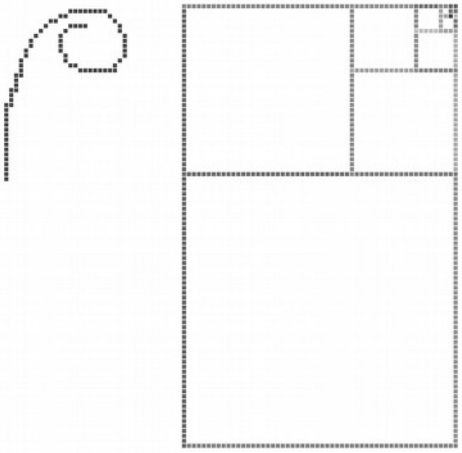
Aristo Tacoma

Art of Thinking



Vol. 2:
G15 PMN Programming for Teens

Art of Thinking, vol. II

G15 PMN PROGRAMMING FOR TEENS

Aristo Tacoma

ISBN 978-82-93128-03-8
Published by the Avenuege Library,
Yoga4d von Reusch Gamemakers, Oslo,
Norway. Print: www.print-shop.no.

INTENDED AUDIENCE

Grown-ups living life, using their minds also. Those who are living sexually in a meditative way, wishing to think about technology, programming and philosophy at the same time. "G15 PMN for Teens" is Volume II in the Art of Thinking series, a series devoted to thinking in its full and holistic format, not merely as a dry logical course, but as a biologically, organically pulsating phenomenon in which the rhythms of our genitals ties up with rhythms of our brains and minds. If this is too much for you, stick to Volume I until you have matured: it will teach you the technical fundamentals here engaged at a higher and more tantric and spiritual level.

Chapter 4 has full source-code and some comments for FCM which is part of G15 PMN Third Foundation standard. FCM is used for pattern matching in Volume 3, and in other forms robot and/or matrix programming, where the limits of what algorithms can do are touched on. Chapters 1 to 3 are juicy enough to make up for the driness of chapter 4, I should hope :)

FOREWORD

Perhaps you haven't cared all that much for programming before now, and now, you care for sex. I am not going to condemn that. You must find your own path. Sex can lead to love, and love has its own intelligence.

The high sense of ecstasy that sex can give is much higher still if the mind is not in a mess, but rather has harmony, love, clarity, tranquility, order. It is here programming can help: it's like a shower or a bath for the mind; one of many approaches to keep the mind bright.

Surely, sex is not just something that goes on between the legs, or in this or that part of the body. It is about being whole, feeling all life as an unbroken, dancing, musical flow, a unity, a throbbing unity with beauty, and with oneself. Obviously your mind must be clear, whole, having good order for sex to be full.

We are going to assume some but not much understanding of what was laid out in Volume 1, and you can always look into it to clarify that which is said here with little explanation. We will repeat some of the themes, such as the golden ratio, but in somewhat fresh ways: and we will keep on returning to the sense in which sexuality can be a living, integral part of great thinking and programming.

It is a question of talking up certain perceptions of the world and the machine and its G15 PMN language so that the sensuality of it emerges in our feeling, rather as how a sense of the world can be forged through a poem.

The approach of sexuality to doing intelligent work is natural, for those who are lucky enough to handle the intensity of sexuality. When you spend much time in some

meditative, creative state of being near-orgasm, while having concrete plans vaguely in mind, these plans sort themselves out. Anything done by this writer has been prepared in this way--including the shape of the G15 PMN language: even its innermost code, just look at how some of the core two-letter commands in PMN are programmed in the underlaying G15 assembly--a theme which we most gently touch on in this book. Several topics hinted at here are explored in some depth in upcoming books in this series.

A PRACTICAL NOTE

Anything quickly said here about programming is likely more carefully said in volume 1, or else explained in some of the documentation and/or example programs that go along with the G15 PMN programming language.

The structuring of the chapters are different in volume 1 because, up to ch. 3, content is interwoven and less sequential. Also, volume 1 was written also so that it can provide an easy look-up of central G15 PMN features.

And let me repeat also here: if you are reading this to actually learn more programming, you absolutely must have some background in G15 PMN essentials first--eg through Volume 1 in this 5-volume series. But you can also approach this in an eclectic sense where you pick out passages that make sense to you for instance if you are interested in the philosophy of sex and various myths we can build over how the universe came to be how it is.

ROOM FOR OWN NOTES /and your own index/

CHAPTER 1

1.1

Sex, in its highest form, is getting turned on and thrown into joy through some form of beauty. Beauty is an infinite concept. But part of it is that the proportions--such as the proportions of the lips--create a pleasant tingling inside you, for you feel that they are harmonious.

To explore harmony, we are creating insights into beauty --and that in turn can make our sexuality deeper and more intense.

When we explore harmony in music, it is often found that when wavelengths add up neatly, they sound more pleasant, more harmonious. The cycles can be counted: when you make a pure sound somehow, and then go twice as high in pitch, the higher pitch has two wave cycles for each wave cycle of the lower pitch. These two sound almost like the same tone.

A more complicated harmony, but very pleasant, is when the relationship of the waves are more like 2 to 3 rather than merely a doubling of them. We can also have 3 to 5, and 5 to 8. And it is possible to make drawings with these lengths. Much happens in our minds when we meet such ratios, or proportions. For instance, as soon as a girl with slender, fairly long legs in a miniskirt takes on high heels, suddenly the length of the legs compared to the upper body is much more a question of 3 to 2 rather than 1 to 1. It depends where you focus--hips, navel, or between the legs; and on how high heels there are; and how long-legged the girl is compared to her torso; and other factos like how wide her hips are also affect the seeing of beauty.

These numbers can be produced with no other means than addition. These numbers of harmony are also called the

Fiobnacci numbers /more or less after the name of a early thinker on the subject/.

The numbers do not say even nearly all there are to say about beauty. But to keep thinking about them, once in a while, can make all you do more fun and more beautiful and it can enhance your sense of sexuality and glamour to be aware of them. Let's make them.

1.2

Start up G15 PMN, its Third Foundation /which is the normal way of starting up G15 PMN when we do programming, and sometimes we only rarely point out/. Type in this, where we use the symbol ⟨LN⟩ to mean the lineshift button, the big 'ENTER' button on the right side of the keyboard.

1 ⟨LN⟩ 1 ⟨LN⟩ ad ⟨LN⟩ nn ⟨LN⟩

When you type in this, the PC tells you '2'. It is simple: you give it 1, then 1, then tell it to 'ad' /ie, add/ up the two numbers, then show the result. Be patient in going through with this, and we'll make the machine to it more elegantly on its own after we've experimented a little.

1 ⟨LN⟩ 2 ⟨LN⟩ ad ⟨LN⟩ nn ⟨LN⟩

Here, the PC says '3'. We added 1 to the result we just had, namely two. Our last two numbers are 2 and 3. So:

2 ⟨LN⟩ 3 ⟨LN⟩ ad ⟨LN⟩ nn ⟨LN⟩

We now have got 5. That number came up in the former paragraph. The most recent two numbers are 3 and 5. So:

3 ⟨LN⟩ 5 ⟨LN⟩ ad ⟨LN⟩ nn ⟨LN⟩

We get to 8 /also a number in the last paragraph/.

We can go on and the numbers will raise rather quickly, and go above the typical 2 billion level of the PC. The PC that properly runs G15 PMN is called "32-bit", and this is the ideal bit-size for a good programming machine, what we also called a "human thought friendly computer" and a PC that allows what we call "first-hand programming". /If we have fewer than 32 bits, on a digital computer that means too many restrictions; but if we go beyond 32 bits, the

numbers and the data sizes are so huge that it leads to a style of programming which discourages human thought./

There are many ways in which we can get the computer to do the above type of thing--'add the two most recent numbers and show the result'--and let us figure out one way. That's when we go from mere calculation into programming.

1.3

As volume 1 also will tell you, the letters W, D and F do simple things on the stack of numbers inside the memory of the PC. So we if type in eg 3 /// 5 then type W it will switch around these two numbers /W=sWitch/. Or type D and it will sort of bridge the 3 over the 5 so it is as if you have typed in 3 /// 5 /// 3 /D=briDge/. Or type F and it will make or 'forge' a new copy of the topmost item on stack, eg if you type in 3 /// 5 /// F it will be as if you typed in 3 /// 5 /// 5 /F=Forge/. Let us put these to good use to add up the last two numbers. I experimented before I started this paragraph, and found this to work--perhaps you want to check for yourself /either by drawing it up somehow, or by typing it on the computer/:
/we use small letters to program the PC, the big letters sometimes in a text like above is just to make them stand out/

```
1 /// 2 /// w /// d /// ad /// f /// nn
```

This switches the two around, copies the second on top of stack, ads them, and shows them via 'NN'. The reason for a 'F' before the 'NN' is to keep the result on stack: 'NN' chews up one item on stack--a number--and shows that number to us at the same time. To counter that chewing-up, the F does the job of protecting a copy of the number.

Normally, when you start up the 3rd foundation, if you type F /// NN /// it will say '123456'. That typically means there is nothing on the stack. Now, if you type F /// NN /// you will see that it shows the last result once more. And because of all our maneuvers, there is one

more number underneath that last result. What we have on stack is not '1' then '2' as we started with, but '2' then '3'. If you go up a paragraph, you see our desired list of numbers begin with just this: 1 2 3 5. Let us see if we can simply type in much the same as above once more and get the next number out, the '5':

```
w /// d /// ad /// f /// nn
```

Voila! We get 5. And the next is supposed to be 8, and that is in fact what we get. Hm. What we if we type in something like this:

```
abitfibo= ///
w /// d /// ad /// f /// nn.
```

This little thingy--notice the '.' /dot/ after the nn, and notice also the '=' after the new word 'abitfibo'--which we call 'a bit fibo' /without the blanks/ is that little efficient series of letters we worked on just above. How about repeating it a dozen times? Something like this:

```
morefibo= /// 1 /// 2 ///
ll:12 /// abitfibo /// lo.
```

The two letters LL and a colon, LL:, is like 'let's have a loop /ie, repetition/. They go pairwise together with LO, playfully what we can think of as, 'the LOop ends here'. We give 1 and 2 to the stack as start numbers for our 'abitfibo'. And the new word 'morefibo' should give it a run a dozen times. As writer of this book, I hasten to try if I got this idea right: the PC will tell.

The PC confirmed it. It gave numbers up to 610, including, on its path, 233 and 377. A little experimention, aided by memory of previous experiments, showed that this particular routine can do up to 43 loops, rather than just 12, without exceeding the limit of about 2 billion. At 44 loops, the numbers start getting funny--signed, for one thing.

1.4

Sex is not just orgasm: it may be much more orgasm before

the orgasm, so to speak. Multiple orgasms require each great energy, great stamina, great bodily hormones, to be really pleasant, so that they are not merely physical, physiological reactions of the body, but something you are participating in fully, in the mind. And sometimes, the sense of multiple orgasms before the first major orgasm can be ecstatic at the mind-level to an extreme degree exactly because the body is not going through its energy-demanding cycles. In EEG terms, there is a build-up of the hypnotic, joyous, deep-learning, intelligent alpha waves, and similar such, just before orgasm.

This sort of EEG brain wave tends to go together with a reduction of pain and anxiety and such. It literally works to reduce pain to be in the pre-orgasmic state. Whatever the body may have of issue--headache or this or that--as long as its sexuality is intact, and one can get to the pre-orgasmic state, pain is instantly reduced. But once one goes through orgasm, there is what is called 'alpha depletion', meaning that the pain in fact may be larger than before the sexuality, eg for a minute or two.

The golden ratio is one example of the type of pattern that speaks of increased wholeness for the mind, and which is spontaneously recognised, and felt, by any child, long before there is any education in the golden ratio. Your mind is every day completely new, when you have had enough sleep and vitamins and so on: and this newness means that just how and what it is that takes the mind to new orgasmic heights--which can involve some forms of golden ratios, in some ways--requires exploration. In this exploration, it helps having a vivid mind, a creative mind --and a mind educated to look for beauty, also through porn-like photos and drawings and paintings.

So I am saying that an education in art and design are some of the things, alongside programming, that create deeper and more fulfilling sexuality.

Another thing that is of essential importance in sexuality, for orgasms to come strongly and holistically,

is an attitude of 'being poly without jel'--put simply--whichi is to say, 'being polyamorous with as little jealousy as possible'. I am speaking of an attitude that cannot be mechanically implemented in your life, but which requires a constant exploration, learning, sensitivity--also to other people's feelings, limits, orderliness, messiness, potentials, levels--so that you get better and better at it.

An element here is to not take own jealousy as possible but steer out of expressing it, and rather take time argue in favour of the 'sexual democracy' of others, being generous in your heart to other people's potential joys with each other.

And this also means being sensitive to how to avoid creating either jealousy or frustration in others, sometimes by avoiding to say things and sometimes by being very forthright in saying things in a general and pleasant way very early on /for instance, that you are dedicated to being polyamorous and do not want to be 'locked into' the approach of trying to pretend you can only love one other person/.

Once you have worked with such themes in your own social life, you are better at creating orgasms in others and better at creating your own orgasms: because you are not trying to steer the mentality of others during, or before, or after sex, into any too narrow channel. You accept closed eyes--assuming it's natural, healthy, holistic and right for your partners to indulge in creative mental porn while having sex with you; you accept it for yourself also; you accept clothes on; you accept heavy petting as an interesting alternative to sex, including but not limited to those who are not yet fully up to full sex; you accept the need for others to 'glorify' themselves a bit and to 'glorify' also you a bit, as part of the sexuality between you and them--and so also accept the need for interesting music, surroundings, dance, talk, time, wine, fetish actions, smiles, quietness, plans, and money. For

those who have too little money to be certain they can make it through to next month, sex is often uninteresting; those who have more than enough to make it for many months can be generous and in that way 'remove the money issue' from their potential sex companions--which has nothing to do with reducing sex to money.

All this flows from having a clear mind, clear insight into all life, and such clarity is helped by programming and by drawing and by exploring own orgasmic potentials through tantric exercises, muscle-work of martial arts or what this author calls 'stamash' type--and many more such activities.

The golden ratio fits with such a polyamorous, highly sexed life where you approach beauty as if it is something fairly religious, and find that you can be generous to beauty because you see it as an objective force that ties people together in bonds that are constantly swirling around in a spiritual way, and not pre-determined in some kind of 'dynasty'-fashion. A girl you thought plain can show sudden superior beauty in an authentic, unedited photo from a certain angle, and by being alive to sudden and total recognition of this beauty, you can give her the energy she deserves to give her an opportunity to express it further. A motto here is 'beauty, rather than ego'.

Programming, including golden ratio thinking, teaches you to such rapid, we may say 'fluid' perception of beauty --and this ties in with the idea of the ambro-sexual, the fluid type of sexuality that, when more pervasive and present, can be called pan-sexuality.

How do you recognise consciously the golden ratio in such as a photo? How do we sum up the fibo numbers above in a sort of easy-to-remember formula? By this formula, you can get a deeper, faster 'high' on seeing beauty: the formula is summed up in this phrase: 618 permille. It is easy to explain why.

1.5

Take the highest numbers we produced earlier in this chapter, by adding the 'two most recent'. We went through 3, 5 and 8 and mentioned 233 and 377. It so happens that all these numbers have roughly the same ratio: more or less '618 permille'. Take anything and then measure half of it--that's '500 permille', right? Permille meaning, of course, 'per thousand'. 50 percent is another phrase, where we speak of 'per hundred', but it may be more suitable for good thinkers to engage in a thinking around permille where the 20th century habit was to do percent /a topic discussed elsewhere in writings by this author/.

Go a little up from 500 towards 600 permille. That's about what we get if we look at 5 to 8. Try it:

To check the relationship of two numbers, multiply the first by a thousand, then do a division by 'di', or, better still, by 'rd'--round division--and show the result by 'nn':

5000 /// 8 /// rd /// nn

This is not quite 618. But it isn't far from it. To get 618, just use 233 and 377 instead, or any of the higher fibo numbers we made--any two beside one another will do:

233000 /// 377 /// rd /// nn

Here we got it! 618. So we are lead into an astonishing result, one of the wonders of numbers, one of perhaps infinitely many numbers about numbers--but also a central organising feature: Addition is forever entangled into the number 618 when we are using the idea of permille.

Using other words, predefined in the 3rd foundation of G15 PMN, we can extract 618 permille of anything, such as from the number 315, by a phrase like:

315 /// 618 /// pm /// nn

Also predefined is 'golden', so you can just write

315 /// golden /// nn

By the way, if you start with the smallest number and wish to get up to the next one, you can use the idea of 1618 permille instead.

Since 5 and 8 have very nearly the same relationship as

233 and 377, and since, when we experience each other in sex and in daily living, we are always in movement, it is often enough to be sure that as many features as possible are having a 5 to 8 relationship, or slightly less than a 2 to 3 ratio: because while moving and dancing around, we are bound to connect to the exact ratios in glimpses. And just such glimpses trigger our sexual potentials.

Find examples of this yourself: the golden ratio is everywhere.

CHAPTER 2

2.1

In the previous chapter you got a blend of some rather deep and advanced 'snapshots of concepts' of sex combined with art and design ideas combined with bits of G15 PMN programming. This blend requires a wakeful mind, and a happy enough attitude about sex to be willing to read about it while also thinking technically; it requires a fairly good grasp of English; and a sense that numbers are okay things--or beings--or whatever they are!--and worthy of being thought about both analytically and sensually.

Those who came to chapter 1 with a haste to get into a particular topic, or very much centered on a particular ideology of sex or of programming, or who just browsed through it without foreknowledge of the style of thinking we advocate in the Art of Thinking series as its most eminent form, may well have got little out of that chapter --except that either it, or themselves, appear a bit "confused". To those, I strongly admonish: give it more chances: read it when you really have quality time, a great hot cup beside you of coffee or whatever you favourite mentality stimulating brew may happen to be, and when you have the 'smile in the body' that comes from it having a memory inside it of some good sexuality, whether it comes from 'partnering with yourself' or partnering with others. When both you and the chapter come across as coherent, it is time to proceed with this book!

Since, on a computer, every pixel on the screen is drawn through numbers--a number for the horisontal X coordinate, a number for the vertical Y coordinate, and a number for the tone--for G15 PMN that means a greentone--and since every letter is represented by a number--called the ASCII

numbers, where eg 65='A'--and where every programming command, such as 'AD' or 'RD' also are represented by numbers, also called, in this case, 'warps', inside the PC, then the big key to love programming is to love these numbers--whole numbers--not meaninglessly high--possibly signed numbers, from about minus two billion to about two billion.

By these numbers we can, if we absolutely have to, make bigger numbers--by putting the numbers beside one another and making suitable functions around them. This is done in the robotic set of functions, which expand on the 3rd foundation. These are socalled 32-bit numbers, called "32 bit" because it takes just 32 "bits" of 0s and 1s to cook up the whole two-billion range /something you can check by multiplying two possible bits with itself 31 times, ie, 2x2x2x2..x2, and leave one bit over for sign/.

When we think in terms of what this writer calls "the super-model theory", we can argue that the world of beings and things as we experience the world through our senses, is in many ways a result of the intermingling of 32-bit numbers--at deeper and deeper levels. At some level, we must leave the domain of science and theory and at that level our spirituality comes in. Between the living, spiritual ground of the universe and the manifest universe there is the domain of numbers in their sometimes more, and sometimes less, algorithmic or machine-like movement. An 'algorithm' is a function, just like we made a loop to churn out fibo numbers in chapter 1. That is a machine-like thing. Those who are socalled 'atheists' or God-deniers, would say that there is nothing that is not machine-like. The more advanced, and, in this writer's opinion, correct view is that the machine-like works in collaboration with a constant potential for something non-algorithmic yet intelligent to work on it. The super-model theory thus have pathways to the spiritual in a way not seen in conventional 20th century dominant physics-inspired worldviews.

A worldview is always throbbing inside the human mind--whether you're aware of it or not. And the worldview sets the context for judging what seems logical and coherent, and even interesting. Those who claim that they are not really interested in worldviews are more precisely not interested in changing their worldviews. Those who claim that science is about studying phenomena without prejudice from worldview merely try to pretend that they themselves are not prejudiced: whereas in fact those who deny the existence of worldview are usually those who are the most heavily prejudiced. And science, at best, lies in its constant aim to be unbiased and unprejudiced and evaluate facts on a premise of pure perception and genuine intuition, not just through the senses but through the mind, and through feeling /not presuming a sharp division here/.

To reach such pure perception the worldview must be of a kind that is more rather than less correct, in the sense of it being a trustworthy enough /even if rough/ 'map' of the universe, of existence. However the worldview is not just one box or one sentence or one thing: it is a combination of many features, including many assumptions. If even any single one of these assumptions is totally wrong, then it will have an effect of creating contradictions and some incoherence in the overall functioning of the mind. This we can call 'ego-noise' in the human mind. And the art of thinking involves, first of all, to recognise that the mind of the mortal human mind is never perfect, and, secondly, that the worldview must itself be made available for dialogue, attention, doubt and exploration, and undergo changes so that it can increase in coherence. In other words, the intelligent human being, at live to the passion of conquering own ego again and again, will raise to the challenge of asking questions that goes against the foundations of own world-view, and aiming at getting pure perceptions and good intuitions about each and every feature of the world-view.

Gradually, over time--for one cannot do it all at once--more and more intelligence can come into the worldview: but one mustn't do this with a perfectionist attitude, for it will never be perfect in the sense of 'finished'.

Also, one cannot do it the whole day long, for a living human being must also do many other things than philosophy every day, and if these other things are ignored, the mind will get sluggish and not be able to do even philosophy. So by necessity, the mind of a mortal human being is in a state of evolution. It is a proposition by this writer that the human mind in its core is soul, in the sense of not being merely a structure of the brain and body, and this is possible to articulate with great clarity when one looks deeply into super-model theory. There can be much more structure to something like a soul and a spirit level than to the brain and the body, even though the brain has a structure far greater than even that of the DNA molecule which exists in every cell of the human being. The brain has, as we know, billions and yet more billions of cells of various kinds and these are interconnected in myriad ways, and all this is living, changing, growing, pulsating --and the coherent brain is able, this writer proposes, to be a vehicle for something far, far beyond it and much more intelligent than it and the true experiencer of things. This is the soul-level and the spirit-level, to use such classical spiritual words for it, though there may be no classical spirituality that adequately talks about these more subtle energy phenomena in a very accurate way.

The fact that little has been measured of such subtle energies in conventional 20th century should not lead us to assume that these levels, if they do exist, are in any way lacking structure. It is typical for scientists to imagine that what they haven't yet measured on is pretty simple when all comes to all; and just as typical to see that half a century later they happily declare that the foregone generation of scientists were all wrong at that

point. This, for instance, has taken place when it comes to the mapping of the human brain, from the early surmises around the brain to the realization that the brain is a veritable galaxy of complexity.

In every worldview that I can think of, numbers are given a role. And so, when we examine our most essential viewpoints about numbers, and open up these viewpoints for change, we are very likely opening up a portion of our worldviews for change.

Let us muse over this question, which relates to all our foregoing discussions in this volume so far:

* what is the relationship between sex and numbers?

Put in other words, when we look at whole numbers--and we're speaking of the 32-bit kind--can we, in some meaningful philosophical way, get a sense of sexuality in or through them?

