

AUCS / TR9111

Beauty and the Beast: new approaches to teaching
computing for humanities students at the University of
Aberdeen.

Simon Holland and Gordon Burgess †
Friday, 24 May, 1991

Department of Computing Science
University of Aberdeen
King's College
Old Aberdeen,
AB9 2UB, Scotland, UK.

Abstract

This paper reports on the history and development of a new undergraduate course teaching Computing for Humanities Students at the University of Aberdeen, and assesses some new teaching approaches developed on the course. It is noted that teaching computing to humanities students has sometimes been viewed with suspicion by both Computer Science and Humanities Departments. The two camps tend to fear, for different reasons, that issues and practices important to their disciplines will be compromised or watered down. Humanities students are often lacking in enthusiasm for computers. This paper describes an attempt to reverse any such attitudes on the part of staff and students and to take undergraduates considerably beyond mere word processing and computer literacy. Various methods and techniques used in the course are presented and their value assessed. The importance of using a consistent computer interface to helping students form a stable conceptual model of computers is considered. The value of teaching more about Human Computer Interaction and Artificial Intelligence than is usual in Humanities Computing courses is considered. A number of lessons are drawn from the course.

Key Words

Education, Human Computer Interaction, Artificial Intelligence, Computing, Humanities.

This paper appeared as

Holland, S. and Burgess, G. (1992) Beauty and the Beast: new approaches to teaching computing for humanities students at the University of Aberdeen. In *Computers and the Humanities* 26: 267-274.

† *Department of German, Kings College, University of Aberdeen, Scotland.*

Beauty and the Beast: new approaches to teaching computing for humanities students at the University of Aberdeen.

Simon HOLLAND

Department of Computing Science, Kings College, University of Aberdeen, Scotland, AB9 2UB.

Tel: 44 224 272284 Fax: 44 224 487048 email (Janet): simon@uk.ac.abdn.cs.

Gordon BURGESS

Department of German, Kings College, University of Aberdeen, Scotland.

Tel: 44 224 272494 Fax: 44 224 276730 email (Janet): g.burgess@uk.ac.aberdeen.

Abstract

This paper reports on the history and development of a new undergraduate course teaching Computing for Humanities Students at the University of Aberdeen, and assesses some new teaching approaches developed on the course. It is noted that teaching computing to humanities students has sometimes been viewed with suspicion by both Computer Science and Humanities Departments. The two camps tend to fear, for different reasons, that issues and practices important to their disciplines will be compromised or watered down. Humanities students are often lacking in enthusiasm for computers. This paper describes an attempt to reverse any such attitudes on the part of staff and students and to take undergraduates considerably beyond mere word processing and computer literacy. Various methods and techniques used in the course are presented and their value assessed. The importance of using a consistent computer interface to helping students form a stable conceptual model of computers is considered. The value of teaching more about Human Computer Interaction and Artificial Intelligence than is usual in Humanities Computing courses is considered. A number of lessons are drawn from the course.

Key Words

Education, Human Computer Interaction, Artificial Intelligence, Computing, Humanities.

Introduction

Teaching Computing to Humanities Students is sometimes viewed with doubts by both Computer Science and Humanities Departments. Some computer scientists suspect that such courses purvey a tepid, watered-down version of their discipline. Some humanities scholars are sceptical of whether mechanical procedures and devices can offer much of real value to the proper study of the humanities at undergraduate level. Students and teachers of the courses sometimes have their own doubts: some students seem to be interested in little more than improving their employment prospects by simple association with computers; teachers are sometimes frustrated by humanities students' indifference, hostility or lack of enthusiasm for computing in the Humanities. In setting up CS1002, Computing for Humanities students, in its first year as a full graduating subject at the University of Aberdeen, a key aim was to turn all of these attitudes, where they existed, upside down. In this article we examine the extent to which the course has succeeded and the lessons to be drawn.

