The Origins of PLATO IV

In fall 1973, I was a newly graduated Doctor of Psychology with an interest in psycholinguistics, looking for something useful to do in Urbana between interviews for jobs in academic psychology. A friend from the French department suggested "Try the Language Lab—they’re doing something with that new PLATO system." This offhanded bit of advice began an involvement with PLATO and with CALL that expanded to occupy the next two decades.

When I joined the Foreign Languages project, the PLATO computer system was already a fixture on the Urbana campus. Three previous versions had existed, products of the dedication of Don Bitzer, a professor of electrical engineering. Bitzer’s original concern had been the nuts and bolts of hardware design, but his work with computers convinced him that they held great potential to do useful instructional tasks. Soon he was tinkering with the cumbersome mainframes of the period, asking how they could be adapted to teaching. The first system, PLATO I, was simply a teletype terminal attached to a mainframe, scarcely more than a typewriter that could occasionally talk back. A few instructors on campus saw it demonstrated and, convinced by Bitzer’s
technical expertise and vision, became collaborators. Paul Tenczar, a biologist who was afterwards to play a major role in the development of the TUTOR programming language, was one of these; another was Bruce Sherwood, a physics professor who was central to various aspects of system development.

Working with real instructors and students forced the PLATO developers to grapple with real world instructional problems. By the late sixties, Bitzer’s group boasted PLATO III, a fully functional educational computing system. Not much attention had been given to the amenities common in today’s learning centers. The building housing CERL (the Computer based Education Research Laboratory) lay in the heart of the engineering campus; used during World War II for research on radar and other military applications, it had never entirely lost the rather grim and dingy ambiance associated with military enterprises. Students entered the PLATO III classroom through heavy double doors provided with porthole windows, and sat at rows of student stations that, to take full advantage of the limited space, were wedged together in a way that would appall a contemporary ergonomics expert. Enclosed on three sides by walls of acoustic tile, each student stared at a fuzzy, flickering tube of a black and white CRT display and typed in answers. Next door in the machine room, operators struggled to keep the system running smoothly. In spite of the lack of atmosphere, the system was heavily scheduled with students doing assigned class work in physics, biology, nursing, and foreign languages.

In 1969, the PLATO group felt they had advanced far enough to demonstrate the feasibility of computer based instruction, and presented a grant proposal to the National Science Foundation to build PLATO IV, an improved, greatly expanded version of PLATO III. NSF also felt that the time was right to devote significant resources to examining the feasibility of computer based education. The agency therefore decided not only to support the development of PLATO IV, but also to fund at the same time development of Brigham Young University’s TICCIT project (see the article by Randall Jones in this volume).

In some sense, therefore, PLATO IV and TICCIT were born as competing projects. In fact, competitive evaluation made sense, because the philosophies of the two projects differed substantially. PLATO envisaged a single large, centralized system that would handle all technical services. The function of the system staff was to provide hardware
and software tools, leaving instructors free to do instructional design (or "lesson authoring," as the PLATO people preferred to call it). TICCIT, based on minicomputer hardware, was less oriented to centralization, and perhaps more important, as a matter of principle gave attention to incorporating into the software and authoring capabilities of the system a model of instructional design that could be used to guide and rationalize materials development.

The PLATO IV project generated enormous excitement on the UI campus. Beyond the enthusiasm that naturally accompanies all new projects, PLATO IV carried along a sense of mission—a belief that the people who made this system work would be accomplishing something of national, and perhaps historic, importance. After all, Don Bitzer had projected that PLATO would eventually serve thousands or even millions of terminals linked together by voice telephone, microwave, and video connections, and deliver instruction over this network for a fraction of a dollar (the exact figure varied from time to time) per hour. Everyone was eager to go, and when the money arrived, CERL started right out in a no-nonsense way with setting up the PLATO IV system hardware and software. It was at this point that I joined the PLATO Foreign Languages Project.

As luck would have it, a brand new Foreign Languages Building was just coming into commission. Keith Meyers, Professor of French and Director of the Language Learning Laboratory, was a great believer in PLATO as a language teaching tool that would eventually absorb and encompass all other forms of teaching technology. During the planning stages, he had lobbied ceaselessly for PLATO facilities to be included as an integral part of the new building. The result was that the entire center of the basement had been set apart as a huge classroom/laboratory site, with long rows of high-sided carrels to accommodate 80-100 PLATO terminals. An adjacent room for machinery and office space for programmers was also provided. Even the PLATO terminals and communications equipment were included in the budget of the building. In consequence, the Foreign Languages Building would have splendid PLATO facilities, the largest single site on campus, and probably in the world.
The new PLATO IV stations were unprepossessing: large, dull brown cubes about eighteen inches on a side, and backbreakingly heavy. What they lacked in appearance, however, they made up for in performance. The display screen, a plasma panel rather than a video tube, showed everything in the tiny orange plasma dots which were later to become so well known, slow to plot, but crisp, without any CRT fuzz or flicker. It was intrinsically a graphics display, so text could be freely mixed with pictures and displayed anywhere on the screen, not just at the bottom. In addition — crucial for language teaching — the terminals could handle user designed foreign language fonts. Graphics terminals existed in 1973, of course, but they were usually dedicated to printing out graphs, and were certainly not the standard for interactive work. For anyone accustomed to dealing with text oriented CRT terminals, the display flexibility was wonderful.