Let me add a viewpoint about golden ratios before we proceed with this question. In the previous chapter, we said, didn't we, that in the context of permille, 618 is a lot of what the golden ratio is all about. This is a valid statement, I feel: but some, especially those who are influenced by 20th century thinking, where there was a tendency to try to speak of 'infinite precision' where in fact no such precision was available, may question this statement. They may say that the golden ratio is not just 0.618 but rather it is something like "0.618..." in which any number of additional digits should be added "for precision". This is a way of talking that comes fast if one has read 20th century literature on the distinction between natural numbers and 'real numbers' in the wake of Georg Cantor's attitudes on the subject, which suggested that the 'set of natural numbers, though infinite, is somehow smaller than the set of real or decimal numbers'.

However that way of talking is just a way of talking. What is correct to say, in the wake of the deductions of essence numbers, after the reasoning around infinity, as written about by this writer a number of places /also

inside the super-model theory text which is at the core of the 3rd foundation/, is that a number, a ratio, has a precision that must be given together with the context of meaning. The context of meaning of the golden ratio, when we are speaking of permilles, is such that it is very precise to say 618. If we speak of percent, it is 62. There are other contexts available, and each has its own whole number that can express a ratio most completely. That something is 'complete' is in a sense greater than that something is 'perfect' /a point made very clear by C.G. Jung/. For 'complete' involves a full unfoldment--the root 'ple' is the same as in 'imply'. But 'perfect' is a hope that something done is entirely finished in an ideal sense, 'per fait' meaning that it is done--ie, sort of closed. That which is complete is thus more a process.

So a number like 618 is complete, in a permille context, to speak of the golden ratio. The very idea of 'any number of additional decimals' has an incoherence about it --for 'any number' involves trying to treat infinity as if it is a machine-like thing. So it really implies an atheistic worldview--though most writers, when influenced by 20th century thinking, would not quickly realize this.

In short, then, whole numbers are psychologically meaningful units when we have a good range of them, and 32 bit turns out to be the most meaningful of all, for the full breath and width of programming. In order to create more complex processes than that which can be created by 32-bit programming, it is a postulate of this writer that it is best not to expand the programming language as much as to expand the quantity of computers and create a series of control programs that each have 32-bit as their foundation. In sum, they can organize processes of a complexity that transcends 32-bit, even if each 'unit of meaning' is still associated closely with a 32-bit number. /For those on a spiritual quest, one can imagine just this underlying the 'super-models' in this vast manifest universe of ours, and so we can visualize a vast number of

suble computers existing 'underneath' the material energy operated on by God and his myriad subtle, powerful, beautiful beings or angels--the muses./

So, then, what is the relationship between numbers and sexuality? We see that the question is intensely philosophical. It is hugely general, it is of a type that can invite meditation and metaphysical inspiration and intuition.

As a working thought, while we explore it further in infinitely many ways, I propose this:

A number is a particular way to do sex.

2.2

Up until this point, perhaps an underlaying assumption has been that the reader is so naturally high on sex that the challenge is to sort of channel this energy into something as apparently different as programming.

What if a person is not high on sex? Not even when trying? To be highly sexed is not a fixed state of mind and body, but something that can come and then leave: and however beneficial it is for certain types of works and exercise and healing, it isn't easy to command forth when it is not there. Now those who rarely experience not being high on sex might blame those who are less highly sexed, believing that they are resisting sex out of a sort of consciously held prejudice or due to an illusion that sex isn't a good thing or something.

While some may indeed harbor foolish forms of illusions, suggesting that sex and porn of some forms are 'bad', let us rather explore the theme of sexedness with an open mind. For it is a reality that a person who is normally highly sexed, and who has every sort of meaningful good belief about sex, can suddenly be depleted of sexual energy for several hours, without obvious reason. However, there are usually reasons, and by understanding these, we can heal the sexual energy. Through healing it, we can get a general re-motivation for many fruitful actions, which

can include programming. So let us explore it.

If we simplify enormously, two things are required for the sexual energy to be high in a human being: that there is good bloodflow to the erogenous zones, especially to the genitals between the legs, and that the brain is pulsating with pleasant alpha waves in a harmonious way that involves sexual visualisation. It makes no sense to take any drug or medicine to stimulate the bloodflow between the legs if the brain is not sexed up. When the brain is sexed up, the subtler parts of the mind are also called on.

For a healthy person under normal circumstances, the bloodflow is good, and especially given naturally aphrodisiac supplements in daily diet beyond normal vitamins and minerals, such as ginseng and maca.

What may create a non-alpha state in brain is a succession of experiences of apparently trivial frustrations, so that these accumulate, giving a sense of lack of connection between intention behind actions and results of actions. For instance, given several hours with hasty practical actions, if a number of these--even if each action is of relatively small importance--lead to a sense of 'resistance' in getting things done, the brain wishes to sort out all this and get back into order. If this wish is ignored, it is unlikely that the harmonious, meditative, alpha-wave intensity will be triggered even when the sexual stimuli are plentifold.

It is this state that may be called 'depletion' or 'exhaustion', and it may be overridden only in a superficial way by intake of chemical stimulants like caffeine. Caffeine works well together with a harmonious, meditative state, but if the harmony isn't there it gives a mental wakefulness that doesn't reach deep. It may be enough for more practical actions to be carried out, but it is probably not enough for sexual energy to arise.

What can be done in such an exhausted state? If the person is normally harmonious, normally sexually energetic

it probably takes half an hour or an hour of some form of meditation, in which each little frustration is looked on from several angles and sorted out and, as far as frustration goes, dissolved. As part of this, to imagine or engage in some situation which provides good feedback, a sense of personal worth and attractiveness, will be of great value to restore the energy.

The more deep and/or lasting the energy depletion is, the more time and varied action may be required to restore harmony. For a person who has experienced something that has created a very sad or even sorrowful state, it may be necessary to spend time on something entirely different for a good while in order for the mind to build up a fresh self-confidence from an activity in a different domain altogether. This other domain should be more controllable, more predictable, more certain in how it gives feedback. For one who is good at programming, programming can work as therapy quite remarkably, in such states.

A complicated set of frustrations and perhaps sorrows deep into the mind may need a rather complicated set of healing actions, combined with much walking and much sleep and so on, in order to be fully addressed--before the sexual energy can be naturally elevated to a high level again.

Whatever it takes, it is important for a refreshing of the highly sexed state that the brain is part of it: it is a brain coherence, a brain wholeness, that is necessary, not merely a physiologically good blood flow in the tissues with all the proper hormones.

2.3

The paragraph numbered 2.2--the foregoing--of chapter 2 in this volume is setting forth an understanding of what it is to be depleted of energies, also sexual energies, and also an understanding of what it takes to replenish them, which includes meditation and time, and sometimes many additional approaches. It is the premise of this writer's

approach to human thinking that mindfulness is essentially tied up to sexual feelings, and these feelings of sex are again tied up to beauty, and thereby harmony; and that thinking must be as it were a wave on an ocean of such feeling to be coherent and harmonious, and not just logical in a narrow sense. Very few of the old, withered socalled 'philosophers' and 'prophets' who are authors of the most famous and influential texts on what it takes for a human being to be rational and whole, in the past millenia, have allotted this much importance to sexuality. Most of them have talked about sexuality in a way that reflects their decayed state--and not reflecting how they probably themselves were taking their best decisions in the youngest phases of their adulthood.

As a result, the approaches to thinking in the 20th century, both from the viewpoint of socalled "religion" and from the viewpoint of socalled "science", rarely lived up to a wholeness of understanding of the art of thinking as fundamentally a sexual type of activity. Instead, what set the agenda in both these leagues, as well as in politics etc etc, was redhot condemnation of sex and a hailing of incoherent standards of thinking. And when politics start condemning something, laws are created to enforce patterns of behaviour in society, and these laws makes the prejudice somehow seem more 'objective'. We saw this with such as the laws prohibiting gay behaviour that led to such as police persecution of Alan Turing, a British war hero and inventor of the first practical vacuum tube based computer, as well as a fascinating thinker. The photos they discovered at his home led to a series of actions by the police that only in the beginning of the 21st century was officially condemned by the British government: they asked for forgiveness to the legacy of Mr Turing. The worldwide lifting of bans on gayism in most Western Europe and USA influenced countries in the late 20th century and early 21st century is however just one element corrected, and

there are a hundred more elements. Society is in a mess, and thinking across all societies is dysfunctional and deeply ingrained with prejudiced against sexuality.

The few thinkers who tried to undo some of the prejudice against sex, and who tried to figure out how sexuality indeed can be a more central part of the entire worldview, had to struggle hard to get every bit of the argument heard--and they were supported but fractionally, compared to collegues who, loyal to the false system, merely promulgated the falseness of society.

In upcoming volumes in this five-volume series we will have a chance to look more into research using statistics and, in some cases, scientific measuring instruments like EEG, to explore also the abundantly important role of sexuality for the human mind. We will look into this through some novel work by the undersigned and others, and as resume of some classical works esp on female sexuality.

Let us now pay attention to the golden ratio again, with an intent to use some words that perhaps have not been used before in this regard, and some ways of thinking about it that allows us more quickly to recognise the golden ratio in daily life perceptions.

We will load in app# 7,777,777, called AngelPen, and sketch some golden ratios on the screen by step-by-step thinking about how to use this app. This app contains a handful of useful functions, slightly expanding on the Third Foundation G15 PMN so that we can do some curves and lines on the screen in a way that is compact, fast, and easy to think about. This comes later in this chapter.

2.4

Consider these lines:

```
-------------/---------------------
=== =====
########:############
-------------------- ------------------------------------
```

What they have in common is that the second part of these lines of characters is larger than the first part of these lines, by about 618 permille. The second line, for instance, === =====, has three then five, two of the first fibo numbers, as we call them quickly. The other number of characters are all found by pairwise looking at the fibo numbers up to 55: 3, 5, 8, 13, 21, 34, 55. The longest line, of -, has 21 - then a blank then 34 -.

Compare === ===== to === ======. The first has a 3:5 relationship, the second is a mere doubling. Doubling is more mechanical: more the thing one expects of a machine. The === ===== is more organic in its feel. It is not 2x, not merely 'twice'. It is === plus a little more than half of it. Let us call this 'essence addition'. Adding a little more than half of it, nearly 62 percent, or more precisely 618 permille, is one way to remember the golden ratio.

It works the other way around also: it is a form of natural substraction. Let us muse about the first line:

```
-------------/---------------------
             -------------
```

The line is composed of 13 _ and 21 _. Now we have placed 13 _ underneath the 21 _. Let us study what happens on the right side more clearly:

```
-------------/---------------------
             -------------........
```

We have put in dots on the right side. That turns out to be eight dots. You see that we have 13 _ and 8 dots. That is again two fibo numbers! Let us put these dots under the _ in the second line:

```
-------------/---------------------
             -------------........
             ........
```

See what happens? Again, we are tracing the golden ratio:

```
-------------/---------------------
             -------------........
             ........;;;;;
```

We put in some semicolons after the dots, so that it matches the quantity of _ in the line above. Here we have, in fact, 5 semicolons.

Why does this matter? There is a playful kind of similarity of form, shaped by easy additions and easy substractions, when we start out with the golden ratio. In other words, when we have used the 'essence addition' once --it keeps coming up in a lively way.

Let us bear in mind that our brains are wired to look for patterns within patterns and so much goes on in our brains whenever we are sensing anything. The toddler may not have heard about golden ratio or essence addition and yet that is not preventing the living, learning, child brain from adding and substracting lines, comparing forms, comparing ratios, and extracting learning. And much will naturally happen when we place these proportions beside one another.

Let us tune into it sexually: you are walking by the poolside, and the / is where the bikini is, to the left is the shining smooth sun-kissed torso of the girl, to the right her legs, playfully stretched, her feet elegantly arched--long long legs compared to the torso, accentuated by the little bikini. As she sees you, she wriggles her toes and stretches a little and many more ratios are dancing before you--and it is all outlined in these ratios if you blur your view of this line and engage in visual beauty imagination:

----------/----------

The second line is like the first line but it has a little more than half of the first line as addition to itself--618 permille more, perhaps. And you watch her fingers, her slender hands; you watch the similarities of form of how her hair is dancing over her head relative to the fascinating spirals of her ear: and in any beautiful image, there are myriad similar forms, and forms that contrast in ways that can be summed up through various resonances including what we call essence addition.

The spiral of her ear may compare to how her hair is made up, it may compare to how the line of her elegant jaw and her cute profile is shaped: and for some shapes, there are other concepts that may come more easy to use than the golden ratio. We have just mentioned 'similarity of form', and a word that covers a kind of research into this sort of thing is, of course, the word 'fractal'. It is possible to argue that all of them are intimately woven together with essence addition: that is an advanced topic, and it is not necessary to always insist to seeing the golden ratio everywhere. But since essence addition is tied in to our very concept of what whole numbers are all about, and whole numbers can be used when discussing any type of similarities at all, it isn't surprising that we can keep on coming back to it.

Let us compare something like =====/======== to something like =====/===== or =====/========== or to =====/=============== /5/8, 5/5, 5/10, 5/15/. 5/5 is symmetrical--nice, but it 'doesn't go anywhere'. 5/10 is perhaps going somewhere, but fast: too fast? Like the sudden jump of a machine into action? 5/15 is almost like two different things: 5 and 15, ===== and ===============.

This suggests--and let us constantly bear in mind that what applies for vision applies equally across all areas where there can be any form of structure, all sensory modulaties, also spread across the dimension of movement, such as dance and music--this suggests, does it not, that with the essence addition we have organic movement--and with many other forms of ratios we have something that is perhaps nice, but not so obviously organic and not so obviously movement.

In a word, essence addition is natural movement.

And if you take this to heart, and masturbate over countless fantastic porn photos while intending to teach yourself orgasmic beauty concepts, you will encourage in yourself a capacity to create great beauty whatever you touch, whereever you go.

Practically, when you plan ahead, being in such a state of sensing orgasmic beauty patterns in this way, you will sense what is whole and what is lacking in that whole; you will sense what is whole and what is opposing that whole, and which should have its kinks ironed out. You will sense what you need to get to know more about, get up to speed about; and where your competence is adequate. You will sense also where words have a relationship to facts and where the words are used more as tools of manipulation; and be able to steer your own use of words so that fairly coherent results after all can arise.

We can call this type of work for 'pre-meditation'. You are engaging in pre-meditation when you sense how you should plan things, and get an overview over what you take to be the patterns, and how these correspond with your own natural behaviour patterns. In a state of pre-meditaion, you can tell yourself how you should counter your own instinctual behaviour patterns in a situation to fit with a larger ratio, a larger understanding, a larger premise for the situation. When you find yourself in that situation later on, you will find that your intuition can play on your pre-meditation and remind you of what you told yourself, and so you are in a kind of dialogue with yourself even in the midst of also speedy and fluid and humorous action.

Pre-meditation, in short, is one of the things that allows you to more truly engage intuition in daily life: an intuition that isn't merely a fancy word for following your instincts, but an intuition that flows from perceptions that you have nurtured earlier on. And these perceptions flow naturally when you have a sense of the sexual beauty of the essence addition, also as a kind of value or ideal for the greatness of any action.

2.5

It is time to play around with Angelpen a little bit, so we connect our capacity to think about essence addition--also called golden ratio--to programming. We will, in the spirit of doing things easy, and not get entangled into a myriad technical discussions, simply load in Angelpen app #7,777,009 and start it and look at its code. The result of running that app--which is having all of 3rd Foundation and then all of the #7,777,777 Angelpen app in it, and then just a couple more cards--is shown on the cover of this volume 2 in this Art of Thinking series. It is a spiral and a bunch of golden ratio rectangles. We now know that we can call them 'essence addition rectangles'. The spiral is just to tell the observer to thinking in terms of spirals while viewing the rectangles. We will turn our attention to how one can make such rectangles in Angelpen.

When we do this, you will also see places where it is easy to begin to experiment with changes to the code. This is a time-consuming thing, but once you begin it, you will find that your understanding of programming leaps /you don't have to do it to read this book, though; I'm simply mentioning that it is an option/.

Here is the result of loading in the app, then using the utility menu in G15 PMN to convert "CAR =⟩ B9EDIT" /ie, the 'CAR edit' form, the program cards, into this text editor, B9edit/, then using app #9328123 which adds the ⟨k2⟩, ⟨k3⟩ to show which card on top and also adds some spaces. Here, we don't include card K1, which only has comments, and card K12, which only has a startup text which says that one can type 'get' to get the graphics. The graphics is, as said, at front of this book--though of course it varies a little bit in pixel intensity, something handled by the first function, 'freeink'.

In case you aren't in the mood of liking to see code like this, don't worry, just glance at this bit and that bit of it; read comments; and read on. Unlike a typical

English text, program code can be studied bit-by-bit, quite meaningfully.

In the upcoming pages, we discuss card k2 to k11, which are all the program cards in the #7,777,009 app. You can read fast over this section and look at it more closely later; in this book, there will sometimes be more tech stuff, like here; and sometimes more free thinking, and you can read in any sequence you like.

Card (k2) coming up, here is what it does: it makes two new words, short for 'free ink' and 'free ink, and walk' --and these can be explained this way: we want to draw the rectangles with varying tone, or 'ink' as it is called here. The word 'ink' sets the greentone 0..255, and 0 is black, while anything about ca 60 is a visible green with 255 the brightest. So let's get it to vary between 60 and 255. The variation word is 'af', eg short for 'a free number'. We give it 195 as input, so it produces a number up to this--and new each time. Add 60 to this and we've got the input ready for ink. Then it says, simply, 'ink'.

In making a golden rectangle, with the shorter and the longer lines being fitted with the golden ratio, we are repeatedly going to use the word 'walk', which in Angelpen /ie, the normal set of extra functions you can load into the Third Foundation to make a certain type of graphics/ does drawing on the screen while it moves a certain number of pixels. Just how many pixels is a number that is left on the stack, usually before 'walk' is called. Here, it can be left on the stack before 'freeinkwalk' is called.

```
(k2)
freeink=        freeinkwalk=
|changes ink    freeink
|rather freely walk.
60
195
af
ad
ink.
```

Let's get on to (k3) to (k6). These make one full golden ratio rectangle, given width and height as input. These four cards are all about making the function 'goldcard'.

Where will it draw the rectangle on the screen? You see, that's one of the relaxing features of AngelPen: you tell it how much it should move in its present direction, and when it is 'walk' it leaves a drawing line, and when it's 'fly' it just flies over without drawing. It itself keeps track of its present position and direction.

To change position, call the function 'turn'. To turn a corner, telll it eg corner /// turn. Half a corner, also called '45 degrees', is halfcorner /// turn. You can give any meaningful number to 'turn'. To make it turn left instead of the normal clockwise turning to the right, you can write 'turnleft'. That, combined with the words to begin and finish use of AngelPen, which are, respectively, 'angelstart' and 'angeldone', is the most important set of tools to do an enormous variety of graphical sketches.

These can also be done by the sine, cosine and such functions directly, if you calculate a bit around them.

In card (k3) to (k6) as follows, the two sides of the rectangle are stored in two places, the ix and i9, two of the many places a function can easily store numbers it wants to have access to, without having it on the stack all the time. To store to ix, use 'sx'. To store to i9, use 's9'. You can think of 's' for 'set', and 'i' for 'inside /this place/', ie, 'get the value inside'.

```
(k3)
goldcard=       s9
|in:width,      |start&finish
|height         |ca in the pos
|gives:         |of top right
|nextw/or h/    |pointing from
|Angelpen       |upper right
|draws from     |along height
|upper right    sx
```

Here, lots of comments are given, with | line first. So it really only does goldcard= /// s9 /// sx on this card. The comment tells that new width or height is given as output from the function. The point is this: it finds out what's the golden ratio reduction of the longest side, and draws the long side and the short side, then it leaves a copy of that calculation on the stack so it doesn't have to be repeated by the calling function when that calling function needs that value in the next loop.

```
(k4)
ix              i9
100             freeinkwalk
mm              corner
golden          turn
100             ix
rd              freeinkwalk
                corner
s5              turn
```

Here, in (k4), things start to happen. Stuff is put to i5. What stuff? The golden ratio of the width, 618 permille. To give it a little extra precision it is multiplied by a 100 first, and round-divided on a 100 afterwards.

And action: i9 /// freeinkwalk /// corner /// turn /// ix /// freeinkwalk /// corner /// turn

In other words--draw a line, turn a corner, draw a line and turn a corner. The first line is ix or width, the second is i9 or height. This stuff is repeated in next:

```
(k5)
i9              i5
freeinkwalk     fly
corner          corner
turn            turn
ix              ix
freeinkwalk     freeinkwalk
corner          corner
turn            turn
```

Note that the i5, the golden ratio proportion, is brought in here as well. So we have not only width and height of the rectangle, but also a third line being drawn up to point out where the golden ratio is here. All the time the greentone varies. The PC is simply told where to 'put the pen' on the screen and there is no explanation given nor any explanation given to the PC. It is an action script.

```
(k6)
i5              corner
fly             turn
corner          |Note:with the
turn            |quantum-like
                |visible pixel
                |we get extra
ix              |variations
fly             i5.
```

As promised, the function in (k3)..(k6) finished by putting i5 on the stack, the result of the calculation.

In (k7) there is a function called 'get' which loops through calls to the function we just made, to draw up

rectangles inside one another; in addition, it loops so as to draw up a spiral beside it.

In ⟨k7⟩, the word 'jump' is used: it is often used right before the AngelPen routines are used for real, after a little bit experimentation, to find the right position for AngelPen to begin on. This is simply the x, y position on the screen: jump just sets the starting position. The slightly related word 'fly' moves along a path in the same way as 'walk' but without drawing. The word 'jump' just puts in the starting-position for all this.

Two times 'pi' is a full circle if you remember your trigonometry. On its own, 'pi' is a half circle, or 180 degrees. So pi /// turn means, simply, turn the pen the other way.

```
⟨k7⟩
get=            angelstart
|Show it! :⟩    105
                3
                jump


                pi
                turn
```

Where did the numbers in ⟨k7⟩ come from? I suppose I experimented when I made the program. You can experiment with other numbers when you have time and see how it looks.

```
⟨k8⟩
89              golden
s6              100
                rd
11:5

i6
```

```
100
mm              s5
```

The program appears to me now to be made in a bit relaxed manner: for we see that the word 'golden' is used inside the loop, while on the next card, our function 'goldcard' is called, which again calls the word 'golden'. I am sure that this could have been compressed into a single call to 'golden': but it doesn't matter when we are doing a loop with just a handful of calls, instead of a handful of millions of calls.

```
(k9)
i5              angelhome
i6
goldcard        corner
s6



lo              turnleft
```

Whatever it does here it is just more of the same type of stuff to make it go around and get gradually smaller rectangles, five of them. In (k10) it shifts position by 'jump' and does 30 loops to draw a spiral:

```
(k10)
                30
                7
                jump

                11:30

halfcorner      3
s4              walk
```

The (k11) has the clue to why it is a spiral rather than something more like a circle: the i4 /// turnleft makes it turn less and less. It spirals, supposedly, outwards: it takes 905 permille off its turn-factor each time, which is again a number that probably came after trying out this number then that to see what works out. Try out slightly larger or smaller numbers and you get various spirals!

```
(k11)
i4              lo
turnleft

i4
905
pm

s4              angeldone.
```

It finishes with 'angeldone'. Alright, such comments on a bit of code are perhaps not the greatest literary pearls in English writing: but that also shows something of why a programming language is an interesting thing. The language used to instruct a computer is radically different, it is contextless, action-oriented, number-oriented, unambigious and often entirely dependent on great precision, however this precision is usually rather simplistic--like being sure that the x and y on the screen are within 0 and 1023 and 0 and 767. The wonder is that by putting in such stuff to the computer that anything meaningful, let alone sensual or philosophical, can arise. Yet it is the case that when the computer does these instructions, it does it fast and with a mechanical sort of perfection that can manifest the idea of the creator behind the program, and often in a way that wasn't entirely anticipated but which after all can be seen to be the logical consequence of the

instructions. Programming is a unique way to train the brain, to make it tickle in otherwise less-used neurons.

