The Fallacy in Intelligent Design

by Lynn Andrew

We experience what God has designed, but we do not know how he did it. The fallacy is that the meaning of "intelligent design" depends on our own experience.

If creation is the result of intelligence, then "design" is implicit and loses its separate significance. "Intelligent design" is a semantic trick that links human experience in the temporal domain to the work of God in eternity.

One of its faults is that it leaves room for the mindless fantasy of non-intelligent design. It has the ill effect that when you point to evidence of "intelligent design" you concede the possibility of there being something else. Putting it in the negative is no improvement: by saying "complexity could not have evolved," you leave the same door open and worse; you pit evolution against non-evolution, lending credibility to the former and putting forth nothing to oppose it.

When I point to the eye as an example of the work of God, I should say, "some genius made it work," not, "this is the work of an intelligent designer." The difference is the word "design," which brings in the anthropomorphism of a drawn-up plan.

"Intelligent design" envisions the universe and everyone in it having been planned by some process that resembles the work of a human designer, only amplified astronomically. But what we know about the work of designing is really discovering and inventing things: finding what can be done and how it satisfies some need. Did God discover anything? We would rather say that he did his intelligent designing as quick as lightening without a single intermediate thought. Did he have some criterion by which his design was judged? It seems rather that Omniscience and Omnipotence would have hit the mark immediately without question. So if his methods are inscrutable, is it reasonable to link our perceptions of his

creations to our experience of what producing our "creations" entails?

The Bible speaks of God as Creator but never as Designer. Creation involves design, of course, but all we know about design is from the manner in which we humans design things for our own relatively insignificant "creations," all of which are works within Creation. When we speak of "evidence for intelligent design" we refer to what resides within the framework of Creation. Since time and space are included in Creation, our sequential thinking about design may have no analog in his work at all.

On the other hand we cannot be sure that it does not. We are made in the image of God. It could be his intention that we understand how he designs things, which is a very exciting possibility. If it is true that the way God designs things is echoed in the way we design things, then which of our many approaches to designing are we going to attempt to ascribe to him? Is he most like an architect designing a building, a city, or a chemical plant? Maybe he goes about it as would an executive designing an organization. How about a legislator crafting laws and regulations? Or does his work resemble that of a composer of music?

Designs are fascinating. They are purely information. You can make copies of a drawing or a chart or a musical score and you have not multiplied the design: there is still only the one. Information must be embodied in some medium, but the medium is not the information. Information has no physical mass and therefore it is not tied to time as we know time: it does not decay or change with time. The physical medium will deteriorate, but copies of the information can be preserved indefinitely for use by a suitable interpreter.

The concept of a timeless design has the ring of the divine, but the types of designs mentioned above leave a large gap between the information in the design and the object of the design. For example, to go from a set of architect's drawings to a completed building requires a great amount of intelligence, skill, and additional information that must be supplied by the builders. Therein lies a fallacy if we try to use one of these types of design to model what we are assuming is the design activi-

ty of the Creator: it does not seem fitting that God would need help to go from his design to his creation, even if he designed us to understand it.

A unique kind of design takes place in the development of computer software. There is no gap here: software, which is the end product, is itself information, and the distinction between design and construction is blurred. Yes, a suitable computer is required to interpret the software, but the "computer" which a software program is written for is typically another software program. So software is not necessarily dependent on particular hardware or even a particular type of hardware. In other words, software not only represents a design, it is the end product of the design.

Someone will tell you that software is basically a set of instructions that make a computer do something. This is only partly true. Computer programmers do not think of it that way. The programmer's view of the software is not in the same domain as the one seen by the computer: he does not write sequences of instructions, but rather methods and procedures for generating instructions.

A procedure might consist of a "loop," for example. The programmer sees the loop as a static entity. The computer experiences the results of perhaps a few million executions of the loop in a very short period of time, each execution issuing different instructions. A very simple program can blast out a complex galaxy of the elements of whatever it is that it is doing without even needing to know what it is doing—because it is a general procedure that can be used in many different applications.

A smart programmer will make methods and procedures that are rather general in order that they may be used over and over in various applications simply by changing parameter data, or better yet by swapping certain procedures at lower levels that get used by the procedure being written. As in the example of the "loop" procedure, a procedure may accept other procedures and not know or care what the others are doing. Procedures can automatically "write" other procedures: often a procedure will generate or modify procedures so that it can address a whole range of applications. A procedure may change its function entirely, either by having its mode of operation depend on external conditions or by being re-

placed by another procedure. Some procedures are designed to execute themselves—and yet there is only one of them. And the order in which such things happens usually depends on other things which are unknown and unknowable to the programmer—who does not need to know, does not want to know, and frankly does not care.