And for students still accustomed to communicating with computers by way of punch cards, batch jobs, and teletypes, the interactivity was wonderful. No more waiting to type a whole line of characters without anything happening until RETURN was pressed. On the new PLATO terminals, every key press was processed right away and response was — usually — immediate.

Maintaining that degree of interactivity on a single mainframe system would prove to be one of the main preoccupations of the system designers. In 1973, there were still only a handful of users on PLATO IV, but the system was expected to eventually support over a thousand terminals, with perhaps half that number actually working simultaneously. This was time sharing on a scale never before attempted, and exceptional ingenuity in hardware and software design would be required to make it work. As a hardware base, the designers intelligently chose the CDC (Control Data Corporation) 6400, a top-of-the-line computer originally designed for scientific and military work. The 6400 featured a processor that for the time was blazingly fast, a 60 bit word length, which allowed data to be moved around in large chunks, and something called extended core storage (ECS), a term with which all PLATO IV users were destined to become exquisitely familiar. Originally designed to facilitate large scale matrix manipulations, ECS was a kind of huge add-on to main memory. Programs could not be directly executed in ECS, but could be moved almost instantly from ECS to main memory and back. On any time sharing system, each user in turn gets to use the
processor for a small fraction of each second while all other users wait. No machine at that time had enough main memory to store 500 simultaneous programs, yet moving them temporarily to disk storage would have been far too slow. PLATO IV resolved this quandary by using ECS as a place to store programs while they were on hold.

This solution, though technically brilliant, had some unfortunate eventual consequences. Since no other computer manufacturer provided ECS, PLATO IV would run efficiently only on CDC hardware. It would not be easily transportable to other machines.

The Language Projects Get Underway

During my first months on the project, our huge PLATO classroom/laboratory was still empty, and the offices themselves were only half furnished, with a few PLATÔ terminals laid out on office desks and tables. Keith, full of nervous energy, was directing programmers how to get on with a number of projects. Since the shift to new hardware and software unfortunately rendered all the PLATO III materials obsolete, much work was devoted to attempts (only moderately successful) to convert language courseware from the PLATO III environment so that it would run on PLATO IV. Beyond this, however, there was no shortage of new projects, because the various language departments were eager to involve themselves in an exciting new technology.

My first project was to produce a set of German reading materials, and I immediately experienced some of the advantages and limitations of PLATO IV as an environment for designing language instruction. The originators of PLATO had realized, of course, that simply providing hardware did not constitute an instructional computing system: tools for designing courseware (or "lessons" in the standard PLATO terminology) were also required. PLATO III already provided such tools in the form of a new programming language, TUTOR, with general purpose capabilities, but specialized for doing instructional tasks (Sherwood 1977). The notion was that instructors would wish to make their own PLATO lessons somewhat in the same way that they produced lecture notes, class handouts, and textbooks; hence, much emphasis was put on making TUTOR easy for the nonprogrammer. Certain things can indeed be done very easily in TUTOR, as Figure 1 illustrates:
These few lines suffice to present a translation item, collect an answer, judge it for correctness, show vocabulary help if the student has made three or more errors, and require the student to keep trying until the answer is correct. Errors in spelling and word order are marked automatically. The strengths of TUTOR - a simple, intuitive format that was easy to learn and allowed efficient handling of display, input, and standard sequencing - are apparent in this example. Also visible, though perhaps less evident, are some pedagogical prejudices. At the time TUTOR was devised, question-answer-feedback sequences were the standard for CAI, and this bias is incorporated to some extent in the language design. Anyone can learn in ten minutes how to write the kind of instruction displayed above. The more one deviates from this methodology, however, the less convenient TUTOR becomes.

There were other, less obvious problems. First, as suggested by the example above, TUTOR encourages incorporation of data directly into the program. Second, its simple command/tag format, though easy to learn, later made it difficult to extend TUTOR to contain the kind of structured programming syntax found in PASCAL. And finally, there was the absence of the subroutine libraries supported by languages like PASCAL,
which allow programmers to borrow the subroutines needed to do common tasks. This meant that our programmers couldn’t easily reuse one another’s work, so they largely reinvented the wheel for each new project.

In its early days on PLATO IV, TUTOR showed some big gaps. My German reading project was designed to present phrases at a set rate of speed. I had expected to write a single pacer program to pull in the text from a database. To my dismay, I discovered that the file reading and writing facilities standard to other systems were lacking in PLATO. There was a primitive database mechanism called "common storage," the system staff kindly informed me, but there was at present no way to edit information into it. My first task thus became writing an editor to create the data that I would use for the reading program! A job I had expected to take a few days stretched out to weeks. This, and most other holes in TUTOR, were eventually filled, but it often took a long time, and we spent a lot of energy writing programs to work around these limitations.

