*Int. J. Man–Machine Studies* (1971) **3**, 219–239

# Some Principles for the Human Use of Computers in Education

THOMAS A. DWYER

*Department of Computer Science, University of Pittsburgh, Pittsburgh, Pa., U.S.A.*

(*Received* 17 *February* 1971)

There are two recurring themes found in recent proposals for educational innovation. The first speaks to the importance of a humanistic approach to education; to the futility of imposing subject content on the student who does not perceive its acquisition as important; to the unlimited potential, on the other hand, of learners who elect to make the pursuit of some educational goal their own private crusade. The second theme is concerned with the potential of computing and information processing systems as instructional devices. This paper argues the importance of communication between these two views. Five principles for relating computer technology to a humanistic view of education are given, and an experimental program in the Pittsburgh public school system which is based on these principles is briefly described. A software system called NEW-BASIC/CATALYST has been developed as a result of this undertaking. Several examples derived from this work are shown.

## Introduction

A recent issue of a trade journal for publishers advised its clients to consider working far more closely with the educational community at the creative level than had been their practice, because "change, which everyone used to believe came slowly in schools . . . is coming at a far less deliberate pace today". While it still is not clear that the massive structures of large school systems are about to shed their innate inertia, there is little doubt that interesting and constructive proposals for educational innovation have begun to take a place alongside the sharp negative criticism of recent years.

These proposals usually fall into one of two broad categories. The first, which might be termed a "new humanistic" approach, speaks to the importance of programs that derive from the inherent curiosity of the human learner; to the futility of imposing subject content on the student who does not perceive its acquisition as a rewarding experience; to the unlimited potential, on the other hand, of learners who elect to make the pursuit of a particular educational goal their own private crusade.

The other affirmative course of action suggests that an extensive use of advanced technology to shore up (or possibly replace) traditional school structures is the hope of the future. In particular, computers and information processing systems are pointed to as instructional tools of great promise.

On first sight these two schools of thought appear to make strange bedfellows indeed. The humanist and the technologist are thought of as natural antagonists, increasingly destined to pull in opposing directions. I believe that this is a superficial conclusion, and that meaningful educational innovation over the next few decades is in fact very much dependent on intelligent communication between these views.

The purpose of this paper is to provide a personal view of some principles that are derived from the humanist point of view, but which are intended as a guide for programs exploring the use of computers in education. It is my conviction that this simple ordering of priorities—principles first, application second—is the real key to successfully tapping the full potential of the new technologies. This conviction is supported by recent experience with applying the principles discussed here to an experiment† in interactive computing for secondary school systems that goes under the name "Project Solo".

## Some Background

Although the principles I will list are general in nature and derive from a variety of educational experiences, they are most directly related to the Project Solo approach to educational computing. Some remarks on the setting of this work are therefore in order.

The experiment is a cooperative effort of the Pittsburgh public school system and the Department of Computer Science at the University of Pittsburgh. Our viewpoint is one which examines the role of computing when considered as a new element to be added to present learning environments. Working as we are in a large urban school system, we are keenly aware that present environments have a potential for crushing new ideas; that there can be an element of folly in attempting to put new wine in old bottles. However, we are even more aware that such schools represent the setting in which most youngsters will have to function for quite some time, and that the operational significance of innovation is directly related to its ability to ameliorate real world systems.

There is an advantage to using computer technology (as opposed to other innovation) in such settings. Computer systems don't "know" that they are

† Supported in part by NSF grant GJ-1077.

supposed to become depressed by their surroundings, and they can function with all the equanimity we wish. Further, the genuine complexity of such machines (which are really not machines at all, but whatever machine we make them) gives them great flexibility.

I believe that some of the errors that have been made in the use of computers arise because of a misuse of these same characteristics. The most common mistake is to use the machines to emulate classroom procedures that have come out of expediency, not thoughtful reflection. This is precisely why principles are needed; a rationale, based on a fresh look at what it is we are about, is the horse that should pull the technological cart.

## Basis for the Principles: Solo Mode Learning

The principles to be given are based on a belief in the value of learner control of certain key aspects of his education. The kind of control I want to advocate is connoted by the word "solo" in our project name, which is derived from an analogy with the dual/solo (with/without an instructor) sequence used in flight instruction. This is a learning situation which develops advanced cognitive and motor skills for students of quite varied backgrounds, and which also involves many affective elements, but which relies heavily on technology for achieving these ends. While it is clear that dual instruction involving close guidance by a teacher is essential (one does *not* recommend that a student immediately go out in an airplane and "do his own thing"), it is equally clear that the student will optimize his benefits from the dual mode if he knows he is preparing for a first solo flight. He knows, in fact, that he can eventually exert more influence on his learning than his instructor.

Our view of computer technology in education is that it should invite similar control at all levels. It should, in particular, invite students to move "behind the scenes" (possibly acting in concert with teachers) at any time they elect. There should be no secrets, no "one-upmanship" of adult world over student world. Some detail on the technology we use to realize these goals is given later in this paper.

## First Principle

Programs that apply technology to education must also continue to support the essential social character of human learning. The mistake that the technologist can easily make is to focus his craft on the learner only, when he should be giving at least as much attention to how he can support the learner–teacher pair, or the learner–peer multiple.