Background

First of all, we will give a little historical context. Although this particular course is new, the teaching of aspects of computing to humanities students has a relatively long history at the University of Aberdeen. In 1981, an undergraduate course for German honours students was introduced, called 'The computer and the literary text'. The coursework was all done using the University's mainframe computer. Students had to learn to use a time-sharing system, and to create, edit and manipulate files (using a text editor rather than a word processor), before moving on to an introduction to FAMULUS, SNOBOL, and concordances (Oxford Concordance Program). The course, one option among many for this particular group of students, proved extremely popular over the years, even if not always for what we might have regarded as the 'right' reasons. Many students felt that listing a course with the word 'computer' in the title gave them an edge in their job applications. However, it was clearly not an interdisciplinary course in humanities computing: it was restricted to honours students of German; all the examples and the texts for manipulation and analysis were taken from German sources; a good deal of emphasis was placed on the problems of dealing with foreign, specifically French and German, character sets; and, as the title indicated, the course was slanted as to how to use a computer for the specific purposes of linguistic and literary analysis. In the main, other modern languages departments took a polite interest, but did not feel the need to copy our initiative or get involved in teaching humanities computing. Moreover, with the exception

of Computer-assisted language learning, humanities departments did not use computers to teach their own subject.

This, then, was the background of humanities computing teaching at the undergraduate level, when in November 1988 Professor Derek Sleeman, Head of the Department of Computing Science at Aberdeen, proposed setting up a new course in computing for students from all departments in the Faculty of Arts and Divinity. Letters inviting expressions of interest or, indeed, offers of participation were sent to all heads of department and to other individuals believed to have an interest in computing, and a working-party was set up. Spearheaded by Computing Science, an initial syllabus was put together which was then approved by the relevant Board of Studies, and the course was allowed to go ahead on a non-graduating basis for one year in the first instance. That course was revised in the light of experience, and is now running as a graduating course with a full number of credits under the modular system for under-graduate degrees introduced in October 1990. There are two very obvious differences between the course as it was then and the course as it is now. First: the change in status from non-graduating to graduating. Second: the earlier course used MSDOS machines, whilst the present one uses mainly Macintosh SE30s.

Aims of the course

The formally stated aims of the graduating course were relatively modest. Students were expected to learn the terminology, key concepts, components, and customs of computing; the basics of using common application packages and very simple programming; the most notable ways in which computers have been used in various areas of the humanities by practitioners and researchers; and how to make everyday use of computers appropriately in arts courses. Our unstated aims were more ambitious. Firstly, we wanted to give our students exposure to a much wider range of applications programs than is normal for a short course aimed at non-mathematical complete beginners. Secondly, we wanted to encourage a confident, independent, exploratory attitude amongst students which we hoped would eventually give them the chutzpah to work out for themselves how to use new machines, new software and how to apply them to new humanities problems. Thirdly, we wanted our students to become critical about many existing applications of computers in the liberal arts, and to understand the depth of intellectual challenge of using computers seriously in the humanities.

A non-graduating 'dry run' of the course

Let us look briefly at the earlier non-graduating course. It was open to all students, undergraduate or postgraduate, and staff in the faculty, whether or not they had any previous computing experience. Two factors in particular, we thought, might militate against its popularity. First: since it was non-graduating, students would get no academic recognition for their participation and work in the course, although students completing the course did have the opportunity to write an essay and sit an examination paper: those who were successful were awarded a certificate. Second: it had to be held at lunchtime (1-2 p.m.), since this was the only available slot, although even this clashed with two large first-year classes.

In the event the course attracted an initial take-up of over 60 students. Ironically, the fact that the course was non-graduating gave us one big advantage in getting it off the ground: we were allowed to advertise it!

The course ran over the first two terms of the session, and took the form of an hourly lecture plus an hourly practical session per week, i.e. 34 hours of instruction in all. Broadly speaking, the lectures and practicals in the first term were designed to give students a basic knowledge of the modest set of three packages used in the non-graduating course - word-processing (Microsoft Word), databases (Dataease), and concordancing (Micro-OCP) - together with hands-on experience of the packages themselves. The introductory lectures were given mainly by Computing Science staff, and the initial practical sessions were supervised by specialist Computing Science demonstrators. The practical work was based on IBM-PC-clone machines, since classrooms full of these machines were already available and were already supplied with software packages that we thought would be workable (or at least liveable with!), if not always ideal.

