Computer Science for Ninth and Tenth Grades

Charles Weems

In *A Modern Art of Education*, Rudolf Steiner says, "People are surrounded by inventions of the human mind, but have no contact at all with them. It is the beginning of an antisocial life simply to accept inventions of the human mind without at least understanding them in a general way. … At the age of fourteen or fifteen, we must focus our energy on connecting children with the inventions of the human mind. This helps them understand and find their place in society."[1] Digital technology is perhaps the most characteristic and multifaceted invention of the human mind in the present era. It is overturning social norms at a rapid pace. Thus, Waldorf high schools owe it to their students to implement developmentally based instruction in computer science (CS) that is suited to all students in all grades, not just elective courses for those who are predisposed to have an interest in the subject.

Waldorf high schools owe it to their students to implement developmentally based instruction in computer science.

Traditional high school CS instruction takes its shape from introductory college courses. There are two CS advanced placement (AP) exams, one of which matches a typical introduction to a programming course that uses the Java language. The other, called CS Principles, is a non-majors overview of the field. These AP exams follow the principle that the earlier a subject is taught, the faster students will progress to more advanced subjects.

As Waldorf teachers are well aware, however, this is not the best practice for truly effective teaching. Rather, it is better to work with students in ways that meet their current developmental stage. Thus, in each year of high school, we have an opportunity to create different encounters with technology that build upon one another, while leaving concepts open so they can grow.

There are many ways to achieve this goal. Various authors have addressed approaches to teaching CS in Waldorf schools over more than two decades.[2–7] Some begin from the traditional questions of what, how, why, and who that we associate with the four high school grades, while others seek to address the thinking, feeling, and willing aspects of the human being. Jamie York's survey,[3] as well as my own research into online descriptions of curricula, has also found that some schools are more focused on computer skills (typing, word processing, digital photography, etc.). While some place CS in a main lesson, others schedule it into track classes with varying amounts of time allocated. It may be a requirement in all four grades, an elective in one grade, or a mixture.

In addition to pedagogical guidance for teaching about technology, Steiner gave his perspectives on its spiritual aspects in different contexts: for example, its relationship to art and culture,[8] the nature of elemental spirits associated with it,[9] and the nature of number, weight, and measure.[10] He describes how technology draws us more deeply into materialism, and although this is a necessary step for human evolution, the being that inhabits technology makes it more difficult for us to connect with the spiritual path leading upward. Seemingly in anticipation of modern experience, he says that when people come together through technology, it leads to conflict and destruction. Steiner makes it clear that technology poses a

threat to our humanness if we don't consciously take up our relationship to it and develop a mindset and practices that help us to keep it in proper perspective.

As digital technology advances rapidly, it is good to periodically revisit our approaches to teaching computer science. It is also worth noting that other parts of the curriculum, especially the humanities, present opportunities to better support the students in finding their way through the fog of technology and toward a path of evolution that leads humanity upward. If instructors who work with those subjects will take the time to immerse themselves in the digital landscape, they will find obvious connections that can enliven their efforts and make them even more relevant to the students of today.

Steiner makes it clear that technology poses a threat to our humanness if we don't consciously take up our relationship to it.

Any curriculum design should begin from consideration of the students (we teach students, after all, not subjects). What is it that we wish to achieve for them? What can we give them of lasting significance for their lives? What do we try to help them develop inwardly as human beings through the encounters and experiences we provide?

Technology offers tremendous benefits to society, but it also comes with many pitfalls. The goal of a CS curriculum should be to help students find a healthy, moral relationship to this artifact, so that they can use it to good effect in the world and not fall prey to its darker side. In this way they gain an understanding that permits them to see it as it truly is, in its different guises and manifestations.

Course content for a Waldorf high school class should also arise from the teacher's expertise, in combination with an understanding of anthroposophy. While we can take inspiration from the work of other teachers, we must be careful to avoid simply replicating their lessons. For the material to live in the students, it must first live in us. And we must constantly evaluate our own efforts and refine them, especially in a subject that is changing from year to year.

Thus, the examples given here should not be seen as prescriptions. They are a work in progress and are shared at this point merely to stimulate creative impulses.

For context, I am a professor of computer science—with forty years of college teaching and fifteen years of Waldorf high school teaching experience—and co-author of twenty-eight introductory textbooks; I have designed and built advanced computers from the design of custom chips to the construction of working systems. The classes I teach in each grade are offered in two forty-five minute periods per week for a trimester that varies from nine to thirteen weeks in length, depending on the school calendar.

