# Think the Image, Don't Make It! On Algorithmic Thinking, Art Education, and Re-Coding

**Frieder Nake**
Informatik, University of Bremen,
Bremen, Germany
-----
nake@informatik.uni-bremen.de
-----

**Susan Grabowski**
Informatik, University of Bremen,
Bremen, Germany
-----
sgrab@informatik.uni-bremen.de
-----

## ABSTRACT

In conceptual art, the idea is not only starting point and motivation for the material work, it is often considered the work itself. In algorithmic art, thinking the process of generating the image as one instance of an entire class of images becomes the decisive kernel of the creative work. This is so because the generative algorithm is the innovative component of the artist's work. We demonstrate this by critically looking at attempts to re-construct works of early computer art by the re-coding movement. Thinking images is not the same as thinking of images. For thinking images is the act of preparing precise descriptions that control the machinic materialization of images. This kind of activity is a case of algorithmic thinking which, in turn, has become an important general aspect of current society. Art education may play an important role in establishing concrete connections between open artistic and more confined technological ways of thinking when thinking progresses algorithmically.

## KEYWORDS

Surface; subface; algorithmic sign; algorithmic thinking; re-coding; Processing; generative art; art education.

## ARTICLE INFO

Received: 18 December 2017
Accepted: 19 December 2017
Published: 22 December 2017

https://dx.doi.org/10.7559/citarj.v9i3.458

## 1 | INTRODUCTION: A FIRST EXAMPLE [1]

Without any initial ado, let us immediately engage in a little experiment. Take a look at the drawing in Figure 1. Imagine you were given the task of finding out how it was made. You are told that the drawing was done by a drawing machine controlled in its movements by a computer program. You are requested to come up with such a program that is capable of generating drawings of the same kind or style. Your code is not required to generate exactly the drawing you see in the figure. But there should be little doubt that your program, if allowed to keep on generating such drawings, one after the other, would some day produce almost exactly the given drawing. How would you approach this challenge?

In all likelihood, after a moment of pondering, you would say that there are only horizontal and vertical line segments.You observe that, at the far left and high up, a line is starting. You follow it down until it curves to the right in a right angle. You will probably find it hard to follow the line much further since you lose track of it. But you see other cases of sequences of consecutive line segments that may produce in your mind the hypothesis that there are straight line-segments alternating between horizontal and vertical direction, going left or right, and up or down. The hypothesis would be a bit daring to claim that you really see one line only starting at top left, and continuing by taking turns between horizontal and vertical lines. You don't see where, in the drawing, this game is ending.

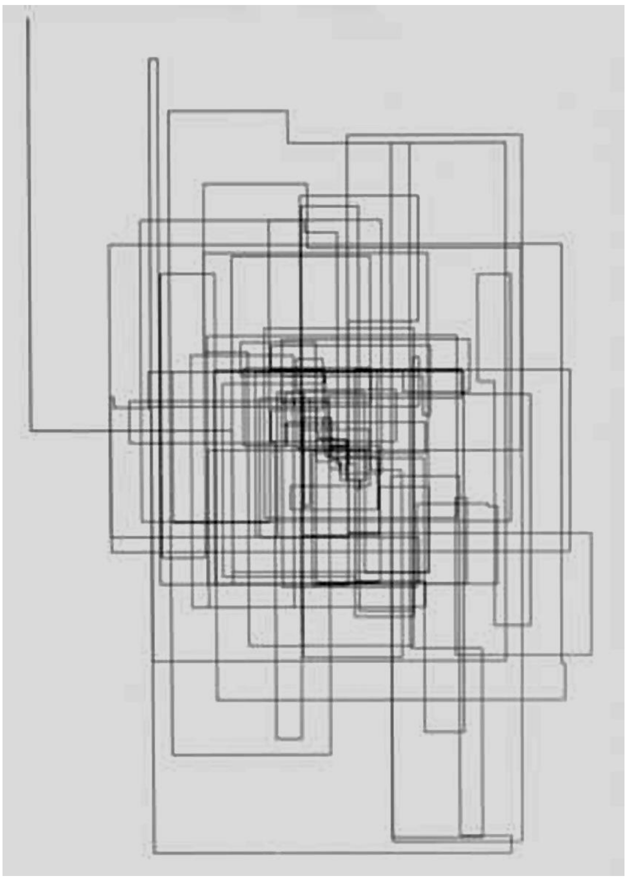

Figure 1 | Georg Nees: Irrweg 1965. Credit: G. Nees [2].