Working on the system in the early days was always an adventure. The CERL staff was constructing the PLATO IV system software and the TUTOR language at the same time that we constructed our language courseware, and to say that the system was unpredictable is putting it mildly. In my memories of these earliest days of PLATO IV, it is the problems and difficulties that stand out in dramatic relief. To be fair, however, given the immense scope of the task, the CERL staff did a remarkably good job of building the PLATO IV system smoothly, with minimum inconvenience to users.

Nor did our early problems noticeably discourage language faculty from exploring the new technology. Part of the NSF grant was earmarked to fund materials development projects in a wide variety of disciplines, with a view to showing the range of things that could be done with PLATO facilities. This support was distributed to various campus departments. In the Foreign Languages Building, two units were involved. On one side of the basement, the Language Laboratory served as a sort of technical support center, with Keith Meyers, the Director, acting as an informal coordinator for projects undertaken by the language departments. On the other, where Bruce Mainous directed the Unit for Foreign Language Study and Research, Fernand Marty was engaged in a massive project to explore how PLATO technology could be applied to the teaching of language. The two units were independent but, being both engaged in PLATO development, cooperated closely.
Several language departments had launched projects early on to develop substantial PLATO curricula for Hebrew, ESL, Russian and Latin. All these projects were housed in the Language Laboratory, which served as the nerve center of PLATO language work.

The most massive materials development effort, however, and that which had the greatest exposure and lasting influence, was Fernand Marty’s. Marty had first worked with Bell Laboratories and later, as a professor of French at Dalhousie and Middlebury in the sixties, had done extensive work on language pedagogy, culminating in a four semester elementary French course-textbooks accompanied by exercises in programmed text format. He had come to Illinois specifically to explore how the new PLATO technology could extend his prior pedagogical work. As a first step, he had set about reimplementing on PLATO IV the curricula he had already developed. He was aided in this by Robbie Ariew and later Susan Campanini, both doctoral candidates in the French Department.

There was, in fact, no shortage of willing assistants. Access to one of the new PLATO terminals was a mark of prestige. A constant stream of graduate students came down to our bare little suite of offices to chat with the programmers (who were often their fellow graduate students) and look at the lessons taking shape on the orange and black screens. The students were excited: for the first time, they saw computers that could do something useful for language teaching. They were accessible, not locked away in a computer site on the engineering campus, but right next door to their offices. After half an hour of training in TUTOR they could write a simple unit of instruction. The arcane preserve of scientists and computer science majors was suddenly open to humanities students. For many, their first hands-on session with a PLATO IV terminal was a formative encounter with computer technology.

Faculty came to the Laboratory by appointment and received more formal demonstrations of the technology. Their reactions were more qualified. A few went back to their offices and proceeded with business as usual. Some were enthusiastic. Many decided, whatever their reservations, that it was a good idea to be involved in a project that might change the face of teaching on campus, and started on plans for some experimental lesson development. The Laboratory had money to provide an assistant to any project, and in addition provided technical support from its own staff of experienced TUTOR programmers. In this way it evolved that most development projects consisted of a faculty director, a member of the Language Laboratory staff to
help out with difficult programming and perhaps provide instructional design
guidance, and a graduate assistant to do nontechnical production work. Faculty
involvement ranged from intimate to Olympian, so that the de facto team often consisted
of the graduate student and a programmer.

The Search for Instructional Models

Before many months had passed, the pace of lesson writing picked up, and language
students began to walk into our PLATO site expecting to use the lessons we had
created. We quickly discovered that the problems encountered when making prototype
courseware for admiring colleagues and a few pilot subjects are very different from
those of an operation responsible for cranking out hundreds of lessons for thousands of
students. We groped for ways to make our lesson authoring procedures less labor
intensive and more efficient.

One obvious source of inefficiency was duplication of programming efforts: Hebrew,
Spanish, French, and Hindi, for instance, all wanted similar drill activities. Keith
Meyer’s proposed solution was to write a single "driver program" (today one might say
"template program") for each type of activity, usable for any natural language. We
immediately encountered a fact that has proved the nemesis of template and lesson
authoring systems from that day to this: no two language instructors ever agree
completely on how courseware should work. Some instructors wanted the correct
answer to be displayed after one try, some after two tries, others not at all. One teacher
wanted problem items saved for mandatory review. Another thought that review
should be optional, and still others considered review worthless. And so on and on.

The technical people in the Language Laboratory tried hard to live by the rule that
language teachers should have the final word about language pedagogy. We thus
eschewed the tempting solution of telling an instructor, "This is how it works and if you
want to use it, you’ll have to live with it." Instead we tried to build flexibility into the
template programs by making the various features into software switches that could be
turned on or off. But after working with half a dozen instructors we had accumulated so
many parameters (language font, screen position of the prompt, scoring method,
conditions for showing help, to name a representative sampling from a drill activity)
that even the programmer who created them couldn't keep track of all the options. In essence, they constituted a sort of haphazard mini-programming language.

We never did adequately resolve this issue of generality versus flexibility, but eventually fell into a sort of compromise. For each new courseware project we started with templates developed by previous projects. These were customized as required. As a result, much of our courseware was eventually based on a family of more or less similar templates. This approach necessitated a messy proliferation of versions, but really did keep programming time to a minimum. In fact, given experienced developers, it could be extremely efficient: Susan Campanini and I once created the online materials for an entire first year Spanish curriculum, including instructional management, in two weeks of intermittent work. Lessons appeared online virtually as fast as we could type in the lesson content from paper.