A common theme among prior efforts has been the revelation of the inner working of the computer: for example, building an adder from electromechanical relays. It is indeed important for students to develop an understanding of how the hardware works. Fundamentally, computers are very simple machines made from switches that turn on and off, just like those that control lights or ring doorbells. If the switches are arranged in a fairly simple manner, they can be made to operate in a pattern that mimics the pattern of 1s and 0s that appear in a base two addition table. Experiencing this reduces the magical, vaunted machine to something that can be grasped by mere mortals.

Early on, I tried the relay adder lesson in tenth grade. However, relays were hard for the students to understand. They were more focused on the novelty of the device than on what the circuit was doing. It was a leap to grasp the interconnectedness of the relays. Yet, I wanted this "how" aspect of computer science to precede the Electricity and Magnetism block that follows in eleventh grade.

Relays illustrate a particular aspect of digital switches: namely, that the circuit controlled by one switch can in turn flip another switch. While this is an essential element in the automatic operation of a computer, it is not a pattern that most people encounter in ordinary life. People press buttons. Buttons don't press each other.

Thus, it is necessary to temporarily set aside that aspect of digital logic and have the students do all of the button pressing themselves. They are given boards that hold batteries, a light, and doorbell buttons that are either wired in parallel or series (logical OR and AND), or a normally closed button (NOT). Each student is responsible for pressing one button. They are told to do so either in response to an input value, or to a light on another board. With two AND boards, an OR board, and a NOT board, they are able to mimic the operation of an exclusive OR (using OR to mean you can have ice cream "or" cake, but not both). Exclusive OR is otherwise known as a half-adder because it has the same pattern as adding two bits of a binary number, if there is no carry from the place to the right.

The goal of a CS curriculum should be to help students find a healthy, moral relationship to this artifact so they can use it to good effect in the world.

With this exercise, it is possible to convince students that switches can mimic arithmetic. They can then be shown a spring-loaded electro-magnetic solenoid that moves a plunger back and forth as it is turned on and off. It is easy to imagine its power leads being connected in place of a light on one of the boards, with the plunger positioned to push a button on another board, thereby replacing the student's eyes and fingers so that the whole thing becomes automatic.

From here, I tried introducing breadboards with small-scale logic chips, so that they could wire up an adder that also accepts a carry from the right. But it was too great a step to go from doorbell buttons and solenoids to imagining the equivalent arrangement hidden inside a black plastic rectangle with protruding wires. Learning to use the breadboard itself was also a distraction.

Strangely enough, the bridge between the two was a logic design program called Logisim.[11] In trying it, I was concerned that wiring together gate symbols on a screen would be too much of an abstraction. But today's students are very comfortable with seeing icons on a screen as representations of concrete objects. They quickly grasp the use of the program and enthusiastically start wiring up and testing circuits. Some even come in the next day having invented more sophisticated circuits at home.

Once the adder is functioning, a simple cut and paste operation enables them to wire together multiple copies, resulting in a multiple digit adder. The students feel a sense of mastery in seeing their creation solving binary addition problems that they had only recently found challenging. With this experience behind them, it is much easier to envision the gates from their simulation residing inside the chips on the breadboard, and then wire them in the same way.

Teams of two build and test their adders, then connect them to other teams' adders, gradually extending the chain as more teams join in. There is considerable excitement as they flip the input switches and see their construction adding binary numbers. We then go back to the logic simulator and see how a memory circuit functions before assembling it on the breadboards.

Seeing these different circuits, the students are impressed with the number of switches that are involved, which is part of what gives the computer its power and basic character: simple arrangements of switches, replicated many, many times. We then reflect on the number of doorbell buttons in each gate and calculate the number of switches that would be present in the

register memory of a typical 64-bit computer. The numbers become staggering.

Frequently, the experience of calculating the number of switches prompts a student to question how they are actually built. With some wafers and bare chips, as well as some photographic negatives to lend a concrete support, it is possible to illustrate the photolithography process used in making chips. While the functioning of transistors is still beyond their understanding of electricity, they can see how it is possible to form wires in different layers to interconnect the switches.

The process of painting on a material, coating it with a photosensitive layer, exposing and developing it, then using a chemical to dissolve away unexposed areas is conceptually simple. Hearing about the precision and cleanliness involved helps them to appreciate the effort that goes into making the circuitry in their computers and phones.

By the end of tenth-grade computer science, students have the sense that nothing in the computer is beyond their understanding if they choose to devote more time to studying it.

