

Dscript 2D Notation

Dscript 2D Notation is an Extension on Dscript 2D Alphabetical Writing System.

The Dscript Alphabetical writing system allows alphabetical strings(words or phrases) to bend, curve, curl, and fork in 2D, enabling a string of letters to be compressed into a glyph, a long string, or fit in just about any shape desired. Turning words into glyphs still means we are dealing with a 1D language. There is still just one “string” of words, no forks in the language itself.

Dscript Notation is designed to address this issue, and bring the language itself into a more fully 2D environment. We currently handle 1D language limitation with various standardized notation systems. Dscript Notation will attempt to find alternatives and seek unification while also attempting to achieve an intuitive nature for someone already familiar with the original format.

There are many standards of notations, so it will take some time before they have all been assimilated. Here is the first round of imports. These standards are actually quite large, and have not been completely assimilated yet, but they have begun to take shape and are developing quite nicely, with intuitive efficiency popping up consistently.

1. **Nodes & Visuals (Page 2)**
Nodes allow 2D connections and visuals allow simple illustration
2. **Wrappers & Chemistry (page 8)**
Text wrappers identify data types, Chemistry notation for chemicals and elements
3. **Programming (page 12)**
Programming notation for conditionals and code
4. **Electronics Diagrams & Logic (page 14)**
For circuit diagrams and logic design

Dscript Notation is still in its infancy.

Dscript Alphabetical built a foundation for strings of letters to exist more dynamically in 2D space. Dscript Notation will attempt to build a second layer for language and meaning to begin entering 2D

This version of Dscript Notation is based on English.

Many elements are derived from words and abbreviations

A whole different version could easily arise by using Dscript in a different language

My choice to use English was more or less arbitrary(my native tongue). It is entirely possible that based on lexicon and alphabetical spellings, other languages may provide different or more or less efficiencies or have different or more or less advantages.

If you are not familiar with Dscript 2D alphabetical writing system you may want to skim this first


[Dscript Alphabetical Introduction : http://dscript.org/dscript.pdf](http://dscript.org/dscript.pdf)

Nodes & Visuals

There are 2 main types of Nodes. "Dot Nodes" and "O Nodes". Nodes connect 2 or more objects (here an object means a glyph, word, section of text, or notation)

O-Nodes evolved from the Dscript character Y  and the Dscript word "so" 

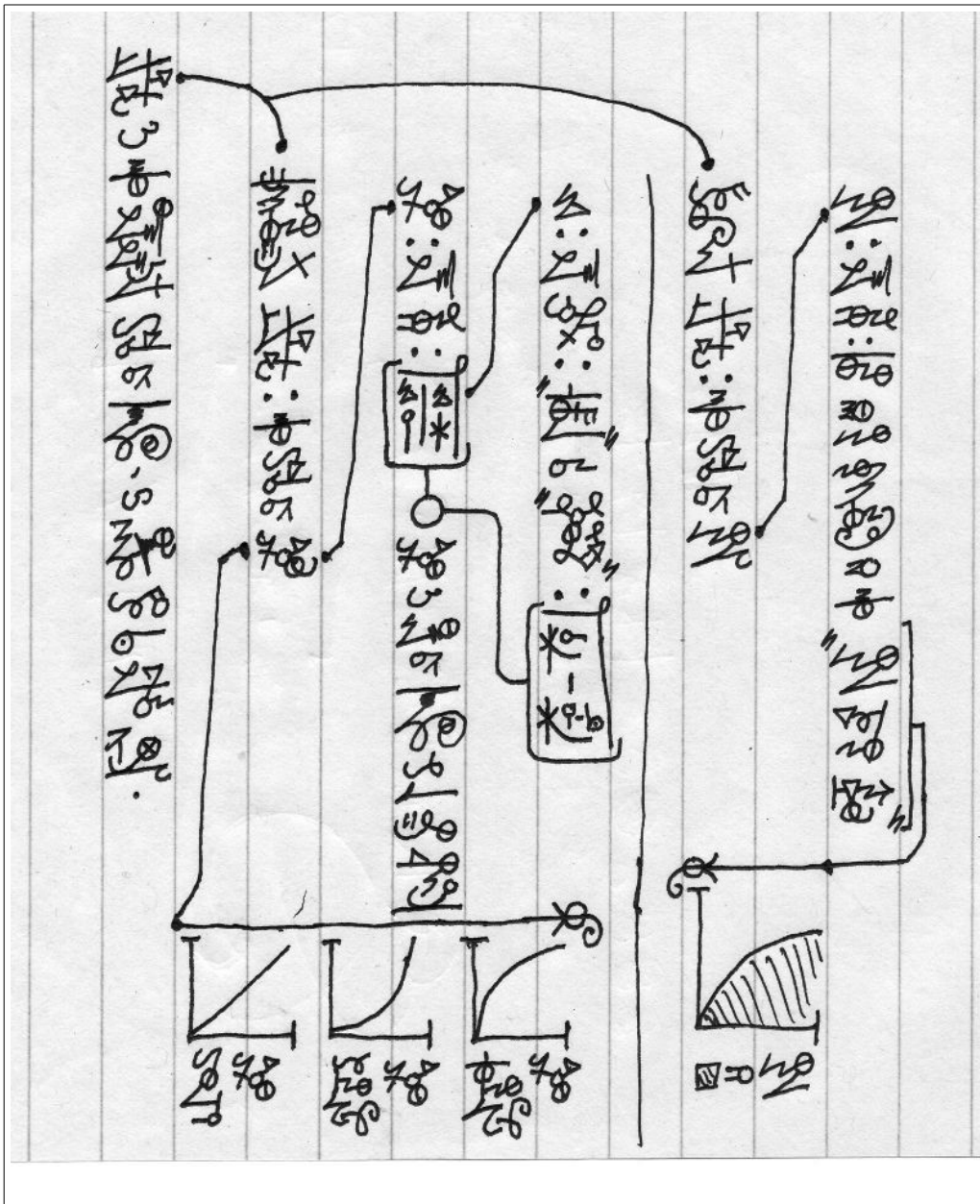
The Circle of an O-Node indicates the "result". You can choose to think of it as

A  B = "A that's why/so/therefore B" or "Why B?.. because A!"