Templating also helped with lesson maintenance. The amount of effort needed just to keep things functioning came as an unpleasant surprise. To begin with, I naively assumed that one wrote a lesson, tested it a bit, released it for student use, and forgot about it. I soon learned differently, and came to appreciate the software engineering maxim that no program is ever really finished. Students doing tomorrow's Russian homework were disgruntled by ugly execution errors that required immediate repair. A German instructor needed to change a menu design that looked good in prototype but just didn't work in practice. Or the system had just improved the character set software and the Hindi character set had stopped working. Most of our programmers, even those with some computer science background, lacked an appreciation of how essential detailed documentation and clear program structure were when it came time to fix these problems. Here too, the practice of using templates came to our aid by imposing some standardization of program style and structure.

It would nevertheless be wrong to believe that all the courseware produced under the guidance of the Language Laboratory had the same look and feel. If templating gave our largest curricula a certain "Language Laboratory" style of programming, every project made its own innovations, and the result was a plethora of activities and designs. Over the years, we explored many CALL design problems such as the use of hierarchical menus, the organization of student controlled help systems, the proper use
of automatic review features, and the possibilities of matching-driven grammar error
diagnosis, feedback, and error markup. We experimented extensively with multilingual
writing systems, touch driven selection activities, combination of graphics and audio
with text material, and performance history and data basing. The solutions to these
problems eventually provided a rich design vocabulary for PLATO CALL. Indeed,
before the appearance of true multimedia courseware, I never saw a micro based CALL
design we hadn't already approximated on PLATO. Since the secret of writing good
PLATO courseware was familiarity with this design tradition, we began teaching a
course entitled "Computer based Foreign Language Teaching" that incorporated the
relevant material.

However, TUTOR did lack features needed for some language courseware designs. Our
more imaginative instructors wanted to sort text and search it, have the computer read
it aloud, build dictionaries, make concordances, identify individual sentences and
phrases, match keywords, analyze words into roots and endings, generate and parse
sentences, and translate from one language to another. To do even part of this would
have required powerful string processing facilities like those found in the SNOBOL
language, together with some of the list processing features of LISP. We felt that the
system staff should extend TUTOR so that we could do computational linguistics.

When we had built up a sizable wish list of features we wanted, Bruce Sherwood
agreed to come over to the Language Laboratory to discuss matters with the assembled
Language Projects staff. We described what we needed. Bruce explained that a few of
the things we asked for could probably be added to TUTOR, but the more advanced
features were simply impractical. "TUTOR isn't LISP and it's not going to be," he
commented. The language people, at least, were not satisfied and left this meeting
rather chagrined. Some of the advanced lesson designs that we had been brainstorming
could be done only with great difficulty: the majority, not at all. Building language
courseware on PLATO IV was going to be mostly a matter of refining drill and tutorial
designs. A handful of sophisticated language activities were eventually implemented,
for instance an elegant parser driven sentence interpreter developed by Bob Yeager's
Elementary Reading Project and a French text-to-speech system that Fernand Marty and
I built, but these represented tours de force of TUTOR programming rather than readily
deployable techniques.
The whole episode simply reflected a structural flaw built into the PLATO IV project. System control of TUTOR development meant that TUTOR could be optimized for PLATO hardware. It also relieved lesson authors of the impossible burden of building a programming language, and it guaranteed a homogeneous, stable programming environment. In these ways central control was valuable. But in more subtle ways, it retarded work on instructional design, because the people building TUTOR were only loosely coupled to the people doing actual instruction. The system staff was answerable not only to the Foreign Languages Project, but to physics, veterinary medicine, accountancy, and chemistry; their time and resources were limited, so when adding to TUTOR, they naturally favored generic features useful to everyone. For lesson designers this loose linkage between instructional needs and software design was a frustration all along the way.

Although the term “multimedia” was not yet popular, the designers of PLATO IV provided multimedia capability in the form of two peripherals: the EIS Random Access Audio Machine and a built-in projector to allow for the display of microfiche images. The audio machine was a sort of combination of a phonograph and a floppy disk drive. It used 12” diameter floppy disks, each of which could hold 256 tracks of prerecorded analog (not digital) information. It was also possible for the student to use a microphone to record his own voice onto the disk. A motor rotated the disk at a fixed velocity and electronics caused the reading or recording head to fall onto the disk, but to make the unit less expensive, the head was moved radially by a blast of compressed air. This meant that every EIS machine had to have a compressed air source attached. In our PLATO laboratory, air lines were built into the floor; for remote demonstrations, one had to lug along not only the machine itself, but also a compressed air tank. Fernand Marty’s close work with CERL soon led to a second, improved version, and later to a third version, completely electronic, with higher fidelity, and quite reliable. In spite of the difficulty of recording and copying audio disks, many of our materials eventually incorporated audio into dictation, listening, and pronunciation activities, the latter including an entire course in French intonation.

