¡h1¿Geometron: A Language for Graphical Communication¡/h1¿

¡b¿Abstract.¡/b¿ This is a white paper describing the basic construction of Geometron and how it can be of use specifically to the quantum information community. Geometron is a metalanguage based on discrete geometric actions used to build symbols in a systematic way. The basis of Geometron is the Geometron Virtual Machine(GVM), a purely geometric virtual machine in which points in space map to actions which have geometric meaning. This paper first motivates the work by surveying existing languages used in the quantum information community, then describes the ideas behind the GVM and gives examples of how it is used in practical work. A deep dive section into the exact structure and implementation of the GVM used here is then provided. Finally the workflow is described which members of the QI community can immediately start to implement, and then future work is described which can leverage the power of geometry based thinking about quantum information for both better communication between the people in the community and better ways to program quantum computers more efficiently.

1 1. Motivation

Graphical communication is a fundamental component of building any new technology. Often the more unfamiliar a technology is, the more we need to rely on simple cartoons and diagrams to tell others about it. This communication between the people who create the new ideas and other people is fundamental, and can either make or break a new technology.

Our existing theoretical models for understanding new machines tend to ignore how machines propagate through the human population, but I believe this is a mistake. These questions get banished from technical theoretical problems because we have arbitrarily divided who thinks about what problems up into "engineering", "humanities", "marketing", etc. We like to think that quantum computing will either succeed or fail based on some intrinsic technical merit, but none of us will ever succeed in this venture without clear communication between the advocates of the technology and the sponsors(funding agents, investors, upper managers). Those people expect very simple and clear stories to be told in a mostly graphical format via PowerPoint, and that story must be more compelling than that of competing technologies. Thus clear graphical communication is fundamental to the very survival of a new technology.

Graphical communication is also used to propagate our ideas outside of

narrow sub-fields, expanding the number of minds focused on the problem. What starts as a tiny minority of presenters at the American Physical Society March Meeting has taken over a huge fraction of all physics conferences, thanks in part to spreading the ideas of quantum information effectively in these venues. This also depends on clear and simple graphics, as that is the only way to get a message across in a 10 minute talk.

As a technology transitions from the lab to the market place, graphical communication becomes even more critical, as creators of technology vie for a very limited attention span of potential users, almost entirely via web-based communication. Quantum computing is now at a stage where it is attracting both user dollars and investor dollars, and a quick survey of leading commercial quantum computer ventures shows that very simple sequences of pictorial communication are becoming dominant.

Quantum Mechanics and quantum information science have always been particularly dependent on custom graphical languages. Two examples that immediately come to mind are Feynman diagrams and the Bloch Sphere. In both cases, ideas are expressed for which we also have words, numerical simulations and algebraic equations. However in both cases, reducing all that to a simple cartoon totally changes how we are able to engage with problems, and makes a huge difference both in the ability to learn and do research in the respective problems they address.

As quantum computing technology has become a part of the computer industry over the last few years, we have seen a number of efforts to build languages targeted specifically at programming quantum processors. These languages are generally intended as part of a "quantum stack" which goes from the physical qubit layer up to the highest level where the end user will ultimately be solving their problem in quantum chemistry or number theory or whatever. Rigetti Computing, IBM, Microsoft, and several other companies have all created such languages. All of them are very clearly heavily influenced by the history of software written for classical information processors. The languages tend to look like C, Python, or Fortran, and to think about information the same way we all learned in classical computing, in terms of numbers and gates.

I would argue that all these approaches are limited by their divorce from the geometric nature of quatum information. At the very least, adding a geometric layer to the quantum stack will allow for rapid, automated documentation to be created which makes the underlying code easier to interact with. By having a language based on discrete symbols which correspond to gates, we can translate between that language, the actual physical gates, and the code in the various languages people have already written. Thus

at the very least, building the quantum Geometron language will allow for rapid, universal documentation to be created, making it easier for everyone using the various competing languages to rapidly communicate ideas online, and internally document their work.

This will be discussed in detail in another white paper, but part of the long term goal of this work on building geometric languages for quantum information is to create a fundamentally geometric way of interacting with the processors, without dealing directly with numbers at all. Ideally, a problem we seek to solve can be posed geometrically, which is straightforward in instances such as protein folding or certain types of optimization. Then, using purely geometry-based languages similar to the one presented here, we build maps from the problem space to a Web browser and from the Web browser to the Hilbert space of the quantum information processor.

To understand why a new language for dealing with graphics in quantum information science is useful, it's worth examining the existing workflows used in the field for graphical communication. We generally present our pitches for funding and support via PowerPoint slides, typically with simple graphics based on a lot of conceptual cartoons and diagrams. Similarly we present to colleagues at conferences and colloquia in this format, but generally with greated detail in the technical diagrams. We also communicate with investors, perspective customers(for for profit ventures) and with the general public using rapid communication over the Web, often via a marketing or PR department. New types of diagram are generally created in vector format using Adobe Illustrator. These graphics are then propagated through PowerPoint, web design software, and other parts of the Adobe suite before being used to communicate.

Art replication, memes, the Internet, and how machines replicate.

Existing art workflows, and how they hinder overall development of quantum information technology.

