Mathematical Tools in a Digital World

7th September 2012

An investigation into their development and their impact on the wider world from the perspective of digital culture

---

by Anthony Durity
May 2012

Digital Culture /
Department of Art and Culture Studies /
University of Jyväskylä

Final year thesis, Jyväskylä 2011 – Cork 2012

From Anthony Durity
At University of Jyväskylä, central Finland
In the International Master's Degree in Digital Culture

Supervisors
(senior) Raine Koskimaa
(junior) Kristóf Fenyvesi

ex-Supervisors
Susanna Paasonen
Kimmo Lehtonen

Publishing, copyright and licensing information:

Typeset in Big Caslon (serif), Source Sans Pro (sans serif), Monaco (monospace)
Published online via https://kirjasto.jyu.fi/publish-and-buy/publishing-your-thesis

Acknowledgments: (in no particular order) Susanna Paasonen, Raine Koskimaa, Kristóf Fenyvesi: all of Jyväskylä university; Ronan Farrell friend and eagle-eyed editor; Giovanna di Rosario for showing me how and encouragement; Anne Durity and Leo Durity for unceasing support; Jonne Arjoranta for translating the abstract; Pamela Jones of Groklaw fame, Tony Wasserman of Carnegie Mellon university, Carlo Daffara of Conecta, Martin Michlmayr of Cyrius, Diana Harrelson of cyber-anthro.com: for helpful pointers, advice and information; Darrel C. Ince, Leslie Hatton & John Graham-Cumming for their timely Nature article and correspondence; to all who responded to my questionnaires and follow-up questions.

Abstract

Humanistinen tiedekunta Athenaeum PL 35 40014 Jyväskylän yliopisto
puh. 040 805 3412, 040 805 3404
humtdk@campus.jyu.fi
twitter: @jyuhum
Käyntiosoite: Seminaarinkatu 15 A-rakennus
Avoinna: ma klo 10-16 ti-to klo 9-16 pe klo 9-15

On monia näkökulmia, joista digitaalista kulttuuria voidaan tarkastella. Matematiikan ja matemaattisten työvälineiden yhteys kaikkeen digitaaliseen on yksi näistä näkökulmista. Tämä viittaa matemaattisten ohjelmien tutkimukseen. Pääasialliset lähestymistavat, joita voidaan noudattaa, ovat matematiikan harjoittaminen, soveltaminen ja koulutus. Tämä tutkielma keskittyy harjoittamiseen ja soveltamiseen, jättäen huomiotta kouluttamisen.

Tässä tutkielmassa osoitetaan miten kasvavassa määrin tietokoneistettu maailmamme vaikuttaa matematiikan harjoittamiseen. Mikään matematiikan osa-alue kuvaamisesta teoreeman todistukseen ei ole jäänyt koskemattomaksi. Esimerkkinä tästä ohjelmoin fraktaaliuniversumin osajoukon, L-systeemin, osaksi tutkielmaa. Matematiikkaa soveltavien ihmisten työ on helpottunut heidän pystyessään automatisoimaan tehtäviään, ovat he sitten insinöörejä, tutkijoita tai taiteilijoita. Olen tuonut tätä esiin keskittymällä matemaattisen taiteen viimeaikaiseen kasvuun.

Kulttuuri ja ideologia vaikuttavat kaikkiin elämän osa-alueisiin, eikä ohjelmistonkehitys ole poikkeus. Keskeisiä kysymyksiä ovat hallinta ja vapaus, suljetut järjestelmät ja läpinäkyvyys. Sillä, miten ohjelmistoja kehitetään on eettisiä/pragmaattisia seurauksia, sillä voi olla lainopillisia seurauksia, tieteellisessä käytössä matemaattisilla ohjelmistoilla on epistemologisia seurauksia, ja sen lisäksi ovat taloudelliset seuraukset. Selvitin ensin tekijänoikeuksien ja henkisen omaisuuden oikeuksien historiaa - sitten tarkastelin, miten nämä toimivat viitekehyksinä ohjelmistoalalla.

Voidakseni vertailla ilmaisia ja avoimen lähdekoodin matemaattisia ohjelmistoja suljettuihin ohjelmistoihin laadin joukon kyselylomakkeita ja käytin niihin digitaalisen etnografian metodologiaa. Tämä tarkoitti avoimen lähdekoodin Sage-projektin käyttäjien ja kehittäjien ja suljetun lähdekoodin Mathematican käyttäjien ja kehittäjien kanssa puhumista. Annoin myös yksityiskohtaisen yleiskatsauksen molemmista järjestelmistä ennen kuin analysoin saamani vastaukset.

There are many vantage points from which to gain a perspective on digital culture. The intersection of mathematics and mathematical tools with all things digital is one such vantage point. This implies an inspection of mathematical software. The broad lines that can be taken are the practice, education and application of mathematics. This thesis focuses on practice and application, disregarding education.

With regards the former it is shown how the practice of mathematics itself has been affected by the increasingly pervasive computational nature of our world. From visualisation to theorem proving no area has been untouched. I programmed a subset of the fractal universe, L-Systems, into the thesis to demonstrate this. In the latter case those who apply mathematics have been aided by the automation of their tasks: be they engineers or academics or artists. I have focused on the recent growth in mathematical art to highlight this.

Culture and ideology affect all areas of life, and the development of software is no exception. At heart are issues of control and freedom, lock-in and transparency. How one develops software can have ethical/pragmatic considerations, it can have legal ramifications, when we get into how mathematics software is used in science there are increasing epistemological issues, finally there are economic ones. First I looked backward in time to survey the history of copyright and intellectual property rights – then I considered how these frameworks operate in the software world.

In order to compare and contrast free and open source mathematical software with proprietary software I composed a set of questionnaires and used the methodology of digital ethnography. In the end this meant talking to users and developers of the open-source Sage project and users and developers of the proprietary Mathematica package. I also gave a detailed overview of both systems prior to analysing the responses.

Table of Contents

"Patterns are everywhere,
Best not to read too much into that."
Pythagoras (to an acolyte), 3rd Book of the Apocrypha Geometrica

1 Prefatory remarks

1.1 These digital times

Humanity is living in the digital age. In the span of a few generations the world has been transformed utterly by the microelectronic revolution. Most recently there has been the transition from wired to wireless – all within the space of a handful of decades – this means that the very atmosphere is suffused with digital information. Gadgets that were the fodder of the wildest pulp science fiction mere decades ago have now become ubiquitous, everyday, commonplace. Every era has a beat [Luhmann, 1982, cf.], [1] to which it thrums and hums. This characteristic feature eventually becomes the calling card of that era. The Renaissance and the Enlightenment being two periods that spring to mind. More parochially the Victorian era in Britain or the Gilded Age of the United States or the Belle Époque era of continental Europe come to mind; less parochially[Vattimo, 1988, cf.] there is the label Modernity.

A characteristic feature of the contemporary world is what is termed churn in technical circles, forced obsolescence and so forth. With what may only be due to the perspective of being in the now – with the (dis)advantages of looking back over centuries and all the loss of detail that entails – the past seems to have been a lot more static than now, or at least so it would appear. The past appears to have been a lot more uniform and stable over longer periods of time than now. Technological process seems to have gone on at snails pace in contrast with the heady onrushing into the future that appears to exemplify contemporary technological progress.

You will notice a lot of equivocation in the preceding paragraph. Many, "it seems", many "it appears". The reason for this hesitancy being that this researcher is unclear as to the accuracy of these observations, whether they are particular to our contemporary times or whether people have always felt this way. It could be that this is just how time always appears from the human vantage point as it unwinds, unfurls and unloops from day, to week, to month, to year, to decade, to century. These remarks are introductory/preparatory, they are – though I share Lyotard's "incredulity toward metanarratives"[Lyotard, 1984]{pg. xxiv} – common knowledge. The distant past was a largely static affair, kingdoms came and went but always with people living in feudal conditions. There was a sliver of aristocracy and royalty that continued on and on via hereditary claims rather than meritocratic or elected means. Work derived directly from the labour of humans and animals. What metal-working there was resulted in weaponry and jewellery. Knowledge was stored up as lore, guilds guarded this lore, it was passed on from master to apprentice in the guild-house – horticulture and husbandry from parent to child in the field. Without much inaccuracy and in a rough comparison to nowadays it could be said that what people knew they knew heuristically[Foucault, 2002, cf.] rather than analytically. These were pre-scientific times, the scientific method was not even clearly elaborated. We think of those people, our distant ancestors, as sleepwalk-

ers – the more cynical amongst us claim that we sleepwalk[Koestler, 1959, cf.] through life to this day.

And what is the characteristic of our age? Is it the Space Age? The Digital Age? The Atomic Age? The Information Era? There is so much change in the area of technology alone that it would be hard to point to any one overarching defining feature. Which is why perhaps that the nebulous rubric Postmodernity has caught on the way it has. Or are we living in the Era of Globalisation? And with the voices from Tahrir Square still ringing in our ears could we say that we have entered, globally, the Democratic Era? And though the world's great monotheistic religions – those of Abrahamic origin – still hold sway over vast numbers of people, could we not cautiously at least whisper it that we seem to be on the brink of a Secular Age?

My point is that it will take hundreds of years, if we survive that long as a species, to settle on an overarching designator for these times, those that are present as this is written. Until then, let us agree for now that one of the hallmarks of the times we live in is that of pervasive computation and leave it at that.

There are seven billion people on this crowded planet right now. As of 2011 there were almost 6 billion mobile-cellular subscriptions in the possession[2] of people around the globe. As you know, each mobile phone is a miniature computer and two-way radio. For most people apart from our clothing, and then our wallet, the mobile phone is the next on the list of items we carry around with us. Computing is now pervasive, soon it will be ubiquitous. As William Gibson has said many times, "The future is already here – it's just not very evenly distributed."[3] It may not be the case this very day, but within two generations there will not be a place on the planet without access to mobile technology, it is inconceivable given current trends that it could be otherwise.

It is a simple observation that the digital is part hardware and part software so it can be said that the nature of the digital is thus a two-fold nature. Bifurcating again, software has a dual nature, source and binary. The hardware is the substrate, the software is the fluid logic that congeals on top. Hardware is silicon, wires, transistors, capacitors, resistors, microelectronic components. The width of the pathways in contemporary microchips are measured in nanometers - so I could just as easily have said, nanoelectronic components. Silicon fabricators like Intel (x86), Global Foundries (x86), Samsung (ARM), Qualcomm (ARM), IBM (POWER) have reached the stage where they are packing billions of transistors onto each high-end microprocessor. The fastest supercomputers have tens of thousands of processors and terabytes of RAM. Their performance is measured in the petaflop range. Computing power is increasing relentlessly and following a very strict power to price ratio and it has done so for many years, as witnessed by the Moore's famous law (Gordon Moore, ex-CEO of Intel). This has also been dubbed the Law of Accelerating Returns. This trend is apparently so uniform and so predictable, from one telecommunication generation to the next, that some technologists[Kurzweil, 2005] have predicted a technological eschaton - a technological rupture, an age of super-intelligent machines. This is the informatic fringe of eschatology with its own attendant techno-rapture whose approaching event horizon of hyper-intelligent machines is termed the singularity.

1.2 The topic of this thesis

We in the humanities have to grapple with the implications of this accelerating digital change. If we are to view all this from a most abstract of perspective we might use the tools of philosophy. If we are to stay within the sciences we would use the tools of sociology. If we were to take an interdisciplinary and more grounded approach we would work within the tradition of cultural studies. Researchers in the humanities - properly called the digital humanities, if the focus is exclusively on the area of information and communication technologies - can decide to take a look of the impact of the digital in any number of ways. I have decided to focus on software, that malleable part of the digital nature, and in particular mathematical software.

This thesis is an exploration of the state of play in mathematical software tools, both open source and proprietary. The goal is to examine these tools from the perspective of how they are built and how they are used, both scientifically and artistically. The goal is to explore how mathematics has been altered by this transition to the digital. Because of this I need to look at mathematics software as software – how does it fit into the usual fabric of software development. Much has been written about the practice of software development, whether, like Donald Knuth, one calls it the art[Knuth, 2009, cf.] (or craft) of software creation or whether, as is more standard in academia, one calls the discipline software engineering[Ghezzi et al., 2002, cf.] reflects on how you view the practice as it relates to other trades and professions. Also to be explored is how mathematical software on a digital platform opens up entirely new vistas in the realm of visualisation that simply did not exist before. I am not going to look at simulation and modelling but I shall look at how mathematical software through realtime visualisation has allowed humanity to begin to comprehend better the chaotic and fractal nature of the world. Finally, also to be examined is how this enabling technology is seeping into art in the form of, as I see it, a logical successor to abstract expressionism and geometric art via the diverse and vibrant genre of mathematical art.

On a personal level allow me to explain how I have arrived at particular topic. My bachelor's degree is in philosophy and mathematics. True, I could by no means call myself a mathematician, I do not have first-hand knowledge about the day-to-day activities of the working mathematician. But there is a part of me that is fascinated by mathematics – Mandelbrot sets, the infinitesimal calculus, mathematical logic, number theory, you name it, I find it interesting. Unfortunately my passion exceeds my ability. I will, alas, never get very far into the world of mathematics, so I see this thesis as a way of being a part of the mathematical world. The free (as in free speech, not as in free beer – liberty versus gratis) and open source software angle enables me to link mathematics to my digital culture studies because software is a prime manifestation of the digital, and because free and open source approaches to software development are fluctuations in culture. The visual mathematical artefacts angle enables me to link mathematics to my digital culture studies because art is always a manifestation of culture though oddly enough when joined with mathematics it seems to bring about creations that transcend any one culture such is the universal aspect of mathematics. The fractal and chaos angle enables me to link mathematics to my digital culture studies by showing how the way study and research in the digital humanities can be radically different[4] from the sort of digital printing press ideology that permeates the humanities at present.

How are all these areas expressions of culture or fluctuations in culture? The way I see it, culture is the sum total of non-universal behavioural traits in humans. I usually help myself along in this understanding with the following refrain, culture is how people do things differently together. By this logic culture is never something purely individual, individuals are vessels of culture but culture is expressed only collectively and never universally. A prime example that I use for explaining this to myself is the observation that everyone dances but that different groups of people have differing costumes, styles and traditions of dance. Culture is in the differences, it is a patina or layer over the universal. With respect to software development there are many cultural differences in software methodology, some of which bear no relation to software development as an engineering discipline. People who believe in free software or open source software believe in the pragmatic or ethical behaviour of sharing the source code to their labour. The contrary cultural stance is proprietary software where the software is not shared. There are many nuances but that is the general gist.

This thesis will contain a distinct bias. I believe that the free and open source software methodology is the better way to build reliable software systems, systems that are less hostile to the users of the software than closed or proprietary software. Taking this into account, not only shall I explore the terrain of open source mathematical tools versus proprietary tools (this survey I will, of course, perform in an unbiased fashion) but I shall also look at ways to help foster and improve the open source ecosystem of mathematical tools.

I have given a flavour of the topic, I now turn to the approach. If you will permit me to say: I have a difficulty with the word approach and with how it is used in academic contexts. For me the word suggests the direction from which a researcher goes towards their research space, rather than a synonym for methodology which is how I am coming to a slow realisation that it is employed. Consider a mountain climber. The approach to the summit is the angle of attack as such, it says nothing about the tools the climber uses – I would prefer if approach held on to its old clothing rather than being coerced into new garments. With that brief but pertinent diversion out of the way let me discuss my proposed methodology. As is appropriate to humanities and cultural studies I shall perform qualitative research. My aim is to interview mathematical software makers, adherents to both free and open source methodologies. I aim to make a survey of the impact of mathematics and computers on art. The approach will be digital ethnography (please see subsection 2.3) say – to a large extent philosophical as that is my bent.

In the humanities the computer is treated as a fancy desktop publishing tool. The process is the research itself. The results are static. If there is anything that could be learnt from the digital and network revolution is that research could become a collaborative open-ended affair and results could be presented in an – to use that rather passé word – interactive manner. What do I envision by this? Picture if you will a thesis where the charts and graphs can be manipulated by the reader. Already the thesis has become a hyperlinked affair, something so mundane that it escapes our attention. But what is more amazing that being able to follow a link in a chain of information instantly so that you retrieve the next piece of information stored on a machine thousands of kilometres away? It seems that we in the digital humanities write up page after page about all the new affordances the revolution in ICT brings and yet are content to let this natural progress wash gently over our shores leaving us undisturbed.

1.3 Coda, what is meant by digital?

It is an observation of mine that cultural studies overlaps with media studies to such a large extent it is difficult to remember that there is a life beyond media. Truth be told media analysis, for whatever reason, makes up a large part of (digital) cultural studies and as it is obvious to anyone that the media has been impacted greatly by the transition to digital technologies what we appear to be left with are numerous articles analysing online sub-cultures and the affect of the desktop technology, mobile technology, and internet technology on human behaviour.

In the Language of New Media Lev Manovich asserts[Manovich, 2002] that the language of the new media (and by implication code) that now surrounds us is the language of the machine, or more succinctly the digital. The nature of this language according to Manovich shall be dealt with now.

Manovich argues that there are five "principles" of new media: numerical representation, modularity, automation, variability, and cultural transcoding. In the first case our real world is sampled and quantised – the continuous becomes binary digits. In the second case the media objects themselves are comprised of parts that can be manipulated independently while keeping a unified systemic integrity – a similar observation is made for code as it is for data. The third case refers to the programability of this new environment. The fourth case refers to the malleability and fluidity of the bits and how the interface is decoupled from the content; it also refers to how constants can be replaced by variables in data. The final case seems to refer (I say seems because I could not commit myself to say with absolute certainty I got this principle) to how these new properties affect culture – transcoding meaning to go from one format to another.

In the What New Media Is Not chapter he says: "Having proposed a list of key differences between new and old media, [...] Following are some of the popularly held notions about the difference between new and old media that I will subject to scrutiny:"{pg. 49} New media is analog media digitised, all digital media share the same digital code, new media (and code also) allows for random access, involves a loss of information, copied endlessly without degradation, is interactive.

Here we encounter the digital. What is the digital? In fact, what is a medium? Here is my take on it. Words like medium, text, and so on have their roots in the physical – letters on a page, the air as a carrier of sound waves. When physicists use the term medium they refer to a specific physical object that has specific physical properties. When we say that print is a medium we are using the term in a slightly more abstract sense. When Manovich puts "the Internet, Web sites, computer multimedia, computer games, CD-ROMs and DVD, virtual reality[5]"{pg. 19} together in a list (though he says that this list is culled from topics of discussion in the popular press) what exactly is he doing? I assert that he is being vague.

I do not think we need a haphazard list. For instance, we have no problem saying that print is a medium, things like books and leaflets, posters, mail, newspapers, photographs are printed onto paper and card and photo-paper. So CD-ROMs and DVD are optical media. They have a name, this is their name. Computer games? These are media? Computer games are games. Games are not media. Though, in one sense imagine if you said that not only is print a medium but so are books – in this way card games could be media but that doesn't feel right. Let us say that computer games are

stored on and accessed from digital media but are not media themselves. How about computer multimedia? This is vaguer and more abstract still, we even have the word media in multimedia – in what sense is the word being used here? Multimedia is already an old fashioned word, I only see it used to describe multimedia courses where the student learns digital audio manipulation, digital image manipulation, 3D modelling, web site creation. Do the education providers see each craft as a medium? Possibly. Let us leave this puzzle to one side for the moment. Is the internet a medium? Are websites a medium. The web is only a part of the internet, just like email, podcasts, VoIP, and so on. If the websites and the internet are both media then media can be nested and there is no reason to think that email is not a medium. Is email a medium? Is it part of the new media? If email is a medium then is ordinary mail a medium?

But I feel that print is the medium and that mail is a use of the printed medium and not a medium in its own right, others could disagree. What is it about digital mail that makes it easier for me to feel like it could be a medium? Medium, in Latin meant "the one in the middle" which is intermediary in English. (If a medium is a channel or a carrier, then mail and email are sub-channels or sub-carriers – which shows that the designation information superhighway is not that wide of the mark after all.)

Taking a step back in order to survey the landscape better, how is new media to be characterised? If new media is anything at all it is digital media. So wherever the bit can reside is a digital medium, like in the conduits of the internet, the pixels of your monitor, in optical media, the wires in the circuitry of the magic box of tricks in front of you, in the registers of the CPU, in the cells of the memory chips, in the magnetic poles of the particles on the hard drive. This you could say is our digital infrastructure. Should we collapse all these places together and call it a digital medium? It seems more like a digital realm than a medium. Besides some of these places are used for storage, some for display, some for manipulation, and some for transmission. It is like having the sheafs of paper and the typesetting machine and the printing press arrayed on the desk in front of you, maybe that's why they call it desktop publishing.

So is the internet a medium? But what is the internet? Well, the internet is the pipes, the computers and the protocols. Bits move around according to protocols. Because the binary digit has been freed from the particular substrate of any one medium. The bit is addressable to use a piece of computer jargon. Thus the symbol is addressable. I realise that I keep referring to print here as if it were the canonical old medium but I feel my musings holds for all non-digital media, with print the symbol is impregnated in the medium – if you want to copy a book you must photocopy the entire page, symbols and all. This is the old media way. Because it appears that anything at all can be encoded numerically once you agree on a convention for encoding then it appears that nearly any type of object can be represented numerically. The chemical sequence of molecules for instance. Vocal patterns. The hue, saturation, colour and luminosity of an image. Entire books. Encyclopædias. It is my assertion that the digital media are multi-format or multi-modal because of the unifying underlying representation and addressability of the bit.

Another way that I look at all this is through the lens of metaphor. Take, "This text is a vehicle for my ideas". N. Katherine Hayles shows us of course that there is no ideal text as such, let us say that we are using the word text in the nominalist sense, not in the Platonic sense. Can we say, "This book is a vehicle for my ideas"? Sure we can. But do

you mean the book in your hand? Or all the books printed? So you mean the text. The book stores the text. The book is a container for the text in some way that the text is a container of your ideas through the beauty and magic of language. If the text is a vehicle, the vehicle must move over some surface. Let us see if the metaphor holds. Imagine the text is a train or a car then the surface must be a track or a road. Is the surface the book? Or is it print? You could also say, "This song is a vehicle for my ideas" or "this poem" or "this story" or "this film" so clearly the vehicle metaphor works across mediums. So even though the surfaces vary the metaphor works. You wouldn't say "this internet" or "this computer" but you could say "this multimedia extravaganza" or "this web-comic" or "this video-game" so we can see that Manovich's list is indeed haphazard as it appears to contain objects from different categories. What is interesting is that we call the internet (and its HTTP subset the web) the information superhighway so we see it as a surface for vehicles in some way.

So now that we are at the conclusion of this overly long synopsis have we arrived at any conclusions? We have come to the realisation that with new media and with computer code the symbols of the content are readily divorced from the medium itself. I do not know what to call this. Manovich is right to be suspicious of the 'random access' principle because it doesn't capture the full idea but it definitely captures some of it. Like I say the jargon in the computer science field is addressability but then again dictionaries are indexed and we have page numbers and tables of contents but then again these only allow you to go to certain crude parts of these works, not to each and every symbol. So we have automated bit-level granularity and addressability in the new media. When it comes to modularity I cannot think of a single book that is not broken up into chapters or a newspaper into sections so I think modularity is not a feature unique to new media. As the bit is an undifferentiated meaningless object we need formats and encoding systems to turn a stream of bits into a meaningful chain of information. Also we need to say where the boundaries of the objects are and we need to initiate and terminate digital conversations so we need synchronisation techniques.

What is most instructive to me are the technical details of cultural theorists. It appears to this reader that the more waffle and verbose the work, the more it is guaranteed that the text is short on or fudges the technical detail. Case in point, other researchers in the digital humanities, for instance Hayles (who I've already mentioned) and Espen J. Aarseth go into fairly accurate and comprehensive technical detail in their various works. Manovich stays on the level of cultural discourse which is always going to be vague. When he does venture into technical terrain: "A computer program written by a programmer undergoes a series of translations: high-level computer language is compiled into executable code, which is then converted by an assembler into binary" he is superficially correct. "A computer program written by a programmer undergoes a series of translations" – yes, they are called compilation passes but also a computer program can be interpreted so they could be called interpretation passes. As in to pass over something while processing it. "High-level computer language is compiled into executable code" – no, as I said it could be compiled or interpreted so why not just say translated? The computer program is called the source and the result is called the target. There may be a number of intermediate steps but the compiler or interpreter generally produces object code, not executable code. "which is then converted by an assembler into binary". An assembler translates assembly code into machine code (binary code) but

rarely nowadays do compilers or interpreters generate assembly language so an assembler is rarely invoked. What actually happens is that a linker turns object code into executable code by resolving symbols and this executable code is binary code. The details are available in any recent book on compiler theory.