In contrast to the audio, which was a useful if occasionally cantankerous component of PLATO hardware right from the start, the microfiche projector was just plain cantankerous, and proved to be a good concept that was never fully satisfactory in practice. Producing fiche was cumbersome and expensive, and the positioning and
focusing machinery was so unreliable and the color distortion so serious, that ESL, like most other projects, eventually gave up on the fiche projector.

A third peripheral, by far the most widely used, was the PLATO touch panel. It divided the screen into a relatively coarse 16 x 16 matrix of cells, and responded to a finger touch anywhere on the screen, thus providing much of the same functionality as a mouse. It was tremendously popular and eventually became standard equipment.

From Laboratory to Classroom

By fall of 1976, the trickle of students coming to our PLATO Laboratory had grown to a steady stream, and the stream soon grew to a flood. Long lines of students with assigned PLATO homework were congregating outside the door, and we were faced with a genuine management crisis. Initially we had expected to get by with a single lab monitor. This calculation was invalidated by the complexity of our scheduling and by the existence of PLATO notes and PLATO games.

PLATO IV was home to what was probably the first computer game culture. Clever programmers bored with programming drill materials first tried their hand at implementing ticktacktoe, solitaire, hangman, and other individual games, then moved on to interactive, multiplayer ones. Probably the most popular of these latter was Empire, an arcade style game loosely based on Star Trek. In the days before arcade games, Empire rapidly developed a huge following, and aficionados were soon competing for PLATO terminals.

The system staff was eager to nurture a "PLATO community" united by on-line communications. To aid this growth they made available an extensive set of communications software including a mail facility called personal notes. Notes could be exchanged by individuals and be posted in bulletin board facilities called "group notesfiles." Since PLATO sites were scattered nationwide, users eagerly exploited this cost free way to contact friends and meet strangers. Soon many of our terminals were occupied by students writing notes and scanning notesfiles for hours. Our priority use system gave preference to language students doing assigned work, then to language students in general, followed by anyone doing "real work," followed by recreational users. But the gainers and notesfile readers simply would not stay out when it was
busy, and neither would the chemistry and physics students, crowded out of other sites on campus, who came to the Foreign Languages Building in desperation. There were shouting matches over possession of terminals.

The PLATO system staff decided that a system level software solution was required. For what turned out to be a major programming job, we were fortunate to get two excellent programmers, Tom Schaefges and George Traynor. After innumerable meetings and about four months of intense work, we had a working package that brought the situation under control. Game players and low priority users were signed off automatically (after a warning) as soon as higher priority users requested space, and if they persisted in violating our priorities, they were locked out permanently.

The battle between "legitimate" and recreational users went on, but the crisis was over. We continued to refine our program for several years, until a single monitor was able to manage the entire site almost effortlessly. The site-management system was by far the most sophisticated PLATO software ever built in the Language Learning Laboratory and represented an unqualified success. Even today, our microcomputer networks lack a set of site management tools to equal what PLATO provided in 1979. Work on site control was also a fine example of how a software development project should proceed: the extensive dialog between users and a responsive system staff produced exactly what users required and served as a model for further PLATO development.

As lessons became more numerous and curricular structures more complex, we badly needed some way to control on-line the sequence of lessons that students followed. Instructors needed to keep accurate records of what lessons their students had done, what score they had received on each completed lesson, and what they were currently working on. The PLATO system staff responded with an elaborate set of routing commands that could be used to keep records and control student progress. It had some very powerful capabilities; for instance, a router program could be made to examine a student's scores on previous lessons and on that basis send her to the appropriate activity. A few groups, among them the Indiana University English Project (Haught, Oates, Sweeney, and Koester 1981), set up complex routers that exploited these facilities to their fullest. Interestingly enough, however, none of the language projects at Illinois ever used fully program controlled routing successfully. The PLATO system
maintained course rosters and basic usage statistics for individual students (for example, total number of hours signed on). To supplement this, our routing programs kept track of which lessons had been completed and the score achieved on each of them. Usually, it was all the information the instructors wanted.

PLATO provided powerful data collection facilities allowing a lesson to output student responses verbatim and record student activity almost at the keypress level; in fact, any information the lesson author wanted could be saved into data files. Here, in principle, was a perfect tool for doing experimental work on language learning. In practice, outside of an elegant study of the learning strategies of Russian students (Curtin, Avner, and Provenzano 1981), and some work by John Haller and me on error frequencies in French and German (presented at the Third Annual Conference of the Computer-Based Language Instruction Consortium in Cincinnati in 1980), I cannot recall a foreign language project that exploited it for experimental purposes. In part, this was because language instructors shied away from the vast amounts of raw data generated, and in part because PLATO provided no resident statistical packages for analyzing the data on-line once they were collected and formatted. The most frequent way of processing datafiles was to glance over a printout. Besides, in sharp contradistinction to theory driven CALL development like the Stanford elementary reading project (Atkinson 1974), PLATO language courseware was built by classroom instructors rather than researchers. This meant that immediate utility guided most PLATO development, with theory and experiment playing a negligible role.

