GEOMETRON

by Trash Robot

The Book of Geometron

Trash Robot

May 25, 2021

Contents

Contents **iii**

List of Figures **v**

1 Civilizations **1**

2 Organic Media **16**

3 Street Network **38**

4 Servers **60**

5 Scrolls **68**

6 Feeds **86**

7 Maps **97**

8 Symbols **110**

9 2d Web Graphics **135**

10 Shapes and Fonts **164**

11 Action Geometry **185**

12 Machine Control **203**
12.1 Machine Control 203

13 Geometron in 3d and Beyond **221**
13.1 Geometron in 3d and Beyond 221

14 Full Stack Geometron **239**
14.1 Full Stack Geometron 239

15 Ontology **245**
15.1 Ontology 245
15.2 On Self-Replicating Sets 248

16 Icons **283**

17 Trash Robot **284**
17.1 Trash Robot 284

List of Figures

8.1 vesicapiscis 116
8.2 pentagram 118
8.3 hexagram 119
8.4 squareroottwo 120
8.5 scales 121
8.6 cursormovements 122
8.7 colors 123
8.8 uparrowspelling 124
8.9 cubes 126

9.1 keyboard 137
9.2 cursoredit 139
9.3 move 140
9.4 angles 141
9.5 scaleactions 142

9.6 basicdraw 143
9.7 colors 144
9.8 pathactions 144
9.9 bezieractions 145
9.10 bezierbracket 146
9.11 panzoom 147
9.12 flagactions 147
9.13 vesicapiscis 148
9.14 cursorscale 149
9.15 cursorangle 150
9.16 cursorsquare 151
9.17 cursorroot2 152
9.18 cursorgolden 153
9.19 cursorroot3 154
9.20 cursor3 155
9.21 cursor5 156
9.22 styleeditor 157

10.1 RLC 170
10.2 inductorloop 171
10.3 inductor 172
10.4 RCline 173
10.5 treeoflife 174
10.6 treeoflifespelling 175
10.7 estrogendiagram 176
10.8 estrogenspelling 177
10.9 hebrew 178
10.10 katakana 179

11.1 shapeset 186
11.2 rulers 187
11.3 protractor 188
11.4 penrose 189
11.5 pilogo 190
11.6 skeletrontripod 191
11.7 skeletron2 192
11.8 shook 193
11.9 tapesnake 194
11.10artboxnet 195
11.11artbox 196
11.12artboxtop 197
11.13pyramidnet 198
11.14pyramid 199
11.15bags 200
11.16outfit 201

12.1 controller 211
12.2 xyzprobe 214
12.3 basicmovements 215
12.4 actions05xx 216
12.5 printerblockdiagram 217
12.6 buildingwallrobot 218
12.7 eblblockdiagram 219

13.1 actions3d 222
13.2 shapebuilding1 224
13.3 shapebuilding2 225

13.4 whimsycastleturret 230
13.5 whimsycastlespelling 231
13.6 whimsycastle 232
13.7 robot0500 235
13.8 trashrobot3d 236
13.9 angles 238

17.1 artboxelements 289
17.2 artboxtriangle 290
17.3 artboxtriangle 291
17.4 artboxtnet 292
17.5 artboxassembly 293
17.6 artboxrainbowskin 294
17.7 artboxhole 295
17.8 artboxtie 296
17.9 artboxcopy 297
17.10trashtie 301
17.11flag 302
17.12flagtrashrobot 303
17.13alphabet 304
17.14alphabet 305

Chapter 1

Civilizations

Dig it up, set it on fire, and bury it. Our civilization is an ever-accelerating destructive flow of material from mine to landfill. This work is based on the idea that we can do better. Much better! We have dug such a vast array of useful minerals up in the last few hundred years and carried out such fantastic transformations on them into useful materials that we now have the opportunity to build a whole new civilization with a completely different structure than the one we presently inhabit.

Our current industrial system is a tripod resting on these three legs: money, mining and property. These ideas make up a philosophical framework for understanding and building our world which is failing at an ever-accelerating rate. In order to build a new civilization, we must study in detail the structure of the existing one.

Money as an idea is so integrated into our world view that it is very hard to even see what it is. At its philosophical core, the idea of money is that there is a property we call "value" which can be denoted by a number. This is considered so obviously true by defenders of the existing economic system that when challenged, they simply re-state repeatedly the basic ideas immediately assuming that any system not based on money is just the same set of ideas re-arranged. That is, people will argue that doing away with money will simply replace selling a things with a certain "value" denote by prices with trading things to which such a value can also be assigned, but just in a much more inconvenient way(barter). Similarly, the defenders of the idea of money will argue that "storing value" is an important task, again assuming that some kind of thing called "value" can be measured with numbers and that doing this all-important value storing will simply have to be done with vaults of metal if we dispense with money.

But this monetary way of thinking does not take into account the possibility that the value of a thing might multiply by replication or that value can be created from nothing. As long as everyone in society is exchanging value along this stream from landfill to mine, we can use the numbers we call "money" to roughly create an equivalent to the main type of value, which is physical material. We can measure how much gold or cobalt or salt we have, and money lets us transform value from one

of these physical things to another, easily trading lithium for silver or oil for aluminum.

Furthermore, of course, there is labor. In the labor theory of value, again, money is used to assign some fixed value to a certain amount of "work" people do to produce a thing. One can superficially dispense with money but as long as we accept that value comes from how many hours an actual worker one does some type of task, we are just shuffling the details around but preserving the structure. But what of automation? And what of other forms of even more drastic increases in efficiency which are possible with information technology? Again, once we allow for full automation and many orders of magnitude increases in efficiency, we find that there is no way to create a value system using numbers which actually describes reality as we experience it in an information economy. Numerous band aids have been proposed, but if we project forward to more and more automation and further increases in efficiency we see that we have to re-evaluate the whole labor theory of "value" along with every other theory of value we hold as a truth today. The problem, again, is that using numbers to denote value simply will never be compatible with the new civilization we must now build in order to survive.

What happens when we create value from something we have an effectively infinite amount of but which we only have a finite need for? What if, for instance, I live near a landfill with a vast store of plastic and electronic

trash, and I want to build a small factory which converts plastic trash into useful furniture for direct use near the material I have. Suppose the electronics trash has all the material I need to build this fully automated factory, and that the material needed to create so much great furniture that no one near me ever has to buy furniture again, that they can get custom high quality furniture which can be repaired indefinitely without even tapping too deeply into the vast store of plastic trash available directly in our community. This act of creation brings a thing of great value into being from nothing but information. In this situation, money fails. As long as we rely on money to denote and store value, everyone creating value from nothing has to either get someone in the money-creating business to create currency specifically for them or to do that themselves as they create.

But the more creative our new industrial processes are, the more catastrophically the money system fails. If I share the information on how to build the thing with the rest of the world, in principle the equivalent of trillions of dollars of value can be created from nothing.

This failure is not hypothetical. Creating value from nothing which can self-replicate freely is precisely what software does. Software produces things of great value with no material input needed at all, and is able to almost instantly replicate to the whole of humanity. If someone can create almost infinite value with no input of labor, energy or materials(after the initial creation pro-

cess), what does that imply for the rest of the people? We are seeing this every day. Every city in the world right now is witnessing a violent takeover by the replicators. Those who can replicate their products infinitely for free include the media industry, marketing, software, finance, and all the numerous information based businesses which make up those business ecosystems. We are finding that at an ever-accelerating rate all the wealth in our society is being transferred from people who produce things that do not replicate, like physical goods or physical labor to the replicators.

The purpose of the work here is to show that there is another way. We can create an information based economy made up of self-replicating information which reproduces things of value which cannot be added up using numbers. This is not some crude replacement for money using barter, but rather a whole new approach to everything: we will have to build a new way of thinking about machines, society, mathematics, and philosophy. Just as the existing system is based at its deepest level on numbers, we propose basing our entire civilization on geometry. Geometron is a universal geometric language which we can use to express the self-replicating information we will need to build this new civilization.

The second leg of our tripod of civilization is property. As with money, the idea of property is very difficult to examine as it is so deeply woven into the fabric of everything we do. The world view we are taught in school

and at home is that just as everything has a price that everything is owned by someone. In some cases that ownership might be the state or even some type of "common property" but all things are in some way property. In the dominant ideology of our time, air and water are property, the land is property, the genes in our DNA patented by the drug companies are property, and even these words I am typing are property.

Again, as with money, we have to examine how this idea is going to fail more and more catastrophically as we evolve into a civilization based on the replication of technology from trash rather than consumption of mined materials using labor. In many ways the purpose of property is to inhibit replication. In the case of intellectual property this is in fact its *only* purpose. But even for physical property like land, the whole idea is that if I "own" land the real purpose of that ownership is to make sure someone else does not own it.

The idea of property makes sense when we are all competing for resources we have to dig up out of the ground. If I dig up 1 ton of silver it means you can't and vice versa. We are all in competition to be the owner of that silver. If you are 1000 pounds of silver richer, I'm 1000 pounds of silver poorer. But we have to see that with an informational economy based on trash as the main input this is no longer the case.

If I consume a ton of trash from a landfill into useful things, we have to recognize that that ton of trash has a

negative value, which changes completely what it means to use it to make things. If I consume 1000 pounds of oil to make a plastic part, that oil has a cost we now measure in money I have to spend to buy the oil. But if I consume that from a landfill, the cost is negative! Regardless of what I make from the plastic, simply having it be something useful at all is of value. In our existing consumer economy we all actually have to spend money to dispose of waste. So an economy built entirely on waste streams breaks the whole idea of property up. If I have a pile of trash on my land and you take it away that has value to me and also to you. We are no longer in competition in this relationship–the more trash you take the more benefit I get but also the more you get.

We must also recognize how an economy based on the replication of technology from trash using free information technology changes the incentives in regards to intellectual property. If I create a new technology now and release it into the world, the only way for me to make money on it is to retain some control, which is now expressed by means of the property system. But if I create a new technology from trash which does something useful, and it's intended to only be used locally, the value proposition changes. I use local materials to make a thing and directly benefit from it. When I share it with you, you also do that, but if you then improve upon it and release it back into the network, I can immediately benefit from the improvement.

If we build feedback loops across a global network we can get exponential speedups of technological development. What this means is that if I am a technology creator and share my creations freely, the thing I create can be instantly transformed into a thing co-created by a global community which is vastly superior to what I would have made alone. If my only goal in building technology is to directly convert the trash in my physical environment into useful things, my choices in regards to how I relate to the rest of the world will be based on trying to do that better and better. If the more I share the more this happens, often with what I make being totally replaced by something much better, my incentives undergo a radical shift. What I now want more than making a good thing and controlling it is making a thing which entices others to improve it. Again, as with money, we find that the idea of property inhibits us from doing what we need to do to build this new civilization.

The third leg of our tripod is mining. This is perhaps the most fundamental. The entire basis of our long strange trip from stone tools to bronze to iron to steel to silicon and so on is based on mining. We need mines to get the materials to make things. In order to make complex things with many materials, we need a global system which maintains physical control over all those mines using the system of property and the governments which uphold that property. We also need a global supply chain again based on stability of governments and

large institutions to maintain the constant global flow of goods. In order for our current system to work, every individual element from oil to lithium to uranium must be transported to everyplace on Earth. Conversely, every piece of land with a resource on it will under the current system be pressured to extract that resource and push it out into the global economy.

Under the mining regime, every single type of mine is a choke point to the whole global system. This regime is inherently conservative: any threat to any part of it will make the whole system fail, harming everyone who relies on it which is presently everyone. In order to keep the system running, therefore, a constant global regime of military force is required. One cannot have a mining based civilization without military empires to control the large scale flow of materials.

And again as with money and property we have to examine how the system of mining affects our relations with our fellow people. As long as value all comes from a mine, we are all in competition at some level for the mines products. Every so-called "developing" nation will be forced by the dominant powers to extract all their resources to benefit someone else since all the nations are in competition to benefit from those resources.

But when we build everything in our civilization from the trash of the old world this situation completely changes. My motivation as a producer of technology is now to turn trash into things of value as much as possible. If you

are thousands of miles away, and we never exchange any physical goods, just information, my incentive is now to have you replicate the trash technology as much as possible. This is for several reasons. First, there is the same reason as stated above in the discussion of property: the network effect. The more people copy my technology the better it will get, and the more comfortable my own life will become. But also, if we build a society which abolishes the mine, as long as other people are mining they will pose a threat to us. As long as anyone in the world bases their civilization on mining, they will need to build empires and dominate large masses of land in order to keep mining. So if I want to not be invaded by a mine-based empire it is in my best interest to help every other place on Earth develop the same technology in order to also prevent the violence of the mine from destroying what we build.

In today's world *every* single useful material we need for advanced technology has been pretty evenly distributed to every corner of the globe. This is totally unprecedented! There is nothing in history even remotely similar to the situation we find ourselves in today. People do not seem to have really grasped how fundamental and irreversible this is. Even if humanity died out tomorrow and were replaced by evolved crows in 100 million years, the distribution of rare minerals around the globe will remain. We will never again have to discover from scratch how to find and extract materials like cobalt or tanta-

lum. We not only have all the materials needed to build an advanced technological civilization from scratch, those materials are already in a very organized form specifically designed to be useful. Aluminum has already been extracted from bauxite, iron has been smelted into steel, silicon purified into wafers of unimaginable perfection and so on.

Examining all three of these legs(money, property and mining) it should now be clear that these ways of thinking fall apart when we build all of our technology from trash instead of mined materials. It is my intent in this work to build a framework for creating this new world. To do this, we will need to build up a whole world of thought and action from scratch. The most fundamental focus of this new world is replication. We study how media replicates, how machines replicate, how software and hardware replicate, how whole systems replicate, and how pure information replicates.

We also take as an axiom that geometric thinking is more fundamental than numerical. This is because geometry is what we use to actually make things. From buildings to microchips to injection molded plastic enclosures, all technology is essentially a geometric construction of one kind or another. So if we are interested in building media the sole purpose of which is to replicate technology effectively, we find that geometry is the most fundamental form of mathematical thinking. As with the existing system of thought we currently live in,

we will need to delve deeply into our most fundamental assumptions of how the world works. But now rather than trying to find some abstract truth, as the mathematicians of the early 20th century did, we build up a system of thought based on outcome: that which freely replicates useful things from trash is the goal, whatever that turns out to be. This will lead to re-evaluating how we think about machines and mathematical philosophy, replacing the theories of "computers" with ideas about geometric machines to print symbols.

One final note to make about this geometric world view. As with all our mathematics in this new civilization, our goal is replication, not finding some higher truth. This means that geometry is all based on its meaning to humans. Even if the meaning is in the angle of a turbine blade which communicates a different level of air movement in an air conditioner, it is this meaning we care about, not some abstract theorem to prove or algorithm to own. We therefore take language and symbols to be the most fundamental elements of which our Universe is constructed. We accept that whatever we may think or do, the "real" universe is separated from our minds by a veil of language we can never fully see clearly through. Reductionist science has made the mistake of ignoring this veil and focusing on a hypothetical "objective reality". Whether or not this is a permanent intellectual dead end is of no interest to us here. We want results, fast. We want a better civilization in our lifetimes. And

to do that we build up a new way of thinking about information where our desire to provide direct value to people and replicating that to as many people as possible is our most fundamental axiom.

We call this geometric system of value, this geometric meta-langauge, Geometron. This is the Book of Geometron, which describes how to replicate the whole system.

To build this world we will first discuss how media needs to change to support this new way of thinking, then how we will physically deploy this media infrastructure to the streets of our world. We then show how the software works, how it replicates, and how you can add to it, improve it and make it your own to share. After that, we step through all the different layers of information that make up this new network. We then talk about our theory of being(ontology), the new underlying mathematical philosophy which we are using to replace the axiomatic set theory that 20th century mathematicians used to describe reality, as well as the theory of machines which replaces the Turing model of computation. Finally we use this idea of how information works to discuss the self-replicating set known as Trash Robot which we will replicate through the network and also use as a vehicle to help stimulate replication.

Chapter 2

Organic Media

In a consumption-based civilization the purpose of all media is to stimulate consumption. This can come in the form of advertising, corporate propaganda, or the legitimization of the imperial power required to keep mines and long distance supply chains operating. In the age of digital media we find that the hardware itself is also a large component of how it facilitates consumption, with planned obsolescence creating a stream from mine to landfill unprecedented in human history. A constant race to build ever more exotic materials and technologies into physical media devices creates a vast suction across the planet, forcing every corner of the globe to exploit anything that can be mined for digital media hardware from cobalt to lithium to be exploited as fast and widely as possible.

This state of affairs creates a powerful opportunity for a new type of media. The fact that the existing system pumps out this constant stream of new objects with all the stuff needed for advanced information technology creates a resource which can be used to build new hybrid technologies designed to incorporate scavenged parts from the discarded tech. The tech industry is now working on turning out *trillions* of "internet of things" devices–objects which have built-in networking capability and could in theory serve as web servers. Based on how these industries are structured we know that these will all be designed to fail on a very short time scale, and what they are selling today will be in a landfill in less than 5 years.

It is worth marveling at the scale of consumption built into the current digital media system before discussing the alternative. People walk around with screens in their pockets, the sole purpose of which is to manipulate them into consuming more. Those screens are built on a technology which uses the most exotic materials known to humanity, extracted at great human cost from every corner of the globe. And then they are forced into obsolescence within months in some cases by a software industry which is based around the idea of planned obsolescence. Having pushed their way into the pockets, homes and workplaces of something like half the humans on Earth, the industry is now pushing to put their devices in things which have no reason to be part of this network, like toasters

and juice makers. The sheer insanity of this is hard to wrap ones head around, but because the entire media is controlled by this industry it is hard to even articulate in public how insane this is. And it is getting worse very quickly. The need to replace this parasitic monster with media which serves the needs of humanity could not be more urgent!

What we want from a media technology to build our new trash-based civilization on is to replace consumption with replication. We are now constantly buying new machines built from mined materials which constantly tell us to consume more things. Consumption-based media forms a consumption information loop. We want to form a replication-based loop, where the media is built from trash and contains the information required to replicate itself. We call this "organic media", because it behaves like a living thing. In fact it effectively *is* a living thing. If we built closed loop systems in which humanity is reusing material again and again forever in order to live in harmony with the world around us, it makes sense to think of us in combination with our media and the ecosystems we live in as a living system.

This idea of "organic" media is in direct contrast to "viral" media which dominates in social media today. In viral media, information replicates, just as viruses replicate themselves inside a living organism, but the this is always happening in a space defined by a fixed media entity. Social media platforms encourage informa-

tion to replicate as fast as possible within their systems, as that costs them nothing and induces more people to keep coming back to their platform to be manipulated by their advertisers. But if someone tries to replicate the media platform itself by for instance trying to start a new platform, they will do anything they can to stop them. Media today can be viewed in biological terms as an apex predator which kills everything in the rest of the ecosystem and is full of viruses.

When we say we want media to be organic what we mean is that we want the media platform itself to replicate. Just as each new tree or squirrel in a forest really is a whole new instance of "tree" or "squirrel", with no central entity controlling them, we want each new instance of our system to be self-contained. We want it to be able to replicate itself right where it stands, with no outside input from some central system of any kind. This is only possible because of the waste of the existing industrial system: all the materials needed for advanced information technology are sitting in trash bins, dumpsters, closets, and landfills within walking distance of wherever you are reading this right now. All you need to build a whole new media ecosystem from scratch is information: the information required to gather the people and materials required to build it. If the system you build tells people how to do this, it can freely replicate across the whole world without any central infrastructure.

It is worth noting that building this is hard. Modern

digital media technology is designed by hostile engineers to be as hard as possible to fix, modify, or use for anything other than consuming advertising for a few months before it goes to the landfill. Those machines are built by vast teams of well funded groups with extremely specialized technical skills. It will take a concerted research effort to fully replace the existing information technology system with a free one built from trash. In a later chapter, I will discuss how we can do this by using a different system architecture in which the purpose of the whole system is displaying of a specific class of documents based on the software presented here. But for the time being, in order to launch our new media system, we will rely on existing off the shelf hardware which is still part of the consumer system but not the main commercial advertising-driven part.

This book is therefore doing two things in regards to launching this system. It is launching a new social media platform based on using the Raspberry Pi as a local web server used over local wifi networks, and it is laying the conceptual framework for building a whole new information technology system from the ground up on new principles. Just as technology people in the existing system refer to a "technology stack" we are building a whole new "stack" in the sense of a collection of technologies which are all related by a chain of increasing or decreasing abstraction or proximity to the user which work together to make our system work. In this work,

we describe the whole stack. We launch fully functioning software and hardware for parts of it, and describe how we will build the other parts that require more work.

The most important part of the current project presented here is that it work for its purpose which will support the rest of the development. This means that the technology has to work to distribute new technology of all kinds built from trash which people need or want, which can be freely replicated. We use this partially consumption driven media system to launch self-replicating media systems which really are built from trash as a demonstration. Our metric of success will be how this self-replicating media technology replicates and evolves. If we can make it replicate by making things people want, and make it evolve by creating a strong incentive for people to improve it, the system will naturally evolve into the one we need, which no longer requires any input from the mine-based system to function anymore.

Just as a relatively small number of people a few hundred years ago sending each other letters built the basis of the current explosion of technology which led to the existing world order, we believe that a small number of people with new ideas today can build a much faster explosion of information which consumes the existing world of consumption and replaces it with locally closed loops of material in a single generation. We also believe that this is of the utmost importance to do as quickly as possible. The existing system is killing us. It is destroying the

natural world, needs constant warfare to function, and is increasingly driving anyone outside the technocratic elite into extreme poverty.

So how do we actually build this "organic media"? We start by looking at living systems, both individual organisms and larger systems like forests. The most fundamental thing life does is replicate. This will probably get tedious for the reader, but replication is the thing this work will come back to with relentless repetition but that relentless repetition of replication is precisely what makes life work. A living system is a system of thing which all also replicate. Living systems replicate over all scales: forests replicate, but so does the RNA and DNA in each cell of each organism in the forest! We will also build our systems this way: many components make up systems from tiny scraps of code or single cutouts of cardboard up through whole vast industrial fabrication systems, and we want *all* of them replicating. Again this is in direct contrast to the existing system in which small parts like shared memes on a platform are supposed to replicate but the company itself is designed around non-replication.

Another property of life is that it is an independently evolving thing. Because organisms have an independent life, they can change in much more unpredictable ways than centrally controlled systems like a large corporation, government or non profit(this includes open source software projects with a central code base that all instances

are copied from).

Finally, all life dies. In order for life to work we need the cycle of death to be natural. Just as fungi in a forest turn logs into soil we need the destruction of all things in our system to be natural. This is again in direct contrast to the existing system in which all media is built out of a company which is designed to grow forever and never die.

In order to build our platform then we want to write down a set of rules which will guide all of our work. These are nine rules:

Everything replicates. This is the most fundamental law. It is what makes life alive. And it is what makes media organic rather than viral or parasitic. This means that all our software contains code to replicate itself without any reference to a central code repository. The code on a server in a coffee shop can directly replicate to the laptop or phone of every person in that coffee shop with no connection to the rest of the Internet at all. All our hardware is built into some kind of media which describes its replication.

Everything evolves. All things can be edited by all users. To be in contact with a thing, be it a file or a physical machine is to have the power to alter that thing totally. There are no "users" or "engineers" in the sense used today. Some people will choose to edit things more than others but everyone *can* edit all things.

Everything dies. All things can be deleted or de-

stroyed by all people. This is particularly important for files. Much of the power structure we are trying to destroy rests on information we are not allowed to destroy, from the intrusive and parasitic industry of buying and selling personal data through the constant advertising we are not allowed to turn off. Also, in order to be able to stop harmful information, we empower every single user with no exception to be able to delete every single piece of information they come into contact with, with no exceptions. This is less destructive than it sounds. Because our network is all physically local, no central bad actor can wipe out the whole network. If all networking is at the level of a wifi network, and they are all constantly being destroyed and rebuilt anyway, the cost of universal destructive power is outweighed by the benefit of people being able to stop bad information without any appeal to authority.

No property. As discussed in the previous chapter, the idea of property is not compatible with a civilization based on self-replication. Since the idea of property fails at scale in a replication society, and since our goal is to scale, we dispense with it immediately and build infrastructure which not only has no intellectual property but where the physical web servers are not owned by anyone. Initially we will have to spend money to buy parts to build them, but as soon as they are built we will release them to whoever we think will get the most use from them, along with instructions for them to do the

same, passing all infrastructure along to wherever it gets the most use. Building network infrastructure without property for the benefit of our communities means that our incentives are now to find whoever has the most need, identifying their needs, and using our technology to serve those needs and fast and directly as possible. If people benefit from the systems, they will naturally be able to replicate, which will further replicate the non-property technology. Initially this means building Raspberry Pi based web servers and giving them away to the people with the greatest need.

No money. This is connected to the rule against property. Initially of course we will need to spend money to buy parts, and will need to have users make money to support their near-term survival. But as we scale up and get more and more basic needs satisfied by technology built from trash, we want to have the elimination of money be the direction we are headed from the start. This is not as far fetched as it sounds. As will be discussed in the next section, barter will be an incredibly powerful tool for scaling. Our network can provide huge benefit to very powerful and wealthy people, and if our people with the most need are dispensing this benefit, we will be able to barter the things we need to scale directly. When our network helps a business with a lot of unused space to make money, they can let us use their space. When a business person makes connections using our network which make them money, letting us scale

our software up on some of their servers will make economic sense to them. We will be able to barter our value as network builders into the things we need for personal survival like places to sleep and food but also the things we need to scale our technology like access to labs and machines.

No mining. Our long term goal is the global elimination of all mining. This includes the whole natural resource extraction industry such as oil and gas as well. This cannot happen overnight, but we don't need it to. Every single mined component we replace with one from a dumpster or landfill takes a little bit of energy and power out of the mining system. if we can remove power from them in a way which self-replicates, our system will simply consume theirs, and mining will be eliminated in a generation.

Everything is physical. This is almost a circular statement. What does it mean for a thing to be "not physical"? This is a statement of belief. We emphbelieve that the idea of information which is not physical is meaningless. All information has a physical manifestation, be it charge on a transistor or bumps on a CD. This law is important as a vocal rejection of any theory of how machines works which states that information or data can exist independently of its physical existence.

Everything is recursive. One of the most notable properties of life is its constant self-referencing. Billions of DNA strands in each individual body of a large organ-

ism all contain a whole copy of the information required to replicate the organism. We see information which points to information which points to information. Life is very self-referential and involves in an abstract sense functions which call themselves constantly. RNA stores instructions to make molecules which replicate RNA, and so on. This law is to remind us as creators of technology to be *constantly* thinking of ways to make things point back to themselves to replicate.

Everything is fractal. This is another property of living systems that we take for granted but which we either ignore or make very crude imitations of presently. Centralized systems of control create technologies which are flat in scale: we build microchips with nanometer precision across millions of nanometers(mm) of scale and so on. In contrast, living systems are fractal in scale such the scale of "error" required to cause catastrophic failure scales with the size of the system. We assume that patterns will repeat again and again at different scales, and expect that our technologies only need to be precise at the correct scale for any given sub-system. This has very specific implications for fabrication which will be explored elsewhere, but as a law we mean that we must simply always have ideas of the fractal nature of living systems in our minds as we create new things in our new trash-based civilization.

