, or a device is supposed to do within a set of algorithms” (2012, p. 77).

Working with 3.5 x 3.5 cm tiles places a particular constraint on us in terms of size and consequent surface space for content. While conscious of the dangers of reductive oversimplification, we did not intend to go into the technical minutiae of the software we drew on. Especially since we aimed to make the language in the game accessible to non-expert players. Moreover, as “[w]ithin code algorithms are usually woven together with hundreds of other algorithms to create algorithmic systems” (Kitchin, 2017, p. 20), it is importantly not disentangling the individual algorithms that is of concern. Rather, we aimed to reflect larger algorithmic systems in our Software

Tiles. Thus we turned to the four categories of algorithmic systems²⁴ that Nicholas Diakopoulos, a notable professor in computational journalism, proposes. Of these Diakopoulos writes: “We can start to assess algorithmic power by thinking about the atomic decisions that algorithms make, including prioritization, classification, association, and filtering” (2013, p. 3). Diakopoulos’ lays these out as follows:

- **Prioritization:** “[S]erves to emphasize or bring attention to certain things at the expense of others. [...] Embedded in every algorithm that seeks to prioritize are criteria, or metrics, which are computed and used to define the ranking through a sorting procedure” (ibid., p. 4).
- **Classification:** “[D]ecisions involve categorizing a particular entity as a constituent of a given class by looking at any number of that entity’s features” (ibid., p. 5)
- **Association:** “Association decisions are about marking relationships between entities. [...] A related algorithmic decision involves grouping entities into clusters, in a sort of association en masse” (ibid., p. 7).
- **Filtering:** “[I]nvolves including or excluding information according to various rules or criteria” (ibid., p. 8).

For Diakopoulos, algorithms do not necessarily fall into one or the other category as they are often “chained in order to form higher-level decisions and information transformation” (ibid., pp. 3-4), leading to composite categories. It is the last of his categories, ‘filtering’, that most exemplifies these composites as he writes: “Indeed, inputs to filtering algorithms often take prioritizing, classification, or association decisions into account” (ibid., p. 8). Given these points, we decided to work with the first three categories explicitly, and use filtering as an implicit category between these. Our design approach to the Software Tiles was thus as follows: we identified three different software features found in social software such as Twitter or Facebook per each of Diakopoulos’ first three categories²⁵. Within these, we made two variable algorithms based on different case-studies investigated by

²⁴ The turn to Diakopoulos is not incidental as his algorithmic epistemology is informed by the same technocultural problematic outlined in our opening literature review. It is through drawing on his work that we can consequently facilitate a relationship between software studies and the problematic addressed by our research within the mechanics of our game (as will be further discussed in Phase 2 of the game).

²⁵ This was done by also looking at how Diakopoulos applied his categories to actual software, for example: “In news personalization apps like Zite or Flipboard news is filtered in and out according to how that news has been categorized, associated to the person’s interests, and prioritized for that person” (2013, p. 8).

computational journalists (which will be addressed in Phase 2 of the game). The product of this design process was documented in a spreadsheet where three software features relating to each of the three categories is outlined, and within these two different possible algorithms per feature (see Fig. 3.4).

UI SIDE		CODE SIDE				
#	Feature	Algorithm /logic	Code A (Logic) WHAT	Code B (Control) HOW	Positive(P) /Negative(N)	\$
1A	Trending topics	Prioritization Prioritization, ranking, or ordering serves to emphasize or bring attention to certain things at the expense of others. [...]Embedded in every algorithm that seeks to prioritize are criteria, or metrics, which are computed and used to define the ranking through a sorting procedure.	Populate trending topics	based on [interests shared across whole userbase]	P	5
1B	Trending topics		Populate trending topics	based on [user's personal interests]	N	10
2A	Newsfeed		Prioritization [uploaded content] in newsfeed	shared by user's friends	P	5
2B	Newsfeed		Prioritization [uploaded content] in newsfeed	boosted by advertiser spending	N	10
3A	Notifications		Displays new [likes, shares, comments]	instantly and automatically	P	5
3B	Notifications		Displays new [likes, shares, comments]	gradually distributed over time	N	10

Figure 3.4. **Part of the spreadsheet we used to develop our Software Tiles.** The full spreadsheet table is available in Appendix 2: Game Database.

3.1.1.2. The Constituency of an Algorithm

As seen in the Figure 3.4, the function or algorithm in the Tiles is divided into two primary components. This follows the often cited (not least so by software studies scholars) computer-science equation of Robert Kowalski: “algorithm = logic + control” (cited in Goffey, 2008, p. 15). Logic here being, as Kitchin writes: “the problem domain-specific component [which] specifies the abstract formulation and expression of a solution (what is to be done)”, and control: “the problem-solving strategy and the instructions for processing the logic under different scenarios (how it should be done)” (2017, p. 17). Therefore, our Tiles are divided into two sections, for easy readability, but also to correspond to the logical structure of algorithms and thus communicate an implied technicity to players. For example, the algorithm for the ‘Recommendations’ feature can be divided as follows:

Logic: Categorize [recommended content].

Control: by contextual analysis of [uploaded content and periodical feedback from users]. (Appendix 2: Game Database)

Yet what algorithms do is also related to another constituent. As Fuller and Goffey point out: “An algorithm is sometimes considered the sum of logic and control. But

equally, a program may be considered the sum of algorithms and data structures” (2012, p. 83). It can be argued that part of Goffey’s initial departure from the computer-science’s abstracted conceptualization of algorithms is in algorithms’ dependence on data in their materialization. As Goffey writes:

Algorithms obviously do not execute their actions in a void. It is difficult to understand the way they work without the simultaneous existence of data structures, which is also to say data. Even the simplest algorithm for sorting a list of numbers supposes an unsorted list as input and a sorted list as output (assuming the algorithm is correct). (2008, p. 18)

Likewise, Manovich writes that the “two fundamental components of all modern software” is “data structures and algorithms” (2013, p. 197). It is following this materialist approach to algorithms that the data becomes an important part of the algorithm on the Tile, which will likewise come into more focus during Phase 2 of the game discussed later on. Taking this comprehensive understanding of algorithms, the design of the back of the Software Tile is bifurcated between ‘logic’ and ‘control’, and it should be noted that the data components are also set apart by a pair of square brackets (‘[’, ‘]’).

3.1.1.3. Code and the materiality of algorithms

On the Software Tiles’ back-sides, the ‘data’ operated by the algorithm follows the notational conventions of the programming language Python²⁶, in which basic data structures are often represented in the following way.

Data Structure	Data	Notation
Lists	1,2,'a','b'	l = [1, 2, "a", "b"]
Tuples	1,2,'a','b'	t = (1, 2, "a", "b")
Dictionaries	'a'=1,'b'=2	d = {"a":1, "b":2}

Figure 3.5. Examples of notation of data structures in Python.

²⁶ The choice of implementing Python notation is three-fold here. Firstly, Python is an immensely popular general-purpose programming language—and increasingly so for machine learning applications. As Müller and Guido write in the *O’Reilly Introduction to Machine Learning with Python*: “Python has become the lingua franca for many data science applications” (2017, p. 5). Secondly, Python’s notational conventions are some of the most simplistic. Hence, it is the same reason why it would appeal to beginner programmers, that would make it appealing to employ in a board game aimed at non-experts. As Phil Spector who teaches python courses at Berkeley University states: “Python grew out of a project to design a computer language which would be easy for beginners to learn, yet would be powerful enough for even advanced users. This heritage is reflected in Python’s small, clean syntax and the thoroughness of the implementation of ideas like object-oriented programming” (2005, p. 7). Lastly, it is the programming language we as researchers are most well-versed in, allowing us employ a modicum of experience in formatting and formulating the content.

In this notation, however, we are coding an important software studies critique of algorithms which relates to our value goals. For in textually writing our algorithm and accompanying data through mimicking programming language notation, we are implicating the materiality of algorithms in software. As Goffey again writes, while quoting the “admirable succinctness” of Les Goldschlager and Andrew Lister’s conception of algorithms in *Computer Science: A Modern Introduction* (1982):

The algorithm “is the unifying concept for all the activities which computer scientists engage in.” Provisionally a “description of the method by which a task is to be accomplished,” the algorithm is thus the fundamental entity with which computer scientists operate. [...] An algorithm is an abstraction, having an autonomous existence independent of what computer scientists like to refer to as “implementation details,” that is, its embodiment in a particular programming language for a particular machine architecture. (2008, p. 15)

Yet conversely, software studies scholars are concerned exactly with these ‘implementation details’, holding that algorithms are not merely theoretical entities, but “have a real existence embodied in the class libraries of programming languages, in the software used to render web pages in a browser (indeed, in the code used to render a browser itself on a screen)” (ibid.). Perhaps it is because of this that software studies scholars have a penchant for talking about software as being composed fundamentally of code, rather than of algorithms. As Adrian Mackenzie writes: “What software does and how it performs, circulates, changes and solidifies cannot be understood apart from its constitutions through and through as code [...] Code cuts across every aspect of what software is and what software does” (cited in Kitchin & Dodge, 2011, p. 24). Berry, who we draw on heavily throughout this thesis, writes: “Attention to the materiality of software requires a form of reading/ writing [...] through attentiveness to the affordances of code” (2014, p. 71). It is also not surprising that in their *Code/Space*, in which they likewise rely on Berry’s work, Kitchin and Dodge write: “software consists of lines of code [...] that, when combined and supplied with appropriate input, produce routines and programs capable of complex digital functions” (2011, pp. 3-4).

However, this above statement is not necessarily incongruous with Kitchin’s more recent equation that software *is* algorithms (2017). While it might be argued that reducing software to being fundamentally ‘algorithm’ or ‘code’ might have

different ontological implications, there is also a direct material relation between the two. As Berry sets out sequentially how software is usually composed:

[I]n computer programming, to explain how a particular piece of code works, and to avoid talking about a particular instantiation of a programming language, algorithms are written out in 'pseudocode'. That is in a non-computer, non-compilable language that is computer-like but still contains enough natural language (such as English) to be readable. That is, the algorithms allow the process to be described in a platform/language independent fashion, which can be understood as a pre-delegated code form. This is then implemented in specific programming languages. But these algorithms eventually have to be turned into a computer programming language that can be compiled into prescriptive code, and therefore run as software. (2011, p. 52)

To talk about software as algorithms, it is thus necessary to talk about the encoding of algorithms as source code, "the textual form of programming code that is edited by computer programmers" (ibid., p. 29). In her *Writing Machines* (2002) Katherine Hayles likewise investigates how the materiality of programming relates to the writing of text and even "adds the materiality of the text itself to the analysis in a similar way to those who consider code to be material" (Cox & Ward, 2009, p. 209). We could thus not *but* codify these algorithms textually in the game—just as in software you cannot *but* express them through the textual materiality of written computer code. For Berry, the mechanism-face of software, as containing the functions of the software, consequently also comprises of source code (2014, p. 69). We thus proceed to refer to the back-side of the Software Tiles as the 'code-side', for it is through the code that the algorithm is materialized in the software. As Kitchin and Dodge argue:

Code at its most simplistic definition is a set of unambiguous instructions for the processing of elements of capta in computer memory. Computer code [...] is essential for the operation of any object or system that utilizes microprocessors. It is constructed through programming — the art and science of putting together algorithms and read/write instructions that process capta. (2011, p. 24)

While code has a performative dimension that will be investigated in Phase 2 of the game, in Phase 1 we focus on "the textual and social practices of source code writing [...] That is, specifically concerned with code as a textual source code instantiated in particular modular, atomic, computer-programming languages as the object of analysis" (Berry, 2011, pp. 31-32). For through collecting Software Tiles, and publishing them in Action 2, which will be discussed next, *Unveiling Interfaces* is

designed to allow players to understand software as constituting of algorithms, and thus allow them to develop an awareness of algorithms in the social software they encounter in their daily lives, as stated in our value goals.

3.1.2. Action 2: Draw a new Developer Brief

A player can also draw a new Developer Brief from the available Developer Briefs stacked next to the board. (Appendix 1: Rule Book, p. 3)

At the beginning of the game players receive three Developer Briefs, yet players can draw more Briefs in their turn. This can be either because players have completed all their initial Briefs and need additional ones to continue playing—or because players want more diverse options in order to strategize how they will play the remainder of the game.

The front of the Developer Brief contains the name and logo of a specific fictional social software applications inspired by real applications found on mobile app marketplaces, such as Twitter or Facebook or Messenger²⁷. This follows the already stated implications of software studies' digital materialism methodology that frames the game, as well as the codification of the content according to a critical pedagogy approach. Additionally, on the front of the Briefs there are four slots into which four Software Tiles can be placed. These four empty slots on the front are filled through following one of the pre-set combinations of Software Tiles set out on the back of the Developer Brief. These pre-set combinations are made up of four sets of different colour combination relating to the colours of the nine different types of Software Tiles. Each of these combinations have a different 'revenue value' (i.e. amount of game points) a player earns for completing it depending on the specific Tiles played. The 'Cheeps' Developer Brief with its Twitter-esque logo might thus be completed by the addition of a specific 'Newsfeed', 'Notifications', and two other constitutive features (Fig. 3.6).

²⁷ A full list of the Developer Briefs can be found in Appendix 2: Game Database and in Appendix 3: Playable Prototype

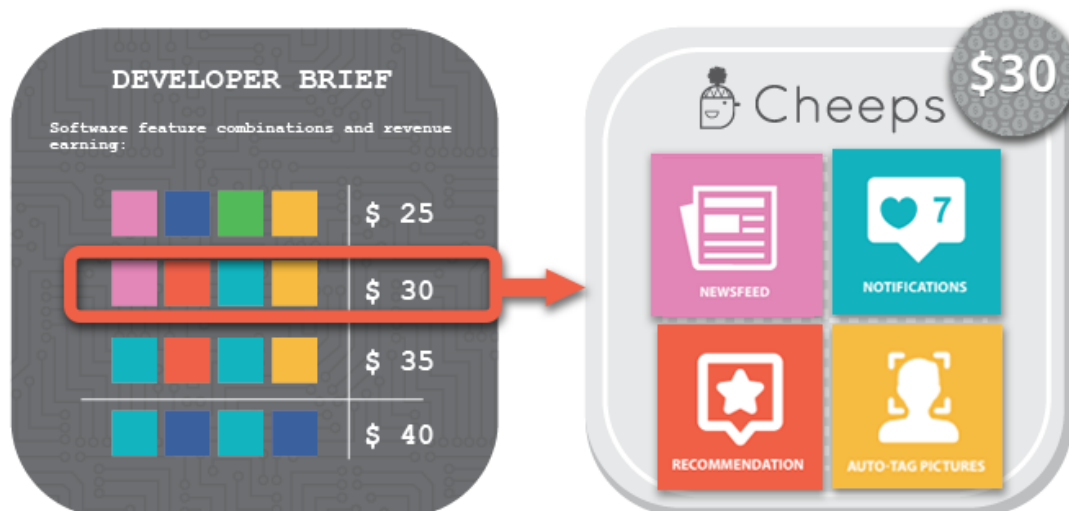


Figure 3.6. **A Developer Brief with a completed Tile colour-combination.** This illustration from the rule book demonstrates how to complete a colour-combination from the back of the Developer Brief, by placing the corresponding Software Tiles on the front and marking it with the correlating Revenue Token (Appendix 2: Rule Book).

While the fact that the Software Tiles necessary to complete a Developer Brief are double-sided, thus reflecting the software critique already discussed, the relation between the Tiles and the Developer Briefs enacts an additional software studies critique as well. This is derived from another of Manovich's initial principles of new media: modularity. Manovich borrows this principle from the modularity that underlay "structured computer programming", writing that structural computer programming "became standard in the 1970s" and "involves writing small and self-sufficient modules (called in different computer languages subroutines, functions, procedures, scripts) which are assembled into larger programs" (2001, p. 52). This also, however, implies that "if a particular module of a computer program is deleted, the program would not run" (ibid.). While the features needed to complete a Developer Brief in *Unveiling Interfaces* are not in any way meant to be totalizing, it is nonetheless based on the premise that no software is a unified monolithic composition, despite the seeming congruity of the final product users interact with via their smooth seamless user-interface²⁸. While we might thus talk about the front of the Software Tiles as 'features' (congruing 'features of a product'), when we flip the Tiles over we are confronted by these features as modular functions of the application. It is in this

²⁸ Here we can perhaps reverse Manovich's analogy, as the author argued that digital media might seem to comprise a unified object, such as an image created with Adobe Photoshop, while in truth it consists of an underlying modularity found in independent overlain layers comprising the final image (2001, p. 52).

that the Tiles hint at the modular interdependence underlying software, as players cannot complete a Developer Brief, and thus publish their apps, without four Software Tiles. For there would be no app without its modular components²⁹.

The Developer Briefs fundamentally function as subgoals in the object of the game, not unlike the 'Destination Ticket' cards in *Ticket to Ride* (2004), or the 'contract' cards in *Master Builder* (2008)³⁰. For as Michael Erhloff Tim Marshall writes: "All games have a win-or-loss condition, which indicates what must be achieved in order to end the game" and the choices a player makes are always "related to the goal of a game, which is often composed of smaller subgoals a player must meet to win the game" (2008, p. 186). How they function as subgoals towards achieving the end of the game is the basis of Action 3 available to players during Phase 1 of their turn.

3.1.3. Action 3: Develop and Publish an App

*After collecting the tiles needed to develop their app a player can publish their app on one of the open slots on the AppMarket on the **playing board**. To do this a player must:*

1. *Check what the revenue value of the **Software Tile** combination you want to complete is. Put a **Revenue Token** of corresponding value aside.*
2. *Place the **Developer Brief** on an open slot on the AppMarket board, with the user interface-side up.*
3. *Place the correct combination of **Software Tiles** in the four (4) slots of the **Developer Brief** with the user-interface-sides up.*
4. *Place the **Revenue Token** in the top-right of the card.*
5. *Collect their initial revenue earnings by adding the amount on the **Token** to the score on their **personal smartphone's** calculator. (Appendix 1: Rule Book, p. 4)*

As both Developer Briefs and Software Tiles have been described already, the only new elements that comes into play during this phase is the Revenue Tokens and the AppMarket board. In Action 3 the Revenue Token serves a double function: firstly, it works as a marker in the board that indicates to which player each app on the board

²⁹ Of course we cannot fully address the question of modularity given our design constraints. For even the modular functions represented by a set of four Tiles are in reality themselves comprised of innumerable modules and submodules.

³⁰ In *Ticket to Ride* players gain points by systematically claiming 'train routes' on the board, as specified in 'Destination tickets' that they draw, while in *Master Builder* players gain points through 'contract' cards that they can complete by building buildings.

belongs; secondly, it communicates the value of the app according to the colour-combination completed from the Developer Brief.

The AppMarket board is the primary playing field of the game, resembling the familiar grid of a smartphone screen that is usually filled with the icons of apps installed on the phone (or in our case, apps available on the AppMarket) (Fig. 3.7). As it was examined in Chapter 2, designer board games need to provide an information-rich environment so that all players can assess the state of the game at any point. Thus, by playing on the AppMarket board, if one player publishes an app on the market all the other players can immediately access this information and strategize accordingly. For example, if there is only one slot left in the board, players might race to publish their app with the Software Tiles they have available. Another type of strategy that can be developed regards the gross profit of the apps in the market. Players have the possibility to decide whether to launch more profitable apps by waiting a turn to get the Software Tiles needed—or to place the app combination with the Tiles they already have and thus get revenue quicker.

The relation between the game elements already discussed and the playing board works according to a game mechanic known as ‘area control’, meaning “players work toward controlling the most area in the playing space” (Mayer & Harris, 2009, p. 113). Mayer and Harris explain that this mechanic is “often used as a criterion for end-game victory conditions or for earning rewards during play” (ibid.). It is in this mechanic that Action 1 and 2 come to fruition as meaningful action within the game during Action 3. For as Zimmerman states, you arrive at identifying your game’s core-mechanic by asking: “[W]hat is the actual activity of the game? What is the player actually doing from moment to moment as he or she plays your game?” (cited in Fullerton et. al., 2004, p. 330). Core-mechanics are “the experiential building blocks of player interactivity, which represent the essential moment-to-moment activity of the player, something that is repeated over and over throughout the game” creating “patterns of behaviour [...] through which players make meaningful choices” (Erlhoff & Marshall, 2008, p. 187).

Following the continuum of the core mechanics of collecting Software Tiles, completing a Developer Brief, and publishing an app to the AppMarket board—the act of placing software tiles with the user-interface side up is crucial to the software critique enacted in Action 3. This is due to the fact that through this, players physically enact the blackboxing of software by user-interfaces (UI), which are

represented by the front-side of the tile. However, before deepening on the blackboxing effect of UI on software, it is necessary to first to define the term “interface” so that the content and attributes of the UI-side of the Software Tiles can be discussed in relation to the core-mechanics, its theoretical justification, and how it relates to our critical play value goals.