2.6

In the 2.2 part in this chapter, we talked about what it is that sets the mood of being 'highly sexed' apart from a mood with less alpha-waves, a more frustrated mood, in which sexuality is less interesting. There is another take on it, through the keywords 'identity' and 'worship'.

When you are in a mood that is perhaps fairly energetic yet without a sexual high to it, and you ask: where is your identity? Where is your sense of self, your "I", where would you point with your fingers if you had to point? In some circumstances, you perhaps have a floating feeling of identifying with much other than your body: but in many cases, perhaps you would vaguely indicate the head or heart region, or a region in between them. That's where the sense of "I", or the personality, even your name, may seem to be feel to be "seated", in a vague way, in several states of the brain.

When you are sexualized, it is the climax areas between your leg, clit, dick, and near, that is mostly what you identify with: the brain is giving peak attention to these areas.

To feel appreciated in the nonsexual state may mean that the area near the head or heart or throat region is getting positive appreciative nods from worthy observers, which can include yourself.

To feel appreciated in the sexualized state is something different: a peak of appreciation involves that somebody who is experienced as worthy in a way does a worship of that centre of your sexualized being that is your erotic, climax-oriented nerve areas. This also suggests something worth being aware of: condemnation is a sexual turn-off. The action of worship by somebody who is 'worshippable' quickly leads to a sense of bliss which can go into orgasm.

We can describe this with less direct reference to the body and the brain and give it a description in terms of the subtle energies of soul and spirit. Beauty comes in here because you are experiencing another's presence as worthy exactly when that presence is beautiful, and it enhances the sense of yourself being beautiful that your experienced centre is also experienced as worshipped. All in all orgasm becomes a kind of overlapping of beauty without borders. And yet the most rewarding part of sex may be before orgasm as a physical event: the pre-orgasm may be more orgasm than orgasm itself, and can go on for hours. Multiple physical orgasms are of course possible, even for hours, but each physical orgasm involves a change of physical energy. So it is the pre-orgasmic state that in many way is spiritually the most orgasmic. And these kinds of states can be nurtured in many different ways, including through BDSM, which often involves stimulating areas near the key sexual nerve centres. Of course lips and breast tips, and earlobes and buttocks and more are also erotic centres in a way, and there are erotic sensitive regions inside the vagina on its upper side near the clitoris, but little can beat the intensity of the clit/dick area.

Orgasm is, chiefly, a certain emotive event of spiritual attention-gathering: it is a peak way to bring new forms of harmony to the mind, clarity and energy to the body. It is also a sense of abundant order, that life has meaning, that cosmos is a whole. The sense of utter clarity, of life being a kind of Concept throbbing with life, is akin to the sense of order that may come upon the person who experiences the freedom of programming a computer. The order that is brought about in the mind from programming may be faint compared to the thrill of the pre-orgasmic play that goes on between two nude or semi-nude or fully clothed human beings, or three, or four, or an even larger group, but it is of the same kind entirely. And it may be that for some, the extra peak of clarity that programming

can give to them, may be exactly what they need to unleash new powers of sexuality and sexual urge and capacity to worship and be worshipped: there is something near the sense of the absolute about both programming and such empowering sexuality.

A good program is also a beautiful program. And it can produce beautiful results, when it is thoughtfully constructed. The beauty-experience is a constant guide in programming, and the computer is in a very real sense 'perfectly forgiving': it is what you type in and give to the computer that matters, not who you are or what body language you had relative to the computer, nor how you treated it the last time. Of course there can be certain programs that have in them a recording of 'user behaviour' and such, and which have security functions so as to protect critical areas in a society: but the computer as such as a clean slate, and in a way allows you to be met as if for the first time, each time. As it is said in a branch of buddhism: the zen mind is the beginner's mind. The meditative mind, in other word, is the mind of the child, the innocent mind, which has the newness of clarity in it. This is something that the computer programmer, by spending enough hours with the computer punching through the syntax, always and inevitably gets through, at least if there is no allergy against numbers. And this is perfectly compatible with dazzling orgasms, a full-blown sexuality, and the art of thinking involves being alive to both fields, sometimes at the same time.

CHAPTER 3

3.1

In exploring beauty, let's be clear, it is an endless exploration. It has no absolute, final conclusion. Let us be religious and ask: Really!? If you imagine God in his essence having absolute wisdom, insight at all levels--absolute enlightenment--does he, too, find beauty to be an endless exploration? That's a perfectly valid twist on it.

In other words, has God any limitations in insight?

Perhaps not, but in having the playfulness and gaiety of mind to make a manifest universe like ours, God surely has a great sense of time, humour, stories, emotions, and also exploration. It is possible to imagine that while he at his essence knows all in an absolute essence, he has gone into his own unfolding storybook with a conscious intent to be faintly less absolute in insight, so that for him, too, beauty can be an endless exploration.

And in this process, he creates beautiful assistants, living beings, girls, so beautiful that even the most beautiful manifest human being is a mere shadow of his essence beings--his muses. That is at least a visualization that could make sense, at the core of this universe: something real and personal and lively, something to account for all the recurrent patterns in the manifest universe.

To help his work processes, obviously, God and his muses want computers, and a language to operate these computers consistently. The computers can handle a lot of things by their own buzzying in the background, so that not every atom requires the full attention of God and his muses. If the ancient Greek myths, a cornerstone of the European civilisation, are anything to go by, these being are both sexual and sensual. The Greek word for Zeus is pronounced, in modern Greece, a bit like theeos. This is indeed the same root in core that became the Latin, Christian word for "God", namely Deus. Yet in Christendom, Deus was visualized with less of the vitality of Zeus: where Zeus floated around in his creation looking for beautiful ladies and doing all sorts of transformations of himself, even into animals, to easen his prospect of having sex with a beautiful mortal girl, the Deus, as spoken about by the followers of Christ become more the judge and the emblem of inner peace. Zeus, too, had a judge-role: but he combined this in one being with his sensual, sexual self.

The lack of wholeness in the Christian conception of God thus made Christian a dualist religion, in which the temptation of sex was relegated to 'the bad one', a laughing, careless, horn-being whose intent was chiefly sex and to make others having a bad time. This 'satan', as visualized by the Christian thinkers, was thus adorned with a sexuality that the ancient Greek thinkes had allocated to God himself. And, it is fair to say, I think: a lot of bad things came from this Christian visualization --and we can say this without considering it altogether impossible that there is something like a satan, only that a satan is likely to be an ungainly thing and likely to have very little to do with sex. And this 'bad' thing, of course, is somehow in God's own story, so that ultimately the role makes sense relative to what God and the muses really want, though in intricate ways.

Such a pattern can be discerned in J.R.R. Tolkien's fairy tale visualization of reality, where the most

'godly' beings--Tom Bombadil, Gandalf, Frodo, Sam and the elfin beings--have their own sensuality intact, and in which here is mostly sadness associated with 'the bad one' --namely Gollum. There are no horns on Gollum, he is not attractive. He is a force of mischief, but only within a sort of cosmic scheme that makes good use even of this mischief. The wisdom of Tolkien in constructing this visualization of reality goes together with his lack of pointing out of the fact that he was a believer in God, and more or less a Christian: but clearly not in the sense that the followers of Christ made Christendom into through the Christian Bible and their typical interpretations of it.

The Elfin language is, perhaps, a pointer to the computer language, that is engaged at the core of reality to "run" this reality rather as if it were a computer game. Perhaps we can stretch the fairy tale intepretation further, and suggest that the mountaineous caves and the rites of the dwarfs with their axes represent the divine computers themselves. The friendship between a dwarf and an elf--a theme in the Lord of the Rings--is perhaps a bit of a visualization of how a longlegged beautiful muse, scantily glad in a slight, perfectly fitting bikini, is having a good time in front of some sort of mechanical instrument, a computer, at the essence level of reality.

And so, out of all this activity--involving, perhaps, trillions of trillions of such muses, all centered on pleasing God and being with God and doing his wishes, and with--as in the elfin society of Tolkien--perhaps just some /I visualize three/ muses on top--rulers of all the other muses--they create a set of universes. One of these is marked 'manifest', the others are experiment to see how things go so as to roll things back and try something else if it doesn't work out so well; yet others are there just for fun, and for training.

In all my studies of physics, I know of nothing that makes such a very rich visualization incoherent or

unlikely. It is the habit of 20th century style of scientists to try to limit their speech of the metaphysical to that which is 'necessary' /as they say/. They try to stick near to their measurements, and avoid framing big hypotheses of explanation, preferring the 'small' explanation, when they can get away with them. This has some merit, ie, it helps scientists focus their minds to do observation and experiment jobs a bit better. But it also means that the activity of using intuition to select between one out of many grand worldviews, and to fine-tune this worldview, has not been finely tuned in the school of 20th century style scientists. They have, in a way, been 'popperian' /see my discussion of this notion in other places/: and I suggest that we need a broader attitude, involving conscious calling on intuition, and what I call a 'neo-popperian' attitude.

The art of thinking is thus a quest not merely to 'use' the power of clear thinking in daily life, but also to look at the core of our thinking processes, and allow ourselves to dwell poetically and in big terms on the worldviews that set much of the context for our thinking. And when I have done this, I come to the God-rather-as-Zeus view just skethced.

In exploring beauty further, let us bear in mind such vast possibilities. It is not merely 'adding a little more than half each time, or 618 permille to be exact', and 3, 5, 8, and on: beauty is also an infinitude we should never entirely connect to any simple formula or ideal or photo or manifest being. We need broader terms, we need philosophy, we need a sense of crossing barriers and letting go of formulae when we explore beauty, also.

For instance, let us visualize something like a rosebud as a symbol of beauty. You have surely played with flowers including roses at various levels in their flowering. They are as if composed of an embracing of themselves, in a flowing, dancing, circular, smooth, elegant fashion. A rose has many levels, we might say: the levels embrace

each other, the rose leaves cling to each other to make up a beautiful whole form. Perhaps the universe is much like a rose, in this way: perhaps the levels are held together just thus.

In the next part of this chapter, we'll go into numbers and some computing again--fairly simple stuff in one way, yet very 'hard core' data in another sense. We will see that G15 PMN is, in its own way, a minuscle rose: the PMN is in a way the outer part of the rose and G15 is the inner leaves. We'll have a brief but memorable look at the inner leaves, and see what happens when we touch them.

3.2

Ready to do some byte-dance, some peek and poke directly into the core of the computer?

One of the design objectives with G15 PMN was to make as large part of G15 PMN as possible steady and trustworthy and concrete, concrete also in which numbers that do what, where. Here and there we have had to let things be open to change but the key point of having a good stage for the dance of your mind and feelings to unfold on is that not all the stage is wobbly. Something of it must be firm, knowable, so that you can put force into your mind-jumps at the right points.

That is why, although we can make as many apps as we want to, and every one of them can contain their little additions to the G15 PMN language--and some even contain variations of it--the core is stable. And we can only have a stable core if it is tiny, so that we have had a chance to look at all of it again and again and perfect it.

If you look at the left side of the menu that comes up when you press the ⟨HOME⟩ button of your PC, the G:15 main menu, you'll see that it says PMN and just underneath it says: d/65000 /where the / is the up-arrow in our G15 PMN font, rather than the slash; the up-arrow is technically in the position of the unnecessary percentage sign in what is called the ASCII set of characters, see Vol. I/.

When you click on the / between d and 65000, you get the smallest PMN possible: only two- and one-letter commands, and a tiny set of them, much smaller than in the standard third foundation. The third foundation has fancy two-letter words like KU /add to what is stored, a combination of AD, which adds, and KL, which stores/, as well as a rich set of three-and-longer set of words such as SCAN /which looks for any text bit over even thousands of cards, fast/.

Let's produce a change of PMN, just for fun: a harmless change, that will exist only in an isolated spot of the PC, and which we can then wipe away, cleanly. But it is nevertheless a big change, so big that in fact nothing of by far most programs would work in their present form if we put that change to them. The point of the exercise: that you get a sense of the real force it is to handle the levels underneath the surface level.

In any G15 PMN terminal, you can of course do stuff like this: 25000 /// 5 /// ad /// nn and get 25005; and if you put in su where it says ad you will get substract instead and you'll get 24995.

So let's kick some ass and change 'su' to work as 'ad'. Ready? Steady, go:

In case you have mounted anything, such as AngelPen or the third foundation, unmount it first /so we're sure that the disk K is free/:

Click CTR-Q at the main menu, MNT, and 4 and it should tell that all disks are normal; type CAR to get back in.

Click CTR-L and type d65000 and press enter. You are now at the first of between 800 and 900 cards which follows, and they contain the slightly cryptic code called 'G15'. This G15 is one huge step nearer the very electricity inside the PC. Each bit of G15 translates into a number and each number into a series of 0's and 1's and each series of 0's and 1's corresponds to a series of slightly stronger stream of electrons versus slightly milder stream of electrons.

Right-click mouse so you're sure you're in Edit mode.

Click CTR-C and throw in 888 and hit enter.

Click CTR-L and load in card k1 instead /I assume you have nothing important stored right now at k:1, if so, backup it to somewhere else first/.

Click CTR-T and as you press (SPACE) you confirm that you copy the 888 cards from d:65000 to k:1. In other words, you copy the core PMN to the start of the k-disk.

You can check your copy of PMN in two ways, either by typing in k/1 /with the arrow instead of slash/ and saving it to a place that is easy to reach like g:14 /you reach the place by (HOME) then (PGUP) keys/. Then go to Menu mode /use CTR-W/, and click on the arrow. It starts at once.

Or, a slightly more technical way to do it, good to know in case the mouse at the moment isn't available, but it does require that it is in the edit mode--so I hope that the mouse is available! That is, begin by right-click of the mouse.

Go to k:1 /by CTR-L/.

Click (CTR)-A and hit enter. It will now 'assemble' the code that starts right where you are /up until it finds a line of &'s/. Click (CTR)-X and hit enter a couple of times. It will start it. When you exit the code started this way, be prepared to hit enter an extra time.

Happy about this? Good. Now you know how to start the code that begins at K:1.

I have done a little bit looking around and can tell you that at card K:418 and K:419 the command SU is defined. So go to K:418 /use CTR-L/ and have a look.

The PC has a tiny program inside it, started when it is turned on, in which the translations of the stuff typed in as G15 commands in text are translated to numbers. This translation, or 'assembler' program, is made so that it can help a little bit with the writing of the cards. One way it helps is that it allows comments to put be put in: they each end with the big + sign you see there. So any

word, or several_words_linked_like_this, which ends with + merely tell us what the programmer wants to tell at this point. The card k:418 talks about this being where SU or substract is defined. Okay, cool. Go to k:419.

See it says 'su123'? That's the sort of little text bits that get transformed into the numbers that in turn what we call the G15 CPU can handle /not 'understand', for a PC doesn't ever compete with the human mind/. These numbers are all between 1 and something less than 300 for there are only 2-300 G15 CPU 'instructions' /as they are called/.

Change it to 'ad123'. In other words, where the textbit su123 appear, be sure it says ad123 instead, and press CTR-S and save it back to exactly the same card, k419.

Let us speak in words what we just did:

We have a copy of the core of PMN. At card 418, and also 419, the command 'SU' is defined. It is defined by several G15 instructions, a bunch of them on each card. One of these instructions are 'su123'. We changed this instruction to 'ad123', and saved it back into the private copy we have made of PMN. Let's now start the private copy and see what it does.

So go to card k:1, via CTR-L. Be sure it is in the Edit mode /right-click mouse/. Click CTR-A /// CTR-X /// and it starts. Now:

25000 /// 5 /// su /// nn ///

In other words, we are instruction the PMN to SU 5 from 25000. What does it say? 25005. Cool, eh?

If that doesn't tell you something of the power of going one level underneath the manifest level, nothing will :)

Put in kinder words: when you do a tiny precision change at a deep level, you can get the most vivid results at the visible level. It is a mere change of two letters at exactly the right place, and, voila! no more arithmetic. It is all a mess, but a very predictable mess, as long as we restrain it to one copy of the core and to just some definite experiments with the core.

However, as means to learn G15 core programming, it is, as metaphor, equal to testing a drill by drilling a hole through the table.

To rectify our conscience, let us do a fresh copy of 888 cards from d65000 to k1 right now. It will again work as it should, the good ol' su. /Note that while you can use CTR-X again and again, it only takes in the cards afresh when you click CTR-A first. So after copying the 888 cards you do CTR-A first, and CTR-X second, to see that the refreshment of the cards in fact did work./

Now the table is complete again: the hole has been 'un-drilled'. This is one of the chief wonders of computing, as I see it: that you can do incredible damage within a limited confinement, then undo all of it and the PC is blissfully happy with you; nothing of the slow process of forgiveness one sometimes find with some people who ought to find a better foot to get up with in the morning. The PC is, or can be, one of your besties; it is your tame dog, always present as a subserviant element; it is your blank canvas, ready for anything and ready also to be instantly cleared at your command. Now it has to be said: this is also a design objective with G15 PMN. We are talking not of any PC, but of the G15 PMN PC. It is perfectly possible to make wobbly PCs where everything done is leaving traces and nothing is completely erased and where only a subset of commands are followed unless the PC receives background commands to back up your command.

The G15 PMN PC is a good mind-enhancement, especially when the G15 PMN is related to also through programming. In that way, it is the claim of this author that it is uniquely 'human thought friendly'.

3.3

If you actually followed the previous part of this chapter and it also was at least fairly new to you, it was tough

going. It is tough going not because there is a lot of code there, but because it is 'strange' and 'deep', full of implications, both easy to imagine and more far-out.

To go one level underneath the PMN programming language and to touch on the G15 core more directly is a bit like a discussion of worldview with your friend. Are there really muses, overseeing all, even though they are invisible? Do they sometimes overtake a mortal human body, as the ancient Greek and Roman myth love to tell, and 'speak through' those we meet in the flesh?

Perhaps you will find that such a discussion, though it seems utterly far removed from all practical issues, later on shows you something of the attitudes to concrete things such as agreements, clock, plans, timing, even music, dance, sexual entertainment and so on. Obviously, somebody who is religiously engaged tend to look at agreements in a light that is deeper than, 'easy come, easy go'. The agreements have got to be "right". And if they are "right" it also matters that they are kept. Making plans for the future together with a person who has a lot of conscience is usually a safer bet than with a person who proudly proclaims that she has 'no morals' /an easy thing to say for an atheist/.

In that way, we are touching on the relationship between daily life action and philosophy. Philosophy lays the ground for harmonious daily life: a harmony that can also be vastly creative and exciting and full of interesting surprises, if the philosophy is truly well tuned.

In the same way when it comes to programming: a program is made at a certain level, and there are always some levels--at least one, as we saw--underneath it. And the levels underneath it have got to be well-thought, clear, orderly, and yet not orderly in a too narrow sense. But rather in a sense of vast potential for good order.

When we are looking at robot programming, we are interested in having programs that can, more than many other types of programs, match on patterns in what comes

in of data e.g. through the cameras of the robots, and from these matchings produce changes in the actions of the action-part of the program. We don't want the PC to wholly re-invent itself: that could go astray in a bad way and we need a limit on how much a robot can engage in mimicking natural human learning processes.

A way to limit it is this:

imagine that a bit of a program is put into a long list, with other bits of the program also in that list--and some bits of the program not yet on the list. Let us further imagine that the program bits can select to put in, or remove, some other program bits from the list. The list can also contain some data--eg the data from a recent camera viewing of the robot--and some program bits can go in and do modifications of some of the data in the list.

Each second, or more likely, many times pr second, the PC has a 'master control program' that runs through the list of program bits, and perform each and every one of them. Each program bit may contain some data as to whether it should be performed or not: a sort of "on/off" switch, some number like 1 and 0, stored in a place, that can, in a flash, activate/deactivate a program bit.

Sometimes the list gets a bit chaotic in its sequence. Imagine that a program bit puts in another program bit to the list but marks it that it should be performed just about before every other bit. How can that happen?

The list has got to be sorted. For this, we have a very simple method, that works in many circumstances, and that constitute a sort of classic program. It is called, often, 'bubble sort', for it sorts in a way that can be visualized a bit similar to how bubbles may float up to the top of a cup if one blows bubbles into it with a straw or something.

To talk of this sort is a bit like talking of infinity: it is a theme that should be done in moderation, for no matter how seemingly innocent the theme is, it has something a bit mind-bending in it. Something hypnotic. Some-

thing that can put the mind on an edge if over-done too many times in a short while like a week or a month.

The 'list' we talked of in this part of this chapter is not merely a flimsy idea: it is actually a list in the G15 PMN robots. It is more precisely a matrix, for it is a list that has several levels. The program bits put a sort of 'level number' to themselves, and within each level, it doesn't matter which sequence the bits are performed in. The 'master control program' will move upwards through the levels, to gradually higher and higher level numbers, until done, and then loop to the beginning again. And each time the levels need sorting, there's a bubble sort, built into core Third Foundation G15 PMN, to do it for us.

The robot programming is, as you perhaps know, also called FCM, or First-hand Computerized Mentality. The robot programmer 'puts' something of his or her mentality into the program. The whole idea of a matrix of program bits is a bit like having a mind to do something in a certain manner, with some open sequences within a master sequence--a bit like sex--as a metaphor we have frequently called on so far in this volume, and will keep up as a theme throughout.

Among the themes touched on in this chapter is just that bubble sort, also technically. If I may suggest so, do not dwell on this chapter for days in a row. Look at it at most once pr month, and don't waste any time over it if in that same month you work on infinity questions. Don't burden your mind with questions that can put it into self-referential loops for any too much: it can fray your brain, make you self-enclosed. Don't risk burning your brain, it is your most valuable asset. With your brain, you can lift your body into the style and fashionable radiance that evokes the sense of beauty in others. But it has to go through your brain: and for that, your brain has got to work well, really well. And for your brain to work well, you mustn't overburden it with the most cryptonitic parts of the art of thinking. /The word 'cryptonitic' is

given a certain sense in some of the writings by same author; please look up these writings if you haven't already acquainted yourself with them./

3.4

What is a fact? Fact goes beyond observation: it is a way to summarize observations, sensations, experiences. When there are competing ways to do so, things get complex. It is a task of philosophy to find out what are the good ways to divide complex tasks up.

In heated arguments, one person may stick to one way of summarizing events, and another person may stick to another way. These ways can sometimes be called stories or 'narratives'. They usually, when forming part of an argument, involving dealing out implicit or explicit guilt to someone, or asserting causes that means that something is wrong somewhere. Each narrative can have a set of accusastions in it. Each person may defend own narrative as 'truth' and try and label the competing narratives as 'fictional' or even 'delusional' or 'crazy'.

When looking closely at a narrative, it usually consists of many sub-narratives, or tales; which in turn consists of summarises of ways of seeing and observing and experiencing something.