Although software is pure information and in its source form is a static, finished creation of the programmer, it becomes very dynamic in use as it unfolds in time—or rather is explored by time. It seems to constantly change itself, adapting to new conditions, but seldom does things perfectly. Internal checks to correct errors, cleanup operations to get rid of accumulated "garbage" data, and embedded defense mechanisms against unfriendly interference are all involved. The programmer cannot tell you much about the time sequence in which these things occur even though he may be the sole creator of the software because his domain is separate.

Then there are what we may refer to as processes, which are like software programs within software programs that run (effectively) concurrently. Hundreds of these may be active at the same time in one computer—all part of what goes on in real time without the programmer's knowledge yet all of it the result of his creation.

It sounds overwhelmingly complex, but from the programmer's point of view it is not as complex as it sounds because he sees methods and procedures, not the embodiment of them in real time except on rare occasions.

At the programmer's disposal may be a host of procedures developed on other projects. In fact programmers strive to make their procedures reusable. So the programmer's work is largely a matter of stitching together procedures and functions that he developed earlier or that someone else provided. Most likely there will be vestiges of other applications.

Suppose you asked me to write an app to simulate an aquarium. I would consider doing it if you gave me the freedom to make fanciful aquatic creatures. First I would find or write a procedure that moves an image across the screen. Then to avoid the tiresome task of drawing images of fish, I would write a general procedure capable of generating any

three-dimensional body, and into it I would plug constraints to force it to make bodies having symmetry, a head, a tail, and fins. This procedure would call on other procedures to determine the sizes and distribution of features. To make life easier for myself and to make the final product engrossing even to myself, I would not expend any effort in designing specific arrangements of features; rather I would hook in a random number generator and pass the results through a filter set to eliminate arrangements that are structurally impractical. A similar set of procedures would provide the colors and markings on the envelopes of the bodies. If I found enough commonality between the shape generator and the coloration procedure, I might share some functions or even make a more general procedure that would emit both procedures. And so on. As soon as the system is far enough along to produce the simplest thing on the screen, I would try it out and then keep trying it out with each new advancement, using the visual evaluation to guide the next step.

If you had insisted that the program replicate a real aquarium or something that has already been done, I would decline, knowing that it would take too long and be no fun at all. I would suggest that you go take some photos and make yourself a slide show. But if you like my unique little aquarium and would like me to do another project that puts a parade of dogs on the screen, I would jump at the chance, for nothing is more fun than this type of programming, and most of what I need for the dogs I have already developed for the fish.

One is tempted to run with this (as some do) and theorize that the physical laws underlying Creation and the mathematics underlying the physical laws are characteristics of celestial computer software, and the quantum level is evidence of its digital "hardware," and every physical thing is a simulation just as a video game simulates "reality" that has no physical existence. That would mean our Creator is a Programmer, we are the outputs of his software, and everything is information, which is itself outside of the "physical" universe. But this is off my subject.

The worst thing about employing the term "intelligent design" when attempting to refute evolution is that all human designers use evolution.

Evolution is a wonderfully powerful tool in the hands of a designer. We speak of "software development" rather than "software design" because evolution is an essential part of the process, which is very much like the work of an inventor: lots of incremental trial-and error (where trials are intelligently chosen and the results are intelligently judged). So another danger in ascribing "intelligent design" to the Creator is that when the connection is made to the human intelligent designer, evolution comes along with it. Of course this evolution is different from what the atheist dreams of; it is a method of intelligence and has no intelligence in itself. But it definitely fits the definition of an evolutionary process, and it is the only kind that works.

But I would not think that God would need to use even this type of evolution. So I think we should stop referring to his work as "intelligent design" and admit that we know nothing at all.

Intelligent designers keep experimenting and pushing the evolution of software design toward something worthy of the AI label. If they succeed to the point where a machine has enough intelligence to produce works of intelligence as useful as those from a human designer, the term "intelligent design" will have lost whatever edge it had in the hands of creationists. No doubt dozens of science-fiction writers have already sketched make-believe worlds where machines have the intelligence to produce machines as intelligent as themselves. If I were to write such a story, I would emphasize the obvious: that this cutting edge of evolution has made no progress at all since the self-generating machine is an exceedingly clumsy imitation of the elegant reproductive ability that came with every living organism in Creation. A good novelist would put an ironic twist on it—for example having natural worms ruin the factory these robots have built to reproduce their kind.

Not being much interested in literature that makes no reference to the Bible or neglects the Gospel, I would write a happier ending. To my way of thinking the logical response to the uninspired triumph of atheism aiming to duplicate the miracle of creation would be laughter in heaven.

†