# CWF/3000: A COMPLETE SYSTEM FOR COMPUTER ASSISTED INSTRUCTION AND TRAINING

HAROLD J. PETERS
EDUCATIONAL SOFTWARE PRODUCTS
9 GEORGETOWN CIRCLE
IOWA CITY, IOWA

## What is CW?

CW stands for "coursewriter", which is an authoring language for computer-assisted instruction (CAI). Figure 1 shows a sample of student interaction with a CAI lesson that was written with CW. While the subject matter can of course vary greatly, this example is typical of the sort of tutorial dialogue that can readily be written with the CW language.

Figure 2 shows the CW code corresponding to the sample student interaction in Figure 1. The two letter "op-codes" are almost self-explanatory; QU: question, CA: correct answer, TY: type, WA: wrong answer, UN: reply for unexpected answer. This example illustrates the principal advantage that CW, or any other CAI authoring language, offers over a general purpose programming language -- that is: implicit branching. If the student's answer does not match the argument of the CA, then processing automatically branches around the TY associated with the CA and on to the next implied comparison, the first WA.

FIGURE 1

```
Most nouns ending in "ch" form plurals
by adding "es".
For example,
           "church" becomes "churches",
           "lunch" becomes "lunches", etc.
What is the plural of "torch"?
torchs
"torch" ends in "ch".  Please try again.
terches
I think you've spelled "torch" wrong.
torches
Good!  Try another.
```

Figure 1. A sample of student interaction with a CAI course written in CW.

FIGURE 2

```
QU     Most nouns ending in "ch" form plurals
       by adding "es".
       For example,
                   "church" becomes "churches",
                   "lunch" becomes "lunches", etc.
       What is the plural of "torch"?
CA     torches
TY     Good! Try another.
WA     torchs
TY     No, "torch ends in "ch".  Try again.
WA(L)    t&ches
TY       I think you've spelled "torch" wrong.
         Please try again.
UN     No. The answer is "torches".  The ending "es"
       is added for the plural because "torch" ends
       in "ch".  Try another one.
BR     PR
```

Figure 2. CW code for the sample student interaction given in Figure 1.

Again, if there is no match a branch is made automatically to the next implied comparison, etc. If on the other hand, the student's answer does match the argument at one of the implied comparisons, then the corresponding TY (and/or other so-called "minor" op-codes) is executed and all further comparisons are skipped over in an automatic branch to the next question.

All this implicit branching does save a lot of busy work on the part of the author and lets him concentrate on the higher-level structure of the lesson. It clearly does not absolve the author of all programming tasks but it does make his job a lot less tedious.

First appearing in the early 1960's, CW has to be considered one of the authentic pioneering languages for CAI. Many CAI languages have come along since the introduction of the first version of CW. And this leads to legitimate speculation as to why this "old" language still manages to survive. Two reasons appear most prominent. First, CW is extensible: the short-comings in its original design can be circumvented by the use of its user-written function feature. Any programming function within the capabilities of the underlying software system can in principle be invoked by a CW course through use of the user-written function feature. Within a CW course, the function call is simply

```
FN MYPROG
```

where MYPROG is the name of the user-written function to be invoked. More will be said regarding user-written functions in a later section.

The second factor contributing to the longevity of CW is that because of its early existence on the most popular equipment, i.e., IBM, a great deal of courseware has been created in CW over the years, and the easiest way for newcomers to CAI to get off to a running start has been to tap into that large reservoir of courseware by getting a CW system themselves. They then in turn create more courseware and so the bandwagon goes on. It was precisely this second attractive feature of CW -- the large accumulated base of courseware -- that originally drew the interest of Hewlett Packard -- and leads to the next question:

## What is CWF?

CWF stands for Course Writing Facility, Hewlett Packard's name for its emulation in BASIC of IBM's CWIII (the version of CW current in the early 1970's). CWF was originally developed for the HP2000 timesharing system, and some 15-20 CWF systems were sold prior to HP's withdrawing it from the market, at least partly in anticipation of the obsolescence of the HP2000 itself. Essentially all features of CWF in its HP2000 implementation have been retained in the HP3000 version, so the description of those features is deferred to the next section.

## What is CWF/3000?

Building upon its HP2000 predecessor, CWF/3000 provides on the HP3000 essentially all the course authoring, course presentation, and student-keeping capabilities of IBM's Coursewriter III, version 3 (CWIII), plus a significant additional feature <u>not</u> offered by CWIII. In CWIII, the extensibility described earlier is achieved through user-written functions programmed in IBM that such functions can be written directly in BASIC.

CWF/3000 consists of four major subsystems: course authoring, course translation, student usage, and record keeping/reporting, each described in turn below.

Course Authoring

CWF/3000 authors use a set of programs headed by CWEDIT to enter, revise and edit new course material, using the CW language op-codes such as shown earlier in Figure 2. Some thirty op-codes, many of which may be modified in several ways, offer wide flexibility to the author. For example, the CA op-code in simplest form accepts exactly one correct answer:

```
CA torches
```

With an "L" modifier, a variety of answers can be accepted through usage of "don't care" characters. For example,

```
CA(L)  t&ches
```

will accept "torches", "touches" or "txyzches" or any other "word" beginning with "t" and ending with "ches", as correct. Another modifier,"W", allows more specific alternative correct answers. For example,

```
CA(W) torches lamps candles
```

accepts these three and only these three as alternative correct answers. And if all this flexibility is not enough, the author can resort to a user written function, e.g.,

```
FN ANSWER
```

and rely on his own BASIC program, ANSWER, to achieve <u>any</u> type of answer processing he may desire that can be implemented within the broad generality of BASIC. The experienced CWF/3000 author moves flexibly between CW and BASIC, taking advantage of the best features of each as the occasion demands. Very roughly, CW is superior in dialog-intensive work such as in tutorials, and BASIC is superior in computation-intensive tasks such as simulations.