A note on the tools I use: I am writing my thesis itself in LaTeX[6], an open source documentation preparation system traditionally used in the scientific and technical community. Rather than hand-crafting the document I am using a package called LyX which when it is behaving takes care of the nitty gritty mechanics and lets me get on with the actually task of converting my observations into something hopefully worthwhile.

2 Research overview

2.1 Thesis outline

There are three main parts to my thesis: sections (3)'The impact of mathematics having gone digital', then (4)'The culture and ideology of intangible media (data) and software development (code)' and then (5)'Specific comparison and questionnaires'. The remaining sections being: the (1st)'Prefatory remarks' section being a mere preamble; and this section here, the (2nd)'Research overview', being an overview; the thesis concludes with section (6)'Summary of results and conclusion'. All the references, be they citations or endnotes and whether they refer to electronic or print resources are contained in section (7)'Endnotes and citations' and finally support documents (online survey forms and the like) are listed in the Appendix.

Section (3) traces the origins and growth of mathematical software. I show how one branch of mathematics, that dealing with chaos and fractals, had to wait for the development of adequate hardware and software before it could be explored. I show how mathematical software is now changing the art landscape.

Section (4) traces the origins and growth of intangible creations – previously the domain of thought and text only. With the rise of the bit (both data and code) humanities notions of property have had to adapt (both media and software). In concrete terms this means looking at: the origins and evolution of the notion of copyright, the nature of how software is crafted whether proprietary or open source, the patenting of intangible objects, the nature of the new economy in the networked society.

Section (5) analyses the responses to two questionnaires whose questions are related to issues I grapple with in the previous sections. These questionnaires are directed at developers and users of a leading open source general mathematical software tool and at the developers and users of a leading proprietary mathematical software tool. Both tools are equally capable of working with the mathematics and generating the art found in section (3) and both projects are embedded in the system of regulations, culture and ideology discussed in section (4). I extend the analysis by considering issues from the perspective of the philosophy of science and point to recent theoretical work in this area.

2.2 Overview of source material

This subsection runs through the available sources of information and which of those sources I have used. One of the first tasks of any researcher is surveying the landscape of existing research to see what prior research is applicable to the research at hand. This includes popular and specialist non-fiction (drawn from coursework and beyond), academic papers, articles within journals, and encyclopædia entries.

As regards coursework only a couple, certainly no more than a handful of titles were strictly relevant to my topic. It is not that my thesis topic falls outside the allowed limits of our course but rather that the course itself only tangentially touches the area of the craft of the development of software (including mathematical software) and the impact of the digital on mathematical art. In the main coursework related to participatory culture, new media, methodological issues in internet research, and theories of digital

culture informed my research. Examples include Charlie Gere's Digital Culture, essays by Wendy Chun such as On Sourcery and Daemons, or Code as Fetish, Hayles's How We Became Posthuman, Espen Aarseth's Cybertext: Perspectives on Ergodic Literature, and as seen in the preamble to Lev Manovich's (though it contains flaws) The Language of New Media. Whether it is because I have a perennial interest in such matters I found that issues in these works relating to openness and transparency, access and control arise again and again - we could be talking about the area of social media, electronic poetry, or digital aesthetics. For instance, when talking about blogging and citizen journalism our class concentrated on the whether being a blogger qualifies one as being a journalist, we spoke about the instant global reach of a blog, we talked about the interactive nature of commenting systems, the side-stepping of the mass media and with that the ability of directly speaking truth to power albeit with a smaller voice than traditional well-know organs.

2.2.1 Mathematical art, fractals, chaos, computer algebra systems

I have drawn a lot of material from the New Encyclopædia Britannica, 15thedition, which I have found to be invaluable [Enc, 1982a]. I have also drawn on two recent art exposés: Philip Steadman's Vermeer's Camera: Uncovering the Truth Behind the Masterpieces and David Hockney's Secret Knowledge: Rediscovering the Lost Techniques of the Old Masters [Steadman, 2001, Hockney, 2006]. I have relied on The Fractal Geometry of Nature by Benoit B. Mandelbrot [Mandelbrot, 1983]. In a similar vein I have extracted valuable and learned scholarship from Chaos and Fractals by Heinz-Otto Peitgen and Dietmar Saupe and Hartmut Jürgens [Peitgen et al., 1992]. I have been inspired by the Bridges series of conferences. Bridges concerns Mathematical Connections in Art, Music, and Science. The 2011 was held at the University of Coimbra. From the proceedings I have drawn two works: The Art of Complex Flow Diagrams by Anne Burns [Burns, 2011] and A Nine- and Twelve-Pointed Star Polygon Design of the Tashkent Scrolls by Lynn Bodner [Bodner, 2011]. From the associated exhibition I have drawn on Aesthetic Explorations by Nathan Selikoff and Biscuit Land by Mehrdad Garousi. I have been in email correspondence with all four artists.

2.2.2 Free and open source software, hacker culture, software development

I made use of a video from The Berkman Center for Internet & Society at Harvard University about the book Common As Air by Lewis Hyde which is a look at the history of the commons [Hyde, 2011]. The Cathedral & The Bazaar by Eric S. Raymond has been a key text for me as it is a key voice of the open source revolution [Raymond, 2001] – it is one of the first sociological takes on the free-software / open source movement by an insider. Dreaming In Code by Scott Rosenberg a book concerning a much talked about open source project helped shape a lot of my thinking, though I do not quote it directly [Rosenberg, 2008] – it gives a detailed business history of and insight into the high-profile Chandler project. Just For Fun by Linus Torvalds and David Diamond [Torvalds and Diamond, 2002] a biography of Linus Torvalds and his baby, Linux – again I haven't quoted it but it was instructional. Free As In Freedom by Sam Williams [Williams, 2009] – a biography of Richard Stallman and his babies, the GPL,

Emacs, GNU and the FSF. Online. The Mythical Man Month by Frederick P. Brooks [Brooks, 1995] – a classic of software development literature, posited that software development doesn't scale by throwing more people at the problem in the same way as other engineering tasks.

2.2.3 Various viewpoints: legal, economic, ethical, epistemological

Riffing on Adam Smith's Wealth of Nations is Yochai Benkler's Wealth of Networks who gives an economist's take on network effects as they relate to intangible products [Benkler, 2006]. Tim Wu delineates the history of monopoly in the telecommunications industry with his text The Master Switch [Wu, 2011]. The Stanford lawyer and policy influencer Lawrence Lessig has written many books and is a driving force behind the Creative Commons one of whose licenses this thesis has been brought out under. He has Future of Ideas [Lessig, 2001] which is available online, also Free Culture [Lessig, 2005] and Code 2.0 which is what I've directly quoted from [Lessig, 2006].

2.2.4 Other notable sources of material

Sage documentation [Stein et al., 2011] whether online or local tutorials and manuals has proved invaluable. Mathematica documentation and Wolfram Mathworld has proved very useful also [7].

2.3 Methodological decisions

As I am looking into the past in a number of ways (the history of copyright, the historical changes in art, the co-development of mathematics and mathematical software) I shall employ Historical Research. Because of my nature and background I often strive to generalise from particulars once having delineated the history of a topic and so I employ a lot of Philosophical Research. The same holds true when analysing the structure of the world around us, as in when I look at the nature of the new economy or the nature of the digital I again use philosophical techniques informed by various sources.

Quite recently it has come to my attention that a researcher, Diana Harrelson, in the United States is writing a thesis on the community surrounding the Fedora distribution of Linux. I have corresponded privately with her. She has classified her work as digital ethnography[8]. This seems like a suitable overall moniker for what I am trying to do with my thesis, indeed it seems like a suitable moniker for many of the investigation in digital culture and the digital humanities. It is ethnography (a thick description of human social interactions, using Geertz's terminology) with a digital twist.

In thinking about methodological decision-making I have drawn on heavily from three sources, an online research method resource of Jyväskylä university[9], the Paradiam Dialog edited by Egon Guba [Guba, 1990] and the Sage Handbook of Qualitative Research edited by Norman K. Denzin and Yvonna S. Lincoln [Denzin and Lincoln, 2005]. Now that I have posed my questionnaires and garnered the responses I have taken both an analytic and synthetic approach (analytic – is there anything more to be learned from the individual responses than meets the eye on the

surface level?; synthetic – what kind of general conclusions can be drawn when all the responses are looked at together?).

As to the methodological issues (by which I mean ethical ones) I am likely to face now that I am aware of the terrain. Ethically if I use information "publicly" available on the internet I will do my utmost to acknowledge the original source, contact the source where possible, and to respect the intentions of whoever posted the information online – one does when taking information from a printed work. Non-ethical issues will involve the accuracy of the information I receive from the persons I interview, the comprehensiveness of the feature-set comparison I compile, and the fidelity of the historical account I provide.

3 The impact of mathematics having gone digital

3.1 A bird's-eye view of mathematical software

Mathematical software is software used to analyse, model, visualise or calculate numeric, statistical, symbolic or geometric data. This would be an uncontroversial definition of mathematical software.

This is a personal observation made from an assessment of indices of mathematical software so the list bears repeating – numeric, statistical, symbolic or geometric.

Let me quickly point out that there is absolutely no consensus on any of the terms involved. Terms as seemingly obvious and transparent as number or numeric at first glance are upon deeper inspection fraught with remarkable tensions. Also symbol or symbolic are amorphous terms surely applicable well beyond the domain of mathematics. The terms geometry or geometric, statistics or statistical are perhaps the least contested of the terms under scrutiny. (Why this is so can be shown to be the case or expanded upon in an appendix perhaps, this is the realm of the philosophy of mathematics and far beyond the scope of this thesis.)

Beyond this, what are we to say of mathematical software that has as its primary function something other than the manipulation of mathematical entities or objects? What about word processing, document processing or typesetting software whose primary function is the creation of non-mathematical documents but otherwise provides exceptional handling of the inputting of mathematical symbols? Word by Microsoft Corporation[10], InDesign by Adobe Corporation[11], and LaTeX[6] are all examples of these. Then consider pieces of software that allow us to explore fractals (using Benojt[12] for example) or chaotic maps such as the Lorenz attractor (using 3DAttractors[13] for example). In these cases the output is pictorial or even artistic in nature. These are partly my concern, but most applicable to me are those pieces of software that have as their primary function the manipulation of mathematical entities

To backtrack or rewind a moment. From another perspective (following Wiedijk among others, see below) mathematical software is generally subdivided into two parts: computer algebra systems (CAS, sometimes computational algebra systems) and automatic theorem provers (ATP). You may think of the first sort as glorified calculators, plotters or modellers. You may think of the second lot as more sophisticated derivers or solvers. The bulk of the wealth generated for purveyors of mathematical software is derived from packages belonging to the first group, that is, CAS. The second group exist more in the realm of academic research rather than as part of the product portfolio of a software vendor. Of CAS, some of these pieces of software cost thousands of dollars and are the fruit of innumerable person-hours of work. ATP is a much more recent advancement in the field of mathematical tools. This is as opposed to CAS which can trace their roots back to mechanical mathematical aids such as the humble electronic hand calculator, the slide rule, the Babbage difference engine, the abacus, even the pen and paper after a fashion. ATP was just not possible before the advent of the digital computer and indeed such are its goals that the success of ATP is intimately intertwined with the success of the project to create artificial intelligence. I will speak about this area of mathematical software now.

The first real headline success in this area (computer-aided proofs) was in the prov-

ing of the four colour theorem[14]. A short paper by Andreea S. Calude from the department of mathematics the university of Auckland, New Zealand details the controversy that this feat aroused[Calude, 2000]

Halmos explains that a proof done with the use of the computer has the same amount of credibility as one done by a "reputable fortune Teller", as these machines are similar to "oracles" since they have certain physical properties that we have yet to understand. Similarly, Deligne, from the Institute for Advanced Study (a 1978 Fields Medalist) shares the same point of view: "I don't believe in a proof done on a computer. In a way, I am very egocentric. I believe in a proof if I understand it, if it's clear. A computer will also make mistakes, but they are much more difficult to find".{pg. 5}

What is undeniable is that an audit trail was necessary to prove that the algorithm that the mathematicians had used had performed as stated. Also, the algorithm itself was an object of intense study and integral part of the proof. A sort of man-machine symbolic symbiosis. I cannot understate the profound consequences of this for mathematics. When taken to its limits we could see that we could ask a machine, "Is the Riemann hypothesis correct?", and it may reply "yes" or "no" and how would we verify in totality its reasoning for if it were simple or trivial we (humans) could have performed the proof ourselves in the first place and not left it to the machine – if on the other hand the mechanical reasoning is non-trivial then the verification will be non-trivial also. In essence, what would have to be done is to prove the theorem prover and as can be readily surmised this is an acta ad nauseam. What we have here is the Halting Problem, but under another guise. It should not be a surprise that we find it lurking here.

Perhaps a word about the four colour theorem is in order here to give the reader a sense of the terrain. It was first asked by Francis Guthrie in 1852 what is the minimum number of colours that are needed to colour a map (really a two dimensional surface) divided into arbitrary regions such that no two adjacent regions share a colour. What is nice about this question is that it is simple to state, easy to visualise, and fiendishly difficult to arrive at a definitive solution. Over the years a maximum upper bound (of 5) was set by Heawood in 1890 but never an absolute minimum. Eventually a method was devised in 1976 by Kenneth Appel and Wolfgang Haken that involved generating all the possible permutations (a set of 1,936 (sub)maps in total) of two dimensional regions, methodically painting and repainting them and all the while checking for the least number of colours. It was realised that the process would be too tedious to be done by hand, the algorithm was translated into a computer program and the result of 4 was arrived at.

One can see how the solving of the four colour theorem involved crafting custom code dedicated to that one problem. ATP generalises this by allowing the mathematician to specify axioms in a mathematical reality, the software may then be asked if a certain state of affairs can be reached from the knowledge it has been fed and in the process it may either prove or disprove theorems. How they perform this feat is beyond the scope of this thesis.

Getting back to CAS. Though I called CAS software "glorified calculators, plotters or modellers" this is a multi-billion dollar industry. As you can imagine these software

packages have a far greater range of applicability and utility than packages which perform single purpose tasks such as statistical functions or geometric functions or what have you. Besides, the more general purpose (and at the same time commercially successful) packages contain statistical or geometric or other features – this is what makes them general in the first place. There are essentially four high profile general CAS packages: MATLAB[15] by MathWorks, Inc., Magma[16] by the Computational Algebra Group at the University of Sydney, Maple[17] by Maplesoft and Mathematica[18] by Wolfram Research. There is no comparatively successful FOSS equivalent to these packages. The nearest to my mind is Sage [Stein et al., 2011] which is why I have chosen to direct a series of research questions at the developers of Sage.

Keep in mind that there are literally hundreds of pieces of mathematical software[19] as this comprehensive online taxonomy by Freek Wiedijk shows. Indeed Wiedijk's criteria for categorisation requires that the item under scrutiny have something to do with mathematics, have something to do with computers; furthermore, the items must be active, public and significant. My criteria coincide with the last three but as was mentioned above my concern is with CAS rather than textual authoring systems or other systems that do not have the manipulation of mathematical entities to produce mathematical results as their primary function. Wiedijk imposes a structure of nine categories on "mathematics in the computer": two are non-software so we may safely discard those, one is in connection with authoring and the rest divide into CAS and ATP. A useful distinction that Wiedijk makes is one of differentiating between proof checkers and theorem provers with the former acting as an aid to the reasoning of a human and the latter working out a strategy to solve a specified theorem in its own right. Wiedijk does not distinguish between the various types of CAS which in a way would be like delineating the pieces of software along lines of mathematical function such as geometry or statistics and so on.

3.2 Historical development and state of play

The history of mathematical software is interwoven with the development of computers and computer software. As has been stated and documented in numerous places (see for example the first chapter of Gere's Digital Culture[Gere, 2002]) computers are an outgrowth of the capitalist system, specifically the late capitalist phase, and in particular the military industrial complex. This narrative is ubiquitous and it is difficult to disavow. The military has always had recourse to avail of the talents of mathematicians, notably in ballistics, owing to the fact that precision is a virtue in war; deception and secrecy use mathematics as their cloak in the murky trade-craft of espionage.

That is not to say that mathematicians have always been the willing tools of the generals but patriotism and monetary gain are charming seductresses. It is well known among mathematicians that Godfrey Harold Hardy, knows simply as G. H. Hardy, a committed pacifist, said in A Mathematician's Apology[Hardy, 1940], "No one has yet discovered any warlike purpose to be served by the theory of numbers or relativity, and it seems unlikely that anyone will do so for many years."{pg. 44} What he meant by this is that he was happy to pursue areas of mathematics, pure mathematics, that had little application to the real world. It is in this text as well that we see the trope of beauty and creativity which is often seen in describing the art of software engineering.

The original computers were special purpose mechanical devices for speeding up computation. They were never called computers. They were one off devices like the Pascal Pascaline, the Leibniz Wheel, the Jacquard loom and the Babbage Difference Engine and Analytical Machine. The first actual computers were the female operators that attended to the calculating machines of the early 20th century. Whether it was calculating ballistic trajectories or tallying up a census these machines were always special purpose devices and any mathematical software such as existed was bound up in their operation. Software implies a distinct entity from hardware or from the ether-ware of abstract algorithms. It wasn't until the notion of stored programs with the Von Neumann architecture of computing devices could we in any way begin to talk about software as a thing in itself. There was no recognisable software industry until computing hardware became fully commoditised. As such, there was no such thing as mathematical software per se, at least not independent software packages whether commercially available or otherwise.

The historical development of mathematical software follows the course of software in general. Mainframe to mini-frame to microcomputer. Special purpose to general purpose to mass market to a certain extent. The first CAS systems can be traced back to the 1960s. The history of mathematics is a well-trodden discipline with a sizeable literature.[1] But the conjoining of the two, a history and overview of mathematical software, being a relatively recent phenomenon, is a path less trodden in terms of the available literature. There are books related to this aspect or that aspect of mathematical software but none that I have found that would appear to tackle the intersection of these topics. Three volumes that I have discovered after a painstaking and exhaustive search are the proceedings from the 2002, 2006 and 2010 international congresses on mathematical software. Each volume is split across mathematical topics. As you can see from the dates I have just given the comprehensive field of study is very recent (at the time of writing).

In any event it seems we can trace this type of mathematical software back to the 1960s. REDUCE seems to lay claim[20] to being one of the first early attempts to create a general computer algebra system. Or rather, an attempt to create a system that allowed the automated processing of Feynman diagrams metamorphosed into something that is taken nowadays to be a CAS.

And now let us look at how computers have allowed mathematicians and artists to explore previously inaccessible mathematical vistas. I refer to the visual display of mathematical or quantitative data. As Edward R. Tufte has pointed out[Tufte, 2001] in the Introduction and chapter on Aesthetics in his seminal book Visual Display of Quantitative Information the field is quite young, perhaps 200 to 250 years old. William Playfair, a Scottish political economist and engineer, developed or improved upon graphical methods of statistics. Up until the latter half of the 20th century the rendering of these types of graphics was done manually but with the advent of the computer a new field called digital image processing was born. When applied to mathematics itself this allows mathematicians to visually apprehend the information that lies behind their algorithms and equations. When applied to art it allows artists access to mathematical

[1] Similarly a sizeable portion of the Amazon's forests have been felled to soak up the ink spilled in the name of the art of software engineering.

landscapes which at times share similarities but at other times diverge wildly from the imaginings of the pre-digital past.

3.3 An excursion into a fractal world

These are the words of the American meteorologist Edward N. Lorenz (1917-2008), who while at MIT came across by chance what is now called the butterfly effect,

> Well, this all started back around 1956 when some [...] methods of [weather] forecasting had been proposed as being the best methods available, and I didn't think they were. I decided to cook up a small system of equations which would simulate the atmosphere, solve them by computers which were then becoming available. [...] I finally found a system of twelve equations that would do this and found that the proposed method didn't work too well when applied to it, [...] I had a small computer in my office then, so I typed in some of the intermediate conditions which the computer had printed out as new initial conditions to start another computation and went out for a while. when I came back I found that the solution was not the same as the one I had before; the computer was behaving differently. I suspected computer trouble at first, but I soon found that the reason was that the numbers that I had typed in were not the same as the original ones, these [former ones] had been rounded off numbers and the small difference between something retained to six decimal places and rounded off to three had amplified in the course of two months of simulated weather until the difference was as big as the signal itself, and to me this implied that if the real atmosphere behaved as in this method, then we simply couldn't make forecasts two months ahead, these small errors in observation would amplify until they became large.[21]

The raison d'être for the existence of the discipline of digital culture, or for that matter the digital arts or humanities is that academia has come to recognise that humanity is at a kind of inflection point in its development. At any point in history this researcher would have been bringing the readers attention to this fact through the medium of the printed word. The humanities have embraced word processing like all other institutional and individual actors. This embrace is an embrace with no end, though the printed word has been the record of man's follies and triumphs for centuries now the truth that the book began its decline with the birth of desktop publishing is inconvertible. We admire the immediacy, the fluidity, the ease of editing in the digital realm - indeed the bound and printed volume is only ever the final link in the chain, bound up with the economic transaction of consumer and producer. And now with electronic-readers, e-readers, on the inexorable increase whose killer apps are convenience and portability even that link will prized apart.

As practitioners of electronic literature and electronic poetry are aware the text is (or rather I should say that in the right hands it has the potential to be) now a textual/computational hybrid. The works of this experimental avant-garde sidesteps the regular distribution channels as has been the case with all experimental prose and poetry down through the years. Limited set of practitioners, limited audience. There is, however, another group of people who create computational/textual hybrids - the users of a

certain class of mathematical software tools. Some of these are scientists and engineers, some are artists. All harness the power of digital computers to bring the mathematics in their documents to life: equations can be drawn in cartesian or hyperbolic geometries, statistics can be plotted, the arcane and complex typography of mathematical structures can be done by machine rather than tediously by hand. The metaphor is the notebook. A notebook is comprised of text and computation. Of course there is a marked difference from electronic poetry and electronic literature in that the text is at the service of the mathematics whereas in e-poetry and e-literature the computational nature of the work is brought to bear on the text in order to enliven the text.

Most word processing or desktop publishing or typesetting tools have built in scripting languages which are designed to automate tedious word processing and typesetting tasks. These are normally bundled into small modular units called macros. They are the stuff of nightmares. There is a tradition among programmers to make the functionality of the complex pieces of software they create scriptable. This allows the tool-user to build the functionality that the programmer did not see fit to provide because it was deemed that only a small portion of the user-base would benefit from said functionality. I can think of very few complex pieces of software that do not have a plugin system, a macro system, a built-in scripting language or whatever.

The complex system, TEX, that I am using to write this thesis has a scriptable core. It is a typesetting system that can create beautiful documents and understands mathematical notation very well. This means that documents with formulae can be rendered with ease. Pushed further it can be used to turn mathematics and algorithms into pictures.

Let me show you what I mean by a brief excursion into the world of fractals. In the process of showing you I aim in some way provide concrete examples of what is possible when the codex[Aarseth, 1997]{pg. 8} (as Aarseth calls the traditional text) meets mathematical software. It is an example of how academic culture of research and exposition must change in the face of the digital.

It is best to give a brief description of what fractals are. They are systems that exhibit self-similarity, scale-invariance, non-integer dimensionality. Benoit Mandelbrot, born in Poland of Jewish descent, coined the word fractal in his influential work The Fractal Geometry of Nature. It is often thought that fractal is short for fractional dimension, the non-integer dimensionality I mentioned before but this is not the case.