Once this is understood, there is no difficulty in seeing the partial role of machine–man tutoring in an educational system; it will work if and only if it develops the skills that the learner perceives as gaining him *recognition.* To say this another way, motivation has been and always will be vital to learning, but lasting motivation almost always involves human relationships. A use of technology that supports such relationships is harder to structure, but it is a serious mistake to ignore this necessity.

**Second Principle**

The development of educational programs that utilize computing systems is a complex business, and a sound, well-developed supporting structure must accompany the hardware of such systems. Support should include teacher development, curriculum development, and attention to developing an educational theory which both takes advantage of, and influences, the technology. In particular, much attention needs to be given to the man–machine interface, with a structuring of the system so that all facets of the educational process (including the individuality of the learner) can exert real influence on the way in which technology is used.

Applying the first principle we also conclude that teachers and students should be able to talk to each other about all aspects of their work with computers—everything from how to author and file tutorials to exploring the use of other processors on their system. When a student, because he can control the system, comes up with (for him) impressive techniques, it is important that he be able to talk about his work to teachers and peers who understand the significance of what he has done.

Project Solo uses a software package called the NEWBASIC/CATALYST System (abbreviated NBS), which is accessed and controlled through time-sharing terminals. Quite a few other processors are available through these terminals, and NBS can interact directly with four of these (an editor, an extended FORTRAN, an extensible library which currently has over 300 functions, and a rather sophisticated executive system). However NBS is self-contained for handling the full range of educational computing. The other processors add a sense of mystery for students who like to explore, giving them a sort of technical "Disneyland" in which to roam.

Some examples of interactions with programs written in NBS will be given in the last section of this article. It will be noted that algorithmic computing can be freely mixed with "CAI" modes (e.g. within a tutorial). The system is interactive and it encompasses the mixed capability Zinn (1970) mentions as the direction in technology he will be promoting in the seventies. We would

agree that this kind of capability is one that educators should insist on when talking to technologists. It returns the full power of general purpose computing to learning. This is a far better match to the capabilities of the human learner than that found in frame-oriented CAI systems.

**Third Principle**

It is vital that continual attention be given to tapping the internal resources that every learner brings with him to any learning situation. This is especially true when complex technology is involved, since the novelty of its presence tends to overshadow the more familiar but quite superior complex that is man.

Much "CAI" work with computers emphasizes programs (unaccessible to the student) that employ pre-determined strategies which successively expose the student to data. Factual information is transmitted with constant reinforcement of the learner who assimilates it in the expected manner. While such machine-man tutoring has its place (see the first principle), it seems clear that a far more profound use of technology takes place when the student learns to control the machine, both in the sense of "teaching" it, and in the sense of learning to retrieve data in the manner and for the purposes he decides. The student thus develops theories with which he utilizes data, moving in the direction of independence of the inevitable flux in factual knowledge.

Along these lines, one of the most interesting uses of computers we have employed in our high school work involves the creation of tutorial, review, gaming and simulation programs by students (not teachers) for the improvement and individualization of various courses. The students are acting at the organization-of-knowledge level, while simultaneously participating in improving the curriculum to which they have been assigned. Their cumulative efforts are in fact building new models for this curriculum. I feel that the reason students have never previously functioned in this mode on any practical level is attributable in great part to a lack of the right technology.

Further, although the students's work along these lines can very well be superior to the teacher-constructed programs from which the students gleaned their first ideas, teachers boast of the improvement, for the simple reason that they (quite literally) made it all possible. It is the teacher who should unlock technology access for the student, but with an appreciation of the contribution some students can make by virtue of the natural talent of youth in such matters. The NEWBASIC/CATALYST system was designed with this requirement foremost in its specification; it is fully accessible to student exploration.

**Fourth Principle**

The development of curricula that involve computing systems has to proceed on a basis that is open to new insights. It is important, in particular, to be wary of the "logical" sequence of fixing objectives first, and then developing the curriculum to match. The present vision of changes that can take place in learning when educational technology is properly mastered is too dim to make more than initial estimates of what our goals can or should be. We must, of course, make such estimates, but we must view them as subject to considerable refinement.

It has been our experience that the catalytic influence of interactive computing on learning holds equal promise for innovating the formal curriculum structure within which this learning activity takes place. Curriculum content, sequencing, mode of assignment, individualization, motivation and terminal goals all fall into place as likely candidates for specific improvement when educational technology is intelligently applied. This perspective comes home very clearly the first time one sees students making discoveries via a system they created with the aid of technology. The constraints that traditional teacher-to-student verbal communication can place on learning are made clear, and the wisdom of introducing supplementary environments that are learner controlled becomes obvious.

**Fifth Principle**

The intrinsic fun of real computing should be preserved at all costs; it will translate into a joy for other learning given half a chance.

**Distinctions in Computer Related Instruction and Learning**

Before illustrating the application of these principles it will be useful to introduce some terminology that has come out of our work. We first make a more formal distinction between *dual* and *solo* mode learning.

*Definition*: Dual Mode learning is the process of acquiring knowledge, skills, or insights by virtue of the imposition of constraints on the learner that have been (primarily) pedagogically determined.

