Affordable Interactive Multimedia Learning Systems

Chris Pilgrim and Mike Creek
Swinburne Computer-Human Interaction Laboratory
School of Computer Science & Software Engineering
Swinburne University of Technology
chris@saturn.csse.swin.edu.au

Abstract

This paper describes a new technique for developing multimedia courseware which reduces development time and enhances the quality of the individual learning experience for the student particularly when learning computer programming. The key to this approach is the use of a new software package called Lotus ScreenCam which can quickly integrate PowerPoint presentations and actual compiler sessions with an audio narrative.

Keywords

multimedia, computer programming, computer-based education

1. Introduction

The School of Computer Science and Software Engineering of Swinburne University of Technology delivers programs to a diverse audience of students. Part-time students studying in the evening have differing needs and commitments from those of full-time students; international students often have poor spoken language comprehension; students studying within multi-modal programs require access to extensive computer support tailored to their study programs. Multimedia courseware offers the ability to present courses meeting these varied needs—part-time students can study at home, attending classes only when absolutely necessary; international students can review material many times, with spoken language reinforcing written material; multi-modal students can access the resource when appropriate. Multimedia development, however, is prohibitive in time, prohibitive in money, prohibitive in the expertise required by developers and prohibitive in the short life span of the product. Techniques are needed that shorten the developmental life cycle, that reduce the need for specialised multimedia expertise and that allow the expert in the knowledge domain to do a great deal of the developmental work. This paper describes one such technique.

2. Background

Traditional approaches to teaching computer programming at universities are inappropriate in that they do not cater for differing student backgrounds, nor do they allow for differential learning rates. Students show a wide range of prior computing experience, extending between several years as practitioners to complete novices who do the course only because of its essential prerequisite status in many degrees. Ellis (1993) describes many of the problems that this diversity of prior knowledge causes, including withdrawal, confusion, failure and frustration for both the lecturer and students.

In addition to this diversity of student background, many students have difficulties with the grammar of the language, comprehending the behaviour of the varied control structures (Tyerman, 1993), and

envisaging the solution to a problem as consisting as a sequence of these control structures (the algorithm) acting upon data (Spohrer and Soloway, 1986). These students require a large amount of individual assistance and time to reach some understanding of the processes involved in programming.

The traditional lecture-based course which guides students through the course material at a set pace compounds both of these problems. It is difficult to predict and cater for the level of initial competence of the incoming students and just as difficult to manage the varied learning rates. Many students do not handle the important initial sections and fall behind, never to catch up. Other more competent students lose interest and become bored.

Current approaches to the challenge of teaching programming at Swinburne incorporate the use of extensive handouts, compiler sessions and PowerPoint presentations projected in lectures, and CML test banks done in laboratory sessions. These have provided students an effective learning experience but still rely on a lecture series to set the pace of the students' learning, hence losing some of the weaker students who, given time and appropriate assistance, might have achieved understanding.

Current research into the delivery of programming courses (Zivkova, 1994; Chandra, 1993; Winder, 1993) suggest that, given the sequential nature of topics in this domain, a self-paced, student-centered, mastery learning approach should to be used to structure the delivery of a course. There should be a logical and consistent progression through the features of a language ensuring that the student has mastered a certain amount of content before moving on to the next section. This mastery approach to structuring a course allows students of different abilities to select an appropriate starting point and then progress through the course at a rate determined by their understanding. Effective pedagogical devices are required to manage the process to ensure that students moves through course modules at an appropriate pace.

The use of text books and learning guides as a method of providing individualised instruction has been trialled at Swinburne University within its Multi-Modal Learning project. Learning guides were published for each of the units in pilot programs to assist students to know what is to be learned, how to learn it independently and how to demonstrate that they have learned the required material. The multi-modal approach to teaching programming experienced some initial problems. The difficult concepts involved in programming require a range of approaches to communicate the basic principles effectively. Text books typically focus on the syntax and semantics of constructs in a language and do not give sufficient instruction in how to 'put the pieces together' (Soloway, 1986). Textbooks are also limited in the amount of feedback they can provide (Tyerman, 1993) and are constrained by consensus as to their overall content (Hewitt, Burden and Chen, 1993).

3. A Multimedia Solution

Multimedia courseware has largely been accepted as an avenue to provide students with an effective indepedent learning experience: Marchionini (1988) states that cognitive skills are enhanced with multimedia; Tan and Nguyen (1992) believe that multimedia makes the learning process more interesting and stimulating, while at the same time greatly increases the rate of learning and knowledge retention. Gibbons (1990) argues that by enabling students to link data, information and ideas, interactive multimedia helps to make the connections that are critical to learning. By integrating text, graphics, sound, animation and video, it addresses different learning styles, providing a truly interactive learning environment that students can explore, add to and compose in, enabling them to become actively engaged in the learning process. 'Generally, there is no doubt about the usefulness of multimedia to convey information to people' (Maurer and Scherbakov, 1995).