> I coined fractal from the Latin adjective fractus. The corresponding Latin verb frangere means "to break:" to create irregular fragments. It is therefore sensible – and how appropriate for our needs! – that, in addition to "fragmented" (as in fraction or refraction), fractus should also mean "irregular," both meanings being preserved in fragment. [...] (Since algebra derives from the Arabic jabara = to bind together, fractal and algebra are etymological opposites!){pg. 4}

What does it mean for a system or figure to non-integer dimensionality? The mathematical notion of dimension is not as simple as our everyday notion which means extended in space. It means that unlike the everyday notions of lines of 1 dimension, planes of 2, volumes of 3, fractals may have in-between dimensionality. The reason that this went

unnoticed for so long is because all fractals have a potentially infinite nature – the only the reason why pictures of fractals are possible is because a cut-off is used to terminate the iterative process of composing the figure.

When speaking of fractal geometry we are speaking of a geometry, a metric or measure of a geography that bears no relation to the Euclidean kind. Euclidean geometry is the intuitive geometry of the point, the straight line, the flat plane. This geometry is so intuitive that it presumed to be the only geometry and of course it matched the data from our senses perfectly: 3 spatial dimensions, 1 temporal dimension. In this limited universe fractal geometries, hyperbolic geometries and eleven dimensional superstring geometries are unthinkable.

According to Mandelbrot, "A fractal is by definition a set for which the Hausdorff dimension strictly exceeds the topological dimension."

The key idea here is that for normal objects are intuitive grasping of their dimension would be the dimension that the object's bounding figure would have. In such cases the Hausdorff dimension and the topological dimension coincide and are the familiar small integer units. The Hausdorff dimension for irregular objects is typically non-integer. For instance, the Sierpi□ski triangle has a fractal dimension of log_3 / log_2which is approximates to 1.5849625 in decimal notation.

Here is a quotation from "A Fractal Life" by Valerie Jamieson in New Scientist (November 2004).

The Mandelbrot set is the modern development of a theory developed independently in 1918 by Gaston Julia [1893-1978] and Pierre Fatou [1878-1929]. Julia wrote an enormous book – several hundred pages long – and was very hostile to his rival Fatou. That killed the subject for 60 years because nobody had a clue how to go beyond them. My uncle didn't know either, but he said it was the most beautiful problem imaginable and that it was a shame to neglect it. He insisted that it was important to learn Julia's work and he pushed me hard to understand how equations behave when you iterate them rather than solve them. At first, I couldn't find anything to say. But later, I decided a computer could take over where Julia had stopped 60 years previously.

So much more of our universe becomes comprehensible when we realise that phenomena that we took to be unrelated turn out to be capable of being modelled in similar ways. Paradoxically a theory about chaos brings a type of order from chaos. Perhaps one shouldn't draw too much from an off-hand remark by Benoît Mandelbrot but I think it is notable that digital technology was instrumental in enabling this transition from the geometries of intuition to geometries which defy intuition.

One type of fractal is the L-System. This is short for Lindenmayer systems after the Hungarian botanist biologist Aristid Lindenmayer(1925-1989) introduced and developed the concept in 1968 while at the University of Utrecht. He wondered if there was a simple mathematical way of describing the growth of plants and algae. He found that some of the growth could be described using a simple rewriting system, what we would know term a formal grammar. It was later found that many different mathematical entities could be described using the same rewriting system. I am able to show demonstrate them here because the language TEX supports their description.

Rules are labelled with capital letters. A rule is such that a letter is transformed into some other string where a string is a sequence of characters in a very restricted alphabet made up of the desired letters and pluses and minuses. The rules must be mutually recursive though in order that one may iterate as many times as one wishes. In the end one interprets the letters as lines (draw forward) and the pluses and minuses as indicating the angles (turn a certain direction) between the lines drawn. One starts off the whole process with an axiom, an initial starting value, which is one of the letters in the alphabet.

3.3.1 Koch curves (curves, islands and snowflakes)

The first fractal, the first L-System that I shall present, is known as the Koch snowflake or island. Helge von Koch (1870–1924) was a Swedish mathematician who first described the curve that now bears his name in articles in 1904 and 1906.[22] It is built by starting with an equilateral triangle. One removes the inner third of each side, then one builds another equilateral triangle at the place where the side was taken away, and so on and so on. By necessity this means that number of triangles scale up as the size of the triangles scales down. Fitting three suitably rotated duplicates together gives us the Koch snowflake. The Koch snowflake can be simply encoded as a Lindenmayer system with initial string "$F + +F + +F$" (which gives us an equilateral triangle – as opposed to just F which would only give us a line). The curve has the string rewriting rule "F" $\rightarrow$ "$F - F + +F - F$", and angle 60°. The zeroth through third iterations of the construction are shown below.

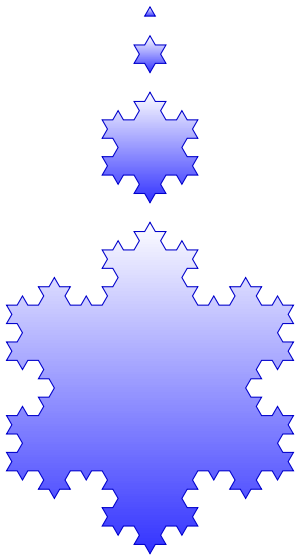

Figure 1: Koch snowflake or island: iterations one to three

The reader may find descriptions of the Koch curve from page 89 to 93 of Chaos

and Fractals (1992). It leads into other space-filling curves by the Giuseppe Peano (1858-1932) and David Hilbert (1862–1943). The author hopes that this has been illustrative.

3.3.2 Sierpi□ski triangle

The Sierpinski triangle (and if you want with the orthography Sierpi□ski), is a fractal named after the Polish mathematician Wacław Sierpi□ski (1882-1969) who described it in 1915. It is an attractive fixed set and it is also called the Sierpinski gasket or the Sierpinski Sieve. However, similar patterns appear already in the 13th-century Cosmati mosaics in the cathedral of Anagni, Italy and other places, such as in the nave of the roman Basilica of Santa Maria in Cosmedin. Escher produced studies of Sierpinski gasket type patterns from the twelfth century pulpit of the Ravello cathedral, designed by Nicola di Bartolomeo of Foggia. Originally constructed as a curve, this is one of the basic examples of self-similar sets, another example of a mathematically generated pattern that can be reproducible at any magnification or reduction.

Unlike the Koch snowflake, where one rule is used to produce the desired result, the Sierpinski gasket is produced using a pair of recursively referring rules "F" $\rightarrow$ "$G - F - G$" and its mirror "G" $\rightarrow$ "$F + G + F$". The axiom that starts it all off may either be F or G and the angle is again 60^o which makes sense because we are talking about triangles.

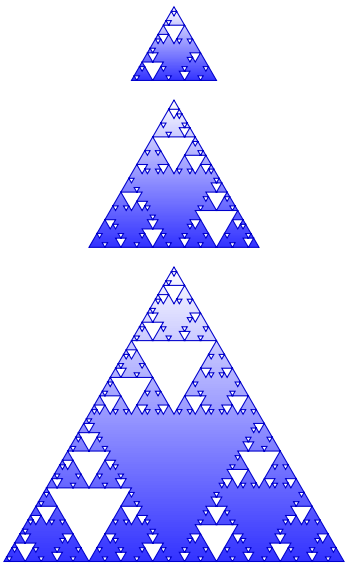

Figure 2: Sierpi□ski triangle: iterations five to seven

The Sierpinski carpet is a fractal with very special topological properties which we shall not dwell on here.

3.3.3 Fractal plant

Here, F means as usual draw forward, $-$ means turn left 25°, and $+$ means turn right 25°. X here does not correspond to any particular drawing action but is instead used to control the progressive changes of the curve. The operation has a stack (where you can save and restore state by pushing and popping items onto and off the stack. [is equivalent to remembering the current values for the angle and position, which are then put back when the corresponding] is encountered. Again two rules are used to produce the fractal plant. They are "X" $\rightarrow$ "$F-[[X]+X]+F[+FX]-X$" and the much much simpler "F" $\rightarrow$ "FF".

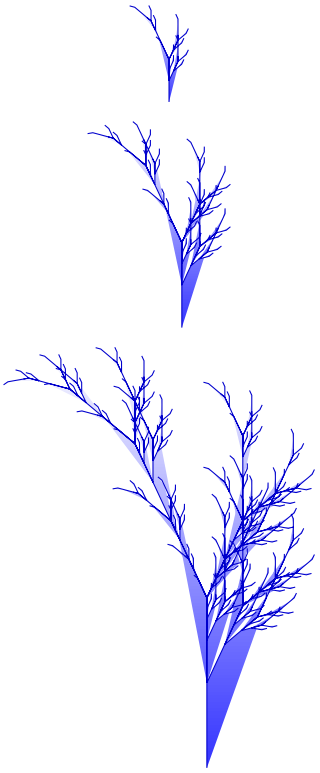

Figure 3: Fractal plant: iterations three to five

I cannot find who originated this fractal, it could well have been Lindenmayer himself as it deals with a botanical structure.

3.3.4 Hilbert curve

A Hilbert curve (otherwise known as a Hilbert space-filling curve because its spatial limit is the plane on which it resides) is a continuous fractal space-filling curve. It was first described by the German mathematician David Hilbert in 1891.[23] It was intended

as a variant of the curves, also space-filling, discovered by arithmetic pioneer Giuseppe Peano in 1890.[24]

Owing to the fact that it is space-filling, its Hausdorff dimension, which I mentioned earlier but did not strictly define, is 2 (precisely, its image is the unit square, whose dimension is an integer whose value is 2 as we know in any definition of dimension; its graph is a compact set homeomorphic to the closed unit interval, with Hausdorff dimension 2).

H_n is the n^{th} approximation to the limiting curve. The Euclidean length of H_n is $2^n - \frac{1}{2^n}$, i.e., it grows exponentially with n, while at the same time always being bounded by a square with a finite area.

Again, a pair of rules is used to describe the Hilbert curve. Here they are: "L" $\rightarrow$ "$+RF-LFL-FR+$" and its mirror "R" $\rightarrow$ "$-LF+RFR+FL-$". The axiom is L and the angle is 90°.

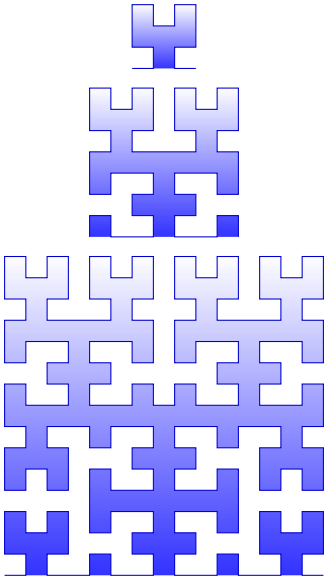

Figure 4: Hilbert curve: iterations two to four

If you imagine iterating this sequence to infinity you can see how the "curve" will eventually fill all the space in the plane.

3.4 The flight to abstraction: abstract expression and mathematical art

The early 20th century saw a flight to abstraction in art. When I speak of art here I am primarily concerning myself more or less with the branches of painting and sculpture in modern to contemporary art, the visual (or plastic) arts in general. I do not refer here to theatre, nor to dance, nor music, nor literature, nor decorative design such as mosaic or tiling or stained glass work – at least not in the contemporary context – in the past such cases may be touched on as the reader will presently discover.

Flight implies a fleeing from. What depiction in art fled from was the semi-accurate representation of the natural or fantastical world, a mode of representation that had long reigned to the exclusion of other modes. Be it a Turner landscape, a Rembrandt portrait, a Greco-Roman[2] statue, or a fantastical scene from Hieronymous Bosch or Gustave Doré the key ideas of perspective and figurative representation underpinned Western art. Outside of European/American art the technique of simulated perspective at infinity was used less to gird or underpin the pictorial scene but again the desire for a faithful representation of a natural or fantastical scene was evident. The only slight deviation from this observation that comes immediately to my mind are mandalas, a geometric sacred art of the Hindu and Buddhist traditions. (I have been told that there are aspects of this slight deviation in the art of the Merkabah tradition, Mayan folk art and symbols, and the architectural plans of Gothic cathedrals. I do not believe that these examples detract much from my observation.) However, these are abstract only in the pictorial sense as they are rich in formulaic symbolism once the mind has been initiated into the tradition - though clearly the intention is to visually overload the cognitive processes.

[2] I really do not mean to privilege the West. The desire to create a disinterested art appears to be a universal human trait, expressed in song, dance, monument, storytelling, craft. I use my examples in an iconic fashion, to call to the mind of the reader a particular example that signifies a global rather than parochial paradigm. I could have said Assyrian, or Egyptian, Scythian, Mycenaean, Minoan, and so forth – any ancient peoples.

Figure 5: Kalachakra, sand Mandala[25] – Tibetan Buddhist sacred geometry

There are many theories to account for this flight to abstraction. When studying the philosophy of art in Galway as an undergraduate my professor Felix Ó Murchadha described the aesthetic shift using the paintings of Kandinsky. He used the jargon of phenomenology and the visual language of the point and the line and the plane. What follows is a synopsis of Kandinsky from the 15th edition of the Encyclopædia Britannica which should help to set the scene[Enc, 1982c],

Among the vanguard artists who changed the history of art in the early years of the 20th century, the Russian painter and aesthetic theorist Wassily Kandinsky occupies a special niche of his own, for he is often referred to as the creator of the first pure abstraction – the first picture that broke with the Western tradition of depicting recognisable objects. His absolute priority in this respect, like most historical firsts, is open to argument. But, in his paintings and his writing just before and just after World War I, he was certainly one of the most undeviating and most eloquent of European abstractionists.{pg. 376}

The jargon of phenomenology as laid out in an architectonic fashion by Edmund Husserl (1859–1938) lends itself well to a description of the process(es) that Kandinsky and his contemporaries were going through. But, I would assert, only incidentally. The conception of phenomenology which Husserl and others – Edith Stein(1891–1942) and Maurice Merleau-Ponty(1908-1961) for instance – formulated is a study of the structure of experience and necessarily amounts to a formulation of how the objects of reality become objects of consciousness. There is talk of επoχηἔ, bracketing, phenomena, intentionality, noumena, and so on. The process entails a certain detachment, an ability to suspend belief, or bracket – the process is necessarily one of abstraction and is only one way among others for describing this process. It is one door of many into the world of metaphysics from the concrete world. This is why I say that as a description of the early 20th century move in art it is merely incidental rather than instrumental.

To imagine a war on the representational mode one must look for casus belli. I would look to technological roots. It strikes me that all such theories (theories of the flight to abstraction in art) must take into consideration the birth of the daguerreotype. This liberated the hand of the artist in the same way as the printing press liberated the hand of the scribe. Whereas the scribe copies a source text, the source for the camera is a snapshot of reality. Man Ray has said in an interview[26],

> I paint what cannot be photographed, that which comes from the imagination or from dreams, or from an unconscious drive. I photograph the things that I do not wish to paint, the things which already have an existence.

We can see that the flight to abstraction came some time after the mid-to-late 19th century. Kandinsky, who is credited with some of the earliest 'pure' abstract pieces, was operating in the early 20th century. In the intervening time the potential of the camera for fixing reality onto a thin sheet of paper was being absorbed and grappled with. The same can be seen in sculpture which started assuming more open forms and began to be concerned with form itself.

Figure 6: Kandinsky's On White II (1923). Centre Georges Pompidou, Paris.

It has recently been shown[Hockney, 2006, Steadman, 2001] that many of the old masters used cameras (both obscura and lucida), mirrors and lenses to project light onto canvas as a technique or shortcut in depicting reality. Now with the camera being perfected there was less and less a need for realism in art, less a need for painting over a projected image. Competition breeds diversity perhaps. Why should art hold a mirror up to nature when a chemical process could do the same and just as accurately? That it is a relatively unknown fact that some of the greatest painters in the West used this trick is due to the secrecy of the guilds. One can go back to Plato and Aristotle to find some of the first theories about the purpose of art. A key innovation in art was Ancient Greek theatre. How it developed is unknown but one theory is that the performances grew out of earlier fertility rituals that involved animal sacrifice, notably goats. Tragedy can

be translated as goat-song[Enc, 1982b]. Another theory is that the Greek plays belong more to a Mystery Cult than anything else. These cults or rituals would have involved a priest of some sort. Eventually a chorus was added. The priest role morphed slowly into a leading actor role. Ritual call and response evolved into semi-narrative. It is often difficult to remember that theatre, painting, sculpture did not arrive fully-formed into the world. Plato and Aristotle were reacting to this cultural innovation when deciding about the purpose and function of art in society.

A number key terms crop up: mimesis, tragedy, comedy, catharsis. Tragedy is high serious that entails an emotional rupture. Comedy is base. Mimesis is authentic reflecting. What I am calling figurative art could also be called mimetic art - art that imitates. The reader may be wondering why this detour-like excursus. It is plain to see that early theory dictated that art be about something. It was imbued with structure and symbolism. Plato argued that art being a pale imitation of reality which itself was a pale imitation of the ideal world of the Forms was therefore a degenerate practice which could stir up the irrational in the populace whereas Aristotle argued that art with proper themes and content could be therapeutic.

Fast-forward two millennia. Whatever the reason the flight to abstraction in art took a very short time once the logic of the process took hold, maybe 30 years. This is an innovation in content, more so than style. We can trace the increasing realisation of this new type of figuration in Kandinsky's work by way of example. Indeed he documented his own experimentation in monographs accompanying his art. Kandinsky himself noted how he was taken aback while looking at Claude Monet's Haystacks painted in the impressionistic style, that he had a moment where he stopped seeing the haystacks themselves and just saw the painting for its formal features of colour and shape alone. But this could only have happened if Monet's painting was already travelling down a path towards abstraction[Enc, 1982c],

> I had the feeling that here the subject of the picture was in a sense the painting itself, and I wondered if I couldn't go much further along the same route. After that I looked at Russian icons with different eyes; I had eyes, that is, for what was abstract ...{pg. 377}

It is in contradistinction to looking up at a sky filled with sparse fluffy clouds or at a rocky outcrop and being struck suddenly that it bears the shape of an animal or some other living being. This coupled with animism leads to the reification of landscape features. What we have here is a cognitive optical illusion where fiction is produced from formlessness - it is a type of over-signification.

Kandinsky's perceptual dissolution of the haystacks is a sort of reverse process, going from fiction to formlessness – and it is a type of under-signification. I do not know the term that is used in psychology to denote these two inverse processes. In the same way Ancient Greek theatre took on the form of theatre from some less theatrical form and perhaps under some kind of external influence be it economic or spiritual or technological. This practice of representing ritualised artifice by symbols, or of investing formal actions or elements (shape, colour) with a symbolic meaning or character is called symbolism. In this theory symbolism – the flight to meaning – would contrast with abstractionism – the flight to abstraction. It could seem to be that aesthetic developments

in art are the result of a tug of war between the symbol and the abstract.

Another key creative mind in the 20th century was Dutch artist M.C.Escher. Few artists break out of the "white cube" into the collective imagination of an era. Dalí is one such artist, Escher is another.

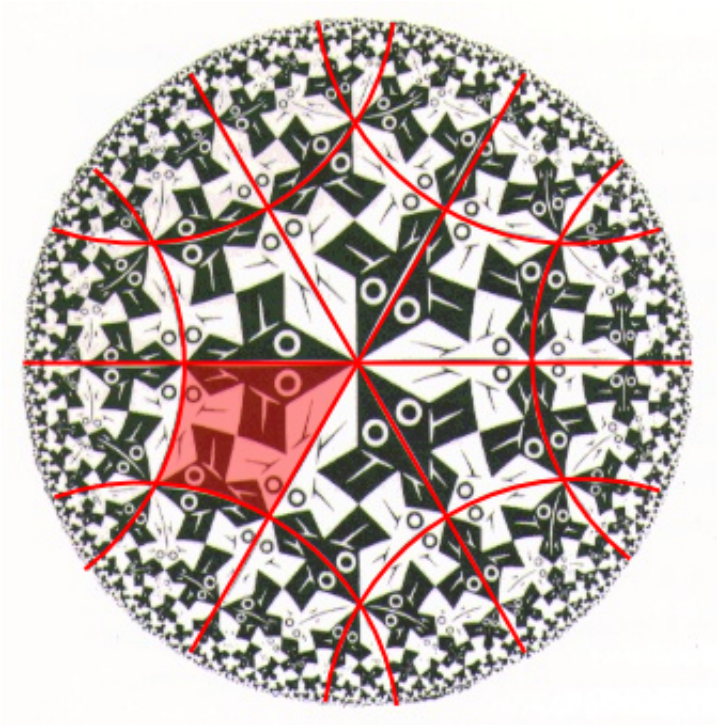

Figure 7: M.C. Escher, Circle Limit I (1958) with geodesics in red.

In 1958 Escher published Regular Division of the Plane, and in this work he says:

At first I had no idea at all of the possibility of systematically building up my figures. I did not know ... this was possible for someone untrained in mathematics, and especially as a result of my putting forward my own layman's theory, which forced me to think through the possibilities.

Again, in Regular Division of the Plane, Escher writes:

In mathematical quarters, the regular division of the plane has been considered theoretically. ... [Mathematicians] have opened the gate leading to an extensive domain, but they have not entered this domain themselves. By their very nature they are more interested in the way in which the gate is opened than in the garden lying behind it.

Escher's meticulous drawings have a signature feel all of their own. Though an individualistic artist I think I have shown how Escher's intricate sketches lie on a continuum of geometric artistic progression. Along with the likes of Mondrian[Tufte, 2001]{pg. 185} he blended graphical elements normally found outside the realm of art with a true artistic elements. He is a transitionary figure to an artistic movement that has come into its own in the digital age – mathematical art.

The machine liberates the artist who explores geometry, symmetry, formula. The artist is liberated by the power of the machine to render the patterns into a visual form. Furthermore the machine can crunch the numbers tirelessly for the artist. The artist is free to explore mathematical space in a way that was simply too time-consuming heretofore. If architecture is said to be frozen music, mathematical art is the opening bar of a sonata – a slice of the harmonic beauty of number made visual rather than aural.

The nod to music is not inapt. As Giovanna Di Rosario has written in an, as yet, unpublished paper[27]:

Benjamin Francis Laposky (1914-2000) was a mathematician, and artist who has been credited with making the first computer graphics. His work in computer art is a form of oscillography, the results of which he has called 'Oscillons' or 'Electronic Abstractions'. Laposky said that "Oscillographic art might be considered as a kind of visual music, as the basic waveforms resemble sound waves". He wanted to reproduce the designs or patterns of natural forms, curves due to physical forces, or curves based on mathematical principles, such as various waveforms (sine-waves, square waves).
Let us look at some pure mathematical artistic pieces now to exhibit the progression taking place nowadays.

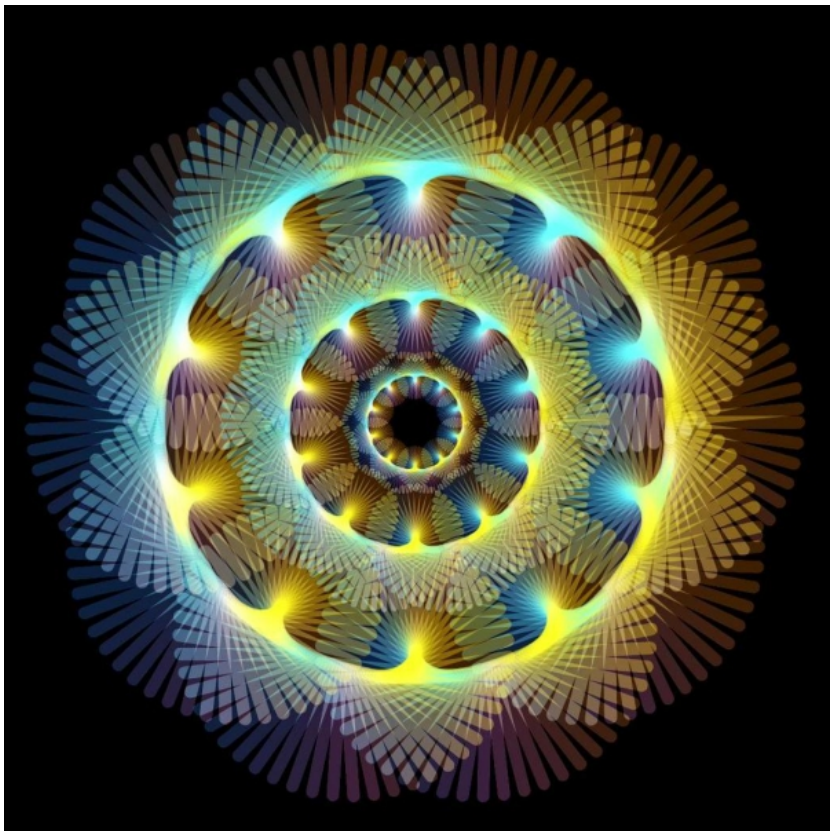

Figure 8: Anne Burns, Complex Flow II (2010) 12" x 12" Digital print. Inspired by the vector field of $f(z) = z^{11}$ along three circles centered at the origin.