The vast majority of participants had no previous experience of computing in any form - or even, in many cases, any knowledge of keyboards. It quickly became apparent that much more back-up than we had envisaged would, in fact, be necessary for the practical sessions. Students required assistance both on a one-to-one basis and in the form of tutorial handouts and exercise sheets. Although students were encouraged in the practical sessions to take an individual and exploratory approach rather than work through a rigidly defined corpus of work, many needed a good deal of direction, guidance and encouragement, particularly when changing from one package to another. This was reflected in a written

questionnaire students were asked to complete on the lectures and practicals for each term. Reaction to the scope and level of the lectures was generally positive, although several students felt that the practical sessions were insufficiently structured and wanted more set exercises to work through, with clearly defined goals for each session. Outside tutorial hours, it may be added, students had free access to the computing facilities whenever these were not being used for other classes.

After the introductory lectures, the majority of talks were given by staff from various departments within the Faculty of Arts and Divinity itself - Education, English, French, German, History, Philosophy and Logic, and Practical Theology. Tutors generally drew on their own research experience to exemplify and discuss the use of computers in their own disciplines. Subjects covered included: bibliographical information storage and retrieval, CALL, artificial intelligence, intelligent tutoring systems, hypertext, using a computer in the day-to-day running of a church ('Practical Theology'), literary and linguistic text analysis, concordances, music, and the social impact of computing. Where viable, the practical sessions were designed around the lectures, with each lecturer responsible for setting exercises and arranging supervision (with assistance from Computing Science) for his own subject. Two major exceptions to this were: the practical associated with the lecture on hypertext took place as a semi-demonstration, with students crowded round the one Apple Macintosh computer we had at our disposal at that time; and the computer music practical took the form of a demonstration of how to write, play and manipulate music, again using non-IBM-type machines. In addition, one lecture slot was taken over by Marilyn Deegan from the Computers in Teaching Initiative Centre for Literature and Linguistics Studies attached to the University of Oxford.

The graduating course

With the introduction of the graduating course in October 1990, we were able to build on the expertise gained from the previous year. The broad range of subjects was retained, and even increased with the introduction of lectures on the use of computers in Law, human computer interaction, and the ethics of computing, two of these topics being included as a result of the interest generated amongst staff by the previous year's course. This was made possible by that fact that the course now runs over twelve weeks, with two lectures and two tutorial sessions per week (i.e. 48 contact hours in all). The additional time also allows us to address some of the 'bittiness' that we previously experienced in some of the topics covered.

An overall lack of cohesiveness can be a major potential hazard with courses of this nature. It is all too easy for students to be given a very superficial overview of too many disparate aspects of a subject, without connections being made between separate elements, so that any one can relate to, and illuminate, others. We sought to address any potential shortcoming in two ways. First: all tutors were given the complete programme in advance, and encouraged to listen to others' lectures. Second, as the two course conveners, we attended every lecture and most of the practical sessions: we had deliberately spaced our own slots over the whole course, so that we were able to draw things together for the students at regular intervals, as well as anticipating what was to come in later lectures.

We had perceived the major failing of the previous year to be the difficulty which many students experienced in the tutorials and the sometimes massive amounts of tutorial help they required, especially to master each new package as it was introduced. We were deeply frustrated by the fact that so much time had to be spent in teaching students to memorize inconsistent sets of commands, each relevant to just one package. All of the systems we used (operating system, word processor, concordance package, database) had quite different interfaces. It became apparent that some students were confusing these interfaces badly, and that this confusion was undermining their confidence in their ability to understand and control computers. This was tackled - and, as our experience is now suggesting, remedied in several ways. Of these, the major factor has been switching from a "command line" computing environment to a well-designed and well-integrated direct manipulation environment. In practical terms this decision was implemented by switching from MSDOS to the Macintosh graphical user interface (GUI) environment.

