Of Students, Computers and Learning

Barry McGuire

Barry McGuire is a physics teacher and science department head at Western Canada High School, Calgary. McGuire holds a B.Sc. and B.Ed. from the University of Saskatchewan, and a M.Sc. from the University of Calgary. He has contributed articles to Alberta Science Education Journal *published by the ATA Science Council.*

In the mid-1970s, like many other teachers, I became excited about the educational possibilities of the then-new microcomputer technology. This interest led, in 1979, to an Innovative Learning Project designed to explore the uses of the microcomputer in a high school physics class. In 1984, I designed a curriculum for a locally approved course called Scientific Studies and Computing. The groundwork for the curriculum was taken from those aspects of the Innovative Learning Project that offered the most interesting and constructive learning experiences.

Scientific Studies and Computing is, in all senses of the words, a science course. All of the objectives defined in the curriculum emphasize the nature, the knowledge and the processes of science. However, the curriculum displays one major difference from the regular science course: students in this course become the teachers; their pupils are the computers.

To fulfill the requirements of the course, students complete two science projects. The student chooses a topic from science and develops a computer application for science within that topic. Since the student is placed in one long problem solving situation, the meta-lessons become the most important learning experiences.

After selecting a topic, the student prepares a project proposal. The proposal outlines the science content that the student expects to learn. The proposal also outlines the nature of the computer involvement in the project. Upon approval, students select the type and sequence of activities necessary to complete the project. My role is mainly that of a resource person who ensures that the students are on task while in the classroom. Neither task is particularly easy.

First, students often choose topics outside my range of expertise. In such cases, if the problem is complicated, I may do quite a bit of background reading. Second, what appears to be an unproductive approach on the surface may have several extremely beneficial long-term effects. If students give the impression that there is some design behind their activity, then I usually let them pursue it. On the other hand, students often perceive that the six to ten weeks allowed for a project is a long time and that to catch up on a little homework during class would not be a serious misuse of time. I do not permit this.

Initially, I was quite uncomfortable with the structure, or lack thereof, and to some extent I remain uneasy. Subconsciously, I want the students to be involved in activities that produce tangible results of their efforts everyday. Consciously, I realize that problem solving techniques evolve from a wide variety of behaviors. Therefore, I refrain from contributing when it is not essential. Indeed, students often request help, not because they are unable to solve a problem, but because they want reassurance that they are on the right track. In such cases, it would be easy to steer them in the direction I think their project should go. I try to give them the courage to proceed, although there is some doubt as to the outcome.

However, not all activities are unstructured. Students must have taken a 20-level science course as a prerequisite to ensure that they have some science background. During the first weeks of class, I establish the groundwork for science. Most students are rather naive when it comes to this activity. I first establish the nature of science by discussing such

activities as hypothesizing, interpreting, classifying, analyzing and problem solving.

Second, I introduce students to the Apple IIe and computer programming. While experience with computers and programming is recommended, it is not compulsory. To ensure that students have the fundamentals of BASIC programming, I give short programming assignments at the time that I discuss several of the science processes. For example, when discussing data and data analysis, I give students a parallel programming problem. This program requests the user to "input" several pieces of parametric data, find the mean of this data, find the standard error and display the results in a prespecified fashion on the video display terminal.

Obviously, students will have a wide range of programming experiences. Problems, such as the one discussed earlier, will challenge some students and be extremely simple for others. Peer tutoring is encouraged and students regularly consult each other on programming techniques. Mini-lessons on programming (which most students require), are presented throughout the course. Books on programming are available in class, and students quickly become familiar with the resources and devise solutions to most of the programming tasks specific to their projects.

Selecting a project topic is very difficult (anyone who ever selected a thesis topic can relate to this task). Moreover, students inevitably perceive their project as a program that will teach the user all the neat information they have accumulated in preparing their project, a sort of computer assisted instructional program (CAI). On the contrary, their task is to create an application or utility program, a program, in other words, that makes the computer a useful tool to a scientist in the area of science from which the project originates.

The application of computers as scientific tools is a difficult concept for students to grasp. The most effective way to teach this concept is to use past projects that demonstrate several successes and failures. Even so, continued reinforcement is required to remind students that the nature of the computer application decides the ultimate validity of their projects. A computer application that attempts to do interesting things in science may have several programming flaws and yet be viewed much more favorably than a slickly programmed application that has a less scientifically valid application.

