Int. J. Engng Ed. Vol. 22, No. 4, pp. 829–838, 2006
Printed in Great Britain.

0949-149X/91 $3.00+0.00
© 2006 TEMPUS Publications.

Integrating Multimedia Technology into the Undergraduate Curriculum*

PHILIP K. MCKINLEY, BETTY H. C. CHENG and JUYANG WENG
Dept. of Computer Science and Engineering, Michigan State University, East Lansing, Michigan 48824, USA. E-mail: mckinley@cse.msu.edu

Given the ubiquity of multimedia technology, it is important that Computer Science students not only learn the basics of multimedia design, but also gain hands-on experience with applications of the technology. This paper describes the integration of multimedia concepts and tools into a Computer Science curriculum. An NSF-sponsored Multimedia Laboratory was established and used to support three senior-level courses: software engineering, computer graphics, and computer networks. Curriculum development activities, laboratory exercises, and the role of course projects are described.

INTRODUCTION

IN MULTIMEDIA applications, computers are used to manipulate, distribute, and present data to human users in multiple forms, including text, images, audio, video and graphics. Recent advances in workstations and computer networks have made such applications not only feasible, but also accessible to a large number and wide variety of users. One need only consider the explosive growth of the World-Wide Web in the last few years to appreciate the increasing demand for multimedia applications.

Computer scientists join millions of other people as *users* of multimedia applications. However, it is also their responsibility to be the *designers* of next-generation multimedia applications, which requires new approaches to the design of software for computing and communication [1]. We, as educators in computer science, must prepare our students to meet this challenge by augmenting the curriculum to include experiences with state-of-the-art multimedia technology [2].

The Department of Computer Science and Engineering at Michigan State University (MSU) created a Multimedia Laboratory to support undergraduate education. The Laboratory was supported by an NSF Instructional Laboratory Improvement (ILI) grant and matching funds from MSU. Over the past few years, we have used the laboratory to support multimedia extensions to three senior-level courses: Software Engineering, Computer Graphics, and Computer Networks. This paper describes the integration of multimedia technology into the three courses, including both curriculum development and laboratory exercises/projects.

An important aspect of this project, and the major emphasis of this paper, is the use of existing multimedia tools. Tools are used:

1. to directly manipulate data and demonstrate physical and system behavior
2. to support the development of software
3. as components in large-scale multimedia applications developed by students.

MULTIMEDIA LABORATORY OVERVIEW

The Department of Computer Science and Engineering at MSU serves approximately 500 computer science undergraduate majors and approximately 150 graduate students. In addition, the Department jointly administers an undergraduate program in computer engineering, which has approximately 280 majors. The Department also serves a large number of non-majors through service courses and courses open to other majors.

The MSU Multimedia Laboratory described herein contained a collection of networked Silicon Graphics Indy workstations and a variety of support peripherals. The SGI systems were equipped with video cameras, graphics accelerators and hardware support for video compression. Also, these systems were configured with a wide variety of multimedia software packages: the Indigo Magic multimedia desktop user environment, the CASEVision Workshop suite of software development tools, the IRIS Explorer modular application builder, the IRIS Showcase multimedia presentation composer, the RapidApps prototype development package, and the ClearCase configuration management framework. In addition, Cosmo Software provides a graphical development environment for web documents, and the IRIS Inventor tool and Alias/Wavefront PowerAnimator provide high-level facilities to develop user interfaces and sophisticated graphics.

* Accepted 16 May 2005.

Our use of the multimedia laboratory has been based on a three-part model of multimedia applications: computing, presentation, and communication. Figure 1 illustrates the relationship among these parts; each block represents a single-user computing platform. *Multimedia computing* refers to the management and processing of multimedia data; multimedia software must be able to accommodate audio, images and video as data types. *Multimedia presentation* concerns those aspects of the application with which the user interacts, such as graphical user interfaces, scientific visualization tools and graphics for object-oriented design and animation. Finally, *multimedia communications* is needed for applications that are distributed in nature or which require data from remote sites; functions include digitization, data compression, real-time delivery, and integration of multiple data types. The combination of these three components make multimedia applications fundamentally different from other computer applications.

We have used this framework to incorporate multimedia experiences into three existing senior-level courses:

- Software Engineering introduces students to all phases of software development, including specification, design, coding, testing, and verification.
- Computer Graphics covers a wide range of graphics topics, including graphics hardware, raster graphics, 3-D graphics, user interface design, object representation, and object modeling.
- Computer Networks covers a wide spectrum of topics in computer communications, from physical transmission media, to network protocols, to applications.

In each of these courses, we have followed a three-pronged approach to the incorporation of multimedia content: teach students fundamental principles (theory, algorithms, techniques), have students apply these principles in the software components that they build themselves, and let students work with production-level tools and applications that also demonstrate these principles.

CURRICULUM DEVELOPMENT

This section describes how the three courses have been re-engineered to add a multimedia dimension.