*Figure 3.7. **Playing board design with an app published onto it.** The figure also includes the user-interface side up Software Tiles and Revenue Token.*

3.1.3.1. Interfaces: APIs, Code, and GUI

Cramer and Fuller argue that “[s]imilar to both its meaning in chemistry and to the meaning in ‘language,’ ‘interfaces’ are the point of juncture between different bodies, hardware, software, users, and what they connect to or are part of” (2008, p. 150). This definition of the interface is key to comprehend what the authors describe as “condensations of computational power that computers embody and that are differently articulated by individual pieces of software” (ibid.). This “condensations of computational” became a challenging task in the process of design of the board game, firstly because interfaces are present in each layer of this condensation, and at the same time they “link software and hardware to each other and to their human users or other sources of data” (ibid., 149). And secondly, because as the authors explain, those layers “are radically alien to most human experiences of the world” (ibid., p. 150). Thus, to address these challenges, we have drawn on the digital materialist approach of Cramer and Fuller, who offer a five-level typology of interfaces, to design the board game. The first two categories correspond to interfaces directly related to hardware and the last three are related to the software:

1. hardware that connects users to hardware; typically input/output devices such as keyboards or sensors, and feedback devices such as screens or loudspeakers;
2. hardware that connects hardware to hardware; such as network interconnection points and bus systems
3. software, or hardware-embedded logic, that connects hardware to software; the instruction set of a processor or device drivers, for example;
4. specifications and protocols that determine relations between software and software, that is, application programming interfaces (APIs);
5. symbolic handles, which, in conjunction with (a), make software accessible to users; that is, “user interfaces,” often mistaken in media studies for “interface” as a whole. (ibid.)

As it was pointed out in chapter two, throughout the history of computing, hardware interfaces have been subject to blackboxing through software interfaces. Kittler described this process as computation finally getting “rid of hardware itself” (1997, p. 151). Thus, the focus of the board game is located between the last three categories: the software levels. However, the first two types of interfaces are represented by the smartphone (hardware) playing board. The selection of this type of hardware was made taking into consideration that smartphones have become ‘intimate interfaces’, as Søren Pold and Christian Ulrik Andersen refer to them. Moreover, the authors claim that smartphones “are woven closely into all aspects of

daily life” (2014, p. 29). Thus, it can be argued that this technology is relatable to the average user population of digital technologies³¹. Additionally, this selection also responds to one of our value goals in terms of contextualization of algorithms present in everyday cultural software. Thus, the smartphone as the board of the game provides an ideal space to observe and interact with algorithms in action through the game mechanics and design elements.

Kittler writes that “the so-called philosophy of the so-called computer community tends systematically to obscure hardware with software, electronic signifiers with interfaces between formal and everyday languages” (1997, p. 150). What Kittler describes correlates to the third type of interface described by Cramer and Fuller as a “hardware-embedded logic” (2009, p. 150). This layer corresponds to software that controls the hardware—or in Flusser’s terms the apparatus—which entails the instructions to the ‘metal’, and is commonly known as operating system and kernels. Cox writes of this type of software that in order to maintain the “inevitable inaccessibility of the machine itself”, it is necessary to have “specialized expertise” (2010, p. 134). Therefore, he explains that the separation between software (through Graphical User Interfaces) and the apparatus reinforces “the split between technical operations and wider cultural work” (ibid.). This entails that user may have a low barrier of access to the computer, but the structure itself prevents them from using it at “a greater level of operation” (ibid.). As a result, Cox writes that these closed systems mystify the complex processes beneath the Graphical User Interface (GUI), and thus become black boxes.

Correspondingly, this type of software has produced new types of tech-business models that according to Pold and Andersen can be described under Striphas’ principles of controlled consumption. These are summarized as follow:

1. A cybernetic industrial infrastructure integrating and handling production, distribution, exchange, and consumption is developed around the product.
2. The consumption is controlled through programming that closely monitors consumer behavior and the effects of marketing through tracking and surveillance.
3. Controlled obsolescence is programmed into the product, limiting its functionality and its durability.
4. The overall effect of controlled consumption is a significant reorganizing and troubling of specific practices of everyday life. (2014, p. 23)

³¹Pold and Andersen explains that the smartphone is “a new important cultural software interface that is already revolutionizing cultural production, distribution and consumption” (ibid., p. 31).

The scenario proposed by controlled consumption's model is clearly related to the one previously outlined in Flusser's post-industrial context, and it is used in the work of Pold and Anderson to describe the current business practices of many digital devices on the market, such as Amazon's Kindle, Apple's iPhone or Microsoft's Xbox Live. For the authors, these type of tech firms are carrying out "the tightest implementation of this scheme with their almost complete control over the integration of hardware, software and distribution" (2014, p. 24). In the context of the board game, controlled consumption is represented by the AppMarket, where the players need to publish their developed apps.

Additionally, it can be argued that it is because of these constrained environments that the "app-based interface" model has emerged in the first place. For as Pold & Andersen posit, the "app-based interface is clearly developed to support an effective and lucrative distribution and business model for the digital cultural content" (ibid., p. 18). Moreover, the authors also propose that app-based interface "opens up for a very particular business model for cultural software" (ibid., p. 26), as it standardizes an "object-oriented cultural model in the sense that consumers are forced to adapt to a specific and rather passive model of consumption framed by the licenses and the technology" (ibid.). Thus, it is precisely this business model that the players simulate by 'developing' and 'publishing' apps in the fictional AppMarket. Likewise, this action can be read as a way to disrupt the "passive model of consumption" (ibid.) that technologies such as smartphones foment.

It follows then, that the fourth layer of Cramer and Fuller's typology relates to APIs (Application Programming Interfaces) and is where the code-side of the software tiles is located. The authors elucidate that APIs have become "increasingly important to the development of network that rely on data and software working without being constrained by hardware platform" (2008, p. 151). Thus, it can be inferred that as the hardware has been obscured by the third interface, it is in the layer that corresponds to APIs where the discussions of algorithmic culture are focussed³²—given that app developers, in general, have more control over the data and networks generated from the interaction between user's smartphones and apps.

³² As an illustration, recent events involving the Cambridge Analytica scandal are centred in this interface, due to the fact that Facebook's user database was deceptively accessed through APIs, and

In like manner, in terms of the controlled consumption business models, APIs also emerge as a control mechanism. As Striphas sketched out in the first principle, there is a vertical integration of the “production, distribution, exchange and consumption around the product” (as cited in Pold & Andersen 2014, p. 23). In the same note, Cramer and Fuller, identify a “asymmetry of powers” which is “mapped and sieved through interfaces in other ways” (2008, p. 151). The authors suggest that this asymmetry can also be held through APIs, as they institute “protocols that operate as interfaces between computers linked over a network, [and] also establish descriptions of operations that are allowed and assigned a priority or blocked” (ibid.). Thereby, for developers this implicates a constrained space to design an app and ‘use’ of the hardware.

Equally important is the approach to the APIs’ materiality: the code. As Kitchin & Dodge assert, “layers of software are executed on various of hardware [...] using various algorithms, languages, capta rules, and communication protocols. [...] At the heart of this assemblage is code—the executable pattern of instructions” (2011, p. 24). For Berry this involves that code is also written through modular systems, thus he posits that “software is written using other software packages,” helped by “software support programs and modular mass-produced libraries of code” (2011, pp. 36-37). These packages are often designated as Software Development Kit (SDK), which comprise a set of APIs that developers can use to create products for specific platforms³³. It can therefore be assumed that if APIs are the interface that enables software to connect to other software, they are themselves a form of mediation; or as Berry writes: programming editing software is “mediating the relationship with code and the writing and running of it” (ibid., p. 37).

Lastly, the fifth layer in Fuller & Crammer’s typology corresponds to user interfaces, which Cramer and Fuller claim are frequently mistaken for “‘interface’ as a whole” (2008, p. 149). The authors outline this interface as “symbolic handles that make software accessible to users” (ibid.), such as the GUI, Tangible User Interfaces (TUI), and Voice User Interfaces (VUI), among others. Likewise, these types of interfaces operate under the paradigm of “user-friendliness”, and it is due to this characteristic that they are often the “most easily recognizable and visible” (ibid., p.

the profile information was used for targeted political ads without user’s authorization (Madrigal, 2018 March 18).

³³ For instance, Google’s Android offers the APK (Android Application Package), and Apple’s iOS the IPA (iPhone Application Archive).

151) of all the layers discussed above. Similarly, Pold argues that “the purpose of the [user] interface is to represent the data, the dataflow, and data structures of the computer to the human senses, while simultaneously setting up a frame for human input and interaction and translating this input back into the machine” (2005, n.p.). In the board game, the GUI is the interface that has been represented through UI-side of the software tiles, which are placed in the board when a player publishes an app, so that the code-side is concealed. And thus, the name of the game, *Unveiling Interfaces*, finds a subject to unveil: the user-interface.

Nonetheless, Cramer and Fuller argue that the differentiation between the software layers is “purely arbitrary” (2008, p. 150). Simply put, the “more complex interfaces to computer functions tend to be called ‘programming languages’ and less complex, more specialized ones are known as ‘user interfaces’” (ibid.). Furthermore, the authors explain that “since the user interface to a computer program is always symbolic, and involves syntactical and symbolic mappings for operations, it always boils down to being a formal language” (ibid.). Conversely, Berry’s layers of computational which draws on the Bashkar et. al (2010) ‘laminated system’ approach, separates “ontological levels [that] work according to both different logics and different mechanisms” (2014, p. 71). The author sketches the following ontological layers as follow:

1. Physical: Material and transactional level (of the hardware)
2. Logical: Logical, network and informational transactional level (level of software as diagram or platform.
3. Codal: Textual and coding logics (level of code as text and/or process)
4. Interactional: Surface/interface level (between human beings and non-humans mediated through code)
5. Logistics: Social and organizational structure (at the level of institutions, economies, culture, etc.)
6. Individuational: Stratification of embodied personality (the psychology of actors, the user, etc.) (Berry, 2014, p. 58)

In this way Berry’s ontological layers map each level with its own ontology, which “serves to ‘make sense’ at whichever level of analysis” (ibid., p. 57). For the author, these strata of ontologies “can be used to critically approach and situate our knowledge in relation to each other and provide some means of orientation in software/code’s obvious multidimensional ontology” (ibid., p. 59). Furthermore, Berry suggests that the layers of the computational share common ontology, where “computational principles that are repeated and generated at different scales through

these computational laminated systems” (ibid., p. 61). By proposing this shared ontology, Berry intends to prevent “some notion of irreduction, [that] otherwise higher strata could, in theory, be ‘better’ studied, or explained by lower strata levels” (ibid., p. 59).

Similar to Cramer and Fuller’s interface typology, Berry considers three layers on the division of digital devices: hardware platform, software platform, and software application/interface layer. For the author it is helpful to draw these distinctions as it funnels the attention to a specific layer where the researcher can focus, without “losing sight of the importance of the supporting hardware and software, in other words, the forces of production” (ibid., p. 57). Therefore, it can be argued that in each of the computational layers there are cascading of *blackboxing* mechanisms, each atop the previous, which ends in the UI. It becomes then necessary to mention that this thesis is focussed on the software application (API) and UI, and has the following ontological layers: codal, interactional, and logistics. The first layer has been addressed in the analysis, in the previous section, of the code-side of the Software Tiles. The interactional layer aims to be problematized in this section, which will further the analysis of the UI-side of the tiles. This has been narrowed down to GUI, because is the type of interface that corresponds the selected hardware of the board game: a smartphone. Finally, the logistics layer will be discussed in Phase 2 below.

3.1.3.2. Software Tile: User Interface Side

To further the analysis of the interactional layer we turn to Berry’s understanding of software ecologies. For the author these ecologies are distributed throughout the layers of the computational, and enable “access to certain forms of mediated engagement with the world” (2014, p. 68). The mediation is achieved through a ‘translucent surface interface’, which in the typology of Cramer and Fuller would be translated as the UI. Thus, the ‘surface interface’ is located in the interactional layer, and it “enables a machinery to be engaged which computationally interoperates with the world” (ibid.). At this point, it is useful to recall Berry’s approach to software as compactants, where software applications have two faces: ‘commodity’ and ‘mechanism’. Berry explains that compactants “are often constructed in such a way that they can be understood as having a dichotomous modality of data collection/visualization, each of which is a specific mode of operation” (ibid.). The

author explains that the modal setting of compactants can be accessed by the user or, in other cases, hidden functions may be “accessible only to certain people” (ibid.) such as web administrators or coders³⁴.

As a result, Berry proposes that compactants “passive-aggressively record data” (ibid.). He specifically uses those adjectives to describe data as ‘passively’ gathered under the surface on the one hand, and on the other hand, how that data is ‘aggressively’ hoarded. As it was explained in earlier in the framework of the game, the interface/surface layer corresponds to the commodity function as it offers a stable layer for “consumption of ends” (ibid., p. 69). Whereas the codal layer is related to the mechanism, as for Berry the ‘mechanism face’ can “be thought of as the substructure for the overlay of commodities and consumption” (ibid.). This analytical model for thinking of software as a dual composition has informed the attributes of the Software Tiles in the game, as they are also composed of a surface (UI) and a codal layer (APIs and code). Moreover, it has also informed the core mechanics of the game which involves the code-side (mechanism) of the Software Tile being concealed by the UI-side (commodity) in the act of publishing an app. As Berry posits: “the complexity of the machinery of code is obscured by its interface, the commodity, which is often only loosely coupled to the underlying logic and therefore to the control of the system under use” (ibid.).

An illustration of this can be seen in Andrew-Gee from *The Globe and Mail* in a recent article in which he reported on the rumoured fact that Instagram withholds 'like notifications' from its users so that the usage of the app increases. As Mayberry, a worker at a Californian start-up expressed in an interview with Andrew-Gee: “it's common knowledge in the industry that Instagram exploits this craving [for positive feedback] by strategically withholding ‘likes’ from certain users” (2018 April 10, n.p.)³⁵. Not dissimilarly, Facebook’s founding president Sean Parker made the claim that he and the other early Facebook employees built the platform to “consume as much of your time and conscious attention as possible” (Kircher, 2017 November 9,

³⁴ For example, with newsfeed features in social network sites users only have access to the information that is already filtered, whereas the developers and proprietors of the platform have access to the operations behind the feature—such as the logic expressed in code on how to filter news.

³⁵ It should be noted that Instagram denies these claims. Chief Technology Officer of Instagram Mike Krieger stated that “replication lag/etc. may mean things aren't instantaneous but not intentionally so. And notifications we try and strike a balance of being timely + not over-sending notifications” (cited in Andrew-Gee, 2018 April 10, n.p.).

n.p.). These practices can be understood by applying compactants behaviour, as described by Berry, through which the commodity layer offers services like photo-sharing or social blogging while on the mechanism side, underlying business practices are enforced that sustain the company's profitability. These practices make use of the gathered database and develop business models such as paid content advertisement or business intelligence consultancy.

By the same token, Pold and Andersen suggest that UI—or as they call them 'cultural interfaces'—"risks, at worst, to turn cultural software into perfect consumer object and use it as bait for increasing control and surveillance" (2014, p. 31). This correlates with Berry's software faces, given that software as a consumer object can be understood as a cover for the underlying mechanisms that enable software to work and may also have other purposes than the ones presented to the user. In this way, the data gathered from the user's behaviour and personal information can potentially be deployed as a source of control, such as affecting the frequency with which their users access the app.

Given that the Software Tiles' content³⁶ was designed taking Berry's compactants approach into account, the commodity mechanism in the game is portrayed by 'feature icons' (Fig. 3.8.). The choice of features answers to our value goals, specifically the one that aims to develop players' awareness of algorithms present in their daily lives. Moreover, this value goal is also supported by the aesthetics of icons, which follows the trend that both Google and Apple have embraced in UI design named 'flat design' (Beecher, 2010). Thus, the flat design also offers a familiar aesthetics to the player. Moreover, Berry reflects that this type of design corresponds to a movement towards 'simple' or 'obvious' design principles. The author further emphasizes that 'flat design' is "a visual aesthetics and method that prioritizes a notion of a priori geons that structure the interface through a semi-platonic ideal drawn from geometric principles" (2014, p. 70).

Additionally, Berry also reflects on how 'flat design', by being "extremely visually pleasing", "enable[s] the machinery level of the codal object to be hidden away more successfully" (2014, p. 70). In other words, Berry refers to the action of blackboxing of the code through the UI design (the commodity face).

³⁶ The full list of Software Tiles can be seen in Appendix 2: Game Database and Appendix 3: Playable Prototype.

Correspondingly, in the game, the UI-side of the Software Tiles must be displayed on the board once a player has published an app, making only the ‘commodity face’ of their app visible to other players. Thus, the purpose of the UI-side is to veil the ‘mechanism face’ of the code-side. Likewise, this also creates a simulation of the current state of code and APIs which are blackboxed by user-friendly interfaces.



Figure 3.8. Software Tile's UI side.

Similar to the attributes of the code-side, each feature has two colours. However, unlike in the ‘mechanism face’, where the code undergoes constant changes, the surface side remains identical. Thereby, the UI-side as a stable icon hides both responsible code as well as the problematic code, which will be discussed in below in Phase 2. The double-faced Tile mechanism in the game thus responds to another value goal, as it conveys the technical aspect of a compactant, which can contribute to a technical appreciation that is not contingent on technical knowledge.

3.1.3.3. User Interface: Blackboxing as Commodity

The rise of the user-friendly software (and their GUIs) is, according to Manovich, a product of the work by the “pioneers” of ‘cultural computing’ (2013, p. 102). The

author identifies the Xerox PARC research group as the epicentre from where the computer was envisioned to become “a cultural medium—rather than merely a versatile machine” (ibid.). Thus, ‘cultural computing’ is defined in contrast to the non-cultural use of the computer's design before the 1960s. Underlying this, as Manovich states, is that “what differentiates a modern digital computer from any other machine—including industrial media machines for capturing and playing media—is separation of hardware and software” (ibid., p. 92). This correlates with the analysis of interfaces and the blackboxing of hardware through software, as has been discussed above. For Manovich it was not only the conceptual work of the inventors of cultural computing that shaped the software evolution but also the fact that “various social and economic factors—such as the dominance of the media software market by a handful of companies or the wide adoption of particular file formats—also constrain possible directions of software evolution” (ibid., p. 93). This connects back to the formation of software oligopolies and the logic of late capitalism as also affecting the development of cultural computing.

As a result, Manovich states that “during one decade the computer moved from being a culturally invisible technology to being the new engine of culture,” with GUI-based software putting the computer “at the centre of culture” (ibid., p. 21). Hence, as “software with [...] GUI aimed at non-technical users” (ibid.) started to be circulated in the market, so too did the paradigm of a user-friendly interfaces become established. It is within this context that Cox reflects on the case of Apple Macintosh, in which he states the aim of the first operating system with GUI, “was to make ‘universal’ graphic user interface, to set a standardized way of operating a computer that enabled the relatively ‘unskilled’ user to gain access to computers” (2010, p. 134). Similarly, for Cramer and Fuller, the traditional understanding of user-friendliness was to “place the user as its subject, and the computational patterns and elements initiated, used and manipulated by the user as the corresponding grammatical objects” (2008, pp. 151-152). Thus, the early conventions of GUI, as Pold argues, can be summarized as WIMP (Windows Icons Menus and Pointers), with these objects circumscribing discrete grammars of actions for its users (2005, n.p.). As Cox suggests, “much of the commercial software appears to be designed to predetermine its use and deny the user autonomy over their work” (2010, p. 34).

Furthermore, Manovich points out that when ‘user-friendly’ GUIs “became the commercially successful paradigm following the success of Apple’s Mac computers,

introduced in 1984, the intellectual origins of GUI were forgotten” (2013, p. 101). The author writes:

Looking at the theories of Kay and Goldberg that were behind GUI design gives a very different way of understanding an interface’s identity. Kay and his colleagues at PARC have conceived GUI as a medium designed in its every detail to facilitate learning, discovery, and creativity (ibid., p. 100).

Manovich contends that human-computer interaction (HCI) experts and designers “continue to believe that the ideal human-computer interface should be invisible and get out of the way to let users do their work” (ibid.). Therefore, the conceptualization of GUI was designed to enable understanding of the algorithmic systems, which differs from the HCI premises. This stands in relation to what Pold calls an engineering tradition, which was “aimed to increase the ‘user-friendliness’ and ‘transparency’ of the interface and over the years has involved cognitive sciences, psychology, ethnographic fieldwork, participatory design, etc.” (2005, n.p.).

Put differently, the software layer of seamless user-friendly interactions created by the rise of cultural computing reflects what Berry refers to as the ‘commodity face’ of compactants. The author describes the action of GUIs as a “convincing narrative [...] [where] software presents a translucent interface that is relative to the common ‘world’” (2014, p. 62). The ‘convincing narrative’, Berry explains, was achieved by providing the user a familiar or skeuomorphic interactional surface. This surface entails a “representational, metonymic, flat, figurative or extremely simplistic and domestic” surface (ibid., p. 63). Thus, the skeuomorphic design, as part of user-friendly interfaces, works on the principle of emulating familiar real-world materials to comfort users in the introduction of new user-interfaces for computers. For instance, in the case of the smartphone, the icon of a telephone receiver—an analogue medium—is used to represent the app that makes calls. While the skeuomorphic ‘commodity face’ communicates a stability of the software, constant changes are simultaneously occurring in the mechanism face of the compactants, which is composed of the codal, logical and physical layers of the software. Berry suggests that software “possesses an opaque machinery that mediates engagement that is not experienced directly nor through social mediations” (ibid., p. 62-3). Thus, it can be inferred that the commodity layer not only obscures the codal mechanism behind the GUI, but also the logistic layer that works under those mechanisms.