With our goals and laws of operation stated, we are now finally ready to delve into more detail into what we

are actually constructing with this work. This is a local network based on a web server loaded on a Raspberry Pi. The Raspberry Pi is a computer on a circuit board about the size of a deck of cards which typically costs about $50. It needs some peripherals including a screen, keyboard, mouse, battery, and memory card, all of which makes it about $200 for a nice, easy-to-use, portable and self-contained system. It only requires a few commands which are easily copy/pasted to install the whole functioning Geometron system on a new Raspberry Pi. It is also modular, and if portability is not needed it can plug into the wall, borrow a keyboard from another system temporarily, and display on a TV, making a non-portable system cost just the value of the board($50).

It is also important to note that this is all modular, easy to buy from many sources, and involves parts many people already have lying around. The Raspberry Pi is widely marketed as a hobby tool and a STEM education tool, but its use case is not always clear. Therefore a large number of people own them but do not use them. They are often sitting in drawers in someones home office or a under-used maker space gathering dust. If we have a use for them and can provide useful services to people it should be possible to directly barter with people who want to support our network who will be willing to donate the Pi boards to our network, where they will become non-property and be distributed to those in need.

The actual software we run on the Pi is what is de-

scribed in the bulk of this book. It is all designed to run in a web browser. Any web browser. So if a Raspberry Pi running the Geometron server software is on a wifi network, all the programs and documents on it can be read, used, edited, replicated and deleted by anyone connecting to that wifi network on any device be it a phone, laptop, tablet, or another Raspberry Pi. Also, the pi itself can serve be used in the same way as any other device on the network.

We must also note an important condition for the pi to be a non-property computer. In order for the pi to be able to freely be shared among the people, it cannot have any personal data on it which causes someone loss if it is read by another person. That means we must never log into private networks like gmail, facebook, or more sensitive things like bank accounts ever on the system. In order for a free and open system without property to function, it must be kept separate from the property based networks. This media platform exists for the sole purpose of free sharing of documents. Any document we do not wish to share we do not put on it.

Also, there are no "users" on this network. This is a network of documents, not users. There are no logins, no passwords, and no databases. User data is not harvested for profit because we simply do not generate the type of information which is considered "user data" in the existing systems.

We take as an axiom in the development of this sys-

tem that documents intended for free sharing are of greater value than private documents, and that the network effect as the universe of non-property documents grows will exponentially increase their value to people until our network out-replicates the existing ones.

Our network is based on sharing several specific types of document which are encoded into the software. We share "scrolls" which are text documents, "maps" which are like presentation slides or memes, "feeds" which are essentially lists of information, self-replicating applications of all kinds(using web based code that runs in a browser, never native code), and generalized Symbols using the Geometron geometric programming language which takes up much of this book. Taken together, these systems of document creation and replication will allow us to describe and replicate any technology of any kind.

This network will be distributed physically, over the Street Network described in the next section. We will initially try to get servers to people who are on the move, living on the streets or in vans and buses, truck drivers, street performers–anyone who finds themselves in nodes of physical networking like dense urban areas or truck stops with many people passing through. The details of how to replicate the server are in the section after next, which delves into the code. Not everyone will need to read this, but everyone should know where to find it, as part of what we will be bartering for as we scale is help from technical experts who can learn the system

and replicate the software as well as add to it to evolve it and decentralize the code base.

The rest of this book describes how to replicate the system, how to use it, how it works, how to develop it into a fully trash-based system, and how we will rebuild mathematics to support this venture.

The Book of Geometron

This book itself is organic media. It is intended to teach its contents to a reader(or rather a small subset of readers) to the level where they can then teach another. This should enable them to re-write future improved versions. In this section I describe how the book was put together, where the files are stored, how to edit them and use the LaTeXdocument preparation system to make the files required to produce a finished book. This means you also need to know what is required to get the physical book printed at an on demand printer, get all the metadata required for publication and distribution, and sell your version in retailers both large and small and online and off. Thus even the book is fully decentralized in principle: if it costs you nothing to set it up to sell, you can sell only a half dozen copies and it will be a net positive, and then if the next person does this it will also be positive and so on.

If the book is decentralized in this way of distribution it has many advantages. If the book turns out to

be disruptive enough that people try to use lawsuits to shut it down or harass an author, but there are 10's of thousands of new authors popping up all the time, it will be impossible to shut down. As some versions turn out to be dangerous or illegal, other versions can immediately be published with omit the offending content. Also, many editions will mean some will get much better than this initial manuscript. Decentralization means that as the manuscript finds its way into communities that speak different languages, the translation can happen without any centralized effort. So for instance if someone translates from the original English into say French, and then it spreads around in areas bilingual with French and some other language like Swahili it can go directly from the French to the Swahili without any involvement of the initial English speaking writers. By avoiding copyright, these improved and translated versions can then get translated back into English and sold yet again under yet anotehr edition. Having editions be unique can create a market for unusual editions, further pumping money into the system and stimulating further development of the book. I would rather see 10,000 people make 100 dollars each selling their own editions of this book to just their friends than see me as the initial author make 1 million dollars on 500,000 copies of one edition.

It is not my intent in the long run to make money on Geometron. It is my intent to create a network which

allows us to live without money by directly bartering what we need to survive(food, a place to sleep and work, medicine, transport) without use of money or any production in the old consumer economy.

All editions are published with a public domain license for everything but the final pdf. The final pdf is published under the minimal copyright required for an author to create the needed publication metadata to get distribution outside of the on demand press used for printing.

Each chapter is a .tex file, using standard LaTeX.

This work must replicate itself completely. We show here how to edit each chapter, publish them to a public Github repository with detailed instructions for further replication, compile the document to a .pdf in book format, and self-publish the fully compiled book on Lulu Press. We then guide the reader to follow the instructions on Lulu to get all the needed copyright metadata for official distribution through normal publication channels. We then describe how to order just a few copies, sell them along with other parts of the system here at a markup, and use the profits to buy more print copies of their own book to place in bookstores and libraries as a fully guerilla activity with no official sanction. This is a little twist on the methodology of Abbie Hoffman's "Steal This Book." In Steel This Book, book sellers had to buy the book, which readers inevitably stole, cutting into book store profits. We use guerilla production meth-

ods to distribute it into bookstores without them spending money. They are faced with a choice: go along with our program and take free money from customers for the book or make trouble for us, throw the book out, and loose what is for them totally free money. If they sell the books for a profit, it benefits our network, because it spreads our ideas but also because it creates a value stream in the existing economy based on what we create, which gives us power in that network. Bookstores want money. But we want network centrality and the ability to control how information flows in a network, and free distribution shifts the power to us. This is why the self-replication of the book is so important. If you want to make your own spin on this book and make it more of a best seller, you do it, but if you leave it open, you hope someone else does it again, and that it keeps getting better as it replicates. Replication instructions for the physical book will live in the README file for the book repository on Github.

Another way to make the book self-replicating is to distribute print copies which a number of blank pages, sold at cost. Artists can buy these, then create illuminated manuscripts with custom illustrations of geometric art using Geoemetron. This can then be bartered based on the needs and powers of the artist, with outcomes as diverse as the artists themselves. We can use this to barter for what we need to expand the network, to survive, and just as another way for artists to exist and

thrive in the world. Also, this can be passed along and shared as non-property. You can buy the hardcopy of the book, add a little art, and pass it along, then the next person adds some and passes it along, and we all help each other out along the way. A famous artist or tech creator might be able to barter for something of very high value. Each physical work will be unique. And since the electronic files which generate the pdf which goes to the hard copy also self-replicate, this entire system can evolve and replicate, as dedicated creators rewrite the whole manuscript and add their version on Lulu press for sale at cost print-on-demand as well, and so on.

Chapter 3

Street Network

What is the Street Network?

The consumer society we live in today relies on a vast and well-organized collection of networks to function. There are the networks which channel freight from one place to another, like trains, trucks and boats. There are the human transport networks like planes, personal cars and buses. There are the invisible networks of power based on who when to school with whom, family connections, or business connections. And of course there is the global information network centered on the Internet which connects everyone together by information.

What we seek to build is the means of replication of trash-based technology in order to propagate our new civilization built entirely from trash. This new system will

be much more localized than the existing one. Rather than needing a constant high rate of movement of very large quantities of physical goods, we plan to build systems which will ultimately use material directly available in our physical environment rather than from far away. Even in places with limited agricultural capacity, I believe that with the superior technology which we can build using a more organic system that we can build dense food production and water purification everywhere once we put our minds to it, making almost all large scale movement of goods un-needed. We will still move goods and people, but more by choice than necessity for personal reasons. When globalization does not force everyplace to be identical, travel can become an adventure again!

Part of the ideology of the existing Internet-based culture is that place no longer matters. People appear on a teleconference or send an email and no one cares where they are. People arrive for a meeting in a business hotel or conference center and it is identical to every other place in the world. The ideological basis of the Geometron Street Network is that place *does* matter and *should*matter. We reject the idea of "nowhere". Everyone is somewhere. Whether you are in a rural area, a suburb, a big city or a highway rest stop you are always *somewhere*. Furthermore, that place you are has its own local geographical logic. It has its own crossroads, its own nodes of power and connection, whether it is the lo-

cal pub or coffee shop or the exit of a subway station. All these little places on the scale of one human body have real meaning for the people who inhabit that space.

We want an information network based around physical replication of technology from trash. To stimulate the replication of the Network, we need it to create value for people who use it and operate it. This value can be of many kinds: it can directly provide physical goods people need, it can facilitate business in the monetary economy, it can provide mutual aid to a community, it can create local social connections, can build network power for users, and any of these values can be traded for materials and space needed to continue to expand the network.

The Street Network consists of the people going out and spreading all this, the web servers we use to do it, and globally visible web pages which serve as links to connect users to us and our network.

We buy domain names which are linked to a place but not property. We avoid .com and focus on .org, .net or .xyz domains. We avoid specific addresses or names of companies. We choose names that describe shared resources which are public enough that no one is in the position to claim ownership of the name. This can be the name of a neighborhood or street, a body of water, a park, or a mobile shared resource like a mutual aid bus used by people who already do not use property. In the physical location described by the domain we create and share physical media which points to the domain. In its

simplest form this can just be a hand written cardboard sign. However, we can also use Geometron to make various self-replicating physical media which transmit the domain, such as laser cut spray stencils and the self-replicating clay tokens described later in this work. We also can fly a black flag of the place with cut out colored felt sewn on a black square of cloth, which is described in the Scrolls distributed with the system.

Each domain is hosted on some commercial server for the time being, from providers such as Dreamhost or Bluehost. Free web hosting can also be used at 000webhost.com and is recommended if you have no money. The Geometron software can then be installed on each of these servers according to the instructions in the next section(and repeated on each Geoemtron instance in the Scrolls), making it just another identical instance of the software to what exists on all the local Raspberry Pi based servers. The primary purpose of these globally visible domains is to point back to the local server. In its simplest form this is just a description of when and where you might find the server. Photographs of the network Operator and their gear can validate the system to passerby. The Map format described in a later chapter can also be used to precisely identify physical places where infrastructure can be found.

The deceptively simple structure described above is the Street Network. We are adding digital media technology to the oldest network in the world: the physical

paths of movement. We will use this to follow all those paths, from superhighways to ancient footpaths to natural harbors, just as other ideas have traveled throughout human history. If we find the most powerful nodes of geography we can build a network of staggering power with a relatively small number of people initially.

Users

The uses of this network are very different for different people, just as is the case for the existing global networks like the Internet. In this section I will discuss some of the different user groups and how our network can provide value for them.

Operators. We are the start of this. Get a Raspeberry Pi and install the system. Get a domain, install the system and point to your server. Go forth and share! Ultimately those of us who build and share this system will form a very powerful network of mendicants. The mendicant tradition has appeared many times in many places in history. A mendicant is someone who is totally devoted to their faith(they are generally religious orders) who renounces wealth and travels with no possessions asking passerby for donations to support them. This has traditionally created contradictions, as these orders have a way of gaining power and becoming anything but poor as they scale up. As our consumer society has destroyed itself it has driven more and more people into this way

of living against their will. If our network provides vast amounts of value to people we will find that the most marginalized people of today when leveraging the power of this new network can barter for not just survival but to thrive in a new civilization without money, mining or property.

We follow the laws of Geoemtron listed in the previous chapter as a guide for building this new world. We teach everyone we meet how this whole system works, and recruit new people as Operators. Note that the idea of a mendicant order has strong religious overtones, but that this is a completely ecumenical order based on the universal language of geometry. Geometry has a central place in all existing cosmologies, both ones considered religious and ones considered secular. The work here presents a way of interacting with the world based on geometry. In some ways this whole project can be thought of as the start of a free school for teaching a new kind of geometry. This is a distant descendant then of the geometry schools of the ancient world. We teach people the whole system; mathematical philosophy, robotics, code, all kinds of industrial fabrication, crafts, fashion, whatever we build we teach and share freely.

Do not misinterpret this idea of the mendicant as a vow of poverty. We will be more wealthy than anyone currently living in the consumer society once we scale this Network. We are building a new world in which no one is poor. By starting from a baseline of people who

have nothing but building better technology than what is presently available in consumer civilization we start by making sure those who have the least have everything: free clean water, free good food, free high technology medicine, free transport, free shelter, free network technology, free air conditioning and heat etc. If those who have the least have better stuff than the richest people in today's world the world of today will dissolve and be naturally replaced by this new order built from the waste of the old.

Traveling kids, hobos, panhandlers, people asking for money or selling things on the street corner. A physically local free bulletin board shared by passerby in a high traffic area can allow people asking for money who are currently ignored by passerby as just another anonymous face and cardboard sign a chance to really tell their stories and to share all that they have to share. When people share their stories they can become part of the emergent physical community of passerby in a location where the network node is located. When people view others as part of their community they not only are more willing to help, they can have open communication about the best way to help, expanding from just spare change to more comprehensive mutual aid. Because we clone content from the local terminal to web pages on globally visible domains linked to a physical place, which are advertised everywhere in that place, marginalized people whose only ability to get online is the public

library can use the computers there to get the information they need to better survive, and ultimately to thrive and build new communities where they already are. The way a local network can help people is twofold. First, it is direct, by asking for money and other mutual aid. But by being physically on location all the time, already with physical media(cardboard signs), people in a given place can aid the network, creating value for the other people in the community who are more resourced, who then no longer view monetary support as "donation", but rather as an expense which supports their other business activities.

In order to see the power of this second means of network support of marginalized people on the street, we have to look more closely at the network nodes we are building. One of the major types of node is in a business district of a city where there are both homeless people asking for money, on the street all day with physical media, and power brokers who make their living entirely from connections. These people include venture capitalists, entrepreneurs, lobbyists, consultants, and the rest of what might be called the "deal-making class". An example of this confluence is some of the parks along K-Street in Washington DC. K Street and adjacent streets is home to a huge homeless population as well as power brokers whose livelihood depends entirely on connections. If a physical network were built which facilitated direct communication between people along K Street, the people

who spend the most time physically on the street can be brokers of information on a network which can be worth a lot to the people who trade in information. Physically local information networks can leverage the power of physical places with very powerful people walking past all the time who normally never communicate. Connecting these people up can be dangerous. But if we provide them with value, it can be worth both a lot of money to them and also potentially something they can barter for giving us space to live and work nearby. If you facilitate a 10 million dollar deal and the customer knows you can do it again, the least they can do is give you a 100 dollar gift card to the nicest restaurant in the block. There is no real upper limit on what an enterprising Network Operator could in theory make if they learned to really channel information efficiently in the nodes of global power. And of course we must remember that when dealing with power brokers their currency is not money. When the people who currently have the most power in society find themselves dependent on free open networks, those networks themselves will gain power which penetrates that of the existing power structures, potentially creating an existential threat to them. We must take note of this.

The elements of traveler culture which overlap with "van life" are also key to increasing the network effects of the Street Network. This also links to trucker networks. People who live their lives on the road can use this network infrastructure to set up complex networks

and markets in highway rest stops, Walmart parking lots etc. using either wifi networks in these places or their own hotspots from their phones. These networks can be of utility to passerby of all kinds, from tourists to truckers to the workers who keep the places running. Just as existing global social media networks provide value they can charge money for, a physically local network can provide value which people will pay for. An example use case here is a Street Network Operator agreeing to maintain a backup of and keep posting an advertisement for something a local entrepreneur is trying to sell to truckers. In exchange for that, they can get directly compensated in gas, right there in the rest stop, without money changing hands.

Food not bombs, street outreach, harm reduction people, mutual aid workers. See above. The people who are working to help the most marginalized members of any given community can better reach that community if there is a physically local media platform where people can share information about resources. Documents can be posted which explain how to get access to resources, when and where resources will be available, etc. Because the whole system self-replicates, as with Food Not Bombs, anything which is successful in any given place can be immediately cloned to other nodes on the network. Food Not Bombs already has a global network of free and open nodes with no property but a very recognizable brand identity and set of behaviors and

actions. FNB nodes are generally already linked by networks both online and via people who travel from one punk house or FNB house to the next. The whole anarchist network of community houses, FNB's, anarchist infoshops and bookstores, really really free markets, free boxes, etc. can form a basis for a truly free information network carried from house to house and city to city, running on house wifi networks.

Business owners in a shopping center. Every business owner has neighbors who are also business owners. You already have an informal network. But installing free digital media infrastructure can provide huge value by allowing more mutual aid between neighbors of all kinds, both owners and customers. Tech giants ignore you. They demand monetary tribute in order to even have your business listed, and then still refuse to give you an equal footing to the corporate giants which dominate their platforms. By controlling a local platform in *your* shopping center, you can provide value to customers with articles they write and share with one another which brings them in(just as advertising-supported media has interesting content to get people to look at ads). And then this medium can have much more than just the ads you would get from a Big Tech platform. You can post really detailed information about everything you do with no restrictions on length, and share across the shopping center. If you own a karate school next to a dentist, the bored people in the waiting room can read about

the history of karate right next to a detailed schedule of your class offerings. And when parents wait for their kids to get out of karate class they can be reading about clean gums in an article written by the dentist. Big Tech doesn't care about you. If you build your own network, you can center it right where it belongs: on the people actually using it, rather than a few oligarchs in San Francisco.

Coffee shop owners. Building a network in a coffee shop on the wifi network which requires purchase to use and which has a time limit can create a huge amount of added business for any local business owner. It also builds community. So coffee shop owners who find themselves with a full shop of laptop drones with headphones on who work for hours, or get kicked out and do the same thing somewhere else can instead find themselves the brokers in a very powerful information network. Much of the commerce of the world is now code written in coffee shops on laptops. Creating physically local networks around these already existing groups can create huge power for the users which then benefits the people who set up the infrastructure(again, just like existing centralized social media platforms.)

Web developers. We need web developers(people who can write HTML and JavaScript code) to be constantly writing more and better software in order to make Geometron a success. Developers who work all day in coffee shops or any other shared space like a co-working

space or pub can have a social network based on both co-developing applications useful to all and sharing other resources. Developers will use the resource of the Street Network terminal/server on the local network in the same basic way as others: they can share their resumes, links to pages of personal projects. Developers are key to the whole system. We must recruit developers with this book who will rewrite all the code and also the book, replicating the whole system. The faster our network can get developers into the swarm, the faster the code itself will improve. Developers are key!! Developers create servers to share into the network. I now ask the reader to look up "steve balmer developers" on YouTube.

Power brokers. Venture capitalists, financiers, entrepreneurs, deal-makers of all kinds, lobbyists, politicians. Your network is your power. Geography matters. Build a network in the lobby. Post things on street nodes, build your network, build your power, build your literal street cred. Deal flow. Deal flow the likes of which you have never seen. Leverage the power of the physical street!

Crafters, makers, jewelers, artists. An alternative to Etsy, street vending, or being in a shop. Post your stuff to the local networks. This is much more free and long form than existing platforms, you can post images, descriptions, contact info, times and places when you'll be in a place. This can be way easier than other sales channels for arts and crafts. You can say when

and where you'll be at a place, post a link for contact, and then show up in the network node like a coffee shop to make the physical exchange. In many cases, because the network is physical and local, there will be barter opportunities as well as direct sales. A barter economy can develop where people donate materials you use for your crafts as part of how they pay for the finished product. Removing shipping or transport costs by dealing directly in a physical location removes friction from the market, amplifying dramatically the power of the market, especially for crafts which involve physically bulky objects. For instance, people can bring in motors and properly prepared plastic sheets and cardboard, as well as rolls and rolls of duct tape, and we can exchange finished products built from these materials and tools, as well as free food, drinks, and supplies, creating a market economy without money as well as without formal business structures(making it easier for marginalized people to participate).

Any labor pool of gig economy workers focused on a specific geographic location. One of the most obvious of these is the drivers who presently drive for the major rideshare apps who all congregate at the airport to pick passengers up in the same exact place, and yet all of it is currently coordinated via the apps(unless you do the cab line). The rideshares apps have proven that cities will ignore illegal cabs if they're done at scale. It would be straightforward for a small team of Network Opera-

tors to run a server which replicates to a page which is advertised around, something like a domain of yourairportnamerides.xyz, which tells users how to log onto the wifi network created by an Operator's hotspot near the pickup zone and with a link on the page to the local network address of the server. All all this IT is doing is directing customers to a dispatcher who manages the drivers over a simple app shared by the collective. The whole network is run by a team of about 2-4 people. One person might be a developer, who creates the app to manage all the drivers and post messages from dispatch. Another person is all marketing, putting up the relevant information in the right places to get seen by travelers but not stopped by the rideshare apps, airport authorities, or the cab companies. Riders will never have their destination information on the public network, nor will drivers put personal information, but they can work on an open trust model where they are known by dispatch, who has code names for them, and operates a queue app which simply adds drivers as the arrive near the Airport and pushes the most senior driver to the top of the stack, which is passed along to a rider. Another Operator might be the one who runs the trust network for the drivers, verifying everyone and organizing meetings for the whole cooperative. This can be used to unionize existing workforces quickly as well, building ad hoc networks which are very hard to suppress visible to everyone on their mobile devices on a local wifi network.

The same model holds for places where workers congregate looking for short term construction work. Those locations can have a server where an Operator runs a labor marketplace where a much larger and deeper labor pool can now advertise, but without all having to be in the physical location. This means a crowd of a dozen workers looking for work can be replaced by an Operator with a sign pointing to the domain where the copy of the market is hosted. Workers who come by can leave an ad on the local Raspberry Pi Geometron server, and anyone coming by looking for construction labor can just scroll through a now much deeper collection of ads and call whoever they need to hire. A market place like this can suddenly go from a dozen general laborers to a construction labor market which includes specialists like plumbers and electricians as well as much larger general contractors just looking to save on marketing costs. A person holding a cardboard sign on a street corner by a giant box home improvement store can now potentially be the broker of an information network on which millions of dollars of commerce flow.

Trash Robot

The Trash Robot is a self-replicating set of things described later in this work. It consists of an open brand combined with a way to create self-replicating symbols which represent in principle anything one can express

with language. We use the Geometron system and the Street Network as a vehicle to distribute Trash Robot.

Trash Robot icon printers form the basis of a symbolic economy. This means that we construct self-replicating physical media which can have symbols representing anything, and we use these symbols to communicate with each other about how to replicate things. In a numerical economy we exchange money for goods or services based on a numerical evaluation of the "value" of those goods or services. In Trash Robot we use the Geometron language to make constructions of pure geometry which can be used to organize our thoughts and discussions as we share with one another self-replicating technology of all kinds.

Trash robot consists of a collection of methods for building the robots and operating existing machines to make all these icons symbols, as well as the open brand of the fashion and accessories and arts and crafts which symbolize the system. This brand consists of googly eyes, rainbows, black cotton flannel, cut felt and rainbow duct tape over cardboard and bamboo. By being a very recognizable brand identity which is generic enough to be impossible to copyright, we make a vehicle for technology which is not property to freely replicate.

Chapter 4

Servers

Servers are machines that store and share documents in the Geometron system. There are three kinds of servers we work with: the Raspberry Pi servers that form the real backbone of the network, globally visible hosted domains used to point people to local Pi servers, and developer servers for editing and sharing new versions of the Geometron software itself.

To spread the Network, the most fundamental form of replication is replicating the local Raspberry Pi server. To do this, we want to either buy or barter for the parts, install the software, and teach someone how to operate it so that it can be sent out onto the Street for public use. Given the choice, we will always barter for the parts. If we can find people who support our cause and have extra technology hardware like old keyboards, screens,

or even Raspberry Pi boards, it will be easiest to take direct donation of hardware on location by an existing server than to deal with purchasing and shipping new hardware.

The elements of the basic Raspberry Pi server are:

- The Raspberry Pi board itself. This is a circuit board about the size of a deck of cards. All of them should work! Older ones might be slower but they should all work. This has all been tested on the Pi 3 and Pi 4. Boards are generally between 40 and 50 dollars, but again if you can barter them that's ideal, as they are often sitting idle in peoples desk drawers.
- micro SD card and SD card reader to write the card.
- USB keyboard. If possible, find the one without the number pad so that it fits more easily in a backpack, this can make a huge difference in portability.
- USB mouse.
- HDMI Display. Not all work, but all can be made to work. Ideally you want a small screen which can run off the same battery as the Pi. If you are setting up in a fixed location you can use any standard modern TV screen, which good both for needing fewer new resources and for visibility. A large

screen display for a server can be a great way to have shared physical social media, where people can read documents on the big screen without needing any device of their own. For small portable ones we recommend buying ones specifically sold for the Pi, and which say that they don't need any special installation of software to work.

- HDMI or HDMI mini cable. There are some tiny Pi displays which don't need this because they connect to the pins on the board. Note also that you will need the HDMI mini cable for the Pi model 4 but the regular HDMI for all other models.

- Lithium ion polymer battery packs. This is only for portable installations, but it is important to have something modular and portable with a lot of power storage capacity. We recommend the TalentCell batteries as being easy to charge, easy to carry, and having both 12 V and USB output. They also sell solar chargers for those batteries, which are useful to have for long stretches of being away from power.

- Wall power. For the Pi model 3 and below this means a wall supply with the same USB micro and for the model 4 it is USB C. You will want a wall supply that can put out 3 amps at 5 volts. Also, if you are running everything of 12 volts you will

already have a wall supply that came with the battery packs listed above.

- Wifi hotspot. This can just be a phone with the hotspot turned on. But it can also be a mobile hotspot from the phone company which has its own wifi and connects to the network.

When you have all the materials to make a server, you will want to start by setting up the Raspberry Pi in the normal way documented on the Raspberry Pi website. Follow the instructions on there to copy the NOOBS operating system onto the SD card. You can also buy SD cards with NOOBS already installed. The Raspberry Pi home page is www.raspberrypi.org. Buying Raspberry Pi stuff can be done in person at some electronics retailers, some maker spaces, and online from numerous retailers, just search and look around. Sunfounder is a great source of compact portable screens for the Pi as well as other Pi things.

Note that part of setting up the Pi is logging onto some kind of wifi network, which is similar to on any other computer system, you click on the wifi icon in the upper right of the screen, select a network and put in the key. If we are using a hot spot we will want to select a simple name and password like using "geometron" for both, and post as widely as possible to potential users what that is so it's easy for them to remember and log

on. In order for this network to be visible, all users must share a wifi network. It is also possible to point a global domain name to a local Pi visibible to the outside world, but that is beyond the scope of this book.