You feel ready to sit down and write code in a rather free symbolic form that you find easy to read. It may in a pseudo-code look like the following.

```
input countMax;
randomly choose a point inside the space provided for the image and call it "P0";
randomly choose a first direction from the two options {vert, hor} and call it "dir";
count := 0;

repeat the following until count > countMax:
{
    randomly choose an orientation from the two options {'+', '-'} and call it "or";
    randomly choose a length for the next line segment and call it "len";

    if (dir = hor) do
      { if (or = '+') do
          { P1.x := P0.x + len; if (P1.x > right) do { P1.x := right } }
        else do
          { P1.x := P0.x - len; if (P1.x < left) do { P1.x := left} }
        P1.y := P0.y }
    else do
      {if (or = '+') do
          { P1.y := P0.y + len; if (P1.y > down) do { P1.y := down} }
        else do { P1.y := P0.y - len; if (P1.y < up) do { P1.y := up} }
        P1.x := P0.x };
   draw line-segment from P0 to P1;
   P0 := P1 in coordinates;
   if (dir = vert) do { dir := hor } else do { dir := vert };
   count := count + 1;
}
```

This is an algorithmic description of a line drawing characterized by the following features:

The drawing is made up of one polygon whose edges alternate between horizontal and vertical direction. Edges are of random length and they stay within the given format of the image (whose left and right boundaries within a given coordinate system go from the x-coordinate "left" to "right"; in the vertical direction they extend from "up" to down"). Whether the edge goes left or right (and, correspondingly, up or down), is decided randomly. The polygon has *countMax* edges. It should be noted that the x-coordinate runs from left to right whereas the direction of y is from top to bottom.

Even though the description may appear a bit cryptic, it should be understood quite easily. We understand the individual lines of this description if we know precisely what must be done to generate the drawing. We may not yet know what must be done in order to make sense of the several appearances of the words "randomly choose ...". This part is still open and must be described before we accept the above as an *algorithmic* description. The symbol ":=" stands for the operation "the variable on the left of ':=' takes on the value that the expression to the right of ':=' evaluates to". This symbol is called the *assignment operator*. Its function is to assign a new value to a variable.

Algorithms are descriptions of calculations, and such calculations are organized in sequences of discrete steps of simpler calculations. Each calculatory step changes in a precise and unambiguous way the state of at least one of the variables of the algorithm. Ultimately, assignment operations are responsible for such state changes. Therefore, the assignment is the most basic in all algorithmic descriptions.

Our example is an algorithmic description because it is of finite length, it is unambiguous, and it is effective. A few conventions must be added to make the description unambiguous and effective. The property of being effective means that in each case of interpreting the individual *statements* that constitute the description, the interpretation must end in operations whose meaning is already known in each and every detail. In the end, a machine must be available that can carry out the description.

To many of our readers this exercise may be pretty boring and trivial. We have started the exercise from one of the first examples of generative art (often called "computer art"), i.e. from a given visual object. This object is trivial enough so that we could easily suggest simple operations of which we are intuitively certain that, if carried out truthfully following the description and nothing else, they are capable of generating drawings of the kind displayed in Figure 1. Even more: We claim that our description is capable of generating *all* those drawings, their entire class.

This is a most important aspect of algorithmic art. This kind of artistic activity is interested in classes of images, i.e. in infinite sets of images! The individual image is reduced to an instance only of the class it belongs to. We may like or dislike one or the other of the productions resulting from executing an image generating algorithm. That's nice and justified but – we are inclined to say – it is no longer at the center of the aesthetics of this kind of art.

We don't want to be misunderstood: Like you, we have our taste, and prefer some image over another one. Such emotional or otherways founded judgements are okay and will stay with us. Our point, however, is the shift from the individual and isolated image to the infinitely many images. With algorithmic art, a new relation between us, the appreciators, and them, the works, was born. Questions can now be raised like: Are all instances of a class of images of high interest? Do most of them arouse deep feelings of pleasure or beauty? Under which settings of the controlling parameters does the algorithm generate works of high quality? And for which of these settings is this not the case? If we are capable of answering such questions, we will come up with statements about good style. Such statements would, most likely, be possible only by experiment.

## 2 | A SECOND EXAMPLE

The heading of this essay contains a provocation: "think the image!" As if this were not outrageous enough, it contradicts explicitly what we would usually think: Images are to be made, drawn, painted, otherwise implanted or embedded into or onto a material. Yes, it is true: as long as we accept something to be an image only if it is an entity to be sensually perceived, it must exist in material form. So, *thinking the image* sounds like a stupid request.

But be aware of conceptual art! Concept was, of course, always already part of the artistic process and practice that artists, to a lower or higher degree, are concerned with in sketching, experimenting, modelling, trying, repeating, revising, versioning, in short, with conceptualizing the work of their concern. With exception of a few cases, an artist is concerned with taking practical steps to get closer to realizing his intuition, his imagination, her intention, her fantasies, etc. When in conceptual art the concept itself was pushed up to the primary concern of an artist, this was an interesting further step in deconstructing the work of art. Sol LeWitt, in 1967, gave the formula: "The idea becomes a machine that makes the art." (LeWitt, 1967)

We do not know whether LeWitt was aware of the fact that two years earlier the first exhibitions had been shown of art that came out of the machine (Georg Nees, A. Michael Noll, Frieder Nake). If not, someone should have written to him: you are right, and it is happening already! But such detail is not important nor interesting. In historic perspective, it is all the same. The idea of thinking the image emerges at different places. At some places only as a nice, surprising phrase, at others in actual artistic practice. Thinking the image, however, in a radicalized form, becomes necessary in algorithmic art. Developing an algorithm for the production of just one image would be stupid and crazy. Thinking the image leads to thinking sets of images.