A calm philosopher, well educated in the art of thinking knows that every way of observing need to be independently looked at and checked separately and that this usually requires much time, indeed it requires harmony and tranquility and a kind of musical intuition that is in essence willing to listen to what presents itself, rather than forcing any narrative on top of observations.

Let us construct an artificial example of this, in which two people are having an argument of whether to bring rain clothes to the walk or not. One person has a narrative that the other person is always wanting to bring

too much clothes and is always misperceiving the weather. The other person has a narrative that the first person is always living on positive expectations in a short-sighted way and tends to get a cold as a result. At issue is also some really dark clouds that are seen outdoors on the left and they appear to move to the right and will shower down soon--or they don't move to the right, but rather will vanish completely.

The example may seem artificial but the question of observing a very, very slow movement is worth musing over: anyone who has tried to make up her own mind about the direction of movement of a cloud when it moves very slowly will have a philosophical experience teaching how the mind is actively forming 'theories' in order to help its sensation of reality. For instance, at one point, you form the opinion, the theory if you wish, that the cloud is moving to the right. Stick to that theory and keep fixating on a point in the sky. If you are right, you will get a sense of confirmation of it after a while. That sense is vague, because fixing on a point in the sky is usually complicated--because we usually have no reference other than the clouds themselves, and they are moving.

Therefore, you keep on observing the clouds for a while with the opinion that they move to the right in mind.

In order to be unbiased, to get a fair summary of the movement, you must however try other 'theories' of their movement. At the moment, let's stick to the left/right direction, and assume that the clouds are either moving very slowly left, or very slowly right.

To check the cloud movement honestly, you create an alternative movement idea, or theory, namely that the clouds move left.

After a while, you will have got a sense of more confirmation of one theory than the other.

In your mind you let a 'bubble sort' select between the two theories: which one got the most confirmations? Which theory, in other words, led to the most effortless sense

of matching reality? In yet other words, which theory gives you a tranquality and harmony inside?

In brain science, we are speaking of 'alpha' waves, associated with harmonious thinking.

Each narrative consists, usually, of bundles of tales, each of which relies on many 'theories'.

Ideally, we would not stick to the idea of evaluating narratives very often. Rather, we dissolve the narratives, check each theory as well as we can. Reaching harmony on each point, we allow new narratives, if we wish, to be formed with these harmonious observations of facts to build up. This fills the brain with alpha waves. The beta waves are associated with nurturing points of conflict. Alpha waves are associated also with good-natured laughter and the type of masturbation, heavy petting and sex that has a quality and flavour of meditation about it.

A positive role of programming is that when a program has got to be corrected, we need to form a theory of where it has got wrong, and that theory must be right for us to formulate a good plan on how to correct the program and to implement that plan. This plan may be conceived in a split second on observing an issue with a line in a program, or--on the other side of the spectrum--it may dawn on the programmer as a fundamental issue that the programmer simply didn't think of when the foundation of the program was made, and that may require rewriting of up to a third of the program, or even that the whole idea of the program has to be fundamentally changed.

This positive side of programming, in other words, involves a teaching, through the interaction that the programmer has with the Personal Computer running the first-hand thought-friendly programming language, of what it takes to form well-founded theories and abolish meaningless narratives. The tranquility of mind on securing a right perception, a correct way of seeing on how to make a program work right, has sometimes an ecstatic flavour, and that flavour comes from the sense of

order and art about it, the sense of beauty of the structure as experienced through a narrative that is fundamentally true and having a pulsating, meaningful relationship to reality at all points.

In exploring complex situations, where it is far from obvious, at first, what is right action, the intelligent thinker learns to trust the art of effortlessness: which is another expression of the natural 'inner bubble sort' of the mind, picking in an almost automatic way the opinion or theory that comes to rise to the top of the 'list' of alternative theories, considering how much verification or confirmation it has got and how little contradictory evidence. Each instance of confirmation is in turn a result of a spontaneous 'inner bubble sort' of reaching a point where the mind nods to a possible insight, a perception.

To live absolutely by facts is, we can propose, only available to a being of absolute enlightenment, and that goes beyond what mere manifest human beings can achieve: and it can be argued to be a severe hubris to try to achieve absolute factuality; and a severe lie to project to others the idea that this has been reached. Of course, the sly egoes who wishes a grand audience and who craves the artificial pleasure of having disciples and being a guru to these people may be hypnotically convincing--at least to stupid people--that they have absolute or near-absolute enlightenment. In the 20th century, one of the most fascinating sly egoes, who mesmerized countless many stupid people, was called "Osho": he got people to talk of him as "on a really high level", and indeed also being "absolutely enlightened". Another was Sai Baba. Yet another was Maharishi Mahesh Yogi. Another such sly ego founded a teaching called Kriya Yoga. And every world religion has had its sly egoes, setting themselves up as more or less absolute--the Dalai Lamas, the rabbis, the imams, and so on. All these sly egoes usually had, however --and that goes for Osho as well--something magnificent to

teach. That magnificence in the case of Osho was his freedom to view sexuality as perfectly valid in a quest to go deeper in meditation. All else about him was rubbish.

Another teacher, Jiddu Krishnamurti, had the magnificence to clothe the language in Zen in a brand new way using only normal English words in an extremely poetic way. Apart from this, he was a sly ego like all the others. Somebody who cloned this language but made it more extreme was the teacher who liked to call himself "U.G.".

In the 21st century, we have countless new types of clever sly egoes, gurus who have browsed through the books of thousand sly egoes and put together their own particular mixture. They may have a wolf's smile on the back of their sly teachings, and yet be in a position to mesmerize thousands and yet mor thousands of followers to speak of them as prophets and pure lights and a real streak of absolute enlightenment and innocence come into this world.

It is easy to see how the sly egoes have an appeal to people on this planet: the planet is miserable and damaged in many ways; countless people are living in poverty; most official 'top ten' lists of winning people and entities are corruptly shaped and bought by powerful people; wise actions are avoided; unfair actions are implemented by societies whose leaders glorify themselves as democratic and noble; cities are so polluted the Sun is not visible for days at a time; rivers dry up and the little water they offer is colored by factories which ignore global environmentalism concerns; and add to this the endless conflicts, fights, wars, and newspapers devoted to selling narratives rather than exploring facts. It is a momentous stupidity, and when a sly ego, a king of astrology and EEG and meditation and kabbalah and zen and sex comes around and points out how bad the world is and how lovely it is to follow the calculation schemes in his or her books, of course there will be followers--maybe a portion of the planet will follow. It is all part of escapism. The

real religion lies elsewhere. It doesn't lie in the glorification of the unfactual narratives provided in the spirit of escaping this planet; it doesn't lie in paying tribute to the prophets who pretend not to be prophets while they are spending every minute of their daily lives trying to be prophets. These sharks of the human psyche are creating stagnation: and yet every one of these sharks usually could not have power unless they have something factual, something good, something true, shining as a power in their belts, in their armouries.

The religious seeker, the seeker after fact, will therefore do well in educating herself to listen to facts without bias; to learn to dissolve the narratives of prophetic people into sub-tales, and the sub-tales into gradually finer levels of sub-tales until the level of individual sensory experiences are reached; and to apply own mindfulness and attention to each proposed theory.

However, a portion of most religious proposals involve things or beings which do not admit to direct sensory experience. This means that we must go beyond the criterion that each theory is to be checked merely relative to sensory observations. We must be willing to engage intuition relative to each bit of each tale also when these bits include references to a reality that is beyond what can be seen with the eye.

For instance, let us create, as a thought experiment, a variation of the cloud observation idea we talked about earlier in this part of this chapter. Instead of talking about whether a cloud moves to the left or to the right, let us talk about beings that are beyond sensory experiences. I have earlier mentioned the word 'muses' to refer to just such beings. Let us imagine that we share in an intuition, a mental nod, that such beings indeed do exist; and as a thought experiment, imagine two people arguing over whether the muses are taller or smaller than themselves. This may in turn be part of a larger narrative involving some kind of false prophet, a sly ego proposing

certain things about reality in the pretext of having some absolute enlightenment.

The religious seeker will, in the opinion of this writer --in such a case--do well in being sceptical about the glorification of any person's 'absolute' insight into anything; and yet be willing to consider--as theory--that it may be correct that the muses have a height that can be conpared to the height of, let's say, the typical child or the typical adult.

The way I see it is that this is much /but not completely/ the same type of mental activity than that of observing very slow clouds in motion. You form a theory, and wait, quietly, and get a sense of whether it is right or not. It has to be a simple theory, like, 'the cloud is moving to the left', or, 'the muses are no taller than children, at least not usually' /by 'simple' I mean that it feels to be a simple theory given the background of word usage by that person; at a later stage, different words may be used in order to create a sentence that feels to be simple--for instance that muses are considered to be of one height when in one 'mode' and another height when they are in another mode/.

The theory you wish to get some information about has to be entertained in the mind in a flowing way. This flow comes after fixing the mind on the theory for a while. In the old Sanskrit language of India, it means going from the phase of Dharana to the phase of Dhyana. Dhyana is the word for flow that in Japan became Zen through the Chinese Chuan. In sticking to the flow, it can get an element of harmony, even ecstatic harmony or tranquility, and this is in Sanskrit called Samadhi, and it can have as many levels as orgasms can have. The triplet of Dharana, Dhyana and Samadhi is, in a lucid tiny script in India, attributed to a certain Patanjali, called "Samyama". That is a religious term but we do not have to be fundamentalists in yoga to use it. The fact oriented person can be eclectic relative to all religious books and scripts, and deny the validity

of any form of nationalism or secteranism, including that of yoga.

Samyama is therefore the way to decide on any question, from the movement of clouds to the heights of muses. We note that Patanjali also mentions its applications in generating events: this is the most ingenious part of his production, that he sees the same mental instrument that does fluid perception as the instrument that can affect reality in some deep way. In other words, samyama is both meditation over fact and what religious people refer to as prayer. It is both intuition and telekinesis. It is the brain/mind/body/soul/spirit organically working as a whole both with itself and with all reality. In this language, samyama is the essence of the art of thinking.

And, by the way, this writer's samyama suggests that the muses are real and suggests also that their heights are, typically, as that of children.

3.5

Imagine a world entirely without computers, just as a thought experiment; a world in which there are buildings, some form of music instruments; food; massage; some kind of cars; this and that sort of nature; drawing classes and painters; cooks and restaurants--but no computers and no option to do even the faintest element of programming in front of a personal computer. There is literature and there is dance, but no computers. In school, it is taught that 2 plus 2 equals 4 but there are no computers.

You have got the vision? The thought experiment?

Let's for the moment not say, 'Oh that's very easy because it used to be so all over the place according to the history books.' Because 'the past' /whether described accurately or not/ is in a sense also a vast piece of literature, a grand type of fiction given an extra

credibility, filling up our minds with interpretations of where we humanity happens to be and where we are going.

And the thought experiment is going to give us a vision and a feeling of that vision; the vision of a society completely and utterly devoid of computer programming; and I wish us to switch between that thought, that vision, and the vision of this our real society that does offer computers, indeed personal computers, and the opportunity to type in commands and having the computers obey them completely.

When you meditate quietly over such a vision, such a vision experiment, and switch between that and the vision of the reality, and give it some time, in privacy, sitting still, listening perhaps to music but not deafening music, just quiet music, you will get, sooner or later, two different sensations in your body. The sensation of the vision of the computerless society, and the sensation of the vision of the society that offers personal computers for programming.

Along with the sensation, which may physically be felt as a sort of mood in your gut, may go certain perhaps vague or perhaps more concrete image glimpses, and some words or phrases or sentences may come around in your mind alongside these glimpses. Pay quiet attention to it all, be mindful of all these sensations and mental images. Encourage the visions to develop a little bit.

After a while, you may find that, quite spontaneously, in your silence, in your witnessing without prejudice of these two societal visisions--one real, one imaginary--the mind sums up a vision through a feeling, perhaps through the word of a feeling. And it may seem to you that your mind has a life on its own, beyond the idea that it is 'your' mind and 'your' thoughts; rather, a part of the mind is giving the rest of the mind some kind of impulse or 'mental nutrition'. This mental nutrition is as it were mentally digested in phases. And in this process of digesting the mental content, various parts of the mind

are activated and as it were speaks back to the sense of yourself. You are not controlling your mind as much as experiencing that the part of the mind that is most you, most yourself, is in a kind of dialogue with other parts of the mind that may be more associated with the feeling of the whole body, the feeling of life, and which are capable of talking back but may do so more fleetingly and more as a suggestion.

In a way, the part or aspect of your mind most giving you the feeling of 'being you, being in control', is like a boss of a company where there are several employees, each having unique talents, temperaments, ways of approaching things; and while they are all in principle obedient to the boss, they are also living and quite their own beings and the boss shall have to have a humility to that fact and not merely dole out instructions to them.

In giving a bit further, we can suggest that these parts or aspects go beyond you as body and brain and go beyond even your personal mind to touch on levels of shared impulses between people and even beyond that, with reality as a whole.

On occasion, you may even find that there is a concert of expressions that can arise from something beyond yourself: indeed, every sentence of this section of this chapter was written without conscious control; was written, indeed, without the slighest conception in the "I" of the writer what was about to come; was written in the middle of the night; the only starting-point was, "hm, there's a sense that something ought to be expressed, don't know what but let's take up the B9 edit and see if something comes around. I mention this also to point out that the aspects of the mind that appear to be silent or very simplistic about words--perhaps on occasion just providing one or two words to sum up a situation--may, when given the proper tool and mood and time, be found to be subtle and rich also in verbal communication capacities --which is a good thing.

I dwelt on the two visions of society, one without computers and computer programming, and one with--and, as you noticed, I drifted a little bit and talked of something a little different. In again giving mindfulness to the two visions, I assumed that the mental work had continued at a subconscious level and that some more mature results could now be extracted. There is a sort of innocence in returning afresh to a topic after having introduced it, then left it; and in this innocence, there is less pressure from the more conscious part of yourself onto the subconscious, or other parts of the mind; and so less bias; and so deeper and better and more coherent intuitions.

The computerless society--the vision of that--is, at this stage in my mind, giving rise to a word, 'sadness'; and a sort of metaphor--food without spice; soup without structure; and the society with programming gives me a vision of something with a kind of bluish electronic sparkle, metaphorically associated to hot chili spice added to food so as to make it more holistic and more pleasing; the sparkle I imagine to be as if fairly square as a sort of frame around content of other type; and these frames are as if beside one another and having the dazzling finesses of electricity.

The art of thinking surely involves cultivating a mindful dialogue within the mind and without the entirely unnecessary preconception that the mind is bordered in or generated by or limited to an organism, even though the mind as experienced by a person usually has a sense of a central locus eg in the brain. The human brain must be as healthy and whole as can be to allow the mind--which does not exclude heart, nor exclude feeling--to flow through it. A damaged human being whose brain is only partially functioning is out of resonance with the mind-capabilities of this human being. The brain is necessary but not in any sense sufficient for there to be mind. Any prejudice involving a view of the human and a view of the world such

that mind is 'generated' by 'the physical processes' of the human body with its brain is, as I intuitively perceive it all, a stupidity, and it is not just any type of stupidity, but a generative stupidity; and in a young person, it is a kind of pre-senility to have such an atheistic view of human mind. Nothing is quite working as it should in someone thus severely biased as to the nature of the human mind. A society full of a cultural conditioning of the atheistic prejudice of this sort connected to the vision of the human being is a bastardly society indeed and no politician or political ideology can go anywhere near in being medicine enough.

The human mind may have a sense of a controlling centre, but this centre must in a way suspend itself in order to call on the contributions from other aspects of the mind, which may be as much centres as the centre of the "I".

Pushing it, the mind must not always call itself "mind", as if there is an inherent clear-cut distinction between mind and matter or between what we call mind and everything else. The flowing undivided attention that becomes a meditation is at times wordless and visionless and in such a state of silence or samadhi or union or nothingness or nirvana or whatever we call it, a fresh insight or even a new reality may emerge.

Part of the dialogue of the mind with itself is not just to 'wait' for impulses, but also to raise questions into awareness and allow the question to 'work', rather as you are letting light and water and soil be applied to a seed so that the seed can begin to sprout.

The question I wish to put to the vision of the PC-less society, which in my mind was summed up as 'sadness', is this: why sadness--why are computers so seemingly necessary--why is programming so important?

A question leaves something out, but includes an intent to get something in.

That intent communicates itself to the rest of the mind, conveys itself.

The intent may be called 'a desire', but it can be very calm, and unlike the feverent desire to eat something when one is intensely hungry. The intent to get in information associated with a question to oneself is a kind of 'sense that there must be some information coming in', and a trust goes along with that sense; and a patience as well. Not an infinite patience, but a patience involving an interval, a pause, a wait. In Patanjali's terms, it is not exactly 'samadhi' in its most extreme form, but rather, 'samadhi with a seed'. The seed is the quiet, persistent urge to get feedback as to the lacking element in the insight.

Supposing that this writer has a well-functioning mind, and not a mad mind, it is a valuable quest for this writer --and perhaps for you as reader--to get an as accurate description as possible of the sort of rules of thumb that this mind applies to itself to get its normal processes done. And this is exactly these five volumes of the Art of Thinking.

I still have the question, 'why is a programming-less society sadness'--and my intent is to complete this section of this chapter with a novel proposition, a kind of explanation that is not merely yet another metaphor.

I am getting up impulses now, in my dream-near state of mind now in the middle of the night, that quietly and yet strongly answers this question in terms that we have discussed earlier in this book. /My approach of writing each book is that of trying to avoid editing too much later on, rather allowing the 'mind-field' of the book as it were grow gradually and unfold more and more./

We have sometimes used the word-pair 'manifest' and 'subtle'. Do you know that word-pair well? It goes into the answer to the question. Look up these words if they still feel foreign to you. Manifest, 'mani-fest', comes from roots meaning something you can manually hold, hold in your hands; subtle, 'sub-textere', comes from roots involving a texture or pattern underneath or in between,

as fine threads in a piece of clothing with a distinct pattern that can only be seen by looking closely at them in a certain light.

The manifest world versus the subtle world: quantum physics and spiritual experiences hint towards visions of the subtle world; the manifest world is directly measured and experienced through sensory organs. The subtle world lies underneath the manifest world and creates it, moment by moment. And in this subtle world there are structures that the manifest world can have words for and images of and ideas about. We have earlier on talked about the idea, the natural proposition, that the subtle world has computers; has programs; and that this is one of the features that makes the beings of the subtle world very happy--that something is going on all by itself; and yet can be intelligently changed by changing the programs, or operating on the programs.

In order to even say this sort of thing--that the subtle world has computers--we need the word 'computer'; and this word is full of meaning to someone who does programming. The meaningfulness of the word computer underlies our proposition that the subtle world has plenty of computers and this is one of the key parts of the subtle world, alongside living beings, the muses, and God.

It is a happiness for a human society to have a vision of the subtle world, and thus of the universe as a whole; because this vision is also a map that allows and even encourages intuition and its wisdom to flow through the mental processes of this human manifest society. The vision of the subtle world is utterly more complete given the presence, the easy, strong presence of the concept of computers; and so it is a happiness that society has a presence of computers and programming. And this explains why there is a sadness with a computerless society: a society lacking in computers is a society lacking in good concepts covering the subtle world; its mind-map is in poor shape. The healing of the mind-map of the world takes

place by inviting the technology of computers to be part of the human society, and allowing programming to be taught to teens and preteens and post-teens.

Programming is only programming when it hasn't too many layers to it. When the layers are so plentiful that the true mechanical nature of programming, its digital features, are given a softy touch, it is no longer as much computer programming as computer 'use'. In other words, a programming language must involve a sense of the nude and raw hard core of the computer and not being a thick sweater around the computer in which the pretense that it is 'smart' is given an upper hand. The computer as such is of course neither smart nor stupid just a machine. The programming language must be a language that is fairly near the electricity and the patterns of the central processing unit, and not be any too much an 'application' that covers up the nature of this processing unit. Each computer must physically be shaped around one processing unit rather than a bundle of them to fortify the clarity with which the computer is indeed a machine rather than some kind of groupy semi-organism. The CPU in the computer must be simple in its instruction set and not have a separate level of instructions that oversees the first part of instructions.

The G15 PMN Personal Computer design fulfills all these design criterions totally and completely. Its design is such that the G15 PMN Personal Computer can be 'emulated' on computers with a complex non-elegant CPU type /or even having a bundle of CPUs/, but its real bonus to human manifest society is when it is run physically on a G15 PMN computer.

3.6

If you have used any battery-driven mechanical device--in which many things of it are analog, not merely a computer,

--then as the battery gets weaker, so does the whole device tend to behave a bit like an exhausted animal.

When you are working on creating a masterpiece from your own mind, you want the battery of your brain to be at top charge; you don't want an exhausted state of mind.

Why is it so that a great deal of meditation, and also meditation involving masturbation and porn, can strongly help recharge the brain's clarity, energy, coherence and capacity to express itself brilliantly, whereas a brain that is lacking in sexual activity over several days can become creatively exhausted?

The 'why' can be approach from various angles--religous, metaphysical, physiologial, psychological, and more.

One take on it is this: the brain couples sexual energy to fascinating holistic shapes, including those involving fractal similarities and golden ratios, during a proper working itself up--alongside the body--to orgasm in such a visual and rhythmic co-experience.

I am using complicated words in this part of this chapter because the conclusion, put in a cut'n'dried form, sounds slightly absurd, when listened to in a typical 20th century connotation. It goes like this,

'those who masturbate better, get smarter'.

Never mind how absurd it is, it is true. And masturbation is the beginning of sex, it is a form of sex, and there is no break in continuity from self-sex as masturbation into having sex with one or more partners; and the many exciting middle-grounds include heavy petting and sex-talk while masturbating together.

A slightly related theme is this:

the beautiful shapes of a sexy attractive longlegged elegant young lady are fascinating to the brain--and in fact help its coherence--because they are, while harmonious with themselves, not monotonously repeating themselves exactly.

Take a bunch of circles and squares for comparison. It is a yawn, too obvious.

Take a pair of well-trained smooth buns photographed at a fascinating angle with light from a good angle. It is circular yet not the geometric circle. Look at the lines of her thighs. They are parallel but not the geometric parallel as in a square or rectangle. Now look at 15 or 20 or 40 photos of the same girl doing good poses, taken out from a vaster collection; the most tantalizing of the pictures selected and presented. The girl may in turn remind you of something in someone you find yourself powerfully attracted you for the time being.

Then you move from one such collection to another, and after half a hour or an hour your brain is full of--what? Don't just say, 'it's full of porn'. It is full of tantalizing shapes, living patterns, hinting on harmonies and resonating in ways that sometimes involve the golden ratio, sometimes have related, perhaps other forms of form-simliarities. Which means what?

It means that the brain is full of a sort of general capability to perceive: it is full of percepts, with which it can perceive--anything. Not just girls. It can perceive itself, its thought processes, the city-life, the books, --the mind comes blazingly alive by transfiguring its hormones through beautiful pornography. The best programmers are also the best pornographs.

We have earlier on, in this volume, briefly discussed how the golden ratio, such as in a line that is size 5, harmoniously relates to a line that is about size 8, in a way that is pleasing to the eye. This is true when the lines are beside one another, on top of one another, put together in a rectangle, or in one way or another related as part of a more complex pattern.

