Last month, the Carnegie Science Center in Pittsburgh opened a new robotics exhibition, which includes a recent version of my AARON program. AARON will be running continuously on a huge monitor for the next ten years. I brought a copy with me and if you've seen it running you've probably spotted how the program maintains continuity; adding new material in the foreground and periodically deleting the oldest material from the background. Since the new material obscures much of what's been left, each cycle results in a substantially new image.

On the faster machine in Pittsburgh, this cycle of adding and deleting takes about five minutes; so if it runs for ten years it'll have enough time to make about a million images.i Fortunately for the environment, they aren't being stored. A hundred thousand images a year; that surely must be a world record; not even the most prolific human producers could come close. So, if I believed that the creativity of a system -- human or otherwise –could be measured by the volume of original and distinctive output it could generate, then I would be claiming AARON to be the most creative system in history.

I don't believe it, of course, and I'm not making any such claim. But I do think that AARON is doing something remarkable. When I take a dozen of the hundreds of images it has already produced and hang them in a gallery, most viewers evidently find it difficult to grasp that the program is responsible for what they see, and conclude that the creativity is mine, not the program's. Knowingly or not, they're invoking the old familiar hack: "computers can only do what you tell them to do" which bypasses the all-important question of exactly what you can tell a computer to do.

In any case, I'm not going to be in Pittsburgh to tell AARON anything for the next ten years, just as I'm almost never around when it's generatng images in my studio. It generally happens at night, while I'm sleeping.

In short, credit assignment is not straightforward. The program is not uniquely responsible for its impressive output, and I am not directly responsible for any single image. But then, the inherent complexity of this situation is already reflected in the title of this seminar, isn't it? What do we understand by the term "computational creativity?" Do we intend it to address the creative status of various computer programs? In which case we would need to ask what exactly these programs are doing that is so different from what other programs do. Alternatively, do we mean it to address an exclusively human creativity of a kind that is advanced only through the medium of computational strategies? In that case we would need to ask why, and how, computers differ from other media.

Or does "computational creativity" imply some as-yet undefined amalgamation of human and non-human; not requiring the science-fictional theatrics of putting computers inside the brain or putting brains inside computers, but nevertheless constituting a new kind of entity; one in which the developing creativity of the one manifests itself in the superior performance – the creative performance? -- of the other.

I think my own history compels me to take that third position, though I've always thought of my goal as program autonomy rather than creativity. Maybe that's just putting a different label on the same package, but where I wouldn't have known how to proceed on the path to creativity, the path to autonomy has been clear. It has simply meant handing off to the program a steadily increasing range of responsibility for the decisions leading to the making of art

objects. That's the path I've been following since I wrote my first computer program forty years ago.

Today I want to focus on two episodes along that path, both critical to the program's development; and both breakthroughs in the sense of enabling something that hadn't seemed possible previously. But I'm less concerned here with how they work than with how they came to be And in describing the circumstances under which they occurred, I want to pay particular attention to the influence of two omnipresent negative forces; the law of inappropriate assumptions, which has us foundering in the wrong conceptual tangles. And the law of unintended consequences, which has us emerging from those conceptual tangles in unexpected places.

The first episode, beginning some twenty years ago, dealt with the problem of having my program function as a colorist, and it illustrates the influence of both laws very well. There are two parts to this story. I'd come to computing as a painter, lomg before there were color displays and printers, and the lack of color had been a problem for me from the very beginning. When, finally, it bubbled to the top of the stack as a problem in need of a solution, my assumption was that a program would have to go about coloring ln the same way that an expert human colorist goes about it. That was not an unreasonable assumption for someone who had spent two years as a guest scholar among the expert systems folk at Stanford. But it was inappropriate, and it kept me from formulating the problem clearly for two more frustrating years before I realised, quite suddenly, that I'd set myself a problem without a solution.

Let me explain. The methodology of human colorists rests upon two features; on the one hand a finely honed visual feedback system; on the other a near total absence of color imagination; and by imagination I mean the ability to form a stable internal representation of a color scheme. The result of this resource profile – for we are all both enabled and constrained by our resouces – is that the near-universal methodology for human colorists is a stepwise process of continuous adjustment. I doubt that any human colorist has ever imagined a detailed color scheme for a painting and then written down the instructions for someone else to do the actual work.