Once the basic operating system is set up(set it up with no password) you will want to install the web server, the PHP language, and the Geometron server. To do this you can follow the instructions at github.com/lafelabs/thing, which is where all the code for this project lives and from which it will all be copied when you install. This copying process copies all the detailed instructions to copy the system, so if you find any instance of the Geometron system on a global or local server you can follow the instructions on there to replicate.

When the Geometron server is installed on the Pi you can interact with it by opening the web browser on the Pi and pointing it to http://localhost. This should now look like any other server in Geometron. This can be used to create, edit and replicate documents of all the formats in the Geometron system, which are documented in the next several chapters of this book, as well as on each instance of the system.

When you set up a new Pi server, you will want to copy the Pi scroll to the home scroll, and there is a link to do that in the default screen. When the Pi has a correct Home Scroll there will be a link to open a page which makes a QR code for the server. You can then scan the QR code on the screen to log on with any mobile device

which is logged onto the same wifi network as the Pi.

The second type of Geometron server is on remote hosted domains. As discussed in earlier chapters, we will buy domain names based on generic places that are not owned by anyone but have a physicality of some kind. For example streets, parks, rivers, neighborhoods, truck stops, or mobile mutual aid stations. To install the Geometron server on a globally hosted domain, just copy the file replicator.php from any existing Geometron server then point a browser to it. This is documented in the README documents.

Finally the developer server is used for local editing of Geometron on a private computer which can then be pushed to public repositories in a host like Github. This is done using the built in web server of the PHP language. If you are using a Mac, PHP is built in and you can run all this from the command line. If you are on a PC you will need to install the Ubuntu machine under Windows 10, install PHP and use that command line. In either case, start a new Github repository, set up whatever you would normally use for development via Github and put the file replicator.php in there. Run it with php replicator.php at the command line. Then while in that directory, run php -S localhost:80 and point a browser on that machine to http://localhost. Now you can operate Geometron as normal.

To edit all the code on the system, use editor.php, which is linked from the README file. To point the next

replicator to your new instance of Geometron, edit the code in php/replicator.txt to point to the /data/dna.txt file of your new instance, and then convert that to php with text2php.php, which is linked from the code editor.

The details of the system should be documented on the system itself and on other linked media, so it is redundant and tedious to delve too deeply into technical details here. This is just here for completeness so that the system is described in broad strokes and so there are pointers to all the bits you need to learn about if you want to move from just using the system to building your own new systems based on it.

These three types of server are the whole system! There is no company selling server space or running a central code base. Each individual server of any kind has the whole system on it. If every system on the planet were deleted in an instant, you could repopulate the entire world with the one copy on the individual Raspberry Pi you are using, or the Github repository you cloned it from or the web page of the local street that connected you to the Network. Github repositories replicate code, hosted domains point to Pi servers, and Pi servers are the medium on which all documents can be created, replicated, and shared freely across the whole rest of the Network. All this can happen with no company, no organization, no centralized code base, no centralized brand or naming convention, no authority, no property, no cash flow, and no presence in any app store. And all of it is

open, clear, self-documenting, and easy to copy.

Now, go forth and multiply! Let us first make a million of these with the Raspberry Pi, then learn to make them on old hardware which we install stripped down Linux systems on, and then finally on Geometron hardware built entirely from trash using the full stack Geometron system described later in this book. Millions of servers can serve hundreds of millions of people. When we scale to using all reclaimed trash for hardware, we can scale to billions of servers, eliminating the personal layer of networking completely, with ubiquitous open and free media shared in physically public spaces. This technology is clearly already possible to build if we choose to do so, and it can clearly be done for free given the very high rate at which the existing consumption-based system is pumping out electronic trash with all the elements of media(screens, batteries, radio transmitters etc).

Chapter 5

Scrolls

Scrolls are the text documents of Geometron. Think of this them as the Microsoft Word of the Geometron ecosystem. While most people are of course familiar with text documents in formats like Word, the form of Scrolls and how they work in our system is quite different and requires a fair amount of explanation.

One of the most important requirements for a system of self-replicating documents to function is that documents be able to be copied to a clipboard from one window and then pasted into another window from the clip board without losing any information. This is not possible in formats like Word, since the system hides all the formatting from you. In Geoemtron, the Scroll format uses raw text with a few additions that allow for things like images, links, headers, and lists to be called using

simple syntax. This is what is called a "markup language", and is the same class of languages as HTML, the "HyperText Markup Language", which is the basis for how all Web documents are formatted.

What we use in Geometron is Markdown, a markup language specifically designed to be as "lightweight" as possible, replacing the cumbersome tags of HTML with simpler and shorter syntax. Markdown was created by Aaron Swartz and John Gruber in 2002-2004 precisely to solve the problem we need solved: the simplest possible way to create a rich text document from plain text. One of the important differences between Markdown and HTML, however, is that if you start with a blank document and just type some text, not knowing anything about Markdown, that will be formatted and readable, without adding any special code at all at the start and end of the document. Markdown is also useful because it is used in several other open technology systems that are useful for us to work with in Geometron, namely the documentation on Github and the text cells of the Jupyter notebook system popular in open source science.

In addition to basic typesetting capability, we use the LaTeX typesetting system as an optional add-on for creating and sharing technical documents which use math. LaTeX is also used to build more complex documents which are in other formats, such as this book, which was written entirely using that system to convert to a .pdf format for printing. This technical document shar-

ing capability is important if we wish to have our network form the basis of a new technological civilization: we will need some very technical documents to make that work. Also, LaTeX is used to typeset the equations in Jupyter notebooks mentioned above, so the ability to work with it is important for integration into existing open science projects.

Since everything in Geometron takes place in a Web Browser, we need a way to translate this raw code that we interact with when we edit and share files and something that looks the way things should look for people to read it. We do this using two open source JavaScript libraries which can be called remotely from a browser. To convert between Markdown and HTML we use Showdown.js, which is published under an MIT license, is easy for programmers to use with simple copy and pasting of code, and widely available. To convert LaTeX code into HTML we use the the MathMax JavaScript library, which is a fantastic resource for making math work in a Web Browser.

In the standard user page on any Geometron system you will start by seeing the home scroll. If you are on a screen which is wider than it is high(like a laptop), there will be a bar on the right side of the screen with a list of scrolls and maps you can click on. On a device which is taller than it is wide, like a smart phone or tablet in their normal orientation, there will be a button in the lower left corner to “show menu” that you will need to

click to see the lists of scrolls and maps. Scroll through the scrolls and click on them to read them.

To edit the scroll you are reading, click the edit icon, which is denoted by a pencil. This will drop you into the editor page, where you can edit the code for the scroll you were just viewing using the Markdown language as mentioned above. To switch which scroll you are editing, click on any of the scroll names that appear next to the edit area and you should instantly switch to one of those. To start a new scroll, enter the name of your new scroll in the "new scroll" input. To view the scroll you are editing at any given time, click the link with the scroll icon, which looks like an ancient scroll with two rollers at each end like a traditional Torah scroll and you'll see that scroll, then when you click the edit icon again you will get back to that scroll. If you click the Home button you get back to the home scroll regardless of what you were editing. The red "x" is the delete scroll page, which we will deal with shortly.

Copy often! We do not "save" in the normal sense in Geometron. Every keystroke edits the permanent file on the server. But all of that can be destroyed in an instant by any user at any time. When we want the documents we create to replicate we simply replicate them: copy all the text from the editor window and paste it into both other backup scrolls on the same system and to other scrolls on other systems, local copies on private machines, or global copies on public pastebins as will be discussed

in detail below.

As stated in the Organic Media chapter, one of the Laws of Geometron is that everything dies: everything must be easy to delete or destroy. We delete scrolls by clicking on the red "x" to get to the delete scroll page. This page is pretty self-explanatory. Delete things!!! Everything is supposed to replicate, so get in the habit of destroying information as you go, and replicating the good stuff rather than holding onto a stale system of information which slowly rots over time as is the case on private hard drive in the property-based systems.

The first thing to understand when formatting scrolls is how you make a paragraph break. To do this you want to hit return not once but twice at the end of each paragraph. It is easy to accidentally only hit return once and that will run paragraphs together without any break when they are displayed.

The next most important thing to know is how to make a heading. This is done with the number symbol "". The more number symbols, the *smaller* the heading. Thus one number symbol is the largest size, and when you want a sub-heading you use a double number sign and so on. Headers will be centered when displayed.

To emphasize a bit of text, put it between asterisks. To make a bullet list of items, put a dash before each item, with a space after the dash. To make a list with numbers, do the same thing as with the dashes but with numbers followed by a period. Again, put a space after

the period to tell the system this is a list. Be sure to put something after each number or bullet in the list, or the last one will end up looking weird.

Dealing with images is a little bit different than what you might be used to on other non web-based systems. As with the rest of the Web, images are put into documents as links to a location where that images is stored. That location can be locally on the Geometron server you are working on but ideally it will be on a globally visible web address. The simplest way to do this is to find an image of the thing you want an image of that is already on the free open Web like on Wikipedia using a search engine, and then right click on that image and copy the image address to the clipboard of whatever machine you are using, then paste it as the address of the image. Images are put into Scrolls using the somewhat confusing Markdown syntax which is an exclamation mark, two square brackets, and then parentheses containing the link. Yes, it's weird, but it's not hard once you get the hang of it, you can just copy paste from existing images and drop you new image address in.

We can also upload images to each Raspberry Pi server for documents which we are just working with locally, and this is useful when you want a locally specific image which is of no use outside that server like a photograph of a physically local place near the server. This is done using the Local Image Feed, which is documented in the Feeds chapter. The other main work flow we will use

in Geometron is taking original photographs with mobile devices, uploading those photographs to publicly visible locations, then copying the address of those images and pasting it into scrolls. One way to do this is with the website Imgur.com, which allows people to upload images both with an app on a phone and over the Web on a browser. You do not need an account to do this in a browser, but will need one to use the mobile apps, but it is free to use. Set the privacy of the image to "hidden" and then always copy and paste that url somewhere. Also note that there are several ways to share a url of an image, and you need the one that links to the raw image, which should end with some kind of image file format like .jpg or .png. To get this the most efficient thing is to right click and copy just as with public images we find in the Web at large(copy image address or copy image link depending on the system you are on).

The last but perhaps most important part of Markdown syntax we need for making and editing Geometron Scrolls is links. A link is created with square brackets containing the text of the link followed immediately by parenthesis containing the link address. For example we might make a link to the Geometron home page which is at index.html with [home](index.html). You can link to anyplace on the Web by navigating a browser to the page you want to link to and copying the address from the address bar and then pasting it into the parenthesis of your link. Be sure to always have no space between the brack-

ets which contain the text and parenthesis which contain the address or the link will break. Also, links can be images. You can put an image inside the square brackets and that image will appear as a link to whatever is in the parentheses like any other link.

Geometron scrolls allow for local links to scrolls and maps in the Geometron system in addition to the global web links described above. All scrolls created on any given Geomtron server are stored in the "scrolls" directory. If you put a link to "scrolls/scrollname" the Scroll software will detect that as a scroll link and clicking on it will load that scroll into the scroll reader rather than linking to the raw scroll file. Maps, which we will discuss in a later chapter, are linked to in the same way with links that start with "maps/". If you want to load a scroll or map stored in a remote location into the scroll reader you can use the scroll() or map() commands in the link, with the destination inside the parentheses, which are in turn inside the larger parentheses of the link. For example, you might put "[link text](scroll(scroll location link))".

Another important aspect of linking scrolls is the user of the active user page called user.php. PHP is the universal web language we use for any kind of active program which engages the files on the Geometron server. PHP allows users to send information to the program which loads the web page they see by adding bits of information to the web address the browser points to. User.php

takes two inputs, “map” and “scroll”. To make a link to a scroll called “trees”, we would construct a link as follows: “user.php?scroll=scrolls/trees”.

Similar to user.php is copy.php, a universal copy program which can copy a file from anywhere on the Web(really anywhere) to anywhere on the Geometron server you are working with. Copy.php takes two inputs, “from” and “to”. When a PHP script is fed inputs, the second one uses an ampersand and the first uses a question mark. The order does not matter, but by convention we start with “from”. So to copy the Scroll “tree” to “tree2”, we would create a link to copy.php?from=scrolls/treeto=scrolls/tree2.

It is worth mentioning that you can always use HTML inside a Scroll and that will work, if you know HTML you can use this to add interesting things. Learning HTML is a useful thing to do, and dropping little bits of HTML into a scroll is a good way to do that when you’re starting out. Also, you can use the code editor at editor.php to edit the main file index.html and change the default style of scrolls if you want to. As with HTML I won’t go into detail on this, but I recommend that people learn CSS if they are interested in getting more into development and then it is pretty self-explanatory how to change the settings at the end of index.html.

Another important part of the work flow is the use of paste bins. You can copy a scroll into a paste bin at pastebin.com without getting an account or logging in. Set the paste exposure to “unlisted”. You can also set

the paste to expire after a certain time, which can be useful. When you paste a scroll into a pastebin, you will want to get the address of the raw file, by clicking on the link which says “raw”. This is important! If you link to the first page you see after posting a pastebin the use of that pastebin in Geometron will cause problems. The address should actually have the word “raw” in it if it is correct.

The raw pastebin address of any scroll on the Web can be loaded on a local Raspberry Pi based Geometron server by putting the address into the text input that appears directly above the list of scroll names and the edit icon in the scroll list. Remote scrolls can be copied locally and edited for further sharing of new versions using the program pastescroll.html, which should be linked somewhere on the home scroll of your Server. This program lets you enter the pastebin address of a scroll, which can also be the address of a scroll stored on a globally hosted Geometron server or another Raspberry Pi elsewhere on the local wifi network, as well as the name of the new local scroll you want to copy that global scroll to. When both of these inputs are filled, an edit link will appear on the screen, and when you click on that the system will copy the remote scroll to the local file and you will be editing it. You can immediately edit it, then click on the scroll icon to see it, then back to the edit screen and so on, all as a local version of that scroll. Having edited that scroll to make it better, you can then copy and paste

it from the scroll editor whatever Pi server you are on back to another publicly visible pastebin which you can then send the address of to another person anywhere in the world. They can then repeat this whole process on their local system, creating a copy of your new version which they edit and improve upon and then send the new link back to you and so on. This workflow enables global collaboration very quickly between people all over the world, all working on physically local networks that are not visible to the outside world, in particular search engines.

Note that public pastebins from Pastebin.com have censorship which can be somewhat random to predict, and they will prevent you from posting scrolls that have certain political topics. To avoid censorship filters, you can encode the document in its ASCII/UNICODE by using the program textconvert.html. This will let you convert any text file to a list of base 16 numbers separated by commas which can then be transformed back manually. You will only occasionally need to do this, and it is somewhat annoying, but it's useful to have as a tool. It also lets you deal with non-English characters which get turned into UNICODE, making it easy to copy and paste in English-only computer systems while preserving the information in non-English text documents. This should also be linked from one of the main scrolls in every system.

Pastebin can also be used to copy to global servers.

This is done with pastebin.html. This is the fastest way to clone a scroll to the home scroll of a globally hosted server. This will not work the first try, usually, you'll need to reload a bunch of times at random for the copy to actually work, for unknown reasons. In general, using Geometron on paid hosting platforms is spotty and you just have to try a few times to make it work. Eventually, when Geometron has its own servers hosting domains we can fix this, but for now just poke at it and keep trying and it will work.

It is difficult to overstate the versatility of a text document with formatting like this. The number of potential uses for a freely shared system of text documents without property vastly exceeds what I can think of in a book like this, but I'll say a few words here about what I have in mind for it.

The primary use of the scrolls which was the initial motivation for creating this system is for replicators, as with all elements of Geoemtron. This whole system is built on the idea that the most fundamental thing we can do in media is communicate how to copy a thing built from trash and other found materials in our environment. The basic workflow here is to photograph every part of the assembly process, link to all the links of materials we might need to buy(yes, we still have to buy things for now), describe steps, and link to scrolls and maps of related things.

When Geometron scales up, this will be a vast net-

work of interlinked documents which connect people with things. Most people will not make most things, just as most people do not make most things in today's consumer society. But the things we use will always be *linked* to their means of replication. This means the physical thing contains the information in some form to *find* someone who can replicate that thing, and that that person will be able to *find* all the materials and learn all the skills. Replication means that this universe of linked documents has to be fractal in the sense that when someone chooses to dive deeply into a thing they learn not just a set of instructions but are linked to all the deeper knowledge required to attain expertise in the technical disciplines required for replication. This means that for example every electrical device should have links from document to document which lead an interested party into a whole curriculum of learning about electronics theory and practice to become competent as an electrical designer and builder.

Scrolls can also play a role in *finding* the free things we create and replicate. This is an important part of any economy: making a web resource that tells a user how to find a thing in a place. This is in some sense the majority of what the consumer media does with their constant barrage of advertising: tell a person how to find a thing in a place. We can do this much more effectively by pointing people to streams of things which are made on location so we are not just pointing to free things but

to a free *stream* of things.

Also, a scroll is just a document. So it can be any document! You can write an article about...anything! Write news stories about the street corner where the Pi server is located. Write editorials about the current pastry selection at the local coffee shop where another server is. Write manifestos. Write the Constitution for the new system of local governance which applies only to people on your local wifi network. Write guides to the edible plants to be found within walking distance of your local Geometron server. Write your life story. Write an ad for whatever it is you do for money to survive our current money-based system. Create a long list of links to all your favorite obscure web pages. Create a recipe with details on how to find the ingredients in your local grocery store.

The use of Scrolls for sharing information locally can be incredibly powerful. We are building local media complete with news articles, history, politics and so on that are physically local to an area limited in size to what you can hear, see, or connect to over wifi. This can be used to organize workplaces against the central bosses who control the workplace from thousands of miles away, in everything from an Amazon warehouse to the drivers in a rider pickup area by an airport. We can use this platform to bring out a sense of place that has been lost in our current civilization and to share that place with people in a deep and complex way. Whole epic histo-

ries can be written of a street corner under a bridge, of how that bridge was built, of who has passed through there, of the bus routes that serve it, of the origins of trash that turns up there, of the destination of the storm sewer that drains water from it. Myths can be written, whole new religions built up, all around objects found in our immediate environment.

Here we recall the Laws of Geometron: everything is physical, everything is fractal, everything is recursive. We are building a network of documents which refer to physically local places where that physicality matters. We also make this fractal, given how much information can be on a server(hundreds of thousands of scrolls can easily be on a single Pi server), so that we can have media that fractally zooms in from a street corner to a trash can to just the plastic bottles that go through that can. And it is recursive, in that our documents are constantly referring back to one another. No document stands alone, they are all always pointing to other documents, just as is the case on the Open Web, but in a physically localized way and without any users, just free un-owned documents.

Physically local media can create value which people are willing to pay or barter for in the existing system in exactly the same way that consumer media creates value. A media platform serving a strip mall can provide interesting and useful content to the workers and customers and passerby, but because it can stimulate commerce, it

will also create value for the business owners who will then have an incentive to contribute by barter to support the Operators of the network, who can in turn provide custom scrolls and links to them promoting whatever those business owners are doing.

This is also true for the informal economies of local places. The ability to have a mobile and compact media platform that can represent a place but not be obvious or permanent allows vast amounts of commerce to happen in a physical place without being visible to the rest of the Internet. In some sense this is a kind of "dark web" even though it is totally free and open and unencrypted. Having no property or usernames or passwords or cryptography means there are no names of people of any kind attached to the documents. This creates a total freedom to exchange goods and services in a physical locality with a physical trust model based on actually talking face to face with people and directly exchanging things and media.

To see how this can have a high impact, we examine Craigslist and why it's awful now. Craigslist is now all scams and spam for numerous reasons. Essentially, it got too big. It was unable to really become fractal, and there is no local control. Because it is centralized, even though they have separate boards for cities and for areas, those areas are still huge, and they are just theoretical: anyone in the world can see them and post, even though they're far away. Also, an area might have millions of

people in it and cover tens of miles. Furthermore, central control means it is not tenable to have human operators involved with document creation, so spammers can spam relentlessly until everything is ruined. Even though posts are quasi anonymous, they still all have users, passwords, and control. If bad things are posted, the vast majority of users are not empowered to delete them, so if a whole board is ruined by a few spammers, no one can fix it and it stays ruined. In an open document model, a spam-filled board will get cleared out instantly and if it keeps getting spammed the whole board can be permanently destroyed, and all useful documents cloned elsewhere. Craigslist wars to "deal locally" for safety but again they mean this on the scale of a vast area with millions of people in it. Our model is more line-of-sight: you deal with people you can see, in a place where people you trust can see both of you, out on the Street Network.

Finally we must briefly discuss the workflow for technical documents which use math typeset in the LaTeX system and how to use that to create documents like this book. The actual details of this typesetting system are not covered here, as that is a very deep subject. There are numerous print books and online guides to learning it. The very short version is that you use dollars signs to enable math mode, and lots of back slashes and twiddle brackets and keywords to create arbitrary math formatting. And it all works in all web browsers! This is thanks to the magic of the MathJax JavaScript library

mentioned above.

Math scrolls can be viewed along with math maps using mathuser.php, which is used the same way as user.php listed above. Editing is still the same basic method. Tex files in the home directory of your server can be edited using texeditor.html, on which this book was mostly written.

Chapter 6

Feeds

A Feed is an ordered list of pieces of information. This structure of information, in which a set of media elements like images or small bits of text are related to each other by order, which a person can scroll through, is the basis of most social media today. What we are doing is so radically different from anything you have ever encountered, however, that it is worth taking some time here to explore how exactly these feeds work in consumer media and how differently they work when we move to Geometron.

In a consumer-driven social media system, everyone involved is either a user, a worker for the company who owns the platform, or an advertiser. In order to interact with the system, a user is required to have a unique identifier based on their user name and password. Every single thing they do, from how fast they scroll to what

they click on to every post they make is tracked and analyzed by the employees of the company. Advertisers then pay the company for the ability to spy on and manipulate those users into consuming more, and it is the job of the company employees to do that as efficiently as possible.

The flow of information in these consumer systems completely structured around these separate user accounts everyone on the consumer side is forced to use. Even in supposedly anonymous forums, posts are tied to IP addresses and can't be edited or deleted by other "users". That is to say, in the consumer driven systems, even without names the basic paradigm of the individual user is dominant.

Each user creates a stream of information which is "owned" by them, which is "their" feed. When each user is on the system, however, they see another feed, also called "their" feed, which is the sequence of information the company chooses to show them. This is a mixture of elements from other users' feeds and all the media being forced on them against their will by the advertisers who are paying for the network. In order for all this to work, the company has to give them just enough elements in their feed that they keep staring at the screen, while feeding them as many corporate manipulation messages as they can get away with to maximize shareholder value and profits. This has to violate the users' consent or it doesn't work–the whole system has to involve a central power who can control what everyone sees or

people would simply turn off the feeds of the predatory manipulators and the system would collapse from lack of financial support. Also, the existence of "users" is needed in the current system again to maximize profits, because that is what allows the manipulators to target based on detailed profiles of who clicks on what and more importantly who will consume what.

Users are encouraged to have things "go viral" but always within a controlled environment where the company can shut it down and censor it and where each copy is linked to a user so that again the users' behavior can be carefully tracked in order better to manipulate it later.

Geometron has feeds as well. We have feeds of images, feeds of text, feeds of symbols, feeds of icons to be printed by robots, feeds of hyperlinks, and various feeds that are complex combinations of these. Our system of creating feeds is so simple that anyone with a very basic command of web development in HTML and JavaScript can create their own feed applications within an hour or two with copying and pasting of existing apps. In this chapter we will examine how our feeds work and how they differ from consumer feeds and then talk about the existing feeds we use in this system, largely to connect together other parts of the system. These feeds are potentially very powerful, however, but it remains to be seen exactly how they get used. Try stuff!

In Geometron, we do things so differently that it can take some thought to to see just *how* different this net-

work is. We now recall some of the relevant Laws of Geometron and discuss how they impact what we do on this system.

No property. Again, while this sounds radical, it is in fact much more radical than it sounds. There are no users on this system. No employees, no company. No "core developer team". No "marketing customers". There is no user data because there are no users. There are no passwords, there are no logins. The servers themselves are non-property objects which exist to be shared freely from place to place. In our system, feeds are just documents. They are always in a pure text format which can be interpreted by the browser and turned into something readable.

Everything is replicates. A feed can always be copied and pasted in pure text(using the JSON format). It can be stored in a public pastebin just like scrolls, or stored in a text message or email or embedded in a scroll. A feed can be copied instantly from one server to another, spreading the whole feed as an object, without any regard for where each element of the feed came from. Everything we create is designed to replicate.

Everything dies. Every element of every feed can be deleted instantly by anyone interacting with that feed at any time. This is how we create a system where replication only happens with the consent of the people.

Everything evolves. Every feed can be edited by anyone at any time. There are no "permanent" files. To

share, read, or copy is to be able to edit.

Everything is physical. The structure described here might sound unimaginably chaotic compared to existing systems where all information is controlled by a central authority and by private user accounts under some type of algorithm. But that is because we are used to global networks, where distance and geography don't matter. Our system is physical, centered on the community in direct proximity to the physical server. This means we are only trying to organize information in a way which is useful to the community physically surrounding this server. This makes the information sorting and organizing vastly simpler than on a global network. This limitation in the who and where parts of our system is important partly because it is what makes our model work in which everyone can edit anything at any time. This only works if people don't edit the same exact piece of information at the same exact time. Because we are all effectively in the same physical room, but are assumed to be wandering in and out, this is a dynamic which can be built up socially in an organic way.

Everything is fractal. Again, this is how we can make sense of the potentially overwhelming complexity of a feed without users. As soon as a feed becomes too complicated, we just fork it into multiple feeds based on sub-topics. Part of the chaos we are managing here is the potential collision of edits referenced above. Again, this can be resolved with a fork. The program fork.html

will create any number of sub-pages below the top level, where as we add more fractal divisions we can avoid edit collisions.

No money. We do not have "customers" who can simply buy their way into forcing people to see their information. While network Operators might take donations for teaching people the system, because everything people don't like is deleted, there is no way to enforce the contract structure used in consumer society to pay for advertisements. Money plays a role in our system initially as we all need to survive somehow, but is not the backbone of the system like on a consumer system. Also, consumer systems require constant money flowing in in order to keep the server farms running and all their employees doing work they hate to manipulate people. Our servers are passed around through a community in ways of direct use to that community, so no global cash flow is required to keep them running in the long run.

While I am trying to keep this discussion non-technical, it is necessary to say something about the format used for Feeds in Geometron. Geometron feeds come in two varieties: a directory with files in it, and a JSON file. JSON is a format which stands for JavaScript Object Notation, and which is a very simple and universally recognized text-based format to organize information. Any of the hundreds of programming languages in common use today will have already built in routines to handle JSON, and any programmer on any system should already be

familiar with it. Like Markdown for the Scrolls, we select JSON because it is easy to copy and paste, and is as "lightweight" as possible, meaning we need very little added information in the form of weird looking symbols and so on to store things.