There is a misconception held by academic administration that computer based education (CBE) will reduce costs (Gillespie, 1993). Full multimedia development, including the use of sound and

animation, is a time consuming and costly process. It is commonly accepted that it takes over one hundred of hours of development for each hour of product (Merrill, 1993; Yaverbaum and Reisman, 1995; Kingsland, 1992). Accordingly, the assumption that CBE saves money is ludicrous.

The discipline of computing, less than 50 years old, has developed rapidly. An example of this rapid evolution is the introduction of the object-oriented approach to analysis, design and programming which has seen the release of a number of new object-oriented programming languages such as C++. This language is itself evolving as our understanding of the object-oriented paradigm expands.

One consequence of this evolving nature of computing is that the active shelf life of multimedia resources to support the teaching and learning of programming is short relative to the long development time required for quality multimedia products. There is a reluctance to invest valuable time and money into the development of multimedia courseware if it means that it must be continually updated. These circumstances explain why there is a lack of quality multimedia resources to support many fields of computing, especially programming.

There is also a certain reluctance to use commercial off-the-shelf courseware. The 'Not Invented Here' syndrome is a reality that must be accepted and worked with. University lecturers are creative people who have their own ideas, points of view, and methodogical approaches which they want to incorporate into the courseware (Maurer and Scherbakov, 1995).

Hence, the current situation may be summarised as:

- Multimedia courseware is required to allow an individual, mastery learning approach to learning computer programming.
- Lecturers are reluctant to use off-the-shelf multimedia courseware.
- The costs associated with the development of multimedia are prohibitive, especially in a rapidly evolving field such as computing.

4. Keep it Simple

Spannaus (1993) argues that time is a resource, not a measure of quality and that, like money, time should be managed so that we use only what we need and no more. Techniques are required to reduce the time investment in any multimedia project whilst retaining the quality.

The Pareto Principle, known better as the 80 / 20 rule, states that the first 20% of work on a problem will usually account for 80% of the solution. The other 80% of work is simply window dressing. This minimalist approach may be applied to multimedia development. Many designers of courseware view the vast range of media tools available to them and have the mistaken notion that the quality of the product is proportional to the amount and range of media used.

> Multimedia practitioners must be careful not to get wrapped up in their own creativity and brilliance. There is an old saying that the simplest things in life are often the best. In Multimedia, the most successful applications are those that are commercially viable and it just so happen that they are often the simplest . (Balson, 1992)

There is a danger, with the current fervour for multimedia, of using interactive multimedia in software for purely cosmetic reasons rather than to contribute to the educational effectiveness of such programs. As Clark (1990) reminds us, the contribution of media to the effectiveness of learning is no more than 'the truck which delivers groceries to the market contributes to the nutrition in a community'. There is a misconception that by encasing an existing course in a envelope of 'whizz-

bang' multimedia, the course will automatically become effective. The motivational appeal of the course may be heightened, but it is unlikely that the educational effectiveness will be increased simply by adding sounds and video.

There is evidence that overuse of multimedia could reduce the effectiveness of educational systems. Langham and Johnson (1984) believe that the use of too much visual detail can lead to too much demand being put on the learner's working memory capacity. They argued that if a visual analogy requires the learner to divert too much of his / her cognitive resources to understanding the analogy, then the chances of the new material being absorbed is reduced. This suggests a general approach of simplicity to the design of interfaces of multimedia courseware. This simplicity rule has been acknowledged as important by a number of authors. Phillips (1993) documents a number of simple rules for the design of courseware which emphasise that screens should not be cluttered with too much information or too great a variety of symbols, colour or other enhancements. Webster and McNamara (1992) also suggest the avoidance of cluttering the application with complex features which may eventually have a negative impact on the teaching / learning interaction. This 'Keep it Simple' approach suggests that the 'bells and whistles', which are time consuming in the development phase, could be removed leaving simply 'good instruction'.

5. Re-use and Recycle

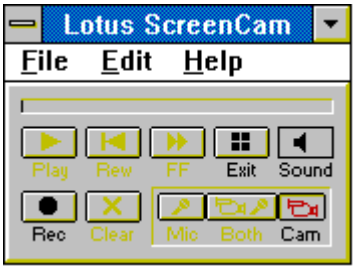
The discipline of software engineering has many techniques and approaches to the construction of software that reduces overall costs without a reduction on quality. One technique that is relevant is 'software re-use'. Software engineers believe that new programs should re-use existing materials as much as possible to eliminate duplicate effort.