The style an artist finally discovers to be the style that he or she, from now on, will be using and varying over and over again, the break-through that they have been waiting and working hard for, is of utmost importance in the course of art, in the unfolding of history of art. The style appears in one after the other work in high times of an artist's art. In the style, the artist expresses his or her thinking of images. So thinking the image has a long tradition and is, during the twentieth century, a major force propelling artistic processes. It is only consequential that around the middle of the century, this thinking the image was singled out as the creative approach to the world of images for the second half of that century and into the next. Thinking the image is the ontology of imagery in postmodern times. The material image is added as a sentimental reminiscence to comfort our dreary senses. The senses need to be nourished by the material image because we are not brains, much more we are bodies.

When Descartes said, "I think therefore I am", we would be liars if we continued by saying, "I think the

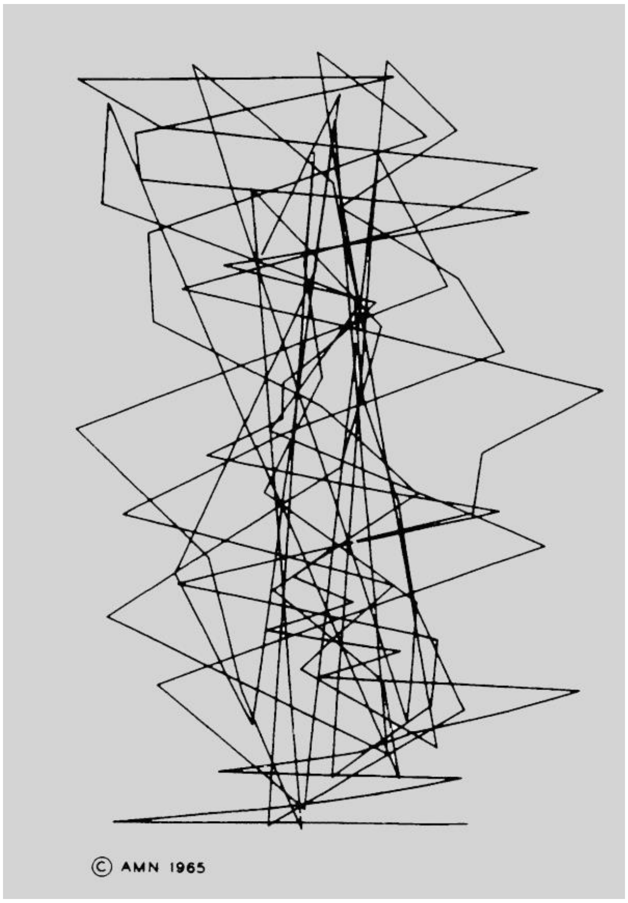

Figure 2 | A. Michael Noll: Gaussian Quadratic. 1963/1965. Credit: A.M. Noll [3].

image, therefore I see it". Only vis-à-vis the extreme semiotic machinery (digital computers) the request becomes possible to think the image. It becomes necessary at the same time.

But let us take up a second example (cf. Figure 2)! It is not too hard to convince ourselves of this line drawing being a polygon, again. A polygon is represented as a finite sequence of points. A polygon as an object of drawing is a specialized visual interpretation of such a sequence of points. The interpretation consists of straight line-segments from the first point to the second, from there to the third, etc. The polygon is closed, if the last point is connected back to the first.

An algorithmic description of the class of all open and closed polygons is easy to sketch. In this sketch, the variable "close" must be set to the value "true" if a closed polygon is to be generated. Otherwise, it must have the value "false". As before, the value of "countMax" is the number of edges in the case of an open polygon. The closed polygon has one additional edge. So here is the algorithmic description in the same sort of formal writing.

```
input the value of countMax; input the value of close;

randomly choose a point inside the image frame and call it "P0";
Pold := P0;
count := 0;
repeat the following until count = countMax:
{
   randomly choose the x-coordinate of the next point, according to function
   nextX, and call it Pnew.x;
   randomly choose the y-coordinate of the next point, according to function
   nextY, and call it Pnew.y;
   draw straight line from Pold to Pnew;
   Pold := Pnew;
   count := count +1;
}
if (close) do { draw straight line from Pnew to P0 };
```

The interesting part of this description are the two lines that are responsible for determining the x- and y-coordinates of the next point, called *Pnew*. Two functions are mentioned that are obviously decisive for what is going to happen, heavily determining the visual appearance of the polygonal drawing: the functions *nextX* and *nextY*.

That choice can be done by actually fixing the coordinates of a next point somewhere inside the image format. A different kind of determining that point is to choose a direction, and a length along that direction. This choice can be made in a deterministic or probabilistic way. The probabilistic choice is one where the new value is drawn from some given interval, according to some probability distribution. If the interval to choose from has length 0, it is a deterministic choice. The kind of probability distribution has a strong influence on the shape of the polygon. Our first example above had some deterministic part (the alternating directions of vertical and horizontal). Other components of the choice were of probabilistic nature.

Michael Noll's polygon (Figure 2) indicates in its title a bit of what distinguishes the drawing from others – a practice often used in the fine arts. Noll called the drawing, "Gaussian quadratic". From this name, we may expect that one of the two coordinates is distributed according to a Gaussian distribution. It is actually the horizontal x-coordinate. The other coordinate is to progress according to a quadratic equation. (Noll, 1994) Noll did not publish more detail about this progression in vertical direction. We see that the line is mirrored back to the bottom boundary whenever the y-coordinate reaches the top.