Dot Nodes Types


1. Simple connectors : indicate "definition", "clarification", "further explanation"
2. Fork Connectors : indicate "subset", "parent-child"

O-Nodes indicate "causal", "reason for", "extrapolation".



In this example there are several Dot-Node connections and one O-Node that connects to 2 "causes".

You will also see many 90 degree rotated ":" characters. These are used for the same meaning they hold in most notation standards. (definition of, value of, next value, segment boundary)

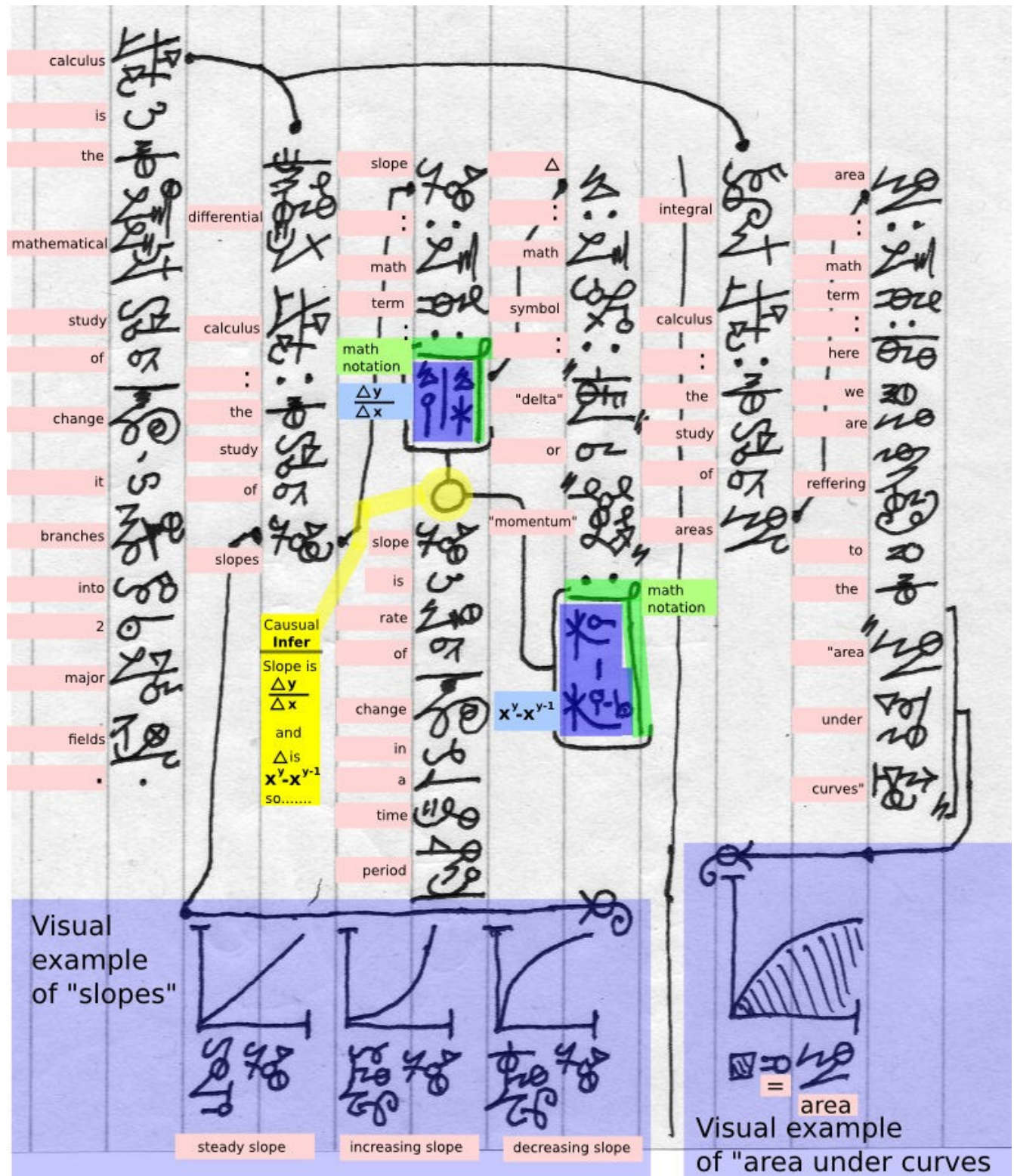
Note the Dscript Alphabetical "veg"  at the end of some nodes. This stands for "visual example". It identifies visual designs/charts/graphs/etc..., it traces out the writing space they occupy, and connects back to the term meant to be visually described.

Here we have the same example with Latin script labels.



Notice the vertical indentation after the first line, this indicates the same as usual. (subset/parent-child)

The Dot-Node in the indentation space describes the parent-child relationship between "calculus"(parent) and "differential calculus" & "integral calculus"(children)

Other Dot-Nodes connect objects(terms and concepts) to further explanation and example of that object