There was no way to build on that model. My program was exactly opposite to the human in regard to its resources. It had no visual system, and it did have an impeccable ability to form a stable internal representation of a color scheme, if only I could tell it how to organise that scheme in the first place.

That realisation, -- the dumping of an inappropriate assumption -- was enough to get me started on a rule-based version of AARON that could handle color on the screen pretty well. But then I was confronted by another assumption, reasonable enough for an artist who had exhibited throughout his professional life, and had moved on from showing the output to showing the process. I'd done that successfully with a series of home-made drawing machines when AARON was only drawing, ***<<< 1 >>>*** so why wouldn't I build a painting machine? That led into a tangle of engineering problems I was never able to resolve to my satisfaction, **<<< 2 >>>** and I emerged two years later to find audiences engaged with AARON as a robot; delighted when it dispensed its own color and washed out its own cups, but largely indifferent to the images it was making. That certainly was not where I'd expected to be.

I retired the painting machine and acquired one of the new wide format printers ***<<< 3 >>>*** that

were then coming onto the market, so finally I had a way of going directly from the program to its output. **<<< 4 >>>** The remaining orthodox assumption, that any expert program -- including an expert coloring program -- would necessarily be rule-based, served well for the next five years **<<< 5 >>>** But as the rule-base got longer and longer, and more and more detailed, the resultant tangle got increasingly intractable, until, finally, I found myself unable to proceed further. **<<< 6 >>>**

And there begins the interesting part of the story.

What I generally do when I find I'm stuck on a problem is to change something fundamental, so as to force a change of perspective and thus – hopefully -- a re-evaluation. My rule-base had generated colors as rgb triples, which is how most programming systems represent color; presumably because red green and blue are the primaries of all electronic displays. A much more intuitive approach, from a colorist's viewpoint, is to characterise colors by their hue, lightness and saturation **<<< 7 >>>** Lightness is by far the most important of the three, and it is very hard to predict or control through rgb specifications.

So, to get my change of perspective, I decided to rewrite the rule base to generate hls rather than rgb specifications, thinking that at very least it should make it easier to think about the visual properties of colors and how to manipulate them. But long before I'd finished the re-write, I awoke one morning with a fully formed plan in my head. It was a plan so simplistic, so off-the wall, that I thought I must surely have dreamed it. And yet... its unexpected appearance was so intriguing – and it was simple -- that I figured it wouldn't be much trouble to try it.

Half an hour later, it was clear that I was on to something.

Here is what it does. This is the current form, which is only superficially different from the original.

First, the program generates three sets of numbers, one each for a high-range, a middle range and a low range. **<<< 8 >>>** There are three randomly assigned numbers in each range – the original had seven -- and they will provide values for both lightness and saturation when the program devises a new color. These nine values are the only values that can be used for lightness and saturation in any one image.

Secondly, it generates a script that controls how those numbers are picked. Each element of the script comprises a two-character combination **<<< 9 >>>** referring to the three ranges; the first character denoting the lightness and the second the saturation, so there are nine combinations in all. For example, **<<< 10 >>>** HL would mean that the lightness should be one of the three numbers in the high range and the saturation should be one of the three in the low range.

And, finally, it assigns a percentage to each element in the script **<<< 11 >>>** that determines how frequently that element will be chosen when AARON generates a new color specification.

A script like this one, for example, says that 70% of the time the lightness of the color will be chosen from the high range and the saturation from the medium range.: 20% of the time lightness will be taken from the medium range and the saturation from the low range: and the remaining 10% of the time they'll both be taken from the high range.

As to hues: the earliest versions of the program using this strategy were more overtly representational than today's version, and hues were assigned on the basis of fami*ly* membership. All the leaves on a given tree would be the same hue, **<<< 12 >>>** for example; they would simply vary in lightness and saturation. The more recent versions of the program are more concerned with the relationship of individual hues to the background and to each other..