Software engineering

Our software engineering course is designed to teach students the fundamentals of software development, beginning with problem identification and requirements analysis through design and testing. The course places significant emphasis on object-oriented analysis and design techniques and on the use of advanced software development tools. In addition to lectures and weekly laboratory assignments, students work in teams on three deliverables for a large-scale project: a requirements analysis document including a prototype, a high-level design document, and a user manual. In a subsequent software engineering capstone course, the students apply the techniques learned in this course to an industry-sponsored project.

The recent focus of the course projects has been on software development techniques specific to embedded systems [3]. Students are asked to model an embedded system using the object-oriented modeling technique (OMT) [4]. The Multimedia Laboratory plays a key role in this process. The RapidApps rapid prototyping utility available on the SGIs enables students to gain an accurate understanding of the behavior of the system, since students can build interactive prototypes that provide graphical animation for sensor input and actuator responses. The students also gain experience with configuration management software using the ClearCase framework to manage different types of artifacts. Finally, we

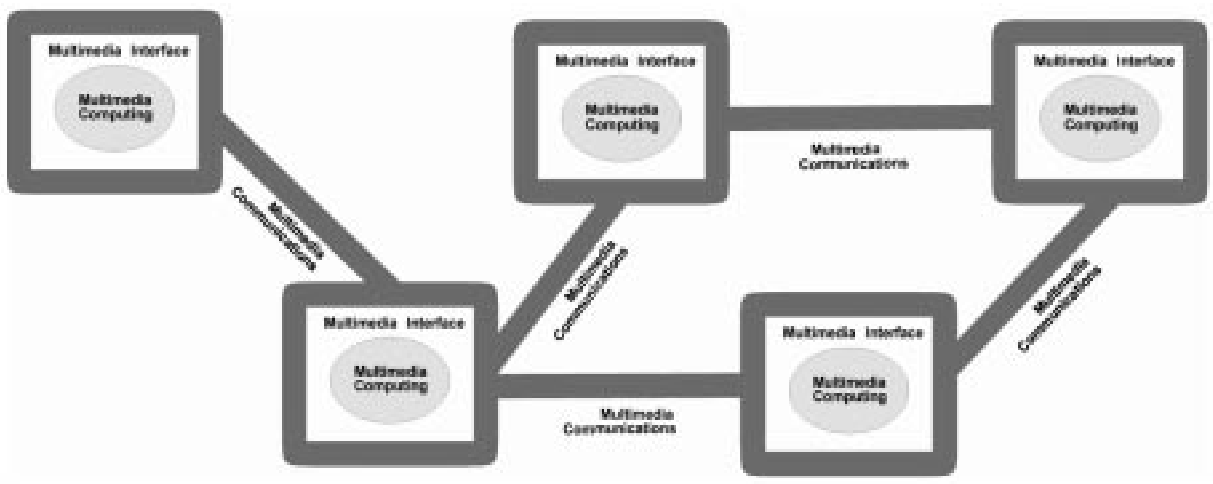


Fig. 1. Model of multimedia applications.

used the World-Wide Web to support project status communication among team members, between the students and the instructor, and between the students and their industrial customers. The Cosmo Software package on the SGIs provides graphical user interfaces that enable developers to quickly build sophisticated web pages and generate Java applications.

Computer graphics

As computer graphics is increasingly used in computing and communications, instruction in computer graphics must remain at the forefront of technology in order to prepare students for the challenging tasks of the present and the future. While our computer graphics course emphasizes basic graphics algorithms, such as rasterization algorithms for lines and curves, shading algorithms, and ray tracing algorithms, it previously did not provide students with sufficient exposure to professional graphics computing environments and the design issues associated with professional graphics software. We have redesigned our computer graphics course to address this problem through the use of the SGI programming environment. A key question that we had to address was how to provide these experiences in generating complex graphics objects and their animation without requesting students to devote an unreasonably large amount of time to the course.

The solution to this problem lay in the proper integration of tools into the curriculum. In addition to providing low-level graphics systems, such as X and Motif, the SGI programming environment provides two {3-D} graphics systems, OpenGL and OpenInventor. The former provides more freedom for designing {3-D} graphics, while the latter provides more high-level graphics objects and tools that enable the programmer to produce complex graphics content in much less time. Another advantage in using the SGI platform is the availability of a wide array of professional user-level graphics software. For example, we acquired and installed a software package from Wavefront/Alias that includes three professional graphics tools: PowerAnimator, Studio Paint {3-D}, and Composer. Such software enables us to provide students with experiences that go well beyond those of a typical computer science undergraduate graphics course.

We revised the structure of the course to emphasize four graphics systems that are designed for different levels of graphics applications: X/Motif, OpenGL, OpenInventor, and Alias/Wavefront PowerAnimator. We restructured the lecture material in such a way that it covers the basic concepts and major characteristics of each of the four graphics systems. Through these lectures, the students can gain a general understanding of the important concepts in each system. Students then have the opportunity to work with each of these graphics systems through the laboratory assignments, as discussed later.