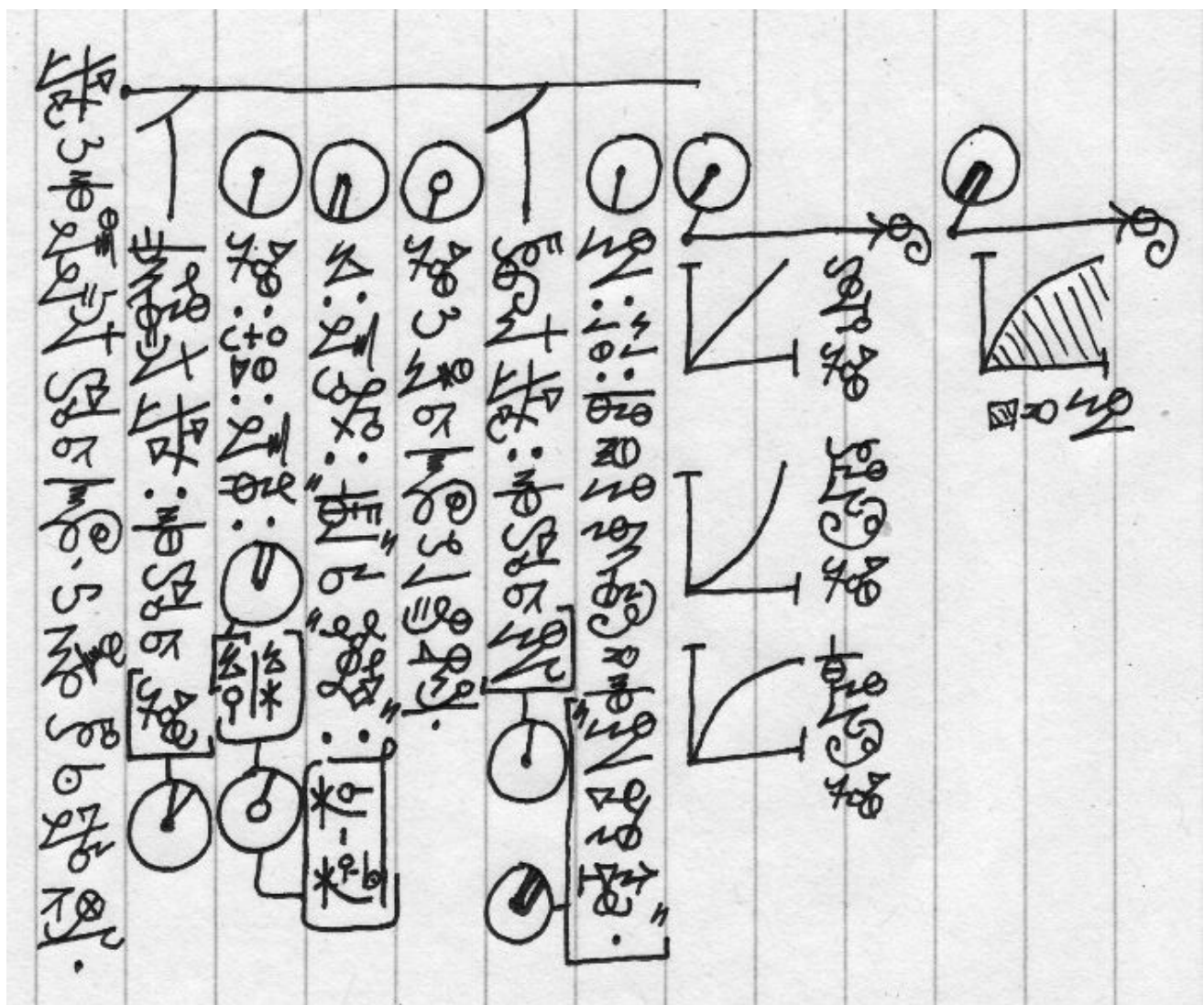
The example on the previous page was one of the first versions worked on, there is one minor revision to the math notation since then.

The Delta was originally designed as  to avoid ambiguity with the Dscript letter Q. After further consideration, this ambiguity is not a problem as the letter Q is rarely used alone. Any future glyph conflicts should be easy to resolved. Now the delta can be written as .

*("momentum" refers to technical analysis of time series vocab where it's equivalent to delta)

Jump Nodes and Forks

Regular lines to connect nodes is fine, if there is plenty of space between lines of text. But if the text is not double spaced it is nearly impossible. This is solved with "Jump-Nodes". Jump nodes allow the line connecting nodes to "jump over 2D obstacles". When you are following a node path and arrive at a jump node, mentally extend the line inside the jump node until you reach a jump node that "catches" it.



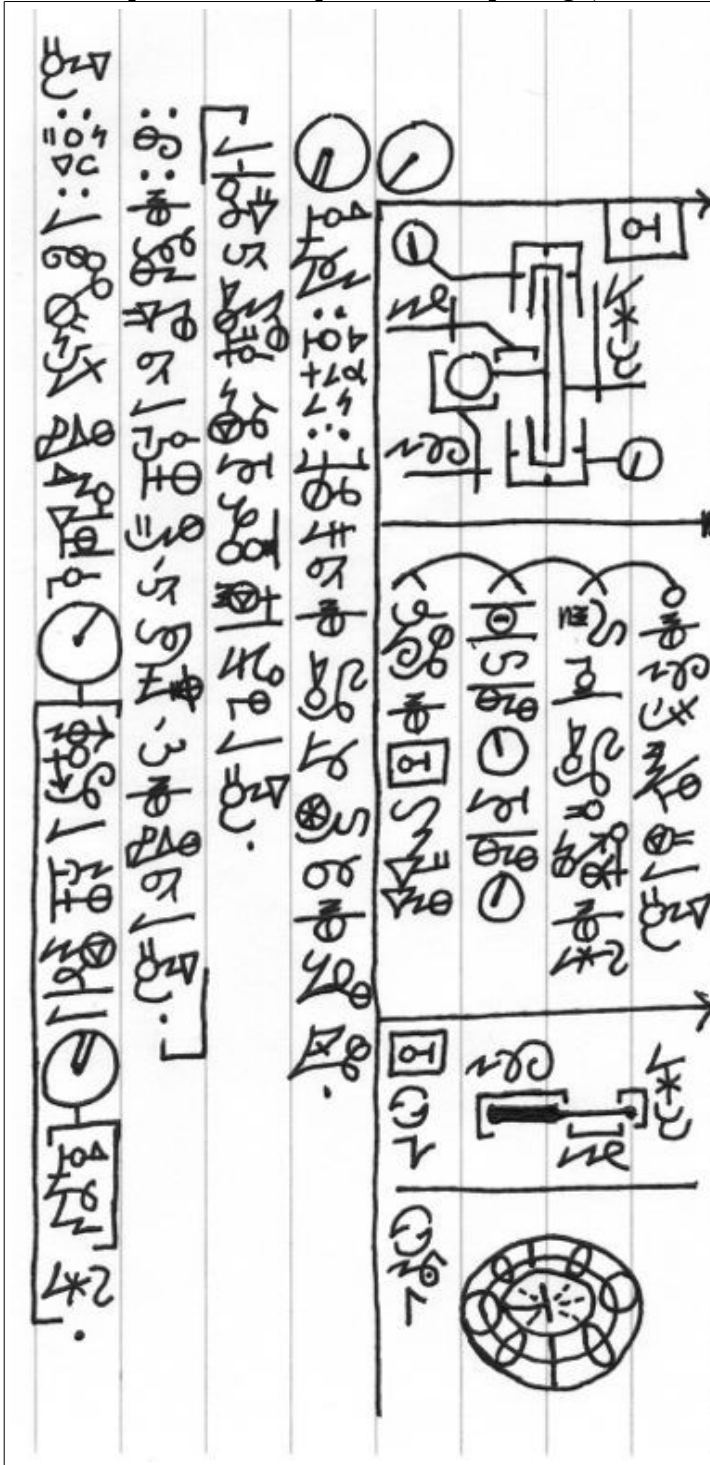
A dot in the center indicates a "Dot-Jump-Node" and a circle in the center is an "O-Jump-node" when jump nodes are numerous and dense, it could become easy to jump to an incorrect node. Ensure that the target node's line also matches the incoming trajectory and correct type (Dot to Dot, O to O). Double or triple lines help, double lines only catch other double lines, triples only catch triples. The example on the previous page uses the exact same content as the first example. There is only one