Now, if all the numbers in the three ranges are selected randomly, the lightness/saturation codes in the script are similarly random, and the chance of any one of them being chosen is determined only in terms of a randomly assigned frequency, why would you expect any useable color scheme to emerge other than by accident? In fact, it produced a very wide range of color effects, most of them perfectly useable. From the very first implementation – on that same morning – I started to see color relationships I'd never used myself and would never have tried to achieve if I'd been responsible. For example, one instance of the script generated **<<< 13 >>>** a range of delicate, silvery grays unlike anything generated by the previous, rule-based strategies, and unlike anything I myself mignt have done as a painter. **<<< 14 >>>** Another produced an almost completely black image.

Of course, not all useable color strategies are of interest to any particular artist, and very few of them were of use to me. **<<< 15 >>>** Consequently, rather than have the program generate a new script for each run, I settled on the small handful of scripts that consistently provided what for me were vivid, interesting results. **<<< 16>>>**

The more interesting thing, though, is that in the three years this algorithm has been in use, I've never been able to predict the outcome of any script by reading it – the way a musician might "hear" a melody by reading the score – I've never been able to invent a script myself to achieve a desired outcome; and I've never known how to modify an existing script, except in the most general way, to modify its performance,.

Notwithstanding all of which, I'm quite sure that it's the best idea I've had in the forty years of my own computational creativity; not only because it opened up a whole new range of achievement to the program itself, but also because it reveals an unexpected dimension in my relationship to the program.

On the one hand, even though I don't know how the formulation took shape, one can hardly doubt that the idea was formulated out of my own experience as a colorist – where else, after all, could it have come from? On the other hand, the terms of the formulation seem not to be human terms. No human colorist could effectively do what the program does, much less achieve a similar level of expert performance with it. Its success, and certainly its inception , clearly owed much to the recent adoption of the hls representation, but it's surely important that by the time this change was initiated, I'd been working on the program, pretty much every day for more than thirty years. I think it reasonable to infer that the formulation would never have been possible if I had not had as much intimacy with programming, indeed, with this program, as I had had as a painter with the paints I used.

On, now, to the second episode.

I delivered AARON to Pittsburgh at the end of May, and immediately started work on a new version, which incorporated an entirely new methodolgy. Five or six weeks is hardly long enough to reach any conclusions about how it will develop and how far reaching an effect it will have. But even at this stage it has required a re-evaluation of some fundamental assumptions.

I'll discuss the what and the how of this new methodology in a moment, but first I'll need to talk a bit about the why.

I've indicated that color has been the central issue in my work for the past twenty years, but I should have said, more precisely, that I've been focussed on the idea that color can be the central organising principle in an image. Of course, short of sitting inside a sphere of uniform color, color doesn't come without boundaries, and I've been obliged to pay a good deal of attention over the years to what those boundaries should be like. The problem, in a nutshell, is that visual cognition deals with the outside world predominantly in terms of the relative lightness of things, and it has relatively little interest in the color of those things. The corollary is that, in dealing with images, the viewer stops paying attention to the color of an element as soon as he recognises it to be a visual representation of something in the outside world. Color can certainly carry an important decorative role in representational painting, but for it to become the central organising principle, as it was for Bonnard – a rare case indeed -- color and representation are all but mutually exclusive.

You'll understand that I'm offering my own view here; I'm not aware of any color theory that supports it or denies it. Following from that view, I've wanted the forms in AARON's images to teeter on the edge of representation; always to retain the authenticity of the thing seen, while still evading their color-robbing identification as visual representations. If evocation is the correct word to describe what I've wanted, then I've spent countless hours hand-crafting procedures to generate classes of forms **<<< 17 >>>** that would satisfy my color-determined requirements; evoking leaves without ever representing a particular leaf, for example, evoking flowers without making direct reference to daisies or roses. **<<< 18 >>>**

For years I've been returning periodically to the problem of form generation, usually when I became bored with seeing what I could recognise – and possibly only I could recognise -- as the same generators at work over and over again. Perhaps it was working on the ten-year version of the program that led me, finally, to confront the obvious question.