In neither example, we have said anything about the graphics. Our discussion was confined to the geometric aspects alone. Geometry, however, is abstract. Thus, it is not visible. A geometric line is an ideal object of the mind. It is not a stroke executed on paper by use of a pencil or any other drawing tool. Only when we add graphic parameters to the geometric objects, does the drawing appear as a visible object. So the two examples above must be amended by parameters like the width and color of the lines, perhaps also a deliberate visual emphasis on the points, changes of the values of graphic parameters along the edges, textures of the line segments and more.

## 3 | RE-CODING

Coding is the activity of describing an algorithm in the form required by some programming language such that the compiler or interpreter of the chosen language can deal with it. *Re-coding* must then be the activity of trying to re-construct the code of a drawing – in the case where drawings are the results of machinic calculations (that's our case).

The situation is simple and clear. We are given a drawing. Perhaps there are more than only one drawing. We know, or assume, they are results of executing a computer program. But we don't have a clue of the program itself; at any rate, it is not available to us. In the case of several given drawings, we assume they came from the same program. In such a situation, our task is: Design code that is capable of generating images similar to the given images, plus any number of more.

We are not interested in *copying* the given images. We say the task is solved, if the new code generates images that come close enough to the given ones. Images must not coincide in each and every detail. It suffices if a person, comparing the given to the re-coded images, concludes that the two evidently belong to the same class of images, in which ever way that class would be defined. Intuitively, we seem to be capable of judging quite well such vague kinds of similarity.

We cannot hope for more than such class-similarity in re-coding. If we define the task more strictly by requesting that one of the re-coded images be equal in all its detail to the given one, we could always provide a trivial re-coding: scan the given image, store it, and output it upon request. If not only one, but several images were given at the start, the trivial code would have to be slightly more complex. We would scan all the given ones, store them, and prepare code containing a switch that randomly selects and outputs one of the stored cases.

We conclude that the task of re-coding always allows for a super-trivial solution that is not based on constructive code. The trivial new coding, in fact, evades the problem of coding altogether. An acceptable solution of the re-coding problem requires that the new image be constructed in such a way that the new encoding requires a good measure of similarity between the given sample of results from the unknown program and the output of the new code. To solve this task, we must carry out an analysis of basic elements, structures and superstructures, measurements, and other analytic investigation of the given sample. The job of re-tracing an unknown algorithm of generative art surprisingly puts us into the situation of an extremely accurate analysis of works of art.

In tasks of re-coding, we see a good chance for introducing generative art into art education. We see chances for school kids, students, and adults to engage in both, artistic and algorithmic, activities offering new and rewarding experiences. Such activities may have the potential of helping us to remain true human beings whilst the environment around us is accelerating the digital race. We see chances for slow lingering against fast glimpsing.

In the fall of 2012, the US-American "creative technologist" (as he called himself), Matthew Epler, came up with the idea of re-coding works of early algorithmic art. Epler considers himself a person "specializing in creating one of-a-kind interactive projects" (Epler, w.d.). On the website of his ReCode Project (meanwhile largely abandoned) he wrote:

> *"The ReCode Project is a community-driven effort to preserve computer art by translating it into a modern programming language (Processing). … The focus of the ReCode Project is three-fold:*
>
> *1. Bring historic works of computer art back into the public eye.*
>
> *2. Make it accessible and useable.*
>
> *3. Save the code."* (Epler, 2013)

Not much later, British artist Mark Webster suggested to Epler to organize a series of events, including the idea of cities declaring themselves to become *ProcessingCities* [4]. Whereas Epler concentrated on his website inviting people to re-code early computer art, Webster wanted people to connect in actual spaces experiencing aspects of art history in a new way by learning from pioneers. It seems that besides a challenging but quite limited re-coding workshop and lecture in 2013 in Bordeaux, France, not much has actually happened. As far as we know, there are hardly any publications about the approach, and after a series of rather trivial re-codings of computer-generated images taken from Grace Hertlein's short-lived magazine *Computer Graphics and Art* (Hertlein, 1976-1978) nothing tangible seems to have come from the idea.

The very idea of re-coding, however, caught on in our research group at the University of Bremen in Germany. Perhaps different from the original intentions, and not without critical analysis of the potentials of the approach, we have over a number of years gathered a fair amount of hands-on experience in study projects of re-coding. As part of our long-term project on algorithmic art [5], we have offered several seminars and workshops for individual re-coding efforts of various kinds. So even if Matthew Epler's first impulse was met by only little resonance, via Mark Webster's French connection the idea of re-coding fell on fertile ground in the North of Germany.

## 4 | TWO MORE EXAMPLES

The idea of re-coding has strong limitations. The two examples we have used for the purpose of demonstration are extremely simple. Even though the drawings contain areas of densely packed intersections of lines, where it is almost impossible to successfully discern the exact paths of criss-crossing lines, we were able to come up with highly probable correct hypotheses.

Such a situation is, however, exceptional and very soon the complexity of a set of images gets so high that only by accident or by great expert experience in a certain field of scientific and artistic decision-making is there any chance to re-code the original algorithm. Based on Figures 3 and 4, we will indicate cases of almost impossible tasks of re-coding.