Computer networks

The senior-level computer networks course combines both theory with practice. The theoretical aspects of the course emphasize topics that are long-term in nature and which may be applied in a variety of ways. Such fundamentals include underlying algorithms and protocols used in digital encoding, error detection and correction, access control, network routing, flow control and reliability. However, an equally important part of the course is to present specific examples of how these fundamentals are applied in real networks. Lectures include many examples drawn from both the Internet research community as well as from industry, and laboratory assignments give students hands-on experience with these concepts. Given the rapid pace of change in computer networking, the content of the practical side of the course has evolved over the years. Extending the course materials to emphasize multimedia communications was a natural part of this evolution.

The multimedia extensions are grouped into three main categories: enabling technologies, network-level protocols, and application-level protocols. Example topics in the first category include encoding schemes for fast modems, signaling in Fast Ethernet and Gigabit Ethernet, design and operation of switched LANs, an overview of ATM services and switch architectures, and an overview of cellular telephony. In terms of network-level protocols, the primary focus of the course is on the TCP/IP protocol suite; in addition to standard IPv4 operation, we now include extensive discussion of IPv6, IP multicast, and IP over ATM/SONET. Finally, the course has long included extensive treatment of application level protocols such as FTP, SMTP, NNTP. We have extended this part of the course to include JPEG and MPEG compression, public key encryption, and HTML/HTTP, as well as an overview of object-based middleware technologies such as CORBA, DCOM, and the Java/RMI.

LABORATORY EXERCISES AND PROJECTS

As mentioned previously, tools played an integral role in adding the multimedia dimension to all three courses. Below is a brief description of the tools that were used in each course.

Software engineering

The software engineering course includes two laboratory components. First, weekly laboratory assignments are designed to teach each student how to use different methods and tools needed in software development. Second, the students work in teams on large-scale projects, where they apply their collective knowledge of software engineering and associated tools.

In the laboratory exercises, students apply object-oriented modeling and design techniques to sample problems. The exercises also expose students to new tools that facilitate the delivery of their projects to their respective customers in the form of prototypes, user guides, and presentations. The course projects are obtained from industrial contacts in two main domains: household appliances and automotive systems. Past projects include design of controllers for washing machines, dishwashers, ovens, ABS brakes, a climate control system, a driver notification system, cruise control, and a monorail system. Based on a brief (2–3 page) project description, students are first expected to perform domain research to better understand the requirements and scope of their respective projects. The students must gain approval of the requirements from the 'customer' (either a teaching assistant or the instructor) before proceeding to the design stage.

Software tools available on the systems in the Multimedia Laboratory are important to both the laboratory exercises and the projects. First, in order to facilitate the requirements analysis stage, project prototypes are built using the SGI RapidApps software, whose graphical front-end enables students to easily build user interfaces for their prototypes. In addition, interactive functionality can be attached to the graphical entities, thus making the prototypes executable. Figure 2 shows the interface developed for the Driver Information Notification System project [7], and Fig. 3 contains the Monorail Control interface [7]. Both of these functional prototypes were developed using RapidApps, thus helping the project teams and the customers to better understand the requirements of the respective projects. The students were able to model the user interface, the sensors, and the actuators for the systems. Second, the ClearCase configuration management framework is useful in maintaining version control between the models used in the requirements document versus those used in the design document. Third, the Cosmo software significantly decreases the learning curve associated with building the web pages and applications. Towards the end of the semester, using