The most basic type of JSON structure we use is the array, which is just a collection of things separated by commas, inside square brackets. The other main thing that exists in JSON is the "object", which is a collection of pairs where one element of the pair is a piece of information and the other is a name for that information. This idea of creating abstract objects which map names of things to things and organize them in this way is part of the "object oriented" idea which is the basis of most modern computer systems. Never be afraid to edit and read the raw JSON! You don't have to interact with it but you should not be afraid of it. If you destroy a JSON file, the file you started with can just be replicated again and again to avoid the fault.

The first feed we will discuss here is Chaos Feed, which is perhaps the most basic, and is really just designed to show the concept. Chaos Feed can be found along with the other feeds from the Feed Scroll which should be linked from your Home Scroll on whatever Geometron system you are using. To post, type and hit return. That's all. To delete, hit one of the red "X"'s. If you are using a keyboard, the up and down arrows scroll through the existing elements. To clear and start

over, click the button in the upper right. To reload click the icon in the upper left. To see and copy the current feed, navigate to data/chaosfeed.txt on any Geometron system and you will see the JSON for that feed. To copy a feed from a remote pastebin, use copy.php as with the scrolls described in the previous chapter. To edit a feed or to edit the application chaosfeed.html, use the code editor editor.php. From editor.php you can edit any code on the system, including both the file chaosfeed.html and data/chaosfeed.txt. While most people will probably not want to edit this code, it is simple to edit for people who know basic web development, and so you can ask around and find people who like to do that to evolve the system. I do not know how people will use this. Try stuff!

Another feed almost identical to Chaos Feed but with different applications is urlfeed.html. This is the same interface: you put something in the input and hit return and it posts. But this is for links. Each entry turns into a live HTML hyperlink which actually links to whatever the link points to. To use this, find a link either locally on your system or globally and just copy and paste it in and hit return. Edit, delete and share just as in Chaos Feed. This feed unleashes the power of Hypertext, the central technology of the World Wide Web. Connect from your humble local feed to the whole world! Connect your system together. Connect everything.

Yet another feed of almost the same format is the Global Image Feed, or globalimagefeed.html on your server.

In this, you again just paste links in and they post, but in this case you are pasting image links which will then appear in the Feed. Again, these can then be deleted. Click on any image to see the url for that image appear in the text area. This Feed links up with the alignment system for tracing Icons which will be dealt with later in this book. But it can be used for anything, including just sharing random images from around the Web or use as inputs for any other Geomtron application. We will deal later with a number of such applications which can call on this feed and use it to load useful images to do things with. Once again, delete anything at any time you want by clicking a red "X".

I said above that there are two types of Feed, and we now turn our attention to the second one: files in a directory. We use this for images uploaded to a server as well as symbols created on a server. Images can be uploaded to a Raspberry Pi Geometron Server using localimagefeed.html. This is also used as part of the workflow in several other part of the Geometron system. Note that there is a maximum size on images, so it can be useful to screen shot images on a mobile device, then crop the screen shot to get something closer to 1 megabyte or smaller before uploading. The maximum is about 2 megabytes. This is to keep us from having exploding sizes of data based on very high density images. This is a key feature of our physically local social media system! We can use this to photograph objects and places in di-

rect proximity to the server and upload them and use them in order to build media which directly represents our physical environment. To upload, click “choose file” first, and select a file to upload from your device, then when that file is selected use “upload image” to upload the image. To delete, use the red “X”. All you are seeing here is a list of the files in the directory uploadimages/, which you can manually examine and interact with as well. This can be an easy way to transmit images and memes in a peer to peer way, as a meme or image can be uploaded from a mobile device to a local server, which another user can then download to their mobile device, allowing for rapid peer to peer transmission locally over the wifi network. As with the other Feeds here, this is deceptively simple, and can by itself form the basis of a whole new type of local social media networking.

The real heart of the Geometron system is the geometric programming language presented later in this book. Among other things this creates image files which are symbols in either the vector graphics format .svg or the bitmap format .png. These are all stored in the directory symbolfeed/. They are viewed and deleted using the feed program symbolfeed.html. This whole system will be explained in detail later in this book, but for now it is just important to know it is there: this is geometric social media, in which you create geometry and share it freely. Again this is deceptively simple, and can form the basis of a very wide range of technological activities

which will be described later(along with the Icon Feed, stored at iconfeed.html).

The Feeds described here are programs which run on a Geoemtron system. We must also consider Feeds in a more abstract sense which we will use to build up our whole system. Ultimately the whole system will be based on the Trash Feed, the vast global feed from mine to landfill which we will redirect toward our new civilization. This feed is made up of many small local feeds, which have a fractal structure. For example, a trash bin near a kiosk in a park selling bottled water will be a constant source of plastic bottles. This type of feed is part of our system in a general sense even though it is not software. But it is not totally separated from our software either. Our system includes robotics which can print icons into plastic bottle caps with a heated tool, and those icons are shared on the Icon Feed, so there is a crossover between the physical feeds of trash-sourced objects and this less tangible software system.

Chapter 7

Maps

Maps are a from of document in which a set of images, words, and links are arranged geometrically on the screen. Just as the Geometron Scroll can be thought of as a replacement for Microsoft Word, the Geometron Map can be thought of as a replacement for Microsoft PowerPoint and Keynote. But it is also a way to make and share memes, to annotate geographic maps, annotate photographs of objects and really do any kind of communication where the relative geometry of objects matters.

It is worth once again examining the consumer civilization's version of this in more detail to understand what we are trying to do differently here. PowerPoint is the language used in the replication of things in today's world. When a new company is born, the founders

use a PowerPoint slide deck to sell that company to investors. Every government applied science project starts with PowerPoint slides(often just a single one in the infamous "quad chart" format). In many ways our whole civilization runs on PowerPoint today. It is hard to imagine any project being funded in the world today, be it government, corporate, or non-profit, without a series of these simple graphic constructions of text and images and vector graphics arranged in a geometric order of some kind. And of course this format is the basis of the meme, this bizarre new type of thing that spreads freely across the web, generally with copy and pasted bitmaps.

We also need this format for replication of technology in our trash-based civilization as well as for all kinds of other communication. However, we need it to fit in with the values and laws of Geometron, and that requires that we rewrite the whole thing from scratch.

As with every Geometron document format, this format has to be something human readable using plain text so that each individual Map can be pasted into a text message, email, or pastebin for freely sharing across the Web without any intermediary. We do this using the same language as the Feeds discussed in the previous section: JSON(JavaScript Object Notation). A Map is an array of objects, each of which has a collection of properties. The array is denoted by a pair of square brackets, and each element is separated by a comma. Each element is inside a pair of twiddle brackets, and

consists of pairs of names of properties and values of those properties. Each element has a position, a size, an angle, a text value(optional), a link(optional), an image(optional), and information on whether it is a global link or a local link inside the Geometron system.

You don't need to understand what all that means to use the system! The technical description is just there for reference. Maps are edited and drawn using a JavaScript library called mapfactory.js, which is replicated with each instance of Geometron. If you are a web developer feel free to go read the source code now to get an idea of how it works(there is not much to it).

By default, maps are displayed in a square area on your screen, and when you load the Geometron home screen that square will be either the left or top part of your screen depending on if your screen is wider than tall(landscape) or taller than wide(portrait). To load a map, just look at the list of maps in the menu either to the right of the screen(for landscape) or in the popup menu you click to open with the button and look at the right side list and click on any map. That will load it. Now you can try clicking around on all the maps on your system, as well as navigating from map to map using internal links inside the maps(some have this some don't).

Maps are used to amuse, to tell stories, to make points, to denote where things are, to point out where a part of an object is located, and to connect web pages to one another. Maps are much more powerful than the Power-

Point slides they replace for several reasons. The ability to have both global and local links to other documents makes them fully integrated into a global, ever-evolving network in a way that makes them much more rich and complex. They are, like all Geometron documents, not owned by anyone. Each individual map is a free document, which can be replicated, edited, deleted, shared an infinite number of times instantly by anyone on the system. This creates a richness of information which is impossible with a dead file format like PowerPoint.

Maps are a great way to build social media around a physical place. When community forms around a local server in a local place, the local media should have photographs of the objects in the environment. Unlike a consumer network made up of "users", we have people in a community who share documents openly and freely. Photographs of people in our community can go here, and we can build up a media pool that includes us all, but as a shared community of documents, not as a database of "users". It is also an important way of denoting the exact physical location of things if we are to build up complex systems of industrial production from trash in our immediate environment.

Maps are also another way to rapidly create web pages in the Geomtron system, as the main home page can be set up to point to them directly by changing one line of code in index.html, by replacing loadscroll() with loadmap() and the name of a map. All maps are stored in

the directory maps/ on each Geometron Server, and you can see all the maps by looking there in a browser, then click on them to see and copy the raw text of the map. Just as we have a home scroll stored in scrolls/home, there is a home map stored in maps/home.

Maps are edited with the map editor, which is at mapeditor.html on your local Geometron Server. The Map Editor looks a little bit different on a portrait versus landscape screen. In either case, the screen is divided into different areas which have links and buttons to do different things. The main map window should look the same as when you are in the passive map reading mode: a square either in the left or top of the screen. The element edit window contains icons indicating various actions, including select next/previous element, move element up/down in list, delete element, create new element, save map, delete image, delete link, and selectors to select which type of object you can select from to replace in the element you are editing using the textfeed described below. It is a good exercise to just try playing with these, deleting all elements, then making a fresh one and playing with that. Whatever element is selected is moved by dragging around on the map display screen, which can be done either with drag-click with a mouse or touch-drag on a touchscreen. Elements are resized or rotated using the zoom/rotate box, which contains slider bars for both scale and rotate as well as buttons for both zoom and rotate which only appear in landscape mode.

There is another window which lists all the maps, and you can click on those to select which one is being edited. That window also contains links to the home screen, a link to the specific map you are editing, and a link to the map delete program. The Map destroyer is exactly the same as the Scroll destroyer. It is a list of all maps on the Server, with a button to delete each map instantly. There is no undo. To click is to destroy. If a map is worth saving it is worth copying and sharing and if it is copied, deleting it costs nothing, so we make it very easy to delete. The destroyer page has a link back to the map editor.

The window with these links also has a text input where you can input the name of a new map. Try entering the name of a new map, and you will see a blank screen. To add an element to that map, click the icon with the plus sign. To delete it click the red X. Add another one, and move it to the top then bottom, move them around. To add an image to a map element, click on one of the images in the window with the images. Then you can click the icon with the red X through the image symbol(mountains and a sun icon) to remove the image and go back to just text.

That scroll of images can also display a scroll of links or text. All of these are taken from a combination of feeds, but primarily the Text Feed, at textfeed.html. This is linked via an icon with squares separated by a triangle. The local image feed is also displayed and you can add

to that using links on here as well with “choose file” and “upload image”. The blue link icon will make that scroll a list of links, and the ABC icon will make it a list of text elements. You can also edit all these manually using the table of inputs.

Unlike the Scrolls, maps are not instantly updated as you edit. They have to be saved with the save button. Whenever you save a map, the text based representation of the map in the JSON format is placed in the text area below the control buttons. If someone sends you a link to a raw map, you can go copy the contents of that map to the clipboard of whatever device you are using and then paste it into that text area and click the “import” button to import it into the current map. You can then save it, and the current map will be replaced by the imported map, destroying the existing map. You can also hit the “reset” button to clear out the map and start over with a fresh one. Try making a new map, then selecting all the text in the text area, and pasting it to a public pastebin, then sharing that link with another user. They can then paste it into the text area of their server to make a new map which is a copy of your map, edit it, and send it along to the next person and so on.

The button at the bottom of the table of inputs to edit the current map element after maplinkode sets whether that mode is true or false. If it is false, and there is a link from the map element it will be a regular hyperlink like any other link on the Web. But if it is set to

true, it can point to either any map or any scroll on the web. If you paste a map in a pastebin and then get the link to the raw version of that pastebin, then put that into the "link" field in the table, you can click on that link and it will load that remote map from anywhere on the Web. This can be incredibly powerful, and can create entire networks of complex interconnected maps, all via anonymous pastebins, all referencing other images around the Web without storing any information on the local server or linking to any specific server or user(there are no users).

The button marked "height mode" changes the relative height of the element rather than the height and width together, which can be useful for sizing text elements exactly how we want. This does not do anything when the element has an image, however as the element will then automatically size around the aspect ratio of the image.

While you can enter the address of an image, the value of a text area, or a link value into the text fields, this is unwieldy and particularly annoying on mobile. Therefore the main way to insert one of these types of information into a map element is via clicking on it in the scroll of images, text or link. This is switched between these three by clicking the appropriate icon in the control button table(there is a link, text, and image icon). The image scroll is listing images from the local and global image feeds discussed in the last chapter, but it also has

a special feed just for feeding information into the Maps. This is the Text Feed, located at textfeed.html. If you go to this page, you can input text, links, and images, and then go back to the map editor and use what you entered there. This is important for working on mobile devices where the manual input in the table is very awkward, but it is also important for another reason: dealing with symbols, icons, and other creations of the Geometron geometric programming language.

Creating symbols with Geoemtron will take up much of the rest of this book, but while it works with vector graphics, we need a fast and simple way to embed them in maps. To do this we use the so-called "base 64 encoding", which allows bitmap images to be inserted into files without reference to an external image file. Symbols made with Geometron can be converted into base 64 encoding using the link from the textfeed page which says "png code", and that will let you choose a symbol to convert and save into the textfeed, along with the ability to select how large that image should be in pixels. As with all Feeds in Geometron, you can delete any element of any of the sub-feeds at any time by clicking the big red X. Textfeed also links to some other applications which use it, such as the Duality and Poetry Engine apps, which you can explore as well.

Just as with Scrolls, we need to have the ability to use mathematical typesetting with maps. While most people may not need to typeset math, it is useful to know how

to point mathematically minded people to these tools so that they can use them if they want. The starting point for this is to find a map or scroll that references math, and they should link to the relevant pages. The relevant pages are mathmapeditor.html, mathmapeditor.php, and mathuser.php. Combining math typesetting with geometric symbols quickly and freely and sharing them can build a whole new method for rapid communication of mathematical ideas. Setting up this system in physical proximity to a location with a community of mathematicians can be very powerful and is highly recommended!

Now that you know how to make, edit, delete, and share Geometron Maps I want to say a few more words about exactly how they can be used in helping with the workflow of the overall system. One of the most fundamental tasks we need to do to build a physical network on the Street is to help people to find things in the physical world quickly. Global map servers exist with maps of the whole world, which everyone can get to on any Internet enabled device. We can use these maps to find where we are, then screen shot the map, save it, upload it to a server, and build Geometron Maps which then annotate that map with symbols, links, and words to show exactly where a Server is, or where physical resources are for our trash-based industrial production. This can also be very useful for ad hoc mapping in an area with specific local maps, such as near the exit of a subway. Subway station exits often have a very specific localized

map centered on the station with clear markings of relevant landmarks. If you photograph that, upload that photograph to a public image server, link to it, and annotate it, you can build a whole system of networked maps from that. Don't forget links! Maps can have annotations which are maplinks, linking to other maps or to scrolls with any kind of document you might want to associate with a place. A physical place can have a rich fractal structure of documents built around it, forming a kind of physical hypertext: a document in which we and everyone around us are physically immersed.

When we want to present a graphical story like the "pitch decks" which use PowerPoint in the consumer media, we can treat each Map as a slide, and then link them together with links on the edge of the screen going from previous to next and so on. This can copy the functionality of slide decks exactly, but with much more richness of content since they can also link to scrolls, do not have to be linear, and are not restricted to loading local maps. Rather than a "deck" being a dead document sitting on a private hard drive, linked Map sets of Geometron join a global swarm of potentially trillions of linked documents all replicating, evolving, and being destroyed in an ever shifting living informational universe.

When we build technology, and attempt to document that technology to aid in replication, we need to be able to label parts of the technology, and then link to images of sub-systems which then also have annotations. Maps

allow us to do this in an infinitely fractal way, zooming in on things with more and more detail, and then forking off to related things which are linked in much the way that Wikipedia forms a vast fractal network of information.

The combination of geometric programming discussed in the next sections with Map creation allows for exploration of symmetries of the world around us which is unlike any other tool in its direct connection to the symmetries and scales of the Universe. This will be explored much more as we go along, but I encourage the reader to keep returning to the Map editor and making maps with all the other parts as we go along and learn the Geometron System.

Beyond these examples, there really are no limits to how much can be done with this format. It is lightweight, simple, easy to copy, and fundamentally more powerful than the predatory consumer systems like PowerPoint. The framework is so simple that interested programmers can easily rewrite the whole thing from scratch in many systems, making all sorts of applications with it. As with the JSON format itself, or HTML, the very simplicity of the specification is what gives it its power. A blank PowerPoint document with literally no content can still be well over a megabyte, and you can't read it or edit it without paying Microsoft rent for the "right" to run code already on your hard drive which they agree not to break if you keep paying them protection money. A Geometron Map can be as small as a few bytes, is human readable,

can be shared by text message or email with direct copy and paste, can replicate itself freely across the network, and can be opened by an application that is itself only a few kilobytes in size. Build on this system! Make it yours! Share it and help it grow!

Chapter 8

Symbols

Symbols are geometry with meaning. The symbol is perhaps the most general idea that exists in human thought, since everything we do is mediated through the use of some type of symbol. When we speak of symbols in Geometron we mean *any* geometric construction which has meaning to people. This includes not written language like text but constructions like the layout of a microchip or the design of a building. It also includes the way we control machines, how we program them build automation. Geometron represents a new framework for working with all these kinds of symbols.

We live in a civilization today totally dominated by numbers and by the people who work with numbers. The machines we currently use to communicate are built by people who believe that the most fundamental task such

machines can do is to work with numbers. They call all these machines "computers" and have a whole theoretical framework for understanding how to build them using the ideas of arithmetic. This works. But it is extremely inefficient and distracts from the real purpose of such machines. Do these machines do arithmetic? Of course. They also keep very accurate time, does that make them clocks? They produce heat, does that make them heaters? No. Just as we don't call a light bulb a "screw" just because it screws into a socket, it does not make sense to let the idea of the arithmetic engine dominate in a technology the sole purpose of which is to communicate with other people using symbols. At some intuitive level, most people understand this, it is why the smart phone is primarily called a "phone" rather than a "computer". But the underlying mathematical constructs which built the digital computer remain, along with a whole lot of mathematical flotsam and jetsam which have held back progress and made simple and free media out of reach.

In Geometron we are switching from a world view based on numbers to one based on geometry. This represents a shift in value system. In "computer science", the manipulation of numbers and logic are considered the most fundamental operations. In Geometron, we consider geometric constructions to be the most fundamental. This is a shift in perspective, which we can apply to the whole of the existing machines. How are these ma-

chines built? The microchips which make them work are nothing but huge geometric constructions, made up of little overlapping rectangles and polygons. These chips are then laid out on circuit boards which are again geometric constructions. The chips are placed automatically on the boards using machines programmed to carry out a sequence of geometric motions. They are packaged in cases made in molds again machined with this kind of geometric programming. And finally when assembled, their main task is displaying symbols on the screen which is again just geometric construction.

It is easy to forget given the onslaught of propaganda from Silicon Valley just how accidental the rise of their machines as the dominant technology was. We also are encouraged to forget that these machines were built *primarily* for war initially, then large authoritarian organizations to track and control people, and only later, almost as an afterthought, as the communication devices we rely on for all aspects of modern life. One of the theses of Geometron is that a shift in thinking from one based on numbers to one based on geometry is a shift away from the ideology of dominating large amounts of land and people toward one of cooperation based on sharing of technology and ideas.

When we build everything from trash found directly in our environment, empire-building doesn't really accomplish anything. The people 1000 miles away from you have the same piles of broken phones you do, so you

gain nothing by dominating them and vice versa. But if you can share with them how to make those phones part of your free network, the value of *your* network infrastructure goes up exponentially, just as we find in all networks. as they scale up.

In computer science, they work with an idea called a Turing Machine, named after computer pioneer Alan Turing, which is a generalized machine for doing arithmetic. An infinite tape of ones and zeros is fed into this imaginary machine, and the contents of that tape give instructions to the machine, which then carries out actions on the ones and zeros on the tape. Any computer, regardless of the details of how it is built, can be shown to be equivalent to this toy model. In Geometron, we are creating a similar object: an abstraction which can construct any symbol, which takes symbols as an input.

The basis of geometric programming in Geometron is the Geometron Virtual Machine, or GVM. Just like the Turing Machine, this is an abstract construct which carries out geometric constructions based on a set of instructions. We assume that there is a main program, which we call a "glyph", which consists of a sequence of symbols, each of which represents a geometric action. Just as the Turing machine reduces every math problem to binary arithmetic, our machine reduces all geometry to discrete geometry. The GVM has an internal geometric state which represents its progress in doing geometric actions. If we are programming a physical machine like a

pen on a plotter, the position of the pen is stored this way. This logic is very similar to that of languages like Logo, a teaching language developed in the 1970s, which uses what is called "turtle logic", where patterns are drawn by a virtual turtle which moves around on a screen, and which can also be performed by a physical robot with a pen. However what we are doing differs radically from Logo as we will see below.

We also have states of this virtual machine which describe what motions it can carry out. The most basic of these is the step size. By creating geometric programs using an abstract step size without actual numbers, we can create programs to draw symbols independent of what machine we use and what scale we are at, copying verbatim a program from a giant wall climbing robot which spray paints symbols on a building to a nanolithography system which prints the exact same symbol in an area smaller than a human hair, without ever dealing with the mechanism of either machine. This unit state is also used to define how we do constructions like "draw a circle".

The most basic geometric program we can carry out is the construction of the Vesica Piscis. This figure, from the Latin "fish bladder", is just two circles each of which has its center along the edge of the other. In the dialect of Geometron presented here, the symbols are all printed inside squares of identical size, patterned from left to right. The symbol describing the action of drawing a

circle is a square with a circle in it, and that draws a circle with radius equal to the current value of the step size. If we draw a circle of radius equal to step size, then move sideways one unit of that same step size and repeat the action, we get the Vesica Piscis. The symbol to move to the side is just an arrow pointing to he side. These symbols are themselves constructed using more of the actions of Geometron: drawing line segments, rotating by discrete angles, scaling the unit down and back up and so on.

Another central organizing principle of Geometron is that there are special symmetries and scales that are intrinsic to the Universe which we use to simplify how we approach geometry. In a numbers-driven system, coordinates and angles are all equal. 37.34 degrees is no different than 36 degrees for example, they are just different numbers with no special properties. But this ignores some very deep patterns in how the world around us functions. Both the natural world and the constructed world of human technologies rely constantly on special rotational symmetries, starting with small numbers of rotations, in particular twofold, threefold, fourfold, fivefold, sixfold, eightfold, tenfold, and twelve fold. From there if we add halving angles and dividing them by 3 we can get all the way to the 360 fold symmetry which defines the degree of angular measure in most common use. Using only discrete geometric manipulations, we can go from fivefold symmetry which is based on 72 degrees, divide

Figure 8.1: Vesica piscis, or "fish bladder" spelled out with Geometron symbol glyphs.

by 2 to get 36 degrees, then divide by 3 three times to get 1 degree. So discrete geometric actions can be used to do a very wide range of geometric constructions without ever relying on reference to numerical representations of angles. The dialect of Geometron presented is based primarily on 4,5, and 6 fold symmetry, combined with halving, doubling, dividing by three, and multiplying by three to construct all angular rotation actions.

By default, we increase or decrease our unit of step movement and construction by factors of two. Just like in digital computers, this binary representation allows us in principle to represent any number using only geometric actions of doubling, halving and moving by discrete amounts. Again this allows us to program a machine to go to any coordinate without actually using numbers, only using symbols which represent geometric actions. Those could all be represented by numbers of course, but we choose to *express* them using pure geometric symbols. And just as binary arithmetic can express any number, this binary geometry can express every position in space.

Another intrinsic geometric property we find in the world around us is the scales which naturally go along with these symmetries. For example, when we deal with fivefold symmetry, everything is based on the Golden Ratio. The ratio of the side of a regular pentagon to the distance along the cords which make up a pentagram drawn inside it is this ratio. If we then build a fractal of pentagrams inside pentagrams, the way we scale down to smaller and smaller pentagrams and pentagons is again and again the Golden Ratio. This number is about 1.6 and is a universal constant built into the structure of the Universe, found in all kinds of natural systems, as well as used throughout human art, architecture, and technology. While a numbers-based system can of course compute geometry using this number expressed as a repeating decimal, in Geometron we simply use a symbol

Figure 8.2: The pentagram in a pentagon, showing the relationship between the Golden Ratio and fivefold symmetry.

to represent this scale, without any symbolic reference to its numerical value.

The square root of three plays a similar role to the Golden Ratio, but for sixfold symmetry. If one connects alternating corners of a regular hexagon, those cords are the square root of three times the length of a side. Again, a fractal construction of hexagons and six pointed stars

Figure 8.3: The six pointed star in a hexagon, showing the relationship between the square root of three and sixfold symmetry.

shows a square root of three scaling over and over. Similarly, the square root of two is intrinsic to four fold symmetry, as if we draw a diagonal line across a square that is the square root of two times the side length.

Altogether, the scales used in this dialect are, in order, the square root of two, the Golden Ratio, the square root of three, 2, 3, and 5. When a scale is set, that scale is the

Figure 8.4: Embedded squares showing the relationship between four and eight fold symmetry and the square root of two.

factor by which the unit is either multiplied or divided when we apply a scale-up or scale-down operation.

The figures show the symbols for all these scale values. We are now ready to understand all 8 of the basic discrete movements: move forward, move back, move left, move right, rotate left, rotate right, scale down and scale up.

The current state of the GVM is expressed with the

Figure 8.5: Scale actions in order: square root of two, Golden Ratio, square root of three, two, three, five.

Global Cursor, a shape which shows the position, scale, the step angle size, the current direction which is "forward" and the current directions which are "left" and "right".

The basic constructions of Geometron are to draw a point at the current location, draw a circle of radius equal to the current unit, draw a line segment along the forward direction, and draw an arc from one of the cursor wings to the other. More advanced actions include writing letters, creating paths(both filled and unfilled, closed and open), and drawing Bezier curves, all of which will be covered in the next, more detailed section on the web-based graphics system which is built into Geometron.

Color and line width of lines are set with the layer system. At any given point along the construction of a Geometron glyph, one of the states is the current layer, of which there are 8. Each layer has a stroke color, a fill color, and a line width. These are set in an object which

Figure 8.6: GVM cursor with movements. The arrows represent movement of the GVM position along the indicated directions relative to the cursor. Angle rotations are as shown. The plus and minus symbols are also shown and how the rescale to where the points are on the cursor.

Figure 8.7: Symbols for the layers have little line segments to denote the layer number, and show the line width, fill colors and stroke colors in the border and fill of squares.

can be edited and customized, which the GVM calls on when it draws symbols.

A very important point to make about how all this fits together is that each of the symbols shown here which represent these geometric actions are themselves constructed using this language. When a symbol is displayed in the spelling out of a Geometron glyph, each of the actions which compose the glyph has a symbol which is itself a Geometron glyph, and the whole sequence of symbols which spells out the glyph is itself one giant glyph which tells the GVM how to spell out this human readable symbolic description.