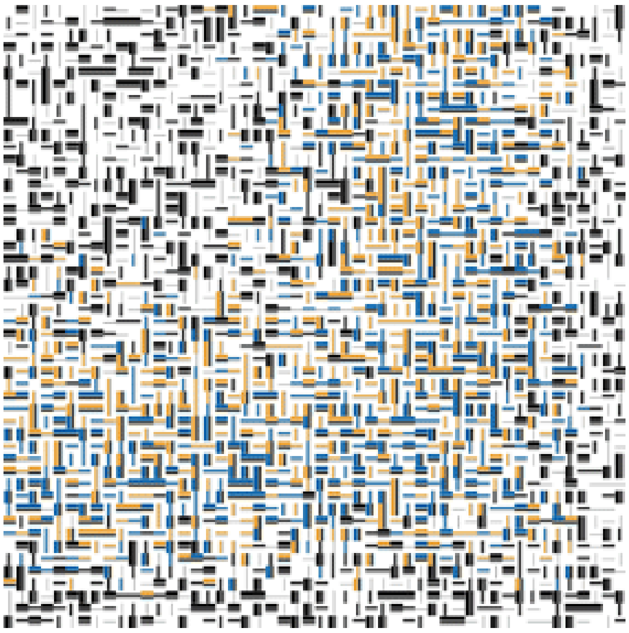
Figure 3. Frieder Nake: Abteiberg Walk through Raster, 2005.

Saying something is impossible, is dangerous. Why should not someone of advanced background in algorithmic art appear who, given enough time, would eventually find the level of analysis where our claim of unbreakability breaks down? We want to indicate this level in the case of Figure 3.

Let us make explicit a few characteristics of the image of Figure 3, and also pose questions that must be answered if we want to stand a realistic chance of re-coding the drawing. We easily discover horizontal and vertical bars of black, blue, and yellow color. They are of varying line-width and length. We see empty spaces between the bars, irregularly distributed, as it seems. An enlarged version of the image would help to measure the lengths and widths and, thereby, come up with a description of the set of elementary signs used for the image. Some initial insight is possible.

But what about the areas where the black color almost disappears to make room for the blue and yellow strokes? What are shape characteristics of that area? Assuming from earlier experience that this kind of algorithmic art is likely to be based on some kind of randomness, and knowing that randomness comes in many varieties according to probability distributions of many kinds and parameter settings, we may generate conjectures. But how to test them in an attempt to get closer to a convincing re-coding? Is there a definite borderline to the somehow curvingly-shaped inner area of blue and yellow? Once in a while, we still see a bit of black but the two black and not-black areas are hard to delineate. If geometry doesn't help the analysis, what else can we try?

We have reason to assume that without help from outside, perhaps from the artist himself who – thinking the image and not making it – made use of something that defies re-discovery unless an accidental flash of thought comes to rescue. In order to offer such a flash, who of our readers would have assumed that the image was first generated as a *Markov chain* with non-stationary transition probabilities? It was then mapped from the linear chain onto the planar area by a space-filling method. Did your thinking go in this direction?

Our claim is that it is extremely unlikely that this kind of conjecture emerges in one of the observers. The hint of "Markov chain" will, in the mathematically educated generate some inkling of "Aha!". But such a mind would still need to do a lot of analysis to solve the riddle in a way that would allow us to accept the new code as a re-coding. We may be inclined to conclude, that the task is, under most circumstances, too complex to solve.

It is definitely too complex to solve if we are given only one image to start with. The situation may change mildly if we start from ten or twenty images that the original program has generated. The larger the sample is, and the more advanced our analytical tools are – themselves given as software –, the higher our chances for a decent re-coding. This would tend to become a case of *big data*!

The program behind Figure 3 was called *Walk-through-raster*. According to the non-stationary transition probabilities and one of several modes of mapping a chain onto the plane, an arbitrarily given repertoire of signs is distributed into the cells of a grid. The algorithm, by its rather strong local control is capable of letting emerge global structural or compositional features. We believe, this was, fifty years ago, a remarkable discovery.

Casey Reas' image of Figure 4 is another example beyond the simplistic assumptions of naive re-coding. Without clues from the artist, no analyst has a chance to re-code it. We do know that there is an algorithm behind the visual appearance. But, in this case, to come up with at least a description of events that may happen (the image is taken from a dynamic installation) appears much too complex. We may admire what we see but to say precisely why we admire it, seems beyond our imagination. Remember: If our descriptions are to be turned into algorithmic form, they must satisfy hard requirements.

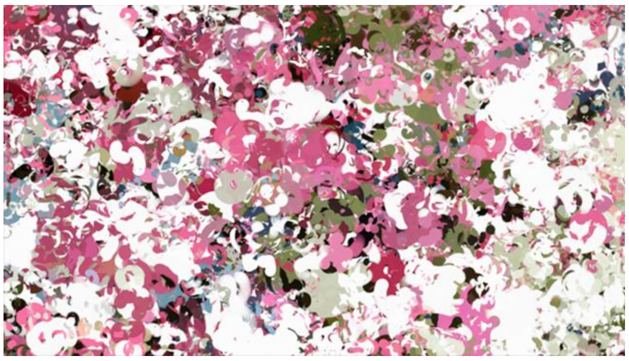
Figure 4. Casey Reas: Process 14 (Software 3) 2012. Credit: C. Reas.