The implication is that another person's influence (teacher) is present (either as one of the constraints or via the structuring of the constraints), and that the personal understanding of the subject being learned, as embodied in that teacher, is the primary guiding force for the learner.

*Definition*: Solo Mode learning is the process of acquiring knowledge,

skills or insights through an interaction between the learner and a set of subject-determined experiences.

The implication here is that the primary guiding force on the learner is generated by contact with an environment related to the nature of that being learned. The word "environment" should be interpreted liberally to include fellow-students, teachers, and other social influences, communication media (including, of course, books), and a variety of physical systems. The function of teacher now shifts to that of (a) prudent delimiter of the world of environmental elements, and (b) scheduler of transitions between solo and dual experiences.

It is interesting to note that the modes just defined are, in fact, closely associated with the dual roles of "master" and "counselor" predicated of great teachers. In rediscovering the significance of the master/counselor function of great teachers, we have in effect come full circle to a basic but quite profound view of the essence of human learning, namely its cumulative nature. The master transfers accumulated past knowledge, while the counselor elicits new growth in this knowledge via his students.

In applying these definitions to the utilization of computing systems, we make a finer subdivision within each mode as follows (a graphical description of these categories is given in Dwyer & Critchfield (1970) ):

DUAL MODE

*Category I*

Student receives information from computer, responds to questions concerning this information, and on the basis of his response branches to a new point where the cycle repeats. This category includes what other workers have called drill and practice CAI, tutorial CAI, and teacher-directed CAI.

Although we do not stress what some workers call CMI (Computer Managed Instruction), Category I programs can also be used to give diagnostic tests, collecting the results on a file. Management programs can then be written to collect these data in report form. Such reports, or "prescriptions", are the basis of a number of individualized learning schemes. To accommodate such applications, general file manipulation capabilities are included in the NEWBASIC/CATALYST software specification.

*Category II*

The general constraints on subject matter and information access are presented to the student, after which he decides how to use that information and what conclusions to draw. The student can usually compare these conclusions with some frame of reference. We classify this as a dual experience because

the student is interacting with a master program stored in the computer that was written by someone else. This category includes simulation, gaming, and structured information retrieval. It is sometimes called 'learner-directed" CAI.

SOLO MODE

*Category III*

The student writes programs for the computer, debugs and executes them. We call this "learner-devised processing". At one level the student is using the computer to process data (the input to his program) and observe the consequences of transforming that data (the output). However, as was noted in the third principle, the fact that the student had to design an unambiguous algorithm to effect this transformation is the deeper educational effect taking place here. What is less obvious is the profundity of the transformations that he can design. This is because the student can build his program in an incremental fashion, tackling bite-sized problems at each step. The evolving program serves to "remember" all of his discoveries, and continually to accumulate and synthesize them.

We have, in fact, found that reversing this effect stimulates us in our role of curriculum designers. When the "terminal goal" of a given segment of the curriculum is redefined in terms of Category III program (that a student might eventually write to explore that goal), the structure of this program can often help define the sub-goals for the curriculum segment in question in a rather precise fashion.

*Category IV*

We first became aware of this category when working with a teacher on a Category II program to investigate Pennsylvania maple trees. After working a few days in helping him get his program to work, we noted that our original state of zero knowledge about maple trees had zoomed upward to an almost professional level. We now try to place students in this mode whenever possible, and call this "Category IV" computing. The students are reorganizing and carefully checking the subject under study. Without the discipline of computing systems, it would be impossible to ask them to do this in any but the fuzziest way; more importantly, they would never get the feedback that occurs when their systems are utilized by their peers and teachers. We call Category IV "learner-organized learning".

It is clear that moving down through these four categories involves progressively higher cognitive skills. We should also note that while both Category III and Category IV involve student creation of programs, there is an

important difference. The student who writes a program to solve problems for his *own* use is in Category III; a student who writes programs where the primary intention is that *others* be illuminated is in Category IV.

**Some Illustrations of Student Work**

Reducing student activities to the printed page is always unsatisfactory. For this reason we are preparing some films that do a somewhat better job of simulating an actual visit to schools where Project Solo work is in progress. However the following examples may help illustrate the kind of activity possible, given the limitation of teletype input/output display.