Regarding the obscuration of the codal layer—the mechanism face of the compactants—Kitchin and Dodge state that code is always “hidden, invisible inside the machine” and it is opaquely sealed and packaged from the engineer’s desktop for the real world as the “prescriptive code” of consumer software applications (2011, pp. 3-4). Furthermore, Berry posits that the complexity of the codal layer is “often only loosely coupled to the underlying logic and therefore to the control of the system under use” (2014, p. 69). Thus, at the moment the surface layer becomes the interface to interact with the machinery, the interaction affords an ‘asymmetry of power’ to occur. This, as Fuller and Cramer states, become expressed in different ways:

[B]y the use of text; visual-spatial structuring devices such as a window and its subcomponents, timeline or button; sounds, such as system event sounds; animated representations of running data-processes such as a “loading” bar, “throbbers” (used in web browsers), spinning cursors; widgets; menus, which describe available functions; and other elements. (2008, p. 152)

Two types of asymmetrical power relations created by UI can be identified: on the one hand, as Kitchin and Dodge suggest, these type of interface elements allow access to the user to data and data structures from the software; and on the other hand, UIs also enable the passive-aggressive recording of data. Thus, in both cases, the asymmetry of power is what becomes ideologically commodified, as the UI remains relatively stable and becomes a mean of consumption for users.

It follows that in the logistics layer, social and organizational structures are also obscured by the UI. Ed Finn, in his recent publication *What Algorithms Want* (2017), claims that the “triumph of gamification, ubiquitous computing, and remote sensing [...] has led to a slew of new businesses that add an algorithmic layer over previously stable cultural spaces” (p. 124). In other words, Finn is referring to the spreading of different types of interfaces, including GUIs, over a myriad of social activities. Thereby, a “computational layer of abstraction” (ibid.) mediates the interactions that were before personal interactions, such as hiring a personal assistant through a job-listing web-portal or taking a ‘taxi’ through a ride-sharing app such as Uber. This appreciation of how software mediates social and economical activities through compactants demonstrates the effects and consequences of algorithms and software in the real world, which also reflects the need for a public algorithmic literacy. For as Berry proposes, it is in the “specific modular organization

and deployment of the 'digital' in both material and ideological moments, which needs to be considered carefully" (Berry, 2014, p. 62).

Pold's analysis of 'buy button' functionalities on retailer websites such as Amazon serves as one of the simplest most yet significant examples of how commodity faces obscure the logistics layer of software (2008). The author explains how a 'buy button' enables a user—through a single click—to perform an economic action, while the "cultural, conventional, and representation elements" of that action "are disguised or 'black-boxed' as a pure technical functionality" (2008, p. 33). Similar to this, Finn suggests that in an "interface economy" the value chain is obscured by UI. He emphasizes that the "socio-economic infrastructure gets swept away behind the simple software interfaces" (2017, p. 124). Moreover, Finn analyses the blackboxing of Uber's mechanisms, an app who connects drivers with riders. He explains how the UI "provides certain forms of certainty in terms of immediate time and distance" (ibid., p. 126). Nonetheless, the calculation of the fares is not visible, nor the legal bound or labour conditions between drivers and the tech-company. The rating system is also mediated through the app which becomes the central mediator in this economic activity. Finn writes: "the real sell is the interface itself: the experience of computationally mediated culture and its underlying algorithmic simplification and abstraction" (ibid., p. 129). In this example, it can be observed that the surface layer remains stable, while the underlying mechanism has numerous changes such as the pricing system, the regulations, and the rating system ponderation. Therefore, as the interface is used as a commodity—services and features—the underlying mechanisms for how the app works as well as the business model underlying it is hidden from the user's experience.

This argument that UIs hide more than they display is codified into the frame of the game and its correlating objective through which players interact with a simulated software marketplace. Importantly, players garner more revenue for their published apps in the game if they employ certain Software Tiles that, even though they might seem similar to others on the UI-side, employ algorithmic mechanisms on the code-side that prioritizes profitability for the producer. For example: a turquoise 'Notification' feature containing the encoded algorithm "Displays new [likes, shares, comments] gradually distributed over time" on its code-side will be part of a more profitable colour-combination on the Developer Brief than one containing a pink 'Notification' feature with its algorithm encoded as "Displays new [likes, shares,

comments] instantly and automatically” (Appendix 2: Game Database). Though these two features, based on the Instagram case study already discussed, are coded according to two different business models—on the UI-side they represent the same feature. Yet it is the blackboxed code that determines not only the profitability of the published app, but also the material effects it has in the world, as will be explored in Phase 2 below.

3.2. Phase 2: Event Cards OR Collecting Revenue

Once a player has completed one of the Actions above, they have to move the Week Marker by one space on the calendar on the playing board. Depending on which space the Marker lands, one of two Events play out in the game:

- *EVENT 1: Get a notification to ‘check your code’*
- *EVENT 2: Collect revenue. (Appendix 1: Rule Book, p. 5)*

The second Phase of a player's turn starts with moving the Week Marker on the Calendar. There are four positions on the calendar, signifying four weeks in a month. Each player's turn thus signifies one week. If there are at least two apps published on the board, and the marker on the Calendar lands on a week with an Event Notification icon (which constitutes the first three weeks on the calendar) (Fig. 3.9.), then players have to draw an Event Notification (Event) Card. This will mean that players will most likely have to ‘check’ the code of their published apps, which might have a number of different consequences for the players. Following Salen and Zimmerman's division of game mechanics, if Phase 1 of a player's turn signifies the actions or choices they have to make, Phase 2 signifies the consequence or outcomes of those actions (2004, p. 34).

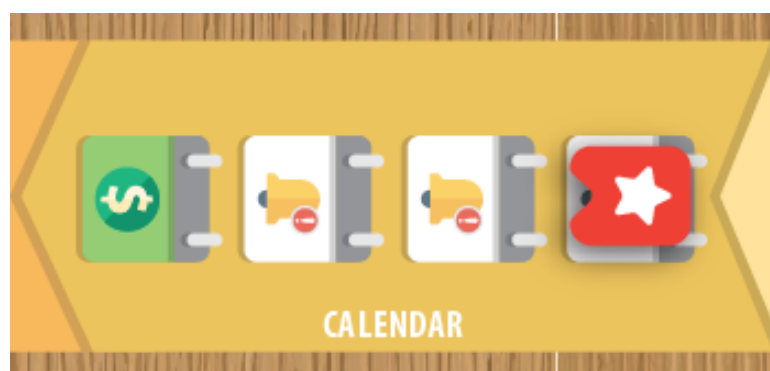


Figure 3.9. **Week Marker on the board's Calendar.** The figure displays the Week Marker placed on the first 'week' on the calendar found on the right-side of the playing board.

3.2.1. Event 1: Get a notification to 'check your code'

Event Cards are available on the board in a deck placed face down on the board and is played by the player whose turn it is in the following way:

1. *Pick the top card from the **Event Notification Card** pile.*
2. *State out loud which software feature is indicated on the top of the Event Notification Card.*
3. *ALL PLAYERS who have the software feature mentioned on the Event Notification Card on the user-interface-side of any of the Software Tiles on their published apps then need to 'check their code' because a feature of their apps have been implicated in an event. 'Checking your code' entails picking up the implicated Software Tile from the published app and turning it around to read its code-side.*
4. *The player whose turn it is then reads the rest of the Event Notification Card while all players 'checking their code' listen whether the algorithm mentioned in the Event Notification Card corresponds to the algorithm written on the code-side of the Software Tile they are 'checking'.*
5. *If the code on the **code-side** of the implicated **Software Tile** corresponds to the code mentioned in the Event Card, that player must do the action that is set out in the Event Card.*
6. *If the Event Card has a 'expose code' icon, then the offending Software Tile has been implicated in a controversy and needs to remain with the **code-side** up to insure transparency of your app. If not, flip the Tile back to the user-interface side up.*
7. *If one of the apps has two Software Tiles with the code exposed, the app must be removed from the AppMarket and the Developer Brief and Software Tiles need to be discarded (Appendix 1: Rule Book, p. 5).*

The primary elements of the Event Cards are divided graphically into two blocks: the first contains a software feature, a headline and description; and the second contains a consequence. Each Event Card has a direct effect on specific Software Tiles, indicated by the software feature icon and name at the top of the Event Card which corresponds to the features displayed on the UI-side of certain Software Tiles. All players with the implicated features on any of their published apps have to then 'check their code', as stated on the top of the Event Card, by flipping over the implicated Software Tiles to reveal their code-sides. The content of the

Event Cards implicate certain possible algorithms of the stated software feature, indicated by highlighted keywords (Fig. 3.10.).



Figure 3.10. **An example of an Event Notification Card.** This card will apply to all apps with a 'Social Login' Software Tiles with the encoded algorithm "Allow users to auto login to other apps by enabling other services to access [profile private information]" (Appendix 2: Game Database).

Event Cards such as these are a standard convention in not only designer games, but board games in general. They are present in popular titles such as *Pandemic* (2008), *1960: The Making of the President* (2007), *The Game of Life* (1960), and most notably in *Monopoly*³⁷. In these games 'event cards' function as a mechanic to obstruct or expediate a player's struggle towards the object of the game. In *Unveiling Interfaces* Event Cards likewise create struggle in a way that enhances the gameplay. As Salen and Zimmerman write: "Goals in a game are never easy to achieve. As players struggle toward the goal, conflict arises. Game conflict provides both opportunity for narrative events and a narrative context that frames the obstacles a player must overcome" (2004, p. 387). In this the Event Cards also offer space for the software critique to be encoded into the game and work towards our value goals. For importantly, the Event Cards contain certain unforeseen social, political, economic, and personal events that occur within the

³⁷ *Monopoly's* 'Chance' and 'Community Chest' cards are perhaps the most well-known example of 'event cards'. With statement such as "You have been elected Chairman of the Board—Pay each player \$50", these cards might help or inhibit players from accomplishing game goals (Monopoly Guide, 2017, n.p.).

‘play world’ of the game as a result of certain Software Tiles having been played. Within the critical play goals of *Unveiling Interfaces*, the Event Cards thus mechanically and thematically actualize the argument that algorithms as materialized in software are never neutral but are always socio-technical assemblages that, vested with ideological interests, exert certain problematic material effects in the world.

3.2.1.1. Software as Socio-technical Assemblage

Kitchin writes that algorithms “cannot be divorced from the conditions under which they are developed and deployed” as they “need to be understood as relational, contingent, contextual in nature, framed within the wider context of their socio-technical assemblage” (2017, pp. 17-18). This term ‘assemblage’ is extremely useful here as it can refer to both something being constituted by a “collection or gathering of things”, or the “action of gathering or fitting things together” (Assemblage, n.d.). In terms of the latter, Kitchin writes that while computer science “seek to maintain a high degree of mechanical objectivity—being distant, detached and impartial in how they work and thus acting independent of local customs, culture, knowledge and context” yet “in the process of translating a task or process or calculation into an algorithm they can never fully escape these” (ibid., p. 17). And so, in the act of assembly, algorithms become socio-technical assemblages.

This becomes even more the case as algorithms are translated into material software through the laborious process of programming or code writing. While this process is “often portrayed as technical, benign and commonsensical” in reality this translation process from algorithm, to pseudo-code, to source-code, involve extraneous practices such as “researching the concept, selecting and cleaning data, tuning parameters, selling the idea and product, building coding teams, raising finance” —all of which are “framed by systems of thought and forms of knowledge, modes of political economy, organisational and institutional cultures and politics, governmentalities and legalities, subjectivities and communities” (2017, pp. 17, 18). As Berry writes, “[c]ode is labour crystallised in a software form” and can thus be seen as a “repository of social norms, values, patterns and processes” arising from the historical context of that labour (2011, pp. 35, 39).

3.2.1.2. Grammars of Action

The relevance of the socio-technical assemblage is not merely that algorithms take on certain non-technical commitments through the process of software production, but that software in turn enact these commitments in the world. It is through the dual nature of code that algorithms, materialized in software, acts in the world. For importantly, source code is not merely textual, but processual as well—something always “in process” as it “executes or ‘runs’ on a computer” (2011, p. 29). Berry consequently writes:

Code is processual, and keeping in mind its execution and agentic form is crucial to understanding the way in which it is able to both structure the world and continue to act upon it. Understanding code requires a continued sensitivity to its changing flow through the hardware of the technology. Indeed, this is as important as placing code within its social and technical milieu or paying attention to the historical genealogy. (ibid., 37-8)

The algorithms textually represented on the code-side of the Software Tiles, written with Python notation, thus only reflects one dimension of code as a constituency of software. It is its processualism as software that comes to light through the Event Cards being played, as these come into play through the symbolic (the Week Marker’s movement on the calendar) and actual (the duration of the game) passage of time. As the game progresses, players are informed that some of the algorithms materialized in their apps have been acting in the world with certain agentic consequence. For as Cramer writes: “computer languages become performative only through the social impact of the processes they trigger, especially when their output aren’t critically checked” (2008, p. 170). For instance, certain ‘Auto-tag Picture’ Software Tiles are implicated when the following Event Card that comes into play: “Female doctors are being labelled as ‘nurses’ in all your photos! Your app tags images using an image recognition toolkit that was trained on a single dataset which unfortunately contained only men tagged as ‘doctors’” (Appendix 2: Game Database).

As in the example above, the social impacts of software relate back to the digital materialist equation cited earlier in the chapter that algorithms and data- or capta-structures are interdependent components. As Kitchin and Dodge write, software “abstracts the world into defined, stable ontologies of capta and sequences

of commands that define relations between capta and details how that capta should be processed” through which “[c]ode performs a set of operations on capta to enact an event, an output of some kind” (2011, p. 27). It is through the algorithmic processing of data that software abstracts its material context—and then enacts those abstractions back onto this material context, creating a system of “[a]bstraction operating on abstraction” (Fuller & Goffey, 2012, p. 78). Fuller and Goffey write that,

While such processes can be captured more or less readily in the notational formatting of algorithms and data structures, the realities on which those algorithms and data structures operate must themselves be organized so that they interface smoothly with them. Simulating real-world processes is made considerably easier if the real world already operates like a machine, with a precisely specifiable set of degrees of freedom accorded to the processes in question. (ibid.)

It is through these processes of iterative abstraction that software creates ‘grammars of action’: “highly formalized set of rules that ensures that a particular task is undertaken in a particular way, given certain criteria and inputs” (Kitchin & Dodge, 2011, p. 87). According to Philip Agre, who coined this term in relation to surveillance systems, ‘grammars of action’ necessarily structures the activities of those that find themselves within a particular system as “people engaged in captured activity can engage in an infinite variety of sequences of action, provided these sequences are composed of the unitary elements and means of combination are prescribed by the grammar of action” (cited in Kitchin & Dodge, 2011, p. 90). Within a software culture it is thus software systems that create grammars of action, as users’ behaviours are regulated by the actions and protocols the mediating software affords. As Fuller and Goffey write once more: “The development of thinking about data and data structures, about abstraction and abstraction mechanisms in computing science, offers precious indicators for the well-crafted development of control”, while the emergence of software culture also created imperatives to “either refashion yourself to meet the imperious demands of more and different types of data, or consign yourself to data oblivion” (2012, pp. 85-86).

Yet software does not only contain and constrain our actions, but also our worldviews. As Berry proposes in *Philosophy of Software*, “to understand and explore the ways in which code is able to structure experience in concrete ways” software can be approached through a “phenomenology of computation” (2011, p. 39). Paul Dourish likewise contends that software is “philosophical in the way it

represents the world, in the way it creates and manipulates models of reality, of people, of action” (cited in Kitchin & Dodge, 2011, p. 23). Additionally, Fuller argues that software can be understood as “a form of digital subjectivity [...] that each piece of software constructs ways of seeing, knowing, and doing in the world that at once contain a model of that part of the world it ostensibly pertains to and that also shape it every time it is used” (cited in Kitchin & Dodge, 2011, p. 27).

3.2.1.3. Ideology and Technocracies

The ontologies created by software are importantly “manifestation of a system of thought”, influenced by “finance, politics, legal codes and regulations, materialities and infrastructures, institutions, inter-personal relations [...] all kinds of decisions, politics, ideology” (Kitchin, 2017, p. 17). Consequently, as noted by Mackenzie, software often possesses ‘secondary agency’, as it “supports or extends the agency of others such as programmers, individual users, corporations, and governments” and through that “enables the desires and designs of absent actors for the benefit of other parties” (Kitchin & Dodge, 2011, p. 27). It is here that we turn, according to our methodological orientation of software studies as a critical theory of software, from a ‘phenomenology of computation’ rather to what Berry calls in his subsequent book *Critical Theory and the Digital*: “computational ideology” (2014, p. 66). For as Berry writes: “Any study of computer code has to acknowledge that the performativity of software is in some way linked to its location in a capitalist economy” (ibid.). Within this context software becomes not merely a product of technocratic ideology, but also computationally recreate this ideology, with software culture in Berry’s words thus “[legitimizing] a new accumulation regime” (Berry, 2014, p.5).

As consequent ideological actants in the world, software creates ways in which “human subjectivity can be observed, nudged and managed through certain technologies” that reinforces this accumulation regime (ibid., p. 64)³⁸. While this has already been explored in terms of what Striplhas calls ‘controlled consumption’ model, Berry moreover demonstrates how such a computational ideology can be found in telecommunication technology’s as it transitioned from wired ‘electric’ telephones to digital ‘smart’ devices:

³⁸ There is even a field within computer science that explicitly works towards this end, called ‘persuasive technology’ or ‘captology’ (Berry, 2014, p. 64).

Our phones become smart phones, and as such become media devices that can also be used to identify, monitor and control our actions and behaviour through anticipatory computing. While seemingly freeing us from the constraints of the old wired-line world of the immobile telephone, we are also increasingly enclosed within an algorithmic cage that attempts to surround us with contextual advertizing and behavioural nudges [...] Indeed, as Steiner (2013) argues, a lot of money is now poured into the algorithms that monitor our every move on the social media sites that have become so extraordinarily popular. (ibid., p. 6)

Berry proceeds to quote Steiner:

[Facebook] built the tools and the algorithms that still monitor the unimaginable amount of data pouring into Facebook every hour of every day. Part of the reason that Facebook has proven so “sticky” and irresistible to Web surfers is because [Facebook] built systems to track people’s mouse clicks, where their cursor stray, and what page arrangements hook the largest number of people for the longest amount of time. All of this click, eyeball, and cursor data gets strained, sifted and examined ... Having a nearly captive audience of billions makes it all the easier, and lucrative, to sell ads that can be targeted by sex, income, geography, and more. (cited in Berry, 2014, p. 6)

It is reflecting this critique that we codified the frame of our game, as players have to produce profitable apps by making certain choices that reflects this computational ideology. For as Berry’s thesis states, as “computer code is manufactured” it should “points us towards the importance of a political economy of software” (2014, p. 39). Yet it is within the relation between the Software Tiles and the Event Cards, forcing players to unveil their UIs, that this ideological codification is revealed to players within the game.

3.2.1.4. Unveiling Interfaces Through ‘Checking Your Code’

In having to ‘check their code’ after drawing an Event Card and reading the code-side of the Software Tiles they have published as an app, players are confronted with the possibility that the production choices they had made to stay profitable within the game might have produced unintended consequences in the ‘real world’. As Kitchin and Dodge write: “although code in general is hidden, invisible inside the machine, it produces visible and tangible effects in the world” (Kitchin & Dodge 2011, pp. 3-4). Having to ‘check your code’ when an Event Card is played consequently enacts the central theme of the game: unveiling the UI and reflecting on the blackboxed algorithms working in the world through software.

This reflects Berry's call for "attentiveness to the layers of software beneath this surface interface" needed in order to prevent a 'screen essentialism' created by the intuitive and user-friendly interface design paradigms already discussed in Action 3 (2014, p. 63). For Berry, compactants are an optimal analytical model that responds to this call of attentiveness, as it "draws attention to a source of stability in the computational society" (ibid., p. 70), namely the 'commodity face', and reveals how "computationally has increasingly become constitutive of the understanding of important categories in late capitalism" (ibid., p. 69).

It is thus the mechanism of unveiling the UI-side of Software Tiles that creates a tension between the software features players employ when developing an app, and the socio-technical effects their subsequently blackboxed code has when published as 'running code'. While the surface layer of the software might be stable—it is revealed in this phase of play that the underlying mechanisms is busy doing work within a computational ideological system. It is also in this phase that our second value goal is then meant to crystalize, as we aim to foster a critical stance for players towards algorithms as they are materialized in their contexts. For example, reading the code-side for the Software Tile for 'Notifications' feature might seem innocuous in its technicity. But when read in conjunction with the Event Card stating: "Your users are developing addictive behaviour! By displaying likes, shares, and comments gradually distributed over time, your users are compulsively opening your app to check for new notifications" (Appendix 2: Game Database) the algorithm becomes contextually and physically unveiled as a computationally ideological actant in its software materiality.