Here are some examples. The first time the course was offered, a student with considerable programming experience who was quite fluent in BASIC wanted to carry out a titration simulation. He wrote a good proposal describing a simulation of titration using graphics and a considerable amount of user interaction. Several discussions throughout the development of his project suggested that he was going not in the direction of his original proposal but more in the direction of a tutorial. When the project was presented, most of the graphics were found in a beautifully prepared title screen. The main routines of the program were merely a titration calculation sequence and a problem generation sequence. User interaction was limited to entering data to complete the calculations for the pH value of the unknown acid or base.

What of this student's learning objectives? Since he never studied titration, he certainly had to extend the base of his scientific knowledge to carry out the project. The output to the computer screen was extremely well-designed and the data handling routines were excellent, but the scientific application of his program was weak. The project is typical of programs in which students misperceive the nature of computer applications in science.

Another student in the same class, however, completed the best project to date. The student undertaking the project had virtually no computer experience but possessed a great love of astronomy. His previous scientific experience was limited to project work and astronomy projects in the science fair. All of the programming skills used in his first project were picked up in developing his project. What made this project succeed? First, the student formulated a hypothesis that he wanted to test. Thus, he had a clear image of the science involved in his project. Second, there was a definite role for the computer to search the data for relationships verifying his hypothesis. Third, he had a clear image of what the final product should do.

The essence of his hypothesis was that the periodicity of the fluctuations in size of red giant carbon stars was related to the periodicity of the fluctuations in the intensity of the light given off in certain areas of their spectrum. Data for the periods of fluctuation in the size and light intensity in each area of the star's spectrum for about 100 red giant carbon stars were entered into a data file. The program grew as the student's programming experience increased.

First, he programmed the search sequence to look through the data to find stars that gave light in the regions of the spectrum specified by the user. Then the program plotted the period of the size of the star against the period of the light. If the graphed data resulted in an approximately straight line, a relating

constant was calculated. In this way, he could explore relationships not only between the periods of the star's fluctuations in size versus light intensity but also between the periods of the intensities of various areas of the spectrum. Many of the programming techniques were quite sloppy, mostly due to the inexperience of the programer, but the science of it was quite exquisite.

Another excellent program was carried out by a student who bred dogs as a hobby. The student was quite interested in the genetics of sex-linked characteristics. She proposed to develop a program that could track sex-linked characteristics throughout a breeding sequence of five generations. She had absolutely no previous computer experience and several times got stuck trying to debug quite convoluted programming sequences. In the end, she learned about programming as well as about preplanning problem solving approaches.

The program enabled the user to specify the characteristic that was sex-linked to specify the genetic structure on which the characteristic was found. Then the user could choose the genotype of the male and female. The computer generated the genotypes of the offspring and allowed the user to specify which offspring was to be bred for the next generation and to specify the genotype of the new breeding partner. The computer repeated the process until all five generations were traced and then presented a summary of the breeding sequence.

What students found most interesting during the in-class presentation was when the program traced the incidence of hemophilia in the royal families of Britain, Germany and Russia in the late 19th and early 20th centuries.

In some cases, previous programming seemed to be a drawback. Many students with programming experience viewed the class as a computer class rather than a science class. As a result of this confusion, several students dropped the class.

The best programmer in the class almost dropped the course. The student could program more proficiently in assembly language than most students could in a higher level language. What kept this student from dropping the class was a discussion about the nature of gravity. We discussed the book *Flatland*, and the possibility that gravity could be a distortion of our three-dimensional world into the fourth-dimension. The following day, the student presented a proposal for his project. He wanted to design a program that would allow the user to enter the data points (ordered triplets) for a "wire frame" diagram of an object in the third-dimension. The program would then permit the user to rotate the object to any view in that space and then translate that view into one from the second-dimension or extrapolate it to a higher-order space. For example, if a cube was entered in the third-dimension, it could be rotated about any of the three orthogonal axes. Then the cube could be viewed as it would appear to a person living in the fourth-dimension.

Two factors almost stopped me from approving his proposal. First, it lay on the fringes of science and really was a project in pure mathematics. Second, and more critically, the project was very complex. However, since the proposal was so clearly presented, I decided to allow it.