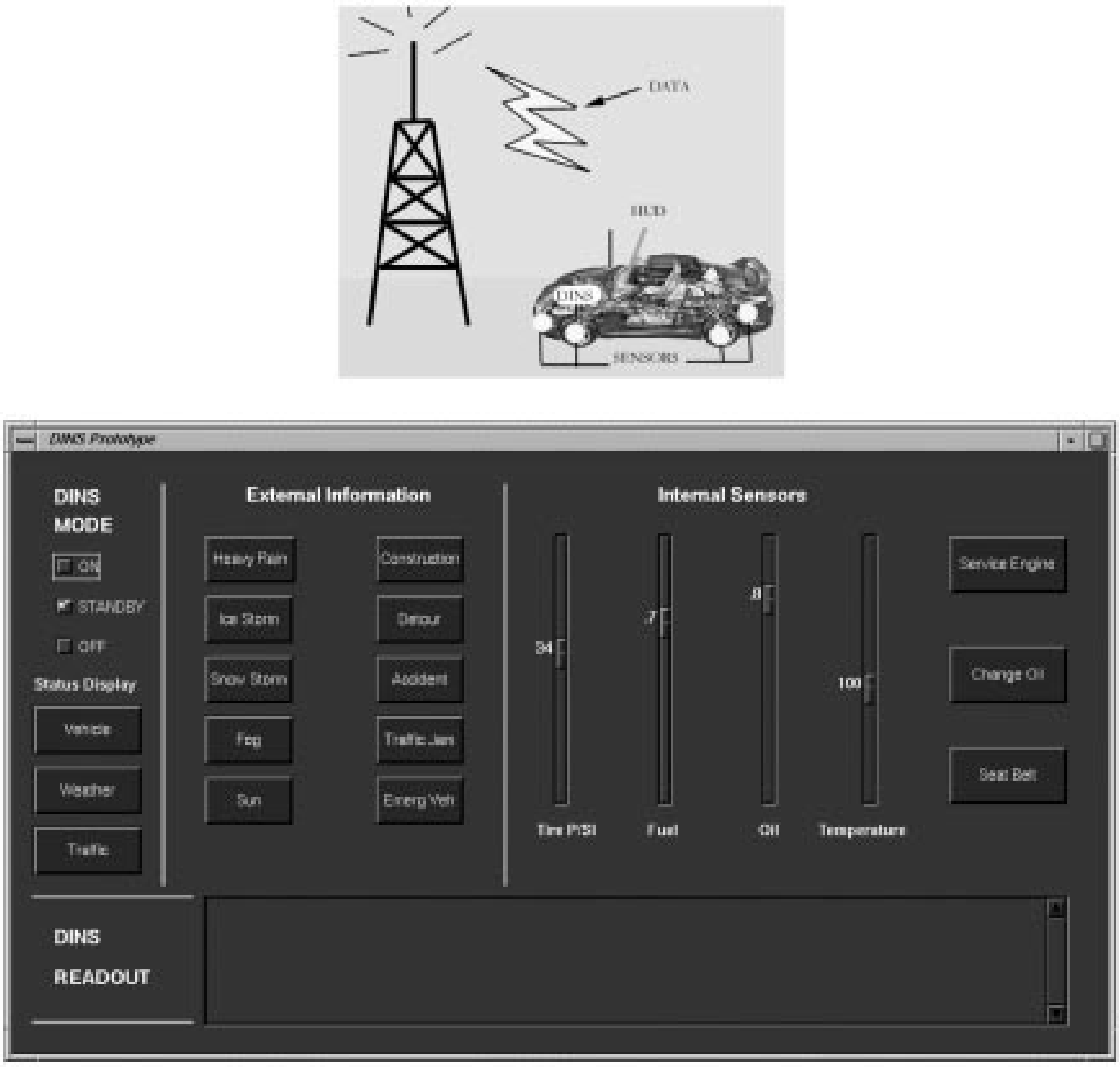
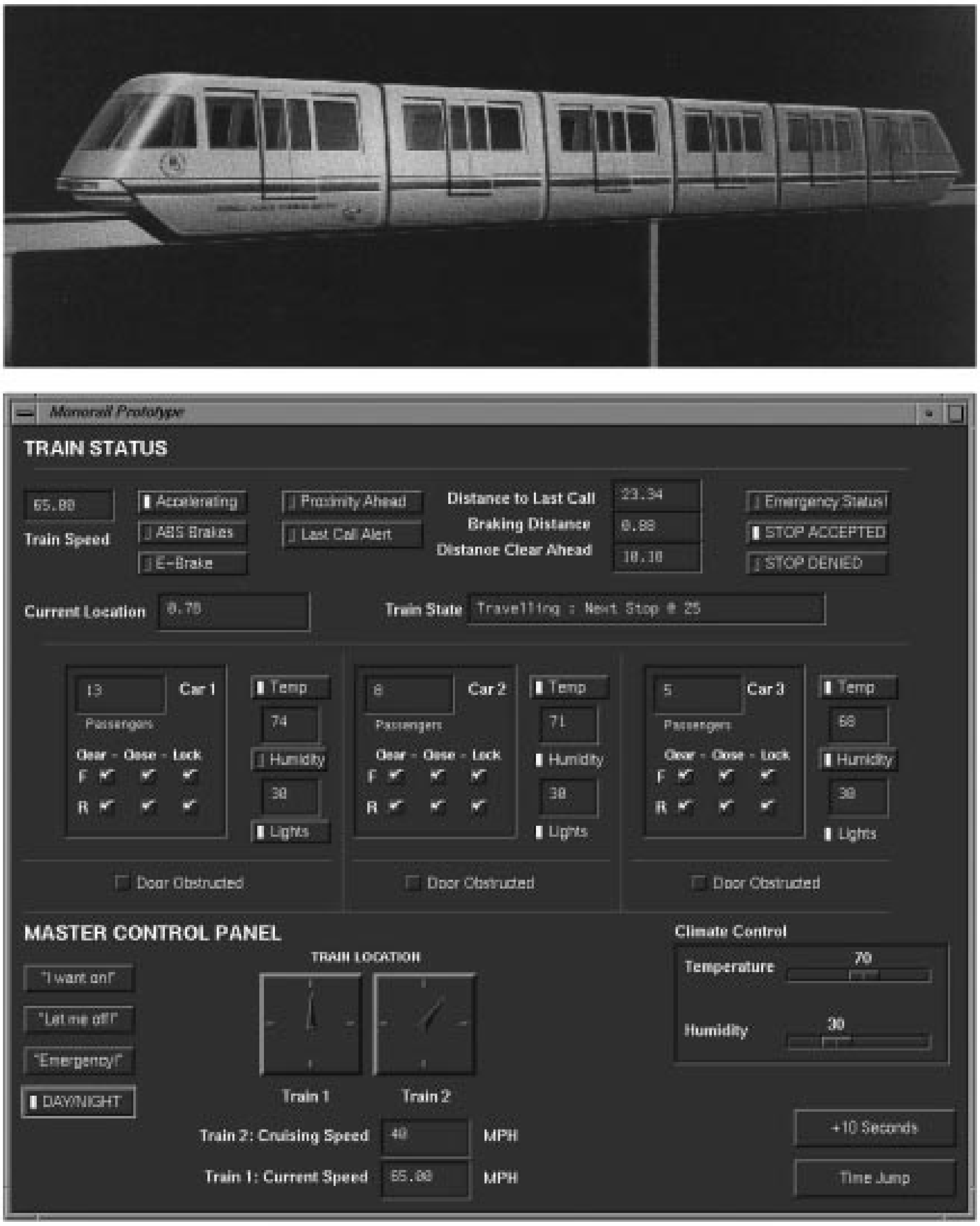