It is here that employing Diakopolous' algorithmic categories for the software tiles became useful as we designed correlated Event Cards and Software Tiles by reverse-engineering examples of biased, misbehaving, or socially problematic algorithms reported in 'algorithmic accountability reporting' or 'computational journalism' as Diakopolous (2014) calls it. An example of this can be seen in the 'Recommendations' feature with the algorithm: "Categorize [recommended content] by automatic filtering according to [content tags assigned by users]" (Appendix 2: Game Database). This algorithm was created from James Bridle's articles "How Peppa Pig became a video nightmare for children" (2018 June 17) and "Something is wrong on the internet" (2017 November 6) in which he wrote about people "using YouTube to systematically frighten, traumatise, and abuse children, automatically

and at scale” through reverse-engineering the Youtube recommendation algorithm (ibid.):

A second way of increasing hits on videos is through keyword/hashtag association, which is a whole dark art unto itself. When some trend, such as Surprise Egg videos, reaches critical mass, content producers pile onto it, creating thousands and thousands more of these videos in every possible iteration. This is the origin of all the weird names in the list above: branded content and nursery rhyme titles and “surprise egg” all stuffed into the same word salad to capture search results, sidebar placement, and “up next” autoplay rankings. (ibid.)

The Software Tile in question was thus developed with the following Event Card based on Briddle’s reporting (Figure 3.11.):



Figure 3.11. Peppa Pig Event Card.

Important to this mechanism of unveiling the UI of the apps by ‘checking the code’ is the fact that problematic algorithms of certain software features are not arbitrarily distributed. The tiles with more problematic algorithms are only found in more profitable apps, as each colour correlates to a specific value that contributes to the total revenue value found on the Develop Briefs (see Appendix 2: Game Database). For example: an app with a blue ‘Targeted Ads’ feature containing “Match [ads] to users by analysing [external content on the device user uses to access the app]” will be worth more than an app with a green ‘Targeted Ads’ feature containing “Match [ads] to users by analysing [content from user's publicly shared

posts]” on the code-side (ibid.). The difference is perhaps subtle, but it becomes explicit when read in conjunction with the corresponding Event Card:

New Privacy Law Passed: You have to tell users if you are **matching ads by analysing external content from the devices** used to access your app. Your userbase is outraged by this invasion of privacy! Your token value has dropped by -\$15. (ibid.)

Set within the frame of the game, we designed this mechanic so that more profitable apps might have more adverse social consequences—as players are encouraged to focus on the revenue worth of their apps, instead of the potential problematics of their software’s underlying encoded algorithms. Some of the problematic algorithms do however have non-punitive consequences, but are designed to foster a critical reading through a satirical slant, such as:

#1 Tool for Terrorist Recruitment! Your automated friend recommendation system based on triangulating common likes, groups, shares, posts are linking extremists all over the world with one another,” which has an unexpectedly profitable consequence: “You just keep gaining more users! +\$15 to your revenue value. (ibid.)

Likewise, there are a select number of Tiles that reward less profitable apps for things like protecting user privacy. These are meant to balance out the Event Cards in terms of creating a dynamic of expectancy that would increase the engagement and enjoyment of players. As Salen and Zimmerman point out: “The carefully crafted arc of rewards and punishments that draws players into games and keep them playing connects pleasure to profitability” (2004, p. 352).

Moreover, there is also a subset of Event Cards that should be addressed as they do not relate directly to ‘checking your code’. These are indicated by a yellow shopping cart icon and contain instructions as can be seen in Figure 3.12. shows. These cards function as a mechanism of uncertainty as they have outcomes in the game irrespective of players’ actions.

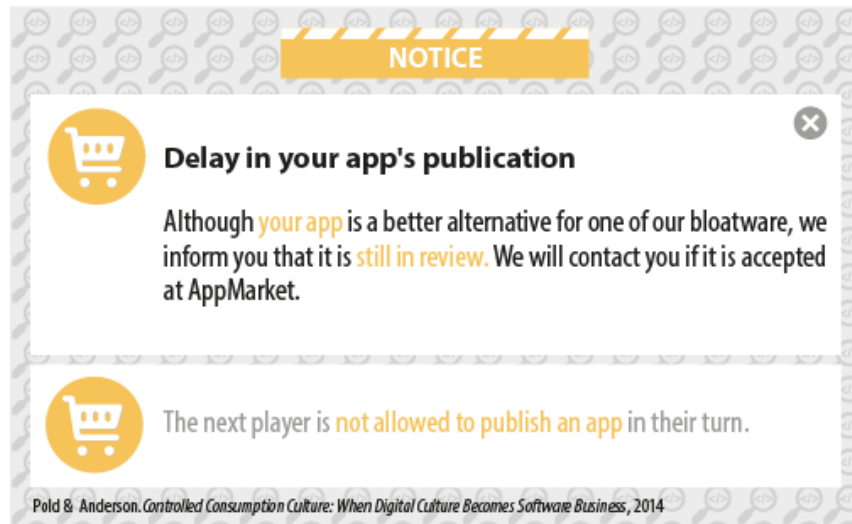


Figure 3.12. A special AppMarket related Event Card.

Within the value goals of the game and the consequent software critique it entails, these Event Cards are designed to draw attention to the software platform on which the player's apps are published (what Cramer and Fuller calls the 'third type' of interface). By doing so, the AppMarket is explicitly addressed as area of critique. This relates to the argument already made that the code the players 'develop' for their apps are based on an app markets' (and the mobile operating system that the app markets are bound to) SDKs and APIs. For as Pold and Anderson contend:

These platform-related structures affect how software can be designed and configured in order to be distributed through monopolies such as Apple's App Store. There are numerous stories about software that has been delayed or even totally rejected by Apple's gate-keeping—for example, because it undercuts or offers alternatives to the business model or is deemed controversial. (2014, p. 25)

The authors argue that digital devices are framed within such a system as "controlled consumption devices" upon which "we can decide to install, run, and modify software" (ibid., p. 31). Consequently, if producers "do not adhere to Apple's guidelines and business model, you are thrown out of the App Store and are thus denied access to an important platform for distributing cultural software" (ibid., p. 28). These special Event Cards in *Unveiling Interfaces* are then designed to exemplify the most controversial cases of controlled consumption imposed by these 'platform-related structures' as detailed by Pold & Anderson. Through these players as software producers also become subjected to the pressures of the larger software ecosystem within which they find themselves.

While the content and the consequences of the Event Cards are varied, the revelation contained within them are uniformly meant to show players that within the process of developing their software, they might not have had envisioned the social, cultural, political, and personal consequences their software might have in the real world. Given the situation within this new ‘accumulation regime’ created by a system of computational ideology, it is thus in designing Phase 2 that we aim to foster a critical stance in players towards the software ecosystems through which algorithms work.

3.2.2. Event 2: Collect Revenue

If the Week Marker on the calendar lands on the ‘pay day’ icon indicated by a dollar sign, no Event Notification Cards are played and ALL PLAYERS collect their monthly revenue. Collecting revenue works in the following way:

- 1. Each player adds up all the amounts indicated by the **Revenue Tokens** of their apps published on the AppMarket. This is their total revenue for the month.*
- 2. Then, each player updates their scores on their **personal smartphone calculator** by adding their total revenue. (Appendix 1: Rule Book, p. 6)*

In case the Week Marker does not indicate that an Event Card needs to be played, all players tally up the revenue from their published apps and add it to the total scores on their smartphone calculators. It is through this action, as well as the initial revenue they gather when first publishing their app (and the possible benefits set out in Event cards) that players can win the game. As Harris & Mayer pose “An inclusive scoring mechanic used in many designer games to track each player’s progress over the course of the game” by “assigning point values to different aspects of the game and indicating victory conditions” (2010, p. 116).

Importantly the point system within the frame of the game is monetary. The purpose of this is that as a critical game we designed *Unveiling Interfaces* to reflect on the correlative “non-game system” our game sets out to critique through its codification. For as players rush to win by only paying attention to their revenue, they are meant to reflect one of the central criticisms that prompted the Pew Research’s call for algorithmic literacy, namely that “algorithms are primarily written to optimize efficiency and profitability without much thought about the possible societal impacts” (Rainie & Anderson, 2017, p. 9).

3.3. Resolution and end-game scoring

When there are no more slots available on the AppMarket, the market has become saturated and each player gets one final turn before the game ends and the winner is announced.

Calculating Scores

- *From the final score, -\$10 must be subtracted for each Software Tile exposed in the players apps.*
- *The player with highest revenue earnings on their personal smartphone calculators wins the game. If two or more players are tied with the most points, the player who has more apps published is the winner and the next high-earning member of GAFA! (Appendix 1: Rule Book, p. 7)*

The end-game scoring and consequent resolution of designer games are “based on victory point gained through the completion of goals or gathering of resources” (Mayer & Harris, 2010, p. 6). Importantly “[d]esigner games that use an end-of-game scoring mechanic like this can keep everyone engaged, reducing the potential for disruptions from disengaged (and perhaps even disheartened) participants” (ibid.). This is crucial for the dialogical pedagogical space that the game aims to create, as such games “engage players in a shared community of play that allows for ongoing development” (ibid.).

As critical board game, *Unveiling Interfaces*’ end-game scoring is also designed to set all of the critique discussed in this chapter above in stark relief with the actions players need to take to win. For it is through successive Pay-days that players garner the revenue needed achieve the game objective. Yet to get to a Pay-day the game has to progress through three rounds of Event Cards. The revelation of the Event Cards during the second Phase of each player’s turn thus create a disjunction between the winning conditions of the game and the consequences set out by the Event Cards. How this creates a play environment in which the codified game elements can illicit critical play will be discussed followingly.

4. Critical Play in *Unveiling Interfaces*

Given the value goals oriented towards generating critical play set out in the beginning of this chapter, the design and conceptualization of the game works against the concept of ‘screen essentialism’. Berry raises concerns over how ‘screen

essentialism’ allows software to be “used/enjoyed without the encumbrance or engagement with its underlying structures due to this commodity/mechanism form,” thus becoming mere “consumption technology” (2014, p. 71). The same issue is also raised in the calls for algorithmic literacy, which is oriented towards enabling members of a public to critically reflect on their relationship with the unseen algorithms hidden behind software interfaces. Berry defines this as thinking “beneath the surface”, which necessarily entails an exploration of the “software [layer] that exists in code, and to use critical concepts and methods to understand and explain its functioning” (2014, p. 63). The idea of ‘thinking beneath the surface’ thus became the main premise throughout the design process. Yet designing for critical play led us to employ this premise in two fundamental ways, namely through discomfort design, and Freirean codification.

4.1. Discomfort Design

Berry’s ‘screen essentialism’ importantly informed the frame of the game, as players act as software developers who, according to the objective of the game, have to publish profitable apps by blackboxing their algorithms through the UI-side of the Software Tiles. By creating the actionable mechanic of blackboxing, we aim to give players an appreciation of how algorithms are developed into software within its socio-technical context, without requiring them to have technical knowledge of the process.

However, through the iconographic information of the UI-side of the Software Tiles, as well as emphasizing the Tiles’ colours instead of content in the Developer Brief combinations, players are led to initially *not* focus on the ‘mechanism face’—the underlying encoded algorithms—of the apps they publish. This focus is inverted as the Event Cards come into play, creating a space in which players are required to reconsider the Software Tiles as more than just UIs. For through the ‘expose code’ mechanism enacted by the playing of Event Cards, the game is intended to develop into a critical play space of discovery and reflection on the hidden implications of the encoded algorithms blackboxed by the UI-side of the Software Tiles. Therefore, as the players expose the code of a Software Tile, they not only unveil the UI, but also the inner logic of the algorithms as acting within their socio-technical contexts.

In the rule book players are informed that the object of the game is to publish profitable apps and get as much revenue as possible. Yet, as Event Cards are

successively played, a discord is created between the players actions or choices—directed towards winning the game—and the outcomes or effects of those actions—the adverse consequences contained in the Event Cards. For the more revenue a published apps earns, the more likely it is to contain Software Tiles implicated by Event Cards. We thus aim to create a space in which players need to reconcile the idea that profitability as a goal might have adverse social consequences for the users of their apps. This space is created through the excessive punishments Event Cards contains, meaning that the players that are publishing the highest-grossing combinations might realize that they are being excessively punished for trying to win the game by following the system of rules we have created. This creates an incongruity in their experience of the game mechanics, something Grace calls ‘discomfort design’ (2014, p. 7). Grace argues that such a game design “[forces] players to reflect on their understanding of a specific scenario” as the game mechanics create a “moment when player expectations are broken” and can thus serve “as a moment to recollect, reflect, and ask key questions” (ibid.). With this, we aim to provoke a change in the approach players have to the game’s initial goal, a change that is subtly prompted in the rule book when we ask: “at what cost are you willing to win?” (Appendix 1: Rule Book, p. 1). The player’s journey from which we aimed to generate critical play departs from an intuitive game strategy focussed on gathering as much revenue as possible, towards a mindful selection of Software Tiles that will not have adverse social consequences.

4.2. Codification

Through the Event Cards we aimed to create a space that encourages players to give a second thought to the content of their published apps and the system the game represents. For through the Event Cards players are forced to not only *read* the code-side of their Tiles, but read it within its material context as cultural software.

This idea ties back to our theoretical orientation towards a critical pedagogy as informing a literacy approach focused on societal reflection instead of mere technical competencies. In *Unveiling Interfaces* this is done through a process of codification and decodification. While the employment of software studies to inform the game elements and mechanics helped us codify the game system for critical play to emerge, within the design of the game we have also aimed to make numerous connections between the game and the social realities of the players. For instance,

by making the connection between the software players publish, and the software they invariably use in their daily lives. This was done through a number of actions, such as the inclusion of the players' real phones on the board in order to keep track of their scores on their calculator apps. Through this action we wanted to connect the ubiquity of smart devices—often placed compulsively in the same manner on tables during other social events³⁹—and the content of the game that takes place on a board resembling a smartphone.

Another example of the correlation between the game and reality is the use of real case studies for each Event Card, indicated in the game through a citation from the actual reported event at the bottom of each the cards. In this, Event Cards can be described as designed according to a 'continuous critique design', a term Grace uses for "critical games [that] are emphatic in their structure, repeating their critique over and over through common game mechanics, a set of repeating scenarios, or explicitly delivered message" (2014, p. 6). Yet in the events of the Event Cards, we strived to not only reiterate our cultural software critique in each turn, but also to create a space for reflection and debate among the players about their social realities with different events as prompts.

It is by designing these connections between the game and reality that we aimed to generate subtle ways of breaking the 'magic circle' of the game and engender critical play related to the social context in which the game unfolds. Critical games, after all, are design to "looks outward from games toward the society and culture in which they exist" (ibid., p. 5). With these contextualization cues, we aim to nudge players to reflect on their own roles not as producers in the game but as users of cultural software on their personal devices, and thus create space within the gameplay to develop an awareness and critical stance towards the algorithms in their daily lives.

The 'codification' of the game components is also informed by how the theme or frame of *Unveiling Interfaces* is supported by the game mechanics. For as Mayer and Harris propose, one of the most important and most "complex feature of many designer games is an intricate interplay between mechanics and theme" (2010, p. 6). According to Mayer and Harris, the rules/theme successful interplay will lead the

³⁹ In a 2014 Pew Research survey, "89% [of participants] said that they themselves used their phone during their most recent time with others, and 86% report that someone else in the group used their cellphone during the gathering" (Rainie & Zickuhr, 2015, n.p.).

players to be immersed in the theme, as the “story elements [...] provide meaning and context for the mechanical actions of the game” (ibid., p. 115). It is precisely in the player’s immersion where the game will potentially become a “powerful tool for learning” (ibid., p. 7), and thus facilitate the critical play value goals.

It is finally as players read the code-side of the Tiles in relation to Event Cards, that the game is intended to prompt decodification. For through the Event Cards we have attempted to create a moment of discomfort that leads players to reflect on the content of the game. Crocco writes that this kind of game design has a ‘defamiliarization’ or—using Bertolt Brecht’s term— ‘Alienation effect’ (2011, p. 30). Crocco writes: “when a game is used as codification material, the new context generates an A[l]ienation-effect that enables students to question its reified ideology and critically reexamine their conscious or unconscious adherence to this ideology” (ibid.). It is this moment of defamiliarizing discomfort in the game that would allow players to decodify the content of the game and engage with the value goals of our *Unveiling Interfaces*.

CHAPTER 4

Playtest and Evaluation of Critical Play Goals

1. Evaluation Design

As part of our evaluation process we drew on Zimmerman's understanding of iterative game design as "a design methodology based on a cyclic process of prototyping, testing, analysing and refining work in progress" (cited in Fullerton et. al., 2004, p. 16). Thus, during *and* following the design process described in Chapter 3 we conducted what is known as playtesting. Playtests within the process of game design comprises an evaluation process in which the game is played by players for the purpose of giving the designers insights into how players experience the game and whether the game fulfils its design goals (ibid.). Additionally, to frame this process in terms of the critical play method, we expanded the evaluation to include the phase of verifying values and design goals. The evaluation is thus meant to verify "that the values goals emerge through play, and revise goals and add or drop options based on feedback to ensure an engaging game and support the project values" (2009, p. 258). It should however be noted that within our practice-led research method playtesting also informed the research process, as well as the formal evaluation of the academic research conducted.

2. Playtests

2.1. Playtests as Internal Design Review

Following Zimmerman's approach, we prototyped and tested at every stage of the conceptualization and design of the game. The first discussions of constitutive rules, game mechanics, and system design were indeed accompanied by the first paper prototype and micro-playtesting sessions between ourselves. Playtesting as game designers follows Fullerton et. al. model (Fig. 4.1.) of playtesting phases, through which he argues that it is necessary to first evaluated the foundations and structures (constitutive and operational rules) through playtesting as an internal design review process.

Prototyping Stage	Playtest on Your Own	Playtest with Confidants	Playtest with Target Audience
1) Foundations	●		
2) Structure	●	●	
3) Formal Details			●
4) Refinement			●

Figure 4.1. Types of playtesters appropriate for each stage of prototyping. Reprinted from Game Design Workshop (p. 252), By T. Fullerton, C. Swain, and S. Hoffman. 2004, San Francisco: CMP Books.

The most remarkable discussions⁴⁰ during our first internal design review were in terms of the components of the Software Tiles, the relation between the Tiles and the Event Cards, and the modularity of the Developer Brief in their relation to the colour-combinations of Tiles. By implementing the insights arising from the internal design review we arrived at a second version of this first prototype (Fig. 4.2.). After another playtest session with this second version of our first prototype, we realized that while the constitutive rules were clear and there was a good flow in the game, the content of the Software Tiles and Event Notification Cards were not developed well enough and would need another iteration. Following, we playtested again to evaluate the structure of the game and made small adjustments such as adding a the calendar and Value Tokens that could be used to distinguish between different players' apps in the board.

A second prototype was then developed, following Fullerton et. al.'s model, for 'playtest with target audience'. Working within a time-constraint, we took a design-thinking approach and developed this second prototype as a minimum viable product (MVP), consisting of nine types of software tiles, improved visual designs for all the game elements, and reworked copy for the Software Tiles, Event Cards, and Developer Briefs. We also developed a partially illustrated rule book (Appendix 1:

⁴⁰ The full list of the discussions can be accessed in Appendix 4: Internal Design Reviews.

Rule Book) for players. As Fullerton et. al. suggests: “[y]ou should be able to give some playtesters the prototype materials, and they should have enough information to complete the game [...] this will require that you write a full set of rules” (2004, p. 250).

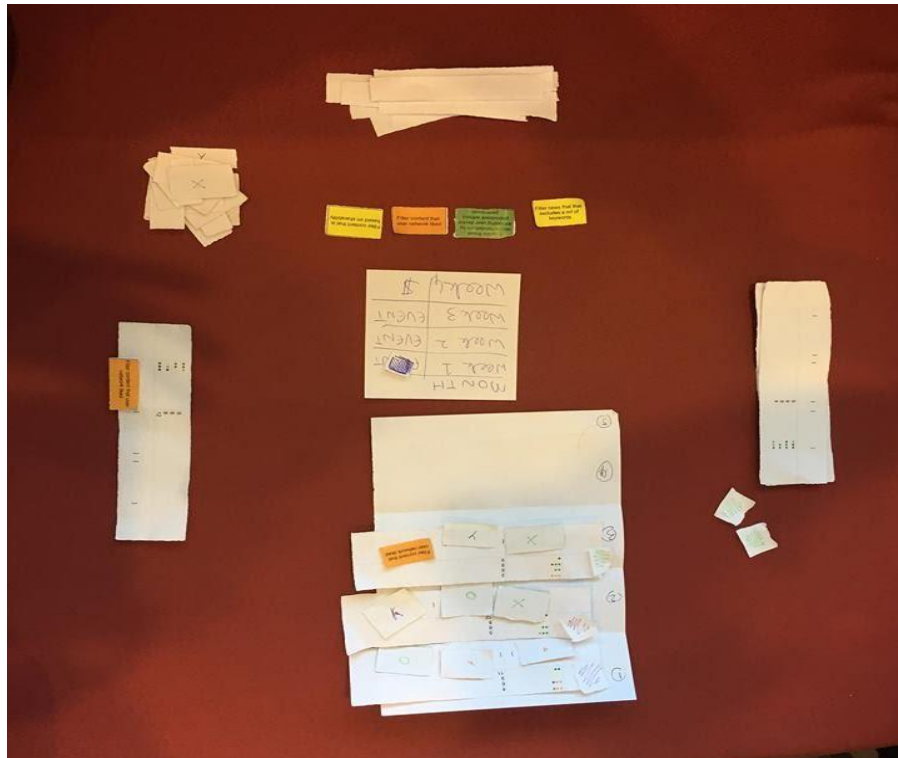


Figure 4.2. Complete second version of prototype 1 after Internal Design Review 1: board, Developer Briefs (colour-coded), Software Tiles (colour-coded), calendar, Week Marker, Revenue Tokens, and Event Notifications (colour coded). See Appendix 5: Playable Paper Prototypes for a list of descriptive illustrations related to different stages of the playtest process (Appendix 5: Playable Paper Prototype).