minor semantical difference. The “parent-child” relationship is defined more clearly. Here we use the “Fork Joint” to indicate the hierarchical status and relationship. The first example using regular node joints relies on indentation and context to clarify the relationship and hierarchical details.

The Fork Joint is an Alphabetical Dscript letter F



Also, the previous example adds the spelling (letters in sequence) as an element for each defined word.



This example reads:

Torus : t o r u s : a geometrical shape produced by JUMP-DOT-NODE (revolving a circle around a JUMP-DOT-NODE(coplanar) axis) .

: eg : the inner tube of a bicycle tire, if inflated, is the shape of a torus.

[column connector]

a donut if perfectly round and smooth would also be a torus

JUMP-DOT-NODE(

coplanar : c o p l a n a r : when all of the points can exist on the same plane.)

The boxed off visual/text sections on the right are explained on the next page.

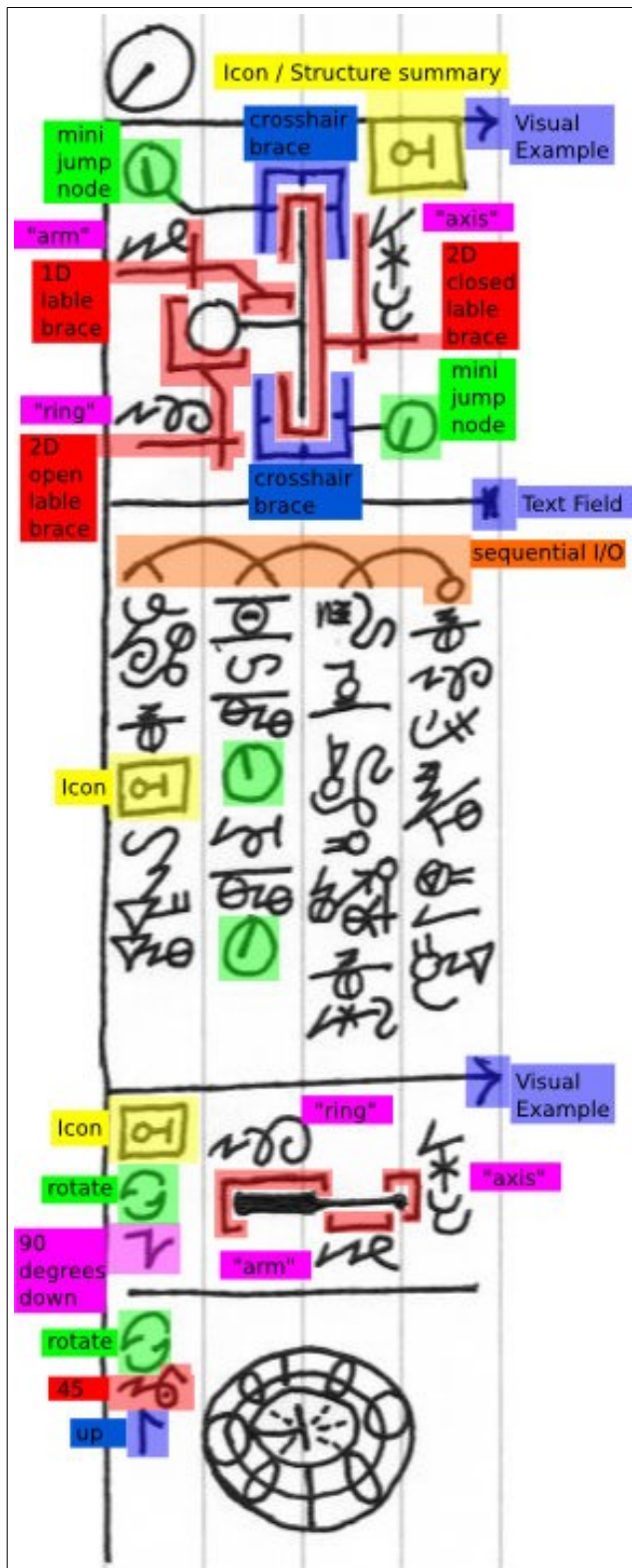
Notice at the bottom of the second line there s a “hook” that runs vertically upwards between lines. And that there is a reverse “hook” on the top of the third line. This is called a “column merger” or “line merger”.

A column-merger means that the “line break” should be semantically ignored.

Also notice the jump nodes are nested inside the text line, and even within each other. This can require some planning, or erasing if you use a pencil/digital pen. You can alternatively first write the text, add the nodes in empty space below, and squeeze node lines between the lines of text, but it can get quite messy that way,

*Drawing circles takes practice&patience

Visual Notation



First notice the “Visual Example” notation has been abbreviated from “VEG” to just “V”.

A “T” is also used to identify fields that switch back to the default text notation standard.

At the top (in yellow) is an “Icon”. This serves as a glyph to easily “call” or “reference” the visual object from standard text notation. It can also be used to provide a “clean view” of the visual object without labels/nodes/etc... (it is not necessary that it is the same structure as the visual object, you could make an icon for calling that does not resemble the actual object)

In green there are “Mini Jump Nodes”. These are less accurate than full sized Jump Nodes, but they come in handy in tight spaces..

The Mini Jump Nodes attach to the blue “Cross-hair Braces”. Treat these as you would any cross-hair. It is used to point at locations that cannot be reached by normal braces and pointers.




The Labels used are “Full Labels”. The label is inside is a Dscript letter L(+) which connects to a brace that identifies the space it is labeling.