Fig. 2. *RapidApps* prototype for driver information notification system.



Train Speed: 65.88

Distance to Last Call: 23.34
Braking Distance: 0.88
Distance Clear Ahead: 18.18

Current Location: 0.78

Train State: Travelling : Next Stop @ 25

	Car 1	Car 2	Car 3
Passengers	13	8	5
Temp	74	71	68
Humidity	38	38	38

Clear - Close - Lock
F ✓ ✓ ✓
R ✓ ✓ ✓

Temperature 70
Humidity 30

Train 2: Cruising Speed 40 MPH
Train 1: Current Speed 65.88 MPH

Fig. 3. *RapidApps* prototype for the monorail control system.

CosmoCode and RapidApps, the students build functioning Java prototypes of their respective embedded systems in approximately two weeks. As part of their user manual requirements, students use Showcase, a presentation software package on the SGIs that supports the construction of narrated videos. Each group develops a tutorial video for their project that included animated sequences of sample sessions with their prototypes.

The course concludes by having students give presentations and demonstrations of their respective projects. The students gain experience in giving concise and team-oriented briefings to their peers. Students use popular presentation software (Microsoft's PowerPoint) with an LCD projector; an SGI system is connected to the Internet for demonstrations of their RapidApps prototype. Each group also played its

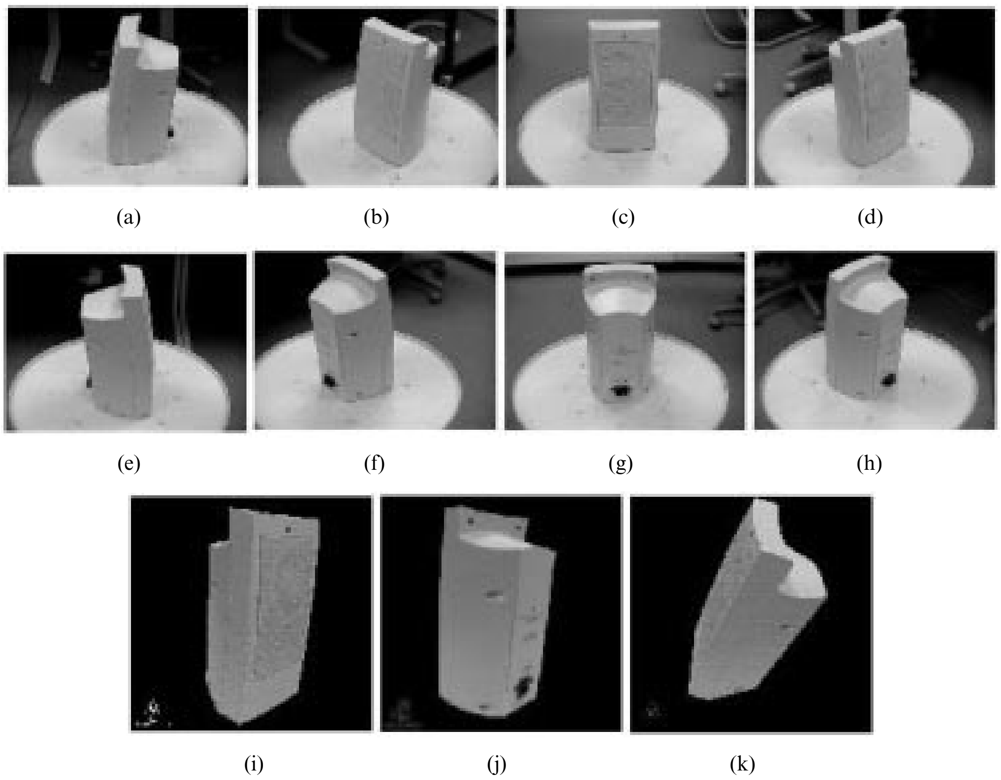


Fig. 4. Example of 3-D model of a computer speaker; (a)–(h) show the images used to generate the model, and (i)–(k) show three views of the 3-D model that are different from all views in (a)–(h).

respective Showcase tutorial video during the presentation.

Computer graphics

Prior to the existence of the Multimedia Laboratory, students in this course learned basic graphics concepts and algorithms. For laboratory assignments, the students learned to use the X-window system, Motif, and PHIGS. While the X-window system is a basic raster-level 2-D windowing system, Motif provides widget sets built atop the X-window system for 2-D graphics, and PHIGS is a 3-D windowing system that also deals with 3-D graphics objects and their rendition [5]. Students were required to implement graphics programs and render the result using these window systems. However, these tools are not efficient enough for production-level use. Due to the limited time that each student can spend on each program assignment these programming assignments involved only simple graphics objects, such as lines and cubes.