In terms of making musical sounds, looking for a golden ratio is like looking for, at least, two or three sounds that go really well together without being identical. If, for instance, you sing one note and you go one octave up, and sing that note: that is almost the same note all over again, right? But find the third note in between that is

about two-thirds slanted towards one of them, and it may suddenly be that it is a vibrant chord. A kind of warmth ripples through it, but gently and with sustained harmony. A world of just boxes and spheres are analogous to a soundscape of just the same note at various octaves.

Our experience of beauty and harmony leaps into being when we have something that resonates without--to put it that way--resonating completely or absolutely. We might put it this way: a chord has in it a kind of potential lack of resonance. If you could freeze it in time and play it only for a little while, the waves would seem to go out of sync--but because of the ratios involve, they come back into sync quickly; and this is notably more fascinating to listen to that monotonous perfect symmetry in the waves.

Now all this is not to say that beauty 'has a formula'. One may understand this fairly well and spend half an hour dutifully in front of some porn trying to masturbate and still make something very other than a masterpiece! :)

To make great beauty, you must also have detailed knowledge of particular patterns inside that beauty which, even though not noticed by most at first, would be noticed when given time to experience the expression many times. This detailed knowledge may be itself nothing other than yet more perceptions of golden ratios required in such-and-such area; but knowledge it is, all the same. For instance, to be able to select a really great photo, an piece of art which is also a photo, from a vast selection, requires a glancing on many levels at once of the photos. The ratios must make sense but also the ratios in the more general sense--in the sense of the harmony of the whole healthy functionality of the human being or beings in the photo. Perhaps all levels of all such functionality, including the high wrists of her feet, can be described by some use of golden ratio; whatever is the case, it all has to work together when, as a photographer, you pick the best of the best photos, and present them.

That many-levelled glancing is also taking place when the brain/mind is at a highly energetic, highly coherent level of functioning, sexualized and ready to laugh in a good-natured way, and purely conceptual questions are considered--such as programming tasks, or how to elevate the revenue for a company. It is a fractal glancing, that perceives wholes within wholes, senses the unfoldment, tunes into the movement, and have intuitions into how it is going; and by selecting right at each step, the best possible unfoldments take place, for sure.

3.7

Earlier in this volume we suggested the playfully absurd and yet perhaps deeply meaningful thought, that a number is a way to have sex.

In music, there is no coming around that numbers can be used to count--and arrythmically upset--the sense of a frame of an underlaying rhythm that can be intoxicating.

In many forms of sex, the sense of the numbers, and most deeply so when it is about the first numbers after 1, can shape the sensation powerfully, even overwhelmingly under certain circumstances.

Let us, for the sake of stimulating visualization and insight into how numbers directly pervade emotions, and affect sex, and may metaphysically be speculated to even constitute sex, mention some images:

* 3 tongues meeting.
* 4 boobs meeting, two pairs touching.
* 1 finger plus 2 perfectly smooth rounded buttocks.
* 6 thighs on top of oneself, 3 pairs of thighs.
* 5 pussies on top of each other.

While the happy, highly sexified, expert thinker takes such noble thoughts into consideration, let me also point out, in the spirit of clarification, just how broadly this writer defines 'porn': it includes any human photo that elevates the joy of a sexualized, enlightened observer.

In a similar vein, let me clarify also that there may be a sort of cosmic male poliarity owned by God which is in tantalizing, energizing contrast to the cosmic female polarities shared by all his highest creations, his muses, and humans /esp when muselike/, this writer does not consider that there is any such thing as an 'essential' male/female division between human beings. Rather, each and every human being partakes in almost uncountable many male and female features, processes, energies, aspects.

This is physiologically mirrored in such fascinating observations, made by startled scientists as soon as they begun studying sexuality relative to hormones in human beings and not trying to whether prove nor disprove any 'binary' theory of genders, that such as testoterone is common in the erectile and orgasmic state of the clit and is also the progenitor of male muscles and orgasmic state.

In terms of enlightened sexuality, it means also that the girl can also be the penetrator, such as by her cute foot, by her shoulder, by her clit, etc, while the male perfectly well be the one who healthily opens to such sexual penetrations. And so there is an interplay of what we can call 'physiological geometry' in sexual action in which it makes absolutely no sense to talk of any one as 'lesbian' or 'heterosexual' or 'perverse' in any strict sense. Rather, sex is an orchestra of generally healthy exploration of human sublime anatomy, with or without clothes, machinery or animals as elements, with or without procreation as a result. It is a dance, greater, in a way, than much of life itself, and certainly more powerful than the sadness that any death can cause. It is the celebration of life by life itself, but has in it a meditation that goes beyond mere clinging to physical survival of the

body into the future. As such, it is does not need any further drugs or stimuli; although in extremely well-thought selection and moderation some degree of drugs, like alcohol or cannabis or mushrooms or whatever, can provide fascinating variations. /However to begin to employ drugs routinely and in doses involve a change of physiology so as to be more suspectible for illnesses; which is to say drugs must only be a dot over the i, and not over every i./

In a spirit of having a jam session, interweaving themes of all sorts with a coherent thought but without trying to push just one or two arguments across, let us look at a concrete way to sort a small list of numbers, like 1 4 9 2 into 1 2 4 9. The principle can be used with millions of numbers, if we get it right. We can write such a sorting routine--here we will use the most elegant shape of all, which sometimes is efficient and at other times take more time than other ways of sorting--called, 'bubblesort', and mentioned earlier in this volume.

Like the core of the open robotics FCM platform, a bubblesort of a very practical kind are all part of the Third Foundation. Now when you start up the TF, the Third Foundation G15 PMN, you have at once a top set of examples from which to remind yourself on how to program. I must confess there are months since I did any prolonged, deep series of hours of programming, and the most elementary things had to be refreshed by some glances at TF. I got around to F:2337 to remind myself of how to make an array of numbers--by SZ and putting a quote, like &&, inside a function definition with =. Then I looked around after F:1700 or so to remind myself that indeed AY is how to get stuff out of a list or array, and that brought to mind that the same two letters in converse sequence, YA, are used to put stuff into a list.

I followed up by a couple of SCAN's. That is to say, I typed in SCAN, and begun looking up definition of AY. The most lazy way to do this, but which works perfectly, is to

type in a blank, the letters AY, and a colon : and press lineshift. After that, give starting-point F1, and 9999 as amount of cards.

The PC responds with having found it somewhere in the first few hundred cards at the F: disk. It suggests, among other things, that you can type CAR to view it, and I did so.

I had, while doing this, the concept of bubble sort in mind: and looked around until I was satisfied that I had all the words of the tools at the tip of my tongue, or to be slightly more precise in this metaphor, at the tip of my fingers, so that the commands would come quickly to type in.

Let us in the next part of this chapter talk us through the making of such a program. Before we talk it through together, for us to really share in the thought processes --which may be this writer later on reading my own reflections after having forgotten most about what I wrote--it is of top value to enliven ourselves to the idea of how a list like 1 3 2 can become 1 2 3 through bubble sort. Let us try it with a slightly more complicated list as well.

Bubble sort idea: begin at the beginning, look at two and two numbers. Do a check: are these two numbers in ok sequence? That's a simple yes/no question of a type that a PC is made to answer. There is but one action to consider: to swap the sequence when the sequence is not ok. Having satisfied ourselves, we go just one step ahead in the list, and look at those two new numbers in the same way. /That will include one of the two numbers we last looked at./

We go through the list, and we make a note in the margin as to whether we had to do any swap. That is a yes/no question after going through the list, or 'doing one loop' as we call it: did we do any swap now?

There is one action if we did any swap now, and that's to get on an do /at least/ an other loop.

There is another action if we did no swap in recent loop, and that is to exit the program and letting the PC do other things, on the assumption that the list is now perfectly sorted.

The words can be shown in these examples:

1 3 2 => /1 3/ 2 => 1 /3 2/ => 1 /2 3/ => 1 2 3 => done.

We use here / / to talk about comparison.

More complicated list, several loops, simplified:

4 5 9 2 8 => /4 5/ 9 2 8 => 4 /5 9/ 2 8 => 4 5 /2 9/ 8 => 4 5 2 /8 9/ => new loop => /4 5/ 2 8 9 => 4 /2 5/ 8 9 .. => new loop => /2 4/ 5 8 9 => 2 4 5 8 9 => done.

There was a dogma in the 20th century that bubblesort is sort of for kindergarten whereas something involving wizardry called by many 'quicksort' is for adults. Like many other dogma, they are not necessarily holding up to a good neopopperian scientific look.

Just like bubblesort is involving very many loops for some sequences of numbers, so is ANY routine involving very many loops for some sequences of numbers. The attempt to find a 'universal mechanism' is, in this writer's opinion, depending on intent of mind, either incoherent or has an almost absurdly simple solution. It is incoherent if one believes that by adding complexity to the most elegant, most first-hand, most simple sorting routine in the world, requiring the least computational electronic activity and only the simplest of first-hand program concepts, one will grow towards something more universal. One may 'optimize' for 'a statistical large portion of cases': that however is not what 'universal' is about. To actually sort fast in ALL cases there is something beyond an algorithm required to structure the algorithm, confer Goedel's incompleteness thinking and the 'et cetera' proofs by this writer to reflect along these sorts of lines.

Once we abandan the notion that one can build, by algorithmic complexity, a pathway into infinity, one is

left with the question: what happens to be the most first-hand elegant way? And we are back on the idea of 'bubbles' of pairs of two and two numbers 'bubbling up' to the surface /eg, to the last/ of the list, again and again, like sparkling mineral water. It is absurdly simple: and it remains simple to spell it out on the level of how the transistors, or mini-transistors, inside a first-hand simple Central Processing Unit work.

Only by making second-hand concepts can one create the apperance, the illusory appearance, that something like the hierarchically organized socalled 'quicksort' is in some way 'simple'. Quicksort is not simple and in some circumstances it is better named slowsort.

A computer does not, repeat not, offer limitless power. The very quintessence of the computer is limitation. That includes limitation on sorting speed. Knowing that, we build programs. Knowing the fullness of a first-hand computer structure like G15 PMN, we employ bubblesort, unless there are particular technical applications that, in exceptional circumstances, demand squeezing the lemon of the computer.

As a next step, building on bubblesort, one can have a first-hand human organized 'index' method, in which portions of a larger list can be kept separately on disk, a sorted partially. However that has to be built by a first-hand understanding of the typical statistical structure of the data. For instance, if you have a very large name list, all sorts of English names, you have good reason to assume that there will be some degree of distribution over all the twenty-six letters A..Z. Which is to say, you can have 26 set of cards, each set capable of being read in as a chunk and, eg through a separately stored number index /that indexes the card numbers/, sort those names.

This situation is quite normal for very long list of words: it is not that the words themselves are directly sorted as much that an index of them. It means that

instead of comparing two numbers like 3 and 8 you compare the name that's stored eg in card 3 with the name stored in card 8.

For yet longer lists one can have a third level.

But the smaller programs look the best.

And bubblesort is a small program.

Yet, even a small program, as long as it has at least a handful of actions and features in it, can be shaped in very many ways, in a language like G15 PMN. In some cases, we can make the program very small but at the expense of readability; or we can make it longer /eg by using G15 code directly without going through PMN/ so as to make it faster but at yet more expense of readability; or we can make it pleasantly long, not too long, but psychologically meaningfully long, divided into meaningful chunks, so that we can read it as well as the PC.

While this may sometimes lead to some more electricity wasted by the PC at first, we can save in later on in case we need to use the same program in a different context and wonder what we the heck we were thinking about when we wrote it. There is a communication gift into the future to do first-hand programming. And the TF is made on the absolute premise to ensure that first-hand programming is stimulated even as practical applications are made.

In other words, we are long-term encouraging both enlightenment, thinking and also efficiency by sticking to first-hand programming in a first-hand programming environment, in which optimalization is never used routinely but only if there are extremely good reasons to call on it.

The G15 CPU is designed with this principle in mind. Nearly all its essential instructions have a sense of leisure and luxury in their elegant sophistication and ease with which they operate. There are a few instructions that are called on very many times every second to do a whole set of routine tasks in the same sequence each time. These few instructions could be omitted; but then

electricity use would go strongly up and speed would suffer for every letter you type on such as a B9edit editor program. The whole paradigm of making the G15 PMN CPU would have to be altered to give human acceptable processing speed for all elementary G15 PMN applications without a couple of these instructions, which bundle a dozen or more of other instructions into a single instructions.

The chief example of this is the G15 command that is handling the calling of a PMN program from within another PMN program and the subsequent exiting of that program again--with room for calling within calling within calling. Another important example is the command to light up the pixels on various positions on the screen in various intensities given a range of numbers beside one another. And there are some more.

However, we are still talking less than 300 instructions and there are no "meta-CPU's" that overlook the CPU. So it is the simplest CPU possible to relate to practical electronics, practical mini-transistor speed, at practical temperatures, allowing first-hand programs to get the human interactors a whole range of eminently useful first-hand applications--as well as full capability to further develop and refine these applications. The G15 CPU is made to serve humans, to serve the human mind--at all levels, also spiritually. And it is an expression of a philosophy of the infinite as constituting ingredience of the finite.

When we use bubblesort to sort a list, an array, let us remember that we have our hands on something that can be used to sort things with more dimensions as well. Put simply, you get a matrix by having several arrays after one another. Yet another dimension, by having several matrices after one another. Yet another dimension, by having several such 3-dimensional matrices after one another. The fifth dimension: several 4-dimensional matrices after one another. The sixth dimension: several

5-dimensional matrices after one another. Which again is one giant list.

Technically, one can argue the point that the universe is but one giant list, given a context where something of list is interpreted by something around the list as algorithms while something else around the list is alive.

In this philosophical perspective, one can, in a similar vein, argue that bubblesort is perception. Bubblesort is getting sorted out what you just saw.

Have you noticed how, after travelling much, fiesting much, sleeping not enough--some days later, having restored rhythm, restored sleep patterns, gotten into good habits again, got enough masturbation, exercise, baths, work, vitamins, ginseng, maca, whatever--the mind attains the glow of having 'sorted it all'? Sleep takes time; so does bubblesort /sometimes/. The glow of clarity means that you are able to make decision: you might say, the time between the observations and the insight involves having your subconscious mind 'work on it'. It delivers results. Effortlessly available, some days later, are insights into what you experienced some days before: which can lead to really important new developments, new action paths, and open up new avenues of intuition.

3.8

Concrete bubblesort in the next part of this chapter!

The next volume, volume 3 in this 5-volume series, G15 PMN Pattern Matching, takes the idea of a matrix /many lists of numbers beside one another, for instance holding line-by-line all info in a black'n'green photo of girls/ further in that we'd like two matrices to be compared in an approximate way that allows for shifting much around in one matrix without loosing the comparison with the

similar patterns in another. This is the sort of stuff a robot needs to avoid collision into a table if it is to pick up a plate to carry it away and wash it. This sort of thing is part of the field we call First-Hand Computerized Mentality, or FCM, and it builds on the core FCM modules already inside the Third Foundation you may know fairly well already.

FCM uses sort, usually bubblesort, all the time, at least potentially. The bubblesort it typically uses is the one inbuilt into the TF. In the next volume, we'll build a rock-solid way for the PC to pick out some general patterns out of a full-fledged photo of the type a camera can provide and which is stored in the G15 PMN Gem format, accessible through the GEM Image editor that is part of the core set of G15 PMN applications at the G:15 Home Screen. The app in its full form, with all the completed elements, working--and tested--even if they would be in need of a bit of optimalization here and there to be used in practise, inside a robot--is actually built while the volume 3 is being written: that is my plan, while now unfolding the volume 2. The point is not to include everything in the app inside the book--it's too much code to be pleasantly shown inside a book of such a side without squeezing away very many interesting discussions along the lines we have begun in this Art of Thinking series--but it is to ensure that the text reflects the questions, feelings, musings, alternatives that actually goes into making a really fairly huge app, instead of being merely a commentary around something that is listed on the G15 PMN app pages on the net long after it is made /when perhaps some of the original ideas are forgotten/.

Pattern matching is a valuable programming task also to call on further mediation on the nature of perception in ourselves. The universe may be a giant list of numbers in the sense of a good deal of dimensions and a very vast structure, and some of it may have an algorithmic interpretation, but there is also life and its living

attention, mindfulness, feeling in what is beyond those numbers, and that means beings, muses and God. Ultimately, an algorithm, no matter how cleverly made, can never pick out all that should be picked out of a situation, because that would involve an infinity beyond the principal capacity of any machine. But the life around those numbers infuse the universe with something that can arrange things so that what should be seen, will be seen. It is in this light that this writer uses the word 'synchronicity'.

Pattern matching on a computer can therefore be seen as an exploration in the "brain" /or "matter"/ side of living pattern recognition, whereas studies of conceptual infinity in philosophy can be seen as an exploration of the "mind" side of living pattern recognition. This need not be any 'duality': the brain/mind distinction, which parallels the pattern matching algorithm/pattern recognition distinction, may go on and on at finer and finer levels, including yet more subtle computers all the way through the levels where the muses exist, all the way to God's own level, who is the ultimate source of all living attention /in one essay I called God the giant Uncomputer/.

As I believe I have suggested earlier in this volume, it is a pleasant thing for an infinite being to visualize a computer and infuse it with a capacity to run a variety of programs. The instant it is imagined it exists, borne of the fluid essence of God in his untouched ineffable essence. The program is conjured up, and God visualizes his visualized program move towards the visualized PC and the spectacle unfolds on the display. To keep things simple, he uses such as a ten-number system, and numbers smaller than that, which means a 32-bit computer.

He plays with it and he makes some swirling galaxy like graphical images. It appears to him that if the galaxy is actually going to have anything in it, and not merely be a nice picture, he must either go into boringly long numbers, or he must have many computers. That seems a more

easy solution: he imagines the many computers, and imagines that they are connected by a nice bright green lucid wire and have some controlling computers with a controlling set of programs to handle their communication.

Good, but as he zooms into the galaxy even this series of computers don't provide much of detail as to what is inside a solar system--it is but a dot in the distance.

On reflection, he visualizes several levels of computers --hierarchically collaborating--and with hierarchically organizing programs given to some of the computers--and suddenly he finds that he can do a lot more faster, and even 'land' on a planet and, by some more visualization of yet more computers and suitable fractal-like programs, get along to see some plant-like stuff on one of them.

Then, we can surmise, he dozes off, happy after this first little experiment.

It appears to him he should have a staff to do this. How many levels are enough levels? Who knows? If he is to have any fun, and fun is what he wants, he needs some muses.

Bringing back the first visualization, he gives attention to the first being, the first muse. He tries out this and that, having her visualized on the most beautiful spot on the most beautiful planet in his first experiment, keeping his hierarchy of computers summing.

He is trying to find a shape of her, his first muse, Athina, and imagines a female counterpart of how he has often liked to think of himself, with two legs and ten fingers and so on, and long thighs. He want her to be like a slender little thing he can take in his arms. So he makes a version of himself as well on that planet. The Greek myths has it that Zeus created his beings out of his thighs, and whether God actually visualized that he sort shook these computers out of his visualized thighs only God knows. In modern English, the word 'muse' fits better than the often-used word 'goddess' for the females he creates--and those who use the word 'goddess' give a much smaller role to the word 'muse'; beside the have many

other often contradictory stories about these beings and their actions and emotions and genders. I do not pretend to render them at all. This is all fresh from self.

Anyway, he takes pains not merely to get the shape right, but to get the movements fascinating. He moves around on that planet in a way that exudes some strength and some degree of robust energy and tallness. Athina is a graceful elegant dancing being but who can suddenly stand firm and insist on something, and look up with her young fresh perfect face into his own more complex, older face, with sweet demand.

Again, we might think, at this point he dozes off and ponders on his next moves.

For sure, he senses excitement at Athina, he has never visualized anything that compelling to him after he discovered that he liked to visualize his own form. And she is intoxicatingly attractive. For several weeks, he finds that he want nothing but to walk on the planet and explore its nooks and trees and light little fires and eat little meals--a novel invention--with Athina, and her her opinions.

He explains to her--he has got around any initial objection he might have had to avoid infusing her with something of his real mind--a bit of tricky visualization, in which something of the computer has to suddenly yield to his attention rather than continue its program--and yet she is registering, through the computers, the very very very many computers at many levels giving her a mentality, sort of in a FCM sense, but vaster--what goes through her, also algorithmically. He finds that he is, in a way, making a female version of himself, but assisted by a structure that comes through a lot of fascination with permutation of numbers in an algorithmic fashion.

By the way, the word 'algorithm' refers to the name of an Arab who worked with procedural handling of numbers, and whose name shared something of the first letters in that word--and his name again begun with 'Al', which in

the Arab language of course refers to God. It sort of ties in with the myth we are weaving right now.

He finds that Athina gives him something new, a sense of mirror that is much more than a mirror and something which is in a way, if not better than himself, something he, as a result of giving himself a more energy-oriented, more robust, more rugged, more complex form, yearn towards seeing: it complements the view he has of himself, and since he has a preferred view of himself, he develops a yearning, a love, for Athina, who drinks in the power he bestows on her, as the many levels of computers also keep gathering in quantity and in elegance in their programs, to accomodate all the experiences of love he wishes to have between himself, on that planet, and on other planets --they build a ship--and Athina. All the time he is watching it from the outside, but he feels that he can do the same with the version of himself inside the visualization as he can with Athina: namely to infuse himself into himself, as it were, and let the computers help the registration of all that goes through the mentality.

In that way, he is both inside his creation and he is also watching the creation, being the ultimate sustainer of all. And the love becomes sex, and Athina suggests to him: let us give birth to new beings. He objects that it becomes too complex, but she has a way. In the ship he has already put a computer--visualized inside the computers--in other words a sort of 'virtual' computer. And, at Athina's sweet bidding, with her pouting lips, standing on tip-toe to kiss her man, God, he has explained as much as he can find words for as to how he has created her.

She suggests, do it over again, but do it inside this creation.

God, feeling impressed by this, is yet uncertain. Why, he asks. Yet another set of planets?

No, she says. So that we can see ourselves in those new levels and together with her sisters, Lisa and Helena. We

can lean back, Athina explains God, hand-in-hand, and watch overselves play along with these two lovely beings. And, Athina says, these are going to look like This! And like This! /For she has at this time also become a fabulous artist./

And that convinces--Zeus, to use that word. She is right. It is not just her and him. These two beings really are--while fascinatingly similar to her--having unique different types of beauties, and he would love to see them all three, and he loves the names she gives them. Lisa, Athina and Helena are soon walking hand-in-hand with Zeus while Zeus and Athina are watching this on a higher level with Zeus watching this on a still higher level. The 'levels' are, to use to the modern jargon, the 'virtual computers'. But the visualization is not merely doing this algorithmically--in contrast to the usual way the word 'virtual computer' is used. Instead, Zeus uses his real mind power--and Athina is very concerned that this is exactly what he must do, and makes sure he does it well--to infuse the lower-level Athina and Zeus with actual non-algorithmic mind power.

And this also goes for Lisa and Helena. The three muses, L., A. and H., hand in hand with Zeus, explores a myriad worlds--and have a myriad forms of orgies. Together,--it is an almost obvious next step--they create their next level, put themselves there, and in that next level they have the trillions of trillions of beautiful children of all their orgies, the submuses.

Only at this point it is possible to being to speak of levels approaching the level of the manifest universe--by imaginging a suitable number of additional virtual computers, infused along the way with real God mind-power.