Anne Burns works at the department of mathematics at Long Island university, New York. In her paper in the 2011 Bridges proceedings[Burns, 2011] she says that the way she achieves images that she likes is by portraying a continuous complex-valued function as a vector field over its domain. Distinctive features of the functions such as singularities and what she calls their multiplicities can be readily made out by plotting the field over a grid where colours are given values based on functions of length from the origin and/or direction from the origin. By limiting the view to certain paths and by experimenting with different metric and angular functions the colours can be varied. By exploiting rotational symmetry images that the artist finds beautiful can be found.

First concentrating on a square grid whose center was a zero or pole of the function under consideration, $f(z)$, and plotting the vector with tail at each grid point, and length and direction determined by the value of $f(z)$, the pictures were not too inspiring unless an exotic function such as $f(z) = e^{\frac{1}{z}}$ near $z = 0$ was chosen. I began experimenting with drawing the vector fields along paths enclosing singular points and

scaling the length and assigning the color of each vector using a variety of formulas. Even as simple a function as $f(z) = z^n$, for n a nonzero integer, inspired the images in Figures 1 and 3. The subject of vector fields and flows is a major subject for study; however in this paper I will describe just a few ideas for producing interesting and artistic images. These ideas should be accessible to a student with a little knowledge of calculus and complex numbers. Using ActionScript/Flash, animations of the images are easily produced. In this paper a complex valued function of a complex number, $f(z) = u(x, y) + iv(x, y)$, is interpreted as a vector in the complex plane emanating from $z = x + iy$ and the domain is restricted to a small region that contains the "interesting" points, such as zeros or poles of f.

Burns found inspiration from two mathematical sources: the first a book called Visual Complex Analysis and the second an article in the February 1996 issue of Mathematics Magazine titled On Using Flows to Visualize Functions of a Complex Variable. She was taken aback by the diagrams of what she calls "flows". She used the opportunity to explore then as an excuse for learning Actionscript by Adobe Corporation which has at its disposal vector-based graphics capabilities.

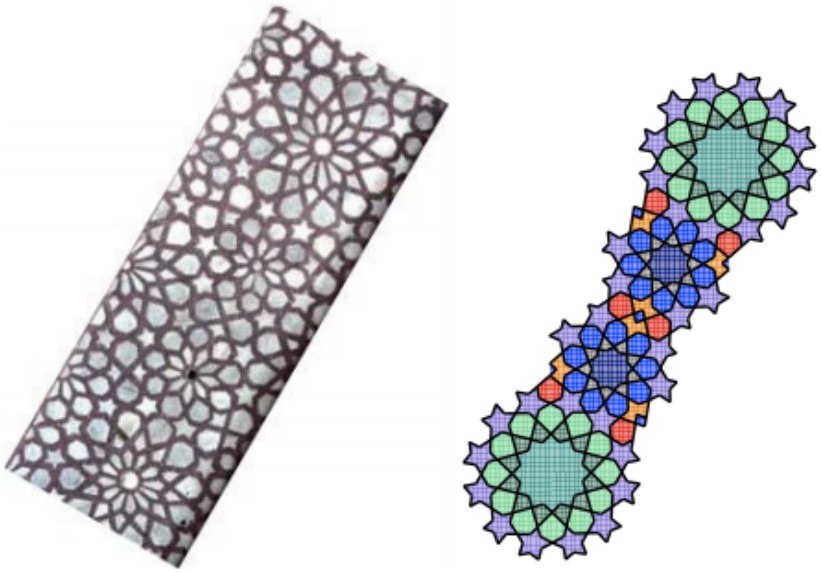

Figure 9: Lynn Bodner, Figure 9 of T9N12 (2011) The Ind 0729 pattern from the Wade collection on the left compared with the author's cropped Tashkent Scrolls pattern on the right.

Next we shall look at the geometric reconstructions of Lynn Bodner. She describes the process of reconstructing nine and twelve star polygons from Islamic geometric tessellations found on a scroll at the Institute of Oriental Studies at the Academy of

Sciences in Tashkent, the present day capital of Uzbekistan. Tashkent is a city whose name means "stone city" and in whose museums rests what is considered to be the oldest Qu'ran on record, apparently dating from 655. Lynn Bodner is at the mathematics department at Monmouth university in New Jersey. In her paper in the Bridges 2011 proceedings[Bodner, 2011] she sets herself the task of reconstructing how it was the original artisans (without mensuration) came about their patterning, a patterning Bonder calls T9N12. The method of compass-and-straightedge is used, a method as old as Euclid. She uses a mathematical software program called Geometer's Sketchpad by Dynamic Geometry.

The pattern as a whole (as you can see from the figure) consists of almost regular nine-pointed, regular twelve-pointed and the left over irregular pentagonal star polygons. The scrolls (the so-called Tashkent Scrolls) comprise fragments of architectural sketches. These sketches are attributed to a guild of architects or an Uzbek master builder plying their trade in 16th century Bukhara. The individual tessellations are called repeat units, they can be repeated by joined up tiling on a plane. Their is also a scroll located at the Topkapı Palace Museum Library in Istanbul, the Topkapı Scroll comprises 114 Islamic ornamental decorative sketches of a similar type to the Tashkent Scrolls, T9N12 does not appear in the Topkapı Scroll. It is put forward by Harvard professor Gülru Necipo□lu that the sketches "served as an aide-memoire for architects and master builders who were already familiar through experience with the coded graphic language used in them".

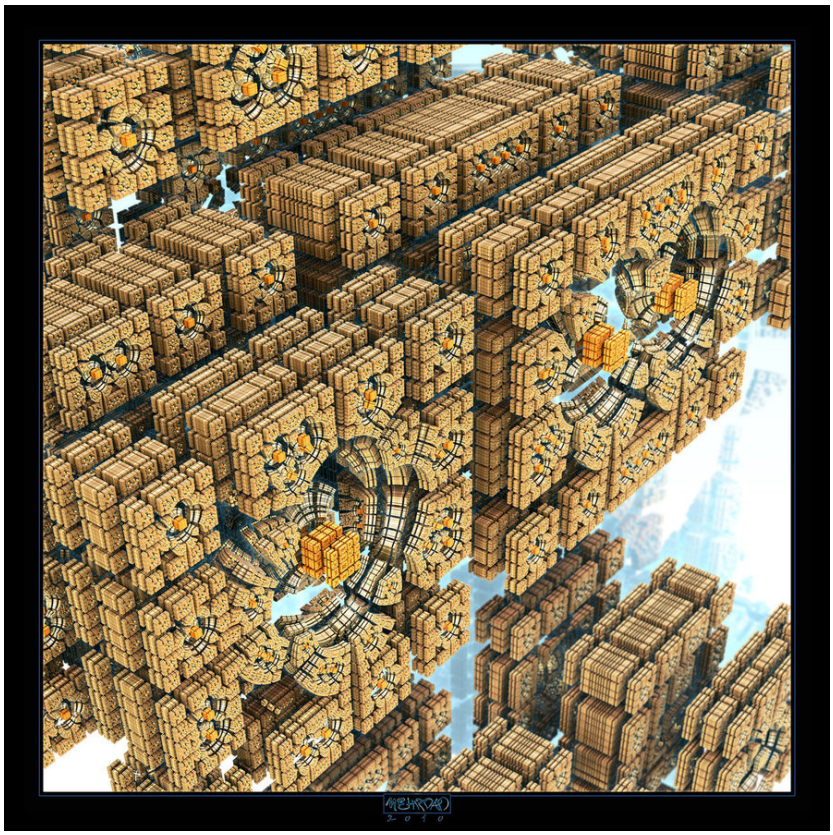

Figure 10: Mehrdad Garousi, Biscuit Land (2010) 20" x 20" digital print

We move now from continuous complex valued functions and star polygons back to the world of fractals. Mehrdad Garousi uses a software package called Mandelbulb3D[28] to produce all his works. This is a package that allows artists to explore fractals in 3D space. There is an exploration window that allows for quick moving around and then at any time a snapshot can be ray traced for a more detailed look. Garousi is not a programmer in the way the next artist is, he is more an artistic cybernaut.

Concerning this work Garousi felt that the landscape he ended up with was composed of a material that reminded him of biscuits, hence the title. He says that the mathematical structure of the composition has been worked according to two connected iterations. There is an iteration that "slices and hatches cubes of material vertically and horizontally [...] one that drills the center of the front face of every sliced cube". The result is something like the above figure with the hollowed out areas being quite unusual with the distinguishing orange cubic shapes left over. As is typical with chaotic iterations very small perturbations in the formula cause unpredictable complexity and behaviour to come to the fore.

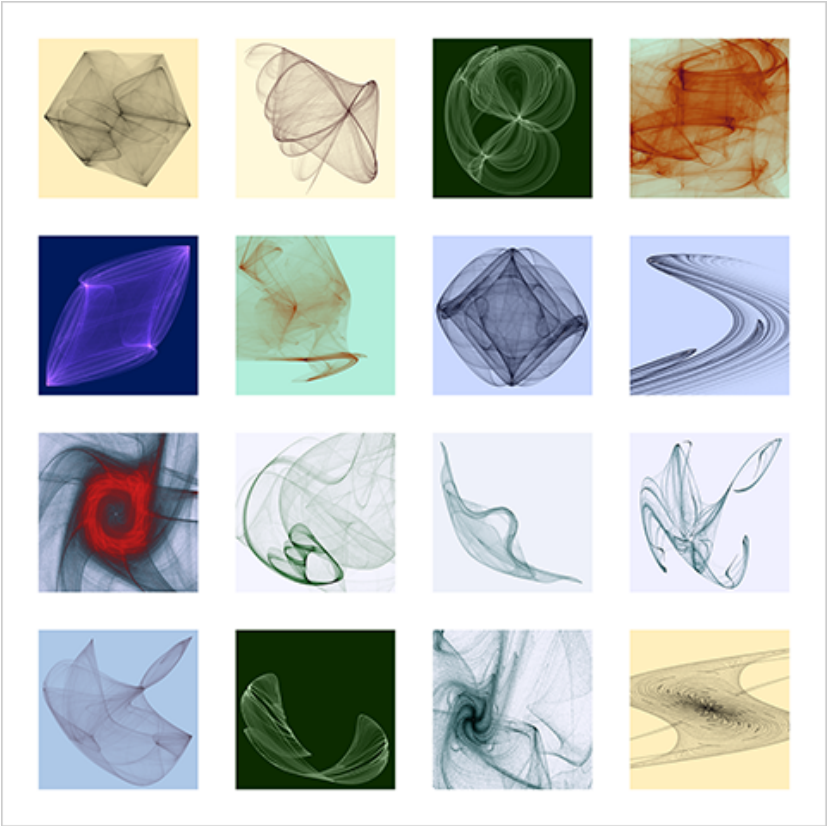

Figure 11: Nathan Selikoff, Aesthetic Explorations (2004) A 4×4 matrix of high quality images, representing a small portion of the parameter space of these particular attractors.

Finally we return to the strange attractors of chaos theory. In an email correspondence with me Nathan Selikoff has communicated that he began "playing" with the strange attractor algorithm in 1999. His artwork has been exhibited in galleries and other venues throughout Europe and the United States. He studied fine art and computer science. He uses a custom program written in the programming language C++, the industry standard 3D api and library OpenGL and GLUT.

Allow me to paraphrase his technical notes. Selikoff first generates his strange attractor artwork with the custom program written using C++, OpenGL, and the OpenGL utility toolkit GLUT. He says that the algorithm that he uses for generating the attractors is detailed in Clifford Pickover's Chaos in Wonderland. The size of the final artwork determines the density of pixels that he needs, between a hundred thousand and a few billion pixels being plotted by iterated functions which simulate the strange attractor equations. He then uses Photoshop to colour in the attractors using gradient mapping with the initial renders in 16 bit grayscale images. How the resulting image of the attractor appears depends on a small number of coefficients and its mutation (that

is to say, which equations it uses).

His program allows him to explore various aspects of the strange attractor he is working on – he can move about the object in simulated three dimensional space by rotating, panning and zooming the viewport into the digital worlds. The program allows him to adjust the parameters of the equation. Once the process has arrived at a pleasing image he save its parameters so that he can then render it another time at a high resolution and it is only at this point that he transfers the images into Photoshop. In Photoshop he is able to tweak the colour, contrast and finer details.

The source code for his custom program has not been published but Selikoff does have code tutorials on his website.

This leads this investigation felicitously on to the next section: an investigation into how the disclosure or lack of dissemination of intagible assets impinges on various human endeavours be it artistic or engineering.

4 The culture and ideology of intangible media (data) and software development (code)

4.1 The origin of copyright

In Common as Air [Hyde, 2011] Lewis Hyde defends the concept of the cultural commons. How has our cultural heritage, the store of ideas, art, and inventions we have inherited from the past, come to be seen as intellectual property? Some people[29] argue that there is no such blanket term as intellectual property and intellectual property rights. They argue against lumping all three categories, copyright law, patent law, trademark law into the same basket as they are quite different beasts. They also argue that the works applicable to these laws very much do not behave like traditional property anyway. Have we , Lewis Hyde asks, taken the concept of ownership too far?

There is a fascinating talk[30] given at the Berkman Center for Internet & Society at Harvard university in which the author of Common As Air discusses that very same book and then takes questions from the floor. This is Hyde's third book after The Gift and Trickster Makes This World (neither referenced here). All three investigate the relationship of artists and their art with the society they inhabit but whereas the first two focus on the possibility of art you might say, the third, the one in question, focuses on the increasing impossibility of creative expression in a world where the intangible has become property and corporations exert more and more control over what is and what is not possible with our store of held-in-common scientific and artistic artefacts.

Towards the beginning of his talk Hyde quotes from one of the classics, work embedded in the bedrock of English literature. Piers Plowman (dating from the 1360s, according to Britannica), or William's Vision of Piers Plowman[31], is the title of a Middle English allegorical narrative poem by William Langland. From it Hyde recalls for us this line, "Human intelligence is like water, air, or fire, shared on earth in common"{5 minutes in}. The concept here is that ideas cannot be chained down. This is a time pre-moveable type and pre-Gutenburg. The most fluid channel for the transmission of ideas was not in print but orally. Print had the benefit of longevity and permanence, but it was difficult to reproduce, and thus difficult to transmit and re-transmit ideas.

What is meant by that line, "shared [...] in common"? It seems simple enough. Neither you, nor I can lay exclusive claim to the fruits of the human mind, what has been wrought by intelligence and imagination is common to all. At that time this was a self-evident uncontested concept. Indeed, in order to prevent others stealing their ideas scientists used to be very secretive. There was no peer review, there were no journals. Many of the greatest scientists and mathematicians closely guarded their discoveries in their own lifetime. I shall not cite any specific cases in detail but I do recall having read that both Gauss and Newton were very secretive. Part of the reason being because there was at the time no legal framework to protect inventors and authors so that they could benefit temporarily from the fruit of their labor, there was no widely held concept of the intangible as property; not yet. (Note that there had been a judgment in the 6th century in Celtic Brehon law enunciated thus, "As to every Cow its Calf, so to every Book its Copy" which was a ruling by Diarmaid, High King of Ireland, against Columba for the act of plagiarizing a Latin psalter of Finian of Clonard.) I am uncertain as to how

pervasive this ruling was or whether it extended beyond the shores of Ireland.

Transferring to the other side of the Irish Sea we encounter the idea of the commons. The notion comes from unusual property arrangements in England (and other regions) in feudal times – say, 9th to the 15th– century where a plot of land could be grazed in common by the livestock of the people in a community. The commons denoted certain access and usage rights. Generally, the land was part of a manor, and a lord owned the land of the manor. There were also cases where the lord would demand a slice of the produce from this shared turf. At any rate the serfs and peasants and plot-holders would share these common plots, it would be held in common, no one could lay claim to it. This was not communism as the state did not own the land – though the crown and the church were the two biggest land owners in the country back then, today in the UK it is the Forestry Commission which is the steward of the largest percentage of land, and thus the common person via the government can lay claim to this slice of the sceptred isle.

From this notion of a physical commons has evolved the notion of a cultural commons: scientific facts, folk tales, old cultural artefacts that are held by the state, the, and so forth. This is why Hyde says, "We all have inherited a cultural commons, and live in it and continue to create", and, "We have access to all the intangible creations", and, "All of this belongs to all of us", and, "All of us can use these things without asking permission".

Into that world arrived paper pulping and milling (exact dates of origin unknown), block printing (7th century), moveable type firstly (11th century) and the printing press (15th century). This was, of course, a very slow process, taking century upon century of refinement. Initially the governing classes (that is the crown, royalty, nobility, aristocracy and landed gentry) and the church were very pleased with these inventions as it afforded them an easier means for disseminating their proclamations, decrees and teachings but they also realised that views and opinions contrary to the orthodoxy would benefit from these tools. Thus the government devised a scheme whereby only certain printing houses sanctioned by state were given renewable license to trade and produce certain works (importing was generally banned) to the exclusion of other printers for a limited number of years; this was the original patent permission, a right to print select works by the crown effectively in perpetuity. Eventually each published work had to be registered. From this state-sanctioned monopoly grew a more powerful entity, called the Stationer's company, a guild of printing houses. In England by the end of the 17th century its power had become too great and in 1695 parliament[3] did not renew its lapsed collective license. Things remained in limbo some 15 years when a historic piece of legislation was proposed, the Copyright Act of 1709 which was put into force in 1710. As this was during the reign of Queen Anne, it has become known as the Statute of Anne.

This law granted a short-term right of ownership to the author of the work. The duration was 14 years, plus a one time renewable period of another 14, giving 28 a possible maximum 28 years in total. This, surprisingly, was a novel concept. And indeed, this concept applied to the physical container, the book, and verbatim copies thereof. Verbatim copying was illegal, the notion that it was the ideas contained with the printed and bound container which were subject to this novel law is not the case.

[3] In the mid-17th century parliament had (as you will recall) replaced the monarchy of the land as the enactor of laws, the United Kingdom had become, via the Cromwellian republic, a constitutional monarchy.

This notion of copyright spread throughout the globe. It was an attempt to solve the problem of public goods. It attempted to balance the wishes of the author, the reader, and the trader. A shocking concept I realise in this day and age where someone can be sued into bankruptcy because they downloaded a handful of copyrighted commercial pop songs from the internet. The word punitive springs readily to mind. But enough about the ills of the 21st century, let us return to the 18th. The reason guilds had sprung up was because they were a vehicle for trade secrets, tricks of the trade literally. Again, the copyright act, and the evolving notion of limited liability allowed inventors and authors to capitalise temporarily on their works and not lose everything through unsound investments. Both of these legal frameworks (copyright and limited liability) protect capital and have underpinned capitalism's rise. Because the tradesman is openly protected, the guilds and their secrets have died out.

As Hyde says in his talk, the idea of property entails certain type of right of action. It entails the right to exclude. In the past, he points out interestingly, works were not copyrighted by default, one had to register one's work, nowadays Hyde notes that the simple writing down of a shopping list or laundry list would by default be copyrighted to its owner. We think that we must "©Anthony Durity, 2011" or whatever but that is not the case at all nowadays, all one needs is proof of authorship; copyright is enabled by default which is in keeping with capitalism's drive to cast the net of private property as wide as it possibly can. Also in the past the copyright act only applied to verbatim copying. Thus, translations and abridgements were deemed fair game. It is was only much later that copyright was seen to to cover the intangible ideas in a work and the ability to spawn derivative works was restricted. A derivative work can be, as I said, a translation, an abridgement, or more up-to-date, a mashup, a remix, a pastiche, and so on.

The idea of what is fair game or not has become enshrined under the banner of Fair Use. The laws define what a person or an institution can do with a work despite it being under copyright. Educational copying is permitted. Personal verbatim copying is permitted. And so forth. There is a notorious law in the United States called the Digital Millennium Copyright Act (DMCA) which allows corporations (or anyone generally, but corporations in practice) to subvert fair use through technological means. Hyde mentions this in passing. It applies to many artefacts in our digital world. The law has been exercised many times and is being actively challenged by the American Civil Liberties Union (ACLU) and the Electronic Frontier Foundation (EFF), who can be viewed as staunch defenders of liberty or crank libertarians depending on which traditional side of the fence one sits.

Hyde show us that while copyright is a boon in that it incentivizes/motivates creativity, it also can lead to "bullying" and "bad behaviour". He gives us two courtroom examples: Diebold versus some students that tried to publish evidence that Diebold's electronic voting machines were flawed – this took many months and sucked up $150,000 in legal fees; the estate of James Joyce versus Carol Schloss – which took several years and $240,000. In the first example the students were brought to court using the DMCA because the voting machines had electronic anti-circumvention mechanisms, the end result is that they were sued under copyright law, which is as bizarre an application of copyright that I can think of. The second example is an example of the abuse of regular copyright law, the term of 14 years is now a distant memory – it is now 95 years in the

States.

I have not related Hyde's interesting tying together of the ideals of a functioning democracy and the commons, I have not mentioned his working out of the concept of agonistic pluralism, and I have only touched on his retelling of the two enclosures or fencings-in, the first - agrarian, the second - more intangible leading to our modern concept of "intellectual property". But it is enough to lead us onto the use of copyright in software licensing.

Fast forward from 1710 to 1970.

4.2 History and snapshot of FOSS

It all started with a printer so the story goes. I have culled most of this story from Free as in Freedom [Williams, 2009] by Sam Williams but the bulk of what I shall describe here is best described as modern folklore. Richard Stallman was working at the Artificial Intelligence lab at the Massachusetts Institute of Technology as a software programmer. Though these people used to prefer to describe themselves as 'hackers'[appendix/endnote]. This was the late 1960s, early 1970s. The MIT AI lab had received a gift from the people over at the Xerox Palo Alto Research Center. A printer, a laser printer in fact. It was many times faster and a good deal more accurate. It suffered from a flaw though, which was that the printer became occasionally jammed. Now as it happened the previous printer had had this problem and Stallman had rectified the flaw by hacking at the software to get it to send a message back over the network to anyone who was waiting on a job in the print queue. Simply put the message said, "The printer is jammed, please fix it." And someone would.

The problem was that in this case this new printer did not come with the source code. So Stallman could not perform the same time-saving hack. When Stallman tracked down an engineer, Robert Sproull it was, of the source who by that time was at Carnegie Mellon (so Stallman's version of the story goes) Sproull refused to give Stallman the source code. Stallman claims it was because the Sproull asserted that he was bound by a non-disclosure agreement (NDA) not to. An NDA is a legally binding contract that requires an employee not to talk about the work they are doing for their employer until a period of time has elapsed. They are commonly used at startups where the idea being developed is novel and where first-mover advantage is key. An NDA is used to protect the innovative and revolutionary nature of a piece of work from companies with deeper pockets or more experience. They are part and parcel of an entire commercial, dare I say, capitalist ethos.

In Williams own words, "Like a peasant whose centuries-old irrigation ditch had grown suddenly dry, Stallman had followed the ditch to its source only to find a brand-spanking-new hydroelectric dam bearing the Xerox logo."

A certain type of hacker ethic[appendix/endnote] had been inculcated in Stallman at the MIT AI lab, this share and share alike ethic had run up against the brick wall of commercial competition. Xerox wanted to monetize the technologies that its researchers were developing, Xerox wasn't a charity, it seems reasonable that Xerox would not want to distribute the source code for its new marvel along with the hardware even though this was the tradition at the time. What it boils down to is how serviceable the item that you have bought is. If something is wrong, should you be able to fix it yourself or

should you send it off to a specialised engineer? Stallman wanted to fix the MIT AI lab printer himself and had been thwarted.