*Example I* (*Categories I and IV*). This is a drill and practice problem. As such it is not terribly exciting. However, when we add the information that the program was designed entirely by a ninth-grade student as a facility for his class in German, with the teacher acting as subject consultant, a very different light is put on this work. The second thing that must be explained about this drill is that it is "file-oriented". This means that the German–English vocabulary resides on a file that is separate from the main program. This is also true of the positive and negative reinforcement messages. There are three simple update programs that allow teachers *and* students to change these files. For example, the first time the program was used, it had English reinforcement messages (Good, Correct, No, etc.). The students quickly got the idea to put interesting German messages on the reinforcement files. We suspect more vocabulary has been learned in this way than through the main drill. This is a good example of the value of learner control.

```
>RUN /GERMAN/
GUTEN TAG. ICH HEISSE HANS. WIE HEISSEN SIE?
?MARK
IF YOU WANT TO TRANSLATE ENGLISH TO GERMAN TYPE ENGLISCH.
IF YOU WANT TO TRANSLATE GERMAN TO ENGLISH TYPE DEUTSCH
?ENGLISCH
WIE VIELE WOERTER WOLLEN SIE GEBEN? TIPPEN SIE DIE NUMMER
?2
TIPPEN SIE DAS DEUTSCHE WORT FUER THE RIVER
?DAS FLUSS
DAS GEHT NICHT
SIE HABEN DAS FALSCHE WORT GEGEBEN
TIPPEN SIE DAS DEUTSCHE WORT FUER THE RIVER
?DER FLUSS
WUNDERBAR
TIPPEN SIE DAS DEUTSCHE WORT FUER THE UNCLE
?DER ONKEL
GROSSARTIG
```

```
WENN SIE WEITERMACHEN WOLLEN, TIPPEN SIE MEHR: SONST
NEIN
?MEHR
. . . . . etc. . . . . .
```

*Example* 2 (*Category II*). This is a tutorial on set theory. It is called a tutorial because a student who makes errors is told what his errors were, not just that he is wrong. We classify it as a Category II program, however, because the student decides what elements are in each set, and what set theoretic expression he wants to evaluate. He thus has control of the problem he works with, and can explore the consequences of using different data for these problems. More advanced students who learn to write the parsing routines behind this tutorial are in Categories III and IV.

```
>RUN /SETS/
THIS PROGRAM ASSUMES YOU HAVE READ THE INSTRUCTIONS IN
MODULE # 8.
REMEMBER TO USE 999 TO INDICATE END OF A DEFINITION.
DEFINE THE SET A YOU WISH TO WORK WITH.
?1,2,3,4,5,6,999
NOW DEFINE THE SET B
?3,4,5,6,7,8,9,999
NOW DEFINE THE SET C
?8, 9, 10,11,12,999
WHAT SET THEORETIC EXPRESSION INVOLVING A, B, AND C WOULD
YOU LIKE TO COMPUTE?
? ( (A I B) U C)
WHAT DO YOU THINK THE ELEMENTS OF ( (A I B) U C) ARE?
?1,2,3,4,5,999
THAT'S NOT CORRECT.
YOU LEFT OUT THE FOLLOWING ELEMENTS.
 8  9  10  11  12  6
YOU INCLUDED THE FOLLOWING WHICH ARE NOT IN ( (A I B) U C)
 1  2
WOULD YOU LIKE TO TRY AGAIN?
?YES
DO YOU WISH TO REDEFINE YOUR EXPRESSION, OR SETS, OR BOTH?
?EXPRESSION
WHAT SET THEORETIC EXPRESSION INVOLVING A, B, AND C WOULD
YOU LIKE TO COMPUTE?
? (A U B U C) I (B-C)
. . . . . etc. . . . . .
```

*Example* 3 NEWBASIC/CATALYST permits what we call "system level" interaction, which allows for the mixing of dual and solo modes. The short program shown here is used as an 'author guide" to illustrate some of the possibilities of this feature. It starts out looking like a frame-oriented CAI

lesson which expects a student response after each occurrence of the symbol "?". However the student can break out of this format by typing "@NBS" instead. This action then turns the entire NBS system over to him for solo explorations, ranging from data retrieval to algorithmic computing. He can also call on sub-tutorials (such as /PERCENT/ in our example) and packaged routines (such as HISTOGRAM). When he is finished, he types "EXIT", which then gives him the "?" again, so that he might supply the response which he has just researched.

```
>RUN /DEAN/
NOBODY WANTS TO BE THE DEAN ANY MORE.
QUESTION—IS IT WORTH IT?

LET'S SEE HOW ADMINISTRATIVE SALARIES COMPARE
TO RANK AND FILE FACULTY SALARIES AT COLLEGES.

BY WHAT PERCENT DO YOU THINK THE AVERAGE ADMINISTRATOR
SALARY EXCEEDS THE AVERAGE FACULTY SALARY AT THE LARG-
EST COLLEGE IN UPPER GRAUSTEIN? 10%, 20%, . . ., 100%?

?40%

LET'S CHECK THAT OUT—YOU CAN ACCESS PERTINENT DATA BY
GOING INTO @NBS AND USING SOME FILES. A LIST OF THE FILES
AVAILABLE TO YOU IS GIVEN UNDER 166TD /DEANINFO/.

?@NBS
VER. JAN 19 14:57
>—COPY 166TD /DEANINFO/ TO TEL
5 FILES AVAILABLE
/COLLEGES/
/SLIPSOCK/
/OBNOX U/
/IOU/
/ZEN STATE/
>—COPY 166TD /COLLEGES/ TO TEL
COLLEGE ENROLLMENTS OF UPPER GRAUSTEIN
SLIP SOCK 963
OBNOX U  543
ZEN STATE 111
COLLEGE ENROLLMENT OF LOWER GRAUSTEIN
IOU 55

>—COPY 166TD /SLIPSOCK/ TO TEL
SALARIES OF EMPLOYEES OF SLIPPERY SOCK
PRES        40,000
V PRES      35,000
DEAN        30,000
FACULTY
```