We hope to have now indicated the extremes (of trivial and impossible), between which a kind of new approach to art education may be tried. Neither the trivial nor the (almost) unattainable are interesting for educational purposes. Interesting is the possible, the challenging but realistic. That's the middle ground between the extremes.

## 5 | ALGORITHMIC THINKING

Like all other animals, humans engage in purposeful activities most of their lives. A very special of our activities is happening without much of other movement: Thinking, reflecting. We can see a person sitting somewhere without speaking, not moving arms nor legs, silently looking in one direction without showing signs of bodily engagement, perhaps even with eyes closed – in short, a symbol of high concentration. He or she, we may conclude, is thinking. If our ordinary activities are oriented towards changing something in the outside environment around us and if we call these *external activities*, our *internal* activities are oriented towards changing something inside. We may say, they change our inner state.

Of course, those two modes of activity cannot be strictly separated. In reality, they are interlinked and interwoven. For analytical purposes, however, separating is justified. Separation permits us to say that internal thinking takes as its subject matter an external operation. An individual's thinking is reflecting an *other*. The other may be the thinker, or an operation by the thinker.

For practical reasons, we distinguish various kinds of thinking: Speculative thinking, logical thinking, associative thinking, and more. Each one of those is a reduction of thinking to a special subject matter or a special manner of thinking. When we speculate, we allow ourselves to think of everything only loosely connected to the current subject matter, or not connected to it at all. In speculating, we allow ourselves to leave behind what we wanted to concentrate on, often together with others.

In logical thinking we allow for progression from one statement to the next only by obeying a set of rather strict rules that control sequences of interconnected statements to be made in such a manner that all those participating in the activity agree on the conclusion derived from one or several of the already established propositions. To say that a conclusion was erroneous, in the case of logical progression, amounts to the proof that a derivation did not obey the agreed-upon rules. In logical thinking, a maximum of non-subjectivity is achieved.

In associative thinking, the connection between one or several already established statements and a next statement is much looser. Statements in a sequence of statements are connected (and, thus, build a derivation or progression of thought) by some common words or feelings or subjective experience, often quite close to speculation. Whereas in speculation each and every neighborhood is permitted in a chain of thought, in association a vague kind of aesthetics is allowed.

Here, our interest is *algorithmic thinking*. It is a way of thinking *towards* algorithms and *in* algorithms, or in statements and formulations that satisfy the rigor of precision, clarity, operability, and unambiguity. Algorithmic and logical thinking are close relatives of each other. Algorithmic thinking is even stricter than logical thinking insofar as all final consequences in algorithmic thinking must be operational. That is, it must be possible to carry out by machine the final operations.

The particular kind of machine, its details of operation (the operating system's and programming languages' intricacies) are of no avail. In algorithmic thinking, we are not thinking of programs, but of algorithms. Algorithms are the abstract forms of programs. But the two are, of course, very close to each other. Somebody may be a good algorithmic thinker without being a good programmer. But it is unlikely that he or she would not digress into a bit of programming just to see whether an algorithmic formulation would survive the purgatory of a concrete computer.

The result of an effort in algorithmic thinking is an algorithmic system. As such, it is in all its aspects unambiguous. This amounts to saying, there is one and only one interpretation of each execution of the algorithmic system. The result of such an execution may be interpreted differently by the human witnesses, of course. But those humans embed the algorithm's result into their contexts, immediately and by necessity. By doing so, the humans interpret in their ways, namely according to their interests, intentions, etc. The machine, when executing the algorithmic operation, has only one context: the context of computability. Between two interventions by the human ("interactive acts") everything is computable and, thus, does not allow for different interpretations.

So algorithmic thinking is a way of thinking by humans who are in full command of all their incredibly rich powers of interpretation. However, in the course of its happening, such a thinking must narrow down everything to uniquely and precisely interpretable actions. Algorithmic thinking is thus a kind of thinking that requires of the human an approach and attitude that is non-human. Algorithmic thinking is deeply heroic because it demands of the semiotic animal (the human) to prepare for the semiotic machine (the computer) to take over, and this taking-over can successfully be prepared only if the human reduces all his or her capabilities to the trivial.

We may describe the human's task in thinking algorithmically by a sequence of three reductions. Whatever the process and phenomenon may be that a group of humans are considering worthwhile to be redone in an algorithmic way, they must do this:

1. reduce the worldly process to semiotic aspects,
2. reduce the semiotics to syntactics only,
3. reduce the syntactics to computability only.