As other authors have noted, disassembling old computers in tenth grade seems to be a natural match for this age. Learning to use a variety of tools, the students gleefully attack these formerly expensive machines. As the components come out, they ask what they are and how they work, which provides many teachable moments. At the end of this process, I have them sort the pieces on separate tables: one for the processors, another for the memory, a third for the disk drives and other peripherals, and a fourth for the supporting hardware (power supplies, fans, cases, motherboards). Students are always surprised to see that the actual processor is a tiny fraction of the whole, and that much of what we think of as a computer is only there to enable us to interact with it.

By the end of tenth-grade computer science, students have the sense that nothing in the computer is beyond their understanding if they choose to devote more time to studying it. They also appreciate that computer technology is the result of decades of work by many people, building upon others' efforts, and that it is amazingly inexpensive, considering what it takes to manufacture it.

If time permits, I have the students program a microcontroller to turn on a light, using assembly language. I give them the code, which is just a few lines long, and explain what the instructions do. The point is not to learn to program, but to see that when a program moves a binary 1 into a particular memory location, it is turning on a circuit that illuminates a light, just as our doorbell buttons do. This comparison serves to connect the very abstract idea of a program to the concrete actions of switches.

Also in tenth grade, we hold a separate computer skills class in which we look at how to distinguish reliable from unreliable sources of information, as well as covering the basics of word processing, spreadsheets, and presentation software.

Much of what is covered in tenth grade depends on the foundation that was built in the ninth grade class. As others have previously documented, the "what" question can be addressed by defining the word "computer" and by demonstrating how binary numbers meet the polar-reasoning aspect of this stage of development. However, there are deeper aspects to the nature of digital technology that also lend themselves to this kind of reasoning, making it possible to examine more carefully what a computer is.

Before we attempt a definition, we explore what computers can and cannot do. The first list tends to be quite lengthy, and it is more challenging to find tasks that the computer cannot accomplish. Out of the discussion it emerges that many of these tasks—having emotions, thinking, reproducing—are essential qualities of being human, and this insight sets a

tone for the class: Humans and computers are very different. Our goal is to experience just how different they really are.

In defining a computer, it can be described as being digital, but most students haven't considered what that means. For this reason, we explore the nature of numbers as being concepts we represent symbolically so that we can communicate them. The computer uses another representation, which is still not a concept. Only humans hold numbers as concepts.

Computers, however, represent a particular kind of number, called *discrete numbers*, which need to be distinguished from *continuous numbers*, which is how human beings can think about them. For phenomena of the world to be represented in the computer, they need to be divided into discrete units that can be measured, weighed, or otherwise converted into discrete numbers. Often there may need to be a preparatory step of converting the phenomena into an electrical analog prior to being split up.

Thus, we find the ultimate polarity between humans and computers: We are alive and they are dead.

For example, variations in air pressure associated with a sound are transformed into an electrical signal by a microphone. The signal is then divided into time units, and its voltage within each unit is measured to produce an integer value within a given range. The image formed by a lens falls on a material that generates electricity in proportion to the light it receives during the period the camera shutter is open. The material is divided into squares, and the charge in each square is measured to give a corresponding integer for that patch of the image. Zooming in on a digital image always reaches a point where the squares are visible, unlike what happens when a human takes a closer look at an object.

As students experience the process of digitizing different phenomena, they gain a deeper sense of how computer representations differ from human experience. They also see the significance of numbers in the computer, which then sets the stage for an introduction to binary arithmetic. In teaching binary arithmetic, it should be kept in mind that most students will never make practical use of it. Thus, the reason for learning it is to experience how simple it is. The challenge is not to understand the operations, but to let go of the mental habits of decimal notation. We have to return to the basic questions: What are digits? What is counting? What is addition?

With just a little practice, students can see that it is possible to change decimal numbers into binary and vice versa. The binary addition table (which has just four entries) is much smaller than the decimal table. Tens complement subtraction is also simpler than our modern algorithm, although it takes some practice to get accustomed to it. In binary, it becomes a trivial extension to addition. The binary multiplication table is also much easier to learn, and multiplication is seen to be another minor variation of addition. Students thus see that computer arithmetic has minimal rules that are applied mechanically, without thinking.

In tenth grade, the students implement the addition table using the boards with doorbell buttons. The emphasis is not on understanding the operation of the circuit, but rather on simply experiencing that a simple set of switches can mimic arithmetic. Thus, a computer is seen as a device built from switches that flip in patterns that correspond to doing arithmetic, which is vastly different from how we think.