The human machine interface as a teaching resource

A well-designed graphical user interface environment has a number of well-documented advantages (Smith et al, 1982) over a command line environment, principal among which is the consistent user interface metaphor and syntax between different applications. This led in our case to four concrete practical benefits. Firstly, at the beginning of the course we could boost students' confidence by introducing them to applications that we knew from personal experience could be learned by five year olds in less than a minute (a mouse-driven public domain drawing package, and a simple game to improve mouse manipulation skills.)

Secondly, we found (as others have) that because all applications used the same consistent interface, once students had mastered one package they could master subsequent packages very rapidly, and focus on the task in hand rather than its syntax. For this reason, we were able to introduce a far wider range of computer applications in the practical sessions than in the non-graduating trial year, even taking into account the extra hours available. New practical topics introduced included communications programs; use of the UK Joint Academic electronic mail network (JANET); use of the UK Humanities bulletin board HUMBUL; spreadsheets; hierarchical outlining, Prolog and Hypercard. A third concrete benefit was that because most actions on screen were reversible (i.e. there was an "undo" button that worked) and since most actions had clearly visible consequences, students were willing to explore new application programs for themselves without fear of irrecoverable error and with confidence in their ability to work out what they were doing. (We had always exhorted our students in the trial year to do this, but with little success.) The fourth and final main benefit was that despite the introduction of new topics, students were able to learn more in the time about the basic topics, such as word processing: for example, a higher proportion of students were able to rapidly master more powerful features of word processing such as indexing, style sheets, outlining, etc.

We feared that one potential disadvantage about the switch to Macintoshes was that in some parts of the course where we still needed to use MSDOS (for example, the concordance program was only available on MSDOS), students might be put under a lot of strain. They would be forced rapidly to learn a completely new operating system (MSDOS) and a new style of interaction for the sole purpose of learning a new application package. Indeed, some of our colleagues worried that our students were being 'featherbedded' on Macintoshes and would later perform badly when exposed to MSDOS. To our great surprise, the Macintosh-trained students by and large not only learned the MSDOS style of interaction very quickly, but went on to master in a single session more of the concordance program than the trial year's students had done. This was an informal observation rather than a formal experiment, but it surprised us and we gave some thought to trying to account for it.

Forming a stable conceptual model of the computer

One way we were able to make tentative sense of these observations was in terms of Sniederma's (1987) syntactic/semantic model of user knowledge. Our hypothesis ran as follows. By gaining practical experience of computers in an environment where the user interface was consistent, our students had built up a stable, meaningful picture of the kinds

of thing a computer can do (in terms of Shneiderman's theory, this is referred to as "semantic knowledge of the computer domain"). By contrast, in the trial year, students had had to focus on the memorisation of syntactical knowledge which, because essentially arbitrary, is unstable in memory without constant rehearsal. When the Macintosh trained students faced an unfamiliar MSODS interface, the combination of a clear general understanding of the semantics of interacting with a computer and a habit of exploring new applications autonomously gave them a stable base from which to learn the new syntax with the minimum of confusion.

Of course, these observations are informal and open to more rigorous experimental testing: it could be argued that the gains were just due to better teaching in the second year. In any case, it is undoubtedly true that students found the graphics interface much easier to grasp and use than the command-line interface of MSDOS. Much time was previously spent on irritatingly minor tasks like memorising the correct syntax for copying files, listing files and sub-directories, invoking applications, and so on: all this was obviated in the GUI environment. It is also true that students have been far more willing to explore applications for themselves. It should also be emphasised that we have no special brief for or against any particular manufacturer or environment. Our aim was simply to use the most affordable, consistent and well-designed user interface for a personal computer - which we judged (and still judge) to be the Macintosh interface.

With so many new applications to be covered, we wished to encourage students to explore applications in ways of their own choice, for example: working through an introductory Hypercard stack, reading the manual, working through our set exercises or exploring freely; but at the same time we wanted to maintain standards and prevent fragmentation in the practicals. No new teaching methods were needed for this: we found that the time-honoured discipline of setting very specific written learning objectives for each session supplemented with optional exercises and explanatory material that provided a set of alternative paths to those objectives worked very well.