## Course Translation

In principle, any CWIII course developed on IBM equipment can be converted via CWF/3000 conversion programs to CWF/3000 compatible form and then used, revised and edited just like any other CWF/3000 course. In practice, virtually every conversion we have seen attempted has succeeded -- eventually. The biggest problem -- when there have been problems -- arises with courses that make extensive use of special display characteristics. The special display features available on the system on which the course was originally offered may not be available on the target system. And even if they are, they are likely to be implemented differently so that considerable conversion attention may be necessary. But whatever effort may be required, it is almost invariably true that starting with someone else's courseware and modifying it in whatever ways necessary to meet one's own needs remains a far superior method for getting usable courseware up and running than by starting from scratch on one's own. So it remains the case that the principal source of attraction in CWF/3000, as in its predecessors, is the ability it gives the user to most easily tap into a large existing base of CW courseware. The 1978 Index to Computer-Based Learning[1] lists 311 instructional modules developed in CWIII that in principle should be readily convertible to CWF/3000. In addition, the Index lists 720 modules written in BASIC, which are compatible with CWF/3000 in important ways, as described in a later section.

Student Usage

CWF/3000 incorporates HP's Instructional Management Facility, IMF, to handle student sign-on, sign-off and some of the student record keeping. Hence a student taking a CWF/3000 course enters through the IMF program START in the following standard fashion:

```
RUN START.PUB

What is your ID number and first name? 1000,Jimmy
Is your last name Carter? Yes
Course name?  ENG4
30 September 1978        14:49        Port 7

Welcome, Jimmy to session number 3 of English 4.
In our last session....
```

Whenever the student signs off by typing //STOP or //SIGN OFF, or is signed off automatically by the instructional program, the CWF/3000 system saves his restart information so that he may begin his next session where he left off in the last one, or wherever else the author may wish to designate.

Record-Keeping/Reporting

In addition to student restart information, as already described, many other types of information relating to student usage of a CW course can be recorded and later reported. Information concerning essentially any aspect of student interaction with the course material can be saved by one means or another.

Perhaps most common is simply keeping track of correct answers in a session, reporting this to the student at sign-off time, and possibly recording the number for reporting to the instructor later, or maintaining a cumulative day-to-day record of scores for benefit of both the instructor and student. Another common type of information saved is the number of times each registered student has accessed a course and how much cumulative time the student has spent on the course.

During the development phase of a course it is especially important to accumulate all <u>unanticipated</u> student answers, i.e., all answers for which no tailored replies had been prepared and for which only the "reply to unanticipated answer" (the argument of the UN op-code) is displayed to the student. Frequently, these unanticipated student answers will suggest that parts of the course need further development work, including perhaps new explanatory passages, or at least a broader array of model answers and corresponding tailored replies. Most of the record keeping and reporting functions that have been mentioned require that students be <u>registered</u> for courses, and that the courses themselves first be entered into the IMF record keeping system. Some examples of instructor or proctor interaction with the IMF programs are shown in Figure 3.

FIGURE 3

A. Entering a course

```
RUN ADMIN.PUB
CODE?MMMMMM
COMMAND?COURSE
  COURSE COMMAND?ENTER
    COURSE NAME?REX
    CODE WORD?R1
    DOES IT HAVE A DEMO MODE?YES
    SPECIAL COURSE TYPE?CW
    REX IS NOW ENTERED AS A COURSE.
    COURSE NAME?//STOP
DONE
```

B. Enrolling a student

```
RUN PUPIL.PUB
CODE?MMMMMM
COMMAND?ENTER
  FIRST NAME?JOE
  LAST NAME?SWARTZ
  ENTERED WITH ID NUMBER 1018.
    COURSE NAME?REX
    GROUP NAME?MS NOEL
    HISTORY FILE OF REX INITIALIZED TO 9 STUDENTS.
    AREA NUMBER?1
    USER GROUP NUMBER?1
    ENROLLMENT COMPLETED.
  FIRST NAME?//STOP
DONE
```

Figure 3. Some examples of instructor or proctor interaction with IMF record keeping programs under CWF/3000.

As a final note regarding recording keeping, we should point out that the CWF/3000 record keeping facilities are available for BASIC language courses as well as CW courses. It is typically the case that computer-based instructional materials written in BASIC have not used any kind of student record keeping functions. Typically instructors have no information as to which students have used the modules nor as to how well they have performed. It is certainly difficult to assess the instructional value of CAI materials in this case. And refinement of the modules, or development of new materials must be based on guesswork rather than hard data. Once again, the record keeping and report facilities of CWF/3000 provide a ready solution to these problems for both BASIC and CW courses.

## Some Technical Considerations

CWF/3000 has not been optimized for the HP3000. As of September, 1978, only the student driver programs have been compiled so that they can run as object code rather than under the BASIC interpreter. This provides good performance for up to 15-20 students running most CW materials.

Some technical problems have delayed compilation of the authoring programs, which means that some authoring functions execute much more slowly than desired.

Beyond simple compilation of the various programs comprising CWF/3000, it is clear that substantial further optimization could be achieved through redesign of the file structures, which still suffer from design constraints imposed by the early

HP2000 series time sharing systems. These and other refinements to CWF/3000 await the indication of further interest in the system by HP3000 users.

## Availability

CWF/3000 has been developed by and is available from (under a license agreement) Educational Software Products. Requests for further information should be directed to the author, at that organization.