Geometric nature of Quantum Information Science and the problems it addresses

2 2. The Geometron Virtual Machine

The Geometron Virtual Machine is a function which maps discrete points in space to actions which manipulate discrete geometry. That is the most general possible definition, which encompasses an infinite number of potential instances we might choose to create. The GVM I'll be describing here is the Geometron Hypercube, which works in a Web Browser, and is written

entirely in JavaScript.

Rather than trying to formalize the math of this idea, I will dive in and describe the specifics of the Geometron Hypercube used in all the work documented here. This consists of two cubes, each with 8x8x8 = 512 cells. Each cell has an address, which consists of three coordinates for the cell and one for which cube the cell is in. Thus the first cell of the first cube is 0, the highest cell in that cube is 0777, and in the second cube it's 0 to 01777. I use a leading zero here to denote that these numbers are base 8. Using base 8 in JavaScript is easy, as JavaScript recognizes the leading zero, as do most c-like languages. One of the two cubes represents actions, and the other represents symbols for the actions in the first cube. The contents of each cell are arrays of addresses of cells.

The choices made here are in some sense arbitrary: 8 could just as well be 4 or 16 or even some random prime number. But the point of this approach to language design is to choose numbers and structures which are well suited to easily being interacted with by the human mind. A 8x8x8 cube can be divided into "tablets", each of which is 8X in size.

The action cube consists of actions or sequences of actions which do something on the screen of a computer using discrete geometry. They all use a set of global geometric variables which are manipulated during the drawing of a glyph in much the same way that registers are used in a traditional microprocessor. These global variables include position of a cursor, angle of the cursor, a length scale, a factor for increasing or decreasing the length scale, a symmetry on which to carry out rotations, and a few other useful parameters.

Glyphs consist of sequences of actions. They are expressed as strings in the form of base 8 addresses separated by commas, e.g. "0300,0341,0333,0341,". Some addresses in the action cube consist of glyphs, which in turn refernce other glyphs or actions. Thus many levels of recursion are possible and used routinely. This is a huge part of the power of this system over existing graphical systems like Adobe Illustrator. Adding glyphs to the shape table and using those glyphs is not like just copying and pasting shapes in Illustrator. It is in fact more like building up cell hierarchy in a electronic layout CAD program.

For each action in the action cube, be it a discrete JavaScript action or a glyph made of of other actions of glyphs, there is always a corresponding symbol glyph in the symbol cube, in the address space 01000-01777. These symbols are *always* glyphs made up of a sequence of actions(rather than an action in JavaScript).

Both the action and symbol cube have a full 64 + 32 = 96 cells for

the printable ASCII characters from space to tilde. The standard printable ASCII characters go from space at 040 octal(0x20 hex, 32 decimal) to tilde at 0176(0x7E hex, 126 decimal), with 0177 being backspace which we leave as a "do nothing" operation in the Geometron hypercube. The symbols at 01040 through 01176 are glyphs which constitute a font, stored in a file called font.txt. The symbols in a font can use a full shape table of up to 64 specialized geometron action sequences, creating new fonts very quickly and with total flexibility even with no coding skills. This is particularly useful for creating new fonts for specialized purposes such as programming easily into a robot or lithography tool. As with Chinese characters, the stroke order maters, each symbol is spelled with a specific order of actions, and the order matters.

The Geometron Action Cube is divided up into layers, called tablets, each of which is a 8x8 array, much like a chess board. Those tablets are then broken up into rows of 8 actions, which we attempt to group in a rational way.

The bottom tablet in address space from 0 to 077 consists of "root magick" actions from 06 through 037 and the first part of the ASCII from 040 to 077. Root magick consists of actions which are not purely geometric in nature, which interact with the Geometron software. These are more specific to the

hypercube, tablets, scales, symmetries, actions, javascript vs symbol, comparison to Logo, role of ASCII,

Figure 1: Figure x.

Figure 2: Figure x.

Figure 3: Figure x.

Figure 4: Figure x.

Figure 5: Figure x.

Figure 6: Figure x.

Figure 7: Figure x.

Figure 8: Figure x.

Overview of discrete geometry: movements, scales, rotations, actions

Mechanics of the GVM in the browser: javascript, organization of code, canvas and SVG elements

3 3. Examples

Polygons, stars, lines in polygons, arrows

Building the perfect inductor

building a circuit

creating a new type of circuit element: example of Devoret amplifier

building a set of gates, a diagram with gates

fractals: building the Koch curve

4 4. Workflow and Applications

Public domain graphics library for quantum information science

Working with MS office suite workflow

Working with Adobe Illustrator and Inkscape

Building and sharing new graphical languages, new symbols. Using pastebin, sharing code on your page.

Running on a local machine, Connecting with Jupyter notebooks and LaTex

Hire me to set up a local node of Network

5 Future Work

microblogging for real time technical graphical communication

Building geometric languages to program quantum computers without the numerical/English based intermediary.

Decentralized art feed network, I will help you replicate the code on your server, then adapt it for your specific applications, and train your people to use it, build custom keyboards, link to other nodes on Quantum AR

Creating direct connections between the web browser and geometry of Hilbert space, to cut out the "middle man" between a quantum user and quantum circuit. Real time quantum controls.

Create a language for protein structure using a custom GVM, another GVM for a many qubit hilbert space, then building a browser based UI for constructing purely geometric quantum algorithms which map problem space to hilbert space.

GVM for Galois groups, geometrizing the problems of factorization so that people can develop new algorithms for factoring using GVM.