Benefitting from network access

As well as having their own hard disk, the classroom computers were linked to a central fileserver, on which we placed files relating to the course (summaries of lectures, notices relating to the course, etc.) which students could pull down to their own floppy disks and manipulate as they wished. The computers were also linked to the university's central sys-

tem and to JANET (the UK academic electronic mail network). As already noted, this made it possible, for example, for students to access HUMBUL (the main UK research-oriented humanities bulletin board) and LANCS.PDSOFT (a UK archive of public domain software). But it also allowed case study work, as in the tutorial on bibliographies. For one task, students had to use KERMIT to log on to the university library on-line catalogue, use the catalogue search facility to locate certain items, and then copy these over, pasting them into their own bibliographical database. This was easily achieved by switching and pasting between processes under Multifinder in the Macintosh environment.

Artificial Intelligence and Human Computer Interaction for beginners

As well as changes in hardware, software and practical teaching approach, we made a small but significant change in the graduating year in our approach to the teaching of theoretical elements of Computer Science. After tackling the basics, we decided to put more stress on areas of Computer Science such as human computer interaction and artificial intelligence where, historically speaking, researchers with arts training and backgrounds have made disproportionately large contributions.¹ Shneiderman (1987) talks of designing user interfaces as a "complex and highly creative process which blends intuition, experience and careful consideration of numerous technical issues". Winograd(1990) stresses its similarity to design disciplines like architecture. We wanted students to become aware of areas within computing where opportunities exist for those with an arts background to make substantial contributions without having to major in computer science.

For related reasons, we decided not to teach programming in procedural languages in the graduating year (in the trial year we did not teach programming at all). Instead, we opted for an outline practical introduction to the artificial intelligence programming language Prolog. We had good precedent for this in the work of the Prolog Education group (PEG, 1987). Prolog is well suited to reasoning about symbolically expressed relationships and it can be argued that in many ways it is better adapted for use as an intellectual tool in the humanities than procedural languages such as Pascal.

¹For example, Sterling Beckwith (forthcoming) has noted that major contributors to human computer interaction and artificial intelligence such as Marvin Minsky, Alan Kay, Bill Buxton, and Jaron Lanier (we could add Christopher Longuet-Higgins, Terry Winograd, Phil Johnson-Laird and Herbert Simon) are all musicians or have been involved in music as part of their research.

We were influenced in our decision by experiments such as those at IRCAM (the Institute de Recherche et Coordination Acoustique/Musique) in Paris, and the Utrecht School of Art (Honing, 1989). At IRCAM, computer-naive conservatoire music students were taught the artificial intelligence language LISP for use as a music composition tool (Wessel, 1987). In this experiment, key concepts of LISP and artificial intelligence programming were taught using direct analogies from music composition and demonstrated using musical instruments connected to computers. We did not go this far (we have also omitted natural language processing so far - although it is an area where Prolog is almost uniquely simple and powerful to use (Bratko, 1990)). We focussed instead on a brief practical introduction to Prolog linked with lectures on logic (from the Philosophy department) and lectures on artificial intelligence. We also briefly introduced some ideas about object-oriented programming using a practical session with the language Hypertalk. Hypertalk is the language of the Macintosh prototyping tool Hypercard, which we used to allow students to begin designing and making prototypes of simple interfaces.

One by-product of our efforts to present this material as clearly as possible was the development of a simple but useful computer-based presentational technique, new at any rate to us. Some lectures were presented using a "hierarchical outliner" courtesy of a luggable Macintosh with an overhead display device. This allowed electronic lecture slides to be dynamically expanded or contracted during presentations depending on the degree of audience comprehension. This not only saved the cost of preparing acetates, it had the added benefit that after the talk, the lecture notes could be dumped on the file server for students to study at their leisure with a consequent saving of handout preparation costs.