| DEPT | NUMBER | AVERAGE SALARY |
|---|---|---|
| ENGLISH | 3 | 13,000 |
| SOCIOLOGY | 2 | 13,500 |
| PHYSICS | 1 | 14,000 |
| MATH | 2 | 14,500 |
| PYSCHOLOGY | 10 | 15,000 |
| BIOLOGY | 4 | 15,500 |
| CHEMISTRY | 2 | 16,000 |
| PHYSICAL ED | 3 | 16,500 |

```
>PR. (40000+35000+30000)/3

35000
>PR. 3*13000+2*13500+14000+2*14500+10*15000+4*15500+2*16000+
      3*16500
  402500

>PR. 403500/(3+2+1+2+10+4+2+3)
  14907

>PR. ( (35000.-14907.)/35000.)*100
  57.40857143

>EXIT
?57.408%
NO, 57.408% IS NOT THE CORRECT ANSWER.

FOR SOME HELP, GO BACK INTO @NBS AND TRY RUNNING 166TD
/PERCENT/. P.S. DON'T FORGET TO USE DECIMAL POINTS.
?@NBS
VER. JAN 19 14:57
>RUN 166TD /PERCENT/
THE PERCENT BY WHICH A NUMBER X EXCEEDS A NUMBER Y IS
GIVEN BY ( (X-Y)/Y)*100. LET'S TRY THIS ON YOUR PROBLEM.

WHAT IS X?35000.
WHAT IS Y?14907.
THEN X EXCEEDS Y BY 134. 789%
DO YOU WISH TO DO ANOTHER?NO
>PR. (35000.-14907.)/14907.*100
  134.7890253

>EXIT
?134.789%

GOOD—THAT'S WHAT WE GOT TOO.
(YOUR ORIGINAL ESTIMATE OF 40 WAS 236.9725 PERCENT TOO LOW!!!)
AS A FINAL EXPLORATION, WRITE AN NBS PROGRAM WHICH USES
THE LIBRARY HISTOGRAM PROGRAM TO DISPLAY THE DISTRIBU-
TION OF FACULTY SALARIES.
```

. . . . . etc. . . . . .

*Example* 4. The text-scan functions in NBS were used to write this English composition game. Note the psychology of allowing the program to be fallible, so that student improvement (Category IV) on this Category II program can be invited.

```
EACH TIME THE COMPUTER SUPPLIES
A QUESTION MARK (?), TYPE IN
A SENTENCE USING EACH OF THE WORDS
'VALLEY', 'IS', AND 'GREEN', EXACTLY ONCE.
IF YOU THINK THE COMPUTER HAS INCORRECTLY ANALYZED
YOUR SENTENCE, TYPE THE WORD 'CHALLENGE'
?IS THE VALLEY BEYOND THE MOUNTAINS GREEN?
ANALYSIS: A LONG QUESTION.
THE COMPUTER WILL IMITATE YOU BY WRITING SIMILAR SENTEN-
CES:
IS THE VALLEY BEYOND THE SNOW COVERED ALPS
INHABITED BY LITTLE GREEN MEN?
TRY ANOTHER
?GREEN IS MY VALLEY
ANALYSIS: A SHORT POETIC SENTENCE.
THE COMPUTER WILL IMITATE YOU BY WRITING SIMILAR SENTEN-
CES:
GREEN IS MY VALLEY. VERY GREEN IS EVERY VALLEY
TRY AGAIN.
?GREEN MEN LIVE IN THAT VALLEY
YOU HAVE NOT USED ALL THE WORDS IN YOUR
SENTENCE. PLEASE REREAD THE INSTRUCTIONS AND TRY AGAIN.
?GREEN IS THE COLOR OF THE GRASS IN THE VALLEY
ANALYSIS: A LONG POETIC SENTENCE.
THE COMPUTER WILL IMITATE YOU BY WRITING SIMILAR SENTEN-
CES:
GREEN IS THE VALLEY WHERE MY TRUE LOVE AND I WILL BUILD
OUR COTTAGE
TRY ANOTHER.
GREEN VALLEY IS THE PLACE WHERE I WANT TO LIVE
ANALYSIS: A LONG DECLARATIVE SENTENCE.
THE COMPUTER WILL IMITATE YOU BY WRITING SIMILAR SENTEN-
CES:
HO HO HO, THE GREEN GIANT IS IN THE VALLEY OF
THE CORN GROWERS, BUT THEY WANT A YELLOW GIANT.
TRY ANOTHER
?IS THAT THE VALLEY OF GREEN PEOPLE WHO EAT YELLOW CORN?
ANALYSIS: ANOTHER LONG QUESTION.
TRY AGAIN
?WHAT IF I DON'T WANT TO USE IS GREEN AND VALLEY
ANALYSIS: ANOTHER LONG QUESTION
TRY ANOTHER
```

?GREEN IS AND VALLEY ARE STARTING TO BORE ME
ANALYSIS: ANOTHER LONG POETIC SENTENCE.
TRY ANOTHER
?CHALLENGE
YOU SEEM TO HAVE OUTSMARTED THE COMPUTER. TAKE THIS PAPER TO YOUR TEACHER TO DISCUSS REVISING THIS PROGRAM SO THAT IT CAN HANDLE YOUR 'CHALLENGE' SENTENCE. GOODBYE FOR NOW.

