This language is not to be case sensitive.

What follows is a brainstorm of what features to implement and what to call it.

I will also attach images of relevant documents, and examples.

Note in images of the papers where a circular array is calculated, 6, the number can be swapped out for whichever growth rate for the array is selected, most of the codes functionality for converting these arrays into radial projections should still remain functional.

Although between you and me, I prefer

(((Sqrt:"X", Sqrt:"Y")×(Sqrt:2))÷sqrt:(2×(X+Y)))× "radius"

Over anything involving pi, for purely personal reasons. It splits the grid into 8 circular arcs. My conversion equation uses pi however to make life easier on folks.

Anyways now to the brainstorm:

I want to include in my programming language:

+

-

÷

×

^

log:

Sqrt:

So 7, base operations.

!= not equals

i= if equals

>= if greater than or equal

<= if less than or equal

i> if greater than

i< if less than

|= if absolute value equal.

|< if absolute value less than.

|> if absolute value greater than.

|<= if absolute value less than or equal to.

|>= if absolute value greater than or equal to.

!> if not greater than

!< if not less than

So 13 comparators.

The digital root of 13 is 4.

I might name the programming language, “Language 7-4”. (Language Seven Four)

L74.

Because that could be the file extension too.

.L74

.txt

= is the same as set to but shorter.

Set: "A" to: "B" = other assignment variable.

Write: write, to, file, array

Ask: a, list, of, questions

Asknum=”N” (any single number not words don't do this)

Print: tell the user something and then continue running.

Goto @named line@

---

7 miscellaneous functions.

// single line comments

@named line@

"Named variable"

<array name<points per point<layers>>>

[Array address point]

|absolute value|

((Nested)operations)

<array name>[Array address point1]iftouch[Array address point2]

Then:

Else:

Check if two points in the same array are touching.

ie if there is a difference of 1 in accordance with the spiral algorithm functions.

---

if <array1 name> __ <array2 name>

Then:

Else:

Compare the size of two arrays.

The array functions break the pattern, but, in return, they have a great deal of power, such as the ability to create cellular automata, or to stack arrays of different sizes.

Or to read out the point value of an array using write() or print() and a loop which adds 1 to a value.

The only thing it doesn't do is write its own code.

But that would create too much complexity.

So perhaps it's better that it doesn't.

Loop[]until "X" __ "Y";

Repeats a code snippet until a condition is fulfilled.

Now... how do I... check an array to see if two points are touching?

....hmm...

Subtraction might be involved, but division might also be involved.

Subtraction is involved if two points are on the same layer, if they are |= 1, then they touch.

Division is involved if they are not on the same layer.

So I hypothesize:

Divide their distances along the spiral by the growth rate, then subtract them from eachother and set "X" to that, then if |"X"|>=0

Then if |"X"|<=1, they touch.

Else they don't touch

Else, they don't.

That's my hypothesized way to check if they touch.

P.S: for the attached image I had to misspell point because spelling it correctly triggered a calculator function.

P.P.S: a square array has an expansion of 8 and a hexagonal array has an expansion of 6, a hexagonal array is the same as a circular one. Fractal numbers are exceptable, as well, for example a 3-sphere array would likely have an expansion of 12.634~ish I'm guessing, because the packing number of a sphere is more than 12 and less than 13.

To determine if two points touch (see image)

$$\frac{p_{oimt1}}{g_{rowthRate}} - \frac{p_{oimt2}}{g_{rowthRate}}$$

To determine how long an array is based off it's layers and expansion rate (see image):

I need to convert this summation to psuedo code. So that it becomes more than useable, so it becomes usable in its native language.

$$\left(\sum_{n=1}^{L_{ayers}} e_{xpanzionRate} \cdot n \right) + 1$$

Old documents related to rendering your arrays. (For Expansion rate = 6)

Algebraic equations describing the relationships between the "Spiral coordinate system" as it pertains to the "polar coordinate system" in the second dimension

(turn back to the page with the table labeled "Key:" if you need a refresher)

"C" = ((((Log(6/"P"))/(Log(2)) / (-)1)+1)

"P" = "C"√((6 *(2^("C-1)))^"C")	"P" = ((("V" - 7)/2)+6)	"P" = (("W" - ("S" - 1)) + 4)
	"V" = ((("P" - 6)+"P")+1)	"S" = (("W" - ("P" - 6)) - 1)
		"W" = (("S" + 1) + ("P" - 6))

***/ This is pseudo code, which can be used to find the value of "C" from the value of "W"*/**

new variable: "Y"; new variable: "X"; set "Y" to: 0; set "X" to: 1;

*/ "X" will act sorta like "C" and "Y" will act sorta like "V"
over time. you'll see. it'll be cool!*/

Loop this: [Set "Y" to: "Y" + (6*(2^("X" - 1))); Set "X" to: "X" + 1;], until: "Y" is >= "W";

*/ "X" is now equal to the "C" variable,
meaning that you can calculate the values of: all the other relevant
variables described in this document.*/

Key:	
The variable representing the distance away from the origin circle in standard circular units, is: "C". (you can think of each "point"/circle-unit away from the origin as a circle on an outer concentric ring comprised of circles, all nested around the origin).	
	The variable representing the distance, away from the starting-point* of the current concentric ring, along the pseudo-circular perimeter of the current concentric ring (as described by the "C" variable), is: "S". *(The starting point is always aligned along a unified axial direction. The direction traveled can be defined as "clockwise" or "counter-clockwise". The given starting point and thus the orientation of the entire system can be transformed to point in any direction, depending on how the users/ interpreters of these maths decide to geometrically transform the outcomes of the following equations pertaining to my coordinate system)
The variable representing the number of circles/"points" that the current concentric ring [as indicated by the "C" variable] is composed of, is: "P".* *The "P" variable is like "circumference" in a standard circle/ring, however, the math seems a lot easier to me.	
	The variable representing the number of circles/"points" including, and between: the Origin and the outermost ring of circles* [as indicated by the "C" variable], is: "V". *It helps to think of the "V" variable as a measurement of a kind of volume that happens to also include the volume of whatever happens to be containing it.
The variable representing the distance traveled along a periodic spiral whose course traverses every conceivable point on the graph*, is: "W" *The "W" variable is the most important variable; the others are solely for the conversion of a given graph into other coordinate systems	