Yet before we could playtest with our external players, we did a final internal design review between ourselves. This gave us some useful insights to add before playtesting with the target audience, such as organizing some elements of the board and implementing an infographic design to improve visual cues for the player’s turns, redesigning the visual elements and copy of the Event Card for ease of reading (and consequent compression during gameplay). After this last internal review session we had a prototype that as a MVP could be evaluated with other players (Fig. 4.3.).

Conclusively, the playtest as internal design review shaped every component of the game as described in Chapter 3. Moreover, through the successive iterations of prototyping and playtesting the research constituting Chapter 3 itself took on an iterative nature—changing with each design insight we gained. Though written up as

a coherent chapter, it should be noted that the research was developed through a rhizomatic process of practice-led research as informed by this iterative design process itself (Smith & Dean, 2009, p. 21).



Figure 4.3. **MVP: Complete version 2 of prototype 2 after the third internal design review.** Changes included: better contextualization for the AppMarket in the board; numbers on each slot of the board; reorganization of the flow of the player's phases as an infographic (Appendix 5: Playable Paper Prototypes).

2.2 Playtest with Different Audiences as Project Evaluation

The purpose of playtesting with external players is to evaluate whether the game “is internally complete, balanced, and fun to play”, but also to fundamentally “make sure the game is functioning the way you intended” (Fullerton, 2004, p. 248). In other words, playtesting with external players allowed us to evaluate our game against its design and value goals, as set out in the beginning of Chapter 3.

Structuring an external playtest can be done in numerous ways, as Fullerton et. al. write, “some of which are informal and qualitative, and other which tend to be more structured and quantitative” (ibid.). The one thing all playtesting have in common, however, is its goal “to gain useful feedback from players in order to improve your game” (ibid.). Yet as Smith & Dean suggests, when conducting practice-led research, an evaluation of creative practice can benefit from appropriating evaluation methods from the sciences (2009, p. 27). In this regard, we did consult social science evaluation methodologies regarding focus group studies as well as qualitative data analysis, working within certain of their concerns to help conduct an evaluation that could deliver findings with more academic validity than ‘informal’ playtesting would have allowed. Through this we designed and implemented the playtest with different audiences as a way to evaluate our project for the purposes of this thesis as well.

2.2.1 Sampling

Bryman writes that the sampling of areas or contexts, and then participants within that context “is a common strategy in qualitative research” (2012, p. 417). As our concern was reaching the general public, we chose to conduct an initial sampling at DOKK1 in Aarhus, an institution that functioned not only as the city library and municipal Citizen’s Services, but also as a community space for “social activities” through offering “multi-functional spaces” and “informal open areas” (Urban Mediaspace Aarhus, 2015, p. 9). However, after struggle to find willing participants here, we realized that visitors to DOKK1 at the time were not predisposed to play board games, especially as it would require a substantial commitment of time⁴¹.

In conventional playtesting, write Fullerton et. al., “[y]ou want testers who actually go out and spend their hard-earned money to buy games like yours” (2004, p. 251). Our audience, we realized, were not just members of the public—but people who have some predisposition to play board games. With only a single playtest resulting from our time at DOKK1, we subsequently changed our area of sampling to a public ‘table-top games evening’ at the *Epic Panda Pop-culture Lifestyle-Boutique*

⁴¹ Given that our playtesting was done during the summer holiday season, we found that most visitors we approach at DOKK1 were either there to complete administrative tasks at the Citizen’s Services, or where vacationers briefly visiting the library as a city landmark and tourist attraction.

in Aarhus⁴². Here we completed our second and third playtest sessions. After three playtests sessions, with a gross total of 7 participants, we ended our sampling process. We concluded that we had reached a state of saturation, which is when “new data no longer suggest new insights” (Bryman, 2012, p. 421). This was supported by the fact that although participants in Playtest 1 played under completely different constraints, such as a protracted play-time, and Playtest 3 presented new variables such as an extra player and players with experience in software development, the same fundamental concerns emerged during all the playtest sessions. We considered that successive playtests not give significant new insights before we address these fundamental concerns in a design iteration.

2.2.2 Data Gathering

The data gathering process consisted of the authors approaching prospective participants with the proposition of helping us playtest a board game developed as part of a student research project. Upon initial agreement we would sit down with prospective participants and read an introductory script that stated who we are, the purpose of the playtest (both in terms of design practices and academic research) as well as what the playtest would consist of (Appendix 6: Playtest Script). We then gave them a consent form to fill in as research participants (Appendix 7: Playtest Sample Consent Form) in which we also stated (and reiterated verbally) that we would audio record the playtest. We then asked them demographic information, such as age, profession, and level of experience with software development (Appendix 8: Playtesters Form). Following, we would give them the rules to read and tell them to start playing whenever they felt ready⁴³.

Importantly, we selected to follow a ‘focus group’ design, as it would offer us “the opportunity to study the ways in which individuals collectively make sense of a phenomenon and construct meanings around it” (Bryman, 2012, p. 504). Firstly, we took extensive notes during, as well as after, gameplay. This was done according to precompiled guides that drew on conventional playtest topics of concern while also focusing on particular elements that emerged as pertinent to our game design and

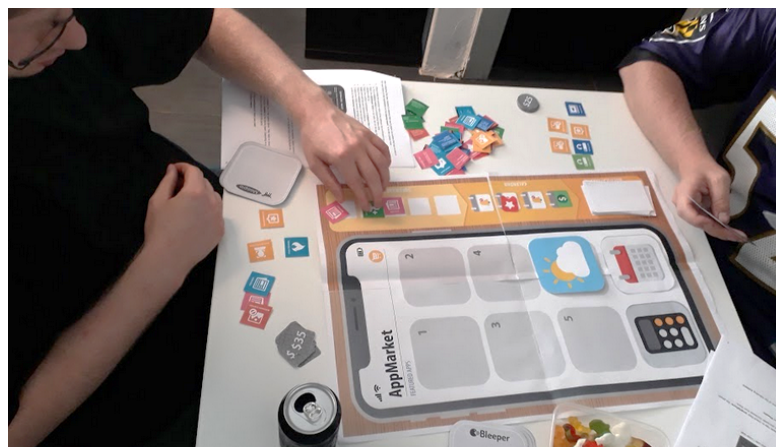
⁴² Epic Panda describe themselves as “the center of everything that fans of all pop cultures are excited about! Whether it's Movies, TV Series, Games, Retro Consoles and more” (2018, n.p.). Importantly, they also have venue space for table-top gaming evenings which they regularly host.

⁴³ Working with time constraints, in Playtest 1 we explained the rules verbally to expedite the process.

value goals during our second prototype design iteration. Secondly, we had a guide for in-game question as well as post-game questions that we could use to gauge participant's experiences as well as illicit conversation and commentaries (Appendix 11: Playtest Guides and Notes). Responses to the questions, as well as conversations that emerged during and after play, were recorded and transcribed⁴⁴ while observations were made with hand and subsequently transcribed and elaborated with the aid of the audio-recordings (Appendix 9: In-Game Transcripts & Themes and Appendix 10: Post-Game Transcripts and Themes).



*Fig. 4.4. **Playtest Sessions 1**, Aarhus, Denmark. July 10, 2018.*



*Fig. 4.5. **Playtest Sessions 2**, Aarhus, Denmark. July 12, 2018.*

⁴⁴ We omitted certain descriptive and non-relevant statements from the transcripts, for as playing games are social activities our recordings included a lot of conversational exchanges between players that we felt would be irrelevant for analysis.



Fig. 4.6. Playtest Sessions 3, Aarhus, Denmark. July 12, 2018.

2.2.3 Data Analysis

With the transcribed data, we had a body of qualitative data to work with. Together we went through the documents comprising our data and started coding common themes that emerged during the playtest sessions, and also drew insights regarding design problems, communication problems, successes and failures that emerged (Appendix 12: Theme Analysis). The insights drawn from this analysis will be discussed in the following section.

3. Evaluating Critical Play Goals

Following Flanagan's critical play method, after the playtest with different audiences, the next task was to verify values and revise goals. We have divided the evaluation according to the two different type of goals needed for critical play, namely: design goals and value goals.

3.1. Revising Game Design Goals

Our goal in terms of game design was to create a 'gateway game'. This premise prioritizes that the rules needed to be few, simple, and easy to learn. Yet the game should still offer space for strategy and interaction (Mayer and Harris, 2009). This goal also informs the time frame of the game which should be held in less than an hour. On balance, we have recognized two key areas that were successful in the process of revising the game design goals: **the theme and game mechanics**. Regarding the appeal of the theme, most of the commentaries were favourable. As

Player 4 stated: “the theme makes it, it makes it unique, in some way” (Appendix 10, transcript 2, line 62). Moreover, in the case of the third playtest session, the game elements and theme seemed to be the main reason why the participants approached our table.

Likewise, players expressed that in overall the game seemed internally coherent and logical. Notably, the dynamic of building software with colour combinations seemed clear, as none of the players asked any questions regarding this rule. Hence, the mechanics between Developer Briefs and the collection of Software Tiles worked as intended. As a matter of fact, players raced to collect the tiles for the most high-value apps, displaying different types of strategy for collecting the tiles and keeping their Developer Briefs hidden from their opponents. The relation between the Developer Briefs and the Software Tiles responds to the intention of balancing the game mechanics with the theme in regards of creating meaningful play. The aim was partially accomplished in this iteration, given that the theme of the game gave context and supported the mechanics of ‘developing’ software apps, and thus the rules were not too abstract for players. This point also contributes to the main goal of the game design as a gateway game, as the rules involving Phase 1 of each player’s turn were indeed easy to follow for all the participants.

However, there were also some game mechanics that were not understood as quickly, especially playing Event Cards in Phase 2 (this will be discussed below). However, during the playtest sessions the game nonetheless flowed after the game mechanics were fully understood by all the players. We asked in two out of three sessions whether they would like to end the game, as the playtest was elongated for more than 40 minutes, and in both occasions, players wanted to play at least one more round. This can be seen to indicate engagement with the game mechanics. As an illustration, Player 7 replied to us before we concluded the session: “I would like to play it more, I think is fun, but it is only a test” (Appendix 10, transcript 3, line 4). Therefore, we contend that after the rules were understood, players seemed to enjoy the progress of play. This result also supports the game design goal of a gateway game, as the game’s simple game mechanics enables all kinds of players to participate. Moreover, we also noticed that after changing the setting of Playtests 2 and 3, the players were having much more fun as indicated by their laughter. Thus,

the game created a social experience, which is the first prerequisite for creating a dialogical space in which discussion and debate can occur.

The most compelling evidence of the reception of *Unveiling Interfaces* as a social game was through the commentaries and conversations that emerged when Event Cards were played. The reading of these cards generated a myriad of reactions, including: suspense, shock, amusement, and curiosity. For instance, Player 7 expressed: "I like the theme, and I really liked the little stories in the notifications. Now, I am a little nerdy, so I think is funny the things like, 'your app recognizes pictures of black people as monkeys' that is a little funny for me, I liked that" (Appendix 10, transcript 3, line 40). Player 3, on the other hand, reacted surprised at another Event Card, exclaiming: "What!? 'Terrorist Recruitment?'" (Appendix 9, transcript 2, line 19).

Given these points, it can be claimed that our prototype as MVP provisionally achieved the game design goal of creating a casual gateway game. Moreover, we have not identified any significant contradictions in the rules that cannot be fixed with the addition of single lines to the rules. Moreover, especially in regard to Phase 2, we feel that players can understand the game event quicker if we simplify the rules as it was especially in the process of 'exposing' Software Tiles that players indicated some confusion. Consequently, we have identified some additional solutions that will improve the game and allow us to fully achieve this goal:

1. Revise and rephrase instructions in the rule book to make them clearer, especially in regards to Phase 2 (Appendix 1: Rule Book, p. 5).
2. Add instructions in the rule book about where to place the bloatware apps during setup.
3. Add instructions regarding in which order apps should be published on the board.
4. Remove the phrase 'blind draw' in Action 1, from the rule book.
5. Rephrase copy of Event Cards so that they don't contain first-person pronouns and rather refer to any player with implicated Software Tiles⁴⁵.
6. Add images to the rule book to illustrate the setup, as well as the components of the Event Cards and how they relate to published apps.

⁴⁵ This reflects our observation that players were confused whether Event Cards applied to the player whose turn it is, or to all players.

7. Remove Action 2 from the rule book. Instead create mechanism for automatic refill of Developer Briefs so that the players always have three Briefs in their hands.

The first five solutions will contribute to the clarity of the rules, while the last solution will improve the pace of the game⁴⁶.

3.1.1. Revising Meaningful Play

Although some game mechanics worked and contributed to generate a flow in the game, the current iteration is still in the process to achieve meaningful play. Salen and Zimmerman explain that 'meaningful play' is considered to occur when the player perceives the relation between the actions they perform in the game and the outcomes those action generates to be discernible and integrated both in the game and its frame. Drawing on this approach, we have identified the following common themes⁴⁷ that emerged during the playtests in relation to meaningful play:

1. **Event Notification Cards:** Players exhibited problems understanding how to play an Event Cards and discerning whether the cards affected only the player who drew and read it or all the players. Further, we observed that the players did not comprehend the relation between certain Event Cards and their correlative Software Tiles. The problems that led to this confusion are: the uncertainty about the 'exposed code' mechanism as players did not always know when to expose the Software Tile and which tiles to check on the board; the ambiguity between 'check the code' title in the top of the card and the 'expose your code' punishment on the bottom; the players did not interact with the highlighted keywords on the Event Cards, but rather expressed that there was too much information. It is especially this last problem that led to players simply skimming the Event Cards for consequences and often applying them to incorrect Software Tiles.
2. **Software Tiles content:** The players did not recognize that the code-side on Tiles that shared the same UI-side could be different. We were asked multiple times why the Software Tiles were double-sided. Some of the players

⁴⁶ This was first suggested by Player 4, who indicated that removing this step in the game will make it flow faster as a player will not have to waste a turn drawing new Developer Briefs (Appendix 10: Post-Game Transcripts and Themes, transcript 2, line: 23). Moreover, this auto-refill rule will prevent players from hoarding Developer Briefs, as occurred in Playtest 3.

⁴⁷ The complete analysis can be checked in Appendix 12: Theme Analysis.

recommended to just have one side with the feature. Consequently, this confusion affected the player's perception of the relationship between the action of publishing an app and the outcome of the code in relation to the Event Cards. This relation was indeed not discernible, nor integrated in the game for players. Firstly, because the participants played exclusively by focussing on making colour-combinations and publishing the highest revenue apps. Secondly, because this led to the assumption that a Software Tile was implicated merely by the feature icon in the UI-side, rather than the code-side as intended. Put differently, in all the sessions the code-sides were barely examined, and the players were reluctant to read it. We consider that this is also a consequence of the Event Notification Cards not being understood, as well as the amount of text on the Tiles and the font-size which was difficult to read once the Tile was placed code-side up on the board.

3. **The balance between luck and strategy:** Even though the game was regarded as a resource management game, due to the misunderstanding of the code-side's content of Software Tiles, the actions of collecting and publishing of the Tiles were regarded as a sort of gambling. As an illustration, Player 3 answered to one of the in-game questions about the reasons for choosing a specific colour combination as hoping to "get lucky" (Appendix 9, transcript 2, line 43). While other players likewise indicated no strategy to avoid punishments for published apps. Players merely devised strategies to gather more revenue. The first one was to collect the Software Tiles that composed the high-revenue colour-combinations⁴⁸. The second strategy was in terms of publishing as many apps as fast as possible before losing them⁴⁹. Nonetheless, the players struggled to find a clear strategy to win the game and avoid the punishments, as they stated that the only space for strategizing was in drawing the right Software Tiles. This is as well a by-product of the problems understanding the Event Cards' relation to *certain* Software Tiles. Punishments were thus considered to be meted out randomly, even being compared with *Monopoly's* 'Chance' cards. As Player 3 stated: "it makes it more random but... then again it is sort like when you land on a question

⁴⁸ This strategy was manifested in Playtest 3 as players were reluctant to draw the pink tiles from the board, referring to them as a "bad colour" (Appendix 10, transcript 3, line 9).

⁴⁹ Player 2, commented that he need to be fast because there is "something" being risked (Appendix 10, transcript 1, line 7).

mark in Monopoly” (Appendix 10, transcript 2, line 11). One player even expressed that Event Cards make “it hard to plan ahead, somehow to be tactical about it.” (ibid., line 7). Finally, this also caused the players perceived that the game was not progressing, as they felt that they could not influence the outcome of their actions, and effectively failed to develop a strategy for mitigating the effects of the Event Cards.

4. **Too much technical language:** The players confused many of the names of the game elements and actions, such as mistaking ‘Developer Brief’ for an ‘app’. The most problematic result was observed that players did not have a technical language to differentiate the code- and UI-side of the Software Tile. As Player 3 expressed, the content seemed too abstract (Appendix 10, transcript 2, line 52). The only group that was amused reading the Tiles in relation to the Event Cards were the third group as they were the only ones with technical knowledge about software development. Players from Playtest 2 even remarked that software developers would have more fun playing this game than “normal people” (Appendix 10, transcript 2, line 49). Player 5, a computer science student, likewise said that he would not play it with his family because they would not understand the software development concepts (Appendix 10, transcript 3, line 49). However, Player 3 commented that perhaps people who are active on social media like herself might grasp the content better (Appendix 10, transcript 3, line 54).

Given these themes, in terms of meaningful play the fundamental problem is localized between ‘ACTION 3: Develop and Publish an App’ and ‘EVENT 1: Get a notification to ‘check your code’’. This problem can be explained by the lack of discernible and integrated outcomes for the players, which in the case of *Unveiling Interfaces* was a result of the incomprehension of the relation between the Event Cards and the Software Tiles. As Salen and Zimmerman explain, “discernability means that a player can perceive the immediate outcome of an action. Integration means that the outcome of an action is woven into the game system as a whole” (2004, p. 186). Adjusting the two elements will provide the game with a better balance between luck and strategy, which is one of the main characteristics of designer games. Additionally, it will also prevent the Event Cards from being perceived as random, and the action of ‘publishing apps’ from being perceived as

gambling. As a result of such adjustments, players will be able to feel that “the choices they make in the game are strategic and integrated” (ibid.). For as Salen and Zimmerman pose, “meaningful play in a game emerges from the relationship between player action and system outcome; it is the process by which a player takes action within the designed system of a game and the system responds to the action” (2004, p. 50).

3.2. Verifying Value Goals

Provided that the starting point to create new meaning within a game system is through meaningful play (Salen & Zimmerman, 2004), critical play was not accomplished in this iteration and our value goals were consequently not met. For this reason, the post-game question regarding algorithms in daily life confused all the participants.

The first value goal which aims to ‘allow players to develop awareness of algorithms present in everyday cultural software’ was intended to be answered by the game mechanics of developing an app that reflects those players use in their actual smartphones. This was the main rationale behind the selection of cultural software as part of the frame of the game, which would allow players to contextualize the fictional software they engage with in the game. Furthermore, the apps that the players published are composed of four Software Tiles that had a code-side that represented the algorithm. However, as the relation of the players with the Tiles was merely in terms of collecting colours and focussing the UI-side, the contextual reflection intended was not successfully created. Additionally, due to the way the information was distributed on the code-side and the amount of text contained therein, players didn’t seem to realize that the code was the underlying algorithm of the Software Tile. Thus, it can be argued that the already simplified technical language hindered the players to fully engage with the game’s frame, as it alienated them on the level of technical appreciation⁵⁰. This insight can be reflected upon as supporting Habermas’ argument that technical language eliminates possibility for public discussions on technologies that affect society.

⁵⁰ Among the playtest sessions one of the phrases that was repeated when players were asked about the theme of the game was that they were just ‘normal people’. The use of this particular word demonstrates that by using technical language players felt excluded from the conversation, as they consider that they are not specialist but just ‘normal people’.

The problems outlined above also affected the achievement of the third value goal. The game failed to ‘instilling a technical appreciation of algorithmic technologies without relying exclusively on technical knowledge’, as the mechanism of ‘exposed code’ was not understood by the players within the frame of the game. We observed that the players who did not have any technical experience in software development didn’t perceive any significant relation between the two sides of the Tiles. In effect, the ‘expose code’ mechanism, which entailed exposing the algorithms behind the UI-side, did seem to rely on technical knowledge and was communicated unsuccessfully. Players thus failed to *read* the Software Tiles in relation to either their encoded algorithms, or how those algorithms worked in a socio-technical context.