Commercially available user-level graphics software can greatly facilitate the graphics content creation, rendition, and animation. The availability and the quality of these commercial graphics software systems depend very much on the hardware platform. The SGI environment is the preferred system for production-level graphics design.

As mentioned earlier, we designed four new laboratory assignments for the course, one to emphasize each of the graphics systems presented in lectures: X/Motif, OpenGL, OpenInventor and Alias/Wavefront PowerAnimator. Example assignments include design of an X/Motif window interface, interactive building and editing of various shapes and objects, 3-D graphics programming of wireframe models, and an animation program that can generate 3-D polygonal objects with different surface color and reflectance.

Each laboratory exercise was designed with the following objectives in mind:

1. the students learn the major characteristics of the system;
2. the graphics content is both interesting and challenging;
3. the balance between new concepts and programming is appropriate.

We have developed two sets of materials for each laboratory assignment, (a) a tutorial for the particular graphics system; (b) a description of the laboratory assignment. The tutorial is supplementary to material covered in lectures, and provides pointers for further reading. Thus, after taking the course, the students are prepared to learn further about these graphics systems whenever the need arises.

Based on course evaluations, the students enjoyed the laboratory experiences, which offered them the opportunity to use professional graphics software in the design of realistic 3-D graphics and animations. Moreover, the students learned which types of advanced tools are required for efficient graphics content generation, and what issues must be addressed for user-level graphics software. Thus, the students not only gained experience in basic graphics algorithms, but also were shown how these algorithms are integrated into professional graphics environments.

Figure 4 shows an example system developed by a student using methods that he learned in the computer graphics course. This system, called Camera-Aided Virtual Reality Builder (CAVRB), constructs a 3-D graphics model from a real object from a number of images taken by a camera at different viewing positions and angles. Using these images, the user interacts with the CAVRB graphical userface to specify the correspondence of surface patches from one image to the next by specifying the corner points of each surface patch in the image plane. Next, CAVRB automatically computes the camera positions and angles and the 3-D position of these feature points from images. Finally, CAVRB computes the texture map of the reconstructed 3-D graphics model by back-projecting image textures onto the curved surfaces in the 3-D model. Figure 4 shows eight real images of a computer audio speaker and three views of a 3-D model constructed using those images. Similarly, Figure 5 shows three real images of a human face and two views of the CAVRB-generated 3-D model.

Computer networks

Although laboratory activities have been an important part of our computer networking course for many years, the availability of the Multimedia Laboratory enabled us to expand this component of the course. The laboratory assignments can be divided into two categories. The first set comprises relatively short assignments, usually one or two weeks, on topics that directly relate to lecture materials. The second set involves projects that are larger in scope and require the students to integrate multiple concepts in the design of a networked application.

We revised these laboratory assignments to make use of multimedia libraries and tools. Early assignments gave the students experience with the environment, such as the SGI graphical debugger; the material covered in these assignments would serve the students throughout the course. In later assignments, we used tools on the SGIs to reinforce concepts related to signaling, sampling, and encoding. For example, the SGIs provide multimedia support as a standard part of the operating system, and these tools allow one to input, manipulate, and convert audio files. Students used SoundEditor, SoundPlayer, and MediaConvert tools (as well as some home-grown error generating software) to conduct a number of experiments with digitally sampled audio involving sampling rates, storage requirements and error tolerance. Additional one-week assignments gave students experience with screen capture facilities and audio and video players, which were later incorporated into projects.

The course projects require students to apply knowledge gained in previous laboratory assignments to the development of larger software systems. Typically, the students build a complete application by integrating their own code with other provided software packages and third-party programs. In addition, these projects always involve an experimental component (usually performance evaluation) and require an accompanying report describing the design and the results of the experimental study.