At the higher levels, things get interesting when it is clear that more than watching is possible. Of course it is possible to make some nuanced changes in the structure of the lower level. And of course this is extremely entertainining. A lot more about this is developed,

including a limited but real way for the trillions of trillions of muses to move 'beyond' levels, and go back. There is much more to all this, but that is another set of tales; this volume is on Art of Thinking in general.

Let me re-iterate, for those who use the term 'virtual computer' in a 20th century style, that the myth we here weaved uses it in a metaphorical sense only: not in a strict sense at all. This is a computer that is of a different type, weaved of a different type of energy, and which--unlike the classical definition in 20th century terms of a computer--is fluidly response to moment-by-moment impulses from a living source of attention in a nonmechanical way.

This is further explored in my supermodel theory.

* A quick footnote for this chapter is in order:

Myth first set forth in the 2006 Firth platform by the same in a scifi tantric manuscript, a very early form of G15 PMN; and consolidated to a neopopperian theory with formalism equipped in the Supermodel theory, documented inside the G15 PMN Third Foundation app, and in a number of earlier writings by same author in the years in between; they have learned much from repeated conversations with David Bohm, first three times in Birkbeck College, London, then later on in Oslo when a friend of the writer housed Sarel and David /and dialogue companions/ for a Soria Moria dialogue arranged by S.Bjork & N.MacLaren under their brand "Forum2000"; a year or two later, this writer with his friend /Henrik/ made, for 3.5 years, a successful academic thinking magazine in Norway; which was closed some years later, after this writer begun the development of the computer and writing projects in New York and other places./

3.9

Let's do the bubblesort. We need first of all a place to put some numbers and in a programming language, that means to define something with a suitable short neutral-

sounding meaningful name that isn't already used in the language; which is to say, a name which isn't already used in the G15 PMN Third Foundation. Perhaps list1 will do. Is list1 defined? We type in ^list1 /// exists and it tells us that it isn't existing and that you are free to make it.

How long should the list be? Okay, up to 10 numbers this time, but another time much bigger. That can be changed later. 10 it shall be.

In the Third Foundation app, app# 3,333,333, which we always assume as foundation for any G15 PMN program unless we specify a particular variation or extension of it, the card F:1498 shows the typical way of making any list. A number is given to SZ, the SiZe, and this is the size of the list plus some extra space for the program to organize itself around it. This is stored in a variable. Find a card, eg on the i: or j: disk, and type in this:

list1= /// ^. /// 20 /// sz /// && /// list1 /// kl

Whenever we want to use this list, write list1 /// lk and we are going to use it with AY and YA to look up numbers in it, and to store numbers in it.

Note that the dot looks really huge when written in the proper programming font for G15 PMN. It is easy to remember to put it in when you use that font, which was made at the same time as the CPU and the programming language was fully getting in shape.

To get anything out of the list, we can write stuff like 3 /// list1 /// LK /// AY. That gives us the number in the 3rd pos. To put in 5 to this place, we would write something like 5 /// 3 /// list1 /// LK /// YA.

If we want to shorten the phrase, we can do list1 /// LK and put it to a local variable like i9, by command S9.

After that, we can write just 5 /// 3 /// i9 /// YA.

What should we do first? It appears to me a grand idea to have a function that does the swapping of two positions when we require it. At the moment I don't recall whether there is an inbuilt function of this sort, whether in the

core G15 or in the G15 PMN TF set. But this is about thinking, not sparing in a few lines of typing or making sure it works a second faster for a giant list. So we are going to make it:

swapthese= ///

Okay, just a moment. What info do we need as input to this function? It exists 'in empty space', and it is ready, as a tool in our toolbox, to do a job for us. But it needs to have some info about what it is supposed to do. We build it so it automatically works on our list1, I assume. But it certainly needs info about which two items to swap. Right? Let's put these to i1 and i2; and to Set i1 we use S1 and to Set i2 we use S2. I typically write them in the opposite sequence, because the stack works on the LIFO principle, First In, Last Out--the top of the stack is popped first out, and so goes to i2, then comes i1:

swapthese= /// s2 /// s1 /// list1 /// s9

Very well. What then? We would like to fetch both of these values, leave them on the stack, and put them back in the opposite arrangement. I imagine it is about this:

swapthese= /// s2 /// s1 /// list1 /// s9
i1 /// i9 /// ay ///
i2 /// i9 /// ay ///
i1 /// i9 /// ya /// i2 /// i9 /// ya.

Seems good to me. The PC will tell us whether we do it right or not. I will of course have checked with my own PC right now before I finalize this manuscript, but it is instructive to put it in, also because the very process of typing it in and seeing it take form on your own screen in a different font, and with much space, and use of the right-column /remember CTR-R to get there precisely when using the CAR editor/, means that you get much learning to the brain; and the art of thinking includes also fields like the art of learning and art of recalling.

Get therefore these things into eg the second card, right after the definition of list1.

I took a pause and, before I started writing again, had a look around in the region of card f:1111 in the TF. That area is good to keep a note of, and it is easy to remember too: f:1111. Very simple, right? Before and after that place a neat one-line summary of a huge bunch of two-letter commands are given. Very practical.

And, if you wish to increase the speed of our little program as we are now doing, there is also a two-letter function, IW, that does the same type of thing as we did before, with our SWAPTHESE. Anything done in G15 is done 'nearer' the electronics and so several times faster than when the PC does it through its PMN layer on top: but it is much more humane to do things in PMN so unless the loop is about millions of iterations, it usually doesn't matter.

By the way, what is the best way to actually record new data in the brain, and in the mind? How make sure it is getting into the brain in a way that can be recalled? The rules of thumb are,

* talk about it--even if only privately inside your thought--don't merely look at it; say, 'the f1111 is a good place with lots of overview, I will remember the f1111, it is the most obvious place for an overview', or something like that.

* visualize it. If you need a long-term visualization of a number, make a story of each digit or digit series, going from digit or digit series to an image and weave some kind of funny interaction between the images. Just what images is a question of 'getting acquainted with' the numbers and see what images feel right or deliberately construct ones according to some scheme that makes good sense /but stick to positive images; they can be sexual/

* don't talk about other things while you are trying to record one thing

* muse on it /rather than rush over it/

* do it again, several times, if you can

* do it in a pleasant, easy, light mood, if you can /alpha waves, right?/

Anyway, we're going on. Let's do the next card. In it we might have the very bubblesort loop itself. However it may be we want to divide it up later and in case we copy some cards 'to the right' and type in some new stuff here. Let us think. We want a loop that continues until something is done.

I got an idea, if you don't mind adding something to card above. This shows how programming works. Going back and forth. Why don't we have a neat little variable made before we have the SWAPTHESE function, a variable that goes about like this:

```
didaswap= /// ^.
```

This can be set to DANCE or BASIS, our pmn-like words for talking about 'yes' or 'no', or 1 or 0. What we want: The variable, when correctly used, can tell us whether a swap has been done, it answers the question, 'did a swap?'

We will set it to BASIS at the start of each loop, with the word BASISTHIS. Then we let our SWAPTHESE set it to DANCE, with DANCETHIS. Okay? In that way, when we call SWAPTHESE from the main loop, we also get the little marker we want to find out whether the sorting is complete or not. Right? So I copy the above definition and throw in something extra in it:

```
swapthese= /// s2 /// s1 /// list1 /// s9
i1 /// i9 /// ay ///
i2 /// i9 /// ay ///
i1 /// i9 /// ya /// i2 /// i9 /// ya
didaswap /// dancethis.
```

If you have time to have a look at the making of these PMN functions, you'll find them, in TF, at F:1991, and onwards. You can also scan for them, and if the word you put in is found other places also, type MMM to get to each next place and CAR after that /each time you type CAR

inside TF, it expects you to press either the letter Q or the space bar to signal whether to quit or go on viewing/.

By the way, the /// means it has got to be on a new line, whereas when I shift line in the above, that's just a matter of how I conveniently write it within this text. Also, when I put in a lineshift after a word or number in this text, whether or not I write /// at the end of that line, it must be a lineshift also in the program card. And the fact that we have that many ///'s now means that the swapthese= definition goes over no less than two cards. Format it the way you like, you have two rows of eight lines on each card and it should look good to you. A function can in principle go over even as many as dozens of cards /though for reasons of clarity, the usual approach is to keep the quantity of cards in each function rather small and rather have more functions/.

When we are going to check whether it is 1 or 0 we could eg call on the function 'SE', which does next line /or, if that line is blank, next line that isn't blank, whether in this or in the next card--though for clarity, usually best to have it immediately underneath the SE/--SE does next line when and only when the input to SE is DANCE. To get the value out of the variable, we do this:

```
  didaswap /// lk
```

In order to use it to exit the loop, we could do something like this:

```
  didaswap /// lk /// se /// q1
```

The Q1 in this case requires a bit explanation if you haven't met it before in this situation--inside a loop. Let me sketch how the loop roughly goes:

```
  ourbsort= /// ll:1 ///
  |do something
  didaswap /// lk /// se /// q1 /// lo.
```

Now the LL:1 sets up a loop to loop just 1 time. But the Q1 is funny because it reduces the value of the loop

counter by 1, so it gets zero again, and the loop keeps going. Let us muse over why.

Q1, like S1, changes the main loop counter, i1 /I write the 'i' letter small to not confuse with lowercsase L right now/. We can print ten numbers out like this:

```
anexample= /// ll:10 /// i1 /// nn /// lo.
```

So you see, the LL naturally affects i1. The loop stops after ten numbers have been shown on the screen. LO is the lower part of the loop, from there it jumps up again--unless the counter has reached its top value. In that example, the top value is 10. If you write LL:1 instead of LL:10 it will just loop once.

But if you put in Q1, it may loop a lot lot more. So we have SE in front of Q1, meaning that have a loop until a condition is 'satisifed'. Got it? A tricky little thing to explain in words, but it is just a number play, and if you find time to make many loops and try out various ways--and there are many!--to modify the index counter, you'll learn by trying out variations. And in some cases, of course, you must reboot the computer: such as if the loop got too big. Rebooting the PC is every programmer's dignified pride. Just don't aim at it, because it is a lot more time-consuming that exiting a program normally and restarting it :)

We seem to be getting somewhere with the program. Let us look back to where it says, as a comment, |do something /more here/. That comment is merely meant to say to us, we are doing the program partially, the rest is 'program description', the intended program. It is not necessary to type in such comments. The comments we should leave inside the program should be extremely to the point and not make a song out of it, because it can be very distracting, when you are trying to correct a program with a subtle confusion in it, if the code has to compete with comments that are pompeous, dramatic or some kind of flawless poetry or fascinating like a detective novel. As

a rule, a comment should be there only to clear up the most significant of confusions for the expert programmer: all else should be readable from the code itself, and that includes the names of the functions and variables. Yet sometimes these names are misleading, in that their actual work inside the program are different than the name given by the programmer, and that is a sort of thing that can delay the spotting of what to fix when you are in program correction mode.

G15 PMN programs are often quick to fix also because the presence of the i1, i2, i3, up to i9 and ix, and the j1, j2, j3, up to j9 and jx, are intensely used inside many functions and these have neutral names. Another reason is more philosophical: G15 PMN doesn't do things as much in terms of 'cause-effect' words as it could. Rather, it suggests looking at a program as a pattern /indeed, one of the early intended meanings with the three letters p, m and n, involved pattern and matching and networks/.

Well, let's get it done, this bubblesort of ours.

What seems to me to be called for is that we get into the main loop something roughly like this:

... list1 /// lk /// ay ... list1 /// lk /// ay /// ...
... se /// ... swapthese

Except for one thing: swapthese must get some input, namely two numbers, and so we need something like D3 in front of it. /As G15 PMN program texts tell, any statement where you use SE can be converted into a statement where you use D2, D3 and the like, if you 'negate' the input. If you use 'greater than' as input to SE, use 'less than or equal' as input to eg D3. If you use 'greater than or equal' as input to SE, use 'less than' as input to D3. The conversion goes like this: GE ⟨=⟩ LT, GT ⟨=⟩ LE./

What we want, in other words, is to pick one element, an the next, and we want to compare these, and SEe the result of that comparison. In the case that these are out of order, get them into order, by swapping these. Right? Hm.

The three dots in our case reflect our thinking. If you have a fast hand with a keyboard, you're in luck if you are going to make a program: talk to yourself, while writing it out like it did, ask questions and read your own questions and ask more or answer them. Have a dialogue with yourself. That is touching on the highest form of the art of thinking.

The ten positions are given by i1. Let us now muse over whether we got the limits right. These things are sometimes creating issues when we overlook them. When you make a list of ten numbers, are you going to put them in pos#1 to pos#10, or--as an alternative--are you going to put them in pos#0 to pos#9? Remember we set aside 10 positions in the earlier card, where we used SZ, in setting up the LIST1. A good rule of thumb is to give it some more extra positions than we need. In that way, we don't have to worry whether it is #0 to #9 or #1 to #11.

Nevertheless, let us use position #0 to #9, and keep that card unchanged. That means that we compare i1 minus 1, and i1, as far as positions goes /for i1 starts at 1/. There is a two-letter word that gives us i1 minus 1 and it is M1. And the two-letter word that gives us i1 plus 1 is F1.

Let us again put LIST1 /// AY into i9.

Given that, we have something like M1 /// i9 /// AY, followed by i1 /// i9 /// AY.

Right? And the former is supposed to be smallest. If it is equal it is okay enough. It is bigger we are supposed to get on to do the SWAPTHESE. The word to figure out whether it is bigger is GT. We would have something like this, LT /// D3 /// M1 /// i1 /// SWAPTHESE, since we convert GT (=) LT when we go from SE to D3.

To say something that is of value to record and keep in mind during G15 PMN programming: i1 in one function belongs to that function. i1 in another function belongs to that function. These two variables don't overlap across functions. They are, to use a phrase for it, 'local' to

each function. If you want a variable to be shared across functions, do as we did with LIST1 and DIDASWAP: give them names of three letters or more, new names, names that you check, carefully, that haven't been used in TF already. /That's a rule of thumb; there are always more ways of doing anything, including how to get functions to share variables./

The beauty of local variables like i1 and j1 is that each function can patiently spend time on getting that handling of those local variables right without you as the programmer handling them having to watch all how all other functions do their i1 and j1.

However, when we want these suddenly shared a bit, we need to put them on the stack. So we are going to put them on the stack: in this case, it is i1 minus 1, and i1 itself. That's why we write M1 /// i1 /// SWAPTHESE.

I now see what I could have seen at once, namely that as we are in this case always swapping two beside one another, we could have just given the first one and let the function add up one itself. Yet sometimes there is a sense of soul in keeping things the way they were first thought even if it is minutely less efficient. That would be measurable perhaps in a scale like microseconds in this case, when it comes to how fast the PC sorts our lists.

We are going to have one more part of this chapter, after we next give our bubblesort in finished form, and that is a routine to conjure up ten numbers freely, show them, push them through the new bubble sort we have made, and show the finished list; and in a way that allows us to repeat it easily. It will be a way to check the program, and it is also a bit instructive in itself.

The way I myself went from the lines above to make the full program next was to gather the lines above and have a cool look at them. I began thinking about how it ever gets to count up to compare all the numbers--it should could up to 9 or something to compare all the numbers.

This is a call for what is named "nested loops"--a loop can "nestle" inside another loop. In such a case, we use i2 rather than i1, and m2 rather than m1. For instance, if we wish the PC to print out the numbers one to five three times, we can do something like this:

```
threeloops= ///
ll:3 ///
  ll:5 ///
  i2 /// nn ///
  lo ///
lo.
```

I added here a lot of lineshifts and indents for dramatic effect. With each LL there must be an LO, okay? Otherwise the function won't even be defined, but the PC will tell, 'hey, have a look at this'.

I guess we got it now /I do this quickly so the lineshifts will be a bit messy but it should make sense; after all, this is a way of writing that is supposed to work well in prose while not being the entire strict way to list program cards as we can also do, eg in next chapter/.

```
list1= /// ^. /// 20 /// sz /// && /// list1 /// kl

didaswap= /// ^.

swapthese= /// s2 /// s1 /// list1 /// s9
i1 /// i9 /// ay ///
i2 /// i9 /// ay ///
i1 /// i9 /// ya /// i2 /// i9 /// ya
didaswap /// dancethis.

ourbsort= /// list1 /// lk /// s9 ///
ll:1 /// didaswap /// basisthis ///
ll:9 /// m2 /// i9 /// ay ///
i2 /// i9 /// ay ///
lt /// d3 /// m1 /// i1 /// swapthese ///
lo /// didaswap /// lk /// se /// q1 /// lo.
```

The PC will attempt to do exactly what you write. If the 'lk' is ommitted after 'list1', it will do something that may cause it to reboot. Smile when it does it, and fix the program. Don't aim at getting it right the first time. Even if you have programmed for a thousand hours in previous years, a simple program can baffle you; let it!

A beauty of the programming process is that the PC does not even begin to try to guess on our intentions, and that is a tremendous sharpening of our own minds: you will know that if it doesn't act as intended, it is because our intentions were not exactly, completely enough expressed on its terms--nothing else. It wasn't because you smiled too little: it wasn't because you called the PC by a bad name the other day: it was because you just needed to give the program a little more thought. So the PC is a fabulous instrument, when the programming language is relentlessly demanding on your mind and never swooshy or corrupt or cloudy about it--to cleanse your own mind. You get into a meditative state because this is a sort of impersonal beauty. It is a blank canvas. But a very particular kind of canvas, the algorithmic sort.

By the way, if you should add any comment to the above list, it ought to point out that the positions are 0 to 9 in the list. To get a comment right, it must be no wider than the program words can be in a column, and begin with the commentary sign, the vertical bar, '|'.

Eg, |Pos: 0-)9

Also, be sure a comment doesn't stop with any highly significant special sign like a dot or a & or a ^--ideally it should not even contain any of the most significant signs. In that way, the comment stands out as human-only readable; and you are sure it is not triggering a process in the PC to try to parse the letters as a program bit.

* Quick Footnote:
The PMN you type in, how does the PC handle it? Where is it checked, processed--"parsed" as we say? Answer is that that is code in G15 which is in the first few hundred cards in F when you start the TF. And whatever higher-level PMN definitions that are between those cards and your program, higher up in TF.

Well, how does the PC parse the G15 code from those cards?

Answer, that's a bunch of numbers it loads from disk during boot-up, written and put in as number-data by this author some time earlier.

This bunch of numbers tells the G15 CPU how to convert the human-readable G15 text code into its own proper instruction numbers. It's a fairly simple bit of code.

Granted, but how come the PC looks up anything at the disk during start-up? Where's the program to do that?

That's part of the G15 PMN PC hardware electronics: it is hard-wired into it, that the PC should fetch a bunch of cards from the socalled A: and B: disks. These disks are only used for such startup-purposes. The disks C: and all the way up to L: are for programs and data, incl.apps.

3.10

Some programs are hard to check well, and yet definitely require good checking. I once typed in a supposedly fast sort algorithm from a big book about algorithms. The programm sorted both 10 and 100 numbers well. But when I checked it for 10,000 numbers, it had odd mistakes in it. It was like right for 9,950 of these numbers. I re-checked my typing, but I had typed in correctly. It was a complicated little thing to correct, which the authors of the book had overlooked. This, however, illustrates something that programmers often encounter, and which is part of the art of thinking: that there is a huge difference between 'fairly' and 'completely' when it comes to the question of correct programs. And only programs that are beautiful and elegant are worthy of being called "first-hand" programs, and deserving of going through that extra test which is your own artistic judgement of it.

I believe the experience of the big distinction between programs that are correct, and programs that are nearly correct, gives us a kind of light that shines in on our daily life language as well. A programmer is more careful to use words of a statistical or 'probabalistic' nature, such as, "It's probable, given such and such, that such and such is the case,", or, "With likelihood, we'll see the same pattern tomorrow." And only someone who has a good relationship to probabilities can find out how to work with logic and intuition hand in hand.

Moreoever, in programming through levels of algorithms, set up in a network or matrix, and in which programs are engaging in such as pattern matching from input from camera, the 'probababalistic' type of language gets even into the program. For instance, when is a squarish area on the screen, which comes from a camera filming what's in front of a robot in search for a box, evidence that a box has been found? The squerish area may turn out to be a

a piece of paper with a drawing of a square on it, rather than a three-dimensional box object. The robot may have to move a little bit to gather up "more probabilities".

The FCM programming framework as included in G15 PMN TF has been very well tested and found to be good; it is also excitingly simple, compared to the extremely wide range of possible applications for it. In the next and completing chapter of this book, we list all the cards, less than 200 cards, which comprise the FCM framework. In the next volume, we put it to a first use as far as this book is concerned--and go through parts of an app that was written especially so as to get some work done in the area of pattern matching in this book; an app that, after the publication of Volume 3, is one of the core apps always available with G15 PMN.

We'll soon check our bubble program, further on in this chapter. But first some more notes on program correction.

As said already:

A program that is only fairly correct is having a very different feel to it, relative to a program that is completely correct. A program that is to be used only once and then dropped can be fairly correct and it would be a meaningless form of perfectionism to get it completely correct. But a program that is meant to illustrate the art of thinking, or be a crystal of thought that a teacher presents to pupils for them to meditate on, or to be a function inside a larger program, a function that may be called on a vast number of times, ought to be completely correct.

But how do we know, for sure, that a program is fully correct? The reality is that absolute sureness isn't something we have. We can regard it as likely, as probable that a program is entirely correct. We can increase that likelihood by pushing the program in testing, and by looking at it--and especially if the program is easy to look at, we can arrive at a great deal of certainty.

What if the program doesn't work in one test? Is that a

sure signal that it isn't fully correct? Not entirely sure in that direction either: because it may be that the testing has an issue about it; or that the PC had one of those /very/ rare occurrencies in which it didn't do as instructed to do.

To be sure a program needs correction, if it isn't obvious from looking at the program, you would typically want to repeat a situation in which the program doesn't work out, and get repeated confirmations that something is amiss with it.

When a program is made out of beautiful, small functions --each checked and found to be pretty good, and even when pressed the program seems to be good, you are building a confidence that the program is absolutely correct. But it doesn't make full sense to say that you are absolutely sure that it is absolutely correct. You are somewhat sure that it is absolutely correct, which is a lot better than to be absolutely sure that it is absolutely correct.

The reason I spend some time on this point is that the art of thinking flows best in a person who have for some time cultivated a subtle precision in his or her language use relative to facts. It is about such things as getting used to say 'probable' and 'likely' when you wish to leave room for other interpretations of the events.

Imagine that some person describes the interaction between two other persons to you, in which one person comes in a bad light. Surely it is better if that description has phrases like, '..and it seemed to me that A tried to do such and such..', rather than phrases like, '..and A tried to do such and such.' Put in the extra care in normal daily life language use, and you are generous to the possibility that you misunderstood; and that other people have more information to bring on the matter.

In contrast, imagine someone who always uses words like, "I am absolutely sure that..", "..it is necessary that..", "..it must be so that..", and so on. Such a person comes across as dogmatic and manipulative, and probably existing

in a state of near-hysteria. If one has never met this person before, one will perhaps listen attentively. But eventually, once it dawns upon you that the person is sticking to using that sort of words most of the time, one automatically assumes that the person is inflating the words, and that one cannot really map reality correctly by listening any too carefully to this neurotic individual.