From Classroom to Marketplace

By 1977 the Language Laboratory had merged with the Unit for Foreign Language Studies to form the LLL (Language Learning Laboratory), of which Bruce Mainous became Director and I, Acting Associate Director. It was an immensely productive time for the new unit. CERL had constructed an admirable instructional management system and almost daily improvements to every aspect of the PLATO system allowed courseware implementation to proceed smoothly. Curricula for Hindi and Swahili were under construction. After long struggles with the display of Chinese characters, C. C.
Cheng, a computational linguist, had built a series of Chinese lessons and was experimenting with Chinese tone recognition. Others were developing materials for German phonetics, Swedish, and Spanish. Nor were we the only site producing language courseware. Schools as far apart as the University of Delaware, the City Colleges of Chicago, and the University of Hawaii created lessons for English literature and grammar, Latin, and Chinese (Hart 1981).

This burgeoning new area of endeavor seemed to call for some sort of organization. In August 1978, Bill Oates at Indiana University hosted the inaugural meeting of the Computerized Language Instruction Consortium (CLIC), a regional association dedicated to aiding the growth of computer based language instruction on PLATO. After two days spent viewing demonstrations of ongoing work across the entire range of English and foreign language instruction, we all gathered on the patio of a Mexican restaurant, watched the sun set and stars come out over Bloomington, and talked enthusiastically about our development plans and the future growth of PLATO.

In the latter part of the seventies, our PLATO Laboratory was delivering over 50,000 student hours of language instruction every semester, plus another 50,000 hours in other curricula. New sites were being added into what appeared to be the beginnings of a nationwide PLATO network. A steady stream of international visitors came to the LLL. NSF had examined CERL’s work and returned a highly favorable evaluation. PLATO IV was clearly a success, and many members of the PLATO community thought it was time to expand the project into the worldwide educational computing network foreseen by Don Bitzer. Such deployment could not be done within the context of the NSF project. A commercial partner was required, and Bitzer and others felt that they had found a suitable candidate in the form of Control Data (CDC). The CDC people had an obvious interest in marketing PLATO software, since it ran exclusively on their hardware, and they seemed to want to break into the education market. After some negotiation, the University in 1976 sold CDC the entire PLATO system as a package, including the "PLATO" name, the system software, and virtually all of the courseware that had been developed up to that point. In return CDC promised to market PLATO and support continued development work in cooperation with CERL.
This agreement did not sit well with many PLATO lesson authors. Although the University technically owned the copyright to nearly all PLATO materials, some authors felt that the courseware had been sold to CDC too cheaply or were disturbed that they had not been consulted more extensively. Others resented the fact that they had not been allowed to do their own negotiating individually. Whatever the justice of these views, they created tension and led to some permanent ruptures within the PLATO community. When it further became obvious that any software royalties from CDC would be minimal, potential new lesson authors shied away and old authors began to look for other alternatives.

Working with CDC brought a fairly open academic development project into contact with a more rigid corporate culture. CDC restricted its corporate notesfiles and forbade its employees to talk about corporate matters, even those concerning PLATO development. CDC established standards for screen display and set up an internal publication process to prepare courseware for sale. All these things were perfectly reasonable from a marketing perspective, but they created friction just the same. Most of our authors had given careful thought to the instructional design and screen display of their lessons and weren't used to having their judgments second-guessed. Sitting down with a four page list of editorial corrections issued by an anonymous technical person in CDC's Minneapolis publication offices was bound to be unpalatable, and it could be plain irritating when distance, impersonality, and differences between educational and commercial perspectives were added in. A number of our language faculty decided to decline to publish their courseware rather than put up with the review process. Later CDC on its own initiative redid some of the mainframe lessons as micro courseware that was marketed under the PLATO label. In some cases, lesson features had been scaled back or dropped in order to make the programs workable in a micro environment, and there was more grumbling among lesson authors that the PLATO name was discredited by association with inferior courseware.

Some systems were sold in the United States and abroad and some schools and corporations did subscribe to PLATO services. Ultimately, however, PLATO did not sell very well. Partly this was simply a matter of money. A full system could run into the millions. Buying services avoided this outlay, but incurred substantial telecommunications charges. Meanwhile, the U. S. economy entered a slump. In fact, I never showed off our language materials at a conference or workshop where a teacher
didn't say, "Your lessons are wonderful, and we'd love to use them, but how can we afford them?" CDC, which had a history of dealing with well-heeled corporate and government clients, was hardly the ideal company to do innovative marketing to these cash starved teachers, principals, and school boards. And of course, for CDC, PLATO was only one venture among others—one which was expendable should it prove unprofitable. On the whole, the University of Illinois/Control Data venture was not a happy one, and it would be fair to say that it disappointed expectations on both sides.

After the CDC venture stalled, CERL tried other routes to making PLATO IV commercially viable. The University Communications Corporation was established specifically to market PLATO (now renamed NovaNET) services to educational institutions. While making PLATO widely available was clearly an important goal, preoccupation with technology transfer took time and attention from what CERL could do best, namely, designing innovative hardware and software. The leap from PLATO IV to the next generation of educational computing technology did not take place.