The last remaining piece that defines a computer is programming. We use the old Bell Labs CARDIAC, a cardboard simulation of a computer, to visualize the major components of a computer.[12] Unlike Setzer's paper computer,[13] which involves active role-play, the CARDIAC has a flowchart and sliding tabs that lead each student through the process of fetching and executing instructions in a program.

The goal is not to teach students how to program, or how the computer works, but rather to give them a sense of what it is like for a computer to execute a program. It is an intentionally tedious and mind-numbing exercise in which students see how little is actually done by each step in the program. Computers are able to do what seems to be so impressive because they can carry out billions of these tiny steps each second. The challenge of the exercise is to resist the urge to think, and instead to mindlessly follow the instructions. This exercise drives home the point that computers do not think.

At the end of the course, students are asked to summarize what a computer is by writing a first-person essay in which they pretend to be a computer explaining itself. They draw upon their understanding of discrete versus continuous, digitization, binary arithmetic, switches, and computer operation to creatively describe computer existence. Once they have completed this assignment, we have a discussion of what it would be like for a computer to be instructed that it was about to be rebooted and turned back into a "normal" computer.

Inevitably, the conclusion is that it would be like awaiting death. Thus, we find the ultimate polarity between humans and computers. We are alive and they are dead. By ninth grade, students have already been exposed to fiction, games, and marketing in which computers are made to seem lifelike. It is thus important to dispel that notion so they can see them for what they truly are: inventions of the human mind.

With that as a foundation, computers become objects for study. We can learn how they work, use them as tools, understand why they are unreliable or vulnerable to hacking, and eventually explore who we are in the context of a society that depends on them.

ENDNOTES

1. Rudolf Steiner (2004). *A Modern Art of Education*, Great Barrington, MA: Anthroposophic Press, p.158.
2. William Steffen, "An Information and Communication Technology Curriculum for Steiner/Waldorf Schools," *Waldorf Journal Project #19*, AWSNA, April 2012, pp.28–32.
3. Jamie York, "Results from a Survey on Computer Curricula in U.S. Waldorf Schools," *Waldorf Science Newsletter*, Vol. 13, No. 23, Spring 2007, pp.43–59.
4. Bryan Whittle (2003). *Computing Curriculum Suggestions for a Waldorf School*, Research Institute for Waldorf Education, 32 pp.
5. Valdemar W. Setzer, and Lowell Monke (2001). "An alternative view on why, when, and how computers should be used in education," https://www.ime.usp.br/~vwsetzer/comp-in-educ.html.
6. David Mitchell and Andrew Linnell, "Thoughts on a prototype computer program for Waldorf high schools," *Waldorf Science Newsletter*, Vol. 4, No. 7, AWSNA, Autumn 1997, pp.14–15.
7. Thomas Schmidt, "Computer science and computers in the Waldorf school: Suggestions for the technology curriculum," *Waldorf Science Newsletter*, Vol. 3, No. 5, AWSNA, Autumn 1996, pp.13–19.
8. Rudolf Steiner, "Technology and art: Their bearing on modern culture," Dornach, Switzerland, Dec. 28, 1914, GA 275, S-2994.
9. Rudolf Steiner, *The Fall of the Spirits of Darkness*, Lecture 4: "The elemental spirits of birth and death," Dornach, Switzerland, Oct. 6, 1917, S-3405.
10. Rudolf Steiner, *Anthroposophical Leading Thoughts*, (XV): "Heavenly history - mythological history - earthly history. The mystery of Golgotha," December 25, 1924.
11. http://www.cburch.com/logisim/
12. David Hagelbarger and Saul Fingerman (1968). *An Instruction Manual for CARDIAC: A Cardboard Illustrative Aid to Computation*, Bell Telephone Laboratories, 1968.
13. Valdemar W. Setzer, "The 'paper computer'—A pedagogical activity for the introduction of basic concepts of computers," https://www.ime.usp.br/~vwsetzer/paper-comp.html

Charles C. (Chip) Weems, *PhD, is a professor of Computer Science at the University of Massachusetts, Amherst, and teaches computer science, geology, meteorology, and astronomy at Hartsbrook High School, in Hadley MA. He is a member of the Pedagogical Section of the Anthroposophical Society, a co-author of 28 introductory computer science textbooks, and has led workshops at national AWSNA conferences on technology as it relates to the Waldorf curriculum.*