## 6 | ART EDUCATION AND THE ALGORITHMIC DIMENSION

The algorithmic dimension of a phenomenon is all that pertains to the phenomenon under consideration when we reduce events, things, and processes to their algorithmic components, aspects, features, characteristics, etc. We enter the algorithmic

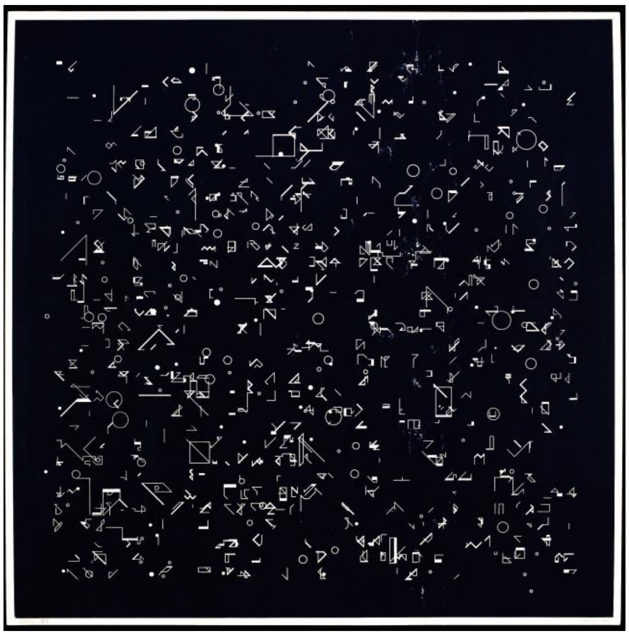

Figure 5. Manfred Mohr: A formal language (P-49), 1970. Credit: V&A Museum London.

dimension when we strictly reduce thinking to its algorithmic forms. In a strict sense, none of us can do this. But we are pragmatic enough to know that, in any real-world activity and contemplation, we can easily jump back- and forward between certain sets of restrictions that we impose on our activity. We may say that, when thinking algorithmically, we inhabit the algorithmic dimension of the world. We have described how this is a domain in the world of unambiguity, of one and only one interpretation, of the reduction of rich human capacities to poor machinic capacities [6].

How different from art that is! Can we imagine a greater difference than that – between the infinitely open space of interpreting and doing anything in the artistic domain, and the poor one-and-only-one interpretation that the machine requires, by the very fact of being a machine, if we want to use it in an interesting, new, exciting, extravagant, arousing, beautiful, surprising, heavenly or devilish manner!

There is, we claim, no greater difference. And yet, we know, that exactly this is happening: the impossible meeting of always only one interpretation on one side, and infinitely many interpretations on the other. People now usually call this enormous revolution the *Digital Revolution*, and they speak of what is happening as the process of digitization. We prefer the term *Algorithmic Revolution*. For, what is happening is characterized much better by processes of reducing anything in society to computable, i.e. algorithmic, form. The fact that this takes place by all objects being coded in digital ways, is only accidental and not really worth mentioning. The belittling of the computability reduction as a digital encoding thus appears as an ideological attack against an enlightening critique.

Art is enlightening, often in the most literal sense. Art education – as a stream of educational efforts towards free and open judgement beyond immediate interest – requires awareness and practice of enlightened critique.

Art education in postmodern times, in times of the algorithmic revolution, in times of permanent accessibility and ubiquitous surveillance, in times when the individual disappears behind a permanent flow of data from all of his or her innocent and everyday activities, the transformation of the human being into a source of data – art education in such times should and can be taken up as interruption and intervention. Art always disrupts and disturbs, at least this was the case during the twentieth century. That century ended in a close connection between human and machine by the machine's transformations into automaton, tool, and medium. The three capacities of computability, interactivity, and connectivity stand for these three ontological modes of the semiotic machine.

Works of art are complex signs, open for never ending chains of interpretation, open for the three ontological aspects of the machine. Outside of the realms of affirmative action, there is only the realm of art where the human can still unfold his or her genuinely human capacities freely and critically. We are all responsible to contribute to this.

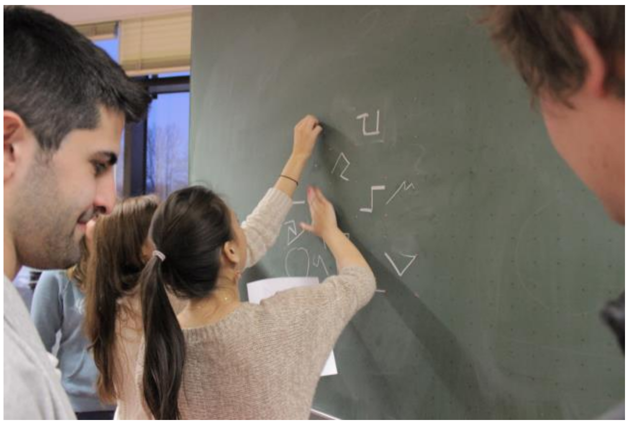

Figure 6. Students analyzing Mohr's "A formal language". University of Bremen 2012.

## 7 | CONCLUDING REMARK

We have come to regard all entities that algorithmic processes get hold of as *algorithmic signs*. (Nake, 2004) Following Charles Sanders Peirce, the sign is a first (the representamen) standing for a second (the object) by virtue of, or creating, a third (the interpretant). (Peirce, 1992) The interpretant is the result of an act of interpretation; that is interpreting ("making sense of") an encounter of a perceivable occurrence of a something (representing) and some other (represented). The interpretant is itself a sign so that Peirce's concept of sign and semiotic processes is in itself recursive. This makes it the most prominent concept of our times.