This event acted as a catalyst for Stallman. He had never been overtly political or interested in what was happening in society in an abstract way one imagines. But here was what felt like a personal snub. How much of the story is apocryphal I do not know at this remove but I have relayed the facts as I have them to hand. This was a trend in the hardware/software industry at the time. You must remember that commercial systems were delivered with schematics and were essentially transparent devices. Not that it was unthinkable not to document every aspect of every piece of hardware you sold, firmware and software included but it was just how the industry operated. If a customer purchased one of these devices from you they wanted to know how it worked and how to fix it when it stopped working.

As the machines grew more complex and as the users of these machines became further and further removed from the engineering tradition it became 1) impractical to print pages and pages of manuals, 2) be able to rely on the fact that your customer could decipher those manuals. It cannot be denied as well that there are some genuine commercial motives for slowly coming to the realisation that a company should withhold information from its customer because it firstly disempowers the customer and makes them more reliant on the sales company especially if what that company sells is non-standard and secondly it cuts down on costs, less user-facing documentation is less costly - in a for profit environment where there is no counterbalance to these commercial choices a corporation will only naturally try to maximise its profits.

Stallman's admixture of talents that made him an effective coder, his natural social disposition, his place in academia, the snubbing, the actions of Xerox, all combined to spur him to create programs and work to ensure that the source code for those programs were freely available. From this grew GNU[4], an attempt to provide a complete operating based on free software. Every computer needs an operating system (OS). Stallman's goal was to create the GNU operating system. For that he needed a compiler, the tool to turn source code into binary blobs. For that he needed an editor. This chicken and egg problem is called bootstrapping. A whole sequence of software architectural moves can be called bootstrapping, or the booting up of an OS can be called bootstrapping. Stallman had long worked on an editor called Emacs so he turned his attention to the compiler toolchain which was called GCC, or the GNU C compiler as C was the language Stallman chose to write the operating system in as C was the language used to write Unix. The GNU operating system was to be a Unix like OS. In order to safeguard his work Stallman made the free availability of the source an ironclad agreement by explicitly freeing the source[32] by crafting a license guaranteeing the user certain rights while enforcing certain obligations on the user – a kind of quid pro quo agreement – the user gets access to the source so long as the user agrees to release any modifications that he or she makes to the code in the event that the user redistribute the software. As this is seen as a hack, repurposing copyright law to enforce sharing these types of licenses are called copyleft licenses, sometimes denoted so: ⓒ. If a piece of software is covered by a copyleft license it is free software. The license was called the GPL, the GNU general public license, it now stands at version 3 and it is this social

[4]GNU(4) is Not Unix

and legal hack that has changed the course of the industry of software development. The software stack is called GNU, the foundation that manages the stack and legal challenges and everything else is called the FSF, free software foundation.

Stallman's foundation never built a full OS because their kernel (the core of the OS – the bit that talks to the hardware, the bit that manages the applications) never really took off. Instead a young Finnish man named Linus Torvalds in building a Unix kernel clone for himself called Linux succeeded in this task. This is why some people argue that Linux should be called GNU/Linux because it is a combination of both these vital pieces of software. At the time of writing, Android, Linux's mobile successor backed by the wealth and technical ability of Google is set to become the world's leading smartphone OS, and as smartphones are set to become the dominant mobile form-factor this means that free software in the space of 40 years, 20 initially by GNU alone, the following 20 in conjunction with Linux has disrupted an industry.

The story of this ideological revolution in the process of software development has been documented in many places. Why I am documenting it here is to contrast the history of the practice of software engineering with the social ethos that evolved around free software. This we will turn to for a moment.

The public surface of a commercial piece of software is limited to the product itself; the semi-faceless corporation behind the product; the marketing, public relations, and advertising spend of the corporation; the technical support; the trade-show booths; and nowadays the corporate website which can serve as a technical point of contact or merely as an electronic brochure.

Contrast this with most FOSS projects which, if they are large, are backed by foundations: think FSF, think KDE, think GNOME, think Apache, think LibreOffice, think Linux. These foundations are non-profits generally, but they often receive commercial backing – so rather than a corporation making an open source play they'll direct funding towards a semi-autonomous foundation. These foundations are stewards of one chuck of large software with many component parts – the way in which Apache and KDE are umbrellas for an ecosystem of projects that taken together have a large footprint. Smaller projects will involve a loose grouping of individuals donating their time and resources. That is not to say that there are no corporations that build their business solely or too a large extent around FOSS. The prime example would be Google which has leveraged Linux to build the mobile operating system Android and also leveraged WebKit to build the web browser Chrome. The money to fund Android and Chrome comes from the search giant's revenue stream in internet advertising. Do companies make money from selling FOSS products? The answer is yes. One of the more successful software companies in recent years has been the poster child of the FOSS world, Redhat. They create a distribution of Linux called Redhat Enterprise Linux where they assemble all the pieces, make sure it all works and then provide commercial support for their efforts. Redhat have been so successful and profitable that they employ a large number of engineers to fix key pieces of software in the GNU/Linux ecosystem. This holds true even if these pieces of software are backed by foundations. The commercial support model is how many FOSS companies derive their wealth. When you think about it, this makes a lot of sense. The FOSS ethos means that the burden of engineering can be shared but because the source is freely available it is often an simple matter to get your hands legally on binary copies of these pieces of software gratis. What makes

them useful lies in the complexity of their operation. It makes sense that people will pay people for know-how and expertise in managing this complexity and as go-to points if something should go horribly wrong.

4.3 The tar pit: software programming is hard

Software engineering even though it is called engineering is a far from precise discipline. The history of software developments is littered with tales of mammoth projects suffering mammoth overruns both in terms of cost and allotted time. Software engineering is not like civil engineering where the cost, length of time, amount of labour, materials and so forth can be calculated pretty accurately beforehand. I am not saying that there are not engineering works that are fraught with disaster, are aborted before completion or have many interrelated unknown quantities. Rather the older engineering disciplines have had millennia to solve the intractable problems they face. And once a problem is solved (the flying buttress, the arch, the suspension stay, and all the rest of it) they stay solved.

There are theories of software development. There are models that organisations follow. They vary by team-size, organisational complexity, bureaucracy but they are all very alike. There is a piece of software (a software system of indeterminable size) that must be built to fulfil a purpose; the people who know what that purpose is are interrogated; specifications, the equivalent of software blueprints are drawn up, those specifications are turned piece by piece into code; iterate until completion; launch product. There are many best practices at every stage that have been discovered over time: optimal team size of individual teams, test-driven design, code reviews, bug and feature repositories, code revision systems. And still projects go over budget and over time.

A manager in IBM called Fred Brooks noticed this and set about analysing in a semi-scientific way the practice of software design and the result was a book, The Mythical Man-Month[Brooks, 1995], that has become a sort of software developer's bible along with a number of other sacred texts like Design Patterns by Gamma et al. [Gamma et al., 1994]. Brooks notes in the preface that his slim tome had sold over 250,000 copies by its 20th anniversary edition, the edition I have accessed.

The mythical month to which the title of the book refers is the idea that if you have X amount of men (they are always men, sorry ladies) working on a project, i.e. the infamous ditch digging, then if one speeds up their rate of work twofold or doubles the size of the team then the time taken to dig that ditch or to produce a N widgets in an assembly line will decrease by one half. This is a the Fordist model of production. What Brooks noted was that counter-intuitively adding more engineers to a software project that was already experiencing delays caused even more delays roughly speaking. Why is this? Building software is not a linear process, it appears that there is a significant amount of overhead in terms of communication between team members that only is exacerbated extra team members and furthermore the cost of training up new members on any non-trivial project eats into the time that has been set aside for actually building the product. You also cannot assembly line software construction because each independent module in the project must be more or less near a state where it can be provably working at any given moment which means that all team members working on individual modules must be working in a coordinated parallel fashion. pg. 17 "The bearing of

a child takes nine months, no matter how many women are assigned".

Now it is interesting that Brooks notes that certain types of system are harder to produce than others. If we take a standalone program as our unit of measurement he claims that with the experience of time he came to believe a certain rule of thumb. There is a threefold increase in outlay if one wants to turn ones program into a programming product, this means that it is generalised, tested, documented, maintained. Similarly there is a threefold increase in outlay if one wants to turn ones program into a programming system, this means that its interfaces are regular and that it can be integrated into systems. Finally combining both to get a programming systems product (like a portable operating system say, or a portable office suite, or a portable compiler tool chain) takes nine times as much effort.

The fourth chapter claims for any large software project that the vision of the software as a totality must go through one mind. Every chapter of The Mythical Man-Month features an engaging piece of artwork, in black-and-white. A beautiful representation of Reims Cathedral is the piece of artwork for the fourth chapter which is entitled Aristocracy, Democracy, and System Design.

Eric S. Raymond has a hypothesis that there is a silver bullet of sorts - the methodology of libre software. In a seminal essay that has since been published though it is available freely for anyone's edification on the interwebs is The Cathedral and The Bazaar. If you have already guessed that the cathedral in question is Brooks's cathedral then you would be right. Raymond's suggestion is that the very process of adhering to a model where the source code is freely available circumvents some of the age-old arguments that Brooks rose. He then has a pragmatic outlook on libre software compared to Stallman's ethical outlook. The potential problem with Brooks's analysis is that software programming and development had essentially become isolated activities that only allowed software reuse within those silos. For many years commercial software developers had been hyping one or more models of reusability that did not include source reusability or modification because the business model around software had coalesced into the idea that you either sold programs, building block components (programming products) with interfaces for stacking and layering, or programming systems products, again without the source barring exceptional circumstances like you were selling to the government or the military.

The Cathedral and The Bazaar is structured by a series of pithy aphorisms designed to inculcate in the reader the essential traits of good open source development. Number 6 is, "Treating your users as co-developers is your least-hassle route to rapid code improvement and effective debugging." He goes on to say, "The power of this effect is easy to underestimate. In fact, pretty well all of us in the open source world drastically underestimated how well it would scale up with number of users and against system complexity, until Linus Torvalds showed us differently." And then in the next paragraph, "In fact, I think Linus's cleverest and most consequential hack was not the construction of the Linux kernel itself, but rather his invention of the Linux development model. When I expressed this opinion in his presence once, he smiled and quietly repeated something he has often said: «I'm basically a very lazy person who likes to get credit for things other people actually do.»"

Fred Brooks's claim is that the non-parallelisable portion of software development is that half of the schedule should be devoted to testing of one kind or another, component

testing and early system testing, system testing itself, and beta testing. The Linux motto as enunciated by Raymond is: given enough eyeballs all bugs are shallow.

4.4 Legal, economic, ethical, epistemological angles

Legal, FOSS is bound by licenses backed by the legal system – economic, it is a qualitatively different method for developing software and there are monetary implications both pro and con – ethical, there are some who argue that free software is a socio-ethical phenomenon and others who argue from a position of pragmatism – epistemological, by which I mean the interplay of FOSS with the methods of scientific discovery and how we come to know what we know.

As the reader will have been able to make out from the potted history I have provided several cases have been made for the effectiveness of one method of software development over another. The most radical departure from the engineering orthodoxy is the stance of Stallman and people who, like him, believe that the issues surrounding the sharing of code are ethical issues. They believe that it your (moral) duty as a good neighbour to share your code. Others, like Linus Torvalds and many others, believe the what is at stake is more the pragmatic decision of building better software. They believe that sharing source code leads to higher quality software, and what software engineer would not want that?

Legal matters are another concern, free and open source software licenses are surprisingly simple legal documents. They provide certain assurances, they demand certain requirements. They come in different flavours depending on which freedoms they tend to emphasize.

Rights to a tangible or intangible entity (creative work) are generally seen as legal rights. With regard to the paternity of the work lawyers now tend to differentiate between the moral rights of an author and the economic rights of an author. The reason that they are seen as legal rights is because they are not seen as part of the inalienable rights of a person but stem from these inalienable rights. Moral rights differ form economic rights in that moral rights pertain to issues of attribution - requiring that the correct author be attributed or creating a work anonymously or pseudonymously and having that veil protected. Note that these moral rights have nothing to do with ethics, they are a concept of copyright law. Moral rights also pertain to the integrity of the work, it is the moral right of an author that is affected if the work has pieces removed due to censorship. The economic rights of an author obviously pertain to the ability of the author to receive just renumeration for the use of the fruits of their creative efforts.

4.4.1 Legal: no, all, or some rights reserved

Public Domain (no rights reserved) Works in the public domain informally refers to creative works that are publicly available, free for all to use. Formally a work may enter the public domain or always have been in the public domain because the rights tied to the work may have been forfeited, are inapplicable or may have expired. Examples could be:

- forfeited: An author may waive their rights on a work allowing that work to enter

the public domain. This varies from place to place so that in some jurisdictions an author must assert their moral rights and economic rights in order to benefit materially and protect against infringement - in other jurisdictions these are accrued to the author by the act of creation and must be explicitly waived.

- inapplicable: This concept relates to prehistoric natural languages like Italian, Finnish. It could perhaps apply to invented natural languages like Esperanto, Loglan, Klingon, Elvish and so forth but this has never been contested legally. As this thesis goes to print Oracle is suing Google over the copyright to the programming language Java, the first such case – it would set a precedent if Oracle were able to assert copyright over the language itself (rather than a particular implementation). A ruling has been passed down at the European level upholding Google's position.

- expired: When the copyright on a work expires it enters the public domain. Works predating copyright law are seen as part of the common heritage and folklore.

Full copyright or copyleft (all rights reserved) Copyright has been the standard law pertaining to creative works since the statute of Queen Anne in 1790. This has changed in four significant ways down through the years. The first change is the length of the term of the copyright has been growing ever longer as time goes by. What was once a method of protecting authors from publishers (and nowadays users, but in those days copying was a privileged task - one had either to be literate or own a press, both privileged positions) is now a method of extracting long term revenue from a copyrighted work. This stifles reuse and remix culture. The second is that corporations are now legal persons and can own assert copyright over copyrightable works. The lifetime of corporations can be measured in the centuries and these corporations continually press for laws to be amended with longer and longer copyright terms. The third change is copyleft which we have discussed before and it is a product of our digital world and the essential change in the nature of digital works from traditional works. The fourth is the most recent innovation and can be thought of a middle ground, some rights reserved, examples of which are given in the next subsection.

The GPL, BSD, MIT, Apache software licenses all rely on strong copyright law. When situations occur where someone or some corporation uses a piece of software that is covered by a copyleft license and does not adhere to the terms of the license they are in breach of copyright. Usually in the case of a traditional work this would entail republishing the work without permission, in the case of copyleft the breach would involve making modifications to a piece of software and redistributing it without making those changes available, for example. Transgressor must be rigorously sued for copyright infringement or the licenses will be seen as having "no teeth".

The Creative Commons (some rights reserved) In electronic media other than software a use case was found where it might be desirable to give up some but not all rights to a work so that the work could be remixed/reused/mashed-up. This works especially well for audio and video. Oftentimes an author will just want to be credited for their work and not care at all what is done to the work itself - ergo, they are concerned with

attribution and not the integrity of the work. This allows for finer-grained control over the work, a more measured and subtle letting go, a more nuanced approached than the all or nothing approach of public domain or full copyright. This thesis for instance is licensed under permissive creative commons terms with the knowledge that it owes its existence to the hard work of many others and though it is a distillation and reworking of many minds is not a work of strongly original character and thus does not need the full rights of a usual creative copyrighted work.

4.4.2 Economic: the new economy

That brings us the final part of this section – the new economy. The rise of digital media has brought the notion of piracy in from the high seas and into our living quarters. The notion of piracy rests again on the notion of property, property in transit that is stolen. When a person illegally downloads a tune, illegally streams a movie, illegally copies a piece of software it is said that they are involved in piracy. The piracy of intangible media is not like the piracy of the high seas because when a person consumes one of these items they have not taken some piece of cargo in transit and furthermore when intangible media is copied the original is not destroyed so what we have here is a proliferation of goods. It is not theft in the traditional sense. Indeed, as recently as this month, Goldmann Sachs sued a former employee Sergey Aleynikov for millions of dollars because the employee had leaked some of their software – they sued for code theft and espionage. Unfortunately for them and in accordance with centuries of legal tradition the court ruled against them because code is not seen as tangible property by the law[33] and so cannot technically be stolen and so Mr. Aleynikov while he did cause Goldmann Sachs harm by releasing proprietary code he cannot legally be said to have stolen it and so he cannot be charged with theft. The most he could be charged with is copyright infringement which is not legally theft.

Chief Judge Dennis Jacobs wrote as part of the court's unanimous decision: "Because Aleynikov did not «assume physical control» over anything when he took the source code, and because he did not thereby «deprive [Goldman] of its use,» Aleynikov did not violate the [National Stolen Property Act] NSPA."

Jacobs went on to add, "We decline to stretch or update statutory words of plain and ordinary meaning in order to better accommodate the digital age,". And furthermore, "The enormous profits the system yielded for Goldman depended on no one else having it," Jacobs ruled. Pointing out that the code was not to be sold on or licensed to others, he added: "Because [the high-frequency trading system] was not designed to enter or pass in commerce, or to make something that does, Aleynikov's theft of source code relating to that system was not an offense under the [Economic Espionage Act] EEA."

The same logic extends to other forms of digital media.

Whether a person excludes another person from a space in the world by enclosing it, by fencing it off, either virtually or not it could be said that this person wants to monopolise this part of the world. A common contemporary myth is that information wants to be free – what is true is that digital information is very fluid and is not generally destroyed as it is copied from place to place. Artificial barriers must be erected to counteract this fluidity. Technical barriers involving encryption and electronic signing called digital rights management (DRM) is one mechanism, legal barriers such as copyright

infringement is another. These are issues of monopoly and plurality, control and freedom, transparency and opacity. Another method employed is in to control the pipes. Tim Wu shows in The Master Switch that every telecommunication (the internet is the fifth) system to date has ended up in the control of a monopoly or oligopoly. Another internet age-myth is that the internet is impervious to this sort of restriction, the internet is somehow inherently open. Wu shows that historically this is a recurrent belief, a belief that has always turned out to be false.

Specifically, what we need is something I would call a Separation Principle for the information economy. A Separations Principle would mean the creation of a salutary distance between each of the major functions or layers in the information economy. It would mean those who develop information, those who own the network infrastructure on which it travels, and those who control the tools or venues of access must be kept apart from one another. At the same time, the Separations Principle stipulates one other necessity: that the government also keep its distance and not intervene in the market to favor any technology, network monopoly, or integration of the major functions of an information industry.

Wu advocates a type of Separation Principle along the lines that are meant to keep congress, the judiciary, and the executive office in the constitutional republic of the United States separate. This separation of powers involving a tri-partite structure is meant to give stability and balance to the nation. Lawrence Lessig argues from the level of protocol in Code 2.0 – he shows that there is nothing in the electronic architecture of the internet protocols that should bear out the notion that the internet is impervious to capture,

This regulator is what I call "code" – the instructions embedded in the software or hardware that makes cyberspace what it is. This code is the "built environment" of social lie in cyberspace. It is its "architecture." And if in the middle of the nineteenth century the threat to liberty was norms, and at the start of the twentieth was state power, and during much of the middle twentieth it was the market, then my argument is that we must come to understand how in the twenty-first century it is a different regulator – code – that should be our current concern{pg. 121}

Both Wu and Lessig caution vigilance. I would argue that there is a propensity to think of the internet as a public space. This is a misconception I would argue. It is as, I said earlier, more correct to view it in an infrastructural way as a network of networks with a hub and spoke layout. All the sites, on the web at least, are interconnected by the zero-dimensionality of the hyperlink – this makes it a densely packed space of private spaces with no public space whatsoever. It is only if the site's system allows for public participation, as in a wiki, that any of the spaces could be said to be public. I can emphasize this point by pointing out that it is impossible to have a public demonstration on the internet, the most that is possible is to mount a distributed denial of service attack on specific private spaces. Similarly there are no public spaces on which to scrawl graffiti Banksy-like, the internet and the web just are not shaped like this topologically.

Finally, from an economist's point of view Yochai Benkler argues that,

Perhaps the most amazing document of the consensus among economists today that, because of the combination of nonrivalry and the "on the shoulders of giants" effect, excessive expansion of "intellectual property" protection is economically detrimental, was the economists' brief filed in the Supreme Court case of Eldred v. Ashcroft. The case challenged a law that extended the term of copyright protection from lasting for the life of the author plus fifty years, to life of the author plus seventy years, or from seventy-five years to ninety-five years for copyrights owned by corporations. If information were like land or iron, the ideal length of property rights would be infinite from the economists' perspective. In this case, however, where the "property right" was copyright, more than two dozen leading economists volunteered to sign a brief opposing the law, counting among their number five Nobel laureates, including that well-known market skeptic, Milton Friedman.{pg. 38}

Intangible media and code is subject to nonrivalry. Unlike physical stuff, physical objects; where if I buy the stuff I deprive you of the object, intangible goods do not operate in this fashion and are said to be nonrival. And, if an idea is let loose in the world it can be built on – plot lines can be reused, algorithms duplicated. Mathematical software is subject to all the perspectives given in this section. The production of mathematics as it comes to depend on mathematical software will adjust to the requirements of the law rather than the requirements of science.

4.4.3 Ethical: pragmatism versus ethics

It may not be immediately obvious how the choice of which software one runs on ones computer can be an ethical choice. As I have mentioned before there are actors, like Linus Torvalds, within the free software and open-source world who view their choice to develop non-proprietary software as a pragmatic decision – that is to say, they view that the free software methodology produces better code than the proprietary methodology. "It started gaining attention when it was obvious that open source was the best method of developing and improving the highest quality technology."
[Torvalds and Diamond, 2002]{pg. 227} The flip side of this is the ethical angle. Someone, like Richard Stallman, views this choice in ethical terms, they believe it is morally wrong to develop proprietary software for distribution because, they say, that decision has anti-social consequences. "Stallman nevertheless can take sole credit for building the free software movement's ethical framework."[Williams, 2009]{ch. 13} In response the proprietary world has responded to the threat from Linux and the free software movement with outright hostility – witness, "Linux is a cancer that attaches itself in an intellectual property sense to everything it touches,"[34]. This suggests the camps are aligned along cultural and ideological fault lines lending credence to the belief that indeed the consequences of ones decisions in this respect are in fact partly ethical in addition to pragmatic.

4.4.4 Epistemological: the case for open computer programs

As I was preparing this thesis an article was published in Nature that was most fortuitous. It speaks directly to the epistemological conundrum I have alluded to a number of times. The argument goes thus. The methods of science require that all the parts of an experiment be reproducible, to satisfy the requirement of testifiability and falsifiability. Prior to the complex software systems of the last 50 years this amounted to merely having access to the exact same tools calibrated in the exact same ways as other scientific teams. To the best of my knowledge the theories underpinning science never required absolute transparency of the tools in question. In practice once a tool is engineered to a sufficient precision, to within a certain tolerance of error then all is well. This is not possible with software as the examples from chaos theory show – essentially one would need identical hardware and software setups.

The authors, Darrel C. Ince, Leslie Hatton & John Graham-Cumming, argue the case for open computer programs. I shall weigh the matter myself, then I shall present their views.

I mentioned earlier about Gauss and Newton being quite secretive in their work. Peer reviewed publications simply did not exist as a concept a few hundred years ago – it is a relatively recent modern invention. It took some time before it was realised that a culture of openness is necessary for evidence-based science. Copernicus went to work as an assistant for Tycho Brahe because Brahe kept the most meticulous calculations but he in no way shared that data with anyone outside his astronomy lab. As the philosophy of science developed in the twentieth century the idea that you could simply assert a proof and let it up to others to believe you or not fell to one side. It came to be seen that no theory is ever fully proven but merely the best candidate or model for measuring and predicting reality out of many competing theories. The best one can do is falsify a theory, disqualify it, one can never categorically prove a theory. Karl Popper was instrumental in making this worldview widespread – and this was in the early to mid 20th century so not that long ago at all.