Why couldn't AARON generate its own forms?

Well, let's see: from a conventional AI perspective, I could assume that I'd need to develop a working model of what made a collection of marks function as an evocator. And I'd have to encode that model so that the results would evoke classes of objects, without appearing to the viewer as merely formal transformations of identifiable prototypes.

I need hardly say that if I'd acted on those assumptions I wouldn't be talking now about rewriting much of the program from scratch in a single month. Fortunately, as had happened three years earlier with respect to color, I awoke one morning with a strong sense that I had a potential answer to the problem of autonomous form generation. But this time it wasn't an outline for a complete procedure, it was a single term, quite possibly something I'd been reading about in New Scientist the night before; self-assembly.

Self-assembly. What was I supposed to do with that?

Well, I thought, self-assembly implies some primitive element that follows its own local rules for assembling itself into more complex elements. Those assembly rules would have to determine how those elements joined an accumulating form, how many of them the resulting form could sustain, and so on. And if I wanted them to generate a population diverse enough to meet my pictorial requirements, then there would have to be different local assembly rules for different populations of primitve elements.

Keep it simple, I thought. The simplest space-filling element would be a triangular cell. The narrow end would be the point at which it would attach to other cells, and the other two points, fogether with the other end of the axis of symmetry, would be available attachments sites – points to which other cells could attach. In fact, the very first sketch of this scheme -- and the most primitive with respect to assembly rules -- **<<< 19 >>>** required only that attachment points fell within the body of any existing cell.

Once the requisite number of cells had assembled, **<<< 20 >>>** I would use one of AARON's oldest available tools to trace the outer boundary of the mass, making a containing polygon from the attachment points that fell on the boundary, **<<< 21 22 >>>** and then rendering the polygon with essentially the same "freehand" algorithm that I developed for the infant AARON nearly forty years ago.

That first sketch quickly demonstrated both the strength and the weaknesses of the approach. On the one hand, this simplest of assembly rules <<< 23 >>> generated a remarkably diverse output. On the other hand, while a large collection of abstract forms would be useful, at least some of them would need to be evocative elements -- elements that teeter on the edge of representation.

That kind of identification came with increasingly specialized assembly rules. For example, a cell can be constrained by its local rules to attach only to the most recently attached cell, with its direction determined by which attachment point gets selected. **<<< 24 25 26 27 >>>** These cells assemble into meandering, tendril-like forms that had been more math-intensive to generate previously. Modify the rule so that it attaches to an earlier cell **<<< 28 >>>** periodically and goes on from there, and it generates **<<< 29 30 31 32 33 >>>** a distinctively plant-like form. Increase the width of the cell in relation to its length, and the slender plants become cactus-like.**<<< 34 35 >>>**

In fact, what began as a plan for a self-controlling form generator quickly developed into a heirarchical structure: cells assemble into simple elements; simple elements assemble into more complex elements, and finally all elements assemble into colonies. **<<< 36 >>>** That is, each new element will gravitate to an area where there is already a number of its own kind, until the colony can support no more **<<< 37 >>>** and the new element goes off to find the least populated area of the image to start a new colony. (It's not easy to spot the colonies, since there is so much variation in form in any given colony). **<<< 38 >>>** Pictorial composition had never been part of the self-assembly plan, but the compositions that result are comparable to anything AARON had produced previously.

It would be difficult to miss a distinctly darwinian flavor in all this, though that wasn't the point

of the exercise and I haven't felt any compulsion to take darwinian theory too literally. There is, after all, a huge difference in the forces acting upon a population in a physical environment and those acting upon elements in a pictorial composition. Nor have I tried to follow the evolutionary aspects of the theory. In darwinian terms, one might say that the current version of AARON occupies an ecological niche; highly successful with respect to its output, and without the means, or any reason, to change.

"So what's new?", asks a puzzled little internal voice. "You could have said that about any version of AARON in the past couple of decades. They were all pretty successful with respect to their output, and none of them could have changed if you hadn't changed them."