Geometron glyphs consist of a sequence of geometric actions. Each action has a symbol, which is itself made up of actions, each of which has a symbol and so on(recall that one of the laws of Geometron is that everything is recursive). Each geometric action is represented by an address in the Geometron Hypercube. The Hy-

Figure 8.8: Breaking a symbol glyph down into its constituent action glyphs. This shows the spelling for the geometric action which moves forward one unit. It includes a sub-action which draws the arrow.

percube consists of two cubes, each divided into 8x8x8 = 512 cells, for a total of 1024 cells. Each cell contains a glyph, which is itself a sequence of addresses in the Hypercube. The Hypercube is therefore a kind of recursive data structure, with many components which all point back to itself. It might be added that human languages are all forms of recursive data structure, as they are described using the language itself(e.g.dictionaries). Hypercube addresses consist of four digits each of which is a number between 0 and 7, and the first digit of which is 0 or 1, just denoting what cube it in.

Why, you might ask, do we add this complexity? It is deceptively powerful to create a structure like this. Note that this structure is completely geometric. While we emphrepresent each cell with numerical addresses, the actual underlying structure is geometric and symbolic. Elements represent geometric symbols, human language describing geometry, computer language describing geometry, and locations in a data space. This is a non-numerical construct, and represents a fundamental shift from the Turing model of computers to a model for a generalized symbol constructor. If our goal in building technology is to draw symbols, our fundamental models should reflect this, and not mask it in "computation".

Of the two cubes in the Hypercube, one is the "action cube" and the other is the "symbol cube", which has a leading 1 in the 4 digit address instead of 0. Each action in the action cube therefore has a corresponding symbol.

Figure 8.9: The two Cubes of the Geometron Hypercube: the Action Cube and the Symbol Cube.

Thus when a GVM is spelling out a sequence of symbol glyphs, it is just carrying out the actions represented by addresses of the form 01xyz, which are in turn simply doing the list of actions stored in that address in order. Any time you see Geometron symbol glyph spelling, you are looking at a sequence of symbol cube addresses.

But what of the Action Cube? This has a lot more structure than just all being actions. As said above, we are looking to build a geometric language which can apply to the widest possible range of generalized symbols. This means we want to be able to not only make 2d symbols, but to make complex fractal structures of sym-

bols made from symbols for specialized graphics, printing in any human language, 3d constructions, editing the hypercube itself and perhaps most importantly machine control for generalized automation.

The Action Cube's bottom addresses from 0 to 037 represent actions directly on the Hypercube and the GVM and environment. These are used for tasks like moving the cursor around, choosing which element of the Hypercube is being edited, deleting addresses from a glyph, and changing the view geometry of the symbol display(zoom and pan).

Addresses from 040 through 0176 represent the printable characters on the standard keyboard using the ASCII code. These addresses all map to actions which are carried out when that key is struck on a keyboard. These are physical inputs. They are geometric in the sense that keyboards are geometric, and that hitting keys is a geometric action. The corresponding symbols stored in the Symbol Cube at 01040 through 01176 represent a font. These can be any kind of character, and can be used for any keyboard mapping to any human language. This represents an alternative to Unicode, in which each glyph is directly created using the Geometron langauge, rather than called from the system's interpretation of Unicode.

Address 0177 represents "do nothing" and can be empty or used as a dummy variable.

The addresses between 0300 and 0377 are the two dimensional actions described above and in the next chap-

ter, which are used for making web graphics, saved vector graphics files, and all the symbols used to represent Geometron glyphs. These are in practice all some type of computer function call, and since computer code is stored in ASCII and ASCII is part of the Hypercube, these little bits of code represent the sequence of Hypercube addresses which can be represented as a sequence of ASCII codes which can map to addresses in the Hypercube, so we retain our generalized structure. Also, if we take this as a total abstraction, we could describe the geometric action using text in a human language, such as "draw circle of unit radius", and that can be encoded in ASCII to make it part of the Hypercube format as well.

The addresses from 0200 through 0277 are the "Shape Table" and all store sequences either in 02xy as well or in the range of 03xy. This is merely a convention, but these are used to build up specialized language, such as for building circuit diagrams or cross stitch patterns. This topic has its own chapter, as there is a very rich range of languages which we can build this way.

Addresses between 0400 and 0477 represent machine actions. The most basic actions for machines we consider are for machines that have 3 perpendicular axes of movement, of which we have several examples later in this work. Using only the actions of discrete movement and binary manipulation on the scale of the step, this system can be used to encode any motion of a robotic probe, but that only takes up one row of 8 out of the total of 64 pos-

sible actions. This is also a topic so rich in structure that it has its own chapter. The ability to use a web interface to do totally generalized programming of the geometry of machines for automation can revolutionize the control humans have over machines and over automation.

Addresses between 0500 and 0577 are another shape table, specifically for referencing machine actions in 04xy. As with all Shape Tables, this can be self-referential, and we can use this space to build up fractal structures of motions within motions. This can be a huge enabling technology for programming of automation using simple machines. The actions in this range are used to create a form of generalized "icon", which also gets its own chapter. Icons are sequences of movements on a rectangular grid, which either involve drawing a pixel or not. This sequence is used to create physical media using a variety of technologies, which are documented in the aforementioned chapter.

Finally, the addresses at the top of the Action Cube from 0700 through 0777 represent three dimensional constructions. These are abstract, and can be applied to any method of three dimensional construction. These also get their own chapter, and can be used to create graphics in the virtual reality format of x3d as well as .stl files which can be printed on a 3d printer. They are displayed in the browser live using webGL libraries discussed later on. The Shape Table at 0600 through 0677 is for calling actions in the 07xy cube, for building up complex frac-

tal structures in 3d construction. This can server as the basis of a whole alternative system for creating parts for 3d printing, as well as a basis for web-based 3d graphics for both virtual reality and augmented reality.

The structures described here, the GVM and the Geometron Hypercube, represent a new way of thinking about how humans control machines. In this way of thinking, the purpose of machines is to encode geometric information on the physical world in a way which communicates information to another human being. We start with this as a goal and build up a set of abstractions to do that as generally as possible. Just as the Turing Machine toy model has been implemented in vastly different physical systems, our abstract idea about symbol drawing machines can take totally different forms which carry out the same task as well. This is a key element of how we will build the hybrid trash-based hardware architecture described later in this work. We do not want to compute things. We want to display symbols and images on screens, and control the movements of the machines we build from trash to automate building more things from trash. That is all. If we can display a generic geometric symbol on a scavenged screen, we can create fully Organic Media as described in an earlier chapter. That is, media which self-replicates openly on a physical network, which needs no mined materials, no money, and no property. If this is easy and uses trashed cell phones, the constant stream of broken phones will allow us to create

a model of ubiquitous networking of Organic Media by way of the Street Network. With the consumer based system creating billions and billions of phones all headed to the landfill, we can imagine a model where there are screens everywhere, but they're not mobile because they don't need to be. In a network with self-replicating documents, you can read what documents you want wherever you go. You can interact with a universe of documents which flows through the physical world, as you traversethat world, interacting with and improving documents as you go as you interact with people in the physical world. The hardware architecture to realize this will be discussed in its own chapter on Full Stack Geometron.

Chapter 9

2d Web Graphics

The previous chapter introduced the abstract idea of geometric programming with the Geometron Virtual Machine and the Geometron Hypercube. In this chapter we dive into the specifics of how this is implemented on the Geometron Server so that we can create, edit, and most importantly share symbols from a web browser. The heart of what is described in this chapter is the Symbol app which exists at symbol.html on every Geometron Server.

In the Symbol app, we can build Geometron glyphs either using a specially marked keyboard or soft keys in a control panel. To learn the system, it is best to first get a keyboard and mark it up with the symbols so that you can see what you are doing as you learn. The examples and figures in this section give keyboard values

along with symbols in many cases so that you can follow along if the keyboard is not marked and can get used to the layout. The layout is easy to change, and presumably will evolve over time, but for starting it makes sense to copy whatever keyboard layout the person sharing the system with you uses, in this case me. See the figure with the keyboard layout for guidance, and go ahead and put markings on keys with these symbols. I also encourage the further decoration of Geometron Keyboards, with extensive paint and modification so that it retains the distinctive flair of Trash Magic(rainbow paint and googly eyes). Paint pens are ideal for this, but nail polish or other permanent paints which can be applied with fine lines can all work. A less permanent solution can be found with masking tape on the keys and symbols drawn on the tape. Stickers can also be printed out and applied to keys.

The control panel of soft keys should be pretty self-explanatory: the buttons do the same thing as the keys on the keyboard and are there to make the system work with a touch screen when no physical keyboard is connected, as with smart phones and tablets.

With the ability to edit the main glyph, the best way to learn the language is to just try stuff. This chapter is largely pictorial. Go through the figures and try to copy them on the system you are using, play with all the different symmetries, scales, layers, and functions. To save a symbol, hit the save icon which is in the far left of the

Figure 9.1: Keyboard layout. Strike a key to add an action to the glyph you are editing. Arrow keys and backspace allow for editing the whole glyph. Up and down arrows move to the end or beginning of the glyph.

menu bar at the top of the screen. When you have saved a symbol you can see it and download it from the Symbol Feed, which is an icon three in from the left showing equilateral triangles separated by an arrow. Each symbol is stored as both a bitmap in .png format and a vector graphics file in .svg format. What is listed in this Feed is simply a sequence of stored files in the directory symbolfeed/, which exists on every Geometron server. Each file is named with the UNIX timestamp which describes the exact moment it was created. The Symbol Feed, like all Feeds in the System, also has delete buttons to delete any of the symbols in the Feed, following the Law of Geometron that everything dies. If you click on an .svg file in the Feed, it will load that symbol into the editor, and you can then go back and edit a copy of it, and save it

again to make a modification. The Symbol Feed also has an upload button just like the Local Image Feed, which lets you upload .svg files from other systems and allow you to edit them using the copy as a template. Thus we can freely replicate, edit, delete and replicate again.

Move the cursor around, draw circles and lines and squares. Make polygons, explore scales and symmetries. Try out the Bezier path drawing. Play with colors. Make closed and open paths. When in doubt, destroy it all and start over.

As you create and edit the main glyph, the address sequence will update live in the text field next to the one where the cursor is for editing. This will be a sequence of numbers all of which begin with 0. As with all other parts of the Geometron system, the ability to share information instantly across the world with it is of the highest importance. If you copy the sequence of addresses in that field to the clip board and send it via text message, email or pastebin to someone else with Geometron, they can paste it in that same field and hit return and they will instantly load the same glyph you created, which they can then edit and send back to you modified.

What you are editing in the system is not just a glyph but a whole JSON structure which is how we can share a whole symbol including style, position, scale, and custom symbolic language tools. This structure will be described in more detail below, but you can immediately start sharing them by clicking on the icon which says JSON in the

Figure 9.2: Edit symbols. This shows the symbols for moving the cursor back and forward, deleting an action, and also shows what the cursor looks like in a glyph being edited.

Symbol app. This will bring up a screen which has the JSON data in a text area, and a set of buttons to export, import, save, or reset. Reset is important because it is possible to corrupt the data beyond recognition and this gets you back to something which will definitely work. As with all other components of the Geometron system, this is a human readable text format which is designed to be shared both by direct text messaging and by copy and pasting into public pastebins and sharing the pastebin link.

Figure 9.3: Movements. Arrows move along directions of the lines in the cursor. Rotation is by the unit indicated by the cursor wing angles. Scale actions are by the current scale value as shown by the dot positions on the cursor. Letters shown indicate the keys which map to these actions on a QWERTY keyboard with the default settings.

Each instance of the Geometron Virtual Machine has a style object, which defines 8 layers, numbered from 0 to 7. Each style has a line color, line width, and fill color. The properties of the style object are stored in the JSON file data/currentjson.txt which is used by the app symbol.html to edit graphics which are used by the rest of the Geometron system.

While the style app edits the data file currentjson.txt which applies to the whole Geometron object used for

Figure 9.4: Angles described by symmetry glyphs. This also shows the actions to bisect, double, trisect and triple angles, and what keys are used to activate each geometric action.

symbol editing, the importing and exporting of data for sharing with other users only includes style information, without the rest of the JSON data. This allows styles to be separated from the rest of the information for the purposes as usual of building a robust remix culture where Geometron users can constantly be sharing each piece of the system. The EXPORT button will always post the current style JSON in the window in the lower left of the screen. IMPORT will import the data, and RESET returns it to a default state. Try creating your own new style with unusual line widths and colors, then exporting it and saving it offline, sharing it with other users, etc.

Colors are in the format of HTML/CSS/JavaScript,

Figure 9.5: Scales, along with keys used to map to them in default configuration. There is no relation between the numbers on the keys and the mathematics of the scales. The scales shown are, from left to right, the square root of 2, the Golden Ratio, the square root of 3, 2, 3, and 5.

and can be either names of colors like “red” or RGB color values like “#00ff00”. This last format is a number in base 16 which has three 2 digit numbers in it(numbers between 0 and 0xFF), where the three numbers are values of red, green, and then blue. So black is #000000 and white is #FFFFFF. Any value where all three numbers are the same, like #808080 will be a shade of grey. Colors can be partially transparent by adding a fourth hexidecimal number which represents opacity. So fully opaque red is #FF0000FF, and red with half transparency is FF000080(80 because 8 is half of 16, this is actually 128

Figure 9.6: Basic drawing actions, along with keys used in default configuration to activate them. From left to right the actions are: draw dot, draw circle of unit radius, draw line segment of unit length, draw arc between cursor wings, draw a square, draw a filled square, draw a filled circle, and draw a line segment while moving forward one unit.

in decimal).

The next section of the JSON we want to know how to edit in order to be able to make useful graphics is the setup, edited in the app setup.html. Setup edits five numbers, all of which are in units of pixels: x0, y0, unit, width and height. Width and height are the width and height of the graphics file currently being edited or created. When a Geometron glyph is drawn with a given GVM, it starts with x and y equal to x0 and y0. Setting these two values is therefore effectively setting the

Figure 9.7: Layers. Each layer has a line color, line width, and fill color, all of which are set with the Style object using the Style editor app.

Figure 9.8: Path actions, with keys used to activate them in default state. From left to right, actions are: start path, draw line segment in path, close a filled path, close an unfilled path, and terminate a path without closing it.

Figure 9.9: Start a Bezier Path and terminate it with the y and u keys.

horizontal and vertical offset of the field of view of the symbol. When we activate a pan function within the symbol.html app what we are really doing is modifying the values of x0 and y0 in the JSON file. These are done manually in this app. Finally, unit describes the initial unit value of the GVM. This is essentially the scale factor. So again when we activate the zoom functions in any other symbol editor what we are really doing is making changes to the variable unit in the global JSON file.

The app setup.html has five fields in which to enter the numbers for the five values. There is also a reset button to restore default, with a 600 by 600 pixel square and 80 pixel unit centered in the center.

The final section of the global JSON file which defines

Figure 9.10: Demonstrating the power of Geometron to make useful symbols with Bezier paths quickly and easily: a twiddle bracket.

Figure 9.11: Pan and zoom the field of view.

Figure 9.12: Drop a flag, return to flag. This saves and then recalls the state of the GVM.

Figure 9.13: The "hello world" of geometric programming, the Vesica Piscis.

Figure 9.14: Cursor scale. This shows how scale works with the Geometron cursor.

Figure 9.15: Cursor angle. This shows how angles work with the Geometron cursor.

Figure 9.16: What the cursor looks like with factor of two scaling and a 90 degree angle. Also shown is a square used in Action Geometry. Try making the square!

Figure 9.17: Another example of a commonly used Geometron cursor state, which combines the square root of two with 45 degree angles. Also shown is yet another shape used in Action Geometry which is also a good exercise to try to copy yourself.

Figure 9.18: Cursor with Golden Ratio scaling and 72 degree angle for fivefold symmetry work. Shown is another shape that is helpful to learn to copy, the pentagon/pentagram fractal.

Figure 9.19: Cursor with square root of three scaling and 60 degree angle. This can be used to make the kinds of symbols shown, and replicating that is a useful exercise, as well as working through the deconstruction of the six pointed star and hexagon.

Figure 9.20: The cursor with a 60 degree angle and factor of 3 scaling, along with another exercise to copy.

Figure 9.21: Cursor with scaling factor 5 and right angles. This can be used along with scale factor 2 to make things with scale factor 10. What is shown to copy as an exercise is a ruler constructed using this tool which can be made physical using a laser cutter as discussed in the Action Geometry chapter.

Figure 9.22: Screen shot of the style editor app at styleeditor.html. The display on the right hand side of the screen shows an unfilled circle and filled circle of each layer's style. The text area in the bottom left of the screen is used to import and export style data, which can be saved offline and shared with other users via text message, email, etc. The RESET button resets the style to a standard setting, which will erase any changes made to the existing style. Enter new values into any field to immediately change it.

the settings of the symbol app is the shape stack. This is a subset of the hypercube, and is stored both in the JSON file data/hypercube.txt and also data/currentjson.txt. When vector graphics files are saved, this shape stack is stored inside them so that they can be reloaded with the whole stack. The creating and sharing of useful shape stacks will be discussed in the next chapter. This is a very important element of the system as it allows for the rapid dissemination of specialized graphical languages for things like drawing circuit diagrams or subway maps.

Other links to other apps from the main symbol app are to the keyboard editor, which edits the layout of the keyboard, the Hypercube editor, which edits the entire Geometron Hypercube, and the Font editor, which edits just the font. All these have the capacity to share human readable text from system to system as is the case with everything in the Geometron system. The font and Hypercube apps are still somewhat crude and could be improved substantially, but they do work for their intended purpose with a little bit of fiddling. They will be used through the rest of this work as we delve into more applications of Geometron geometric programming.

There is also an app separate from the main symbol editor called symboltrace.html, which allows you to trace images into symbols, which you can then copy and paste at will. This takes images from both the local and global image feed, so you will need to load the image you want to copy in one of those first. Still another random app

of some utility is action2symbol.html, which converts a glyph made up of actions to one made up of the symbols which correspond to those actions. This is how we put glyph symbol spelling into a symbol, which is very useful for documenting things which reference the language and how it is used.

Part of the power of the Symbols in Geometron is how they are integrated into the rest of the system. When you save a symbol, a copy of the base 64 encoded bitmap is stored in the textfeed which gets used by the Map editor to create maps. This means that you can create a symbol, then go import it into the Map editor immediately. Because it is a self-contained image url which has the actual graphics encoded in the url rather than as an external file, this Map you create can then be shared with anyone else anywhere in the world with another Geometron system and they will be able to import and use that Map without any other image files. This ability to instantly integrate Symbols into Maps can enable a sort of graphical meme system which can be extremely powerful for numerous applications.

Also, we note that all icons used in the Geometron system are created using the Symbol editor described in this section. These are then stored in the directory icon-symbols/. This directory is all copied with every copy of the whole self-replicating system. Therefore if you want to add another symbol to the next copy of the system, just add your new symbol to this directory, run dna-

generator.php, and the next instance will have your new image.

If you are a coder, you can read the whole of the system documented in this chapter in the JavaScript library stored at jscode/geometron.js, which is on every instance of the system. You can of course edit this, and your edits will replicate along to the next instance of the system.

Chapter 10

Shapes and Fonts

To see the full power of the Geometron idea of geometric programming, we must explore how we build custom graphical languages. Language customization happens by editing the contents of the Geometron Hypercube, the geometric structure described in the Symbols chapter which defines the meanings of geometric actions carried out by the Geometron Virtual Machine. Each action also has a symbol. So to create a new graphical language, we decide what graphical elements we want to create, build those, and then also build symbols for each one which fit the format of fitting inside a square with a left to right progression so that they can be part of the glyph spelling we use when creating and editing glyphs.

This chapter is more technical than most people will need or want, and is here for completeness so that the

system is fully documented. Primarily it is a gallery of pictorial examples, and the reader is encouraged on a first reading to skip the text and look at the pictures before moving on to the next section. This is partly intended as a stub of sorts, a documentation of software which people can re-write to be easier to use as the system evolves.

In principle, there are hundreds of addresses in the Geoemtron Hypercube which we can edit to create custom languages. These are divided up by function and will be dealt with in different sections of this book. In practice when we use the basic symbol editor for making 2d web graphics, by far the most commonly used is the Shape Stack. This is a selection of addresses which are reserved for building languages which have up to 16 shapes, with a matching set of 16 symbols. The important thing about the Shape Stack is that it gets saved inside each .svg file when they are created in a human readable format which can be extracted back out of the file either automatically or manually. This is how it is possible to click on a .svg file in the Symbol Feed and have the correct symbol load up and be edited, even when symbols have vastly different specific graphical languages they use. Engaging the 16 elements of the Shape Stack is done in the default keyboard configuration with capital letters in the second two rows on the keyboard from left to right, or Q, W, E, R, T, Y, U, I, A, S, D, F, G, H, J, K.

To edit the Shape Stack, we use the shape stack editor which is at shapestack.html. This is linked via an icon

with a 4x4 grid on it. The shape stack editor uses only the keyboard, and does not have a touch screen interface at the moment. To edit a shape, you use the customized Geometron keyboard just as you would for creating and editing glyphs in the main symbol editor. To move to the next and previous shapes, use the up and down arrows. Try stepping through up and down and get a feel for that. Also, don't forget that you can use the pan and zoom actions to change your view to zoom in on details you are editing, and that these actions are located at the lowercase "l", the semicolon, and n, m and comma and period.

After you create a new shape, you can change the symbol which represents that shape by clicking on the hypercube icon to go to the shape symbol editor at shapestacksymbols.html. This is basically the same editor, but pointed at the addresses corresponding to the symbols of the actions you edited in the main editor. We always want to edit the symbol for a given action so that at the end the cursor is one unit to the right, a square has been drawn, and the state of the GVM is set back to 90 degree angles and factor of 2 scaling.

The shape stack editor also allows us to trace over images, and you can click on the images in a window which lists the images in the image feeds to select an image, then use the slider bars for scale and rotation to set the image where you need it to in order to trace it.

As with all elements of the Geometron system, the

most important thing we can do with a shape stack is to share it from person to person across the world. And as always, this information needs to be human readable text in the smallest possible format so that we can copy it to a clipboard and paste it in text messages, emails, directly from browser window to browser window or into pastebins which we can share. The format for a shape stack is an array of quoted text inside square brackets, separated by commas each of which has an address followed by a colon and then a sequence of addresses which represent the glyph stored at that address. Once one gains familiarity with the address system of the Geometron Hypercube it is possible to read this code and manipulate it by hand. However, for people who don't want to dive deeply into any of this, or even to learn how to edit it or interact with it at all, the important thing to know is just that you can use the export and import buttons to exchange shape stacks between one Geometron server and another.

Also, if you want to see the shape stack used in the creation of a Geometron symbol in .svg format, you can open that symbol in a web browser and use the "view source" feature which exists in all browsers. Inside the source code for the .svg you will see a little bit of JSON code at the top inside an XML comment, as well as commented out XML tags which mark where the json is. This JSON can then be manually copy and pasted into the JSON importer of the whole symbol system or just

the shape stack can be removed and used.

If we want to edit and share Hypercubes, we use the Hypercube Editor, also linked from the main symbol app at symbol.html. This is at hypercube.html on every Geometron server. In this editor we can edit all parts of the hypercube which contain Geometron glyphs in the byte code format which is sequences of base 8 addresses. These include all the symbol glyphs, as well as the 8x8 tablets stored in the address ranges starting with 2, 5, and 6. Once again, we can export and import using buttons and a text area as in all parts of the system. The actual file being edited with the Hypercube editor is stored at data/hypercube.txt, and you can always look at that in raw plain text and copy it from there as well. Again, this editor is a sort of stub, and as our swarm grows, more and more people will create their own editors for the Hypercube and various parts of it, as the logic is very simple and it is not hard to make a much better editor than the one presented here. Also, in future editors, the fully three and four dimensional structure can be brought out, using the three dimensional web graphics described in a later chapter.

The font editor works the same way as the Hypercube editor, with the same basic edit functions but for the range of the Hypercube which holds the font: from 01040 through 01176. Again, this is designed to be shared, with text based import and export of the human readable text which holds the byte code. The two most important

fonts we use are the Robot Font which has pixels which are drawn by the various robots described later, and the Laser Font which has cutouts which when used with a laser cutter can make stencils for spray painting text onto physical objects. This is part of how we jump the gap from the digital to the physical: text describes a font, text describes a glyph, then that exports to a vector file, and that loads into a laser cutter which makes a stencil which makes a physical link in a physical space which points to a domain which points to the physical location of a server which contains all the information required to replicate that entire system.

The rest of this chapter is simply a gallery of examples of using the shape stack and fonts to make and use different graphical languages to show the potential of Geometron. Also blank pages and margins can be used to add hand drawn graphics documenting graphical languages for custom illuminated manuscripts to exchange for barter via the Street Network.

Figure 10.1: RLC circuit. Circuit diagrams are a great example of how powerful it is to create a custom graphical language. Once the schematic symbols for the components are programmed into the Shape Stack and mapped to keys on the keyboard, it is possible to create any circuit diagram with just a few keystrokes. This simplicity of construction also makes it possible to use any other software tool to generate schematic diagrams automatically. The symbol glyphs shown spell out the symbol, showing the extreme simplicity of construction once shapes are in place.

Figure 10.2: Construction of a single loop of an inductor. Once this loop has been created, it can be used to make inductors of any length as well as transformers and any other schematic symbol of something with inductive elements. This shows the power of building up symbols as fractals. Note the geometric precision and simplicity which are not present in the commercial vector graphics packages.

Figure 10.3: With the loop in place, just a few keystrokes makes a full inductor.

Figure 10.4: RC line circuit. Another demonstration of the power of Geometron for circuit building.

Figure 10.5: The Tree of Life from Jewish mysticism. This symbol also is used in various occult systems. There is quite a bit of geometry hiding in here, and you can see numerous published examples which get that geometry wrong, breaking the six-fold symmetry of the hexagon. With Geometron we can build a language for moving around on a hexagon and creating all the various bridges between nodes.

Figure 10.6: Symbol glyph spelling of Tree of Life. What makes things like this easy to make is building up building blocks like the cross pieces of all different scales, and the use of universal symmetries and scales(6 fold, 12 fold, and the square root of 3 and 2).

Figure 10.7: Molecular symbol for the steroid hormone estrogen. Note the characteristic hexagon-pentagon combination which is repeated throughout self-replicating chemical systems(life) as well as throughout the Geometron system/language.

Figure 10.8: Symbolic spelling of the chemical symbol in the previous figure. This spelling is only possible because of constructing a symbolic language specific to drawing organic chemistry symbols.

אבגד

Figure 10.9: Some letters from a Hebrew font. Typesetting a language which goes from right to left is a powerful demonstration of the way Geometron works. Each letter is a sequence of geometric actions, just as it is when a human uses a pen to create it. This means that if the actions end with the cursor to the left or bottom or diagonally that is where the next letter will be. There is no need to mode switch based on direction, the direction is built into each individual character.

Figure 10.10: A katakana font using pixels. Again, we can create this with a sense of direction built in, which for Japanese has a few choices. We make this vertical so that it can be used to make Matrix rain, which is katakana in mirror writing. The font is made as regular characters, and then by simply changing the value in the hypercube of movement from left to right and from right to left from pixel to pixel, the katakana can be made mirrored or not without affecting the actual font elements.