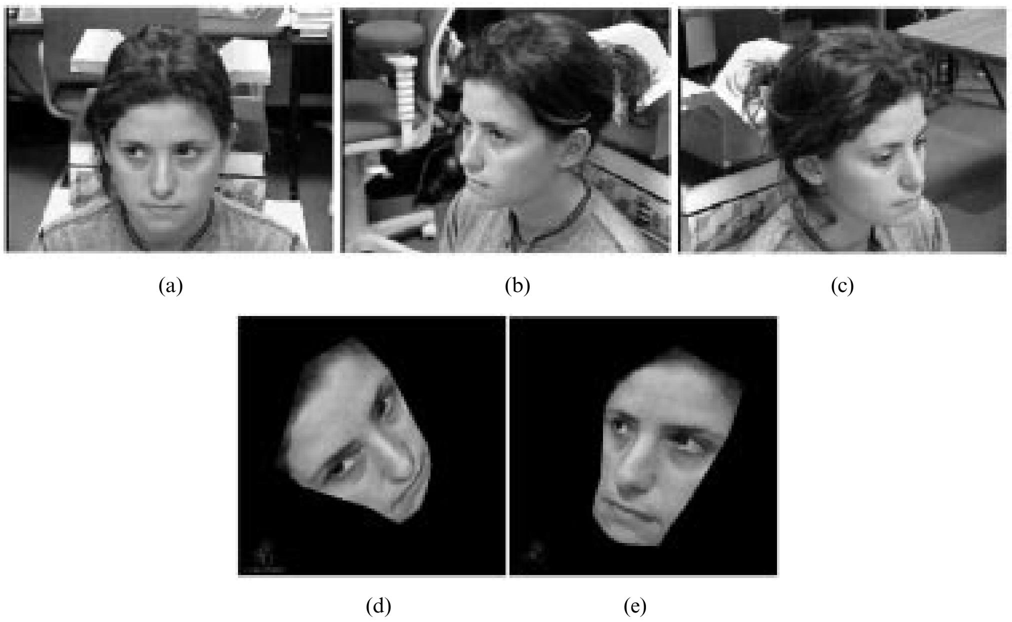


Fig. 5. Example of 3-D model of a human face; (a)–(c) show the images used to generate the model, (d)–(e) show the 3-D model from two different views.

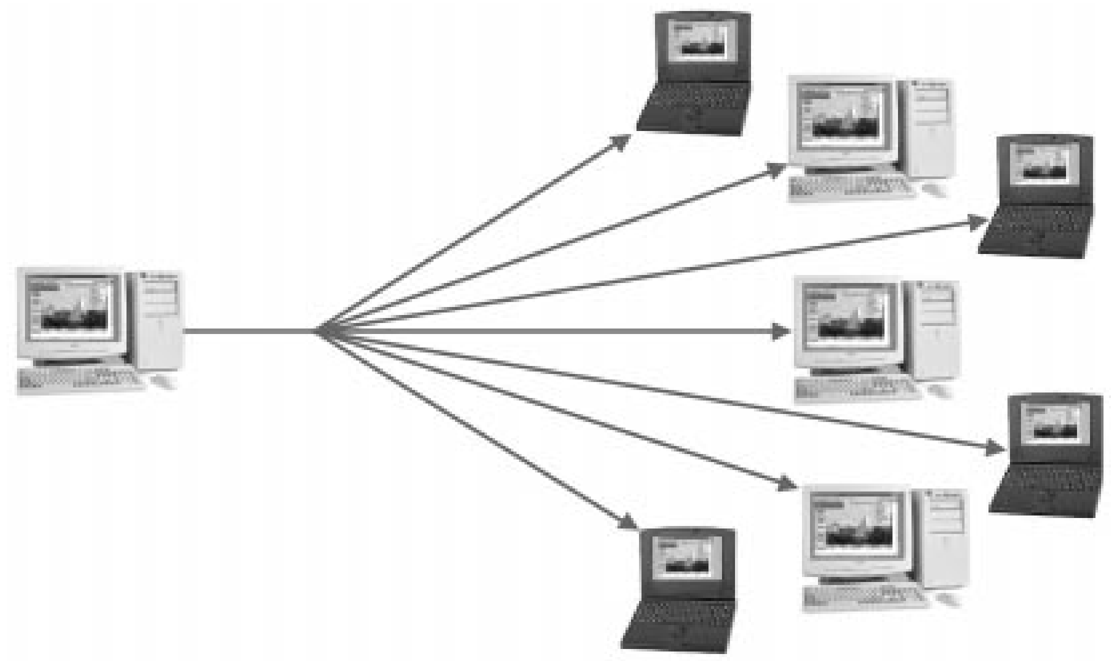


Fig. 6. Synchronous collaborative browser operation—all URLs displayed on the instructor browser are multicast to programs controlling the student browsers, where they are displayed simultaneously.

The theme of these projects varies from term to term. One semester, the students conducted a study of data compression. They implemented the core algorithms in the JPEG image compression standard (specifically, they implemented the discrete cosine transform, quantizer, and binary encoder) and linked their code to a provided set of functions in order to create an image compression library. The students then used their code, with different quantization levels, to transmit images across the network, dynamically compressing the images prior to transmission. Students measured the compression and network delays and evaluated the tradeoff between delays and image quality in their reports. During another semester, the students, having already built a simple file transfer protocol (ftp) program, converted it into a multimedia ftp program. In addition to saving files to disk, this program enabled the user to dynamically play audio and video files, which are streamed over the network in response to user commands. Students accomplished this task by integrating their programs for streaming data with existing audio and video players.

Finally, in one of the more interesting projects, we gave the students an assignment intended to enhance their experience with a specific network service (IP multicast) and enable them to create a relatively complicated networked application. The students were asked to implement a synchronous web browser, as might be used in distance education; see Figure 6. The tool treats a browser as a multimedia blackboard: all web resources brought up on an 'instructor' browser is displayed simultaneously a set of 'student' browsers. The students developed two programs. The first monitors the instructor browser and multicasts any changes in URLs, downloaded files, and so on, to the student machines. The second program, which executes on each student machine, receives these changes and feeds them to the student browser. We provided the students with code to interact with the browser, and asked them to implement the main thread of control as well as the communication-related code. This project was not only very popular among the students, but also led to the development of a production-level version of the tool, called WebClass, for use in instructional laboratories [6].