The floundering of the commercial venture with CDC, the end of NSF money, campus budget cuts, and the absence of a major new design initiative at CERL eventually slowed PLATO courseware development to a trickle at Illinois. Within foreign languages, however, an even more important development was the decline of audiolingualism and the concurrent rise of the communicative competence movement. PLATO was an ideal vehicle for delivering the drill and practice emphasized in the audiolingual approach, but what it might contribute to a syllabus that emphasized communication rather than form was unclear. As "drill and kill" exercises came into disrepute, instructors saw existing PLATO courseware, and PLATO itself, as increasingly irrelevant to their teaching methodology. By the time C. C. Cheng replaced Bruce Mainous, who retired as LLL Director in 1985, use of our language courseware was declining. Though two more large courseware projects were done—a package for Agricultural Spanish and a Chinese curriculum— the handwriting was on the wall. If LLL was going to continue to explore CALL, new instructional designs and new money would be essential. Like many others, we turned to the microcomputer to find them.
The Microcomputer Challenge

The first reaction of PLATO devotees to microcomputers was to discount them. Next to the computing power, storage, software, and communication facilities offered by a mainframe time sharing system, an Apple 11, with its 48K memory, 256K floppy disk, and 6MHz processor didn’t look like much of a threat to PLATO preeminence. A room full of stand-alones could not communicate, there was no on-line library of lessons, no central place to store performance data, no facility for doing foreign language fonts, and no way to mix text and graphics. A little experimentation at reimplementing some of our LLL drill templates in BASIC showed how much faster and more powerful TUTOR was for programming CALL. The micros did have several big advantages, however: they were cheap, they were not restricted to running TUTOR, and they entailed no telecommunication expenses. Soon the Macintosh and IBM PC offered better display and more computational power than a PLATO terminal; there was no possibility, for instance, of running high resolution 3D graphics or a state-of-the-art word processor in the PLATO environment. AppleSoft Basic was supplemented by Pascal, and then HyperCard, Lisp, and Prolog appeared on micros. TUTOR, which had seemed so convenient in the early days of PLATO, felt distinctly awkward after one had used HyperCard a while.

CERL wavered on how to respond to microcomputing. For a while, PLATO development flirted with PLATO V, an “intelligent” PLATO terminal with a microprocessor and local memory. One could download TUTOR code to one of these terminals and run it locally, thus increasing processing power, but the machine was expensive and it never became the standard PLATO terminal. LLL never acquired a single one of them. Paul Tenczar, who saw the implications of micros early on, left CERL to re-implement TUTOR (eventually reborn as TenCORE) on various microcomputers. Later Bruce Sherwood and others at CERL became convinced that small scale PLATO systems could make PLATO economically and technically competitive: a cluster of microcomputer stations would be locally networked around a server machine that would handle communications and act as courseware library and instructional manager. TUTOR programs would be downloaded to individual stations for execution. Not only would this be relatively inexpensive, but the micros could run word processors and other software when not executing TUTOR lessons. This idea
elicited some excitement in the PLATO user community. Several clusters were actually created and used in Japan, but when Sherwood left for Carnegie Mellon University, CERL’s efforts in this direction languished. When Don Bitzer departed in 1989, CERL apparently still remained committed to the viability of mainframe PLATO.

By then, most primary schools, high schools, and colleges had turned to microcourseware (most of it none too good by PLATO standards) for computer based education, and our language faculty had discovered that micros were indispensable for word processing. Instructors were enthusiastic about international video, which provided more cultural and communicative impact than traditional materials, and were experimenting with ways to incorporate interactive video into microcomputer programs. Digitized speech became a viable way of providing audio. In short, the microcomputer revolution was accomplished. As the old PLATO terminals in our laboratory began to lose the orange dots on their plasma panels, we replaced them with Macintoshs and IBM PCs, using CERL-developed software and the campus network to make them simulate PLATO terminals when required. Very few Illinois students were assigned PLATO language materials, but every semester, a good many of them still came to our newly named Microcomputer Laboratory on their own to use the PLATO lessons. Others dialed them up through their home computer modems, and NovaNet subscribers all over the country could access them directly. Considering that most of our language materials had not been updated for nearly ten years and were no longer connected to current syllabi or textbooks, the level of use remained surprisingly high and testified to the quality of their design.

In June 1993, the University of Illinois, struggling with budget restrictions, announced that, as part of an ongoing internal reallocation process, PLATO services would in the future be purchased from NovaNET. In addition, CERL, which had designed PLATO IV and guided its development at Urbana for more than a quarter century, would be dissolved, thereby marking an official end to the PLATO era.

The Legacy of PLATO IV

Looking back at the PLATO Languages Projects, an undertaking that engaged dozens of talented people for several decades, it is natural to ask what lasting contributions all this time and effort made to CALL.
First, CALL on PLATO succeeded as mainstream instruction, and on a large scale. Hundreds of thousands of hours of language instruction in a dozen languages were delivered to many thousands of students throughout the United States over a period of twenty years. Our language instructors stopped using PLATO courseware not because they thought it was bad or ineffective, but because they thought that new technology and new instructional designs could do even better. And, in fact, PLATO foreshadowed much of the newer multimedia technology. Our lessons didn’t have digitized speech, but they did have random access audio, and that gave us a beginning with audio based designs. As it turned out, the fiche projector didn’t work very well, but it did point the way to what could be done with a visual component. And the touch panel was standard on PLATO terminals long before microcomputers sported mice.