Chapter 11

Action Geometry

Action Geometry
- the idea
- shapes
- products
- the geometry, relation to geometron
- shapes and rulers, protractors, laser cutting
- methods of construction
- materials
- skeletron
- artbox
- pyramid

Figure 11.1: Shape Set. This is the basic shape set of Action Geometry. It has the symmetries and scales of Geometron. What is shown should be printed exactly 6 inches wide, making each shape three inches on a side.

Figure 11.2: Rulers. Make this 6 inches wide and each ruler is a 6 inch ruler, 1 inch across, with both tenth and factor of two divisions.

Figure 11.3: Geometron protractor. While not really needed for Action Geometry, this protractor is a nice accessory which emphasizes Geometron symmetries rather than numbers, and allows drawing of circles of radius 3,2 and 1 inch without a compass. This is mostly useful if cut out with a laser cutter.

Figure 11.4: Penrose. Penrose tiles, the rhombi construction.

Figure 11.5: Spray paint stencil for laser cutting. You can use the Geometron software, selecting the built in laser font, to make a custom spray paint stencil pointing to the domain name which points to your Geometron server. If you are Trash Robot, that can be Trash Robot, but we mostly point to a local non-property place.

Figure 11.6: Construction of the Raspberry Pi logo for server bags. The top two shapes are green the rest are red.

Figure 11.7: Skeletron tripod. Three 6 foot bamboo trash poles wrapped in rainbow colored duct tape with quarter inch holes drilled just back from the end. An 18 inch nylon parachute cord trash tie is used with a square knot to secure the top. Many things can be hung from this, including servers, terminals, robots, boxes, flags, lights, textile arts and pendants on more trash ties. The tripod can be carried over the shoulder to be mobile, without untying the joint at the top for rapid deployment.

Figure 11.8: Skeletron cross bar configuration. Two tripods can be converted into this stable configuration quickly to have a horizontal cross bar which S-Hooks can hang from to hang numerous objects of all kinds.

Figure 11.9: S-Hook. Note the repeated use of the square shape, and the use of the 45 degree triangle, making this easy to replicate using the Shape Set. This hook is used to hang things from the Trash Poles in Skeletron. Fabricate by stacking 4 identical layers of corrugated cardboard cut in this pattern and wrapping them in rainbow colored duct tape. Googly eyes can then be applied.

Figure 11.10: Tape Snake. Duct tape rolls of all useful colors, namely the rainbow colors plus black and pink, are strung on a 6 foot Trash Tie made from a clothesline which is looped through twice and secured with a square knot in a bight(like a bow on a tied shoe) for rapid replacement of rolls as they are used.

Figure 11.11: ArtBox Net. Cut out 10 equilateral triangles from corrugated cardboard. Duct tape the joints as shown.

Figure 11.12: ArtBox assembly. The fully assembled box is a tetrahedron on an octahedron. It should contain the means of its own replication, which is a box cutter, a ruler, an equilateral triangle, and a sharpie, along with a Tape Snake for duct tape fabrication, extra Trash Ties, and googly eyes. Use duct tape colors in sequence to create a fully rainbowed effect, then apply googly eyes and add a black duct tape mouth.

Figure 11.13: Top view of ArtBox. Cut out little triangles in each of the top three petal triangles with the box cutter. Thread a 6 foot Trash Tie with ends taped with duct tape as shown, and tie off the two bitter ends with a double figure eight knot for convenient purse-strap geometry.

Figure 11.14: Pyramid net. Use the Shape Set and Ruler to cut out corrugated cardboard patterns and stitch together with duct tape as shown. Cutouts include a 6 inch square, a 3 inch square, and four trapezoids with 3 inch top and 6 inch base with 72 degree angles on the bottom angles.

Figure 11.15: Pyramid assembly. Fold it all up and then cover the whole thing in a skin of rainbow duct tape. Open the base, insert technology and re-seal. Add cutouts as needed for cords in and out.

Figure 11.16: Bags are cut from black cloth, which can be scrap. Black cloth bags are sewn up with Trash Ties as draw strings.

Figure 11.17: Draw your personal Geometron outfit which is black with solid rainbow color felt or similar cloth sewn on in geometric patterns and geometric font.

Figure 11.18: Draw your Flag. A Flag is a black square cloth about 3 feet on a side, with a sewn hem with loops to tie a trash tie. It is decorated with solid block letters and geometric shapes cut from solid color rainbow felt or similar. All elements can be from scrap. Flags point to domains in places.

Chapter 12

Machine Control

12.1 Machine Control

democratization of machines

We should have as much control as possible of every machine we interact with and every machine which affects our lives and the lives of our communities. We currently live in a time where our ability to control our own machines is decreasing very rapidly, and it is the intent of this work to build technology which reverses this trend. For the last few hundred years, as machines have become more powerful and complex there has been a constant conflict between machines' ability to reduce labor and their ability to also increase the power and control of whoever controls the machines. In the early days of the

Industrial Revolution we started to see battle lines being drawn around the forces of mechanization which rage as hot today as they did then.

The basic conflict of the Industrial Revolution was between laborers who saw machines as replacing their role in society and those who saw the machines as having an overall benefit to society at large by producing more products faster and more efficiently. This conflict stemmed from the way we have structured labor and work during the time and place that these technologies came to power, namely the wage labor system. Under wage labor, the more "labor" someone does, the more money they get to survive. This creates an incentive for automation for the capitalist class, and create the opposite incentive in the laboring classes. Marxists took this structure for granted and argued that the laboring classes could liberate themselves through political control, but took the basic premise of "work" or "labor" for granted as ideas.

The time we find ourselves in today differs so vastly from that of the industrial revolution that we must pause and re-evaluate some assumptions. The most fundamental assumption of the time of the Industrial Revolution which we must dispose of is the need for a constant stream of mined materials and materials extracted from far away places. This element of industrial production is often hidden from the consumer but is far more fundamental than people often recognize. Every early industrialization process, involved some type of empire-

building over large land masses in order to build stable supply chains to maintain a flow of materials from mine to factory to consumer. The more deeply one studies industrialization the more clear this becomes. Some countries industrialized without building their own empires, but they did so by leveraging powerful trade relationships with existing empires. In all cases, the power of the physical network of raw materials was what made or broke the process.

But today the world we live in it materially different in this fundamental way: all the materials of a mechanized civilization are now readily available everywhere in the world, without exception. The global consumer industrial capitalist system has consumed the mineral wealth of the last 300 years of imperialism and industrialization remarkably evenly through the world by pushing the same consumer products everywhere, all of which form localized waste streams of highly processed materials.

Just take aluminum as one example. Aluminum is one of the most powerful tools we have ever discovered to create technology, with a very wide range of uses. However, before we can even shape it we have to reduce the bauxite ore, which is a very energy intensive process. I strongly recommend that the reader spend some time choosing random elements like aluminum or phosphate and learning where they come from. All the minerals we use take vast amounts of energy and complex mechani-

cal processes. And yet, all of them are now free! If we want to use aluminum in a technology today not only do we not have to reduce the ore to pure metal, we don't even have to machine it. We can find fully processed and coated aluminum shaped into already-useful shapes from beer cans to light poles. If we build our civilization on this waste stream, all the assumptions of politics and economics from the last 300 years of Western thought must be abandoned.

Today, we can build machines directly from the materials we find in our environment, without the need for a global supply chain we control. Even if the existing supply chains collapse, the existing cache of waste we find in our immediate environment should be sufficient to live technologically advanced lives. Therefore, under these new material conditions, machine control is direct, physical, and built into the design of the machine, rather than based on "power" in relation to large organizations like nation-states and corporations.

[all the above paragraphs should move to chapter 1, then just get referenced here as we dive into the following list of principles]

If our goal is to have the most control possible over our own lives in a technological society, we now aim to design that control directly into the machines. Democratization of this control(avoiding building a power elite of technology experts) then becomes a task of user interface design. If we look at the last few hundred years of build-

ing labor-saving machines, we find some good things and some bad things. In this work, I aim to create a framework for making those choices and for building technology which follows the principle that the more direct control the individual has over their technology the better, and will now delineate those principles.

Direct control before automation. This means every operation of a machine has a physical control which a user can use to directly control that operation. If a tool can be moved up, down, left, right, forward, and back, we will always have a controller which enables them to directly carry out that action without any further automation. This controller should be as physical as possible(rather than through software and touch screens). Unlike the automation discussed below, it will generally be as continuous as possible, allowing a user to directly move small or large distances with feedback being directly trough observation of the machine outside of any control technology.

Controllers are sized for humans. We want controllers which are as closely matched to how humans think, feel, and act as possible. This means things will be much bigger and more redundant than they need to be by the absolute minimal requirements of control. Buttons and levers will be as large as they have to to allow the absolute maximum range of human users to operate it without difficulty. We will not have defaults be small buttons and have a special case with large buttons, but

will always have buttons sized for people with limited sight or dexterity.

Controls should use geometry to be obvious. A lever next to a hoist should have pulling it up move the hoist up, and down be down. A pair of buttons should have the top one move up and bottom one move down. We also shape the controls to have geometric meaning as much as possible, for instance shaping a button as a physical arrow pointing in the direction of the motion it causes.

All actions should be convertible to automation. This means if we can use a controller to move a tool along some path, we can build a program which repeats that action with a single button press. Thus a controller will have, in addition to the direct controls, a "go" button which will initiate the automation program. The previous two principles dictate that this button be large, green, and round.

All automation actions can be stopped at any time. The big red "stop" button must also be universal. Pushing this button immediately stops the automating action. This is different from the "EPO" or Emergency Power Off button which can also be a good idea, which shuts down a machine completely. A stop button simply stops whatever action was initiated by the go button.

Discrete geometric actions of Geometron, using geometric symbols enable programming with no specialized technical knowledge. This is the heart

of where Geometron enables democratization of machine control. The idea of "move one unit to the right" is something humans can understand without any special training. The idea of making a symbol for that action which consists of an arrow or equivalent is also universal enough to require to special training and also eliminates the language barrier one gets from using word-based human language. These discrete motions sound limiting, but as with the earlier chapters of this work, if we allow for operations which change the unit of movement, we can make arbitrarily complex and precise motions built up from combinations of different units of movement.

Symbol hierarchy. Symbols are used describe action sequences which are built up to make more complex actions. So an action like "move right, lower drill press, raise drill press" can be condensed into "drill a hole", which has its own symbol, and then "drill a row of holes" has its own symbol, which is made up of a simple sequence of the "drill a hole" symbols. Those rows can be terminated with a "move back to start" action by chaining "move left" actions, along with a "move towards" action. When these are put together into a bigger sequence we can build up to a symbol which means "drill an array of holes", symbolized by just an array of dots in the standard Geometron convention of a square with symbols in it along with movement to the position of the next symbol.

Upcycled hardware. This is the principle used

throughout this work, but must be re-emphasized here because it is so important and because it is so easy for controllers. All we need is buttons and mechanical structure. The main body of controllers can be made from cardboard trash, as can all the surfaces touched by the user. Using simple geometry for all these parts allows them to be replicated effectively using social media without sending any files from user to user, just copying a geometric construction using simple shapes and symmetries. Buttons themselves can be found in any of numerous consumer waste products, often in fairly large number. An old stereo or printer can have dozens of buttons sometimes, and since the basic controllers we will build generally have about 8 buttons, this means one scavenged electronic junk item can yield multiple controllers in some cases. Ribbon cables and connectors can also be scavenged, but they are so cheap and easy to buy that for the time being buying them from consumer off the shelf sources can be more efficient.

Structure

The control of machines using Geometron follows the pattern of the rest of Geometron as documented in earlier parts of this manuscript. We assign addresses in the Action Cube in the Geometron Hypercube to geometric actions. These actions all have symbols in the Symbol Cube of the Geometron Hypercube. The Geometron Hyper-

Figure 12.1: An example controller showing all of the above design principles. Buttons are 1.5 inches on a side and raised up above the surface of the controller enough to be easy to feel. Their shape indicates their function, being triangles pointing in the directions of the actions. The go button is big and green, and the stop button is big and red. The whole thing is built from cardboard and duct tape, avoiding any use of circuit boards which require custom fabrication in a factory. The shapes of all components, from button cutouts to the top and bottom layers of large cardboard slabs are all designed using very simple geometric constructions using the laser cut acrylic shapes described elsewhere in this work. This geometric construction makes the whole artifact easy to replicate accurately without any contact with the physical thing to copy or any skill or knowledge in the relevant craft.

cube, as described earlier, is a pair of cubes, each with 8x8x8 = 512 cells. One cube represents actions, and one cube represents symbols. Each cell in each cube contains a sequence of addresses which describe either a geometric action or a symbol of an action. Layers of the action cube are divided up by what kind of action they represent.

Actions which represent direct control of machines are assigned to the addresses in layer 4 of the Action Cube. All addresses in the Geometron Hypercube begin with a zero to indicate that they are in a range from 0 to 7(base 8). The addresses representing machine actions are of the form 04xy, where xy are numbers from 0 to 7. Most of this set of actions is empty in the version of Geometron presented here, leaving a large space open for future development, potentially adding vast complexity if that is useful. The way actions in the 04xy block are performed varies by the specifics of what hardware is used. Design of symbols and programs can be carried out entirely in a Web browser using the main system of Geometron. No hardware control is needed at all to design these systems. It can be totally self-contained in the browser, building up custom languages of symbols in the Symbol Cube of the Hypercube in the address space defined by 014xy, where x and y are the same coordinates between 0 and 7 as in the Action Cube. Again, sequences of addresses form glyphs which are just text which can be copied and pasted and shared instantly throughout the global network by private text message or public text file sharing

and replication as described in the code chapter.

Actions representing combinations of basic actions, like the "drill an array of holes" action mentioned above are put in the

2 and a half d printing and the Trash Robot icon token printer

The wall robot

Microfabrication and Nanofabrication

While we can use Geometron and Trash Robot control of electron beam lithography for all sorts of technology, as with the rest of the system our initial goal is simply to be able to print symbols and make self-replicating media. One simple way to do that is to just program a "write pixel" action which is un-blanking the beam for a fixed amount of time, putting that in the pixel-write actions in the Hypercube at 0500 through 0503, along with xy control and printing identical symbols of pixels to the rest of the Trash Robot system. To make this self-replicating we want to not just print a symbol on a substrate, but to make stamps which can be used to *replicate* symbols from prints from the lithography system. Ideally, as with the clay tokens documented in the last chapter of this work, an initial print can be used to make many stamps which in turn are used to make many of the final print.

Figure 12.2: A probe move over a sample in either the x or z direction, and the sample moves along the y axis. Repeated poking of the sample prints dots a simple but very versatile tool.

Figure 12.3: Basic geometric actions of machine control for an arbitrary machine that moves along three perpendicular axes.

Figure 12.4: Dot actions from which symbols are constructed.

While a fully demonstration of this technology is beyond the scope of the present manuscript, I believe this can be done relatively easily using spot size of about 1 micron, etching in polished brass wafers, using a negative resist. Blasting the resist with extremely high doses of beam should be sufficient to reverse the effect of standard PMMA from a positive to a negative resist, making an etch process possible with just a single coat of

Figure 12.5: Block diagram of Trash Robot printer. The motion stages are salvaged from DVD or CD ROM drives. The motor drivers are off the shelf from Pololu Robotics. The Arduino is the Arduino UNO, and the whole system including the motors gets power from the USB connected to a computer. The computer can be used to interact with a Geoemtron server over the local network or on the global Internet to create programs which are copy/pasted into the Arduino IDE as described in the Trash Robot chapter.

Figure 12.6: A hoist run along a rail going across the edge of a roof of a building can make a simple robot which can move to anywhere along the wall.

Figure 12.7: Block diagram of electron beam lithography Geometron robot. The Arduino here drives three analog to digital converters. Again the user can design a program for the beam path in a web browser using any computer, which can then also control and power the Arduino and its accessories. In this case the controller only needs one huge green go button and one huge red stop button.

thick PMMA spun on polished brass. Printing in brassh should enable replication by imprinting patterns in soft materials in the brass. Also, it should be possible to cast materials into the brass which can then act as stamps after they cure in other soft materials.

With pixel size of 1 micron, a typical symbol could be smaller than 100 microns. Or a 1 cm area could have 100x100 = 10,000 symbols printed! There is plenty of room to store quite a lot of information on a 1 cm chip this way, including simple human readable text, QR codes to links, images, symbols, and Geometron byte-code. In raw data density, 1 micron pixels on a 1 cm square amounts to 10,000 pixels on a side or 100,000,000 pixels, theoretically 100 megabytes. With a lot of empty space and error correction, this should easily be 1 megabyte, and if it's in high relief brass, it should be possible to replicate that information many times without any further use of advanced technology. If the brass wafer makes many stamps which themselves make many copies, we can hope that in theory a *very* large number of copies are made for each original.

Chapter 13

Geometron in 3d and Beyond

13.1 Geometron in 3d and Beyond

Structure, Definitions and Formats

As discussed in the section on Geometron structure, there is a layer of the Geometron Hypercube devoted to 3d geometric actions in software(as opposed to machine control). These actions are assigned the addresses between 0700 and 0777(base 8). For each action, there is a two-dimensional Geometron symbol with addresses equal to the actions, but with a 1 before them, ranging from 01700

Figure 13.1: Actions. This is a minimalistic action set. More complex actions are added based on specialized applications.

through 01777(base 8). As with the rest of Geometron, a user can edit a glyph which is made up of a sequence of geometric actions using an interface based on manipulating the symbols which represent those actions in a Web browser.

While it is possible to use the full grid of all 64 actions to create extremely complex three dimensional constructions, for the purpose of this work we restrict ourselves to a few basic actions. As with all other parts of Geometron we use the fact that discrete motions along the available geometric directions combined with halving and doubling of the unit of movement is sufficient to move a cursor to any place in space. We therefore have as our basic actions 6 discrete movements along three axes, the halving and doubling of the unit of motion, and the construction of a sphere and a cube, along with actions to select colors.

The addresses in the Hypercube from 0600 through 0677 represent "shapes" in three dimensional Geometron, which also have two dimensional Geometron symbols in the range from 01600 to 01677. These can be used to build up arbitrarily complex objects with layers of structural hierarchy. As an example of how useful this is, we will show how we can build up a universal rectangular base on which to print symbols.

In Geometron we are always concerned with making useful files which can lead to either physical production or have some function in operating the overall system.

Figure 13.2: shapebuilding1.

Figure 13.3: shapebuilding2.

We also want to be able to replicate symbols created in other parts of the system in the three dimensional spaces we create using the software. To see what is useful for this, we have to discuss some of the formats and tools available for three dimensional graphics in Web browsers and in everyday use.

The most basic way to render three dimensional graphics, and the oldest is to use the current updated version of what used to be called VRML, or Virtual Reality Markup Language. This was the original format back in the 1990's when people were first talking about a new way to exist on the Internet with ubiquitous virtual reality. This new web never materialized. Also, VRML was replaced by a newer more widely used format, called x3d, which is a subset of XML, just like HTML.

The most general three dimensional Geometron app is called voxel.html. A "voxel" is the three dimensional extension of the idea of a "pixel", and this software can be used to quickly draw three dimensional objects in the browser using either softkeys or a special keyboard mapping specifically for 3d. The way voxel.html works, much like most Geometron apps is that it edits in real time the value of a file stored in the Geometron server at data/glyph3d.txt. This glyph is a string consisting of a sequence of base 8 numbers separated by commas as usual in Geometron. Also, as usual, as we edit a glyph the text input with the glyph code is updated live. We can always copy and paste any glyph from any instance of

Geometron to any other. So if you are making a glyph in your local private server and send a private text message to another user with the text in the glyph input copied, they can paste that into the private server on their network and instantly replicate your glyph in their system. The way the app voxel.html works is that it has a local function in the HTML code which acts on the Geometron Hypercube by manipulating global geometric variables x, y and z, and constructing shapes in an x3d object. The specification of x3d contains basic geometry primitives such as cube and sphere, as well as transformations like translation and scale operations.

All of this work is carried out using the x3dom JavaScript library x3dom.js. This is documented at www.x3dom.org. This library allows for creation of x3d objects in an HTML file using DOM(Document Object Model) JavaScript commands. As the JavaScript code alters the x3d object, however, it creates actual human readable code which can be copied into a separate x3d file. This is done with the "save" button which copies the x3d code into another html file wrapped in a header and footer to make it render in a browser, called three.html. Every time a user hits the "save" button the old version of three.html is destroyed and replaced with a new version with whatever the current status of the 3d object is in voxel.html. The source code of three.html can be copy/pasted as a x3d object which can be embedded in any type of VR system which uses that format. This can be used to create games and

virtual reality or augmented reality applications based on Geometron.

Another method for creating 3d graphics in the Web is the use of WebGL, a cross-browser graphics framework, via the JavaScript library Three.js. This fantastically powerful and well-documented library is another free open source library with publicly available CDNs for easy use. For more details see the home page of the project at threejs.org. The Geometron app threejs.html opens the same glyph file edited by voxel.html and renders it using three.js. This passive program serves two purposes. First of all, it implements the whole basic Geometron Hypercube using the three.js library, which can form a jumping off point for developers reading this to create more interesting applications such as games or other dynamic user interfaces. But its more immediate utility is the three.js library's ability to export to the .stl format. This format turns 3d surfaces into meshes of triangles, and is the universal language used by 3d printer software. Thus the ability to export from a Geometron glyph to .stl allows us to 3d print objects we create in Geometron. This adds yet another physical layer to the types of media which we can exchange using Geometron code. When a user loads the page threejs.html it automatically saves another file called data/three.stl, which a user can download by right clicking on the link, and can then share that .stl file with anyone with a 3d printer to print it out. That .stl file can also be opened with a lot

of 3d editing software and integrated into other projects, allowing the Geometron user to make icons which other people can integrate into other physical designs for 3d printers or even CNC and/or injection molding. Note that the .stl files can get very large, and above a certain size the Geometron software will fail to save, as there is a limit to the size we can pass via the POST command in PHP. We leave this limit in place to keep the system working with modest sized files. This limit often makes spheres problematic and we replace them with cubes when we want to 3d print.

The app shapes3d.html allows users to edit Geometron glyphs stored in addresses from 0600 through 0607 in the Geometron Hypercube using the same editor style as in voxel.html. Up and down arrows cycle through the addresses. As a user edits the glyphs in each address, the global value of the hypercube stored at data/hypercube.txt is edited in real time. Each action sequence glyph in the range 0600 through 0607 has a corresponding symbol glyph made of 2d Geometron actions in the address range from 01600 through 01607. These are edited in a similar manner to other Geometron Hypercube stacks using the app shapes3dsymbols.html, which is linked from shapes3d.html.

As an example of using this method of building up actions to make shapes which are built up into more complex shapes, we will create a simple castle object. This starts with constructing a single tower made up of cubes

Figure 13.4: Turret construction.

of decreasing size(see figure). We can then repeat this turret and move repeatedly to build up a square wall to make a sort of castle, the Castle of Whimsy(see figures).

Tool trajectories can be turned into 3d files.

Icons and Trash Robot

We will now explore how the system described above is combined with the overall power of Trash Robot and Geometron to build symbols which link to the rest of the system here. The whole system of creating symbols for sharing in Trash Robot is based on pixel movements

Figure 13.5: whimsycastlespelling.

Figure 13.6: castle 3d.

stored in addresses 0500, 0501, 0502, and 0503. These elements as previously documented both draw pixels on a screen in a canvas element(or SVG) and also control the motors in any of a number of robotic systems which prints in any of various materials. We now add yet another sequence of actions to these: movement with the drawing of a cube as a pixel(also known as a "voxel"). As with the robot control, We further add 3d geometry actions to the movement commands 0504 through 0507, so that we have the ability to move the 3d cursor around either with or without drawing a voxel. If we combine this with creating a base tablet in the form of a large array of cubes in a plane, we can create either 3d web assets

for VR, AR and other 3d web graphics and also can create 3d printable objects to share, all from the same robot code used in the rest of the Trash Robot system. Just as we have an app for choosing an icon from the Trash Robot icon feed to print in the Trash Robot two and a half D printer, we have an app called icon3d.html which has a listing of all the Trash Robot icons, and when one is selected it loads up a 3d object with the icon printed in cubic voxels on the surface of a 8x8x1 rectangular solid. As in voxel.html, clicking on the "save" button will save the file to three.html, and loading the page threejs.html will resave the value of three.stl, overwriting the old version, for download and 3d printing. This system means that we can freely share feeds of icon glyphs which can be physically shared by 3d printing in addition to all the other methods of sharing.

3d printed parts can be used to stamp the symbol into soft objects, or used to make molds used to cast pourable objects with the icon in them. Simple modification of the values of the hypercube to reverse the sign of the x direction can create a mirror image in the 3d object, which can be useful if we wish to create a stamp which when imprinted into soft objects directly prints the icon without having to create another negative. Thus we can use 3d printing and basic Geometron operations to create copies of freely shared icons in any of a vast potential universe of materials both hard and soft. We can store feeds of icons in Geometron Trash Robot code in globally

accessible text files, and link to those text files on pages on Geometron server domains which we link to in URLS printed directly on 3d printed parts. This makes the whole system self-replicating. A user can pass you a 3d printed part which has an icon on it which directs the user to a domain name which hosts a site which links to a feed which has a list of icons all of which can be replicated in another 3d printer just by clicking on them. This represents a decentralized network of free sharing of physical objects imprinted with icons which represent arbitrary things(see the ontology chapter for details on the implications of this).

Rotations and Higher Dimensions

As with every part of Geoemetron, we are only scratching the surface in this work of what can be done with these methods as we add actions to the Geometron Hypercube. Of the available 64 actions in the Hypercube reserved for 3d geometric actions we have so far only used 18. Any type of discrete geometric action can be added. In particular, rotations of rigid bodies is a deep field of mathematics which we have so far ignored. So far we have only moved our geometric axes around along the x, y and z axes or scaled them all together. We can add non-uniform scaling to make a much wider range of objects, and discrete rotations to create different orientations of objects. We can also add a larger range of

Figure 13.7: robot0500.

Figure 13.8: trashrobot3d.png

geometric primitives, such as cylinders, cones, and other custom, manually-constructed polyhedra.

Discrete rotations can be a huge enabler of applications for Geometron. By constructing spheres, moving along a radius vector, rotating, and moving, we can build up three dimensional models of molecules. These molecules can then be simulated by recasting problems of how they fold in terms of angular rotations of the Geometron glyph used to construct them. Sequences of chained discrete rotations can also be used to describe state vectors in the Hilbert space spanned by the qubits in a quantum processor. An alternative way to think

about quantum computing can therefore be constructed in which instead of programming bit states, we use Geometron to express states in terms entirely of rotation angles, and all gate control pulses can then be thought of as further discrete rotation operations. If we map both states and dynamic operations in a quantum computer to Geometron glyphs, and also map problems of quantum chemistry(e.g. protein folding) into the same language, is seems at least plausible that such mapping could yield useful results in attempting to solve protein folding problems using the so-called NISQ(Noisy Intermediate Scale Quantum) systems which are presently available. Also direct mapping between geometric operations in graphics displayed in a web browser and states or operations on a quantum processor admits the possibility of a very simple and powerful real time user interface for development of quantum algorithms, again on existing NISQ devices. Such software could be easily created on top of the software frameworks currently under development by several parties.