There have been a number of investigations into methods by which existing material may be quickly incorporated into new independent learning systems: Hewitt, Burden and Chen (1993) suggest that teachers could produce their own material in a 'less that multimedia manner' by using existing lecture notes and slides that were already computer-processed as a base resource in the development of independent learning materials. Maurer and Scherbakov (1995) proposed an 'on-the-fly' method of courseware production through modular authoring including the integration of different types of source materials into newly created courseware and the flexible re-use of existing material.

The technique suggested by this paper is based on the re-use and recycle principle. Most lecturers today use presentation software such as Microsoft PowerPoint to prepare and even deliver lectures. This lecture material is usually highly evolved and polished and provides a rich foundation for any multimedia development. The proposed technique takes the existing lecture series and quickly adds an interactive structure and audio narrative to produce a system that may be used independently.

6. Lotus ScreenCam™

Lotus ScreenCam™ is a program that allows any Windows session to be recorded, saved and then later played back. It captures all screen activity including mouse movements, menu manipulation and screen progression. It also enables a voice-over narrative to be captured during the session. Movies may be saved as an executable file with a built-in play-back only control panel which may either be embedded in an OLE-capable application or simply called as an executable from another program.



Lotus Screencam™ is very simple to use. The Screencam Recorder has a familiar VCR interface.

By selecting the Rec button, a recording of all Windows events as well as the audio narrative occurs. A small Stop button will be displayed in the bottom right corner of the screen to allow the movie session to be terminated.

Figure 1. Screencam Recorder.

The Screencam Player is a play-back only program which may be distributed along with movies or may be embedded into an standalone movie. The interface is also familiar and provides standard rewind, fast-forward and play controls for the user to manipulate.

Figure 2. Screencam Player.

7. Sample Module

One module of an existing programming course has been developed using the technique. The raw material for the module consisted of a PowerPoint slide show of 35 slides along with a demonstration of constructing and running example programs using the Borland C++ compiler.

The development process was undertaken in the following stages:

1. The existing 35 slides were split up into eight logical subsections each dealing with a specific area.

2. A separate ScreenCam movie was made for each of the eight subsections. The ScreenCam movies recorded the progression through the PowerPoint slides along with a voice-over narrative explaining the material on each slide. Each subsection went for approximately 5-10 minutes.

3. A further ScreenCam movie was made for each compiler demonstration. These movies recorded the actual construction of a program, the debugging process, an explanation of the tools used and the execution of the completed program.

4. A small driver program was then written using Authorware Professional to allow access to each of the different sections.

5. The system was burnt onto a CD to be placed into the library as a resource for students to use.

The total development time for the package was approximately 10 hours and it provides around one hour of instruction.

8. Discussion

The system that has been developed may not be the most feature-packed piece of software developed, nor does it have a glossy interface, but it does contain the essential elements of instruction in a simple

and consistent structure and may be used independently by students in a mastery learning approach. The system has two strong pedagogical features: the audio narrative and the demonstration of an actual compiler session.

The use of the audio narrative is the key to the instructional effectiveness. Audio has largely been ignored as a tool to enrich learning experiences in multimedia systems (Bradely and Henderson, 1995). The narrative provides the learner with a personal instructor whose explanations may be listened to, pondered over and reviewed as many times as required.

Another feature of this system that is extremely powerful is the ability to capture an actual programming session showing the writing of code, the use of debugging tools and the execution of the code overlaid with an audio narrative. This provides a deeper understanding of program development and the practical use of the programming concepts being introduced by the module. This is an experience which no text or traditional lecture-based course can provide.

The use of visual demonstrations is an extremely effective way to convey complex ideas: Reeves (1992) believes that 'if knowledge is learned in a context of use, it will be used in that and similar contexts'. In traditional instruction, information is presented in encapsulated formats, often via abstract lectures and texts, and it is largely left up to the student to generate any possible connections between a problem the use of knowledge as a tool to solve the problem. The narrated compiler sessions assist students in being able to transfer theoretical learning to a real-life situation such as the development of programs. Birch *et al.* (1995) speak of the virtues of program animators which provide a visual view of the code during execution permitting the study of program constructs. Rodgers (1995) argues that animations of algorithms and demonstrations of development tools is beneficial: 'Showing several examples using a particular tool will let the students see its capabilites. When students see how easy or useful a tool is, they are more inclined to use it themselves'.

9. Conclusion

The original development technique used in this project allows cost-effective and timely production of quality multi-media courseware within a framework readily permitting the updating and addition of further material. The speed of development and ease of use of the ScreenCam software is the key to this project.