To lay the ground for handling complex social interactions--when thinking becomes dialogue, conversation and, in some cases, with some, perhaps also conflict--one need to not inflate words and not over-state certainties /nor hide them needlessly/. In nurturing a balanced relationship to facts through one's language, one will be equipped also to handle a situation in which two people's priorities seem to clash and variations of interpretations of facts--and the 'narratives' of facdt--may be components of the clash.

I am not saying that someone who is having a good relationship to facts, and who is used to using language carefully so as to seek out what is more certain and distinguish it from what is less certain, cannot ever have a conflict with someone else; and also, I am not saying that full honesty is always the right tool; just as I am not saying that passifism can always be the best way to meet something, someone who is violent. But someone who is concealing fact from herself or himself is more likely to be seriously confused in a conflict situation. If facts have to be concealed to solve a severe crisis and get it to become much less serious, so be it. If lying is the pathway to a temporary harmony that later on can become a more real harmony, lying is the ethical pathway to go. But someone who is lying inside thought cannot make clear decisions about this sort of thing. To live in a state of constant lying is a neurotic way of living in which the art of thinking is not flourishing. Someone who is doing the art of thinking can call on the twisted language use as a tool to achieve an effect, just as violence can in

some specific cases be the appropriate response: but this is very different from being a violent person in general.

The art of thinking requires a harmonious sensitivity that in general is most compatible with a tranquil way of living, in which loving-kindness and factualness are cultivated; in which language use is careful and as objective as can be; and in which silence is the preferred mode relative to some person's insistence on twisted expressions--unless there are larger priorities of upcoming collaboration that demands a union of understanding. If practical collaboration is required, some misunderstandings must be cleared up, even at the expense of loud-mouthed conflict. But the art of thinking will generally guide a person from moving out of situations in which loud-mouthed conflicts are the rule of the day. Of course, this depends on the person living in a society which has some promise of harmony in it, and is not entwined to living or working with people who are thoroughly disharmonious inside.

How do you nurture harmony, if your mind feels to be often at the breaking-point?

The most simple advice /which does not always work/ is to feed the mind more harmony--of all kinds--and let the body express something harmonious--of several kinds--in daily life. Add to this the advice to engage in longer walks, longer and better sleep, with harmonious music in the background perhaps, and good cleanliness and more beautiful clothes also perhaps. The yet deeper advice is also to pray for harmony, for it is a thing beyond the human level, something that must in a way be granted rather than 'attained to'; which means paying attention to spirituality rather than merely assuming the universe is a machine.

I assume you do not now feel at the breaking-point and that you rather want code to check the bubblesort from the previous part of this chapter.

We would want to have ten numbers in position #0 to

position #9 of LIST1. These ten can vary each time. We will make a routine to put ten 'free fluctuations' numbers into the list, show them, sort the list by our new routine or function, and show them again. By not too many clicks we want this to be repeated again and again. Let us see.

We have the two-letter command AF, "A Free fluctuation number", which produces a fairly free number each time, from 1 up to the number given, as long as the number is fairly small, under 30,000. In some G15 PMN PC's, it won't make higher numbers than 32,768, which is 2 times itself 15 times /2 to the power of 15/. The Third Foundation provides a stronger version of this, the RFFG function. This is a good name for this sort of numbers in any case--"Relatively Free Fluctuation Number"--for the numbers aren't really 'free' in any absolute sense. It is just a sort of messy enough arithmetic hoovering in the background to provide a rather strong variation of numbers; and at bootup, the G15 PC has a milliseconds clock that typically is slightly different, at least, each time and this number is starting off the RFFG numbers in a different way /that goes for AF as well/.

You can type like 1000000000 /// RFFG /// NN to check the idea /that one billion, ie, 1 with 9 0's/. It will give all sorts of different numbers, between 1 and a billion.

Got it?

Therefore:

```
varylist1= /// list1 /// lk /// s9 ///
ll:10 /// 1000000000 /// rffg ///
m1 /// i9 /// ya /// lo.
```

Alright?

And:

```
showlist1= /// list1 /// lk /// s9 ///
ll:10 /// m1 /// i9 /// ay ///
nn /// lo.
```

Bubble sort check function is therefore--and I give it an easy-to-type-in name, so we can type it quickly again and again:

```
  xxx= /// varylist1 /// showlist1 ///
  &******& pp /// ourbsort /// showlist1.
```

Done! As a variation, you can put in a screen cleanse, CE, at the beginning of the routine, if you like.

Did I try it myself? Yes. The code in this manuscript SEEMS to me to be ABSOLUTELY correct ;/

SPACE FOR YOUR OWN NOTES

SPACE FOR YOUR OWN NOTES

SPACE FOR YOUR OWN NOTES

CHAPTER 4

This is a complete listing of the 179 cards, starting at F:2231 in G15 PMN Third Foundation, that define the FCM core and a couple more things associated with that. This chapter is here so you this is also a handbook when you program using the FCM core in G15 PMN.

The FCM core is what we can call a 'framework'. It does not tell you whether it is for robotics or something else. It is a way to put in data and what we call 'warps' to functions in a series of levels, which are performed again and again; where it is not precluded that these levels change as a result of performing the functions in these levels. Each function may or may not use near its place in the level. Each function may or may not change other data in the network, in the levels; it may or may not list up other functions somewhere in the levels; or activate or passify other functions, or itself. All that is up to you.

What the FCM core does give is a set of suggested slots for putting in data and functions and a main routine for looping through them elegantly.

Philosophically, "FCM"==Fist-hand Computerized Mentality is a vast theme, and one which we will never finish discussing, whether in this book series or in any other book series in any era. Computationally, it is the intuition of this writer that it is hard to get anything more adaptable and fitting for any FCM approach than this.

This is also a very well-checked core. It is versionless in the sense that no new versions are planned.

There are comments between some of the cards. These comments are only there when it seems to be valuable to clarify some points, eg about word usage in the cards; in some cases, the comments we have inserted between the cards simply explain the terse mini-comment inside a card.

The robotics program elements discussed in here and

there in the upcoming volumes in this five-volume series will generally be easier to understand when looked at in connection to an overview like this.

Good luck with your FCM programming, always!

The journey begins at F:2231 in G15 PMN TF:

```
(f2231)
fundnet=        |a tight list
^.              |over all
|make 150*n     |foundries in
|matr by sz,&&  |use; your fcm
|this can have  |app gives ram
|unused parts   |to this; 1st
|of it, for     |x and 1st y
|fcmindex has   |are basis
```

The word 'fund', the two letters FN, and the longer word 'foundry', all refer to the same kind of thing in FCM: namely an array possibly having data, possibly having warps, being organized in a certain way as depicted next.

```
(f2232)
|fnpos#0:level |#47 qty links
|#1-8:p4g4name |#48 marked as
|#9 has act#?  |highpriority?
|#10-39:3 & 3  |#49 active?
|n,act#,extnum |#50-149:up to
|ten triplets, |100 links to
|can be array  |funds, called
|#40-46:luxury |on by act's
```

The chief action of the FCM framework is to sort an index of all the foundries, or funds, or FNs, or nodes, or whatever we call it--and the whole thing can be called a matrix or we can use the fancy word 'network'--and go through all the foundries, again and again, in a loop until program signals that is done. To 'go through' is

done in the TRANSLUCENT loop, which starts at F:2309, and what it does is simple, but eminent: when both #9 and #49 are 1 rather than 0, it calls up to ten functions which are indexed in positions #11, #14, #17, #20, #23, #26, #29, #32, #35, and #38. These functions are called not directly by warps, but by socalled "action numbers", which is written "act#" in the comments, and which is simply a postion in a list /also called "fnacts"/ set up in the initalization of any FCM program.

As input to these foundry acts is an indication of which of the ten places it was called from. Whether or not the function makes use of this input is up to it. The rest is really all a matter of convenience--we name each three positions for a "triplet", and assume that the function may often make use of a number just before, and possibly also after, its own action number: but these are merely suggested hints. The programmer has full freedom in what this is all about. The TRANSLUCENT loop is as simple as can be. We do not give the programmer a universal kind of 'crack of the nut of Artificial Intelligence' because there is no such nut and therefore it cannot be cracked. We simply say, when you wish to program in a way that more expresses your own human mentality, in which there are patterns and probabilities and changes of setup during performance, then here's a framework that lends itself to such coding.

That's the background. Now, assuming you understand this at least in a general way--and for more concrete understanding, start with the TRANSLUCENT loop in F:2309, we begin wading through the terse comment in the card just listed. All this is entirely as in the Third Foundation.

The position 0 in the foundry array is used to store level number. This is of chief importance. It gives the sorting sequence for the main FCM loop which is defined in the main function FCM at F:2315, which calls its main

subfunction TRANSLUCENT at F:2309. It is sorted by the bubblesort BS function during start-up. It is assumed that it is kept in a sorted sequence and the FCM programmer must keep this in mind, and be sure that resorting takes place whenever required. The level numbers may be the same for a whole set of foundries. Each foundry can have functions associated with it. In case that these a foundry is marked as 'active', the functions will be performed. The sequence within a level of this performance should not matter. If it does matter, that's a calling for two different level numbers to be made. Any numbers can be used as level numbers, as long as they are valid numbers; and there usually are wide gaps in the use of level numbers. They should make sense to the programmer.

A foundry can have a name. This is packed into position 1 to 8. A packing means that four characters are put into a single big 32-bit number. There are functions that can quickly pack and unpack a name for the foundries.

In position 10 to 39 of a foundry we have what we call "triplets" in a foundry. Each triplet has one possible reference to a function, and two possible numbers as data for it. These are sometimes called the triplet's "value", the triplet's "action number", and the triplet's "extension value". Instead of storing warps directly to a function inside the foundries, there is a storing of such an "action number", which is a simply a number in a list kept separately. This means that it is easier to put and retrieve a FCM network /as we can call the whole thing/ to and from the disk /for warp numbers may change from time to time depending on how the PC is set up; whereas these action numbers provide a region of predictability/.

There are functions to set up the action numbers as part of the following cards: see for example fnactcherish and an example of its use after defintion at F:2400.

The TRANSLUCENT loop checks #9 and #49. Much of the node can be used more freer than the comment indicates: for instance, it is possible to set up a sort of array or

matrix spanning the triplets of several nodes /these must have a correctly set #9/.

The 'links' are typically used when a program is created to 'entrain' the FCM framework, such as to make a robot behave in a certain manner given a certain not entirely precise 'style' of input. A robot should not go on and on 'entraining' itself because then it may mess up the structure and run afoul; the most ethical way is to have a phase called 'entrainment' which is strictly controlled by the programmer or programmers, and to switch off entrainment--at least for all essential action variations --when it comes to the normal use of the robot.

The FCM network can be stored to disk and retrieved as long as thought goes into the foundry action indexing.

As part of entrainment program the 'high priority' flag can be given a role, such as to mark certain nodes as too important to be changed at all during the performance of the entrainment program. This can typically be considered to be the nodes setting the 'highest priorities' of the robot tasks.

```
(f2233)
thisfcmnet=     |sort-by-level
^.              |list fcmindex
|This is used   |--it can be
|by fcm main    |changed BY
|routine        |the net when
|translucent,   |done careful
|and it must    |& at start or
|match the      |compl of loop
```

This is the pointer to the whole FCM network which is being presently worked on by TRANSLUCENT. Typically, the level numbers are arranged in this way: the lowest level numbers have nodes that parse input from keyboard, from cameras, etc. As we progress higher up, this input has been structured into patterns.

As we go still higher up, there are task priorities, in case of an FCM for robots for instance. This suggests, in terms of the patterns that the FCM is capable of handling, what the ideal states are, and what general sorts of actions are to be taken when there is a difference with the ideal states.

Beyond this 'middle' or /if we imagine a bowing of the FCM network/ 'top' of the network, we will typically have patterns of actions, with the highest level of these actions specifiied first. As we progress along the levels, the actions get more and more concrete, and are gradually translated into concrete electrical events such as of servos and other engines of robots.

This is a typical, and masterful, layout of the FCM.

There are additional considerations, and these also involve the fact that the FCM loop goes through all levels all the time. In other words, there is an activation, steered within the FCM, of its own nodes in a meaningful sequence. And this activation may be so that, in practise, there are no big tasks initialized, because much activity must be spend in fine-tuning the pattern matching. This fine-tuning can involve a variety of sometimes very complicated activities, such as using motors to move cameras, and to permute alternative ways of summing up the matched patterns in terms of different scenarioes. This is akin to how we may sometimes have to be active in finding the right interpretation of what we are seeing: mere passive seeing may not be enough.

To correctly structure FCM to handle such very sophisticated algorithmic processes require artful and skillful programming and much time and experimentation: but in a billion years, the FCM framework can still be perfectly well used, because it doesn't limit how this is done--at all!

```
(f2234)
fcmindex=       |is at pos 0;
^.              |a tight list;
fcmindqty=      |size ) 2;
^.              |used by sort,
|this has nums  |by levelnum;
|of all funds   |then used by
|in use in the  |translucent
|fcmnet; 1st    |main loop

(f2235)
fundlevel=      |This, with eg
^.              |thisfund,
fundlevel       |nextfund,used
basisthis       |during making
|We often use   |of fcm net
|10,20,30 & up  setfundlevel=
|in organising  fundlevel
|foundries !    kl.

(f2236)
nextfund=       thisfund=
^.              ^.

nextfund        thisfund
basisthis       basisthis
|read about     |foundries in
|the meaning    |tf docs via
|of the fcm     |b9edit, h33

(f2237)
fnwarp=         0
|In:number      w
|Gives: warp
|to start of
|the foundry
|of this num    thisfcmnet
```

```
|in              lk
|thisfcmnet      w9.

(f2238)
fnmainval=       10
|In:number       w
|Gives: the
|first value
|in triplet#1
|in foundry of thisfcmnet
|the number in lk
|thisfcmnet      ww.

(f2239)
setfnmainval=    10
|In:val,fnnum    w
|action: sets
 |first value
|in triplet#1
|in foundry of thisfcmnet
|the fnnum in    lk
|thisfcmnet      yy.

(f2240)
wtofnnum=        thisfcmnet
|In:warp to eg   lk
|foundrystart    su
|Gives:fundnum
|Assumes:that    150
|thisfcmnet is
|set;BE SURE:
|get adr right   di.

(f2241)
whereisfund=     thisfund
|In:position     lk
|0-)149
```

```
|Gives: warp
|to this pos
|in thisfund    thisfcmnet
                lk
                w9.

(f2242)
adjustfund=     thisfund
|In:newvalue,   lk
|position       |This is 'y',
|Action: sets   |while 'x' is
|'thisfund'     |position ;/
|at position    thisfcmnet
|to value;uses  lk
|'thisfcmnet'   yy.

(f2243)
cleansefund=    150
|Action: puts
|basis to the
|150 positions
|in present
|thisfund       0
|in             whereisfund
|thisfcmnet     clrthismany.

(f2244)
makefoundry=    nextfund
|Action:        lk
|Assuming that  thisfund
|fundlevel &    kl
|nextfund is
|right, sets
|thisfund and   nextfund
|inits fund     danceup
```

```
(f2245)
|Note that      fundlevel
|some vars      lk
|here are       |The level
|oriented       |is a sort
|towards        |item; the
|construction   |very first!
|of funds       0
cleansefund     adjustfund

(f2246)
1               |When the main
|Status:active  |fn loop runs,
|Note:in fn,    |translucent,
|triplets plus  |only fcmindex
|luxury can be  |& thisfcmnet
|array w/sz 37  |/but not eg
49              |thisfund/ is
adjustfund.     |assumed

(f2247)
fnamtx01=       &&
^.
                |used for
                |putfnam,
                |getfnam &
                |wgetfnam
35              fnamtx01
sz              kl

(f2248)
putfnam=        tx
|In:name        |Be sure this-
|Action:it      |fund is set!
|sets name of   jx
|thisfund but   lk
|exits if name  |also, no
```

```
|isn't of len  |spaces in it
|from 3 to 32! s5

(f2249)
i5             n?
3
32

               se


iswithin       ex

(f2250)
32             jx
               up
               fnamtx01
               lk
fnamtx01       up
lk
up             i5
clrthismany    fw

(f2251)
fnamtx01       8
lk
up


               |pack 8-bit to
1              |32-bit
whereisfund    p4.

(f2252)
getfnam=       sx
|In:foundrynum |Copy,eg by tt
|Gives:quote,  |to somewhere
```

```
|its name;      |if you want
|assumes that   |to keep it;
|thisfcmnet     |quotespace
|is set and     |is
|fund# exists   |fnamtx01

(f2253)
ix              32



fnwarp
up              fnamtx01
                lk
tx              setlenandnil

(f2254)
jx              8
                g4
                |unpack 32bit

fnamtx01        fnamtx01
lk              lk

up              clipniltrail.

(f2255)
wgetfnam=       |the start;
|In:fundwarp    |Copy,eg by tt
|Gives:quote,   |to somewhere
|its name;      |if you want
|uses           |to keep it;
|thisfcmnet;    |quotespace
|the fundwarp   |is
|must be at     |fnamtx01
```

```
(f2256)
up              32



                fnamtx01
                lk
tx              setlenandnil

(f2257)
jx              8
                g4


fnamtx01        fnamtx01
lk              lk

up              clipniltrail.

(f2258)
fnamw=          tx
|In:name        |be sure name
|Gives:warp     |has no extra
|/or 0/ to      |spaces cmprd
|fund w/name    |to original
|/sz 3-)32/;    jx
|uses nextfund  lk
|& thisfcmnet   s9

(f2259)
i9              an
3
32
iswithin        d2

nextfund
```

```
lk              basis
ye              ex

(f2260)
0               jx
fnwarp          |Note:nextfund
                |shows qty and
                |in this func,
nextfund        |flags aren't
lk              |parsed at all
dc              |Pd for funds:
fnwarp          nf.

(f2261)
fnam=           fnamw
|In:fundname    |Uses nextfund
|Gives:number   |& thisfcmnet;
|Action:gives   |in this func,
|number 0-)n    |flags for fn
|of foundry,    |aren't parsed
|or -1 /note!/
|if unfound     sx

(f2262)
|Note:when      d2
|you program    |For speed, at
|a net with     |runtime, fcm
|an arrayfund,  |should only
|fnam may make  |namefind when
|it easier! :)  |important
ix              oneminus
ye              ex
```

```
(f2263)
ix




wtofnnum.

(f2264)
fnloopcont=     |used by
^.              |translucent;
|'fn' is        |fnact27
|short for
|fund, ie, for  |This says:
|foundry        |is main fund
fnloopcont      |loop
dancethis       |continuing?

(f2265)
fnya=           |This uses
|In:num1, pos,  |thisfcmnet;
|fnnum          |this assumes
|Action:stores  |first pos in
|num1 in the    |array is 1;
|foundry at     |mimicks 'ya';
|position pos   |use fnyay for
|range: 1-)37   |range)37

(f2266)
sx              ix
|For speed,     thisfcmnet
|this assumes   lk
|that the       |The 37 are
|range 1-)37    |30 tripletnum
|is prechecked  |plus the 7
```

```
9                 |luxury nums
ad                yy.

(f2267)
fnay=             |see comments
|In:pos, fnnum    |at fnya; also
|Gives: number    |note that all
|Action:gets      |fnay,fnya,
|num1 in the      |fnaya,fnyay
|foundry at       |have first
|position pos     |arrayposition
|range: 1-)37     |as #1 /not 0/

(f2268)
sx                ix
                  thisfcmnet
                  lk



9
ad                ww.

(f2269)
fnyay=            |fnya but pos)
|In:num1, pos,    |37 fine: make
|fnnum-for-1st    |sure enough
|Action:stores    |funds /each
|num1 in the      |w/triplets &
|foundries at     |luxuries for
|pos 1-)n;        |this/ are IN
|this is as       |SEQUENCE
```

```
(f2270)
sx              i5
dc              37
s5              di
                ix
|We use num     ad
|sub one to     |New fnnum;be
|do modulus     |sure sequence
s3              sx

(f2271)
i3              ix
|Ten=first      thisfcmnet
10              lk
i5              |Note: for
37              |arrays 1-)37
mo              |size use
                |fnya & fnay
ad              yy.

(f2272)
fnaya=          |been set up
|In:pos /1-)n/  |so that all
|fnnum-for-1st  |ten triplets
|Gives:value    |plus luxury
|at this pos    |nums in them
|when funds w/  |shall serve
|SEQUENTIAL     |as one longer
|fnnums have    |than 37 array

(f2273)
sx              i5
dc              37
s5              di
|Note: for      ix
|compact        ad
|arrays 1-)37
```

```
|size use
|fnya & fnay    sx

(f2274)
10              ix
                thisfcmnet
                lk
i5
37
mo

ad              ww.

(f2275)
fnyy=           |the first
|In:value,      |value of the
|xpos/1-)n/,    |first triplet
|ypos/1-)n/,    |must be set
|fundnum        |to x-width of
|Action:as      |the matrix;
|fnya but as a  |when)36 items
|matrix; note:  |use fnyyx

(f2276)
tx              10
s9              jx
sx              thisfcmnet
                lk


                w9
s5              t1
```

```
(f2277)
i5              j1

i9              ad
dc

j1              ix
lk
mm              kw.

(f2278)
fnww=           |the first
|In:            |value of the
|xpos/1-)n/,    |first triplet
|ypos/1-)n/,    |is assumed to
|fundnum        |be x-width of
|gives: value   |the matrix;
|As fnay but    |when)36 items
|matrix; note:  |use fnwwx

(f2279)
tx              10
s9              jx
sx              thisfcmnet
                lk


                w9
                t1

(f2280)
                j1

i9              ad
dc

j1              ix
```

```
lk
mm              wk.

(f2281)
fnyyx=          |mainval 1st
|In:value,      |fn is xwidth;
|xpos/1-)n1/,   |37 nums pr fn
|ypos/1-)n2/,   |in sequence
|fundnum        |but only 36
|Action:as      |nums in 1st;
|fnyay but as   |when(37 items
|matrix; note:  |use fnyy

(f2282)
tx              10
                jx
dc              thisfcmnet
s9              lk


sx
|stk:value      ww

(f2283)
|Stk:value,     ad
|x-width        up


i9
mm
                jx
ix              fnyay.
```

```
(f2284)
fnwwx=          |mainval 1st
|In:            |fn is xwidth;
|xpos/1-)n1/,   |37 nums pr fn
|ypos/1-)n2/,   |in sequence
|fundnum        |but only 36
|Gives:value    |nums in 1st;
|as fnaya but   |when(37 items
|matrix; note:  |use fnyy

(f2285)
tx              10
                jx
dc              thisfcmnet
s9              lk



sx              ww

(f2286)
|Stk:x-width    ad
                up


i9
mm
                jx
ix              fnaya.

(f2287)
fnactlist=      fnactlist
^.              kl
|Foundry        |most defined
|actions        |in fcm apps
|1-)5000        5050
5100            fnactlist
```

```
sz              lk
&&              clrthismany
```