Figure 13.9: angles.

Chapter 14

Full Stack Geometron

14.1 Full Stack Geometron

- end goals
- hybrid fabrication technology
- clockless operation, hardware GVM
- image stack, hardware map processing
- roctal, storage hardware, scaling, read/write/operate
- large projections in Skeletron booths, Geometron station, fully immersive VR/AR at 2 meter scale

Goals

Our end goal is an information technology which has zero mined materials. Every material input in the creation of any technology in our system should come from either a

waste stream or from the living world in a closed loop(e.g. naturally harvested sticks and tree sap, goose guano).

Our initial goal has been to rely on the most open and lowest cost off the shelf hardware possible, which right now is the Raspberry Pi. As the system expands we will want more and more ways to fully wipe hard drives and sim cards and replace them with Raspberry-Pi-like systems running some form of Linux, with web servers installed with PHP running the same sofwtware. That will be the next step after the Pi phase. But after that we will try to break up the system into modular parts which can be scavenged from more and more destroyed electronics.

We see this as a continuous transition where in the beginning we are just running an app on an existing phone, then we wipe the phones and run our own OS, and then we start taking screens, batteries, and other components out and mixing and matching them. It is in this mixing and matching where the next phase of development will happen.

As this mix-and-match upcycled Linux system evolves, the Trash Robot will in parallel evolve more elaborate ways to fabricate electrical connections at the millimeter scale. This will eventually lead to more and more complex circuits being built up from scratch, connecting various upcycled components at the individual level(a single resistor, capacitor or MOSFET for instance). When these circuits can be fully integrated in three dimensions

into collections of upcycled components, and when we have some degree of automation in this process, we can start to say we have a fully upcycled system, but it will still be the same software stack, with Geometron running on PHP, Apache, and Linux.

This hybrid interconnect technology can be based on immersion of circuits in an electrolyte solution with a robotic probe which applies current to deposit metal, as well as techniques where the robot probe is modifying clay, and a series of clay fabrication steps are carried out in layers just as conventional micro-lithography adds layers.

In parallel to all of this we need to be developing battery technology using aluminum from beverage cans with carbon from charcoal made from organic material like burned dung. We also need energy production based on ultra small scale generation using found materials. So our goal will be to find a reliable waste stream like fans from broken computers, turn that into a generator of electricity which can run on very small scale water or wind at the level of from 100's of watts down to single digits of watts. Using the aluminum air cycle with organic waste and beer cans from the trash it should be possible to have a zero-mine closed loop energy production and storage system at the level of IT platforms. Building IT infrastructure as a shared resource in some local public place which stays fixed in a location removes the severe demands currently placed on battery technology for net-

worked devices which have to fit in a pocket. If servers also serve as the information terminal, with a giant screen for all passerby to see, a battery could be a 6 foot high stack and it would have no real impact on usability. This relaxed restriction combined with using waste aluminum cans and locally sourced burned dung should make aluminum air cycle batteries practical in a way they have not been traditionally.

Another technology which has to be developed as part of the Trash Robot road map is printing in permanent materials at the micron or sub micron scale for long term information storage. This has to be developed in parallel with readout technology which uses upcycled camera chips from trashed smart phones as sensors in our hybrid fabrication technology which can read the information back out.

Developing our own information storage system is also a part of a much more radical shift in information technology, where we abandon the whole operating system and build hardware for direct Geometron-based document processing. To do this, we re-adjust what a machine does. The only goal of our machine is to freely share documents over a network. When a machine is not operating something mechanical like the Trash Robot, all it does is display pixels on the screen. Those pixels only change if a user engages with the system. We therefore totally replace the model of a Turing machine doing arithmetic on ones and zeros with a model which

just uses the Geometron Virtual Machine to draw vector graphics in the screen pixels(this can include all text in all human languages, which can all be drawn using Geometron fonts) and a system for displaying bitmaps in geometric positions as described in the Geometron Map format. To do this, we will need an image stack, a memory system which records an array of bitmaps of images, and a Geometron Virtual Machine. We will also need a network of switches, where user interaction with keyboards, pointers, buttons etc. can trigger behavior like scrolling or editing a scroll, zooming on a map, drawing a symbol, or clicking on a link which goes to some other file.

All this does not need a constantly running clock. This is one of the biggest differences between traditional computers and a full stack Geometron machine. When there is no user interaction it simply does nothing. It doesn't just have minimal operating system services running in the background, it doesn't even have a clock. No aspect of the system changes state in any way. It is just a collection of pixels getting enough power to stay in whatever state they are in, until the user engages, and then it only changes state in direct response to each user interaction.

The data we store in the system can be either short term or long term. Short term data can be stored in a modularized memory unit based on easily found memory chips from trashed phones, but with standardized

interconnects added by the Trash Robot fab technology so that they can be easily plugged in and replaced after they are upcycled into our system. Long term memory is designed to be *very* long term. We can use scaled down versions of the Trash Robot clay printer presented here to print in clay at a much smaller scale. As with the larger clay symbol tokens, this can also be self-replicating, as the depressions in the clay can again be used to stamp out inverse images which can be used to make more. So with no automation or IT at all someone with no tools or even electricity should be able to clone data storage media.

The format for long term storage is in arrays of squares or dots which are either in a 1 or zero state, filled in or empty. If possible, the data all have redundancy, with Arabic numerals written along with each Geometron bytecode byte, as well as in some cases an actual print of the symbol being represented, and alignment marks for that individual byte. This redundancy and alignment mark system should make the data maximally robust against decay and future technological limitations, as if it were absolutely necessary, the

Chapter 15

Ontology

15.1 Ontology

1. the branch of metaphysics dealing with the nature of being.
2. a set of concepts and categories in a subject area or domain that shows their properties and the relations between them.

The purpose of our whole structure of knowledge is to create a civilization in which technology freely replicates. We will choose terminology and methodology entirely based on this criterion. The breaking down of things into sets composed of elements which are themselves things is based on this. We always choose that which maximizes the probability of replication and the

freeness of the thing being replicated. To this end, how do we define a "thing"? Topology of replication, meaning the connection in space and time between people, actions, places, materials, energy, information and other things which are required to replicate the thing. Set structure of replication, meaning the way we break the thing down into elements and those elements into sets with more elements to maximize the ability of the human mind to understand how to replicate the thing. And geometry, the specific geometric constructions that make up the physical replication of the thing. To create a Geometron Thing is to build whatever media we need to create and document the topology, set structure, and geometry required for replication.

We consider things in the most abstract sense. From things we create sets, which are also things. This section is a more formal mathematical treatment of the material in the "magic" chapter. It is more technical than that chapter, and does not reference magic, but does reference our specific implementation.

This is where the Theory of Self replicating sets is described. We point to the failures of the project of set theory in the 20th century, how it was a dead end, and how we can navigate out of the dead end by placing the mathematicians ourselves in the sets we describe. The whole failure of the old models of self replicating machines(perhaps this gets moved back to organic media chapter).

set notation, departure from ZF-C paradigm. New way of dealing with foundations of mathematics: the mathematician is part of the sets considered. And we continue this through the whole construction of math.

Foundational math in the paradigm of the 20th century was from ZF-C set theory to arthmetic, to algebra, to geometry, etc. . . but the whole edifice is based on sets which follow axioms, but those sets do not contain their makers. We create a foundation of mathematics in which the creators of the math are always in their own calculations, starting at the primitive level of what an "object" is and what a set of objects is. We consider not only ourselves to be elements of the sets in our universe, but every aspect of ourselves, our communities, our minds, and our ecosystems. For example, looping back to SRS, the desire to replicate a set is part of the set. A set might be

{ - powerpoint slide about self replicating sets - the desire to tell people about self replicating sets - the persuasiveness to induce others to desire to tell people about SRS - the means to replicate the powerpoint to the next user(e.g. posted on slideshare) }

From these sets we construct geometry which is used to make generalized symbols, which are used to make all the things in our technological complete set, which is again a part of our new approach to interacting with abstract sets.

Random notes:

unlike normal set theory, sets can be elements of each other(we call it "element", not "subset", to erase the distintion between set and some hypothetical non-set thing)

The structure of sets, the elements we choose to describe a set, are completely determined by what will be the most useful for replicating the set, for a human operator.

example Things:
The Terminal
The Geometron Terminal
The Pyramids
Audion
ArtBox
Trash Robot Icon Printer
Skeletron
laser cut shape set
laser cut ruler
custom laser cut shapes and stencils

15.2 On Self-Replicating Sets

1. Computer Science and the Theory of Self-Replication

Throughout the history of modern computing machines, people have contemplated the idea self-replicating machines. At the dawn of the information age, John Von

Neumann in particular devoted thought to the subject, creating a blueprint that people continue to use both for understanding hypothetical self replicating machines and for understanding the architecture of existing computing machines. At the same time, Alan Turing developed a similar toy model for how generalized computing machines work, which is taught in basic computer science classes to this day. We will not delve very deeply into these models, but will instead present a very crude sketch of them in order to discuss the assumptions made by computer scientists in the models they build to understand their world.

The Turing model of a computer consists of an infinite tape of 1's and 0's along with a machine to both read and change the state of the numbers in the tape. The string of numbers describes instructions for the machine, which follows those instructions by moving the tape back and forth and changing numbers from one state to another. Turing was able to show that this toy model of an abstract computing machine could be proven mathematically to be equivalent to any other abstract toy model of any other kind of computer, including the complex machines built today(hence the continued use of this model in teaching and scholarship). This is considered to be one of the most important results in theoretical computer science.

Before discussing this model's limitations we must say a word about the nature of scientific models. When we

investigate a thing using the scientific method we have in principle the entirety of science knowledge to call on, built up from a vast number of models in different fields and sub-fields. For example, if we are presented with a rock to analyze, a physicist might ignore everything but the crystal structure of some prominent material in the rock and bring the modern understanding of crystals to bear on it. The micro-biologist will only be interested in the aspects of the rock that interact with the organisms on the surface, while the ecologist will be interested in that but primarily how the rock regulates the flow of water through the ecosystem of which it is a part. As scientists we may agree that models of surface chemistry for microbes, models of how atoms arrange in crystals, and how water flows through rocky soil are all "good science", but in any given context the model we choose to focus on depends on that context.

It is our assertion that while the model of Turing and his contemporaries is not wrong, that it is deeply misleading because in most cases it does not describe the most relevant aspects of the machines we call "computers". Computers do, of course, compute. That computation is described by the mathematics of computation. They also create heat, described by thermodynamics, and are we therefore to call them heaters? They keep time with extremely fast clocks, do we simply call them clocks? No. But what actually are they? What is the model for a modern computer which is most useful when trying to

understand them? In this era, in the year 2020, the most useful model for a computer is the one that describes how they have totally changed all aspects of global society in a relatively short time. For this we must expand the models available, and in particular we must focus on a specific shortcoming not just of mid-20th century computer scientists, but of most scientists in the "modern" era, namely our refusal to put ourselves and the societies of which we are a part into the systems which we study. In some abstract sense, one can argue that we put ourselves into the models with the role of the observer in quantum mechanics for instance. But we do not put the role of something like university politics into the models, even though these forces clearly influence the science we use and hence the conclusions we draw. It is the assertion of this paper that this blindness has become so critical in the understanding of computers as to be wrong in a way that has real world consequences.

So what is the problem with a Turing machine? If we look at it naively as a physical thing, not as a mathematical toy, we see a number of things that are not realistic, such as the infinite tape and the lack of any meaningful human readable input or output. But these are typical of useful scientific models: while the tape is understood to never be really infinite, the results you can get from the machine with "such a large memory that it doesn't matter" and infinite mostly don't matter, so our toy model still works. But the critical flaw of the machine which

gives us an incorrect understanding of how computers function in society is that it has no origin, no purpose, and no destiny. Who built this machine? Why? How long can it run before it breaks? What happens when it breaks? When it ceases to function, what replaces it and why?

We now live in a world where a large fraction of the computing machines that exist live on a trajectory from a mine to a landfill of less than 2 years. During this brief journey from mine to landfill, they are mostly used for communication, and much of that communication is marketing information the primary purpose of which is to lead to further consumption of similar machines. That is, *in their current state* the *primary* action of these machines is rapid replication.

This is the reason we must shift the *primary* scientific model of computing machines from the Turing model to a more biological understanding. If we seek to study a new species of life that is taking over an ecosystem rapidly, we will always try to focus on the models that allow us to understand that phenomenon because it is what affects outcome. If a farmer asked us to evaluate a new crop blight and we came back with a deep study of how carbon chemistry works because all life contains carbon they would be very disappointed in the results even if they are technically correct. Likewise if we are to deal with the explosive changes to our physical world caused by computing machines we must focus on the means of

replication. We will give an analysis of these means in the next section. However before doing that we must turn to the history of self-replication as an idea in mainstream computer science.

There are two main intellectual threads in this story: cellular automata and self-replicating robots. Interestingly, it is the former of these that actually have a vast experimental component and the latter which is entirely theoretical. Cellular automata are in some sense a generalization of the Turing model: they are sets of rules for multidimensional(usually 2 dimensional) grids of numbers, generally 1's and 0's, which follow some set of rules. These systems are simulated on real computers, and as time advances in a program we can see fascinating patterns of what appear to be naturally oscillating structures moving around in a visual display of dots on a two dimensional canvas. If the rules are created correctly, these structures can be made to replicate themselves. The literature on cellular automata is vast and complex and continues to be a very active field. Nonetheless, it suffers the same flaw as the Turing model: it exists in a vacuum, with an assumed infinite time, and no reason to exist(at least this reason is not part of the theoretical framework used to describe them), no origin and no ultimate destiny when it breaks. There is beautiful pure math to be found in these systems, but no illumination of how computers function as machines that copy themselves.

The second thread of self-replicating machines is the

study of theoretical "robots that build robots". This has attracted some truly wild speculation. What is generally imagined is a totally automated system without human intervention in which a computer controls robots that mine minerals which are used to build both more computers and more robots. This is then imagined to be so self-contained that it can be used to expand out into the universe outside of Earth, eventually consuming all things into this vast, automated, technical ecosystem which does not need any living thing to grow. Technologists have imagined these systems, then promised that they can be a fantastical utopia of free things, but also warned that they can destroy the world by consuming everything in site. For decades, theorists have written very detailed descriptions of such systems, delving into metallurgy, synthetic chemistry and the like to try to prove that such a thing can be built. Most recently, the work of K. Eric Drexler and Ralph Merkle in the 1990s pointed to a system like this built from precise positioning of atoms in matter–essentially treating physical matter as just another Turing machine. These theorists constructed very detailed imaginary worlds where atomically precise machines manipulate atoms to both compute and create, replicating themselves on a global scale. Perhaps their day will come at some future time and such machines will be built, but right now these models are of no help in understanding technology as we find it today. The rest of this paper is devoted to developing both a theory

of self replication in modern technological society and in actually *using* this theory, just as the pioneers of modern computer science used Turing's model early on, to build new things based on the new model.

2. Self-Replication in Human society

First, a word on self-replication in biological systems that exist outside of human society. "Natural" biological organisms never replicate themselves in a vacuum. On Earth as we find it today, all living systems replicate as part of larger ecosystems, and the parameters of those ecosystems are *fundamental* to the overall replication. No animal can live long enough to reproduce without a constant flow of oxygen from plants needed for respiration. Conversely plants need the carbon dioxide we produce in order to live long enough to replicate. No one would dispute that a tree is a self-replicating thing, and yet trees only replicate in collaboration with a large number of other organisms, generally other plants, fungi, animals, all working in concert to make the overall forest system replicate itself, of which the trees are only a part. What we find in spite of this, however, is that scientists outside of biology put much more restrictive rules on self-replication, saying a thing does not "really" replicate itself if it replicates as part of a larger system. Hence the flaw in computer models: if humans replicate machines, those machines are not called "self-replicating" because

for the overly restrictive definition of the "self" of the machine they do not spontaneously replicate. But this type of isolated spontaneous replication does not exists in nature even for purely biological systems, so applying it to systems outside of biology will give results that are at odds with how the biological world works. Again, we must distinguish between models that are "right" and models that describe the primary characteristic of a system under study.

In the pre-industrial societies, *all* technology is self-replicating. Historically, people make technology using the materials found in their environment, generally from other organisms like trees(wood) and animals(bone) or found objects of which there is a plentiful supply. People then reproduce and teach the system of constructing such technology to the next generation, along with enough understanding of that technology that the young can in turn pass along the information when they age and are passing it along to yet another generation. This type of self-replicating technological system can exist in stable dynamic equilibrium with existing ecosystems. Trees can grow, be converted to boats to hunt in, which lead to cooking technology to cook the animals killed in the hunt, and trees and game animals can then re-grow as future generations replicate their technology.

If we take this broader view of self-replication we don't even need to restrict ourselves to humans to see self-replicating technology. The beaver dam self-replicates

quite easily and indeed before the rise of modern society one can imagine the dam itself as a self-replicating entity which uses the beaver as a vector, but which transforms the landscape vastly in excess of what one might imagine possible for a single small animal. To make sense of a landscape transformed by beavers it does not make sense to either study just beavers or just dams, we much consider the self-replicating set of dam-and-beaver as a single system. The same is true with human technology.

Moving into the very early reaches of our history as we learn it today based on written records, the next self-replicating systems are religions and empires. A religious text is perhaps one of the best prototypes for self-replicating technology which can shed light on the current state of affairs. Religious texts such as the Torah or the Koran describe both a world view which gives people a "why" to what they do which includes the replication of that text. They also include the description of "how" to replicate the text, building up complex structures of education which teach the next generation of humans what they need to know to keep replicating the next down through the generations. The other main replicating structure is that of the military bureaucratic empire. An empire replicates by expanding to incorporate more and more people into the actions of further replication. This is generally done by force, which can keep growing by taking more land for more mining resources and also more people to continue to gain power to continue to ex-

pand, consuming other systems and turning them all into that one central imperial system.

The entirety of human written history can be looked at through the lens of these self-replicating systems, where the means of replication is the primary descriptor of the systems. The history of what used to be called "Christiandom" can be seen entirely through this light. The Torah was replicated by Jews for thousands of years, and was limited in its replication by Jews' only replicating it to other Jews, so the growth was limited by the biological reproduction of the people who did the replication. Then, when Christianity appeared, the same text was suddenly being replicated by the people of one of the vastest military empires every built, Rome. Ultimately this replication consumed Rome and became the Holy Roman Empire which among other things was a vast replication machine for the scripture. Then, due to technological advancements, it became possible to replicate that scripture mechanically in bulk with the printing press, and we see another explosion of change in that world where the press and how it replicated religious scripture caused some very radical change. As Western capitalism developed, the King James version of the Bible, printed in bulk on mechanical presses defined the beliefs of the military empires that then went on to conquer the globe(singling out the British Empire, followed by the American as the most powerful of the lot). Nothing in our world today makes any sense without this story of evolving self-replication.

So now this brings us at last to the discussion at hand: self-replication in regards to modern computer technology. How do computers replicate? Just as an analysis of beaver dam replication requires understanding the trees from which sticks are harvested and the rivers which feed the beaver ponds, this analysis must include externalities that are ignored by theorists, such as investors and marketing. Modern computers exist as creations of a combination of mass market consumerism, venture capital investment, and government research and development mostly focused on the military. Every company that makes computer hardware and software today is the creation of a very specific process whereby an entrepreneur pitches a company to investors, who in turn pitch their fund to larger institutional funds like pensions. After they get funding, they grow using a very specific type of worker, the modern tech worker fully indoctrinated in a certain culture. That growth is made possible by a system of mass media that transmits the information to consume the products to the masses outside of "tech". Those masses of people both put money into the products of this creation process and also put investment capital into the financial system that funds the venture capital that creates all this. Nominally all this is enabled by the "money" system which used to be based entirely on mining of minerals, but is now based on some complex system of faith more loosely based on mining. The computer systems which out-evolve their competi-

tors are the ones that replicate the fastest. They are the ones that convince people to consume more and faster, and put more money and time into the system. The venture capitalist David Horowitz has explicitly said that in the future they are building everyone will either be forced to obey the media on these systems or will become one of the people building them. And indeed this is the logic of the self-replicating machine. People like David Horowitz have to exist in order for the machines to out-replicate other machines.

This picture presented here is of course a vast oversimplification and the product of a fairly brief and superficial analysis. It is not the purpose of this paper to create a full model of replication of modern "tech", but rather to convince the reader that *such a model is needed*, in the hopes that people will develop more accurate models that we can use to try to gain some control over this system and ultimately over our lives.

In summary, the simplest model for computers that I think we should consider now is not that of the Turing machine but of the advertising machine which exists for the sole purpose of convincing people to consume more advertising machines. This might take the form of presenting PowerPoint to investors, using computers to train the workforce to build the technology, or spreading an article in the tech press about some new technology, but in most cases it is just direct advertising to the consumer. But in all cases the primary characteristic which

determines form is replication. This is why evolution of machines has favored more and more of the machine being screen, with the highest possible pixel density and color contrast, rather than maximal computation power. Pixels are what sell pixels.

We must also distinguish between viral replication and independent replication, although the line is blurry. Viral replication assumes a fixed system in which the information replicates. For instance, information can replicate itself within a commercial social media platform like Facebook or Twitter just as influenza can replicate in a host human, but this does not replicate the *system*. The means of replication of a system such as Facebook is in fact not replication of content, but the whole system including venture capital, technical labor, media to sell it, the legal framework to enforce the power of the company, etc. It is this full system replication that we are concerned with here, not the replication of information within such systems, known colloquially as "memes".

3. The *Potential* Power of the Open Web for Self-Replication

What is the Web? The Web is not the Internet. The Internet is a network that connects almost all computers in the world, both physically and with some software protocols. This network traces its origins to the network

of military and academic(but military-funded) computers that emerged from ARPA back before the modern commercial Internet, going back to the end of the 1960s. The Internet can in principle be used to exchange any information and treats information in the same abstract sense as in the models of theoretical computer science discussed above.

The World Wide Web was initially the creation of one person: Tim Berners Lee. It was initially created as a directory for the large science institution Lee worked for(CERN). The Web works beautifully on the Internet and the Internet is what made it huge, but it is not the same thing. The Web is a system for encoding information for and by humans to communicate with other humans. It includes human readable code designed to create universal documents which link to each other and can contain images of all kinds and text in all languages, creating a sort of universal document in a universal language in which all of humanity can communicate. While the modern Web is commercialized and increasingly not open to all users by default, this is a choice we have made that can be un-made. In principle any computer *can* be both a web server and a web browser. If we call the "open web" the collection of all web files which are openly viewable to anyone connected to the Network, the open web can in theory physically grow to include every computer in existence using the existing physical telecommunications network.

Let us now estimate the size of this potential Open Web network. We suppose that given the multiple billions of smart phones, laptops, servers in server farms, embedded systems, supercomputers, etc., that the total number of potential web servers is of order 100 per person. We then round human world population up to 10 billion, and estimate that there are 1 trillion total potential servers on the Open Web. Given that even a modest cell phone has a few gigabytes, but many servers and deep storage computers have terabytes, we can round up a little and say the average server can host 10 gigabytes of data. If a file like this one(this paper, the one you are reading rightnow) is 100 kilobytes to 1 megabyte, let's round to 1 meg and say there are 10,000 documents like this one(they could in theory all be math papers) for each server. So the total number of documents per person is 1,000,000. But we as a society *share* these documents, so in some sense what we all have is not 1 million but 1 million times 10 billion or 10 quadrillion documents(10,000,000,000,000,000). The informational universe in which we construct this new mathematics consists of this network of 10 quadrillion linked documents.

This universe of information exists on web servers which can in general be made to run code that edits and replicates the documents. Thus *every* document in this universe of information can self-replicate and be edited in situ. If this is all on the Open Web with code that can be edited by anyone on the Web, all users can constantly

edit all documents, so potentially we have 10 billion people all simultaneously editing 10 quadrillion files all of which are able to instantly self-replicate from any node on the Network to any other Node. This vast network effect can create power in the same what that billions of brain cells with massive interconnection, creating a document of greater power than any that has ever been possible before. The power of such an open system will be so vast that it will make no sense to have any private data. Without any property on the Open Web, things can replicate freely, and the increased value will be so great that it will consume private property online. This evolution will be physical as the value to the physical caretaker of a physical web server becomes greater to participate in the Open Web than to keep it in the commercial web. Note that if we try to simply write down the number of ways that these documents can point to each other to self-replicate, since each document can replicate from *any* other document in the collection of documents(and any number can replicate from any one other document), the number of ways they can be structured is the number of documents to the power of itself. This is 10 to the 16 to the power of 10 to the 16. 10 to the power of a few hundred is already exceeding the number of estimated protons in the known universe. So one can make similar arguments about this system as people make in regards to quantum computation systems: we can even for a very small network build something that is totally impossible

to simulate on a classical computer.

The power of a fully self-replicating and evolving Open Web on this scale is that documents can describe the replication of *physical* things, and the replication of the documents can include replication of the things. If things we use in our lives are replicated rather than purchased or mined, it changes the basic assumptions about what value is. Note that like "set", "thing" is used in a maximally general sense to include things like "a feeling of awe at the largeness of a tree" or "the tendency of cats with white fur to cause a change in the appearance of black clothing". The word "thing" is used here as a placeholder for *anything* which human language can be made to describe or point to.

In order to build these documents we must first define the idea of what exactly a self-replicating document is, and how it fits into more general concepts of self-replication. To that end we will take an excursion into the math known as set theory, which is the next section of this paper.

4. Self Replicating Sets/Documents

Motivation and definitions

Set theory, is the mathematical study of sets. Sets are simply "collections of objects". The idea of a set as a collection of "objects" considers the idea of the "ob-

ject" at a level of generality perhaps shocking to non-mathematicians. "Object" here can mean *anything*. Mathematicians generally mean by "any object" any object which a mathematician might talk about. However in principle it can be anything that anyone might talk about(as we seek to generalize these ideas beyond mathematics). For the purpose of this work we well define a generalized object to be anything which human language can possibly describe. Any word, symbol, or collection of words or symbols which point to something–that something is an "object". And a "set" is just a collection of such general objects.

The notion of a generalized object is familiar to modern computer programmers who use the idea of "object oriented programming" to create generalized objects which are used to build linguistic handles in human language on computer programs. Thus a modern programmer might define something abstract like "shopping-cart" for an e-commerce website, and then that object will have properties like "list of objects" and "total price". We choose to take the path taken by foundational mathematics and have our basic concept from which all other concepts will be built be the collection of objects(which are themselves objects) be the fundamental idea.

In order to understand the motivation of this work it is necessary to trace very briefly the history of set theory. Through the end of the 19th and beginning of the 20th century there was a vast effort by some of the

most brilliant mathematicians in the world to construct a universal mathematical theory base on the theory of sets and symbolic logic. Axioms were proposed, used to prove things, argued about, and re-written. The goal was to base *all of mathematics* on the axioms of set theory, and to go from there to a universal system of truth in which statements may all be proven to be true or false.