*Example* 5 (*Categories I, III, IV*). This is a short excerpt from one of a series of student-authored programs which were prepared as a means of 'cramming' for state final exams in chemistry. The students who authored the programs (using their teacher as a subject consultant) were in Category IV; their classmates who use the reviews to get ready for the exams *elect* to be in Category I. They can briefly move into Category III by using the @NBS feature for computation. The student humor in the reinforcement messages is quite acceptable because it is student-authored; teacher-authored programs would have to be more reserved. It should be noted that data in problems (e.g. the "31" in "31 GM of MN02") are randomly generated, effectively making lessons new each time they are run. In order to evaluate the different student responses for each run, the NBS drill program continuously calculates problem solutions that match the new data.

RUN /WILL:1/
LET'S LOOK AT SOME WT.–WT., WT.–VOL., AND VOL.–VOL. PROBLEMS.
ONE DAY, WILLY WIZBANG WAS WORKING IN HIS LAB AND FOUND THAT HE COULD MAKE CHLORINE GAS, MNCL2, AND STEAM (H2O) BY REACTING MNO2 WITH HYDROCHLORIC ACID. HE THEN DESIGNED AN APPARATUS SO THAT HE COULD COLLECT THE WATER GAS IN ONE CONTAINER AND THE CHLORINE GAS IN ANOTHER CONTAINER. CAN YOU GIVE THE BALANCED EQUATION FOR THIS REACTION? TYPE IT IN THE FORM: 2H2+O2=2H2O.
?MNO2+4HCL=CL2 +MNCL2+2H2O
MAZELTOV!! (AND IN CASE YOU DON'T SPEAK FRENCH, THAT MEANS CONGRATULATIONS!!) YOUR EQUATION IS CORRECT.
LUCKILY, WILLY HAD WEIGHED THE MNO2 BEFOREHAND AND KNEW HE HAD USED 31 GRAMS.
WHAT IS THE WEIGHT OF CL2 PRODUCED?
?@NBS
>PRINT 31./87.*71
25.29885057
>EXIT
?25.2988
SEHR GUT! (SEE, YOU'RE NOT ONLY LEARNING CHEMISTRY, BUT FRENCH AND ITALIAN TOO!)
25.29 IS THE CORRECT WEIGHT OF GAS.

WILLY THEN WANTED TO CALCULATE WHAT VOLUME CONTAINER WOULD BE NEEDED FOR STORING THE CL2 GAS AT STP. WOULD YOU CALCULATE IT FOR HIM AND ENTER YOUR ANSWER BELOW?
?25.29
WRONG!! REMEMBER THAT ONE MOLE OF ANY GAS OCCUPIES 22.4 LITERS AT STP. THIS IS YOUR LAST CHANCE! ENTER YOUR ANSWER AGAIN!
?@NBS
>PRINT 25.298/71*22.4
7.981340845
>EXIT
?7.981
WHAT A SUPER-CHEMIST YOU ARE! YOU AND WILLY BOTH GOT THE SAME ANSWER 7.98 LITERS. THAT MUST BE RIGHT!
NOW CAN YOU HELP WILLY IN FINDING WHAT SIZE CONTAINER HE NEEDS IN WHICH TO STORE THE STEAM PRODUCED IN THE REACTION AT STP.
?15.96 LITERS
GOOD, GOOD, GOOD! ! !
WILLY NOW USED THE CL2 GAS PRODUCED TO REACT IT WITH H2 GAS STORED IN THE CYLINDER AT 4.65 ATM PRESS, AND 25C DEG. TO MAKE HCL GAS. WHAT VOLUME OF H2 DID HE NEED AT THESE CONDITIONS?
?@NBS
>PRINT 7.981/273.*298./4.65
1.873518453
>EXIT
?1.8735
GREAT!! THAT'S RIGHT. NOW FOR THE LAST PROBLEM (SIGH!)
WILLY WANTED TO STORE THE HCL PRODUCED IN A STEEL CONTAINER AT A PRESSURE OF 4.49 ATMS. AT TEMPS. OF 12, 15, 23, AND 30 DEGREES C. WHAT SIZE CONTAINERS WOULD HE NEED? GIVE THE ANSWERS IN LITERS:
?@NBS
>10 PRINT "TEMP.="
>20 INPUT T
>30 PRINT "VOL=": 15.96/273.*(273.+T)/4.49
>40 GOTO 10
>RUN
TEMP.=?12
VOL=3.710810348
TEMP.=?15
VOL=3.749871509
TEMP.=?23
VOL=3.854034607
TEMP.=?30
VOL=3.945177317

```
TEMP. = ?←←ESC:
>EXIT
?3.7108, 3.7498, 3.8540, 3.9451
CORRECT! NOW WRITE A NEW ADVENTURE FOR WILLY SIMILAR TO
THIS!
```

*Example* 6. Category III is difficult to illustrate with printed program listings, since such listings do not say much about what went on in a student's head during the creation and debugging process. Instead of listing a long and complex program written by a student in this category, it will be preferable to present a short one where the analysis that the student had to carry out is more transparent. This particular program produced its output on a Hewlett Packard plotter, a device we like very much.