Here you see that Third Foundation G15 PMN with its FCM framework suggests a normal maximum number of foundry action numbers. You can use pretty freely within this range. The way it is defined allows, though, as most things about G15 PMN FCM, later re-definitions to be possible without actually having to go in and change anything in the stable and beautiful G15 PMN TF core.

```
(f2288)
fnarrayup=      ix
|in:pos,fn#     thisfcmnet
|/fn of type    lk
|fnay & fnya/   w9
|Incs by 1
sx
9
ad              danceup.

(f2289)
fneasy=         |this assumes
|In:mainvalue,  |vars fcmindex
|foundryname    |thisfcmnet,
|Action:        |nextfund;uses
|easy way to    |thisfund; the
|make funds     |'mainvalue'
|for a new fcm  |is 1st num in
|application;   |triplet#1 :)
```

```
(f2290)
tx              0
sx              whereisfund
makefoundry
                fcmindqty
                lk
ix              fcmindex
10              lk
adjustfund      ya

(f2291)
fcmindqty
danceup




jx
putfnam.

(f2292)
fneasyact=      |flag about
|In:a b c       |actions
|name           |set to
|Action:        |dance   :)
|as fneasy,     tx
|but with       t3
|full triplet   t2
|#1, and        t1

(f2293)
j1              j3
jx              12
fneasy          adjustfund
|j2 has num
|in actlist
j2              dance
```

```
11              9
adjustfund      adjustfund.

(f2294)
fnactcherish=   jx
|In:name,num    ff
|Action:stores  |This is to
|warp in array  |be used while
|Name is of a   |compiling in
|new function   |the fnacts
sx              |only
tx              s5
```

While you are compiling the fn acts, you typically put in the name and number of each just-defined function and give to this function, 'fnactcherish'. It will 'cherish' the function in the sense of getting it listed as a proper fnact. See the K-disk for a wealth of both simple and highly sophsticated examples, also showing how the 'wave/particular' duality is solved in Super-Model Theory, which sets forth a scientific /ie, neo-popperian scientific/ formulation inspired by the myth-like thinking we indulged in, in earlier in this volume and elsewhere. All this is documented well in the Third Foundation G15 PMN, which reached its completion in 2016 and 2017.

```
(f2295)
i5              cl
ye              jx
                pp
                ix
                nn
                ki
                sh
d8              ex
```

```
⟨f2296⟩
|Note: when    i5
|name of new
|func isn't    ix
|found it
|will show the
|num and name  fnactlist
|and wait for  lk
|keypress :⟩   ya.

⟨f2297⟩
fnact27=       |translucent
|In:triplet#,  |loop will
|foundrywarp   |/for now/
|Action:       |exit
|sets the      sh
|fnloopcont    sh
|to basis,     fnloopcont
|so the        basisthis.

⟨f2298⟩
|This, then,   &fnact27&
|is an         27
|example as
|to how your
|fcm app can
|set up new
|triplet
|foundry acts  fnactcherish

⟨f2299⟩
fnact251=      sh
|In:triplet#,
|foundrywarp
|Action:
|main value of
|1st foundry
```

```
|is decreased
|by 1 /is #0/  sh

(f2300)
0
fnmainval



dc
0
setfnmainval.

(f2301)
               &fnact251&
               251




               fnactcherish

(f2302)
fnact271=      |basis or
|In:triplet#,  |signed; 1st
|foundrywarp   |foundry is
|Action:       |#0; the main
|exits like    |value is the
|#27 but only  |first number
|when main val |in the first
|of 1stfund is |triplet
```

```
(f2303)
sh              se
sh

0               ex
fnmainval

                fnloopcont
ispro           basisthis.

(f2304)
                &fnact271&
                271




                fnactcherish

(f2305)
permuteacts=    sx
|In:fundwarp    |these CAN use
|Action:        |/and change/
|performs       |'thisfcmnet'
|up to 10       |and fcmindex,
|triplet acts   |fnloopcont,
|for a foundry  |and several
|marked active  |more
```

All the action in a FCM matrix, or network, or whatever we call it happens through this function, which again is called by the TRANSLUCENT loop defined at F:2309. This functions diligently calls up to ten functions. These are listed by number in a list, and the list contains warps. This is a safe way to allow a whole FCM network to be saved to disk, and later recovered, because warps should

be generated at runtime and not assumed to be the same number between two different runs. Of course, the program that loads various parts of a FCM framework from disk must either have set up a full list of foundry actions, to cover all variations, or it must actively reorganize that list right after loading the FCM framework.

Only funds marked 'active' are performed, and only when it is marked that it does indeed have fund actions in it: the F:2232 tells you that this is #9 and #49 positions, which are set to 0 or 1. Both need to be 1 for this to be performed. In addition, the fund action must be a number in the list that is not basis /not zero/.

A fund action gets two inputs to the stack: the triplet number, and the warp to the fund. See eg F:2400 as an example of a fund action. It may or may not make use of this input, but it must be taken off the stack, as no output is expected on the stack after a fund action has performed.

```
(f2306)
ix              11:10
11              i9
ad
s9              f

fnactlist       3
lk              ad
s3              s9
```

```
(f2307)
lk              i5

|Note:
|entrainment
|can happen by
|tailor-made
|triplet-acts
s5              n?

(f2308)
d6              lo.
                |There is a
i1              |vast spectrum
ix              |of possible
i5              |entrainment
i3              |solutions;
ay              |make triplets
pf              |to do it! :)

(f2309)
translucent=    fnloopcont
|Action: this   dancethis
|is fcm main    ll:1
|algorithm;     fnloopcont
|makes use      isvarbasis
|of fcmindex    se
|and assumes
|thisfcmnet     ex

(f2310)
q1              ll:2000000000
|Note:both
|thisfcmnet     m2
|and fcmindex   fcmindex
|CAN change     lk
|within loop
```

```
|when done      ay
|thoughtfully   sx

(f2311)
|Note:the       ix
|'is active'    9
|flag at #40    wk
|allows parts
|of a net to    ix
|be switched    49
|on and off
|during run     wk
```

The card here has an innocent mistake in its comment: it says #40, whereas in the right column you see that it refers to #49. In the next card, you see the essential mechanism: when the card is set as active, AND the card also lists functions /fnacts/, it goes through the list of ten triplets to perform each listed function there properly. See F:2232 for comments of content of a foundry.

```
(f2312)
an              ix
n?
                |Note:the
                |variable
                |'thisfcmnet'
                |can be used
                |by these:
d2              permuteacts
```

```
(f2313)
fcmindqty       d2
lk

i2


                twobillion
gt              s2

(f2314)
|Note:the       lo
|flow of 'fcm   |By this
|light' thru    |feature, the
|thisfcmnet is  |checking of
|inner loop &   |'fnloopcont'
|supposed to    |can happen
|finish fast    |in outerloop
|each time      lo.

(f2315)
fcm=            |programs by
|action:fcm is  |networks of
|firsthand      |what we call
|computerised   |foundries to
|mentality;     |reflect the
|allows the     |programmers'
|shaping of     |mentality
|g15 pmn        |without 'ai'

(f2316)
|Indexlen)3?    fcmindqty
fcmindqty       lk
lk              fcmindex
4               lk
lt              bs
se              |Assumes var
```

```
                |'thisfcmnet':
ex              translucent.

(f2317)
fnact2500=      |for board
|In:triplet#,   |Result:
|foundrywarp    |triplet #10:
|Action:scans   |num1:boardx
|5x5 board      |num3:boardy
|This triplet:  |These are set
|num1:piece#    |to basis when
|num3:fn num    |piece unfound

(f2318)
s6              |ix=warp to
s5              |triplet#10:1
|i5=triplet#    |jx=warp to
|i6=warp to fn  |triplet#10:3
|We'll set:     |j5=number of
|j3=fn# that    |piece to
|has board      |scan for by
|as 5x5 array   |this fn act

(f2319)
i5              i8
3               lk
mm              t5
i6              h8
ad              h8
7               i8
ad              lk
s8              t3
```

```
(f2320)
i6              qx
39
ad
f
tx


sx              qx

(f2321)
ll:5            j5
ll:5            eq


i2              n?
i1
j3
fnww            d7

(f2322)
i2              lo
ix              lo
kl              0
                ix
i1              kl
jx              0
kl              jx
ex              kl.

(f2323)
                &fnact2500&
                2500
```

```
                fnactcherish

(f2324)
prt2numcont=    &,&
s2              prtcont
s1



i1              i2
prtnumcont      prtnumcont.

(f2325)
easyfnlist=     fcmindex
|lists fnnames  lk
|w/mainvalue,   sx
|triplet#10     prtclr
|when 1st in    prtsuspend
|it is nonnil;  ce
|You can make   ^***all listed
|variations!    t9

(f2326)
&(space)=more&  lk
t5              le
32              d4
t8
                prtrelease
11:2000000000   j9
i1              prt
nextfund        ex
```

```
(f2327)
m1              prtcont
ix              i5
ay              wgetfnam
s5              prtcont
i5
wtofnnum
prtnumcont      &' &
& '&            prtcont

(f2328)
i5              &, lev# &
10
wk              prtcont
|ie, mainval;   i5
|as for #10:    lk
|only when      prtnumcont
|1st isn't 0    &, tripl#10:&
prtnumcont      prtcont

(f2329)
i5              d7
37
ad              jx
tx              lk
                jx
jx              up
lk              up
n?              lk

(f2330)
prt2numcont     i1
|Note the use   24
|of 'de', it's  mo
|like 'se' but  n?
|jumps a given  11
|nums of lines  |ie,11 lines!
```

```
& &
prt             de

(f2331)
|11 lines:      se
prtrelease
j5              ex
prt             prtclr
ki
j8              prtsuspend
eq              |::11 lines
n?              lo.

(f2332)
fnammustfind=   tx
|In:name of fn  jx
|Gives:number   |useful when
|As fnam, but   |lining up
|tells          |new
|programmer &   |foundries
|calls 'qu'     fnam
|when minusone  sx

(f2333)
ix              ^?foundryname
ix              pp
isunsigned      jx
                pp
se
                kk
                sh
ex              qu.
```

```
(f2334)
|Each fcm app  |for apps that
|branch will   |advanced:
|have its own  |instead of
|'pet set' of  |limiting the
|functions;    |scope, we've
|and its own   |fcm here so
|brand of      |open it has
|entrainment,  |unlimited use

(f2335)
|Consult the   |standing free
|text called   |from fixing
|'learningpmn' |too much the
|part of       |'resolution'
|intraplates   |of how we
|dot com:this  |approach the
|shows the     |concepts that
|importance of |point at mind

(f2336)
|What we have  |open ideas of
|in the core   |how to make
|fcm is then   |fnacts for
|the semantics |them--incl
|of foundries, |open ideas
|cfr comments  |for subfuncs
|at f:2232     |for these, eg
|--and some    |the 'pairstk'

(f2337)
pairstk=       &&
^.             |we assert the
|good eg in    |max here, 250
|fnacts        |/note this/
255            |& set also in
2              |'pushpairstk'
```

```
mm              pairstk
sz              kl

(f2338)
resetpairstk=   |pairs;typical
0               |use in fnacts
pairstk         |is so that
lk              |each fnact
kl.             |completely
|helpful with   |finishes use
|an extra       |of pairstk
|stack for      |within itself

(f2339)
poppairstk=     pairstk
|Gives:either:  lk
|x,y,flag or:   |Note: if u
|flag=basis;    |increase sz
|Gives the      |of pairstk
|most recent    |also inc in
|pair, if       |pushpairstk
|any, w/flag=1  sx

(f2340)
ix              basis
lk              ex
s7
                q7
i7
ye              i7
                ix
d2              kl
```

```
(f2341)
i7              h8
sl              i8
up
s8              ix
                ay
i8
ix
ay              1.

(f2342)
pushpairstk=    |there is more
|In:x,y;        |computation
|as defined,    |'between' the
|max 250 pairs  |foundries
|Note that      |rather than
|when we have   |inside fnacts
|entrainment    |--pairstk is
|for fn nets,   |for fnacts

(f2343)
t5              i8
t1              250
pairstk         gt
lk
sx              se
ix
lk
s8              ex

(f2344)
i8              ya
sl
up
s3              j5
                h3
j1              i3
```

```
i3              ix
ix              ya

(f2345)
|note that     h8
|the 250       i8
|limit for
|this little
|stack is such
|that this
|func exits    ix
|if overflow   kl.

(f2346)
init3rd=
ce
&3.foundation&
825
5
rp
freshsketch
newmatrix.

(f2347)
showoff=        |Splashing a
|Startup        |little bit
|display for    |pixel stuff
|tf with some   |to stimulate
|useful inits   |all the
|of some        |g15 pmn
|drawing vars   |programmers'
init3rd         |minds ;/
```

```
(f2348)
255             10
15              10
15              25
120             10
8               50
50
60              shapelines
newtriangle     approvesketch

(f2349)
^Third          11:50
pp              freestars
^Foundation
pp
^PMN :)
pp
^^
pp              lo

(f2350)
20              newmatrix
10
120
60
150


matrixrectf     approvesketch.

(f2351)
car=            b9
showcars.
lush=           ^with lush b9,
quietb9more     b9
ce              ^& where 'q'
&You can now&   b9
```

```
b9              ^will quit
&use 'more'&    b9.

(f2352)
tripletpos=     3
|In:triplet#    mm
|Gives:start
|pos in fund
|of this
|triplet,
|where 1 is     7
|first triplet  ad.

(f2353)
fnnextval=      ww.
|In:fnnum       |Remember to
|Gives:main of  |use wtofnnum
|2nd triplet    |when you've
13              |a warp and
w               |calls on func
thisfcmnet      |that wants
lk              |fn# as input!

(f2354)
pos30x50=       fund30x50=
|in:x 0-)29;    |fn# -) x,y
|   y 0-)49     f
|/or any y/     30
|gives: fn#     mo
30              w
mm              30
ad.             di.
```

```
⟨f2355⟩
get30x50=       thisfcmnet
|in:pos x y     lk
|/0-⟩29,0-⟩n/   |Note that
|gives:value    |these '30x50'
|in fn          |routines have
30              |a fixed x
mm              |range only
ad              ww.

⟨f2356⟩
put30x50=       thisfcmnet
|in:val pos     lk
|x y
|Puts to fn     |So, y can be
                |smaller or
30              |larger than
mm              |50 in these:⟩
ad              yy.

⟨f2357⟩
up30x50=        w9
|in:val pos     |'ku' is
|x y; adds it   |defined in tf
30              |as 'kl with
mm              |up', ie, kl
ad              |that adds a
thisfcmnet      |value :⟩
lk              ku.

⟨f2358⟩
fnsetval=       yy.
|In:val,pos,
|fnnum; sets
|the position
|in the fund
|to value;
```

```
thisfcmnet
lk

(f2359)
fnaddval=       w9
|In:val,pos,
|fnnum; adds
|to position
|in the fund
|this value;
thisfcmnet
lk              ku.

(f2360)
fnfitaddval=    |Prettnumber,
|In:min max     |pn, defined
|value pos fn#  |in 3rd
|Adds val to    |foundation,
|pos in fund    |does just
|but keeps it   |this thing!
|within range
fnwarp          pn.

(f2361)
fnposwarp=      thisfcmnet
|In: pos, fn#   lk
|Gives: warp
|Like fnwarp,
|but instead
|of pos basis,
|here pos can
|be any         w9.
```

```
(f2362)
fngetval=       ww.
|In: pos,
|fnnum; gets
|the value
|from position
|in the fund
thisfcmnet
lk

(f2363)
fnaddmainval=   w9
|In:val, fnnum
|Adds val to
|mainval of fn
10
w
thisfcmnet
lk              ku.

(f2364)
fnaddnextval=   w9
|In:val, fnnum
|Adds to main
|of triplet#2
13
w
thisfcmnet
lk              ku.

(f2365)
graphfnval=     s1
|in:x y v1 v2   s2
|x/y is here
|switched;      |v1, v2 range:
|average of v1 |0 to 800,000
|& v2 is tone
```

```
tx              sl
sx              s1

(f2366)
i2              d
25              up
ad


i1
15              d
ad              up

(f2367)
ix              makefit
jx
ad
sr
3137
di
0
255             matrixrect.

(f2368)
graphboundary=  sl
|in:x y         15
|x/y switched   ad
sl              s1
25              i2
ad              i1

s2              d
```

```
(f2369)
d                i1
up
                 d
255              d
matrixrect       up
i2
                 0
up               matrixrect.

(f2370)
graphhere=       j1
|in:v1,v2,x,y    800001
                 lt

s4
s1
t9
t1               d4

(f2371)
i1               i1
i4               i4


graphboundary
                 j1
                 j9
ex               graphfnval.

(f2372)
|Startwave may thisfund
|use factor in basisthis
|tp, 62832,    wavenumconst=
|/ca two pi/,  7854.
|eg 7854 or    |'fastvar':you
|3927, which   |can modify it
```

```
|factors tp in nextfund
|resp 8 & 16   basisthis

(f2373)
startwave=     ad
|In:tr#,fnwrp  t5
tx             j5
sh             tp
10             lt
jx             d2
wk             basis
wavenumconst   t5

(f2374)
j5             50
si             jx
40             wk
mm             setfnmainval
|Main to basis j5
|Uses link#1   10
400000         jx
ad             kw.

(f2375)
bringwaveon=   s3
|In:angle val  s9
|triplet# fn#  s6
|Via carrywave |chks fn isn't
               |a boundary:
fnwarp         jx
tx             10
tripletpos     wk
```

```
(f2376)
800000          0
gt              800000
                i9
se              i3
                jx
                |We call tf
                |prettnumber:
ex              pn

(f2377)
i6              sx
i3
u2              0
jx              9
kw.             ix
                thisfcmnet
relaxfn=        lk
|in:fn#         yy.

(f2378)
selfactivefn=   yy.
|In:fn#
sx              spreadwave=
1               |in:fnwrp
9               |Via carrywave
ix              |Assumes lux
thisfcmnet      |value #40 has
lk              |triplet#

(f2379)
tx              wtofnnum
40              fund30x50
jx              |Note: spreads
wk              |to y+1 and to
tripletpos      |x-1 thru x+2
sx              up
```

```
                s2
jx              s1

(f2380)
10              t5
jx              |Both
wk              |intensities
13              i1
jx              u2
wk              i2
ad              pos30x50
sr              s5

(f2381)
f1              m1
i2              i2
pos30x50        pos30x50
s6
i1
i2
pos30x50
s7              s8

(f2382)
j5              1
ix              ix
|intensity      u2
|of wave:       |angle:
|1st val of     |3rd val of
|triplet        |triplet
i5              i5
fnsetval        fnsetval
```

```
(f2383)
j5              2
ix              ix
                u2



i6              i6
fnsetval        fnsetval

(f2384)
j5              3
ix              ix
                u2



i7              i7
fnsetval        fnsetval

(f2385)
j5              4
ix              ix
                u2



i8              i8
fnsetval        fnsetval.

(f2386)
|simple 30x50  |  * *      *
|waves guided  |  |/     /
|by angle#, #40| #2    #1-)*
|*     *  *    | ============
|  \    \ |    | #5-#8 are
|*(-#4   #3    | side&beneath
```

```
|#1-#4 are side| in y, w/#8
|&forward in y | nearest #1

(f2387)
|For eg         | #7    #8-)*
|reflected      |  |\     \
|waves, #5-#8:  |  * *      *
|*(-#5     #6   | ============
|  /     / |    | circular
|*      *  *    | waves using
|Use eg func    | a good deal
|'si' for more  | more nodes

(f2388)
carrywavehere= i9
|In:tr#, fnwrp u2
|Via carrywave jx
tx              wk
sx              |i4 is angle#
ix              |as prev cards
tripletpos      |explain
s9              s4

(f2389)
i4              i9
1               jx
8               wk
isnotwithin     |Note:angle#5
|angle# here?   |to #8 used in
se              |eg reflection
                |of waves
ex              t5
```

```
(f2390)
j5              400000
400000          j1
su              680
|waveresonance pm
|tuningnumbers ad
810             i9
pm              jx
t1              kw

(f2391)
jx              |By tn we get
wtofnnum        |16 simple
|pd that does   |lines; tf has
|nothing: tn=   |'dh' which is
|TraNquility:)  |like 'd2' but
fund30x50       |counts *16*:
s2              m4
s1              dh

(f2392)
i4              i4
j1              j1
ix              ix
f1              f1
i2              f2
tn              pos30x50
pos30x50        bringwaveon
bringwaveon     ex

(f2393)
i4              i4
j1              j1
ix              ix
i1              f1
f2              f2
tn              pos30x50
```

```
pos30x50        bringwaveon
bringwaveon     ex

(f2394)
i4              i4
j1              j1
ix              ix
m1              i1
f2              f2
tn              pos30x50
pos30x50        bringwaveon
bringwaveon     ex

(f2395)
i4              i4
j1              j1
ix              ix
m1              m1
i2              f2
tn              pos30x50
pos30x50        bringwaveon
bringwaveon     ex

(f2396)
i4              i4
j1              j1
ix              ix
m1              m1
i2              m2
tn              pos30x50
pos30x50        bringwaveon
bringwaveon     ex
```

```
(f2397)
i4              i4
j1              j1
ix              ix
m1              i1
m2              m2
tn              pos30x50
pos30x50        bringwaveon
bringwaveon     ex

(f2398)
i4              i4
j1              j1
ix              ix
i1              f1
m2              m2
tn              pos30x50
pos30x50        bringwaveon
bringwaveon     ex

(f2399)
i4              i4
j1              j1
ix              ix
f1              f1
i2              m2
tn              pos30x50
pos30x50        bringwaveon
bringwaveon     ex.

(f2400)
carrywave=      800000
|In:tr#,fnwrp   gt
tx
sh              se

10              ex
```

```
jx              |This isn't
wk              |a fence

(f2401)
40              d3
jx              |this fnact
|The 'luxury'   |doesn't
|pos #40 is     |reflect at
|only used for  |boundaries
|spreadpoints   jx
wk              spreadwave
n?              ex

(f2402)
11:2            &carrywave&
i1              1234
jx


carrywavehere

lo.             fnactcherish

(f2403)
fcmheadertxt=   fcmpausekey=
^.              ^.
                32
fcmshowpause=   fcmpausekey
^.              kl
100
fcmshowpause    fcmlooptxt=
kl              ^.
```

```
(f2404)
fcmmaybepause= n?
ki             se
fcmpausekey    ex
lk             ki
eq             sh.
f
fnloopcont     fcmgraphloop=
kl             ^.

(f2405)
60             20
fcmgraphloop   bx
kl
fcmdrawintro=  fcmlooptxt
ce             lk
fcmheadertxt   105
lk             673
105            bx.

(f2406)
graphsomefns=  s5
|Fnact to      i5
|show activity n?
tx
sh
10             se
jx
wk             fcmdrawintro

(f2407)
i5             mo
up             ye
10
jx             se
kw
i5             ex
```

```
fcmgraphloop
lk              freshsketch

(f2408)
i5              11:35
makenumber      11:30
                m2
860             m1
668             pos30x50

                f
rp              fnmainval

(f2409)
w               fcmshowpause
fnnextval       lk
m2              activepause
m1              ck
graphhere
lo              se
lo
approvesketch   fcmmaybepause.
```

Some of the within 200 cards listed here have other aspects to them than setting up the FCM framework, but for completeness we included them all exactly as in the standard G15 PMN Third Foundation, which is the core--with few or none extensions needed in the essence package--of all future G15 PMN work including with FCM and what we call Open Robotics. It is good to have this overview, as presented in this chapter, handy in reading the advanced G15 PMN examples involving FCM in the remaining volumes in this book.

Let us say what we regularly say in connection to all

program, but strongest of all when it comes to FCM: give deep thought and mindfulness to the full range of possible effects of putting your program into action in the real world, especially if it is connected to motors, engines, such as robots, or affect people emotionally. FCM can and should be built with ethical constraints when it comes to huge applications.

*****