Finally, the second value goal, which aimed to ‘foster a critical stance towards algorithms in context’ failed as it likewise relied on the understanding of the relation between the Software Tiles and the Event Cards. The concept behind the relation of those two elements was based on placing the algorithms in its social contexts as materialized through cultural software. Even though placing the citations for the events in the Event Cards were aimed to engage players to reflect on the consequences of algorithms in society, and thus generate critical play, there were many problems delivering the information. On the one hand, players commented that there was too much information on both the Event Notification Cards and Software Tiles to pay attention to all the content. On the other hand, we observed that an implicit rule emerged in which once the Tiles were placed, the Tiles were just flipped over on the board and not held up to be read when their code needed to be checked (as specified in the rule book). Thus, the code was lost among the many other elements on the board. This also revealed a ‘desired path’⁵¹ that we did not foresee, as the size of the text was intended to be read from a closer distance.

Our efforts to create a system of codification and decodification thus seemed to fall short, as our use of software studies theory were not interpreted in the intended manner. However, despite these problems we did observe some spaces in the game environment and mechanics that worked incipiently towards generating critical play and fulfilling our value goals.

⁵¹ This is a concept used in design, derived from the phenomenon where pedestrians traverse unpaved roads and “[create] spontaneous new trails” that were not envisioned by urban-planning designers (Kohlstedt, 2016 January 25, n.p.).

Firstly, the concept of modularity of software was communicated successfully through the game mechanics without relying on technical knowledge of players. As one of the Player 4 expressed, “I think in the very abstract way of the apps having targeted ads, well you see that all the time, Facebook targeted ads. I think is perhaps a bit funny to see the idea to putting in targeted ads into the apps to make them ‘work’” (Appendix 10, transcript 2, line 62).

Moreover, it is interesting to point out that the mechanics in the Event Cards that had bad social consequences, but rewarded players nonetheless (see Figure 4.7. for an example) created surprise and shock for players. In the case of Playtest 3, this game mechanic made the participants realize that a good consequence could be read as a bad event in the real world when the Event Card was played:

P6: ‘Number one tool for terrorist recruitment. Your automated friend recommendation system based on triangulating likes, groups, shares, are linking extremists all over the world with one another. You just keep getting more users. You add \$15 to the revenue’.

P7: (Surprised) You add?

P6: Yes.

P5: Yes (laughs).

P7: (Sarcastic) Thank you terrorists!

[...]

Karla: What do you think about that event.

P6: Like, it kind of makes sense-

P7: It makes sense!

P6: Because more users – but it’s still bad.

(Appendix 9, transcript 3, lines 8-17)

This showed an interesting tension between players’ expectations and mechanics of the game, as players had gotten used to a correlation between Event Cards and punishments, which could have prompted a degree of Crocco’s ‘Alienation-effect’ with the codified content.



Figure 4.7. **Event Notification Card “#1 Tool for Terrorist Recruitment!”.**

Similarly, the severe punishments in the Event Notification Cards did seem to create a discomfort for players regarding the game mechanics, prompting some players to try and decode what was written on the code-side of the Software Tiles and why they had been punished. Yet, the code-side was too cryptic and difficult to comprehend for most of the playtesters and thus intended ‘discomfort design’ didn’t result in any sort of ‘decodification’ of game content. The fact that players could thus not decipher the relation between their actions during Phase 1 and the outcomes of those actions in Phase 2 also inhibited this ‘discomfort’ effect to come to fruition and allow players to reflect on the game mechanics and themes and perhaps revise their strategies. Player 5 stated: “The fines – the problem with the fines are that they are too negative” and went on to suggest that this was a design flaw (Appendix 9, transcript 3, line 143). The excessive punishments were thus seen as bad design, as they didn’t illicit critical play and thus encourage players to revise their actions. Conclusively, regardless of the creating a potential space where critical play could have emerged, none of the value goals were attained. However, as will be explored below, the few successes we had—as well as the more insightful failures—allows us to offer possible solutions to achieve both meaningful play, and through that, critical play.

4. Suggested Solutions

These failures discussed above that emerged during our playtest sessions should not be taken as indicators of the failure of the conceptual premise of *Unveiling*

Interfaces as a board game or a research project. Rather, we hold that the playtests prompt insights into certain game design and communication failures that can be addressed in a successive iteration to better create a situation of meaningful play which would provide the space for critical play to emerge. For this purpose we have outlined possible implementable solutions to help a next iteration of *Unveiling Interfaces* meet our game design goals as well as our value goals:

- 1. Re-phrasing the Event Cards:** We designed the factual citations on the Event Cards with the intention of piercing the insular ‘magic circle’ of the game and thus generate critical play. While this did not happen spontaneously, during one playtest (Appendix 9, transcript 2, line 35) we noticed that some degree of critical reflection on the game content seemed to happen when one of the authors mentioned to a participant that an event they just encountered through the Event Cards was indeed based on a real case. It is thus clear that there is potential in improving the how the Event Cards are communicated as being underlyingly factual. This leads us to ask firstly: what would happen if the game was explicitly presented to players as an educational game based on factual research? Would the reception and attention to detail have been different? Secondly, we propose to improve the Event Cards both in terms of design and copywriting to emphasize their factuality, beyond the almost illegible citation at the bottom of each card. We have also considered changing the phrasing in the rule book, so that the events are presented more in a journalistic frame that emphasizes their factuality as case-studies. With such changes, we predict that the Event Cards might go from being perceived as random fictional in-game events, to information about the real world. This shift, coupled with the strong reaction to the punishments triggered by the Events Cards, can perhaps enhance the space for critical play to unfold. And as already noted, it is the content of the Event Cards that generate most in-game discussion between players, which might then facilitate debate and reflection about the real events during gameplay, and how they relate to the context of the game. Romero’s critical game *Train* likewise relies on players reading the ‘Terminus’ card which reveals that the train destination is Auschwitz, and consequently reflecting on the factual basis of her seemingly innocuous transportation game. As commentator described observing a play-session in which the players didn’t seem to arrive at ‘critical play’:

At this point another audience member decided to get involved by asking the players if they had read the Terminus card that had been played and the game slowly shut itself down as the players took a step back and looked at what they were doing. (Logan, 2011, p.7)

2. **Critical Framing:** The game could benefit from balancing its theme and game mechanics through a more explicit framing of the game as enacting a technocultural critique. This could help create a predisposition in players for critical play to unfold and make them more receptive to the game's underlying values. One way of doing this can be in changing how we frame the players' roles, for example. Compare how in *Unveiling Interfaces* players are an "app developer, trying to succeed in a competitive tech-market" (Appendix 1: Rule Book, p.1), while in *Bad News* (the online critical game about 'fake news' creation) "you take on the role of fake news-monger. Drop all pretense of ethics and choose the path that builds your persona as an unscrupulous media magnate" (Bad News, n.d., n.p.). Throughout our game we deliver our software criticism in understated ways. Thus a more explicit framing can help codify the content better as it will enact a stronger 'continuous critique', as Grace would say, and more easily prompt a 'decodification' of the game content and system.

Primary sites for effecting this change would be the description of the game and the roles of the players and adding more explicit cultural critique in communication cues throughout the game elements, as well as within the thematic elements in the rule book which effectively frames the whole game. Additionally, adding illustrations to the Rule Book, and box-art illustrations, can function as visual communicational cues for a more critical frame (Figure 4.8. illustrates how other games do this). The use of explicit communicational prompts aligned to our value goals would effectively predispose players to see the 'codification' we intended with the game elements and mechanics.

3. **Potential of game punishments:** Our intention of generating critical play through the discomfort created by the excessive punishments of the Event Cards can be improved through making the outcomes of the players' action of publishing apps in the board more clearly related to events and punishments constituting the Event Cards. This will make the outcomes of that action more discernible and integrated in the game. One way of doing this would be through a more critical framing as suggested above. As some players have already come

to the realization that Tiles of certain colours are more valuable as constitutive parts of higher grossing apps (see Appendix 9, transcript 3, line 138), we can design the association between higher value and negative effects more clearly in the rule book by explicitly stating which colour Tiles have negative consequences (and still leave it up to the revelations in the Event Notification cards as to *why* these tiles are problematic). Another solution here is inspired by the comments from one of the players with proclaimed experience ‘spread sheeting’ games⁵² (see Appendix 10, transcript 3, line 91) when he indicated that he was at a loss as to how he would be able to reverse-engineer the constitutive rules governing the punishments. This not only suggested to us that there was a perception of imbalance between strategy and luck, but inversely that this problem could perhaps be fixed *using* spreadsheets as constitutive rule design tools. Game designer Ian Schreiber, co-author with Brenda Romero of *Game Balance* (2017), suggest on his online ‘Game Design Course’ under the topic of ‘Game Balance’ that designers need to “[l]earn to love Excel” as “[y]ou can use spreadsheets to run statistical simulations” and thus “see the overall range and distribution of outcomes” in your game (Schreider, 2013, n.p.). Drawing on Schreiber’s instructional work, we can perhaps model, test, and re-develop constitutive rules that strike a better balance between strategy and luck and thus facilitate the critical play scenario we envisioned in our goals.

4. **Use of similes:** Given our findings regarding the unsuccessful use of technical language in the game elements, especially in the Software Tile’s code-side, we suggest that there is a better way to communicate complex technical concepts from software studies in the game. A solution to accomplish the balance between theme and rules is the use of figures of speech such as similes in the rule book. The use of similes will enable the “comparison of one thing with another thing of a different kind” and help create analogies to better communicate concepts such as blackboxing, UI, and algorithms (Simile, n.d.). We do not feel it is necessary to use the correct computer scientific terminology as long as the underlying logic of how we designed the game according to software studies theory is sound. The use of more colloquial language to explain

⁵² The practice of ‘spread sheeting’ a game is formally known as ‘Theorycraft’. This is a practice that emerged amongst players of MMOPG games such as *World of Warcraft* in which players employ mathematical and statistical analysis to reverse-engineer the underlying constitutive rules of a game, through which they can develop statistically substantiated strategies to win (Paul, 2011, n.p.; Nardi, 2010, p. 137).

these technologic concepts will help create an engaging learning experience for players about algorithms and software. This might also help to situate the players in the adequate mindset to relate the content offered by the game to their real context, as the language will be less abstract, and thus accomplish our value goal of generating an awareness of algorithms in everyday cultural software. To aid this process, it might be useful for us to look at introductory computer scientific pedagogical games and how their communication strategies for non-experts work.



Figure 4.8. **TerrorBull's 'Our Sonofabitch'**. This game in which players "[s]ell weapons to your favourite dictators while ignoring their crimes" (TerrorBull, 2015b, n.p.) effectively use communication cues in the copy and illustrations of the rule book to explicitly frame the game as a space to reflect critically on geo-politics. Reprinted from TerrorBull, by TerrorBull, 2015c. Retrieved from https://www.terrorbullgames.co.uk/games/our_sonovabitch_pnp_game.php

5. **Amount of information:** In order to prevent the content in the Event Cards and the Software Tiles to be skimmed instead of read attentively, we concluded that the amount of text needs to be reduced and simplified. Thereby, the consequences can be quickly foreseen in the code-side of the Tiles, and thus allow space for the player to strategize. Additionally, with this adjustment the reading time of Event Cards can be reduced, which will improve the current interruptions in the flow of the game. The key point in this part is to redesign the triggers words in both Event Cards and Tiles, so that the connection is entirely clear for the players. In the case of the Software Tiles we also noticed that due to the implicit rule that developed of not picking the Tiles up to read at close range meant that the text need to be enlarged, necessitating further simplification as well. On a design level we can also improve the Software Tiles' readability by adapting the one of our discarded design concepts (Fig. 4.9.) in which the trigger words were bigger and the classification of information is indicated more clearly. Finally, by designing a clear connection between the trigger words in the Event Cards and the Software Tiles, our aim is to facilitate the reading of the code and producing more discernible outcomes to the action of publishing apps.

Figure 4.9. First wireframes for Prototype 2 of the code-side of the Software Tile.



6. **Interaction:** As some players pointed out, the game needs more spaces to interact with other players. Therefore, we have considered that some of the Event Cards need to have consequences that allow players to interact more. For example: allow one player to punish, or reward, another player depending on the nature of the event in the Event Card. These improvements will not only enhance the interaction among the players, but also offer space to affect the outcomes of the game for other players, and thus also create a space for more strategic play.

Conclusively, it should be noted that the improvements we recommend are all garnered from the perceptible failures of *Unveiling Interfaces* during our playtests evaluation. Yet within the iterative cycle of game design, failures in a playtest session is meant to offer insights into how to improve the game—rather than offer conclusive judgements. Thus, while our initial MVP prototype might have not succeeded in reaching our design and values goals, we argue that the implementation of the improvements outlined above in a successive iteration (as well as the improvements outlined earlier in the chapter, such as simplifying the rules) will allow the game to reach these goals. We thus hold that it is would be premature to make any conclusive statements about *Unveiling Interfaces*' success or failure as a critical board game or research project.

CHAPTER 5

Limitations, Reflections, and Conclusions

1. Limitations

Before our final conclusions it is necessary to acknowledge some limitations of our research, both in terms of practice and theory. Firstly, we concede that we could assuredly have refined the evaluation design and piloted the evaluation process and guides more thoroughly. As Flanagan does not give a guide for evaluating value goals, we drew primarily from traditional playtest guides focused on evaluating design goals. There is thus perhaps a better method for evaluating the critical play elements of our game—as can perhaps be found in impact evaluation methodologies. Likewise, a larger and more diverse sample size would have allowed us to make definite, and perhaps even generalizable, conclusions about our game as a tool for fostering algorithmic literacy. Yet, as already argued, the question of designing a more rigorous scientific evaluation might be much more relevant in terms of assessing a final version of the game, as the conclusions of the prototype playtests pointed us towards the need for at least one more design iteration. This perhaps then touches on the biggest limitation of this research project.

Though arriving at an MVP in our prototype allowed us to achieve certain research insights that could be applied towards answering our research question, we would have preferred to arrive at the resolution of the iterative design process with the completion of a final product (perhaps even publishing it in order to fulfil the transformative agenda of this research project). In traditional game development the iterative process of design, testing, and revision is an ever-ongoing process of refinement, only constrained by product release dates and available resources (Fullerton, et. al., 2004, p. 23). Within these constraints professional board-game designers state that the game development process often take between 6 months to 2 years⁵³. Thus, even after an initial extension for our thesis deadline, with six

⁵³ Matthew Leacock, renowned designer of the board game *Pandemic*, states in an interview with *Time* magazine: “My first game, a racing game I brought to SPIEL took about six years, because I didn’t really know what I was doing. *Pandemic* took about three years. Now, it typically takes me about six months to 12 months to develop a game, and then the publisher takes another 6 to 12 months to produce it and manufacture it” (Fitzpatrick, 2016 June 30, n.p.). This is a time-frame not far off from those described by other board game designers such as Garret Rempel (Rollins, July 31, 2017, n.p.) or Teale Fristoe (Fristoe, 2013 August 19, n.p.).

months to complete this research project we did not have time for a next design iteration following our first round of external playtesting.

This shortfall in our practice-led research method can perhaps be ascribed to our inexperience in game designing and our consequent underestimation of the number of iterations needed—and how much time it would take—to arrive at a final product. Yet this particular limitation can also be attributed to the context in which the research was conducted. As the game design process was structured within an academic Master thesis, there was a preoccupation with producing the written research outputs. This perhaps produced an imbalance towards the ‘left side’ (academic research) of the practice-led research iterative cyclical model (Fig. 2.1.). Consequently, *Unveiling Interfaces* can benefit—both as an actualisable algorithmic literacy initiative and as a research project—from a successive iteration outside of its current academic context.

Our research also afforded us insight into the limitations of software studies within a practice-led research method. We found that though software studies theory was immensely helpful in developing the mechanics and conceptual frame for the game, as source for the content of the game it obscured rather than helped us communicate critical insights to a technically and academically non-expert public. Though a bridge between cultural studies and computer science—software studies still importantly produces knowledge for an expert academic public. This, as noted already, is an issue that we would need to address in successive design iterations.

Lastly, as has emerged from our bibliography in which we disproportionately cite text from the MIT Press in Cambridge, Massachusetts, software studies can be argued to reflect a particularly Northern American and Western European technocultural context and perspective (despite Manovich’s Russian origins). In this our use of concepts such as ‘software culture’ and ‘algorithmic culture’ should thus not be seen as universalistic. The same critique can likewise be applied to the foundational premises for this research—as our literature review draw on publications originating primarily in the US (with the exception, perhaps, of the WWWF publication which is oriented towards international perspectives). Consequently, taken from these publications our use of the concept ‘public’ is limiting and should be scrutinized further in successive research. As these texts underpin our research, we consider that there is a need to view the theoretical premises of this thesis within this limitation.

The critique above should however also be taken in context of our selective reading of software studies texts. There are possibly publications that address the limitations above that we have not had the opportunity or time to read. Moreover, other key software studies texts such as Wendy Chun's *Programmed Visions: Software and Memory* (2011) and Adrian Mackenzie's *Machine Learners: Archaeology of a Data Practice* (2017) (which would have given an updated perspective on current technological trends from a software studies perspective) are glaring omissions from our bibliography.

2. Reflections

This project in its current state lends itself to further development. Our primary reflection is thus that there is space for further iteration that might even take completely different approaches to the game. For example, after the playtest sessions we noted that with minor improvements the board game can potentially be used in educational settings. This project could thus be useful in code-literacy or computational thinking curricula to incorporate a humanistic component to STEM education. This is central to our conception of an algorithmic literacy as necessitating not only technical skills, but socio-cultural competencies that could help create more conscientious technology users and producers. This conclusion is drawn from the observations in Playtest 3, in which all the participants had some knowledge of computer science. These participants were thus prone to read the consequences of the Event Cards more carefully in relation to the Software Tiles.

Furthermore, in order to be adapt the game for pedagogical use, a possible next iteration could be accompanied by an educational guide for an educational facilitator to use, guiding players through the game and helping them decode the game content as well as prompting debate about topics that might arise. This proposal follows the examples of Crocco's (2009) codified modification of *Monopoly* which was conducted as a classroom exercise, as well as TerrorBull's *IAAWGMOOH* which was to be conducted in SOAS undergraduate classrooms. For Crocco it was "the questions and discussion after the game" that "enabled students to synthesize their varied experiences into a new consensus about social mobility" (2011, p. 35). While in the case of *IAAWGMOOH*, the role of the moderator is to take part of the game and provide players with "various tools with which they can interfere with the

[agricultural] market” simulated by the game (TerrorBull, 2015c, n.p.). In this way, the moderator can generate debate through the game mechanics they control. These examples demonstrate that a moderator can help overcome knowledge gaps, and prompt decodification, in a successive iteration of *Unveiling Interfaces*.

Yet more than exploring hypothetical iterations, reflecting on the potential to develop *Unveiling Interfaces* further speaks to the necessarily transformative impetus of our research. It is thus that we do not merely speculate about future iterations, but have been actively working towards extend this project beyond the current academic research context. We have already implemented some of the academic confines and iterate it after this thesis. Thus, we have implemented some of these insights and aforementioned improvements in a funding application for a next iteration through the Mozilla Creative Media Awards 2018⁵⁴. Barring the success of this application we aim to continue looking for funding opportunities.

It should be noted that the possibility of continuation of the project in successive practical iterations is a primary advantage of the practice-led research method. For as Sullivan pose, “an important part of practice-led research involves making sense of the information collected so that it can be translated into interpretive forms able to be communicated to others” (2009, p. 50). It is thus in the synthesis of practice and academic research that the value of this method to our research question lies, as Smith and Dean writes: practice-led research can help in “generating new pedagogical tools and shifting educational paradigms” (2009, p. 9). Part of this thesis research will thus be to continue developing *Unveiling Interfaces* as a possible initiative for algorithmic literacy.

Looking back on the use of the term ‘algorithmic literacy’ both in our literature review and consequently by us throughout this thesis, we argue that it should not be taken to mean a literacy that is separable from other ‘new literacies’. As Kahn and Keller write from a critical pedagogical position too: “Theorizing a democratic and multicultural reconstruction of education in the light of Freirean and Illichian critique demands that we develop theories of the multiple literacies needed to empower people in an era of expanding media, technology and globalization” (2007, p. 440). For us ‘algorithmic literacy’, as we’ve understood and defined it, should fit within this

⁵⁴ The Mozilla’s call for ‘Art and Advocacy Exploring Artificial Intelligence’ can be accessed in the following link: <https://blog.mozilla.org/blog/2018/06/04/mozilla-announces-225000-for-art-and-advocacy-exploring-artificial-intelligence/>

multiplicity of literacies—especially as algorithms in their materiality are not separable from software, data, information systems, or code. We thus advocate for an approach to the further development of algorithmic literacy that looks at data literacy, code literacy, or information literacy (while also arguing for the inverse should be practiced as well). As D'Ignazio & Bhargava write, “data literacy has only sparingly tackled algorithmic literacy” as this “nascent and the field has yet to articulate a well-formed pedagogical approach” (2015, p. 3). As such, we have already been put in contact with the project manager of DOKK1's Data Democracy project to potentially collaborate on their data literacy initiatives with our ‘algorithmic literacy’ project in 2019. Likewise, for our proposed successive iteration in the Mozilla Creative Awards application—we have argued that our project falls under Mozilla's internet health concerns in terms of ‘web literacy’—stating that:

As the web is increasingly structured by [Machine Learning] algorithms, web literate individuals need algorithmic literacy if they are to read and understand the mechanics and structures that shape their experience of the web. For example, Alan November writes in his book *Web Literacy for Educators* that it is important for web users to know how results from a search engine might be algorithmically ranked and filtered based on things such as political censorship related to a user's geo-location. (Odendaal & Zavala, 2018, p. 5)

It also should be noted that while we could not make conclusions about whether our project would be able to contribute to the algorithmic literacy of players—during the design and research process we as researchers do feel we developed a degree of algorithmic literacy ourselves. When we presented our thesis proposal to a panel of professors at Aalborg University, we were asked whether we as humanities students considered ourselves to have algorithmic literacy. Unsure of how to answer the question at the time, we now feel able to better answer in the affirmative. For without too much presumption, both authors consider that the process of doing this academic research, as well as codifying that research into the procedural system of a game, has given us a depth of awareness and an informed critical stance towards algorithmic systems and how they work within the software we ourselves use daily.