Like all other fields, science has become computerized. To my mind this means that there should be a certain elevated requirement for computer programs used in scientific research – there are epistemological concerns I believe. I wouldn't go so far as to say that withholding the source code for the computer programs used in scientific research is anti-scientific but I feel the act does reside in a questionable place on the continuum of scientific integrity. Stallman contends that all proprietary software is essentially anti-social. While this line of reasoning seems plausible I feel that a strong claim such as this is unnecessarily provocative and would only truly merit attention if some kind of legal framework was being put into place to make free and open source software an untenable pursuit. I do feel that in the area of scientific research some kind of strong claim could be made that the withholding of source code is anti-social. As mathematics is such a large part of science this places a certain extra burden on the makers of mathematical software I feel – in some ways you could say that we have regressed to the time of Newton and Gauss in this one respect. This is surely a less than ideal situation. These men profited from their secrecy, we must ask ourselves if this is the situation we want socially as time progresses and computers and computer programs become even more embedded and enmeshed into the fabric of society.

Here is the abstract of the Nature authors,

Scientific communication relies on evidence that cannot be entirely included in publications, but the rise of computational science has added a new layer of inaccessibility. Although it is now accepted that data should be made available on request, the current regulations regarding the availability of software are inconsistent. We argue that, with some exceptions, anything less than the release of source programs is intolerable for results that depend on computation. The vagaries of hardware, software and natural language will always ensure that exact reproducibility remains uncertain, but withholding code increases the chances that efforts to reproduce results will fail.{abstract}

They have two real life data sets which contain these types of failure: data sets known as HadCRUT and CRUTEM3 produced by the United Kingdom Meteorological Office with the aid of the University of East Anglia's Climatic Research Unit, data sets used by geologists to site extremely expensive marine drilling rigs.

They say that there is the practice of including textual descriptions of algorithms (they do not use the term pseudo-code by I have seen that term used in similar contexts) but they go on to say that this is inadequate. They say that the source code that supports some published scientific result (or data set) must be released – owing to ambiguities in natural language, variability in hardware, and floating point representations, in conjunction with programming errors.

They cite 4 reasons: the encouragement of non-institutional researchers, the benefits of extra teaching material, reusability of modules, improved scientific accuracy quicker. I couldn't possibly do justice to the entire article – I encourage anyone with interest in the area to read it, it is quite brief, lucid and approachable.

This concludes the general investigation into software practices, culture and ideology – as mentioned earlier the next dives into a couple of specific mathematical software ecosystems.

5 Specific comparison and questionnaires

5.1 Compare and contrast open versus proprietary tools

Now that I have given a detailed description of the software development landscape in general it is the correct time to make an analysis of one particular category of software, namely the software of interest to this thesis, mathematical software. This is by no means a forensic analysis of code quality, working practices, and so forth. Instead a questionnaire was posed to two groups of people, there was no time limit imposed. When all the respondents had come back to me I made an analysis of the results and went back to them with a number of follow-up questions tailored to each individual. Not everyone who had responded the first time responded the second, unfortunately. As I point out below, owing to some badly structured initial questions couple with a regrettable lack of precision my follow-up questions were therefore an attempt to clarify earlier fudges. This explains the unfortunately.

Mine was not a large questionnaire, nor were the number of responses overwhelming, nor were the responses incredibly detailed. What I succeeded in obtaining was an (I believe) authentic snapshot of the state of mind of the actors in this particular niche. Because of the small sample size, and because I see no reason not to, I am publishing the actual responses as-is, bar correcting a few typos. I am randomising the comments because I see no reason to allow anyone to reconstruct any particular individuals totality of responses – I am interested in the aggregate response, the group response.

I will turn first to the open source crowd, then to the proprietary crowd.

5.2 Sage

5.2.1 Lead-up to open source questionnaire

The Sage mission statement according to its website: "Create a viable free open source alternative to Magma, Maple, Mathematica, and Matlab".[35] A "viable alternative" will have five distinguishing features (and I paraphrase): it will have the mathematical features of Magma, Maple, Mathematica, and Matlab with comparable speed; it will have beautiful interactive 2d and 3d graphics; it will have a notebook interface and an integrated development environment; it will provide many educational books (full undergraduate curriculum); it will come with commercial support (e.g., customised notebook servers).

As you can see, the goal of Sage is to disrupt a multi-billion dollar niche software industry. Why would a group of academics want to do that? How successful have they been? In order to answer these questions I asked the project members eight questions that I hoped would essentially elicit their opinions without bluntly posing those exact questions in quite so pugnacious a way. Here are those questions.

1. :- What is your role and how long have you been involved in Sage Math?
2. :- What is your background and what motivated you to become involved in the project?

3. :- What is your personal opinion about the quality of Sage Math and the quality of open source mathematical tools in general? How does this relate to proprietary tools?

4. :- What is your opinion of proprietary software mathematical tools? Do you use any?

5. :- What is your opinion about the GPL and free software licenses in general?

6. :- In the broad arc of scientific research and development, how do you see your involvement in Sage Math?

7. :- Are there any views and opinions you would like to share with the world regarding these topics: consider this your soapbox.

8. :- Is there anything I am not asking that you think I should be asking?

I received seven answers after a little prodding and cajoling. I'd like to take this opportunity to remark upon the journey to receiving these responses from this FOSS project.

I initially asked on internet relay chat (IRC) as to whether participants in the project, be they users or developer, would like to take part in a survey on open source mathematical software for my thesis. I was directed by a kind and helpful soul to the developer mailing list. I subscribed to the mailing list (to which I am subscribed to this day) and posted a message with the questionnaire in OpenDocument (text) format, a format I presumed would be amenable to such a group of individuals. I received but two responses and I thought that this part of my thesis was scuppered. A number of people replied to me saying: this is a nice idea but you should have made this a web survey because it is handier or because it would be possible to reply anonymously. And also: why was I asking so many detailed personal questions in the questionnaire. This annoyed and bemused me because a) the document was to hand and in a free (as in freedom) format as an attachment to the email, b) I was not asking particularly sensitive information and I had guaranteed I would not disclose the identity of the recipients, c) I had let it be known that they could answer as much or as little as they liked. After waiting a couple of months and receiving no other replies in addition to the first prompt and generous two I relented. I turned the document I had previously prepared into a Google Documents form that is at heart a spreadsheet and posted it again to the list. I also suggested that there would be a material reward in the form of a piece of Arduino hardware. This time I got five responses, not one of which chose to enter the data anonymously and not one of which were any of the people who had harangued me about the sloppiness, ill-judged nature, or the insensitivity to issues of privacy and identity of my initial post. Moral of the story: consider giving people multiple avenues in which to respond, do allow anonymity and pseudonymity unless one has a good reason against, realise that people can be quite willing to create more work for you while having little intention of contributing themselves.

The exact questionnaire forms, both of the OpenOffice OpenDocument (text) .odt variety and the online Google Docs variety are included in the appendix of this work. As you can see they are the same barring the requirement of personal data. Before I get to the answers I was given I would like to give William Stein's views on what the impetus

for the project is. I am culling this information from an introductory video[36] that is hosted on Sage's website. Stein is one of the project's founders – and so is undoubtedly in a key position to be able to articulate these views. He is a professor of mathematics at the university of Washington.

According to Stein the requirement that Sage adhere to the mathematical features of the Big Four allows them to easily decide what parts of mathematics to include in their project. Furthermore, a feature is not deemed acceptable unless it performs as speedily as the feature in a commercial product it is emulating. The need for beautiful interactive 2d and 3d graphics stems from the desire both to use Sage as a valuable teaching aid (and so to give it a push into academia) and from the necessity of being able to adequately visualise the mathematical symbols and objects you are prototyping and building. With respect to the notebook interface and integrated development environment (or IDE to use its shorthand) Stein says that both Emacs and Eclipse have been specialised to allow programmers to get programmatic access to Sage. While this is fine it is obviously only suitable for programmers and is a barrier to regular users of mathematical software. A notebook interface would bring the familiar pen and paper interface to a larger audience as the preferred method for interacting with Sage. To that end Stein says that in about 2007 he and Alex Clemesha developed a web-based graphical interface for Sage, choosing the web model as it is cross-platform and allows for collaboration. In order for Sage to be a viable platform for use in academia it is necessary for the students to have high quality documentation that explains how to use Sage itself and also how to work with certain areas of mathematics in Sage. Stein points to the website[37] which contains books on differential calculus, elliptic curves, modular forms, linear algebra and more. A final point emphasises that institutions typically need the assurance of paid support in order to be able to have the knowledge that they can get someone on the end of an email or phone when they run into trouble. The hacker or open source mantra of RTFM (if you'll beg my pardon) does not apply here.

Sage (which at that time was an acronym for Software for Arithmetic Geometry Experimentation) was started in 2005. David Joyner soon joined. Stein had a desire for a system like Magma for his personal research and for his teaching, but he needed it to be open source. Joyner had a need for the coding theory functionality of Magma so together they built on top of PARI[38] which had been started in 1985 in France and GAP[39] which had been started in 1986 in Germany. PARI provides number theory functionality, GAP provides group theory capabilities thereby providing coding theory capabilities. The glue in in this is the general programming language Python.

Why was Magma not good enough for Stein and Joyner? One must take into consideration that Stein had used Magma for 5 years and had contributed thousands of lines of code to Magma. The following reasons are given:

Expensive for collaborators; This means colleagues and students had to part with quite a lot of money to work with each other. Though Magma had discounts for people outside the US (a so-called "third world discount") it typically cost several months salary depending on the country. This was seen as a hinderance to frictionless collaboration if a vibrant research project was to be built atop of Magma.

Closed; Although state-of-the-art algorithms are used inside Magma they are completely secret. The project is run by the university of Sydney, it is treated like a research project but the algorithms and code that is developed is used for commercial gain. In

the interests of science these should be available to mathematicians in the same way that every line of a traditional proof is.

Frustrating; Professor John Cannon who is the head of the project at the university of Sydney exerts a strong control over what goes into the project. Thus the Magma project is highly centralised and community-building is hindered. Stein personally had some code that he had developed for automated testing of Magma rejected by Cannon. Here we see parallels with Stallman's personal snubbing by Sproull.

Static language; Though the mathematical capabilities of Magma are superb the bridge language that is used is not as powerful as general purpose computer programming languages. An example given is the inability to create user-defined data types which is possible in any decent modern programming language. These are called type extensions. Generally speaking a language will provide the user with a mechanism for extending the type system of the language in a controlled fashion.

Onerous copy protection; Magma uses a copy protection scheme that ties the licensed instance of Magma to the MAC[5] address of the machine. Obviously this means if you want to run Magma in a virtual machine, or transfer it to a new machine, or if your network card breaks then the value of your copy of Magma suddenly becomes zero.

Other language deficiencies; Following on from the gripe about how static the language is the remark that many other features from modern programming languages are missing. Such as: no exception handling, no eval(), no namespaces, and so on.

Small developer community; Because the developer community of Magma is made up of researchers who are given access to the codebase and because there is no public mailing list the developer community at any one time remains small.

Subpar graphics handling; Magma has no graphical user interface, at least in the sense that Maple or Mathematica or MATLAB do but because Magma is so far ahead of these in terms of number theory it is quite painful that it is lacking in this area.

Inefficient bug tracking; There is no public bug tracker or list of reported bugs.

Bridge language is not compilable; In order to achieve very high speeds one needs to be able to ultimately either compile the language you talk to the machine in to machine code or have a just-in-time interpreter that does this for you on the fly. Because Magma does not do this the only way to achieve this in Magma is to write C code that interfaces with the Magma source code but because very few people get access to the source code this avenue is blocked to general Magma users.

After having looked at the reasons for going it alone let us get back to the responses to my questionnaire. In addition to the eight questions I asked the respondents to provide me with a name or pseudonym (or to leave this field blank if they wished anonymity) and to provide me with details such as: email address, website, nationality, native language, profession, present occupation, present position, employer. All turned out to be from academia: students, lecturers, professors – English and Spanish are their native languages. It was only after all these pieces of information were requested that I asked if there were Any other (personal) information that you feel might be of relevance. Without any further ado here are the results which have been subjected to scrutiny, both

[5]The unique numeric identifier (xx:yy:zz:ww:uu:vv) assigned to each local area network ethernet card be it wired or wireless.

analytic where I unpack any extra detail and synthetic where I connect the various dots.

5.2.2 Feedback to open source questionnaire

Speaking about their role and how long they have been involved in Sage Math –

- "Observer and critic."
- "My involvement is not very extensive. I went to a "Sage Days" in May 2010. I organized a Sage Days in May 2011, with about 30 participants. I am in the process of refereeing a couple of patches. One patch of mine is up on the Sage-Combinat patch server."
- "Been involved since 2006. Did significant work on notebook in its early development cycle, other involvement has been focused on performance/consistency."
- "I started using it for computer assisted proofs related to my research in early 2009. Now, I contribute with some code to the main base."
- "1 month"
- "I have been involved in Sage since late 2010. I have submitted several patches (which have been since committed) and bug reports. Basically, I am an intermittent developer."
- "I've been involved with Sage since around 2007. I've contributed lots of code to various parts of Sage, mentored new contributors and users, and I think I may be the de-facto head of sage notebook development now (but there is no such official position, it's just that I think I have the current master tree and no one else is doing much work on it)."

From this we can see that none of the respondents has more than five or six years experience on Sage, this is in contrast with the proprietary world where the respondents in that case have 18, 20, even 35 years of experience. We can surmise by the occupations which involve lecturers, professors (retired or otherwise) that these people have many years of mathematics and computer science experience but generally from an academic point of view; it is the youthfulness of the Sage project that accounts for the shortness of time. In creating a fully-rounded CAS system they are 25 to 30 project years in arrears. Because Sage is a piecing together of many sub-projects some of those sub-projects will have a considerably longer life-span to date than Sage itself. For your edification these[40] are the components used by Sage. The speaks to the organic nature of the project, its independently bottom-up approach and top-down – a sort of integration of parts, almost like a GNU/Linux distribution which nevertheless is steered towards Sage's own goals. If there were a study following this research it should delve into the components sending questionnaires to the mailing lists of the sub-projects.

Speaking about their background and what motivated them to become involved in the project –

- "I feel I know about computer algebra and programming languages generally. I learned about Sage and was impressed by its naïve approach to programming, mathematics, and management."
- "I like the idea of cooperative endeavours. I also think that the possibility to contribute to the growth of Sage makes the process of writing one's own code more interesting, and provides motivation to document it and polish it (which in the long run is probably useful for me even if I wouldn't do it without additional motivation). It also feels less of an imposition to ask more experienced developers for help if the code I'm writing might eventually make it into Sage."
- "Sometimes proprietary tools are good, but they may not be."
- "I am a mathematician (lecturer). Initially my motivation was using it to help me solve my problems, but later on I got interested on contributing to it."
- "Hobbiest. Using sage to build website for physics publications and experimental project development. The reason for my interest in sage is simple, I is supposed to be able to do almost everything Mathematica can do, & it's free. I have determined that sage is far from ready for prime time use as it's extremely unfriendly to most of us who are simple dabblers in this area. I feel that it may well succeed as a viable tool in a few years. It is currently under VERY heavy development & requires a lot to make it useful as a "tool for the masses". That is the area that I have chosen to work on. I'm integrating sage into a mediawiki installation that will allow real, productive, use as a tool for doing real science & publication quality material. At least that is my intention."
- "I am a high school student interested in mathematics and open source software. I knew Python previous to finding out about Sage, and wanted to find some good math libraries. I tried some others (NumPy, SymPy, etc.) but liked Sage's integrated approach better."
- "I have a PhD in math and a B.S. in computer science. I needed software to compute rank for matrices over finite fields for my PhD thesis, and was using Magma, but switched to Sage. I also needed software to compute various graph theory parameters, and Sage conveniently had lots of code to do that. Finally, I was asked by William if Sage could include a database I built, which pulled me a little more into the development community. Then I worked on graph theory, linear algebra, graphics, and other parts of Sage. The community was friendly and including, so I became a contributor. Sage Days and IRC were big factors in connecting with the community."

From this we can detect multiple motivations, some clear, some not so clear: there is cynicism, there is the scratching of your own itch that was talked about before, there is the art and craft of development. Someone speaks of the challenge of developing ones code out in the open, there is a feeling that this raises the bar and forces documentation to be written due to outside scrutiny and finally there is becoming interested in the project through having used it for some other purpose.

Speaking about the quality of Sage Math and the quality of open source mathematical tools in general and how this relates to proprietary tools –

- "Code written originally for Sage? Some of it is very poor. I am unaware of any high-quality code written specifically for Sage, but there may be some. Most open source math tools are unfinished and unmaintained."
- "I don't have a global opinion of "Sage Math". I do really like the fact that, with Sage, if there are issues, you will get an honest response, and might be able to participate in fixing a problem (should you uncover one)."
- "I think Sage is improving, but still has a long ways to go."
- "The quality is very good, through a bit inconsistent in different areas. In my particular areas of expertise the only proper proprietary competition is Magma. I don't know if I could call either system better than the other."
- "Sage is on the right track as it seems headed toward something that can perform as does Mathematica only free & open source. As far as I can tell all of these mathematic apps need to be overhauled so that they present a MUCH easier to us graphic interface. The real world does not any longer use command line inputs. Something that is more akin to the interface presented by apps such as Texmaker, or the graphics apps like blender would be even better. I don't have any experience with proprietary math tools other than Mathematica."
- "I am very pleased with the quality of Sage, although it is inconsistent sometimes due to it having to interface with many different software. The lower-level libraries (e.g.. NumPy) are also excellent."
- "This is a very broad question. In some cases, the quality is excellent; in some cases, very poor. As for Sage, I feel that the quality is good, but could (and does) improve. A lot of my work is in polishing things to have better quality."

A non uniform opinion on code quality. This shows that there is no metric as such for determining code quality. The trickiness of creating precise questionnaires is evident here. Code quality could refer to the, shall we say, beauty of the source code itself and it could refer to how free from defects the product is. The Sage project has quite a detailed developer guide[41] which and there are developer guidelines such that every patch (code fragment that implements a feature or fixes a bug) must be associated with a corresponding ticket and should be accompanied by unit tests. All user visible mathematical tasks must be accompanied by documentation, a system called doctests checks for this.

Speaking about proprietary software mathematical tools and their use thereof –

- "Occasionally I use Mathematica, which I find interesting but sometimes frustrating."

- "I have used Maple a bit, but not enough ever to get comfortable with it. I've also heard my friend who uses it more complaining about various issues (in particular the way new releases break his old code), which didn't encourage me to get better acquainted."
- "I use Mathematica occasionally to see if it can simplify integrals/sums I need for research."
- "I used some earlier in my career for lack of better choices, but I don't like not being able to understand how a certain algorithm works."
- "I have used Mathematica. Way too expensive."
- "I use Wolfram Alpha when I need a quick answer to a question. For any other mathematically intensive task I use Sage. I also use Mathematica on occasion. It has many features, but I dislike the language and prefer to use open-source software."
- "I use Mathematica occasionally. I think they are useful, and certainly pour a lot of resources into computational mathematics, which is great."

It is clear from the responses that there is but one tool that the people developing Sage feel they are up against. And that tool is Mathematica. With this is mind, and having tried comparing the two pieces of software I think that the documentation should provide tutorials for users migrating from Mathematica to Sage in order to ease the transition – large software vendors have consistently done this. I can think of Microsoft Word having created dedicated help sections for Wordperfect users[42].

Speaking about the GPL and free software licenses in general –

- "GPL, in my opinion, is a bad idea. "LGPL" is sometimes useful. I prefer a BSD style license. Yes. I just got a note from someone that Maxima could not be made available in the Mac app store because GPL made it impossible."
- "I don't have a finely honed technical knowledge of the GPL. It seems to work adequately, and it seems to be (mostly) standard – those are the main important things."
- "I'd prefer to use a freer license than the GPL."
- "I don't feel entitled to talk about the GPL or other free software licenses. I am just glad we have a way of making our research/code easily accessible to everybody. I have a favorable attitude. But if you are asking about a dispute whether it is better to use GPL, BSD, Apache or MIT-type open licenses then I am not sure on what to say as I don't know the specifics of GPL (or any other of these licenses) well enough. There are half a dozen discussions on sage-devel about compatibilities of different licenses and endlessly arguing whether we can or cannot ship/distribute/link against some library released in (pick your favorite license). And honestly all that discussion bores me to death, I just want to do my math."

- "Excellent!!"
- "I am very supportive of free software licenses; I license all of my code under open-source."
- "I like the GPL, and also license things as (modified) BSD or MIT. I think they are great licenses in many situations, but are not the best thing in some situations."

The responses here show that a project-wide policy on which license most benefits Sage and for what reasons could be developed. This would not be a lot of work and would clearly help the Sage project. As the transparent nature of the project is a key differentiator (regardless of tool capability) the choice of license is a reflection of how the project as a whole view the nature of that transparency. In the case of Linux the benevolent dictator for life Mr. Torvalds has stated why he likes the GPLv2 and why even in the face of the organisation behind the GPLv2 superseding it with the GPLv3 he is going to stick with the older version due to how he views the changes the FSF has made.

Speaking about their involvement in Sage Math in the context of the broad arc of scientific research and development (amended responses) –

- "No more than other programs. I'd like to know that the program that gives me access to my bank account is keeping my password secure, so why not demand that it be open source? In reality, open source code will be read by almost no-one. Just as in reality proofs in math journals are read by almost no-one."
- "So I guess the short answer is: no. Transparency of experiment wasn't a big motivator for me, because I mainly adhere to the model whereby what you publish are your proofs, not your experiments (be they computer-aided or not). It's not that I think this is the only model for mathematics, but it is the model that I am used to, and I think there is social pressure to adhere to this model."
- "This question makes little sense. In the broad arc of research and development, I'm a mathematician and Sage user. This leads me to submit the occasional bug report; very rarely to contributing to Sage."
- "In an ideal world yes, but at the end of the day I (and most of my colleagues) will rather use something that works than something with a nice license. My personal reasons to get involved with sage came from the need of doing some specific calculations that could not be done in any existing software. Sage provided me a nice framework so that I wouldn't need to code everything from scratch. But I wouldn't have got involved with sage just because of its transparency if it hadn't been out of need."
- "As far as 'experimental' results devised from proprietary software, well they would seem to be proprietary and should be judged by the peer community as such. In my opinion: non-verifiable results that claim to be 'proven' by software written by the experimenters, whereby 'the experimentors' do not provide the code involved, should be adjudged as 'non-verifiable' unless the producing code is also verifiable.

Does that mean that they must reveal the code? NO! It just means that their credibility should be called into question. The current goings on at the Cern facility regarding 'particles moving faster than the speed of light' is exactly this scenario. Though their claims may be true, everyone in physics is waiting for the other shoe to drop. Why? because they are not revealing all their processes."

- "Not very significant; only relatively minor patches for now. However, I will be submitting a patch for physical constants, which could be useful for educational purposes."

- "I see a trend in scientific research and development towards open software and materials (like open free textbooks, etc.). I see my involvement in Sage as pushing that trend a little further, opening up opportunities for everyone to learn and grow."

These responses originally showed that almost nobody could see that what I was driving at here is the case for open computer programs. I have amended these answers with those ones I received in the follow-up. Indeed, it was the lack of feedback on this point, a point I see as crucial to the overall scope of this thesis, that motivated me to solicit clarification in the form of a couple of follow-up questions. As you can see the epistemological concerns that I raise in subsection (4.4.4) are not an issue for users and developers of mathematical software – pragmatic concerns are the overriding factors.

Speaking about further views and opinions they would like to share with the world regarding all of the above –

Nobody had anything further to contribute.

Speaking about whether there is anything that I am not asking that you think I should be asking –

Again the was little in the way of a response except for the helpful, "Sure, you could ask, «What do you think math software should look like in the future?» and, «How might we get from here to there?»" This I think is very forward looking, and I included it in my follow-up questions.

5.2.3 Follow-up questions to open source questionnaire

Speaking about what they think math software should look like in the future. How might we get from here to there? –

- "Usually people who consider spending a large part of their time writing software also have to find a way to pay for food, clothing, shelter. Now it could be that there are enough people who will write programs, free, but will they be the best equipped to do a good job? Maybe some academics, whose pay is somehow related to research grants, papers, etc. As you may realize, the VAST majority of free software is buggy, ill-designed junk. There are a few exceptions, but they

are rare. There are what 500,000 "apps" for iphone? How many programs on sourceforge? I think it is fair to expect that most open source math programs (i.e. user-contributed) will be less than fully professional. (i.e. debugged, tested, documented, efficient, robust, etc.)"