In summary, the new Multimedia Laboratory has enabled us to expand the scope of the both the laboratory assignments and projects in the computer networks course. State-of-the-art visualization tools enable students to observe and interpret low-level communications phenomena in ways not previously possible. Moreover, by integrating third party packages and components, such as off-the-shelf web browsers and audio tools, students are able to develop large-scale applications efficiently, spending most of their time on the networking aspects of the program.

CONCLUSIONS

In this paper, we have described our experiences with an NSF-sponsored Multimedia Laboratory. Using the new laboratory, we have integrated multimedia technology into our computer science curriculum by expanding the content and laboratory components of three courses: software engineering, computer graphics, and computer networks. A key part of this project is the use of existing software components and multimedia tools. In a field as fast-paced as computer science, it is important that students learn fundamental principles that can be applied to many problems. However, we feel that it is also important to show students how those principles are being applied in the current state of the art. Ours is a three-pronged approach, teaching students fundamental principles, requiring students to apply these principles in the software that they build themselves, and giving students experience with existing software packages and commercial tools that also incorporate these principles.

Our experiences with the laboratory have been overwhelmingly positive. It has strengthened our curriculum and enabled us to provide students with a much richer set of experiences than previously possible. While laboratory exercises have always been an important part of our computer science curriculum, the multimedia extensions are particularly motivating for the students. This observation is supported by course evaluation feedback forms that included supplementary sets of questions on the laboratory-related activities.

The level of interest in this type of material among students, and the experiences gained in the three target courses, have resulted in the development of a new undergraduate multimedia course. This course, which was offered for the first time in Fall 1998, exposes students to the different dimensions of multimedia computing and communication, including digital libraries, data transmission, storage and retrieval, image and audio processing, virtual reality, media transformation, and the group development of multimedia projects. The course makes use of many of the same software packages, laboratory modules, and project assignments that were used in the three target courses.

Acknowledgements—The authors would like to thank the National Science Foundation for their support of the Multimedia Laboratory under NSF grant DUE-9551180. We would also like to thank several faculty members and students who contributed to the development of course and laboratory materials, including: George Stockman, Kurt Stirewalt, Nasrollah Alavi, Barbara Birchler, Robel Barrios, Nicholas Basker, Gretel Coombs, Scott D. Connell, Barb Czerny, David J. Hammond, and Juana E. Nakfour.

REFERENCES

1. W. H. Graves, Toward a national learning infrastructure, *Educom Review of Learning, Communications and Information Technology,* 29, March/April 1994, pp. 32–37.
2. Course on designing a senior level or graduate course in interactive multimedia, *Second Annual ACM Multimedia Conference,* San Francisco, CA, October 1994.
3. B. H. C. Cheng, D. T. Rover and M. W. Mutka, A multi-pronged approach to bringing embedded systems into undergraduate education, *Proc. ASEE Conf.,* June 1998.
4. J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy and W. Lorensen, *Object-Oriented Modeling and Design,* Prentice-Hall, Englewood Cliffs, New Jersey (1991).
5. ANSI (American National Standards Institute), *American National Standard for Information Processing Systems—Programmer's Hierarchical Interactive Graphics System (PHIGS) Functional Description, Archive File Format, Clear-Text Encoding of Archive File,* ANSI X3.144–1988, ANSI, New York (1988).
6. P. K. McKinley, R. R. Barrios and A. M. Malenfant, Design and performance evaluation of a Java-based multicast browser tool, *Proc. 19th Int. Conf. Distributed Computing Systems,* Austin, Texas, 1999, pp. 314–322.
7. Software engineering course projects for CSE 470, Fall 1997. http://www.cse.msu.edu/~cse470/F97.

Philip K. McKinley is a professor in the Department of Computer Science at Michigan State University and was previously a member of technical staff at Bell Laboratories in Naperville, Illinois. Dr. McKinley serves as an Associate Editor for IEEE Transactions on Parallel and Distributed Systems and is co-chair of the program committee for the 2003 IEEE International Conference on Distributed Computing Systems. His current research and teaching interests include distributed systems, mobile computing, and network protocols.

Betty H.C. Cheng is a professor in the Department of Computer Science and Engineering at Michigan State University. She serves on the editorial boards for IEEE Transactions on Software Engineering, Requirements Engineering Journal, and Software and Systems Modeling. Her research and teaching interests include formal methods for software engineering, software development environments, object-oriented analysis and design, embedded systems development, multimedia systems, visualization, and distributed computing.

Juyang Weng is a professor in the Department of Computer Science at Michigan State University. He is an Associate Editor of IEEE Trans. on Pattern Recognition and Machine Intelligence and he is the chairman of the Autonomous Mental Development Technical Committee of the IEEE Computational Intelligence Society. His research and teaching interests include computer vision, computer graphics, artificial intelligence, mobile robots, human-robot interaction using vision, audition, touch, speech, languages and actions.