At the top of the Text Field is a Sequential I/O Node Array. Sequential I/O Node Arrays are a type of O-Node. O-Nodes indicate Causal relationships, Sequential I/O Node Arrays Indicate the sequence of causes for an effect. A sequence of inputs, in the correct order, produces an output . Here they indicate instructions, like the steps of a recipe.

Instructions Read:

- In-1 – imagine the [ICON] structure
- In-2 – hold it here [MINI JUMP NODE] and here [MINI JUMP NODE]
- In-3 – twist both points to revolve the axis
- Out – the ring will trace out a torus

continued next page.....

I/O nodes are derived from the causal O-Node and the Dscript letters for I  (upside-down when above the inputs ) and O  (at the output)

Here we have 4 sequential inputs and one output



When the sequence elements are neatly aligned the I-cups can be produced by overlapping the curving node lines and adding an appendage to the first input line rather than drawing the upside-down I-cup at each input.



	<p>The last visual sections include viewpoint commands.</p> <ul style="list-style-type: none"> -Start top left (but could be in other corners) *yellow* calls previous structure via its icon. *green* issue rotate command *purple* 90 degrees down (combo glyph) <p>The next section starts with “rotate”. Same visual block, so we will take the view point we just had and move from there.</p> <ul style="list-style-type: none"> *green* rotate *red* 45 (Dscript base 100 number) *blue* Up symbol (simple arrow based)
--	---

Rotation here refers to only 2 of the 3 axis.
 Imagine a sheet of paper on a desk. We use the 2 axis that require lifting the paper.
 The axis that would just spin the paper on the table, is ignored for the commands.

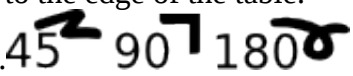
Another way to look at it, is that we only use the rotation axis that can be drawn on the paper, the axis that would be a stick puncturing the paper, is ignored for these commands.

So in the first picture we rotate **90 degrees down**.

That means you will be looking at the paper at its bottom edge (closest edge to you sitting in front of it)

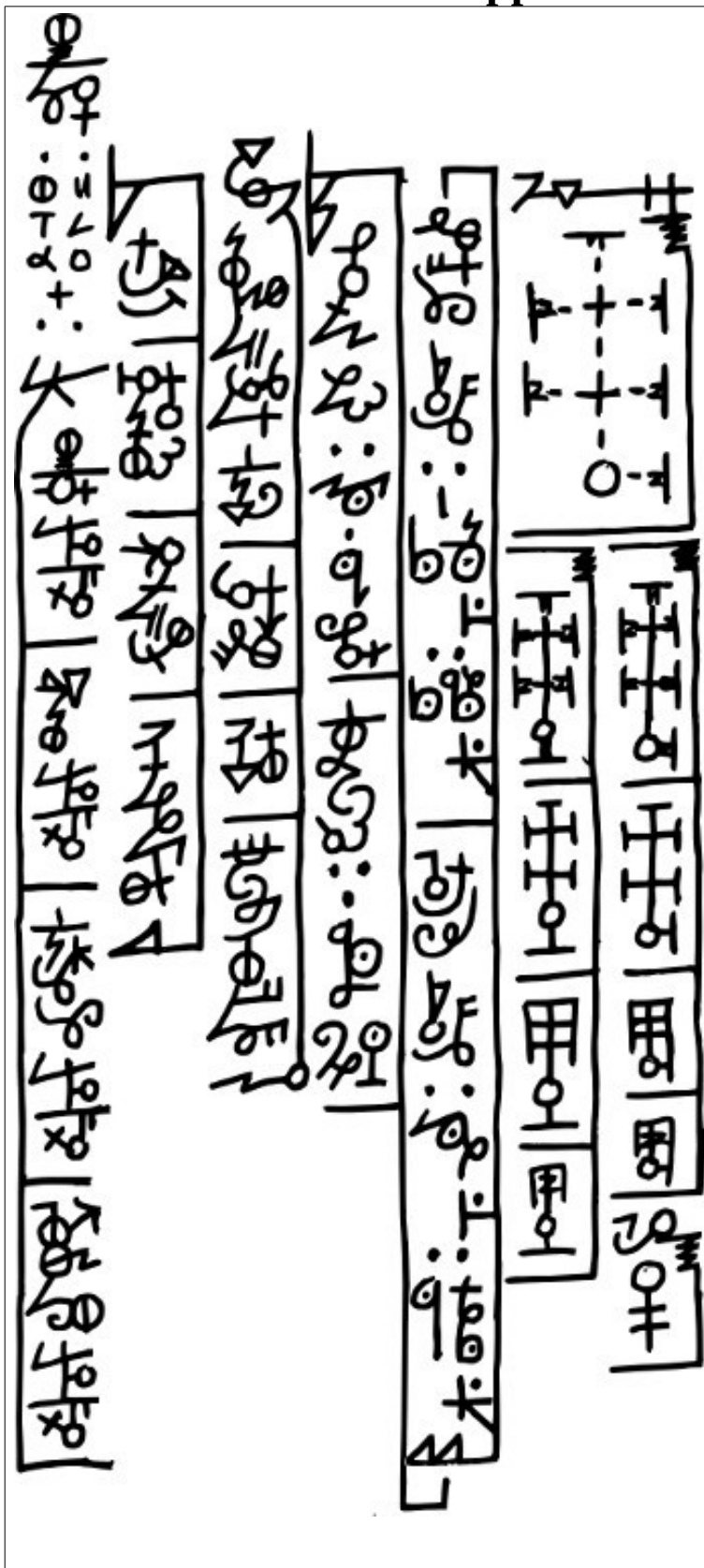


Imagine looking at the paper, which is on a table, from directly above, that is your current view point.
 90 degrees down means pull back, and bring your eyes all the way down to the edge of the table.

The number of degrees can be written out, or these shortcuts can be used. 
 The shortcut symbols can also merge with the directional arrow into a single glyph.

Knowing this you should be able to visualize the torus shape with the instructions on the previous page.

Wrappers & Chemistry



On the left here we see a simple definition for ethanol.