/BOUNCE/ is a program that simulates inelastic collision for a ball moving in a planar "slice" of a three-walled court. It assumes that the ball is given an initial horizontal velocity of 200 ft/sec, and that the only force acting on the ball while in air is that of gravity.

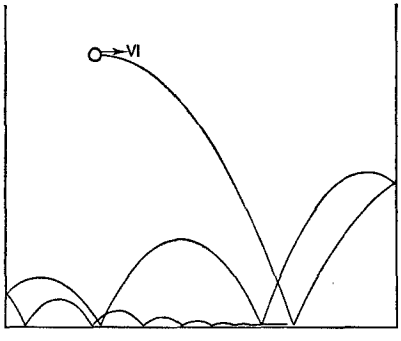


FIG. 1. Plotted output of program /BOUNCE/.

When the ball hits a wall, the $x$ component of velocity is reversed and decreased by 25%; when it hits the floor, the $Y$ component is reversed and decreased by 25%. The value 25% can be changed to examine the effects of various losses.

LISTING OF THE PROGRAM /BOUNCE/.

```
10 PRINT "PLTL"
20 LET X=3000 LET Y=7000 LET V1=200 LET V2=0 LET I=0
30 PRINT INT (X); INT (Y)
40 I=I+1
50 X=X+V1*0.04
```

```
60 IF X>1000 AND X<9000 THEN 170
70 IF X=1000 THEN 90
80 X=1000 GO TO 110
90 X=9000
100 Y=Y+V2*0.04
110 PRINT INT (X); INT (Y)
120 LET I=0 LET Vl=(-.75)*V1 GO TO 40
140 Y=Y+V2*0.04
150 IF Y>1000 THEN 70 ELSE LET Y=1000 PRINT INT(X); INT(Y)
160 LET I=0 LET V2=(-.75)*V2 GO TO 40
170 V2=V2-1.28
180 IF I< >25 GO TO 40 ELSE PRINT INT(X); INT(Y) LET I=0 GO TO 40
```

*Example* 7. Documentation of /BOUNCE/

One way to get a student to tell us something about what cognitive processes were involved in creating a program like the above is to ask him to document the program. This is hard for most students to do at first, and examples of good documentation are needed to guide them. Computer programs have the distinct advantage of having structure, however, so suggesting documentation on a line-by-line basis helps students a great deal. *It should be noted that this is an off-line activity.* For this reason, we believe that on-line computing costs provide an erroneous basis for judging the worth of solo mode educational computing.

Sample documentation of /BOUNCE/

| Line | Explanation |
|---|---|
| 10 | A command which puts plotter in line mode. |
| 20 | Position of ball is ($x$, $y$). Velocity components are (V1, V2). Line 20 initializes these variables. |
| 30 | Plots initial position of ball. |
| 40 | The new position of the ball is calculated every 0.04 sec; however, the position is plotted only every second. The variable 1 is a counter (incremented in 40) which keeps track of when the plotting should take place. Hence it is initialized to zero and when it equals 25 (see line 180) the position is plotted and 1 is set back to 0 etc. |
| 50 | The $x$ co-ordinate of the new position is calculated as new value = old value + horizontal velocity* change in time. |
| 60 | The "walls" are at $x=1000$ (left wall) and $x=9000$ (right wall). If the ball is between the walls, skip the bounce. That is, go to 170. |
| 70 | If the ball is not between the walls and $x>1000$ then $x$ must be *at* the right wall. (It can't go past the wall.) So go to 90. |
| 80 | Otherwise the ball must be at the left wall. Therefore set $x=1000$. |
| 90 | The ball is at the right wall so set $x=9000$. |

100 The $y$ co-ordinate of the new position is calculated. See explanation of statement 50.

110 When the ball is at a wall, its position is plotted whether the counter is 25 or not.

120 Reinitialize counter to zero since a point was just plotted. The bounce was an inelastic collision so the $x$ velocity is decreased by 25% and reversed.

140 Compute the $y$ co-ordinate of the next position (see 50).

150 If the ball is above the floor which is at $y=1000$ skip the bounce. That is, go to 170. Otherwise, the ball *is* at the floor so set $y=1000$. Plot the position when the ball is at the floor regardless of the counter.

160 Reinitialize the counter since a point was just plotted. Reverse and decrease by 25% the vertical component of the velocity, then go back and compute the next position.

170 In 0.04 sec the force of gravity causes the vertical component of the velocity to decrease by 1.28 ft/sec.

180 See explanation of statement 40.

## Summary

The approach to educational computing described in this paper is best summed up in the word "control". Both students and teachers are encouraged to control technology to serve their educational goals. This means that both parties have to ask themselves what these goals are, a very beneficial exercise indeed. The planning they engage in takes on a meaning that is quite different from that usually associated with such activity, however, because it is always verifiable through the impartiality of the machine. When things finally run as expected, there is an excitement and sense of achievement that triggers a cascade of new learning activity.