- "As regards what mathematics software should look like in the future, I don't have a strong opinion. I think part of what makes Sage a lively locus of activity (which is part of its value, as well as being part of what makes it fun) is that there is a lot of room for people to jump in and help, and I'm curious whether that will continue indefinitely. At some point, perhaps the sense that there are simple and useful additions waiting to be made will diminish – so paradoxically, improvements to the software may lead to its becoming less useful as a collaborative project."
- "It should be as easy to use as a pencil & paper, for individual projects; or a chalk board for collaborative projects. It should be completely integrated into graphics production tools so that "seeing" the results is as simple as hitting the graphing function on a spreadsheet.

  Far better input tools that work in a ubiquitous manner; touch screens that use standard math notation (or other standard notation; chemical, electrical, etc.) for all areas of science. Perhaps grid mapped 'work pads' with a simple drag & drop process for standard notation, so that we can put on the display exactly what we would put on paper.... We are currently 'making the wagon pull the horse': i.e. we users have to 'learn' a whole programming language just to be able to get the computer to print/display math symbols. Learning advanced math is though enough. Using it purposefully is the challenge. Spending a lot of extra effort on getting a computer to display what we are saying is just stupid. They are tools and should be useful."

What can be seen here is that there is a desire among developers and user to see the computer display mimic the domain specific notation already acquired on the road to mastering some discipline. It is seen that the built-in language of the mathematical software package is yet another thing to learn, an obstacle that gets in the way of doing useful work. Some express hope that this barrier will be overcome, some are far less optimistic.

5.3 The proprietary world

5.3.1 Lead-up to proprietary questionnaire

As part of this thesis I was able to contact people who are involved with commercial software. This was enabled by one of my supervisors, the very helpful and supportive Kristóf Fenyvesi. Over the course of 2 days, on the 14th and 15th of December I received ten responses. One of these responses was too casual, uninformative and expletive-ridden to be taken seriously. So nine in total then. This is slightly more than the survey I distributed to the Sage community to which, as I say, I received seven responses in total.

As I said before, but it is worth reiterating it:

The public surface of a commercial piece of software is limited to the product itself; the semi-faceless corporation behind the product; the marketing, public relations, and advertising spend of the corporation; the technical support; the trade-show booths; and nowadays the corporate website which can serve as a technical point of contact or merely as an electronic brochure.

Corporations extract wealth from software by treating, and this may seem obvious, the expression of the software as a product – that is to say commercial property. In the era of shrink-wrapped software this is enabled by the dual nature of software itself. A deliberate withholding of the source is performed in order to safeguard the binary product, the idea being that this corporation has a monopoly on the source and corresponding build system, the methodology of turning that source into a binary blob. This is why the term proprietary software is apt, and not the term commercial. It is plain as day that inverse of commercial is non-commercial, and whereas many small FOSS projects have very little turnover, most FOSS projects have a commercial aspect to them even if they are non-profits. To think other wise is to elide the commercial act with the capitalistic act.

Many commercial entities structure their businesses in such a way as to withhold a piece of information. One can think of pharmaceuticals, one can think of brewing, one can think of many areas. In the transition to the digital world the rules and regulations that bound the material world were slowly reworked for the digital world. Thus we have copyright previously applied to text now being applied to source code, we have the rubric intellectual property, we have patents now applying to intangible objects. The effect is to create what some people[citation] view as an artificial scarcity.

The public surface of a FOSS project differs considerably from that of a commercial piece of software. Whereas by definition the activity on a commercial project takes place behind closed doors the activity in a FOSS project can exhibit varying levels of openness and transparency. There may be chat rooms where developers and/or users hang out. There may be public mailing lists. There may be a wiki or a blog, there may be a bug and feature tracker. In fact, most FOSS projects have all these avenues of interaction and more. Producers of proprietary software have these pathways of interaction as well but they are internal to the corporation that owns the piece of proprietary software. FOSS projects do not need these avenues but to not have them is sort of anathema to the ideology that motivates people to engage in this mode of software development.

This is all to say that the public persona of a corporation is brochure-like, managed – all image, never mere process. Their impact on the world beyond their product are what are called externalities. This refers to how they affect there environment beyond the insertion of their wares into the marketplace. These externalities could be in the area of pollution, politics, harm to the biosphere, harm to the food-chain. This can bring these entities in contact with the legal system which becomes one record of a corporations existence.

In the end it turns out that I was very fortunate that the respondents to my proprietary questionnaire were employees of Wolfram Research or used Wolfram Research's premier product Mathematica. In the light of the preceding paragraph it could be said that to know Mathematica is to know Wolfram Research. Most of the information I provide here I have culled, unless otherwise stated, from the official corporate website

and a certain online encyclopædia which shall remain nameless. This information is easily accessible, non-authoritative but nevertheless for the purposes of this thesis entirely adequate.

The private company was founded by its namesake, physics doctorate and author, Stephen Wolfram. It was founded in 1987 to commercialise a piece of software that Wolfram had been developing for a number of years prior. This work was born out of [citation] and this became Mathematica. Today the corporation is a multi-national, spanning at least 4 countries: the US, UK, Japan, France – it has four hundred plus employees. A regular single-user license for Mathematica used in a commercial environment costs $2495. A student licenses cost $140. Products like Microsoft Office are similarly discounted. One could never sell a car for $24,950 for use in commercial environment but sell students the same one for $1,400. It is unthinkable. This amazing the difference in cost shows that the cost of software is not in the bits, at all.

With all that out of the way here then are the ten questions, two more questions than the FOSS group, were as follows:

- A :- How long have you been involved in mathematical software?
- B :- What are the projects you are involved in and what are your main roles?
- C :- What is your background and what motivated you to become involved mathematical software?
- D :- What is your opinion of free and open source software (FOSS) mathematical tools? Do you use any?
- E :- Do you have any opinion on the advantages or disadvantages of FOSS mathematical tools and how they relate to proprietary tools?
- F :- What is your opinion about copyright, software patents, the GPL and free software licenses in general?
- G :- In the broad arc of scientific research and development, where do you see the place of mathematical software?
- H :- In the development of the arts, where do you see the place of mathematical software?
- J :- Are there any other views and opinions you would like to share with the world regarding these topics: consider this your soapbox.
- K :- Is there anything I am not asking that you think I should be asking?

As before, in addition to the series of questions I asked the respondents to provide me with a name or pseudonym (or to leave this field blank if they wished anonymity) and to provide me with details such as: email address, website, nationality, native language, profession, present occupation, present position, employer. It was only after all these pieces of information were requested that I asked if there was Any other (personal) information that you feel might be of relevance. In contrast to the Sage project the target

group was not well defined and consisted only of people I could reach through my supervisor. However, this is not a major problem as I initially had only hoped to survey developers of open source mathematical software and so one could consider this second group of respondents as bonus data. Here is a run down of the nine respondents that I felt returned with valid and interesting talking points.

Taking the questions in order.

5.3.2 Feedback to proprietary questionnaire

Speaking about the length of involvement in and around mathematical software –

Bar two who had less than 5 years direct involvement in mathematical software. All respondents had multiple decades of involvement, we are talking about claims of 18, 20, 30, and 35 years. This means that these people have seen how this software has evolved with the introduction of the microcomputer and the internet.

Speaking about the projects and the respondents' main roles –

- educator: "CAD tools for circuit layout CAD tools for conceptual phase of architecture CAD tools for machining mechanical parts CAD tools for geometrical sculpture Mathematical visualization models"
- user/engineer: "Mathematical graphics and animation with the program Wolfram mathematica"
- educator: "I am involved in teaching regularly these programs:
  DERIVE, PHASER, use of EXCEL to do mathematical calculations, recently Mathematica for students of mathematics, physics, engineering, economics, cognitive science etc. I also use them in applications and research, and also in teaching subjects not directly connected with them (e.g. differential equations)."
- developer: "user interface development for Mathematica, primarily interactive graphics"
- data curator: "oil rig data - catalog data keyboard data - catalog data quasicrystals - computing aperiodic lattices"
- developer: "Implementation and/or debugging of symbolic computation algorithms. This includes functionality that is to greater or lesser extents approximate numeric, exact numeric, or (almost) purely symbolic in nature. Areas include equation solving, linear algebra, optimization, number theory related functions, "classical" computer algebra, and more."
- architect: "SMP: creator, architect, company co-founder Mathematica: creator, architect and company founder & CEO Wolfram|Alpha: creator, architect and company founder & CEO"
- business person: "Creating math manipulatives and books for K-12 education and creating mathematical art. I handle all roles related to these activities."

- QA: "I use files that other people create using my company's proprietary math software product. I use these files to process data, to automatically generate unit tests, to download and manipulate source code, and to manually optimize other aspects of the web product I work on. I do not actually develop software, but I understand some code and have made changes to legacy math software code in the source files to fix minor issues."

In contrast to Sage these people live and work outside the walls of academia. For a lot of them how they put bread on the table is through the practice of selling crafted software. This difference in perspective comes into play throughout the replies. A good range of experience and positions is represented which bolsters the usefulness of the data.

Speaking about the respondents' backgrounds and their motivation for becoming involved in mathematical software –

- science/academia: "Math and Physics. – A strong love for geometry."
- science/industry: "Educated as a mechanical engineer. My major interests is to depict natural phenomena, including mathematical theorems and operation of machines."
- mathematics/academia: "I am a mathematician by training, started to teach computer science in the nineties, and realized how good a tool Derive and - later - Mathematica is."
- comp sci?/industry: "I have always been interested in interactive graphics and worked in some field related to the same. I used Mathematica in my dissertation work and thoroughly enjoyed exploring with it. When the opportunity arose to work for Wolfram Research, I saw it as a chance to contribute to a quality piece of software that I loved using myself. I love mathematics, but it has always been important to me to connect the math to graphical results. Mathematica has given me that opportunity."
- mathematics/industry: "lots of programming at a young age, exposure to specialized mathematics, individual research Trained as mathematician in grad school. Vagaries of job market led me to this line of work."
- science/industry: "Physics PhD. Wanted to have tools for my own and other peoples' science and technology"
- science/academia: "My training and early professional activities were in semiconductor and device research. I have used mathematical software as required to complete professional tasks and to design products for K-12 education. I have also used mathematical software to aid in the design of mathematical art and to explore recreational mathematics."
- science/industry: "My background is in Biology and Physical Anthropology, in which I have a Bachelor's (undergraduate) degree. I worked as an undergraduate assistant in a Human Biomechanics lab and a Balance and Gait lab. The scientific lab experience and the advanced computer proficiency that was required

for my research in these labs made me a desirable candidate for my current math software company, who were looking for people from a variety of scientific backgrounds who understood programming and could help optimize software and manage bug fixes. They hired me before I even graduated college, and I was flattered. I finished my degree and have been here ever since."

Most have backgrounds in science or mathematics, only one has a computer science background. This implies that mathematical tools are far too specialised for ordinary software engineers in the main. Taken with the results from the other questionnaire where all the participants are in mathematics departments in academia I think allows us to make the claim that the engineers of mathematical software tools are drawn from a smaller pool than is normal for software projects.

Speaking about their opinion of FOSS mathematical tools and whether they use any –

- positive/does not use: "This is a good thing. But I have not personally used it."
- none/does not use: "I do not use free software, because I am content with Wolfram Mathematica."
- positive/does not use: "I know something about Sage, this will be a threatening alternative to Mathematica - in ten or twenty years from now."
- negative/does not use: "Very few. Open source is a great way to harness collective programming power. But it's not the be-all, end-all answer to the world's software needs. Some projects require the control and thoughtfulness that open source can't provide. One of the strengths of Mathematica is that since *all* of the design has gone through one brain—Stephen Wolfram's—it all works together very smoothly. A system as complex as Mathematica would be a mess if it were designed open-source."
- none/does not use: "Some interest but no experience"
- negative/does not use: "I do not use any. My opinion is that some are good, though few are well maintained over the long haul. But stable functionality tends to be well regarded in some of these tools. The hype about "open source = more reliable" is garbage. I refer, by and large, to the paper available at the following[44] links[43], and to various talking points that have grown around this work."
- none/does use: "We use quite a few such libraries in our technology stack"
- negative/does not use: "Most FOSS tools have limited capabilities, and I have only used them on occasion."
- positive/does not use: "I do not use any open source mathematical tools because I do not use any mathematical software in my daily life other than at work. I am a proud supporter of open source and free software, however, and I have heard very good things about Sage."

It would seem that people in industry, at least this small sample, have an overall positive view on FOSS mathematical tools but do not appear to use any personally. Some FOSS tools related to mathematics are used in the technology stack of proprietary products, presumably such tools carry a BSD license or something similar.

Speaking about the advantages or disadvantages of FOSS mathematical tools and how they relate to proprietary tools –

- none: "Not enough knowledge to comment."
- none: "n/a"
- no specific: "I think there is a level which is not too high required by people who are say engineers (or economists, chemists etc. if you like). People who study 3-4 semesters at the university and are not specialized in a mathematically or computationally intensive area (such as e.g quantum chemistry). The needs of these people are well fulfilled even today by Maple, Mathematica, or even Derive or Sage.
  People who are interested in the use of programs in mathematical (or mathematically intensive) research need the best programs. The reason I like Mma is that it has been planned more than 20 years ago and not is not a result of patchwork. Another reason is that you can use any of the programming paradigms, even the best fitting one to your problem.
  I do not care bugs, they will exist for ever, but a large group of users will eliminate those."
- "Quality is often surprisingly good, but in my experience, usually slower than proprietary tools and not as nicely designed."
- none: "No opinion"
- "Advantages: Easier to alter if one finds bugs and has the time, inclination, and expertise to fix them. Can share work with colleagues who might not have it, provided the software is readily available and easy to install. Easier to contribute to, if one (or one's students) is/are so inclined. Disadvantages: Maintenance is generally not too great, especially regarding legacy bugs. In this respect, one gets what one pays for. Interfaces tend not to be too good. Overall design is often weak due to patching together work from various projects, disparate contributors, or in some cases (e.g. Sage) even entirely different programs. It is a bit harder to request new functionality and actually obtain it, unless said functionality happens to be something of interest to the current developers. That is to say, the "market place" (howsoever construed) has relatively less influence here. There is a greater chance of project forking, thus diminishing to some extent the chances for decent code maintenance."
- "They can be good in providing well-defined lumps of functionality. They do not usually provide polished coherent user experiences, or innovate in the area of user-level functionality."

- "The primary advantage to FOSS tools is cost. Some proprietary software is too expensive to justify the cost for the my applications."
- "The advantages are numerous: the tools are free and easily attainable via internet download. The community of users can also be the community of developers, meaning that the development process can focus on what users actually need and want as opposed to what some marketing team has decided will generate the most publicity and revenue. The main disadvantage I have heard about Sage in particular is that it is unstable and often crashes. If open source software crashes, no one is going to lose her job because of it – there is less pressure on the developers to fix bugs since no one's livelihood is affected by malfunctions or breakage."

We have here the usual role call of observations with respect to FOSS: cheap, inferior quality to proprietary software, easier to fix in theory but not in practice, danger of forking, good as glue code but not as good at the user interface level, frequently buggy.

Speaking about copyright, software patents, the GPL and free software licenses in general –

- "Free software licenses are great. The patent system is broken."
- "n/a"
- "Maybe because I live in a country which does not belong to the richest ones :)? Maybe because I had two earlier students this week also who came for advice and I didn't ask them to go to the cashier first - and I myself often receive support from other people in this absolutely obsolete way."
- "People have a right to own the results of their programming efforts... or to give them away freely."
- "In a perfect world, everything would be free, but in a practical world it may be better to create software using commerce as a means for progress."
- "Copyright of source code is fine, provided it does not serve as a hindrance to independent development of substantially similar algorithms (I believe copyrights do not cause such hindrances). Software patents are another matter entirely. They seem to be, generally, a hindrance to the (both free and commercial market place. Suffice it to say, these issues are going through various legal systems with some immediacy. One hopes the laws will catch up with technology... The GPL is a perversity. The claim is that it helps to make software universally available. In actual fact it serves as a major barrier to what the NSF terms "transfer of technology". Generally this involves going from seeded research funding to mature projects within the market place. More often than not this latter step is in the commercial sector, as that is where there is a greater likelihood of adoption, maintenance, and furtherance of (many, though by no means all) useful software. Generally speaking, free software that is available both for commercial and non-commercial usage is better positioned to be deployed, thus serving the needs of more users. Software licensed as BSD is a good example of such; it is useful both in free and commercial domains."

- no response
- "I understand the need for people and companies that develop software to charge for it in order to pay for the hours that go into writing it. Copyrights and patents are necessary instruments to protect intellectual property."
- "I support the EFF and the FSF and have been using Linux at home since I was a teenager. I favor models similar to e.g. RedHat: open source software that is still able to generate revenue and be the main source of people's employment. I understand that this is difficult, but I wish it could be the reality for more software initiatives."

As the reader can see the range of opinions on display even in such a small sample size is remarkable. Cutting through the extraneous garb and we see emotive language, we see ideological language. In this way we can show that mere outward cultural differences, that is to say stylistically superficial ones could not evoke such polarity. The culture has to be somewhat grounded in ideology. It is the nature of ideology that within an ideological framework its competing systems are mutually hostile – the same cannot be said for competing cuisine, hair styles, or dance moves[6] . The reader can clearly see the distinction I referred to earlier: positive and negative freedoms, freedom from something versus the freedom to something – hence the rift between BSD-style licenses and GPL-style. Software patents are viewed, when an opinion is expressed, as a hindrance rather than a help to the industry. The ambiguity of the term free in English is still an issue. As one would expect with respondents from the commercial sector, commercial scenarios is seen as pragmatically dictating reality.

Speaking about mathematical software and its place in relation to the broad arc of scientific research and development –

- "At the center!"
- "I share this view: "Four centuries ago, telescopes were turned to the sky for the first time – and what they saw ultimately launched much of modern science. Over the past twenty years I have begun to explore a new universe – the computational universe – made visible not by telescopes but by computers" Stephen Wolfram"
- "Where mathematics is used, math software should also be used"
- "There's no getting around the fact that many scientific problems are mathematical, and require mathematical software to attack. The challenge for developers is to move closer and closer to the goal of letting the user specify what solution is required, and letting the software figure out the best way to get to a solution. Too much software today requires you to have intimate technical knowledge to make any progress.

[6] How does this relate to the tribal aspects exhibited within cultures, for example football rivalry and the like? Oddly enough this could be a sort of striving for group identity within a sea of undifferentiated sameness – at least ideologies provide reasons, tribalism provides only one reason – this place rather the other place.

I agree [...] that working code should be supplied with scientific articles whenever possible. (Ironically, I find referees complaining when I put actual code in my submissions. Since I believe it belongs with the work, I have learned to isolate it in an appendix.)
My opinion is that if research is funded by one's home institution, then it gets to make the rules regarding what is proprietary. If it is from taxpayer funds, I'd rather see guidelines that are in the direction of easy licensing and dissemination of actual code (I have indeed raised this issue with one such funding agency). I do not particularly like the thought that submissions unaccompanied by code (e.g. because it is proprietary) might face discrimination in terms of acceptance for publication. [...] My reasoning is this. I see quite a few articles in places like Communications of the ACM, written by researchers from industry (Yahoo, Microsoft, Google, and elsewhere), that are interesting but touch on proprietary work. I vastly prefer having such articles published, sans code, to not having them at all."

- "it belongs everywhere, but not all the time."
- "Numeric software is pervasive. Likewise statistical software. Both are likely to grow in importance as "big data" becomes more integral to science and social sciences. Symbolic computation is, at present, more of a niche. It is starting to see wider usage in the sciences, technology, engineering, etc. But it is by no means used to the same extent as stats and other numerical software. Of course this could change over time, and indeed there has been a slow but steady rise in usage of exact and/or symbolic methods."
- no response
- "Much research and development is experimental in nature and doesn't require sophisticated math software. However, much of it does require modeling that needs sophisticated tools. In most of these cases, at least in the USA, a company or research institution is behind the work, and the funds are available to purchase software."
- "Math is the foundation of every scientific discipline. I have never worked on a research project, even in anthropology (which people erroneously assume is always a "soft" science – not so with physical anthro!) that did not make extensive use of mathematical software. Once the study is proposed and the funding secured, the data is gathered. Once the data is gathered, the data is processed with math software. This processing constitutes sometimes 70-80% of the time spent on an entire research project. Math software sits at the very core of modern research."

Without a doubt, the ability to perceive the role that mathematics and by extension mathematical software plays in the shaping of the world but at the same time not to grasp the fact that a degree of transparency is necessary for the full benefits of this shaping to be accrued by everyone is a function of ideological and cultural bias.

Speaking about mathematical software and its place in relation to the development of the arts –

- "One new powerful and flexible tool to complement other means of creating art."
- "There is a category called Mathematical Art, but I think nice pictures themselves are not art."
- "Animation, film making, perhaps also painting, (textile) design"
- "Now that computing has made mathematics accessible and employable, there are enormous possibilities in the arts. In architecture, for example, the availability of mathematical computing has opened up new formal worlds that were previously inaccessible. There has been an explosion of new architectural forms that will only accelerate in the future."
- "It's more about the artist than the computer. What is the place of acrylic paint in the development of the arts?"
- "I do not have a great feel for this. I can only say that I have seen fantastic creations in graphics and, to a lesser extent, sculpture, that were created primarily using mathematical software."
- no response
- "In the visual arts, the area I'm most familiar with, the use of math software allows the creation of new and unique designs and forms. For some two-dimensional designs, math software is needed, but a lot can be done with non-math-specific software such as Illustrator and PhotoShop. For three-dimensional designs, math-specific software is often essential."
- "There is little general interest in mathematical art, although I personally find it fascinating what beautiful creations people like Michael Trott and Igor Bakshee have been able to assemble using Mathematica. In the art world, though, I don't get the sense that this medium is respected. Perhaps it is because the medium of mathematical and computer based art is too new still to the artistic community. I wish to share the observation that of the people who genuinely do appreciate and admire math software based art, the vast majority are mathematicians and scientists rather than artists."

The analogy with a previous medium doesn't hold. As I have argued for elsewhere – I do not see the computer as a medium, rather due to the nature of the digital I see it as a medium of mediums – similarly the internet is not just a network, what makes the internet different from what preceded it is that it is a network of networks. This is a recursiveness is a key feature of cybernetic systems. No other man-made medium heretofore has exhibited this self-referential nature so fluidly as does the digital. Otherwise these opinions speak to the ongoing slow-burning revolution in art, design, and architecture. The impact of computers and maths and engineering software on architecture would be a thesis unto itself. It is interesting that the idea of the authenticity of art is questioned because of this mechanical intervention, we should view this as an opportunity rather than with caution.

Speaking about whether they had any views and opinions that they would like to share with the world regarding any of the above –

- "Good computer programs can be an amplifier of one's creativity."
- "In spite of the many benefits of using computers, disassociate yourself as much as possible from using them."
- "At the moment I would only compare Mathematica with Sage, all the other programs are much less relevant. (Except very specialized fields, like GAP for group theorists.) WolphramlAlpha and the plain language input to Mathematica is also a great step to the Holy Grail of artificial intelligence. I appreciate google's Graph function."
- "Archimedean principle: Math and Science tools should not be used for War computations. If people want to participate in war, then they need to develop their own tools to do so. Nor should we teach them how to develop these tools."
- "Regarding the use of math software in art, I find it incredibly disheartening that most artistic types close their minds to the mathematical and the scientific. They see science and art as this irreconcilable dichotomy, whereas in reality, does art not exist to exemplify and express beauty? And what could be more beautiful than the idea of superstrings, or the plot of a system of numbers that represent the ratios of petal sizes in a flower, or a fractal repeating infinitely, or the very concept of infinity itself? Even a simple column of values in a spreadsheet is beautiful if those values are based on scientific observation. Those numbers represent our human need to understand our place in the universe, to know more of who and what we are, and I can think of nothing more lovely in the world."