The text reads :

ethanol : e t h a n o l : aka(ethyl alcohol, pure alcohol, drinking alcohol, beverage alcohol)

p[properties] (liquid, colorless, volatile, flammable)

usefor[uses/applications] (recreational drug, solvent, fuel, disinfectant)

pp[physical properties] (
 molar mass : 46.07 gmol ,
 density : .79 gcm³ ,
 [column merger]
 melting point : -114°C : 159°K
 boiling point : 78°C : 351°K
)

and then on the right there are chemistry fields. (explained on next page)

The vertical indentation indicates that all of the text belongs to the “ethanol” topic.

Here we see “wrappers” being used.

A wrapper is similar to the way we use brackets in standard notation.

Standard brackets can still be used, but wrappers offer some interesting potential.

Wrappers can be used like brackets and just omit the long vertical lines connecting the start and end.

Drawling the line can be bothersome and occasionally tricky, but it offers the advantage of being able to visually grasp the 2D groupings faster and can sometimes help save time in reading.

There are 2 main kinds of wrappers, “Label Wrappers” and “Verb Wrappers”

Label Wrappers are most commonly used to identify data or information type. Segment lines can be added to list several objects.

	<p>On the left you see some Label Wrappers. The label can be a whole word spelled out or an abbreviation shortcut.</p> <p>These are three of the basic Label Wrappers. Labeling with a single letter “p” indicates “properties”. Labeling “pp” indicates “physical properties”.</p> <p>The “aka” Label Wrapper uses an existing English abbreviation fully spelled out as the label, it indicates “also known as”.</p>
--	--

These labels are descriptive information labels, and refer to the object/term/concept at hand or the one most recently mentioned/directly preceding object. If you wish to connect them to an object far away on the page, use nodes.

Verb Wrappers are wrappers that can accept more than one type of input and express a relationship between those inputs.

<p>A simple example of a verb wrapper is the “use for” Wrapper.</p> <p>The Dscript words “use” and “for” are combined and 2 large gaps are formed, one inside each word.</p> <p>Each gap has text put into it “use X for Y” Here X=“cat” and Y=“controlling mice” so “use cat for controlling mice”</p> <p>In the example on the previous page you will notice there is no first(X) input, if there is no first input then it becomes a single input label, and like a Label Wrapper it assumes the object/term/concept at hand for X.</p>	
--	--

	<p>If you break off the the X, or verb element, and use it alone, then it can describe an action or be issued like a command or instruction.</p> <p>The “use” has been broken off alone in the image on the left. X is placed inside, so we know what to use, just not how/why/when/where The Y could be either before or after. Everyday linguistic context rules apply. eg. In order to solve your mouse problem USE(cat) USE(cat) to get rid of the mice in your house.</p>
--	--

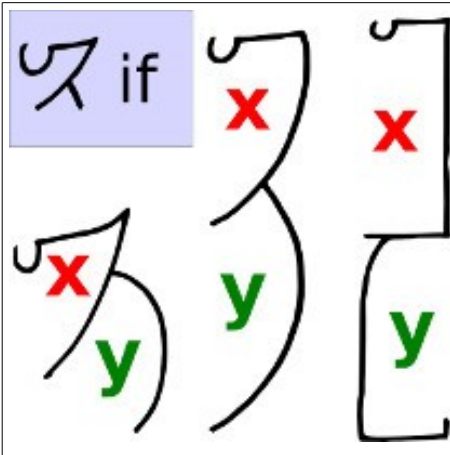
	<p>On the left and right here we have two examples of “ch”(chem) blocks.</p> <p>“ch” blocks indicate that a molecule will be defined or described.</p> <p>In “ch” blocks there is no stereo chemistry or bond strength. Only elements. The elements can connect to each other and be simplified.</p> <p>The methods of simplification and connection are numerous and have yet to be fully indexed, there are so many variables and each method raises its own unique conflicts within itself and with others.</p> <p>For now this will be considered a “semi-creative” process.</p> <p>“ch” can be called in 2 major ways.</p> <ol style="list-style-type: none"> 1. To simply act as the descriptor/name in language. 2. To define a standard simplification as a shortcut glyph. <p>Here we have two examples of usage number 2. Both for ethanol.</p> <p>First there is a molecule, with all elements fully written.</p> <p>Next it evolves towards simplicity.</p> <p>The last version can be used as a shortcut in text.</p> <p>Define it once, and then draw the shortcut instead of writing its name in text.</p>	
--	---	--

	<p>“bioch” blocks drop bond strength and stereo chemistry like “ch” blocks, but “bioch” blocks also drop hydrogens.</p> <p>This helps simplify complex, often biological, compounds and interactions. Rough metaphor: “think of hydrogens as electrons”</p> <p>Notice the 2 forms. In all Chem Block types Carbon can be drawn as a Dscript letter C(“ -” shape) or L(“+” shape)</p>
--	--

The ethanol example also uses some “measure glyphs” or “measure words”

	<p>The measure glyphs °C and °K are a dot on the right side with Dscript letters “c” and “k”</p> <p>g/mol is “gmol” in Dscript and g/cm³ is a combination of “gcm” and “3D” in Dscript</p>	
<p>The “3D” character used in g/cm³ is a modification of the Dscript number 3. It works on any Dscript number. Just take the number glyph for X. Then add the Dscript alphabetical letter D to the end of the(or one of the) lines It will then mean “XD” or “to the power of X”, where X is the number used.</p>		
<p>The D portion can also help “wrap” the number so it is clear they go together. Here are a few examples using g/cm and an cm.</p>		

Programming



Programming notation allows description of programming logic for writing code/logic/or even metaphorical use in language.