On this point, we recognize the need to reflect on our own compliance with the software ecosystems we critique, while writing this thesis. Microsoft and Apple products were used respectively by the researchers, while the greatest portion of our

collaborative work was done through Google Drive and Docs. It is especially given that our argument of algorithmic ‘grammars of action’ relates to the insights of Geoff Cox on his own usage of a “less prescriptive” simple rich-text editor, that this becomes a point of reflection. As he writes: “[i]n word-processing a text with Word, the writer becomes part of the machine, thoroughly embedded in the choice of computer and software program” (2010, p. 135). Moreover, conducting our research through software systems such as Google and Google Scholar reflects Berry’s insistence that we need to pay attention to how “code is infiltrating the academy itself” (Berry, 2011, p. 6). As Fuller writes: “in a sense, all intellectual work is now ‘software study’, in that software provides its media and its context” (cited in Berry, 2011, p. 6). Though insights from these realizations can take up a thesis in itself, we offer here merely a self-conscious admission of complicity not unlike that of Kittler when he confesses to using software to write the essay ‘There is No Software’ (Cox, 2010, p. 135).

Lastly, we want to reflect on the research praxis as we gained important insight into the challenges and value of doing practice-led research. Firstly, we can comment on how within a practice-led research method we, as humanities researchers, were propelled into a transdisciplinary research space. This afforded a unique framework for academic investigation, as the project was a fundamental part of the research design both in terms of processes and outcomes. While selecting a project format—critical board game design—that we considered could be feasibly realized within our cumulative skillsets⁵⁵, conducting research using these non-academic skills presented a challenge. In this sense, the work constituting Chapter 3 necessitated a rhizomatic process that engaged game design, software studies, critical design, and communication while throughout responding to our research question. Though challenging, applying theory to practice also allowed us to move beyond concepts that are academic, and evaluate if and how they can be applied as practical knowledge through game design as a space of exploration. On the other hand, the process also compelled us to explore game design in a more thorough manner as a space for expression and for applying theoretical concepts as game mechanics as we worked towards fulfilling our design *and* value goals. The method we employed was experienced as extremely enriching and stimulating, offering a

⁵⁵ Communication, graphic design, game design, UX design, copywriting, educational training, platform design, and others.

space for exploration that can be beneficial to researchers from any discipline. As in Smith & Dean argue, drawing on Barbara Bolt:

There can arise out of creative practice 'a very specific sort of knowing, a knowing that arises through handling materials in practice' (Bolt 2007: 29). This is what she means by 'praxical knowledge' (Bolt 2007: 34); its insights, she argues, can induce 'a shift in thought' (Bolt 2007: 29). (2009, p. 6)

3. Conclusions

With this thesis we intended to concatenate the different disciplines of critical theory, software studies, critical pedagogy, critical play, and game design to answer the call for a public algorithmic literacy. As our problem formulation in the literature review outlined, algorithms have increasingly been delegated to social activities are traditionally considered the work of culture, while this work they do is obscured and hidden from public scrutiny through the black boxes of their interfaces. Given the perceptible dangers of biased and misbehaving algorithms becoming new cultural power-brokers exercising a secondary agency on behalf of their producers, there has consequently been a call for public algorithmic literacy initiatives to empower those who engage with these systems in their everyday lives. As set out in Chapter 1 of this thesis, such an algorithmic literacy can consequently be provisionally defined as: necessarily citizen-centric; fostering of an awareness of algorithms in everyday life; not contingent on technical competencies; promoting a critical stance towards algorithmic technologies. Following this synthesized definition, we argue that this definition has at its centre an understanding of literacy as including socio-cultural competencies, as opposed to consisting merely of technical skills.

Given our working definition for algorithmic literacy, throughout the first two chapters of this thesis we postulate how software studies can contribute to algorithmic literacy initiatives. It is especially in software studies digital materialist approach that it provides a critique of algorithms as more than just theoretical constructs or models. Rather, software studies regards algorithms as materialized in software and constituting of a complex socio-technical assemblage. Moreover, in its focus on software, it also offered a useful theoretically tool to critically investigate the spaces where the public interact with algorithms, namely software interfaces. We thus set out to answer the following research question: **How can software studies contribute to algorithmic literacy?**

Given our aim to contribute to a public algorithmic literacy aimed at social empowerment, we decided to frame our research within the transformative concerns of critical theory and critical pedagogy. Hence, we state that our use of the term ‘critical stance’, which underscores our definition of algorithmic literacy, stems from ‘critical theory’ in which ‘thinking critically’ is considered a reflection on the “relation between individual social transformation, personal and political emancipation” (Griffin, 1998, n.p.). An algorithmic literacy, within this context, thus responds to a necessary ‘concientization’, a pedagogical action that through the development of literacy awakens a “critical consciousness” that can lead to the expression of social discontent and consequent empowerment (Freire & Macedo, 2005, p. 36, Freire 2005, p. 40). This is then opposed to an algorithmic literacy merely focused on technical competencies, which would not respond to the problematic which prompted the call for a public algorithmic literacy in the first place. Therefore, we argued that algorithmic literacy includes reading algorithms within their socio-technical materialization, employing software studies as a ‘critical theory of software’.

By framing our theoretical approach to algorithmic literacy within a practice-led research method we aimed to answer our research question as this would allow us to translate theoretical insights from software studies into a practical project accessible to the public. Our chosen practice was critical board game design, as motivated in Chapter 2. Drawing on critical play and game design principles we proceeded to develop *Unveiling Interface*, a board game aimed at creating a play environment where members of the public can reflect on and discuss the unseen algorithms in their everyday lives, and thus improve their algorithmic literacy. Within a practice-led research method, designing this board game allowed us to develop our theoretical research through the process of game design. However, for the practical components we employed Mary Flanagan’s approach to critical play, while also drawing on traditional game design methods. It is especially a critical play approach that afforded a space for social critique, as well as critical pedagogy. Following Francesco Crocco’s Freirean approach to game design, throughout Chapter 3 we perform the codification of software studies theory into our game mechanics and elements.

Chapter 3 consequently consists of our development of a software studies codified board game. This aligned with our game’s value goals, which were derived from our criteria for fostering algorithmic literacy in players. *Unveiling Interfaces* thus

afforded a framework within which to develop a software studies digital materialist critique of algorithms. This consisted of critique of how algorithms are materialized in the source-code of software, blackboxed through different layers of software interfaces, and compiled as socio-technical assemblage. Through framing this critique within a critical theory approach, we could also reflect on how algorithms in their materiality exercise ideological power within their social contexts through hiding their mechanisms under obscuring commodified UIs. Through this, and the formulating of design and value goals for the *Unveiling Interfaces*, we aimed to answer our secondary research questions:

RQ1: How does software studies foster a critical stance towards algorithms materialized in cultural software?

RQ2: How can insights from software studies be implemented in an algorithmic literacy initiative?

The practice-led research method allowed us to codify software studies theory into a critical board game designed according to our algorithmic literacy value goals. Through this, and the rhizomatic academic research it entailed, we conclude that we did answer RQ1 throughout Chapter 3. However, we only arrived at an MVP prototype for *Unveiling Interfaces* that did not satisfy our design and value goals after the evaluation process described in Chapter 4. We consequently feel that we did not satisfactorily answer RQ2. It must however be stated that we do not consider this to indicate failure of this research project. Game design is importantly part of an iterative process that requires designers to revise their design until it meets the value and design goals set out (Flanagan, 2009, p. 258). At the end of Chapter 4, and throughout this Chapter, we propose different ways in which *Unveiling Interfaces* can be improved to answer RQ2 satisfactorily. From the insights we did get regarding RQ2, it can be argued that critical play board game design hold potential to answer this research question in future iterations. Moreover, we hold that the format of critical board game design still offers an optimal medium to communicate humanistic research outside academia, and thus achieve the transformative outcome of critical theory and pedagogy in applying software studies. And finally, we consider that the strategy of designing with an ‘analogue’ medium to address abstract entities such as algorithms offers an interesting and useful contribution to the call for algorithmic

literacy. The design space for algorithmic literacy initiatives demonstrated by *Unveiling Interfaces* is immense, and it represents an exciting opportunity to deploy research infused through humanities in a way that appeal to a broader public audience.

Conclusively, though we haven't succeeded in proving our hypothesis, we do not outright reject it. We still hold that through developing a critical board game applying software studies theory, we can contribute to a public algorithmic literacy. Yet future research needs to be conducted to achieve this. Adapting our prototype into new research projects as already suggested is one way of doing this. One direction would be to adapt the game mechanics and design approaches explored in this project into a new frame or medium, as we suggest to do in the Mozilla Creative Awards application. An alternative direction would be to develop *Unveiling Interfaces* further as a pedagogical tool for computer science classrooms, adding a cultural and humanistic component to such curricula. This hints at a rich field to explore in terms of expanding coding literacy and associated fields through the introduction of software studies theory. This project in its current state can be reformulated in a number of ways, and we would not be opposed to other researchers or practitioners taking further imitative with *Unveiling Interfaces*. Future work could easily expand from the prototype we have developed, with the solutions we have proposed serving as a prompt for improvements. Once a second iteration is completed, there are boundless research applications for such a game—especially in developing more thorough methods of evaluation that might answer different research questions.

Beyond the project itself, this research can perhaps prompt further investigation (and perhaps intervention) in the topic of public algorithmic literacy. As we have made clear, it is a concept that is not well explored in literature and which could benefit profoundly in theory and practice from further research. There is much more that can be added to this topic from a literacy study perspective, from a more comprehensive critical pedagogical approach, or from associated fields of 'new literacies' that are more developed (such as data literacy). We believe that the topic of algorithmic literacy is extremely pertinent given current techno-cultural realities that publics across the world face. On this note, it should be added that as the authors of this thesis come from post-colonial academic and cultural backgrounds, the topics explored in this research can be further explored from a digital post-colonial, or other alternative perspective. This would go towards mitigation the

already stated limitation of software studies as representing a European and Northern American perspective.

On a final note, as the purpose of this research was not simply to satisfy an academic curiosity, but rather to ostensibly contribute to a public algorithmic literacy—responding to the problematic as set out in our literature review—we aim to continue developing this project after the conclusion of this particular research project. The continuation of this project will allow us to fulfil the intended transformative outcome of this thesis as we work onwards to a final product. In this we hope to contribute towards the call for algorithmic literacy, as the necessity for public empowerment increases as more complex and inscrutable machine learning and other algorithms keep being developed and deployed into our daily digitally mediated lives.

BIBLIOGRAPHY

- Andrew-Gee, E. (2018 April 10). Your smartphone is making you stupid, antisocial and unhealthy. So why can't you put it down!? In *The Globe and Mail*. Retrieved August 1, 2018 from <https://www.theglobeandmail.com/technology/your-smartphone-is-making-you-stupid/article37511900/>
- Angwin, J. & Mattu, S. (2016 September 20). How We Analyzed Amazon's Shopping Algorithm. In *ProPublica*. Retrieved April 28, 2018 from <https://www.propublica.org/article/how-we-analyzed-amazons-shopping-algorithm>
- Angwin, J., Tobin, A., & Varner, M. (2017 October 28). Facebook (Still) Letting Housing Advertisers Exclude Users by Race. In *ProPublica*. Retrieved April 28, 2018 from <https://www.propublica.org/article/facebook-advertising-discrimination-housing-race-sex-national-origin>
- Assemblage. (n.d.). *English Oxford Living Dictionaries*. Retrieved August 1, 2018 from <https://en.oxforddictionaries.com/definition/assemblage>
- Bad News. (n.d.). *How does Bad News work?*. Retrieved July 23, 2018 from <https://www.getbadnews.com/#intro>
- Baker, P., & Potts, A. (2013). Why do white people have thin lips?' Google and the perpetuation of stereotypes via auto-complete search forms. *Critical Discourse Studies*, 10(2), 187–204.
- Beecher, F. (2010) UI Guidelines for Skeuomorphic Multi-Touch Interfaces. Retrieved July 17 2018 from <https://web.archive.org/web/20140208145437/http://userexperience.evantageconsulting.com/2010/11/ui-guidelines-for-skeuomorphic-multi-touch-interfaces/>
- Berry, D. M. (2011). *The Philosophy of Software: Code and Mediation in the Digital Age*. Hampshire: Palgrave Macmillan.
- Berry, D. M. (2014). *Critical Theory and The Digital*. London: Bloomsbury.
- Berry, D. M., van Dartel, M., Dieter, M., Kasprzak, M., Muller, N., O'Reilly, R., & de Vicente, J. L. (2012). *New Aesthetic, New Anxieties*. Rotterdam: V2_. Retrieved April 28 2018 from <http://v2.nl/publishing/new-aesthetic-new-anxieties>
- Bhargava, R., Kadouaki, R., Bhargava, E., Castro, G., & D'Ignazio, C. (2016). Data Murals: Using the Arts to Build Data Literacy. In *The Journal of Community Informatics*, 12(3). Retrieved April 18, 2018 from <http://ci-journal.org/index.php/ciej/article/view/1276>
- Bogost, I. (2009 May 7). Persuasive Games: Gestures as Meaning. In *Gamasutra*. Retrieved April 18, 2018 from https://www.gamasutra.com/view/feature/4064/persuasive_games_gestures_as_.php?print=1
- Bogost, I. (2005). Procedural Literacy: Problem Solving with Programming, Systems, and Play. In *Journal of Media Literacy*, 52(1–2), 32–36.
- Bohman, J., & Rehg, W. (2017). Jürgen Habermas. In *The Stanford Encyclopedia of Philosophy, Fall*. Retrieved March 20, 2018 from <https://plato.stanford.edu/archives/fall2017/entries/habermas/>

- Bridle, J. (2017 November 6). Something is wrong on the internet. In *Medium*. Retrieved August 1, 2018 from <https://medium.com/@jamesbridle/something-is-wrong-on-the-internet-c39c471271d2>
- Bridle, J. (2018 June 17). How Peppa Pig became a video nightmare for children. *The Guardian*. Retrieved August 1, 2018 from <https://www.theguardian.com/technology/2018/jun/17/peppa-pig-youtube-weird-algorithms-automated-content>
- Bronner, S. (2011). *Critical Theory : A Very Short Introduction*. New York: Oxford University Press.
- Brown, A. R. & Sorensen, A. (2009). Integrating Creative Practice and Research in the Digital Media Arts. In Smith, H. & Dean R. T. (Eds.) *Practice-led Research, Research-led Practice in the Creative Arts*, pp. 153-166
- Bryman, A. (2012). *Social Research Methods* (4th edition). Oxford: Oxford University Press.
- Busl, G. (October 19, 2015). Humanities Research is Groundbreaking, Life-changing... and Ignored. In *The Guardian*. Retrieved March 20, 2018 from <https://www.theguardian.com/higher-education-network/2015/oct/19/humanities-research-is-groundbreaking-life-changing-and-ignored>
- Caplan, R., & Reed, L. (2016). Who Controls the Public Sphere in an Era of Algorithms? In *Data & Society Research Institute*. Retrieved March 18, 2018 from https://datasociety.net/%0Apubs/ap/CaseStudies_PublicSphere_2016.pdf
- Casemajor, N. (2015). Digital Materialisms: Frameworks for Digital Media Studies. Westminster Papers in *Culture and Communication*, 10(1), 4–17.
- Chander, A. (2017). The Racist Algorithm. *Michigan Law Review*, 115(6), 1023–1046.
- Chester, M. D. (2016). 2016 Massachusetts Digital Literacy and Computer Science (DLCS) Curriculum Framework. In *Massachusetts Department of Elementary and Secondary Education*. Retrieved April 28, 2018 from <http://www.doe.mass.edu/frameworks/dlcs.pdf>
- Cohen, D., & Crabtree, B. (2006). *Qualitative Research Guidelines Project*. Retrieved March 20, 2018 from <http://www.qualres.org/HomeCrit-3518.html>
- Cox, G. (2010). *Antithesis: The Dialectic of Software Art*. Aarhus: Aarhus University. Retrieved August 1, 2018 from https://monoskop.org/images/8/89/Cox_Geoff_Antithesis_the_Dialectics_of_Software_Art.pdf
- Cox, G. & Ward, A. (2009). Perl. In M. Fuller (Ed.), *Software Studies: A Lexicon* (pp. 207–213). Cambridge, Massachusetts: MIT Press.
- Cramer, F. (2008). Language. In M. Fuller (Ed.), *Software Studies: A Lexicon* (pp. 168–174). Cambridge, Massachusetts: MIT Press.
- Cramer, F., Fuller, M. (2008). Interface. In M. Fuller (Ed.), *Software Studies: A Lexicon* (pp. 149–153). Cambridge, Massachusetts: MIT Press.
- Crocco, F. (2011). Critical Gameplay Pedagogy. In *The Radical Teacher*, 91(Fall), pp. 26-41.
- Data-Pop Alliance. (2015). *Beyond Data Literacy: Reinventing Community Engagement and Empowerment in the Age of Data*. Retrieved May 25, 2018 from <http://datapopalliance.org/wp-content/uploads/2015/11/Beyond-Data-Literacy-2015.pdf>
- Diakopoulos, N. (2013). *Algorithmic Defamation: The Case of the Shameless Autocomplete*. Retrieved March 20, 2018 from <https://towcenter.org/algorithmic-defamation-the-case-of-the-shameless-autocomplete/>

- Diakopoulos, N. (2014). *Algorithmic Accountability Reporting: On the Investigation of Black Boxes*. Tow Centre for Digital Journalism: A Tow/Knight Brief, 1–33. Retrieved April 18, 2018 from http://towcenter.org/wp-content/uploads/2014/02/78524_Tow-Center-Report-WEB-1.pdf
- D'Ignazio, C., & Bhargava, R. (2015). Approaches to Building Big Data Literacy. In *Bloomberg Data for Good Exchange Conference*. Retrieved April 28, 2018 from https://dam-prod.media.mit.edu/x/2016/10/20/Edu_D'Ignazio_52.pdf
- Dourish, P. (2016). Algorithms and their Others: Algorithmic Culture in Context. *Big Data & Society*, 3(2), pp. 1-11.
- ECDL Foundation. (2015). Computing and Digital Literacy Call for a Holistic Approach. In *ECDL Foundation*. Retrieved April 18, 2018 from http://ecdcl.org/media/position_paper_-_computing_and_digital_literacy.pdf
- Epic Panda. (2018). Epic Panda. In *Facebook*. Retrieved July 18, 2018 from https://www.facebook.com/pg/epicpandaepic/about/?ref=page_internal
- Erlhoff, M., Marshall, T. (2008). *Design Dictionary: Perspectives on Design Terminology*. Basel: Birkhauser Basel.
- European Commision. (2017). *Open Call for Tenders: Algorithmic Awareness Building Study*. Retrieved April 20, 2018 from <https://ec.europa.eu/futurium/en/blog/open-call-tenders-algorithmic-awareness-building-study>
- Finn, E. (2017). *What Algorithms Want: Imagination in the Age of Computing*. Cambridge, Massachusetts: MIT Press.
- Fitzpatrick, A. (2016). This Board Game Designer Isn't Sorry About Taking a Big Risk. In *Time*. Retrieved July 23, 2018 from <http://time.com/4385490/board-game-design/>
- Flanagan, M. (2010). Creating Critical Play. In *Artists Re:Thinking Games*, 49–53.
- Flanagan, M. (2009). *Critical Play: Radical Game Design*. Cambridge, Massachusetts: MIT Press.
- Flusser, V. (1983). Towards A Philosophy of Photography. Retrieved April 19, 2018 from <http://x/www.altx.com/remix.fall.2008/flusser.pdf>
- Foer, F. (2017). *World Without Mind: The Existential Threat of Big Tech*. New York, NY: Penguin Press.
- Franco, D. (2015). *Critical Theory: The Key Concepts*. Oxon: Routledge.
- Freire, P. (2005). *Pedagogy of the Oppressed* (30th Anniv). London: Continuum.
- Freire, P., & Macedo, D. (2005). *Literacy: Reading the Word and the World. Contemporary Sociology*. London: Routledge.
- Fristoe, T. (2013 August 19). The Cost of a Board Game: Time. In *Nothing Sacred Games*. Retrieved July 25, 2018 from <http://nothingsacredgames.com/the-cost-of-a-board-game-time/>
- Fuller, M. (2003). *Behind the Blip: Essays on the culture of Software*. Brooklyn: Autonomedia.
- Fuller, M. (2008). Introduction. In M. Fuller (Ed.), *Software Studies: A Lexicon* (pp. 1–15). Cambridge, Massachusetts: MIT Press.
- Fuller, M. (2008 September 13). "Software Studies" book series @ MIT Press. In *Software Studies Initiative*. Retrieved August 1, 2018 from <http://lab.softwarestudies.com/2008/07/software-studies-book-series-mit-press.html>