Use of the system will lead to enhanced learning of programming concepts by allowing students to move at their own pace through a range of learning experiences in a multi-sensorial environment. A further addition planned for the system is the development of a CML section at end of each module to test student understanding and to offer feedback. This feedback is essential for the mastery learning approach by allowing students to control their progression through the course at an appropriate rate. This type of system will be especially useful at Swinburne University of Technology given the diverse needs of our student population. These students are in desperate need of a resource such as this with which they can review lecture material unconstrained by place and time.

Although the aim of the project was to address the needs of students studying computer programming, the development technique used may be applied to any field that is evolving so quickly that the cost of full multimedia development is prohibitive.

10. References

Balson, C. S. (1992). Multimedia: How does business define acceptable standards? *Proceedings of the International Interactive Multimedia Symposium*, Perth.

Birch, M. et al. (1995). DYNALAB: A dynamic comptuer science laboratory infrastructure featuring program animation, *ACM SIGCSE Bulletin*, Vol. 27, No. 1.

Bradely, S. and Henderson L. (1995). Voice-overs and auditory cues: Their impact and role in learning through IMM, *Proceeding from EDMEDIA' 95*, Graz, Austria.

Chandra, K. (1993). C++ in eight weeks, *ACM SigPlan Notices*, Vol. 28, No. 8.

Clark, R. E. (1990). Evidence for confounding in computer-based instruction studies: Analysing the meta-analysis, *Educational Communications and Technology Journal*, Vol. 33, No. 4.

Ellis, A. (1993). Bridging the gap: A CAL solution to diversity of starting levels in an undergraduate computing course, *Proceedings from ASCILITE' 93*, Lismore.

Gibbons (1990). In J. Barker and R. Tucker (Eds.) (1992). *The interactive multimedia revolution*, Kogan Page, London.

Gillespie, L (1993), Developing computer-based learning / instruction in the real world, *ASCILITE Newsletter*, Vol. 7, No. 1, p. 26.

Hewitt, D., Burden, F. and Chen, A. (1993). Total course management on the cheap, *Proceedings from ASCILITE' 93*, Lismore.

Kingsland, A. (1992). CAL need not be resource extravagant, *Proceedings from ITTE 1992*, Brisbane.

Langham and Johnson (1984). In J. Barker and R. Tucker (Eds.) (1992). *The interactive multimedia revolution*, Kogan Page, London.

Marchionini, G. (1988). Hypermedia and learning: Freedom and chaos. *Educational Technology*, Vol. 28, No. 11.

Maurer, H. and Scherbakov, N. (1995). Support for teaching in a hypermedia university transaction, information and communication system, *Proceeding from EDMEDIA' 95*, Graz, Austria.

Merrill, M. D. (1993). *ID Expert: Reaching out for automated instruction design*, Presentation at *ASCILITE' 93*, Lismore

Phillips, J. (1993). Interactive courseware design - Is activity enough?, *Proceedings from ASCILITE' 93*, Lismore, pp. 479-490.

Reeves, T. (1993). Conducting science and pseudoscience in computer-based learning, *Proceedings from ASCILITE' 93*, Lismore.

Reeves, T. C. (1992). Effective dimensions of interactive learning systems, *Proceedings from Conference on Information Technology for Training and Education*, Brisbane.

Rodgers, S, (1995). An interactive lecture approach to teaching computer science, *ACM SIGCSE bulletin*, Vol. 27, No. 1, March 1995.

Soloway, E. (1986). Learning to program = Learning to construct mechanisms and explanations, *Comm of ACM* 29/9.

Spohrer, J.C. and Soloway, E (1986). Novice mistakes: Are the folk wisdoms correct?, *Comm of ACM* 29/7.

Spannaus, T. (1993). Better faster cheaper: What we can learn from manufacturers, *Proceedings from ASCILITE' 93*, Lismore.

Tan, W. and Nuygen, A. (1992). Costing models for computer-based interactive systems in the Australian tertiary education context, *Proceedings of the International Interactive Multimedia Symposium*, Perth.

Tyerman, S. (1993). Using computer-aided instruction in an introductory programming course for first year engineering students, *Proceedings from ASCILITE' 93*, Lismore.

Webster, L. L. and McNamara, S.E. (1992). Power at my fingertips but which button do I press - Researching and evaluating learner interactions in hypermedia, *Proceedings of the International Interactive Multimedia Symposium*, Perth, pp. 285-297.

Winder, R. (1993). *Developing C++ software*, Wiley.

Yaverbaum, G. and Reisman, S. (1995). Multimedia paradox: Where do we go from here?, *Proceeding from EDMEDIA' 95*, Graz, Austria.

Zivkova, E. (1994). Experience in teaching object-oriented programming with C++, *ACM SigPlan Notices*, 29/5.

10. Acknowledgements

Lotus ScreenCam^{TM} is a registered trademark of the Lotus Development Corp.