First lets looks at IF, the most fundamental programming operation

The Dscript word “if” provides 2 spaces for inserting variables
First the condition, and then the action
All of these examples read as if(x){y}“if X then Y”

This works the same as the Verb Wrappers, except these describe a decision process as opposed to an action. They could be used nested with verb wrappers as in “IF(x){ GIVE(y)TO(z) }”

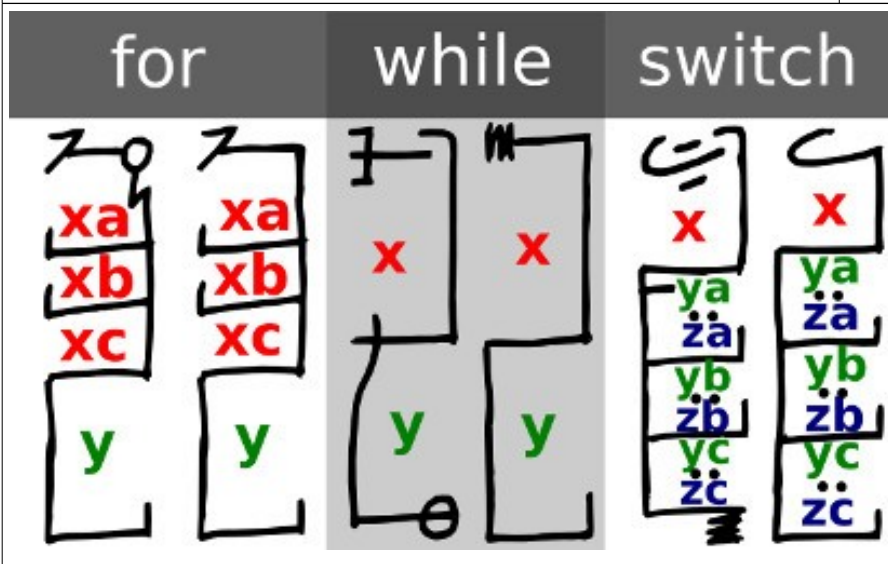
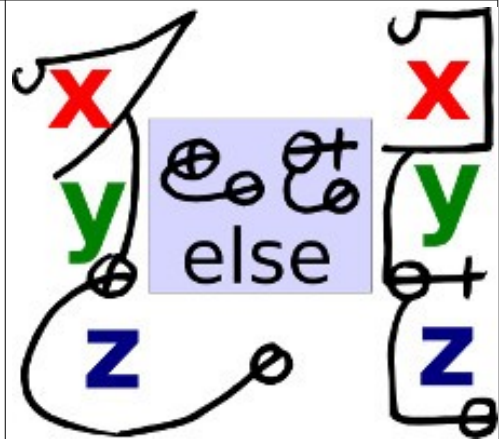
The IF is often paired with an ELSE clause.

The Dscript word “else” can be drawn in various ways, on the right in the center of the image you see 2 possible forms.

Simply attaching this to the bottom of an “if” easily produces a third space for a third value of Z.

The examples on the right describe “if X then Y, otherwise Z”

An “elseif” could be derived using the same rules, But so far cases that require “elseif” instead use “switch”without problem.



On the left are 3 more fundamental programming conditional wrappers.

Each show it's full form (all the letters of the word spelled out) and its abbreviated form.

The letters F,W, and S when used as a wrapper header are reserved for these wrappers.

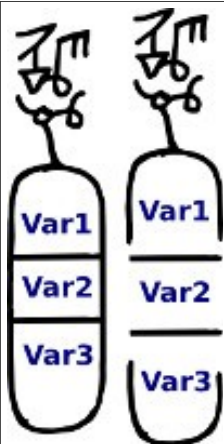
F=FOR
W=WHILE
S=SWITCH

The examples above in software code would be:

FOR : for (xa ; xb ; xc) { y }
WHILE : while (x) { y }
SWITCH switch (x){

 case ya : za; break;
 case yb : zb; break;
 case yc : zc; break;
}

*for now breaks are assumed at the end of each case
a “break glyph” may be introduced later



Next we need functions to form collections of operations.

Functions are called for action by their “name-glyph” which attaches to a block.

The block is for Input Variables or “function targets”. Variable names are separated by long horizontal bars.

The programming code written in the image on the left is:

```
function( var1 , var2 , var3 )
```

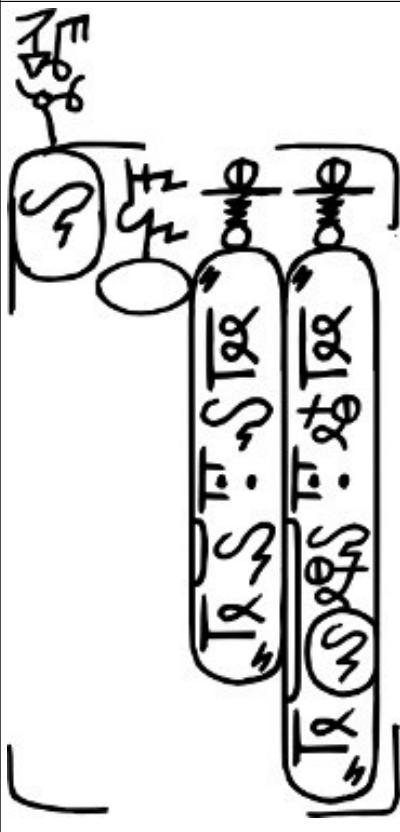
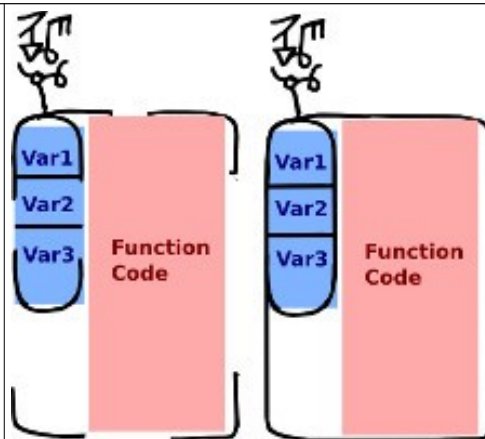
Both read exactly the same, closing off the box just aides visual identification.

Next we need to be able to define a function
Simply draw the Input Block again, somewhere where you have plenty of space and encase the Input Block with another larger block.

Extend the outer block to encase the code you want and use the variables from the Input Block in your code.

You will probably want to first only draw the top-left corner of the Code Block and add the rest when you are finished.

These read as : `function (var1,var2,var3) { function code }`



Escaping is a key element in code scripts. Especially in string mode. The example on the left reads as(in php):

```
function($str){
    clrscr();
    echo("\n\n str : $str \n ")
    echo("\n\n len : ".strlen($str). " \n ")
}
```

The basic escape is a vertical line on the side, much like the “\” but not slanted.



When in string mode you can call on variables or functions with the Code Escape, this escape has corners pointing outwards. If there is a border this escape will connect to the border and bulge outwards.

Both types of escape can, for now, be used on either side of the column, or even both at the same time(for artistic effect).