On the negative side of the ledger, I see three principal difficulties barring the successful use of technology along the lines described. The first, and to my mind the most serious, is the manner in which the rigidity of present school systems militates against free access to technology when-and-as the spirit moves. The second is the patience needed in bringing teachers up to the confidence levels needed. The third is one of economics. The cost of operating in the manner we have described is about a 10% increment in the annual cost of educating a secondary school student (e.g. from \$900 per year to \$990 per year). On a total school system basis, where grades 1 through 12 are serviced, the cost of allowing students in grades 8 through 12 to use the computer in about 25 % of their courses is a 2% increment in the total school budget.

Nevertheless, many school system administrations are still reluctant to make this move.

One final comment should be made, and this has to do with an unsuspected benefit of educational computing of the kind described. If one were to select a single action for transforming the schools overnight, a very good choice would be to upgrade teachers to the rank of "master educator", using this term in the best possible sense. From what we have observed in our first year of experience, the use of computers in the manner we have described is a promising tool for accomplishing this goal. We have seen high school teachers making genuine scholarly discoveries at terminals. Their new expertise soon rubs off on their charges. The converse effect (students making discoveries under the guidance of their teachers) supports a mutual respect between teacher and learner that may very well be the most important contribution computer technology can make to education today.

### References

DWYER, T. A. & CRITCHFIELD, M. D. (1970). CATALYST: CAI in a General Time-Sharing Environment. *EDUCOM Bull*, **5**, 5.

ZINN, K. L. (1970). Instructional programming languages. *Educ. Technol.* pp. 43–46, March.

### Appendix I—The NBS System

NBS (The NEWBASIC/CATALYST system) is an extension of standard BASIC in four respects:

(1) Extension as an algorithmic language.
NBS allows such things as multiple data types (REAL, INTEGER, COMPLEX, DOUBLE REAL, etc.), suffix control (WHILE, UNTIL, ELSE, etc.), multiple statements (IF X > 5 PR. "LARGE" ELSE PR. "SMALL" GOTO 120), picture formats, dynamic string allocation, and extensive file handling commands.

(2) Extension as a CAI language through the CATALYST functions.
String functions that allow for the scan of user input lines are provided to allow students greater freedom in responding to questions posed by drill or tutorial programs (cf. Appendix II).

(3) Extension as an interactive language.
(a) Because NBS is compiled incrementally, it has the interactive capability of interpretive languages such as JOSS or PIL. Thus

execution can be halted, variables examined, changes made, and execution resumed. A "direct mode" can also be mixed with the "indirect" or stored program mode.

(b) The @NBS feature (illustrated in examples 3 and 5) permits a second level of NBS access for students who are under control of a first level NBS CAI program.

(c) Interaction with executive commands (e.g. looking at directories of files or asking for the editor) is possible while remaining in either the execute or create phases of NBS.

(4) Extension to meet new needs.

Users can write functions to suit their own particular needs, using assembly language, FORTRAN, or NEWBASIC. These can be added to the list of public library functions any time that general interest warrants. There are over 300 public library functions at the present time.

## Appendix II—the Catalyst Functions in NEWBASIC

Three examples of CATALYST scan functions as used in NBS are listed here. As mentioned in II-4 above, new functions can be added as need develops.

Function: ICO(A$, B$, IWRD)

Sample use: 10 INPUT A$

20 IF ICO(A$, "NO", 1) THEN 220

The parameter IWRD is set either to 0 or 1, depending on whether we want B$ to be treated as a *string* or *word.* It is considered a *word* if it is *not* surrounded by alphanumeric characters. Thus the "NO" in "HEAVENS NO!" is considered a *word,* while the "NO" in "I DON'T KNOW" is not considered a word. The function ICO returns the value TRUE if the string (or word) B$ is *contained* in the string A$, FALSE otherwise. Thus we can read line 20 in our sample to mean "if the user's input sentence contains the word NO branch to line 220, otherwise execute the next line."

Function: IBEF(A$, B$, IWRD1, C$, IWRD2)

Sample use: 10 INPUT A$

20 IF IBEF(A$, "RED", 0, "FLAG", 1) THEN 50.

Line 20 can be read as "if the user's input contains the string RED and it also contains the word FLAG, with RED appearing *before* FLAG, branch to line 50; otherwise execute the next line". Since IWRD1 = 0, REDDISH FLAG would be accepted; since IWRD2 = 1, REDDISH FLAGPOLE would not be accepted.

Function: IEQIV(A$, B$, IWRD)
Sample use: 5 B$ = ", YES, SURE, YEP, YUP,"
10 INPUT A$
20 IF IEQIV(A$, B$, 0) THEN 70

Line 20 can be read to mean that branching to line 70 will take place if the input string is any one of the strings "YES" or "SURE" or "YEP" or "YUP".

Since all CATALYST functions return value TRUE or FALSE (1 or 0), they can be interconnected with Boolean operators. For example:

IF IBEF(R$, "MOLECULAR", 1, "WEIGHT", 1) AND ICO(R$, "72", 1)
BUT NOT ICO(R$, "ATOMIC", 1) CALL REIN, GOTO XXX

is a legal statement. CALL REIN calls the function REIN which supplies a reinforcement message randomly selected from a pre-stored list.