People like Bertrand Russel pointed out that such systems can create paradoxes that make it impossible to create a self-consistent system of logic/truth/math. This paradox can be summarized by considering the "list of lists that do not list themselves". The list defined here is a list of lists. Is this list on itself? If it is, it cannot be by the definition of itself. If it is not, it must be by the same reason. In spite of having publicly stated this paradox, Russel and his co-author Alfred North Whitehead wrote what is now considered a seminal work in mathematics *Principia Mathematica*(not to be confused with a book by Isaac Newton of almost the same name), which attempted to create such a universal basis of mathematics. While the achievements of 20th century set theory, logic and analysis are fantastic and useful, they ultimately failed in their goals, and this was proven mathematically by Kurt Goedel in 1931.

In the post-Goedel world we should take for granted that no universal logical construction can be built which defines truth and falsehood without contradiction. Goedel's proof presented a fork in the road intellectually. We could

have used this as the sign to go back through mathematics, accept contradiction as part of life, and build a math based on desired outcomes. In some sense this is what society did by mostly ignoring the work of most post-war mathematics(with some very narrow exceptions like number theory for cryptography). While professional mathematicians took the opposite fork, building increasingly complex systems based on each other, where a vast tower of ideas linked by formal logic built up from the axiomatic set theory of the early 1900s to create a bridge to nowhere. The sole purpose of most mathematical concepts and theory today are to advance the career of working mathematicians. We forget, both outside and inside mathematics, that people used to believe ideas in math actually *mattered*. We also forget that mathematics has for thousands of years been one of the most powerful tools the human mind has for understanding and interacting with our world, and indeed mathematicians have traditionally played a central role in the largest power structures throughout history.

Having proved that a universal truth machine cannot exist, we may now abandon the project of early 20th century mathematicians such as Russel and Whitehead and proceed to reconstruct axioms of set theory based on a *desired outcome*. This system will not be judged on its ability to prove theorems, eliminate logical contradictions, or get tenure for math professors. It will be judged *only* on its ability to improve the human condition. It

is time, finally, after almost a full century, to inherit the true legacy of Goedel.

Right now all of humanity is locked into one giant self-replicating set which has as elements all of industrial society. The purpose of this work is to create a set theory which enables people to construct sets which create the maximum amount of human freedom. To that end, we seek to make sets have elements that are defined as generally as possible and also which always have the *desires* of the creator of the set as an element. When we move forward replicating these sets in the new society we build, each act of replication involves also replicating this desire. We therefore only replicate that which transmits a desire that we consent to and agree with for some reason.

Let us make some statements here about the sets we want to define. We define self-replicating sets as sets which contain as elements the means to replicate themselves. We also state that since our goal is to create the objects of our desire with our mathematics that whatever that desire or intent will always also be an element of the set. We maintain the tradition of both formal set theory and object notation on computer science and define sets in writing by listing elements separated by commas and contained in "curly braces" "{" and "}".

We also state that in general the sets we will construct will have a primary element, the replication of which is the purpose of the set. There might be many other elements and subsets which are needed for replication, but

the *primary element* is defined as the element the replication of which represents the *primary intent of the creator of the set*. We thus make the human will, desire, or intent a fundamental element of our entire set theory. To distinguish our set theory from that of mathematics as it exists today we coin the term "set magic" to be the theory of sets which contain both the desires of the creator of the set and the means to replicate the entire set. This is loosely based off of the quasi-secular use of the word "magic" from both chaos magic and Thelema magick to indicate the attempt to impose the human will on the world we find around us.

Thus a way to express the most general possible self-replicating set from our newly defined set magic is:

```
set = {
    desire,
    object of desire,
    means to replicate this entire set
 }
```

Note also that in order for the whole set to replicate, the desire of the creator must also replicate. This is what in our existing system is called "marketing" and "sales". Without first convincing another mind to share our desire for replication, it will not happen. The power of the open web is thus not just about replicating documents but replicating the desire to replicate documents–in mod-

ern parlance, just marketing. The commercial web has proven excellent at this, and the open web has the potential to vastly improve on that.

We further state from the outset that all self-replicating sets have externalities. These will be elements or subsets of elements which draw on the resources outside any given "closed" system in order to replicate. Thus rather than attempting to build a conceptually closed system and then finding that it is not closed as did earlier set theorists, we accept that we are building a less formal construction that will always have an externality and that we must describe what this is in order to properly define a set. The number of degrees of freedom of this externality is what limits the overall degrees of freedom of a system. For example, if an element of a set is "text reading system" that can be on any of many different technologies. Whereas if something has an externality "lithium ion polymer batteries", the entire system is reliant on that one technology and any threats to large scale extraction of lithium from the Earth are threats to the whole system. We thus seek to constantly struggle to improve on the externalities, with the ultimate goal for them to be in equilibrium with the living Earth.

A self-replicating document is an example of a self-replicating set(a "document" is just a collection of symbols, hence a set whose elements are symbols). This document is created as itself a self-replicating document according to the prototype we propose for the whole sys-

tem. We now set forth to define the set which is this paper in theoretical terms, then to describe all the subsets which together make for a self-replicating set which can be evolved into other self-replicating sets. We now proceed to formally define the set which is this document.

This Set

The prototype self-replicating set we define in this paper is itself this paper, and is formally defined as follows:

```
On Self Replicating Sets[This Paper] = {
    The desire of the author to describe self-replicating set
    A description of self-replicating sets,
    The means to replicate this set
}
```

The first of these will always be a part of the system of sets we are constructing here: desire. All sets are defined with an element which maintains the desire of the author/creator/artist, and this is maintained as sets replicate to ensure that sets only replicate with the intent/consent of someone. Furthermore, we separate the thing being replicated from the means to replicate it. Elements are generally themselves sets which are broken down into more finely defined elements. This paper is part of the "description", but what formally is "this paper"? We seek to define it as a set, or a collection of ele-

ments contained in “description of self-replicating sets”. This includes the following elements:

```
Description = {
    narrative structure,
    definitions,
    digital text document,
    pdf document
}
```

Now again we break off the replication methods from the thing being replicated. Replication is to happen on the Open Web. That is, it must have self-contained and self-replicating code that can copy itself from any web server to any other, and be edited after being copied, then copied again from the new instance so that the information is totally decentralized and evolves naturally as it’s copied and edited. We also need this document to include instructions in human language(English in this particular instance) on how to replicate the whole system, by either buying a domain and setting up paid web hosting or building a physically local web server to server the files. This document will contain that information. Furthermore, the replicator set must include the other media that are used to replicate the whole set, including content on commercial social media.

```
Replication = {
```

```
    code replication,
    server replication,
    pdf replication,
    in-person pitches and classes and discussions,
    media: email, post cards, posters, signs, commercial soci
}
```

Code and replication of code

The elements of the self-replicating code that can propagate this document across the Open web are as follows:

- replicator.php
- dna.txt
- filesaver.php
- fileloader.php
- `README.md`
- editor.php
- `index.html`
- `pageeditor.html`
- dnagenerator.php
- text2php.php

All of this code can be edited using the program `editor.php` which runs on any server that has php installed(most web servers). The file dna.txt represents this list of files, which replicator.php uses to fetch them and copy them to a new server(see below). README is

the text of this document itself, and the name of that file is set by the standards used on Github so that by default any self-replicating document that's put up on Github has its content readable immediately. The format of the README file is by default Markdown. The save and load scripts are required to edit files on the server, and pageeditor is the page that uses these files to edit the main manuscript README.md. All php files are stored as .txt files so that they can be readable and easily accessed from the open web. The program text2php.php copies all the files in the php directory to the main web directory and changes the file extension from .txt to .php so that they can run. The file editor.php edits the copies in the /php directory, and then running text2php makes those programs executable.

Server replication

In order for replication to take place from server to server we need the ability to "colonize" new web servers with this code. This is done in any of several ways. Right now the main way is to buy a domain(usually about 10 dollars for the first year), pay for web hosting(5-20 dollars per month), then put the replicator program on the new server and run it. The second method used is to put a web server on a Raspberry Pi, a computer that can be bought for as little as 35 dollars and fit in a pocket which is excellent for serving web files over a local network.

This allows for a grey area open web that is open to anyone on a local wifi network, and can see the rest of the Open Web but cannot be accessed by users outside that wifi network. In either case, the replication of the server consists of placing the file "replicator.php" in the main web directory of the new server, then pointing a web browser to [your new servers web address]/replicator.php. This will then cause the program to run, copying the rest of the system and linking to the main page which will display the newly replicated document.

The technical details of this process must be described briefly here in order for this document to truly self-replicate. We recommend buying a domain and getting shared hosting on dreamhost because they have proven to work with this code and are affordable and not a scam. One can also use any of several free web hosting options, including 000webhost. In both cases you will find a file editor screen(this can be a little frustrating to find but always exists) which will allow you to create new files and edit them. Use this to create the file replicator.php and copy the code from here into it, save it and close it. On the Raspberry Pi, replication starts by making a new flash card with the operating system on it. One must then install the web server and php language using this set of instructions. After that one can move to the main web directory, copy all the files, change permissions, delete the existing page, and run the replicator:

```
cd /var/www/html
sudo rm index.html
sudo curl -o replicator.php https://raw.githubusercontent
php replicator.php
sudo chmod -R 0777 *
hostname -I
```

Once the whole thing has been copied, to have the *next* copy be a copy of this copy and not its predecessor, use the code editor to edit the file replicator.php so that it points to the global url for the dna on your new page, not the previous one. This is done manually for now. So if you do not do it, the next copy will be not of the new document but of its predecessor. Note also that once this system is installed anyone can run any code on your server, so no private or personal data of any kind should ever share a server with this system. This system assumes that no private data exists unless it's on a physically isolated wifi network, and it must remain separate from the private Internet(especially anything with e-commerce, as stealing financial data would be trivial if that is ever done).

Another good practice as one works with files on the open web is that since we must assume all files might be deleted at any time, but we want the current copy to remain readable not just by us but by all users on the Network, we constantly back up text to free and open but non-editable paste sites like pastebin.com. To

fork the whole code, one can edit on a live Open Web server, back up to a pastebin, and copy. But one can also create a github repository, copy the code locally to your hard drive, edit it on there, run dnagenerator.php to make a new dna file, then point your replicator to your github repository so that many copies can be made without the original being corrupted(using a hybrid between the password-protected space of Github and the Open Web which copies files from there). Note that while one can use any code editor for editing the local copy, we can keep a consistent system with the Open Web servers by running php -S localhost:8000 at the command line(I assume people who fork the code in this way know what this means) and connecting a browser to http://localhost:8000 to edit *in situ*.

Pdf document

As useful as the Open Web is, it is also useful to have documents in a traditional format which is compatible with physical paper printers to make copies we can carry outside of digital readers. To do this we have several options. We can print from the browser(which will look bad), we can convert to Microsoft Word which will corrupt the file and also look bad, or we can use the LaTeX system which will look great but take more work. For this initial instance we focus on the LaTeX version. As this document replicates and evolves hopefully more

skilled users than the initial author will make smoother systems for conversion but right now a combination of the Haskell library "pandoc" and manual editing of the output file are the easiest way to convert from markdown to LaTeX. A document produced in this way is included in the replicator of this set.

```
pandoc -o paper.tex README.md
pdflatex paper.tex
```

5. Generalization and Social Implications

All ideas which we desire to propagate on the Open Web need all the typical means of use of media to try to convince others to replicate the document. In some cases we seek to explain the whole thing in a tiny capsule of information, as with the "elevator pitch". In other cases we seek to show by example the power of these ideas. The media we expect will be used in the spread of this system include *all* media in the most general sense, including physical things like machines which carry various writing on them. Any physical thing can have a domain name on them which points to a document which describes how to replicate the physical thing. This extremely generalized definition of self-replicating document(as a type of self-replicating set) means any physical object can be thought of as a self-replicating document. Combined with the

"Open Web" defined above, this can create an entire universe of useful things we can use to make up the fabric of our lives, building sets to live in which are independent of the existing industrial order.

Ultimately if we can build sets that we have the capability to evolve based on our desires, we can push that evolution toward what we call a "technological complete set". That is a set which describes a full self-replicating system that we can live in, which can exist in equilibrium with its environment just as pre-industrial societies did before the whole world was consumed by one very destructive set. In a complete set, the people who live in that set(we place ourselves conceptually into the set) have everything they need for a good life such as medicine, abundant food, clean water, the ability to live in a comfortable temperature etc. In addition, a set is not complete if it is out of equilibrium: if a set requires constantly destroying things and not replacing them to exist it is not complete. As we look to the future this is possible in a way that is totally different from what was possible before the rise of industrial society. A future complete set based society has no reason to mine, since the quantity of material that has been extracted from the Earth and processed into very ordered structures is more than enough for a large human population to live on indefinitely.

This Technological Complete Set does not need to be designed and built by any one person or group. All that

is needed to achieve such a set is to build sets which people have the capacity to evolve based on desire, and to impart into a group of people the desire to achieve this set(given the assumption that such a set is physically attainable). This document is meant to describe such a set, but also to serve as a seed which the author hopes will evolve in such a direction. By itself it is probably insufficient to build a large complex system, but it provides a prototype for a number of other self-replicating sets which we will construct and replicate over the Open Web. It is the nature of such sets that as they are created and released into the wild, they will all build on each other with network effects and that a small amount of exponential growth early on can create very large effects as the system evolves.

What is next in this program? The self-replicating sets which are currently being created and released are largely about replicating symbols. Symbols and logos play a very powerful role in how our minds process the world around us. The ability to create a self-replicating symbol which has some intention imposed on it is one of the most powerful forces in our world today. This is the power corporations wield with their brands, logos, and marketing messages. By building systems for very rapidly creating symbols, giving them meaning, and replicating them, we empower the masses with this same power. The media for the symbols includes artistic tools for physical media creation(stationary, wall art, post-

cards, signs) and digital media creation(vector graphics of simple geometric logos). From this power, we hope to build a fire that consumes the media landscape and transforms the nature of human existence on this planet.

Chapter 16

Icons

Chapter 17

Trash Robot

17.1 Trash Robot

What is Trash Robot?

Trash Robot is a self-replicating set.

The set contains all the media required to replicate it. It also contains an Operator, described here, who is the actual human who learns the system and teaches other people to replicate it. The set is described here, but some of the most minute details of replication will be left out for brevity because they are all described in technical detail in the media transmitted with each instance of any part of Geometron. That is, any domain on the Street Network, running the Organic Web and Geometron will have scrolls and maps with guide a user

through the whole process of replication of this entire set as well as how to use all parts of it. In this book it is sufficient to simply tell the reader how to find these scrolls and an Operator who can teach them, and they will have the information required to *get* the information to fully replicate the system and become an Operator.

This chapter describes the structure of the SRS called Trash Robot. It does not explicitly have all the instructions, but it *describes* all the elements, how they are used, and what they are for. Extreme detail is reserved for the Scrolls.

Everything here refers back to Ontology. We build this thing, show how to replicate it, show how replication can benefit the replicator, which stimulates further replication. Build things and sell them. Build things and use them. Build things and share them for mutual aid and benefit.

Metabusiness. SRS. Thing. Organic Media. Geometron vehicle. Maker swarm. Methodology. Trash Magic(self replicating media from trash, seed of full complete set).

Some roles in Trash Robot:

Printer Operator. You carry a printer with you, and print coins on demand.

Symbol Artist. Build sets of things, assign them symbols, find those symbols in image form, trace into robot code, save as sharable data, share with people, participate in symbol markets with other people around the

world.

Token replicator. Carry bags of tokens, stamps, prints, use them to print replications of themselves as well as the tools to further replicate them and sell them or exchange them with other people, again as a global network of self-replicating media.

the terminal, the EXACT terminal, carried by Trash Robots is an element of Trash Robot.

The Operator is an element of Trash Robot.

The Street Network can be used to sell a huge range of goods and services. Trash Robot is a specific collection of self-replicating technologies which can also be sold directly via the Street Network. It includes a list of things which will be described here, which readers are encouraged to copy and sell. But more importantly it provides a prototype for a new type of product which can be built at scale using the decentralized craft mode of production. As soon as this book is released and we build a swarm to start living Geometron and selling the elements of Trash Robot and bartering for the services provided by the Network we will also start building totally new products using this method. We will work to build products which have not exact equivalent on the market in the consumer economy and which would be very difficult to replicate in that space for various logistical reasons of how technology scales. IN particular we need to explore in this chapter the economics of enclosures and how cardboard and duct tape and geometron built by crafters and im-

mediately sold at a profit on a one-at-a-time basis differs from injection molding of enclosures at scale.

The Operator

We are the Operators. We share the elements of Trash Robot with everyone we interact with. We do whatever we can to help people as much as we can using our tools from Trash Robot, Geometron, the Street Network, and the Organic Web. We wear the Trash Robot brand, and carry all the tools of Trash Robot.

The Open Brand

rainbow and googly eyes. Soft black textiles. Felt. Geometric constructions from Geometron. Use of geometry in cardboard fabrication, HDPE sheets. Modularity. Things carried in cloth bags. Blocky geometric font. duct tape and cardboard and sticks. Things built from trash. Images of the things: box, flag, shirt, pants, bags with robots. The fashion brand.

The ArtBox

As with everything in Trash Robot the ArtBox is a self-replicating set. The elements are the Trash Ties, the Tape Snake, a purse which is the box itself, and a collection of standard art supplies which sit in the box and

are used to make another box. Trash Ties are 6 foot long sections of cotton clothesline, with the ends duct taped. We buy a whole package of 100 feet of clothesline, and cut into 6 foot sections, duct taping where the cut will be before cutting so each section has both ends taped. We then buy a collection of colored duct tapes which includes black, red, orange, yellow, green, blue, purple, and pink, or one more if it is available. We thread the Trash Tie through all of them twice, and join it to itself with a square knot. This is the Tape Snake. We carry this around with us to anyplace where we are working on replication.

The contents of the box are box cutters, sharpie markers, a laser cut acrylic shape set as documented elsewhere in this chapter, and a bag of googly eyes(which can be purchased in bulk online for very little money), scissors, and a ruler, also documented elsewhere here.

This is a nice purse! If you make one and have the supplies you can go make one with a friend and they can make more and so on. And as you all make them and grow the swarm you can use them to carry around art supplies for other art projects as a super stylish art supplies purse until people notice and want to buy it, then a market will exist and we will sell them to whoever wants them. These are so striking and nice to use and cost so little to make and are so easy to copy that they can be a big part of what replicates the whole Trash Robot system which in turn helps replicate all of Geometron

Figure 17.1: Elements of the ArtBox and a Tape Snake.

Figure 17.2: Cut out a 6 inch equilateral triangle from thick corrugated cardboard.

and the Street Network.

We will make millions of these to give away, share, or sell. The can be freely shared in our Network with-

Figure 17.3: Copy so that you have a total of 10 triangles.

Figure 17.4: Stitch together the net pattern as shown.

Figure 17.5: Assemble into octahedron with three top triangles meeting in a tetrahedron top the octahedron.

Figure 17.6: Use all colors of duct tape to create rainbow skin effect as shown. This is the prototypical Trash Robot rainbow open brand after we add the mouth and googly eyes.

Figure 17.7: Stab holes as shown in each of the top three triangles at the upper tips.

Figure 17.8: Thread a Trash Tie, as described above, through the the holes as shown and tie a square knot.

Figure 17.9: Complete the box design with a black duct tape piece for the mouth and crookedly placed asymmetric googly eyes. Multiple faces can all have the face design repeated. Now we replicate all the contents, finding another sharpie, another box cutter, etc. and put a whole kit required to make another box in our box to use to make more boxes. We also replicate the Tape Snake so that the new box has a Snake for replication, and pass some extra Trash Ties along for the next replication as well.

out money or even barter and just used as needed for other Geometron constructions, but can always be sold to people not in the Network.

Laser Cut Acrylic

The Geometron workflow for creating .svg and .png files is designed among other things to make it as easy as possible to create files to use in commercial laser cutters. These have been used both in the kind of laser cutter available in maker spaces and from the professional service at Ponoko.com(these require slightly different files).

Laser neon green acrylic shapes in the basic Shape Set and the 6 inch Geometron ruler are key elements of our system, and replicate along with the rest of the ArtBox. Making custom or specialized geometry tools like this is a very good metabusiness for Trash Robot. We can sell the standard Shape Set, sell the Golden Triangle, the 6 inch Ruler, the Geometron Protractor, Penrose Tiles, custom spray stencils using the laser cut stencil font, and just custom arbitrary laser cut shapes for design clients.

A good design target for custom spray stencils is the domain name which points to the local Geometron Raspberry Pi Server on the Street Network.

Show each design. Link to global files. How to use Dremel laser cutter, how to use Ponoko.

skeletron

poles, geometry, tetrahedron, octahedron, icosahedron, tripods, flag poles, S Hooks, photos of constructions

textiles

Textile products are an important part of Trash Robot. We make clothes with the Trash Robot open brand, as well as bags for the various parts of the Trash Robot system and Flags which are used to point to domains on the Street Network, advertise the Trash Robot, and form a surface on which to place Trash Robot elements.

The basic aesthetic of all textile elements of Trash Robot is black cotton flannel with bright rainbow colored felt sewn on. Everything in this system is designed to be tied using Trash Ties. These are either 6 foot long clothesline sections terminated in duct tape or 18 inch sections of black nylon parachute cord with ends burned. Letters printed on the Flags are in the special Trash Robot felt font, which is made from square cutouts of each color of the rainbow in order. Letters are formed by cutting minimal, simple, geometric pieces out from full squares of felt. Black sweatpants are sewn with the letters, in all caps spelling out "TRASH ROBOT", with one word going down each leg of the pants. The black cotton t-shirt has the Trash Robot brand logo of the rainbow rectangle with googly eyes and a black rectangle for

a mouth.

Bags are used to carry the Raspberry Pi Geometron Terminal as described in the Street Network, to carry the clay icon tablet printer described in this section, and to carry the three types of clay tablet: prints, stamps, and finished tokens. Prints are printed directly by the printer robot, and are in small black flannel bags with a symbol for the alchemy element "earth"

the font. flags. Bags. Clothes. Methodology. Fashion business on the street. all the layouts of the designs. Waving the flag with the pole. Power of the flag to direct traffic to pages in the Network over the cardboard sign, legitimacy via the open brand.

The exact dimensions of the various bags are given in the Scrolls that replicate with all instances of Trash Robot, as well as on physical Replicator cards distributed with the physical set Trash Robot.

The icon token printer robot

The exact description of how to replicate the printer is given in the Scrolls distributed with Geometron/Trash Robot. These scrolls should be readable on any local Raspberry Pi server running the latest version, as well as any Street domain like www.us29.xyz or global domains like www.trashrobot.org. The exact means of operation of the printer is also documented in the Scrolls. In this manuscript we discuss how this all fits together, and rely

Figure 17.10: Trash Tie. A universal building block of the Trash Robot system is the Trash Tie, which is either a 6 foot long segment of clothesline with duct taped ends or an 18 inch long black nylon parachute cord with burned ends.

Figure 17.11: Domain Flag. Flags are squares approximately 36 inches on a side with a 1 inch hem(cut a 38 inch square out).

Figure 17.12: Trash Robot Flag. This flag says "TRASH ROBOT" just like the Trash Robot sweatpants.

Figure 17.13: Trash Robot Flag font. All letters shown. Copy this font to create domain names for the domain name flag. Note that web addresses are always case insensitive, so that for example, www.us29.xyz is WWW.US29.XYZ.

Figure 17.14: Trash Robot Flag font. All letters shown. Copy this font to create domain names for the domain name flag. Note that web addresses are always case insensitive, so that for example, www.us29.xyz is WWW.US29.XYZ.

on the instances of the software to transmit the details of replication.

Build the brain.

Build the controller.

Build the mechanicals.

Workflow: image feeds, align, trace, share, print, bake, stamp, replicate, paint and sand, build into sets, share and sell, make pendants, stitch into clothing and accessories, make jewelry. Make more robots. Donate, share, and sell robots.

The symbolic currency, creating sets, the sigil boards, sets as fashion, sets as symbolic currency which can mean *anything*. This symbolic currency goes back to the previous chapter, where we discuss the general idea of self-replicating sets. For any given set of generalized "things" in the sense of the previous chapter, each thing can be assigned a symbol. That symbol can be traced into the robot code which drives the printer. All those symbols can be stored as a token feed which is saved and shared with other users, who can then print them all out. So an abstract set can always be represented by a collection of Geometron icon symbols shared over a network and also a collection of clay painted icon tokens, carried in a bag as described in the Textiles section of this chapter, and replicated using more clay without use of a robot. So we can represent any self-replicating set with a set of symbols shared over both a local and global network and *also* a collection of physical tokens in a bag which are like coins

in a coin purse. Attractive and convenient coin purses which contain symbols representing a self-replicating set which themselves are self-replicating media can form the basis of a very powerful symbolic economy based on sets, networks, and symbols rather than numbers.

The creation of a self-replicating symbolic currency which has both physical and digital versions is a fundamentally different type of currency than money. Because it all self-replicates, the idea of property simply does not make sense, because the value of replication is reversed. If I own a plot of land or a car, its value is based on how hard it is to replicate. If it were possible to freely replicate a billion acres of the best land imaginable with warp drive to get to the best location, your million dollar plot of land will be worthless. If someone builds a free car that grows naturally in a forest and runs on the sun, your 40,000 dollar gas car will be worthless. But if you build a robot from trash to fabricate useful things, the more people copy the robot the more value there is in the whole network. If a billion people copy my robot, I can get a million of the cleverest makers in the world all improving the design, and what I end up using a few years from now light years beyond anything I could have even dreamed of. So the more people replicate a free thing the *more* that thing is worth to me, even as the creator of the thing. This makes the way value works the opposite of property. Value goes up with abundance rather than scarcity. If we make the choice to value abundance,

meaning the freeness of things, then that is what will end up with value. But this replication also means that any numerical unit of value will very quickly break as the supply of value exceeds the total currency available again and again(essentially deflation).

Some notes on the pi version, how this can be a pi driven robot.

Arduino Generic Shield

This is a stub. It expands into many technologies but we document just the board and most basic of codes. Future technologies will reference this.

```
Trash Robot = {
    ArtBox,
    Trash Tie,
    Tape Snake,
    Token Printer,
    Terminal,
    Brand,
    Textile,
    Shop,
    skeletron,
    constructions,
    Operator
}
```

```
constructions = {
    duct tape,
    cardboard,
    trash ties,
    HDPE sheets
}

shop = {
    ArtBox purse,
    Trash Robot branded clothes,
    laser cut acrylic,
    token printer kits,
    tokens,
    pendants,
    printed bottle caps,
    terminal install
}

laser cut acrylic = {
    golden triangles,
    penrose tiles,
    full set,
    ruler,
    protractor,
    custom shape,
    spray stencils
}

printer = {
```

```
    brain,
    controller,
    mechanicals,
    workflow
}
```

Construct the brain, controller and mechanicals according to the Scrolls.

```
    printer workflow = {
        build, share, sell, use printers, following instructi
        use Geometron server to follow the rest of this workf
        image feeds,
        aligner,
        trace,
        share feed with other users, save, copy, paste, share
        load code into Arduino, print in clay tablet, bake it
        use print to create stamp, sell or give away or use,
        use stamp to create both coin-like tokens and pendant
    }
```