Conclusions

In general terms, the interdisciplinary nature of the course has had three major effects. First: there has already been some useful interdisciplinary cross-fertilisation of ideas and methods. For example, the course has led to a small but highly energetic research project on AI and Film with the collaboration of the English and Computing Science Departments. Second: the Faculty of Arts and Divinity as a whole has attained a higher profile in computing need and usage than could ever have been achieved by individual departments (or individuals within departments). Apart from anything else, this should give it a stronger voice in the future shaping of hardware and software requirements within the university.

Third: it has raised the computer-awareness of colleagues both within and outside the Faculty of Arts and Divinity who hitherto have not seen anything in computing for them.

Resources in both human and technical form were, of course, the essential prerequisites. It is important to stress that, throughout the planning stages and in the teaching itself, we were fortunate in being able to call on the support and the expertise of Computing Science on the one hand, and of many individuals scattered throughout departments in the faculty on the other.

Another important factor has been that we were able to utilise some of the fairly extensive centrally provided computing facilities, with both hardware and software provision, available on campus. In addition, we were awarded a small materials grant from central University funds which enabled us to acquire items of software and site-licences requested by individual lecturers for specific applications or subject-areas. The course was also included in the University's successful bid for funding from the Department of Trade and Industry's Enterprise Initiative scheme, which will provide future funding.

Perhaps the most important thing to stress is that we did not pre-select or pre-determine one particular sphere of activity for students. If a university education, particularly in humanities subjects, as opposed to some sort of narrowly focussed, goal-directed professional training, is supposed to be about expanding students' awareness, stretching their imagination and their intellect; and opening their eyes: then I would suggest that we have mostly succeeded in fulfilling these perhaps unfashionable ideals.

The final question to consider is how the course fared in terms of the attitudes mentioned at the beginning of this article. The Computer Science Department now considers this course to be one out of which innovations in teaching computing have arisen and can be expected to continue to arise. The extent to which attitudes amongst Humanities staff have become more positive can be judged by the fact that several faculty members have attended lectures and started talking to the Computer Science Department about collaborative projects. Students have given enthusiastic feedback, worked hard on a number of extra-curricular humanities computing projects of their own devising, carried out independent research as part of the course, and, in some cases, written assessed essays of publishable quality. As an informal indicator of attitudes, humanities students from the course have also variously

bought their own Macintoshes (unusual in Britain) held course parties, formed a course band and started editing and producing a campus-wide computing magazine.

Computing and Humanities Departments have co-operated extensively to create this course. The course allows a lot of freedom within limits that are carefully structured and tightly controlled, and offers students an introduction to humanities computing that goes some way beyond mere word processing and computer literacy. We consider that finding ways to use computers to their fullest extent in the Humanities is one of the major intellectual challenges of the next few decades. We have tried to communicate the excitement of this challenge to our students, and in future years we hope to find ways of continuing to do this yet more effectively.

References

Bratko, I. (1990) *Prolog Programming for Artificial Intelligence*. Menlo Park, Addison Wesley.

Beckwith, S. (forthcoming) Hunting Musical Knowledge in Darkest Medialand. In Edwards, A. and Holland, S. (forthcoming) Eds. *Multimedia and multimodal interface design in education*. Springer Verlag, London.

Honing, Henkjan (1989) *Personal communication*. Utrecht Art School, Utrecht.

PEG (1987) *Proceedings of the 2nd International Conference of the Prolog Education Group (PEG 87)*. Exeter, UK 8th-10th July 1987.

Shneiderman, B. (1987) *Designing the User Interface*. Menlo Park, Addison Wesley.

Smith, D.C., Irby, C., Kimball, R., Verplank, W., and Harslem, E. (1982) Designing the Star User Interface. *Byte* 7(4), April 1982 242-282.

Wessel, David (1987) *Personal communication*, IRCAM, Paris.

Winograd, T. (1990) What can we teach about Human Computer Interaction? *Proceedings of CHI 1990* (ACM SIGCHI conference on Computer Human Interaction). pp. 443 - 449.