At a time when "Bitnet" was still a meaningless term to most language faculty, PLATO’s communication facilities allowed the development of a PLATO user community that included developers, instructors, and students. Students wrote notes to one another and to their instructors; instructors in Urbana and Hawaii could correspond about teaching French or Japanese, and everyone could read about and comment on subjects of interest in PLATO’s public notesfiles. For a great many users, ready access to this huge community was their most important PLATO experience. In spite of ventures like the use of Hebrew group notes, this capability was underexploited for CALL purposes, but the potential was there. Only recently have micro environments supported this much interactivity.

PLATO taught us about instructional management. We found out that management tools, particularly a well organized lesson library with a well organized catalog and uniform access procedures, together with the ability to track student use, are essential to systematic, large scale employment of CALL. PLATO built a superb instructional management environment that today's micro networks don't even approach. In consequence, instructors trying to assemble CALL courseware have to locate and sort through a hodgepodge of materials that are frequently unrelated, uncataloged, and sometimes incompatible with local micro environments.

At the time PLATO IV came on-line, computer systems normally communicated with users in some variety of command line gobbledy-gook. PLATO was perhaps the first large system intended to be used mainly by non-experts, and thus PLATO philosophy
dictated meticulous attention to interface design. And the display tools needed for implementation were provided. Since language students often had even less than the average computer experience, LLL tried to assure that every lesson we wrote had a clear, intuitive, and internally consistent interface. Where feasible, we encouraged a consistent style of interface across projects too, so that students would have less relearning to do. We have tried to refine and carry this tradition on in our work with increasingly complex microcomputer interfaces.

Within limits imposed by TUTOR, we intensively studied the instructional design of drill and tutorial, on-line help systems, error diagnosis, review, performance history, audio material, item data bases, foreign language writing systems. Some of these features, now appearing as innovations in micro based CALL, were standard in PLATO lessons twenty years ago. The PLATO CALL design vocabulary can be carried over almost intact to today’s micro environments, requiring expansion only with the recent advent of multimedia, hypertext, and "intelligent" CAI designs.

To a considerable extent the legacy of PLATO has been carried forward by the remarkably talented community of individuals who worked on the system and our CALL projects; when they moved on, a surprising proportion of them continued to work on educational computing, gravitating to IBM, CDC, various academic positions, or setting up on their own as software entrepreneurs, so that their PLATO experience has passed into the mainstream of courseware development.

Of course there were important respects in which PLATO IV fell short. From the viewpoint of CALL perhaps the greatest frustration was the fact that developers could program only in TUTOR and run only TUTOR programs. The necessity of programming in TUTOR cut us off from most of the computational linguistics required by more sophisticated CALL designs and largely restricted us to variations on drill and practice. Inability to run non-TUTOR software on PLATO was equally problematic, since it isolated PLATO users from useful software (such as statistical packages) on other systems, and allowed PLATO to ignore software developments in the outside world (for many years PLATO intentionally operated without any network links to non-PLATO machines). Conversely, reliance on TUTOR later made it infeasible to transfer PLATO's courseware library to other machines. And, as it turned out, TUTOR
simply could not evolve fast enough or far enough to compete with subsequently developed programming tools. Nor did PLATO's mainframe design give it a very good base for dealing with emergent multimedia technologies. In retrospect, a great strength of PLATO IV—its self-enclosed and highly integrated design—turned out to be a great weakness too, because various aspects of the system could not be adapted piecemeal to the decentralized computing environment of the eighties and nineties.

NOTES

By one account, the name "PLATO" originated as an acronym of "Programmed Logic for Automated Teaching Operations." Another version has it that PLATO was chosen for its association with the philosopher, and that the acronym was invented later to satisfy people who asked "What does it stand for?" With the sale of the University of Illinois PLATO system to the Control Data Corporation, the word "PLATO" became a registered trademark owned by Control Data. In this paper, it is used to refer to the PLATO system prior to that time, and as a convenient abbreviation for referring to the University of Illinois PLATO system as developed subsequent to that time. NovaNET is a registered trademark of the University Communications Corporation.

No single viewpoint can do justice to a complex undertaking like the Illinois PLATO Foreign Languages project, which involved dozens of people and continued for many years. This paper attempts to recount the PLATO CALL experience from my own perspective, and is in no sense intended as an official or exhaustive account. In particular, language courseware development at other PLATO sites has been largely passed over. Many individuals at LLL and CERL besides those named here helped to make the Illinois project a success; I regret that there is not sufficient space to give all of them the credit they deserve.

I would like to thank Cynthia Cochran, Bruce Mainous and David Manton for their helpful comments, and Susan Campanini for her thoughtful criticisms and the benefits of her accurate memory for PLATO history.

AUTHOR'S BIODATA

Robert S. Hart is a lifetime resident of Illinois, holds the B.A., M.A., and Ph.D. in Psychology from the University of Illinois, with concentration in mathematical modeling and cognitive psychology. Assistant Professor of Humanities and Associate Director of the University of Illinois at Urbana-Champaign Language Learning Laboratory since 1978, he has directed numerous projects for language courseware development both on PLATO and various microcomputer environments. His current interests are multimedia instructional design and error tolerant parsing.