Should the need for more escaping arise it can be solved by:

- differentiating between side of column (left, right, both)
- differentiating between “corners style” of the Code Escape

*returning is done the same eg “return x”;

X											
+		*	-		^	∨	↘	↙	↗	↖	
y											
x+y	x-y	x*y	x%y	x==y	x<y	x>y	x!=y	x>=y	x<=y	x=y	x++

The simple math and programming operators are listed above.

The Dscript word “to” is used as “=” in some cases, specifically when the “=” means “set to”.

This frees up the “=” for use as the “==”.

Electronics Diagrams & Logic

drawn into line	drawn on top of line	name
		resistor
		variable resistor
		thermal resistor
		capacitor
		transistor
		diode
		LED
		inductor
		cell
		power source
		switch
		potentiometer
		AC power
		ground
		push to break
		push to close
		speaker
		crystal
		signal generator
		chip, unit

There are many electronic component types and various existing standards for drawing them in circuits.

On the left you see electronics notation for some of the most common component types.

Most are derived from the letter that starts their name. (R for “resistor”, T for “transistor”, D for “diode”, I for “inductor” etc...)

Single Pole
Single Throw

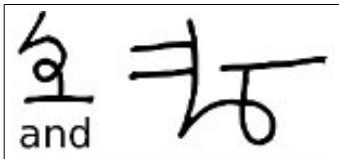
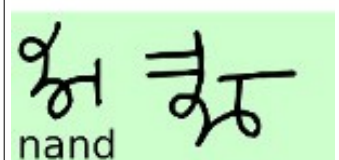
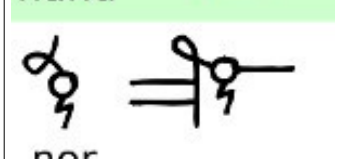
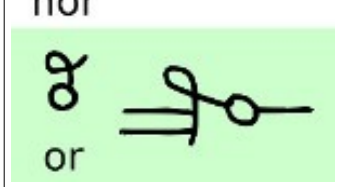
Double Pole
Double Throw

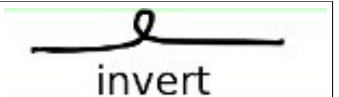
Double Pole
Single Throw

Switches are the Dscript letters “s” & “w”

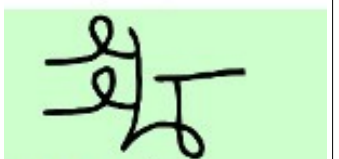
There is much to assimilate, but so far even some simple circuits are quite fun. Some might have potential use in language as “concept glyphs” or linguistic metaphor.

	<--- 2 resistors connected in parallel to a power source this glyph could be used to mean “connect(ed) in parallel”
	<--- 2 resistors connected in series to a power source, this could mean “connect in series/daisy chain”

 <p>and</p>	<p>Here on the left are the 4 key logic gates.</p> <p>They all contain the Dscript letter N, accept multiple inputs, and give one output.</p>
 <p>nand</p>	<p>A lone “n”, or loop, is an inverter.</p> <p>These simple logic gates provide all that is needed to build complex logical operation.</p>
 <p>nor</p>	<p>All the gate symbols are composed of their names written in Dscript, except for the “or gate”, it is written as “no”. This may feel counter-intuitive, but the meaning is that the “n” represents logic, and the “o” represents “or”.</p>
 <p>or</p>	<p>It is best not to connect the output to multiple destinations directly. Instead take a single output and fork it further away whenever possible, this allows them to be visually identified as logic gates quicker (“many in one out”)</p>

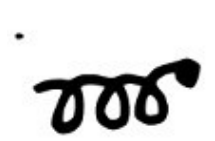









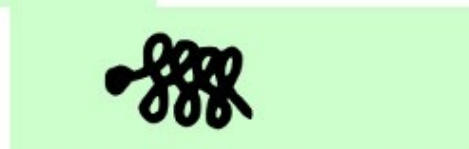


invert



AND with inputs inverted

Both electronics and logic notation are relatively new. These areas are very complex, along with chemistry and programming, these fields can be labeled as “very deep rabbit holes with no real bottom”. This is only “stage 1” so there may arise need to modify these standards in the future.

						
x0.000001	x0.01	x0.1	x10	x100	x1,000	x10,000
						
x100,000	x1,000,000	x1,000,000,000,000	x100,000,000,000,000			

Finally we have the Dscript Notation equivalent of scientific notation for large and small numbers. Start at the dot and read out the string.

- corner = X10
- loop = X100
- perpendicular line = Raise total to the power of the number of perpendicular lines
- eg. “2 loops and three perpendicular” means $(100*100) * (100*100) * (100*100)$

You can also just use measure glyphs like Kg, Km, MW, GB, etc...

Resources

Dscript Alphabetical introduction :

<http://dscript.org/dscript.pdf>

There are plenty of learning materials and examples of Dscript online
There are 2 main official websites for those interested in Dscript

Dscript.org

This site contains recent updates, examples, and materials, it is always the most current information. Dscript materials are all under the “Dscript” menu option on the site’s main menu.

Key dscript.org resources

- *Full Sized Full text art*
- *Full Size transparent Reflected text art*
- *Word Art Examples*

Dscript.org is loaded with graphics content.. all is free to copy edit and sell, no royalty, fee, etc. so please feel free to do with as you please :)

Dscript.ca

This site documents the origins and development of Dscript. There are some tools and resources, but they are somewhat outdated.

Some content is also collected on DeviantArt and Facebook. DeviantArt in particular allows storage and public access to some many high quality files so it is worth a look. (I store plenty of the reflected text art on DA)

<http://www.facebook.com/dscripting>

<http://dscript.deviantart.com/>

***If you like Dscript, you may also like*

WireScript, a 2D/3D writing system that can be written by bending wires. Works great for art, sculptures and jewelry.

<http://dscript.org/wirescript.pdf>

NailScript, A layered wrting system for writing with hammer and nails.

<http://dscript.org/nailscrip.pdf>

“Mad Science”/“Technology Art” inventions and experiments. Great DIY fun.

<http://dscript.org/inventions.pdf>



Dscript by [Matthew DeBlock](#) is licensed under a [Creative Commons Attribution 3.0 Unported License](#).

Based on a work at www.dscript.ca and www.dscript.org