We have extended the Peircean concept of sign a bit by considering sign processes between computers and humans, between semiotic machines and semiotic animals. The first, the machine, is an interpreter that cannot really interpret, since its interpretation is not characterized as open according to contexts, but rather restricted to the only context of computability. Thus, the machine's interpretant is really a determinant: the result of the computable kind of interpretation that must determine the one and only one meaning that a piece of code can have if it is correct code.

The algorithmic sign thus, additionally to Peirce's sign, contains another component, the *determined interpretant*. It makes sense to call the old interpretant, generated or given by a human, the *intuitive* or *intentional interpretant*.

Instead of "algorithmic sign", we also use the metaphor of an inseparable union of *surface and subface*. (Nake, 2008) According to it, anything on a computer must be characterized as a pair of surface and subface. The surface is for the human to perceive by one or several of the senses; the subface is for the computer to manipulate by one or several algorithms. What is happening on a computer is a permanent switching between those two. But the human is usually aware of only the surface processes, the computer is only aware (if it could be "aware" of anything) of the subface processes.

A programmer, and any other person doing something involving periods of computation, engages in such complex processes. The programmer is preparing subfaces for other persons later perceiving surfaces. In contemporary processes of art education, teachers and students have a chance to better become aware of what is happening in the algorithmic dimension by preparing subfaces that will later generate surfaces of aesthetic qualities. This is new. It is essential. It can be done, and is a bridge between science and art.

## ENDNOTES

[1] This paper is a completely new version of our earlier paper Grabowski & Nake (2017).

[2] Georg Nees was the first to exhibit computer-generated and automatically drawn graphic works. This show was put up in rooms of the Studien-Galerie of the Stuttgart Institute of Technology (today: University of Stuttgart) from 5 to 19 February, 1965. For the occasion, Max Bense wrote his paper, "Projekte generativer Ästhetik" (Bense & Nees, 1965). Nees later became the first to do a doctoral thesis on computer art (Nees, 1969).

[3] The second exhibition of so-called computer art took place at the then famous avantgarde Howard Wise Gallery in New York City. It was dedicated to works by A. Michael Noll and Bela Julesz, both from Bell Laboratories in Murray Hill, NJ. It was on display from 6 to 24 April, 1965.

[4] The term "Processing" here refers to the programming system of Processing (Reas & Fry, 2014)

[5] The roots of a series of projects under the title of "compArt" reach back to the earliest experiments of algorithmic art in the mid-1960s.

[6] Figure 5 shows an early computer-generated picture by Manfred Mohr, and Figure 6 is taken from students engaged in analyses of Moohr's piece.

## REFERENCES

Bense, M., & Nees, G. (1965). Projekte generativer Ästhetik = rot 19. Stuttgart: Eigenverlag

Epler, M. (w.d.) http://mepler.com/CONTACT-BIO

Epler, M. (1973?) http://mepler.com/The-ReCode-Project

Grabowski, S. & Nake, F. (2017) Between the trivial and the impossible. ReCoding as learning strategy. 2017.xcoax.org/pdf/xcoax2017-Grabowski.pdf

Hertlein, G. (ed.) (1976-1978) Computer Graphics and Art. A magazine published quarterly in the years 1976 to 1978

LeWitt, S. (1967) Paragraphs on conceptual art. Artforum vol. 5, no. 10 (June 1967), pp. 79–83

Nake, F. (2004) Das algorithmische Zeichen und die Maschine. In Paul, H.J. & Latniak, E. (eds.). Perspektiven der Gestaltung von Arbeit und Technik. Festschrift für Peter Brödner. München: Rainer Hampp, 203-233

Nake, F. (2008) Surface, interface, subface. Three cases of interaction and one concept. In Seifert, U., Kim, J.H., & Moore, A. (eds.). Paradoxes of Interactivity. Perspectives for media theory, human-computer interaction, and artistic investigations. Bielefeld: transcript, 92-109

Nees, G. (1969) Generative Computergraphik. Berlin, München: Siemens

Noll, A.M. (1994) The beginnings of computer art in the United States: a memoir. Leonardo, 27 (1) 39-44

Peirce, C.S. (1992) The essential Peirce. Selected philosophical writings Vol. 1 (1867-1893). Ed. by Houser, N. & Kloesel, C.. Bloomington: Indiana University Press

Reas, C. & Fry, B. (2014) Processing. A Programming Handbook for Visual Designers and Artists. Cambridge, MA: MIT Press, 2nd ed.

## BIOGRAPHICAL INFORMATION

Frieder Nake is a professor at the University of Bremen's Informatik department specialising in computer graphics, computer art, digital media, semiotics, and theory of computing. Since 2005, he has also been teaching at the University of the Arts in Bremen. He is a pioneer of computer art who first exhibited his computer-generated drawings in 1965. He has his doctoral degree from the University of Stuttgart, in mathematics, and has taught at many other places, particularly at University of B.C. in Vancouver, Canada.

Susan Grabowski is media educator specialising in digital media and art education. Her Ph.D. thesis was on aspects of signs and space in algorithmic art. She was for many years a member of the compArt Center of Excellence at University of Bremen, where she took part in the development of the compArt database Digital Art.