The student immediately plunged into researching the mathematics of drawing three-dimensional projections on a two-dimensional space—the display screen. (The techniques for programming this are found in BASIC.) The key is the matrix; a cube, for example, is a three-by-eight matrix containing the ordered triplets for its corners. To make the cube undergo a realistic rotation, a matrix multiplication with another matrix (when the trigonometric functions occupy the cells rather then numerical data), was required. Since this program ran so slowly in BASIC, the student translated the entire program to machine language. For weeks on end, he was immersed either in books on matrix mathematics or in books on spatial projections.

The next problem was that of dimensional translation; it proved much more difficult than he first imagined. In the case of matrix mathematics and spatial projections, all previous analysis was by mathematicians. All the student had to do was figure out what they were talking about and translate it into computer language. Because he had to create the mathematics before he could begin programming, he had limited success. Nonetheless, during the four months he spent on the project he did learn an immense amount not only about mathematics but also about problem solving.

His presentation included the rotation of a wire frame diagram of a simple car. However, the extrapolation of the car into the fourth-dimension did not succeed entirely: portions disappeared in the translation because the matrix generated by the mathematical operations to translate even a simple solid from the third- to the fourth-dimension was so large that it required more memory than the computer had available. Therefore, much of the data generated by the program was lost. As a footnote, about a year

and a half later I received a disk and the documentation for the completed version of this programming that he was preparing to market.

Students with sufficient programming background can undertake projects that involve interfacing the computer with laboratory equipment. The gameport on the Apple is fairly simple to access from any type of program. Several sensing devices are available and quite easy to use. Phototransistors, photoresistors, thermistors and transistors can be connected to the gameport, thus providing a relatively safe (for the computer) and easy (for the student) connection between the computer and the outside world.

One student's initial proposal was to develop an interface so that the computer, in conjunction with a spectroscope, could do spectral analysis. Unfortunately, the difficulty was that the light levels from the spectroscope were too low to activate the phototransistor. After several unsuccessful attempts to develop an amplifier to make the system more sensitive, the student changed the direction of his project. As a result, the new project tried to use the phototransistor as a light metre.

Although this project didn't have quite the romance of the original, it still had considerable merit. The student researched light intensity and luminance. He researched how the computer interacted with the phototransistor in order to tell his program how to read the phototransistor. Then, to translate the value the computer read from the phototransistor into an intelligible number, the student had to understand the nature and importance of instrument calibration. As a final step, the computer collected and stored the data in a format that made it intelligible to a commercially purchased graph analysis program. Using his own program, the student then collected data for luminance versus the separation of light emitted from both a point source (a bulb) and a rod source (a florescent tube) of light. Subsequent graph analysis of the data showed the inverse square law for the point source of light and the inverse first power law for the rod source.

At some point in developing any project, the original excitement of the project wears off. When that happens, persistence in problem solving becomes extremely important. Some students have confidence in their ability and do not need much encouragement; others need regular shots of enthusiasm. Once the first project is successfully completed, the difference in the students' approach to their second project is quite remarkable. Although the second project promises to be longer and more difficult than the first, students are much more confident of their ability to handle the challenge. They are more independent and flexible in their approaches to problem solving. They are less threatened by peer criticism than they were initially.

How students react to peer criticism is especially noticeable during the evaluation stage of the project. Projects are evaluated on the basis of three criteria. The science aspects of the project comprise 50 percent of the final mark. The nature of the computer involvement and the final program contribute 30 percent to the grade. Finally, an in-class presentation of the project, adjudicated by the students, earns 20 percent. Not only do the students respond more positively to peer evaluation on their second project, but, having all been through a peer evaluation, they are much more perceptive and constructive in their criticisms of others the second time around.

I would eventually like to have several students cooperate in a major project. Each student would develop a separate segment of the computer program, which would then be merged with students' sections. Considerable group planning in both the scientific and the computer aspects of project development would be realized. Perhaps the potential for disaster in this approach looms too large.

In the meantime, students benefit from their experiences in several ways: they increase their problem solving ability, develop persistence, come to understand the nature and process of science and learn to appreciate the symbiotic relationship between science and technology.