Apart from the survey-oriented response (the contextual response shall we say) these replies speak to markedly different aspects of the human spirit: war, creativity, beauty, dehumanisation. One could say that a general point that I have been attempting to get across heavy-handedly in this thesis is made here for me – the fact that the digital heralds not just a boost in speed but in expressivity. The rancorous two cultures divide in contemporary academia (see for instance C. P. Snow's 1959 The Two Cultures Rede Lecture turned essay[Snow, 1993]) echoes the tug-of-war dichotomy expressed here between science and art. The divide is reflected in the new priesthood role of high technology and big budget science. The dehumanisation angle has been tackled adequately by theoreticians in digital culture, most notably N. Katherine Hayles foretelling the coming epoch of posthumanity[Hayles, 1999] in her influential book, How We Became Posthuman, from the turn of the millennium. Witness also Sherry Turkle and her recent text[Turkle, 2011], as of the time of writing, Alone Together.

Speaking about whether there is anything I am not asking that they think I should be asking –

As in the case of the Sage questionnaire only one person responded to this question. However, rather than giving me suggestions - this time the person thanked me, in their

words, "I am pleased that you have asked anything at all. Math software is usually seen as an ugly means to a tedious end – this is the first time I have encountered interest in it as a unique entity worthy of study and consideration. Thank you." In the dry and tedious world of academic research it is wonderful to get positive feedback and encouragement. So, no dear datum, thank you.

5.3.3 Follow-up questions to proprietary questionnaire

Speaking about what they think math software should look like in the future. How might we get from here to there? –

- "I consider the free form input of Mma and of wolframalpha useful for many users, still I myself would prefer a form as closely to mathematics as possible. I am not interested in fancy solutions like vocal input etc."

- "(1) It will have a better notion of what is feasible for computations, how long they might take, and some ability to let users know in advance. (2) More natural language (written), handwriting, and speech recognition input capabilities. Note that some amount of this exists already. (3) More deployment in ways that users never see, e.g. controlling devices (homes, cars, robots, etc.), use within AI (which in turn is seeing more usage in controlling things), and in general seeing more use in automated environments. Sorry for the run-on sentence. And again, note that this already is happening, though I expect the scale to increase perhaps vastly. (4) Development of more math libraries, better maintained, more easily extensible, etc. This may be wishful thinking on my part. Probably at a minimum will require more public/private partnering. Reason being, libraries developed for academic purposes often get initial funding but then do not get well maintained; there is usually far less incentive for the original authors to put resources there.
  How do we get there? We're already headed in most of these directions, though at present not so much item 1 (it's a difficult task in general, and there is not much in the way of market or other pressure to invest resources in it). I see much of this as simultaneous pushing from research labs (university, government, and industrial), and pulling from consumer demand. Or is it pulling from the former and pushing from the latter? I can never tell."

- "I don't know that I use math software enough to have really thought a lot about how it should be different in the future. I can comment that I would do more with three-dimensional structures and also more animation if it were easier to use software for those tasks. I've always got several things that I want to do with my time, and I tend not to pursue things in those categories because of the time it would take both to get up to speed with the software tools I would need and the time it would take to create the sort of things I want to create once I was up to speed. So maybe that's saying I would like to see tools that are more intuitive and easier to use. When using Mathematica, for example, you're basically writing code, which is a grind for people like me that haven't done a lot of programming."

Again, in keeping with the answers from earlier we have the desire for the man-machine interface to be made more natural, to take on the form of mathematics itself as it is done with pen and paper. It is arguable that there is more optimism on display here from those that chose to reply to the follow-up. It was disappointing and hurt the solidity of the data that while a number of people agreed to answer a couple of more questions (out of this group) only three did.

5.4 Results

I have been scrutinising the individual responses (taking an analytic approach) en route so now is the time to draw some general conclusions from the responses taken as a whole (synthetic approach).

The impression one gets is that engineers, lecturers and scientists generally take a pragmatic approach to software – this means that they will reach for the tool that does the best job. There are those who are obviously passionate about open-source mathematical tools but given the size of the entire ecosystem (Linux, BSD, LibreOffice, Firefox, most recently and notably Android, ...) my suspicions have been confirmed that the entire open-source mathematical tools sector is underdeveloped vis-à-vis the its proprietary counterpart and lags the ecosystem as a whole.

Regarding overlap. It would appear there are very few exclusive users of either proprietary or free software. It appears that most contributors to the Sage project try to push things forward so that there reliance on proprietary software is diminished. All of the artists use proprietary tools to the extent that one artist even develops proprietary custom software – I am not sure how much of an issue this is for the artistic community. Certainly epistemological concerns are not as important.

Almost no respondent from either camp has thought of the big picture question about the use of mathematical tools deeply connected with the arc of scientific progress, in fact most misunderstood my question.

5.5 What I learnt about posing surveys

Ask for only one piece of information in each question. Do ask what formal education the person has received, if any. When asking about software ask about the names of the individual software packages used, not vague markers like projects or categories. Do ask whether or not it is okay to contact the person for follow-up or clarifying information if they have left an email address. Do state that the research is serious and that casual answers are not appreciated. If you ask people to volunteer any further personal information they will not.

6 Summary of results and conclusion

The final part of the thesis is to hand. Now is the time to recap. It is an unenviable task to disentangle mathematics from its surroundings, from the world around it so to speak. The same observation holds true when we speak of the digital world. Mathematics is inexorably interwoven with the way of human life and its cybernetic subset the digital human life. But like nearly every story concerning the artefacts of contemporary life mathematics (be it the practice, education or application) has itself in turn been altered by the impact of the digital world acting on it. This thesis has focused on some of those alterations.

A word about tools: the traditional tools of the manipulator of mathematics has been primarily the pen and paper.There have been mechanical aids such as the compass and T-square, the set square, the tally stick, the abacus, the ruler and the slide-rule. These have been eclipsed by electronic computation. The manipulators of mathematics, I am thinking here of educators, researchers (mathematicians, those who deal solely in mathematics), engineers, actuaries, statisticians, artists, architects, these multiple and diverse fields along with the ordinary and everyday user of electronic calculators have benefited from the computerisation of mathematics.

Mathematical tools as they exist in a digital world is mathematical software. With that in mind this thesis has looked at mathematical software from a number of orthodox angles.

I have assembled the various parts of this thesis to give some unifying structure to the totality of this thesis. The word mathematics has a double meaning rendering its use to be ambiguous at times. (In English at least) it refers to a shared formal language and a discipline. Contrast this with other similar words that refer purely to a discipline, like biology for instance. Unless this distinction is kept in mind some confusion shall arise. It must be stressed that in this thesis I refer to mathematics the discipline, which is why I speak of practice, education and application.

In section (3) I have tried to touch on how the practice and application of mathematics has been affected by mathematics having gone digital by taking a look at that intersection, at mathematical software that is to say. With respect to the practice I have delineated on the one hand the growing dependence on core mathematics on computational proofs and on the other of digital imaging being an immediate aid in the visual exploration of mathematical landscapes. Highlighted also was the coalescence of complex systems around packages called CAS, computer algebra systems. These packages would be taken up again in section (5) in more detail. With respect to application a journey was taken in the world of mathematical art and in doing so I attempted to place this field along an organic continuum of sorts – this is now the exploration of visual space itself with the aid of mathematics.

In section (4) the focus was turned on a subject that has a remarkably sparse corpus in the field of digital humanities (Wendy Chun excepted). I have given a very clinical legal-historical account of copyright and then showed how the issues of the integrity and paternity of the work and how the issues surrounding the commons has taken on a new life in the realm of software development. Personal and private interests conflict with the social requirement of some sort of digital commons. Moving from the historical I attacked the problem space from a number of angles, these angles as I have mentioned

a few times now are: epistemological, legal, ethical/pragmatic, economic.

Building on strands of study from both section (3) and (4) I was finally able to situate the questionnaires in the theoretical context of the previous sections. These questionnaires, directed at both open-source and proprietary actors, along with an introduction and critique of each of the type of software used by these actors rounded out the meat of the thesis. The approach I took was to interweave the actual responses themselves with a running commentary followed by a high-level summary.

6.1 Brief finishing remark

Some of the goals of this thesis have been met, others have not. In the first case an adequate picture of the landscape of mathematical software has been given. In the second though the various topics are linked by mathematical software, and though the discipline of digital culture anchors all that goes on in these pages it would perhaps been wiser to choose a narrower subject-matter and to pursue that in depth.

I believe that I have shown that it is possible to ground the study of digital culture and the digital humanities in a mode of learning that is more focused on the stuff of the digital both practical and theoretical than is the norm at the moment. I believe that this means teaching the fundamentals of hardware and software, understanding software development methods, creating bridges to those disciplines that are viewed as been on the sciences side of the fence rather than the humanities. Inevitably society itself will enforce these changes so it would be better if the change came from within academia than without.

7 Endnotes and citations

A thing is funny when it upsets the established order.
Every joke is a tiny revolution.
George Orwell, Funny, but not Vulgar[7]

The endnotes are in the order they appear in the body of the text. The references and citations in the following subsection are listed here in alphabetical order – the year in brackets is the original year of publication in the original language. I have also supplied the publication year of the copy/edition that I have accessed. I have attempted to include the translator and the year of first publication in English for texts that were originally published in a language other than English. These references are generated automatically by the dastardly combination of RefWorks, JabRef and BibTEX (using LYX); and the format is APA-like.

Standard legalese boilerplate: all (registered) trademarks and logos are the property of their respective owners.

[7] http://www.tinyrevolution.com/mt/archives/000006.html

Endnotes

[1] Public domain text. Modern History Sourcebook: Immanuel Kant: What is Enlightenment? (1784):
http://www.fordham.edu/halsall/mod/kant-whatis.asp

[2] The world in 2011 courtesy of the International Telecommunication Union:
http://www.itu.int/ITU-D/ict/facts/2011/material/ICTFactsFigures2011.pdf

[3] "He is reported to have first said this in an interview on Fresh Air, NPR (31 August 1993)", as reported by:
http://en.wikiquote.org/wiki/William_Gibson

[4] Markku Eskelinen & Raine Koskimaa, Introduction:
http://cybertext.hum.jyu.fi/articles/19.pdf

[5] In The Theater and Its Double. "Discussing the "mysterious identity of essence between alchemy and the theater," Artaud argued that the theater creates a virtual reality – "la réalité virtuelle" – in which characters, objects and images take on the phantasmagoric force of alchemy's visionary internal dramas."
http://fusionanomaly.net/antoninartaud.html

[6] The homepage of the LaTeX software project, a typical typesetting tool:
http://www.latex-project.org

[7] The homepage of Wolfram Mathworld, curated by Eric Weisstein:
http://mathworld.wolfram.com

[8] From an exchange with Diana Harrelson, PhD Student in Human Computer Interaction, who shared a PDF of her PhD, Fedora Practicum, with me:
http://www.cyber-anthro.com/

[9] Jyväskylä university hosts a website dedicated to research methods, both qualitative and quantitative:
https://koppa.jyu.fi/avoimet/hum/menetelmapolkuja/en

[10] The homepage for Microsoft Word, a typical word processing tool:
http://www.microsoft.com/word

[11] The homepage for Adobe InDesign, a typical desktop publishing tool:
http://www.adobe.com/products/indesign.html

[12] The homepage of Benojt, a fractal exploration tool:
http://benojt.inner-space.de

[13] The homepage for 3DAttractors, a Mac application:
http://amath.colorado.edu/faculty/juanga/3DAttractors.html

[14] Four Colur Theorem, overviews:
http://en.wikipedia.org/wiki/Four_color_theorem
http://people.math.gatech.edu/~thomas/FC/
http://mathworld.wolfram.com/Four-ColorTheorem.html

[15] MATLAB, a programming environment for algorithm development, data analysis, visualization, and numerical computation:
http://www.mathworks.co.uk/products/matlab/

[16] Magma, a computational algebra system:
http://magma.maths.usyd.edu.au/magma/

[17] : Maple, mathematical computation engine:
http://www.maplesoft.com/products/Maple/index.aspx

[18] Mathematica, an application for computations:
http://www.wolfram.com/mathematica/

[19] Comprehensive tabulated online resource by Freek Wiedijk of software realted to mathematics in some way:
http://www.cs.ru.nl/~freek/digimath/

[20] REDUCE: The First Forty Years:
http://reduce-algebra.com/reduce40.pdf

[21] H.-O.Peitgen, H. Jürgens, D. Saupe, C. Zahlten, Fractals – An Animated Discussion, Video film, Freeman 1990

[22] H. von Koch, Sur une courbe continue sans tangente, obtenue par une construction géometrique élémentaire, Arkiv för Matematik 1 (1904) 681-704. Another article is H. von Koch, Une méthode géométrique élémentaire pour l'étude de certaines questions de la théorie des courbes planes, Acta Mathematica 30 (1906) 145-174.

[23] D. Hilbert: Über die stetige Abbildung einer Linie auf ein Flächenstück. Mathematische Annalen 38 (1891), 459–460.

[24] G.Peano: Sur une courbe, qui remplit toute une aire plane. Mathematische Annalen 36 (1890), 157–160.

[25] "In tibetan Buddhism the Kalachakra Mandala is a cosmographic representation of the inner, outer, and alternative dimensions of reality. The Body Mandala is surrounded by crescent-shaped areas, which are the offering grounds. Next follow circles which represent the elements: earth (yellow), water (white), fire (pink/red), wind (gray/black) and space (green). The outermost circle is the "Great Protective Circle", "Mountain of Flames" or "Circle of Wisdom" which represents the wisdom element. The differently colored areas represent the five Wisdoms of the Buddha in the form of a rainbow. Picture take from a Kalachakra Mandala tanka in Kathmandu"
http://michel8170.deviantart.com/art/Kalachakra-Mandala-79844211

[26] Man Ray - Interview in Camera (Paris, reprinted in "Man Ray: Photographer", ed. by Philippe Sers, 1981).

[27] Giovanna Di Rosario « On the Surface of Aleph Null », presented at the conference And Now Festival 2012, 6-10 June 2012, University of La Sorbonne.

[28] "3D Mandelbulb Ray Tracer: Mandelbulbs are a new class of 3D Mandelbrot fractals. Unlike many other 3D fractals the Mandelbulb continues to reveal finer details the closer you look."
http://www.subblue.com/projects/mandelbulb

[29] Explanation why the term intellectual property rights might be ill advised – GNU website:
http://www.gnu.org/philosophy/not-ipr.html

[30] Video talk on Hyde's book Common as Air:
http://cyber.law.harvard.edu/interactive/events/2011/hyde

[31] Line 54, "As to buy water, or wind, or wit, or fire the fourth:/ These four the Father of Heaven made to this fold in common:
http://www.ancientgroove.co.uk/books/PiersPlowman.pdf

[32] What is free software?
http://www.gnu.org/philosophy/free-sw.html

[33] Code cannot be stolen, a review by ZDNet:
http://www.zdnet.com/blog/btl/code-not-physical-property-court-rules-in-goldman-sachs-espionage-case/73984

[34] Original link: http://www.suntimes.com/output/tech/cst-fin-micro01.html unretrievable. Quote by Steve Ballmer, CEO of Microsoft Corporation from 2001 in the Chicago Sun Times – unsurprising as Microsoft is the leading proprietary software company with the most to lose as free software gains:
http://www.linuxjournal.com/article/5007

[35] Website address of the Sage math press kit:
http://www.sagemath.org/library-press.html

[36] Online help video for Sage:
http://www.sagemath.org/help-video.html

[37] Online resource of Sage library publications:
http://www.sagemath.org/library-publications.html#books

[38] (PARI) Wikipedia page:
http://en.wikipedia.org/wiki/PARI/GP

[39] (GAP) Wikipedia page:
http://en.wikipedia.org/wiki/GAP_computer_algebra_system

[40] These software packages are used by Sage:
http://www.sagemath.org/links-components.html

[41] Online resource for developers new to Sage:
http://www.sagemath.org/doc/developer/

[42] Switching from Word Perfect to Word 2000 – Microsoft Corporation online help:
http://office.microsoft.com/en-us/office-2000-resource-kit/migration-guides-for-office-2000-HA001138391.aspx

[43] Oscas NSF white paper (PDF) online resource:
http://sage.math.washington.edu/home/wdj/.../oscas-nsf-white-paper12.pdf

[44] Oscas NSF white paper (TEX) online resource:
http://wdjoyner.com/writing/research/oscas-nsf-white-paper12.tex

Citations

[Enc, 1982a] (1982a). Art, theory of. In New Encyclopædia Britannica, volume 2. Chicago, 15th edition.

[Enc, 1982b] (1982b). Tragedy. In New Encyclopædia Britannica, volume 18. Chicago, 15th edition.

[Enc, 1982c] (1982c). Wasilly kandinsky. In New Encyclopædia Britannica, volume 2. Chicago, 15th edition.

[Aarseth, 1997] Aarseth, E. J. (1997). Cybertext: Perspectives on Ergodic Literature. The Johns Hopkins University Press.

[Benkler, 2006] Benkler, Y. (2006). Wealth of Networks : How Social Production Transforms Markets and Freedom Contract : Freedom in the Commons. Yale University Press, New Haven, CT, USA.

[Bodner, 2011] Bodner, B. L. (2011). A nine- and twelve-pointed star polygon design of the tashkent scrolls. In Sarhangi, R. and Séquin, C. H., editors, Proceedings of Bridges 2011: Mathematics, Music, Art, Architecture, Culture, pages 147–154. Tessellations Publishing.

[Brooks, 1995] Brooks, F. P. (1995). The mythical man-month: essays on software engineering. Addison-Wesley, Reading, Mass., anniversary edition.

[Burns, 2011] Burns, A. (2011). The art of complex flow diagrams. In Sarhangi, R. and Séquin, C. H., editors, Proceedings of Bridges 2011: Mathematics, Music, Art, Architecture, Culture, pages 51–58. Tessellations Publishing.

[Calude, 2000] Calude, A. S. (2000). The journey of the four colour theorem through time.

[Denzin and Lincoln, 2005] Denzin, N. K. and Lincoln, Y. S., editors (2005). The Sage handbook of qualitative research. Sage, Thousand Oaks, Calif., third edition. editors: Norman K. Denzin, Yvonna S. Lincoln.

[Foucault, 2002] Foucault, M. (2002). The Order of Things: An Archaeology of the Human Sciences.

[Gamma et al., 1994] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-oriented Software.

[Gere, 2002] Gere, C. (2002). Digital Culture. Reaktion Books.

[Ghezzi et al., 2002] Ghezzi, C., Jazayeri, M., and Mandrioli, D. (2002). Fundamentals of Software Engineering. Prentice Hall, Englewood Cliffs, N.J., second edition.

[Guba, 1990] Guba, E. G., editor (1990). The Paradigm Dialog. Sage Publications, Inc.

[Hardy, 1940] Hardy, G. H. (1940). A mathematician's apology. at the University Press, Cambridge. by G. H. Hardy.

[Hayles, 1999] Hayles, N. K. (1999). How We Became Posthuman: Virtual Bodies in Cybernetics, Literature, and Informatics. University Of Chicago Press, first edition.

[Hockney, 2006] Hockney, D. (2006). Secret knowledge: rediscovering the lost techniques of the old masters. Thames & Hudson, London, new a exp edition. David Hockney.

[Hyde, 2011] Hyde, L. (2011). Common as Air: Revolution, Art, and Ownership. Farrar, Straus and Giroux, first edition edition.

[Knuth, 2009] Knuth, D. E. (2009). The Art of Computer Programming. Addison-Wesley, Boston, Mass. London.

[Koestler, 1959] Koestler, A. (1959). The Sleepwalkers. Hutchinson, London. Introduction by Herbert Butterfield.

[Kurzweil, 2005] Kurzweil, R. (2005). The singularity is near: when humans transcend biology. Viking, New York. Ray Kurzweil.

[Lessig, 2001] Lessig, L. (2001). Future of Ideas : The Fate of the Commons in a Connected World. Random House, Incorporated, Westminster, MD, USA. ID: 10005111.

[Lessig, 2005] Lessig, L. (2005). Free Culture: The Nature and Future of Creativity. Penguin, NY, USA.

[Lessig, 2006] Lessig, L. (2006). Code 2.0. Basic Books, New York. http://creativecommons.org/licenses/by-sa/2.5/.

[Luhmann, 1982] Luhmann, N. (1982). The Differentiation of Society. translated by Stephen Holmes and Charles Larmore.

[Lyotard, 1984] Lyotard, J.-F. (1984). The Postmodern Condition. University of Minnesota Press, Minneapolis. translation from the French by Geoff Bennington and Brian Massumi ; foreword by Frederic Jameson.

[Mandelbrot, 1983] Mandelbrot, B. B. (1983). The Fractal Geometry of Nature. W. H. Freeman and Company, New York, updated and augmented edition.

[Manovich, 2002] Manovich, L. (2002). The Language of New Media. The MIT Press.

[Peitgen et al., 1992] Peitgen, H.-O., Saupe, D., and Jürgens, H. (1992). Chaos and Fractals: New Frontiers of Science. Springer-Verlag, New York, Berlin, Heidelberg, London, Paris, Tokyo, Hong Kong, Barcelona, Budapest.

[Raymond, 2001] Raymond, E. S. (2001). The Cathedral & the Bazaar. O'Reilly Media, Sebastapol, CA, USA, revised & expanded softcover, january 2001 edition.

[Rosenberg, 2008] Rosenberg, S. (2008). Dreaming in Code. Three Rivers Press, Random House, Inc., New York, first paperback edition.

[Snow, 1993] Snow, C. P. (1993). The Two Cultures (Canto). Cambridge University Press. Rede lecture: 1959.

[Steadman, 2001] Steadman, P. (2001). Vermeer's camera: uncovering the truth behind the masterpieces. Oxford University Press, Oxford. Philip Steadman.

[Stein et al., 2011] Stein, W. et al. (2011). Sage Mathematics Software (Version 4.7.x). The Sage Development Team. `http://www.sagemath.org`.

[Torvalds and Diamond, 2002] Torvalds, L. and Diamond, D. (2002). Just for Fun: The Story of an Accidental Revolutionary. HarperBusiness: An Imprint of HarperCollins Publishers, first paperback edition.

[Tufte, 2001] Tufte, E. R. (2001). The visual display of quantitative information. Graphics Press, Cheshire, Conneticut, second edition.

[Turkle, 2011] Turkle, S. (2011). Alone Together: why we expect more from technology and less from each other. Basic Books, New York. Hardcover.

[Vattimo, 1988] Vattimo, G. (1988). The End of Modernity: Nihilism and Hermeneutics in Postmodern Cultures. Polity Press, London. Gianni Vattimo ; translated and with an introduction by Jon R. Snyder.

[Williams, 2009] Williams, S. (2009). Free as in Freedom. SoHo Books, Lexington, KY, USA, paperback edition reprint edition. reprint, 07 February 2011, http://www.gnu.org/licenses/fdl-1.1.html.

[Wu, 2011] Wu, T. (2011). The Master Switch: The Rise and Fall of Information Empires. Alfred A. Knopf, Random House, New York, Toronto, trade paperback edition.

The above bibliography is created from a BibTEX database.

8 Appendix

Included with this thesis is (1a) a copy of the OpenDocument (text) format revised questionnaire delivered to the Sage mailing list and (1b) a copy of the online survey form of the same document powered, as you can see from the bottom of the document, by Google Docs. The answers arrive in spreadsheet format. I cannot stress enough the sheer elegance and utility of this method. Also included is (2) a copy of the slightly modified questionnaire to the group of people dealing primarily with proprietary software – again this was online. One piece of advice, continually check for new responses, new ones are easy to miss. I have made HTML files from the online questionnaires rather than sharing including the links but the links are available on request.

- (1a) FREE & OPEN SOURCE MATHEMATICAL TOOLS RESEARCH - Spreadsheet View Form.zip
  contains HTML + CSS files
- (1b) FREE & OPEN SOURCE MATHEMATICAL TOOLS RESEARCH - Revised Questionnaire.odt
- (2) MATHEMATICAL SOFTWARE TOOLS RESEARCH - Spreadsheet View Form.zip
  contains HTML + CSS files