- Fuller, M., Goffey, A. (2012). *Evil Media*. Cambridge, Massachusetts: MIT Press.
- Fullerton, T., Swain, C., Hoffman, S. (2004). *Game Design Workshop: Designing, Prototyping, & Playtesting Games*. San Francisco: CMP Books.
- Gillespie, T. (2014). The Relevance of Algorithms. In *Media Technologies* (pp. 167–194). Cambridge, Massachusetts: The MIT Press.
- Gillespie, T., Boczkowski, P. J., Foot, K. A. (2014). Introduction. In T. Gillespie, P. J. Boczkowski, K. A. Foot (Eds.), *Media Technologies: Essays on Communication, Materiality, and Society* (pp. 1-21). Cambridge, Massachusetts: MIT Press.
- Gillespie, T., & Seaver, N. (2016). *Critical Algorithm Studies: a Reading List*. Retrieved March 20, 2018, from <https://socialmediacollective.org/reading-lists/critical-algorithm-studies/>
- Giroux, H. A. (2005). Introduction Literacy and the Pedagogy of Political Empowerment. In *Literacy: Reading the Word and the World* (pp. 1–20). London: Routledge.
- Griffin, C. (2003). Critical thinking and critical theory in adult education. In *1988 Conference Proceedings, pp. 176-180 ã SCUTREA 1997*. Retrieved August 1, 2018 from <http://www.leeds.ac.uk/educol/documents/00002723.htm>
- Goffey, A. (2008). Algorithm. In M. Fuller (Ed.), *Software Studies: A Lexicon* (pp. 15–20). Cambridge, Massachusetts: MIT Press.
- Grace, L. (2014). Critical Gameplay. In *Designing Games for Ethics*, (December), pp. 128–141.
- Granieri, G. (2014). *Algorithmic culture*. "Culture now has two audiences: people and machines": A conversation with Ted Striphas. Retrieved June 20, 2018, from <https://medium.com/futurists-views/algorithmic-culture-culture-now-has-two-audiences-people-and-machines-2bdaa404f643>
- Griffith, E. (2017 December 16). The Other Tech Bubble. In *Wired*. Retrieved July 15, 2018, from <https://www.wired.com/story/the-other-tech-bubble/>
- Gustavsson, B. (2007). *The Principles of Knowledge Creation: Research Methods in the Social Sciences*. Cheltenham: Edward Elgar Publishing Limited.
- Hamilton, K., Karahalios, K., Sandvig, C., & Eslami, M. (2014). A path to understanding the effects of algorithm awareness. In *Proceedings of the Extended Abstracts of the 32nd Annual ACM Conference on Human Factors in Computing Systems - CHI EA '14*, 631–642.
- Hanemann, U. (2015). Lifelong literacy: Some trends and issues in conceptualising and operationalising literacy from a lifelong learning perspective. In *International Review of Education*, 61(3), 295–326.
- Harpham, G. G. (2005). Beneath and beyond the "Crisis in the Humanities". In *New Literacy History* 36(1), pp. 21-36.
- Harvey, F. (2018). Critical GIS: Distinguishing critical theory from critical thinking. In *The Canadian Geographer*, 62(1), pp. 35-39.
- Held, D. (2004). *Introduction to Critical Theory: Horkheimer to Habermas*. Cambridge: Polity Press.
- Horkheimer, M., & Adorno, T. W. (2002). *Dialectic of Enlightenment: Philosophical Fragments*. California: Stanford University Press.
- Jameson, F. (1991). *Postmodernism, or, The Cultural Logic of Late Capitalism*. New York: Duke University Press.
- Jameson, F. (2011). *Representing Capital: a Commentary on Volume One*. New York: Verso.

- Kahn, R. & Kellner, D. (2007). Paulo Freire and Ivan Illich: technology, politics and the reconstruction of education. In *Policy Futures in Education*, 5(4), pp. 431-448.
- Karahalios, K. (2014). Algorithm Awareness: How the news feed on Facebook decides what you get to see. In *MIT Technology Review*. Retrieved May 25, 2018 from <https://www.technologyreview.com/s/531676/algorithm-awareness/>
- Kincheloe, J. L., & McLaren, P. (2002). Rethinking critical theory and qualitative research. In Y. Zou & E. (Henry) T. Trueba (Eds.), *Ethnography and Schools: Qualitative Approaches to the Study of Education* (pp. 87–138). Boston: Rowman & Littlefield.
- Kircher, M. M. (2017 November 9). Sean Parker: We Built Facebook to Exploit You. In *New York Magazine*. Retrieved August 1, 2018 from <http://nymag.com/selectall/2017/11/facebook-sean-parker-talks-about-psychology-of-hooking-users.html>
- Kitchin, R. (2017). Thinking Critically About and Researching Algorithms. In *Information, Communication & Society*, 20(1), 14–29.
- Kitchin, R. (2011) The Programmable City. In *Environment and Planning B: Planning and Design*, Vol. 38, pp. 945–51.
- Kitchin, R., & Dodge, M. (2011). *Review, Code / Space: Software and Everyday Life*. Cambridge, Massachusetts: The MIT Press. Retrieved June 18, 2018 from <https://www.technologyreview.com/s/604087/the-dark-secret-at-the-heart-of-ai/>
- Kittler, F. (1997). There Is No Software. In J. Johnston (Ed.), *Literature, Media, Information Systems: Essays* (pp. 147–155). New York: Routledge.
- Knight, W. (2017). The Dark Secret at the Heart of AI. In *MIT Technology Review*.
- Koh, P. W., & Liang, P. (2017). Understanding Black-box Predictions via Influence Functions. In *Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia*. Retrieved June 18, 2018 from <https://arxiv.org/pdf/1703.04730.pdf>
- Kohlstedt, K. (2016 January 25). Least Resistance: How Desire Paths can Result in Better Design. In *99% Invisible*. Retrieved August 1, 2018 from <https://99percentinvisible.org/article/least-resistance-desire-paths-can-lead-better-design/>
- Lankshear, C. & Knobel, M. (2003) *New Literacies: Changing Knowledge and Classroom Learning*. Buckingham: Open University Press.
- Lapowsky, I. (2018 February 12). Google Autocomplete Still Makes Vile Suggestions. In *Wired*. Retrieved March 21, 2018 from <https://www.wired.com/story/google-autocomplete-vile-suggestions/>
- Larson, J., Angwin, J., Kirchner, L. & Mattu, S. (2017 April 5). How We Examined Racial Discrimination in Auto Insurance Prices. In *ProPublica*. Retrieved April 28, 2018 from <https://www.propublica.org/article/facebook-advertising-discrimination-housing-race-sex-national-origin>
- Larson, J., Angwin, J. & Parris, T. (2016 October 19). Breaking the Black Box: How Machines Learn to Be Racist. In *ProPublica*. Retrieved April 28, 2018 from <https://www.propublica.org/article/breaking-the-black-box-how-machines-learn-to-be-racist?word=Trump>
- Latour, B. (1999). *Pandora's Hope: Essays on the Reality of Science Studies*. Cambridge, Massachusetts: Harvard University Press.
- Lecher, C. (2018 March 21). What happens when an algorithm cuts your health care. In *The Verge*. Retrieved April 28, 2018 from <https://www.theverge.com/2018/3/21/17144260/healthcare-medicaid-algorithm-arkansas-cerebral-palsy>

- Levi-Strauss, C. (1961). *Tristes Tropiques*. New York: Criterion.
- Logas, H. L. (2011). Meta-Rules and Complicity in Brenda's Train. In *Proceedings of DiGRA 2011 Conference: Think Design Play*. Retrieved from August 1, 2018 from <http://www.digra.org/wp-content/uploads/digital-library/11301.05058.pdf>
- Madrigal, A. C. (2018 March 18). What Took Facebook So Long? In *The Atlantic*. Retrieved August 1, 2018 from <https://www.theatlantic.com/technology/archive/2018/03/facebook-cambridge-analytica/555866/>
- Manovich, L. (2001). *The Language of New Media*. Cambridge, Massachusetts: The MIT Press.
- Manovich, L. (2013). *Software Takes Command: Extending the Language of New Media*. New York: Bloomsbury Academic.
- Mantzavinos, C. (2016). Hermeneutics. In *The Stanford Encyclopedia of Philosophy, Winter*. Retrieved February 20, 2018 from <https://plato.stanford.edu/archives/win2016/entries/hermeneutics/>
- Mayer, B., & Harris, C. (2010). *Libraries Got Game: Aligned Learning through Modern Board Games*. Chicago: American Library Association.
- Mertens, D. M. (2015). An Introduction to Research. In *Research and Evaluation in Education and Psychology: Integrating Diversity with Quantitative, Qualitative, and Mixed Methods* (Third edit, pp. 1–46). Thousand Oaks: SAGE.
- Monopoly Guide. (2017). *Monopoly List of Chance Cards Main Version*. Retrieved August 1, 2018 from <https://monopolyguide.com/traditional/monopoly-list-of-chance-cards-main-version/>
- Moore, M., & Tambini, D. (2018). *Digital Dominance: The Power of Google, Amazon, Facebook, and Apple*. New York, NY: Oxford University Press.
- Morrow, R. M., & Torres, C. A. (2002). *Reading Freire and Habermas: Critical Pedagogy and Transformative Social Change*. New York: Teachers College Press.
- Mozilla. (2018). *Web Literacy*. Retrieved July 25, 2018 from <https://learning.mozilla.org/en-US/web-literacy/>
- Müller, A. C., Guido, S. (2017). *Introduction to Machine Learning with Python: A Guide for Data Scientists*. Sebastopol: O'Reilly.
- Nardi, A. F. B. (2010). *My Life as a Night Elf Priest: An Anthropological Account of World of Warcraft*. Michigan: University of Michigan Press.
- Niman, N. B. (2014). *The Gamification of Higher Education: Developing a Game-Based Business Strategy in a Disrupted Marketplace*. New York: Palgrave Macmillan.
- Noble, S. (2018). *Algorithms of Oppression: How Search Engines Reinforce Racism*. New York: New York University Press.
- Noothout, L. (2017). *The Altar of the Algorithm: A Critical Design Inquiry into the Ideology of Algorithms*. Retrieved July 25, 2018 from <http://lukenoothout.com/M12-TheAltarOfTheAlgorithm.pdf>
- O'Neil, C. (2016). *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. New York: Crown.
- Odendaal, W. A., & Zavala, K. (2018). *Blackbox Mysteries: A Web-based Game for Unveiling Machine Learning Algorithms*. Unpublished funding application.

- Osoba, O. & Welser IV, W. (2017). *An Intelligence in Our Image The Risks of Bias and Errors in Artificial Intelligence*. Santa Monica: RAND Corporation. Retrieved March 20, 2018 from https://www.rand.org/pubs/research_reports/RR1744.html
- Park, J. Y. (2012). A different kind of reading instruction: Using visualizing to bridge reading comprehension and critical literacy. In *Journal of Adolescent and Adult Literacy*, 55(7), pp. 629–640.
- Pasquale, F. (2015). *The black box society: The secret algorithms that control money and information*. Cambridge, MA: Harvard University Press
- Paul, C. A. (2011). Optimizing Play: How Theorcraft Changes Gameplay and Design. In *Game Studies*. Retrieved July 23, 2018 from <http://gamestudies.org/1102/articles/paul>
- Phillies, G.; Vasel, T., 2006. *Contemporary Perspectives on Game Design: Interviews With Gaming's Greats*. G. Phillies, ed., Tempe, AZ: Third Millennium Publishing
- Pold, S. B. (2005). *Interface Realisms: The Interface as Aesthetic Form*. Retrieved August 1, 2018 from <http://pmc.iath.virginia.edu/text-only/issue.105/15.2pold.txt>
- Pold, S. B. (2008). Button. In M. Fuller (Ed.), *Software Studies: A Lexicon* (pp. 31–37). Cambridge, Massachusetts: MIT Press.
- Pold, S. B., & Andersen, C. U. (2014). Controlled Consumption Culture: When Digital Culture Becomes Software Business. In Miller P. & Matviyenko S. (Eds.), *The Imaginary App* (pp. 17-34). Cambridge, Massachusetts: MIT Press.
- Rainie, L., & Anderson, J. (2017). Code-dependent: Pros and Cons of the Algorithm Age. In *Pew Research Center*. Retrieved April 20, 2018 from <http://www.pewinternet.org/2017/02/08/code-dependent-pros-and-cons-of-the-algorithm-age/>
- Rainie, L. & Zickurh, K. (2015). Americans' Views on Mobile Etiquette. In *Pew Research Center*. Retrieved August 1, 2018 from <http://www.pewinternet.org/2015/08/26/americans-views-on-mobile-etiquette/>
- Ricoeur, P. (1990). Hermeneutics and the Critique of Ideology. In G. L. Ormiston & A. D. Schrifft (Eds.), *The Hermeneutic Tradition: From Ast to Ricoeur* (pp. 298–335). New York, NY: State University of New York.
- Roberge, J. (2011). What is Critical Hermeneutics? *Thesis Eleven*, 106(1), 5–22.
- Roberge, J. & Seyfert, R. (2016). *Algorithmic Cultures: Essays on Meaning, Performance and New Technologies: Abstract*. Retrieved April 20, 2018 from https://www.researchgate.net/publication/318585512_Algorithmic_cultures_Essays_on_meaning_performance_and_new_technologies
- Roberge, J. & Seyfert, R. (2016b). What are Algorithmic Cultures? In J. Roberge & R. Seyfert (Eds.), *Algorithmic Cultures: Essays on Meaning, Performance and New Technologies*. Oxon: Routledge.
- Rollins, B. (July 31, 2017). What to Expect When You're Making a Board Game: Time, Money, and Effort. In *Brandon the Game Dev*. Retrieved July 25, 2018 from <http://brandonthegamedev.com/lets-set-expectations-time-money-effort/>
- Romero, B. (2018). Work. In *Brenda Romero*. Retrieved July 25, 2018 from <http://brenda.games/work-1/>
- Salen, K., Zimmerman E. (2004). *Rules of Play - Game Design Fundamentals*. Cambridge, Massachusetts: MIT Press.

- Schofield, L. (2014). Critical Theory and Constructivism. *International Hospitality Research Centre Switzerland*. Retrieved April 21, 2018 from <http://www.ihracs.ch/?p=92>
- Schreider, I. (2013). Level 12.0: Game Balance. In *Game Design Concepts*. Retrieved July 23, 2018 from <https://learn.canvas.net/courses/3/pages/level-12-dot-0-game-balance>
- Sensor Tower. (2018). The Top Mobile Apps, Games, and Publishers of Q1 2018: Sensor Tower's Data Digest. In *Sensor Tower Blog*. Retrieved July 27, 2018 from <https://sensortower.com/blog/top-apps-games-publishers-q1-2018>
- Sidhu, R. (2014). Code Monkey Island - making programming child's play. In *Kickstarter*. Retrieved July 28, 2018 from <https://www.kickstarter.com/projects/rajsidhu/code-monkey-island-making-programming-chilids-play>
- Simile. (n.d.). English Oxford Living Dictionaries. Retrieved August 1, 2018 from <https://en.oxforddictionaries.com/definition/simile>
- Simonite, T. (2017 August 21). Machines Taught by Photos Learn a Sexist View of Women. In *Wired*. Retrieved April 28, 2018 from <https://www.wired.com/story/machines-taught-by-photos-learn-a-sexist-view-of-women/>
- SoftWhere. (2008). *SoftWhere 2008*. Retrieved August 1, 2018 from <http://workshop.softwarestudies.com/>
- Spector, P. (2005). Introduction to Python Programming: Course Notes. In *University of California, Berkeley: Department of Statistics*. Retrieved August 1, 2018 from <https://www.stat.berkeley.edu/~spector/python.pdf>
- Smith, H., & Dean, R. T. (2009). Introduction. Practice-led Research, Research-led Practice - Towards the Iterative Cyclic Web. In *Practice-led Research, Research-led Practice in the Creative Arts: Research Methods for the Arts and Humanities*. Edinburgh: Edinburgh University Press.
- Striphas, T. (2015). Algorithmic Culture. In *European Journal of Cultural Studies*, 18(4-5), (pp. 395-412).
- Sullivan, G. (2009). Making Space: The Purpose and Place of Practice-led Research. In *Practice-led Research, Research-led Practice in the Creative Arts: Research Methods for the Arts and Humanities*. Edinburgh: Edinburgh University Press.
- TerrorBull. (2015). About Us. In *TerrorBull*. Retrieved July 25, 2018 from <https://www.terrorbullgames.co.uk/about/>
- TerrorBull. (2015b). Our Games. In *TerrorBull*. Retrieved July 25, 2018 from <https://www.terrorbullgames.co.uk/games/>
- TerrorBull. (2015c). Our Sonovabitch. In *TerrorBull*. Retrieved July 25, 2018 from https://www.terrorbullgames.co.uk/games/our_sonovabitch_pnpgame.php
- TerrorBull. (2015d). I'm An Agricultural Worker, Get Me Out Of Here - a game for SOAS. In *TerrorBull*. Retrieved July 25, 2018 from https://www.terrorbullgames.co.uk/games/iaawgmooh_classroom_simulation_game.php
- TerrorBull. (2015e). War on Terror, The Boardgame. In *TerrorBull*. Retrieved July 25, 2018 from https://www.terrorbullgames.co.uk/games/war_on_terror_game.php
- TerrorBullGames (January 7, 2010). TerrorBull Games on BBC Games Britannia, episode 2 "Monopolies & Mergers". In *Youtube*. Retrieved July 25, 2018 from <https://www.youtube.com/watch?v=CPS28BVq7ik>
- Truscello, M. (2003). The Birth of Software Studies: Lev Manovich and Digital Materialism. In *Film-Philosophy*, 7(55).

- Tygel, A. F., & Kirsch, R. (2015). Contributions of Paulo Freire for a critical data literacy: a Popular Education Approach. In *The Journal of Community Informatics*, 12(3), 108–121.
- Urban Mediaspace Aarhus. (2015). *Dokk1 and the Urban Waterfront*. Retrieved August 1, 2018 from http://www.urbanmediaspace.dk/sites/default/files/pdf/uk_ums_haefte_2015.pdf
- UNESCO. (2005). Understandings of Literacy. In *Education for All Global Monitoring Report 2016* (pp. 147–159). Paris: United Nations Educational, Scientific and Cultural Organization.
- Van den Boomen, M., Lammes, S., Lehmann, A., Raessens, J., & Schäfer, M. (Eds.). (2009). *Digital Material: Tracing New Media in Everyday Life and Technology*. Amsterdam University Press. Retrieved April 18, 2018 from <http://www.jstor.org/stable/j.ctt46mxjv>
- Vee, A. (2017). *Coding Literacy: How Computer Programming is Changing Writing*. Cambridge: The MIT Press.
- World Wide Web Foundation (WWWF). (2017). *A Smart Web for a More Equal Future*. Retrived April 28, 2018 from http://webfoundation.org/docs/2017/07/Algorithms_Report_WF.pdf

Media

- Batura, A., Leong, S. T., Lieanata, F., Xuan, L. J., Cheng, T. Y., Heng, L., Kaur, R. (2017). *Potato Pirates*. [Board game]. Singapore: Codomo.
- DROG. (2018). *Bad News*. [Computer Software].
- Grosser, B. (2017). *Textbook*. [Computer Software]
- Klamer, R., Markham, B. (1960). *The Game of Life*. [Board game]. Danvers: Winning Moves.
- Kramer, W., Witt, H. (2008). *Master Builder*. [Board game]. Warsaw: Egmont Polska.
- Lagrange, C. & Gintili, M. (2014). *Code Cards*. [Card Game]. London: Self-published
- Leacock, M. (2008). *Pandemic*. [Board game]. Mahopec: Z-Man Games.
- Leonhard, C., Matthews, J. (2007). *1960: The Making of the President*. [Board game]. Mahopec: Z-Man Games.
- Moon, A. R. (2004). *Ticket to Ride*. [Board game]. Roseville: Days of Wonder.
- Magie, E. (1904). *The Landlord's Game*. [Board game]. Arden: Economic Game Company.
- Magie, E. & Darrow, C. (2018). *Monopoly*. [Board game]. Rhode Island: Hasbro.
- Petschel-Held, G. & Eisenack, K. (2017). *KEEP COOL: The Climate Change Game*. [Board game]. Germany: Spieltrieb.
- Romero B. (2009). *Train*. [Board game]. Self-published.
- Sheerin, A., Tompkins, A. (2014) *I'm An Agricultural Worker, Get Me Out Of Here*. [Board game]. Cambridge: TerrorBull.
- Sheerin, A., Tompkins, A. (2006). *War on Terror*. [Board game]. Cambridge: TerrorBull.
- Sidhu, R. (2014). *Code Monkey Island*. [Board game]. Brookelyn: Code Monkey Games.
- Walker-Harding, P. & Stephan, C. (2015). *Cacao*. [Board game]. Dreieich: ABACUSSPIELE.
- Ward, A. (2002). *Auto-Illustrator* [Computer software].
- Wrede, K. J. (2000). *Carcassonne*. [Board game]. Munich: Hans im Glück.