True enough; but all the earlier versions had some range of knowledge embedded IN the program, and required the exercise of some level of intelligence BY the program. AARON could never modify its own behaviour, but pushing its autonomy up a notch has always meant either increasing its knowledge or amplifying its intelligence; usually both. This new version is purely algorithmic. It has no discernable knowledge base, and it doesn't have to make a single decision about how to proceed. All its rules are local; there is no grand plan. In short, and to the degree that it acts with neither intelligence nor purpose, it resembles a natural system more than an intelligent system.

And, in the sense that it functions like a natural system, it is no longer relatively autonomous, it's absolutely autonomous.

Well, that illustrates the law of unintended consequences with a vengeance. That isn't at all what I've understood by program autonomy. Program autonomy has been a measure of the level and range off responsibility borne by the program. What would absolute autonomy mean, in my sense: that the program has all of the responsibility? That surely wouldn't be possible unless the program had all of the knowledge and all of the intelligence. It wouldn't be possible unless the program had purpose. Apparently this new version of AARON has none of those things.

Having spent half my life trying to make AARON intelligent enough to take over much of what we know would take intelligence in a human agent, I was, I confess, deeply troubled by the sudden apparent irrelevance of intelligence.

But then I began to wonder whether my distress wasn't simply a response to the law of inappropriate assumptions coming into play once again. My assumptions about what a program should be like were formed long ago, as I said, by my stay among the expert systems people at Stanford. Their whole idea was that, not just the accumulated knowledge but also the intellectual methodology of human experts could be expressed in a computer program. Certainly, much value was placed upon the succinctness of the result, but since the people who wrote the code were, almost without exception, NOT the people who had the knowledge, how likely is it that they could even conceive of encapsulating all that knowledge into a simple algorithm having no apparent methodological connection to its source?

Similarly, how likely is it that AARON's coloring algorithm could have been invented by someone without considerable knowledge of color and considerable experience in manipulating that knowledge, as well as considerable experience in programming?

Not very likely. AARON's coloring algorithm allows it to function as an accomplished colorist.

**<<< 39 >>>** That has to mean either that this level of performance can occur by accident; that I somehow stumbled across a very unlikely strategy that just happened to work extremely well; or that the algorithm is, in fact, profoundly knowlege-based. **<<< 40 >>>** In that case, the remaining problem is to understand how the connection was made between a knowledge base accrued over half a lifetime and a few lines of code that were written in half an hour.

I have no solution to that problem, but it was not done, certainly, by trying to model how human colorists go about their business. In fact, we know very little about how human colorists go about their business, but we can be quite sure of one thing; no human colorist could do it the way AARON does it. One of the most interesting things about both of these new algorithms is how very un-human they are. At the risk of sounding fanciful, it is almost as if I had told the program what I wanted done and the program had told me how it could best go about doing it.

And that brings focus to what is, for me, the central issue of this talk. I'll allow that AARON is creative, with respect to both color and form; after all, one can't entirely discount a million original images. But that creativity, non-trivial as it may be, functions for me primarily as an existence proof that "cognitive creativity" is a property, rather, of the relationship between program and programmer; because it is only in the dialog that can develop in that relationship that the transition from human purpose to machine implementation becomes possible. Lacking that dialog, we are reduced to defining the machine as an imitation human being; a dubious undertaking, in my view, and one having little or no bearing on the issue of creativity.

This view implies , of course, that there are no short-cuts; that computational creativity requires the same accumulation of expertese that has always sustained human creativity, including, in this case, an adequate level of expertese in programming. It also implies that, while the future will no doubt bring computing machines orders of magnitude more powerful and more sophisticated than any we have today, they will not, by dint of their power alone, become autonomously creative. It seems very unlikely to me that they will achieve that state – if they ever do -- other than by the steadily accumulating gains made by the computational creativity manifested in the program/programmer dialog.

As for AARON, now that it has abruptly manifested absolute autonomy; my present task is to find a way back into the dialog. I'm too sensitized to the law of inappropriate assumptions to venture a guess as to how that may be accomplished, but if I wake up tomorrow morning with another unlikely idea in my head, I promise to let you know.