

ADAPTING MULTIMEDIA FOR DIVERSE STUDENT LEARNING STYLES*

*Glenn D. Blank, Soma Roy, Shreeram Sahasrabudhe,
William M. Pottenger and G. Drew Kessler
Computer Science and Engineering,
Lehigh University, 19 Packard Lab
Bethlehem, PA 18104
610-753-4867
blank, sor2, sas4, pottenger, kessler {@cse.lehigh.edu}*

ABSTRACT

Multimedia can accommodate diverse learning styles. By giving students different ways to learn material, we hope to attract more novices, especially women and minorities, to computer science. We designed a user interface that is independent of both book metaphors and familiar web browsers. It supplies sound and animation for sensory learners, while letting verbal learners disable sound or switch altogether to a JUST THE FACTS mode. Interactive materials include learner-controlled simulations of algorithms, links to programs that students can try immediately after learning related concepts and before exercises that make sure the learner has studied the programs, constructive exercises in which students build programs or models by dragging pieces into place, and inquiry-based exercises in which students learn by doing research, using the web.

1.0 INTRODUCTION

CIMEL is a multimedia framework for Constructive and collaborative, Inquiry-based E-Learning supplementing computer science courses. *Constructive* learning goes beyond learning by receiving knowledge, to learning by building systems, with immediate, visual

* This project is funded by National Science Foundation (Grant Number EIA-0087977).

Copyright © 2002 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

feedback. *Collaborative* learning encourages students to interact with instructors and librarians via live links and remote-controlled “show me” sessions and by reviewing multimedia FAQs of recorded “show me” sessions. *Inquiry-based* learning guides the student into pursuing exploratory research in a community of students and scholars. A text mining and visualization tool enables students to identify and explore emerging technology trends in computer science as part of our inquiry-based framework. Within the CIMEL multimedia framework, we have developed prototypes of new materials for different courses in computer science at different levels: a first semester course in computer science (CS0) and an upper level course in software engineering (SE). New multimedia modules, implemented in Macromedia Flash and played through high-speed connections on the web, feature audio narration, animation, simulations, quizzes, and constructive exercises.

For CS0, we began revising both the manuscript and multimedia of *The Universal Machine: A Multimedia Introduction to Computing* [1], tentatively re-titled *The Universal Computer: Introducing Computer Science with Multimedia*. A first draft of the new textbook and an alpha version of the multimedia was completed in time for use in the fall of 2001. During the fall semester, students in the Introduction to Computing course at Lehigh University began using the new material, introducing Java programming and the breadth of computer science.

For the SE course, we developed multimedia units on Inheritance and Dynamic Binding and on Abstract Data Types (ADTs). The latter formalizes the meaning of classes in connection with object-oriented design. In the past, upper level students have found it difficult to master this material from lecture and textbooks alone. Our premise is that multimedia will help students understand the material better, objectively, and also improve their design of actual ADTs to solve a problem.

Figure 1 illustrates several features of the CIMEL multimedia framework:

- The TRACKLIST at left displays the content of a lesson as a sequence of screens. The menu uses check marks to show progress and highlights the current screen in red. Some chapters, such as those of Java and software engineering, will have material in more advanced tracks, which will go beyond the content of the proposed textbook, for students and faculty who wish to explore these topics in more depth.
- Multimedia personae model a community of learners and instructors. The personae include two professors, a teaching assistant (shown here), a reference librarian, and two students. In addition to graphical images, they speak in audio and/or text boxes. These personae model students and teachers studying material together, working through interactive and constructive exercises, and suggest exploratory research on relevant topics using online information.
- The COLLABORATE button brings up tools that facilitate network-based interaction with other students, instructors, teaching assistants and librarians. Live links, remote-controlled SHOW ME sessions and a multimedia FAQ (mFAQ) of recorded SHOW ME sessions will encourage students to get help.

1.

Figure 1: Screen Capture from *The Universal Computer* (as of June 2002)

- The EXPLORE button facilitates inquiry-based learning, via directed queries on the web and an “emerging trends” text mining and visualization tool. An emerging trend is a topic area for which one can trace the growth of interest and utility over time. (A tutorial setting up an emerging trends exercise is included in the inheritance section of the prototype.)
- The FIND button supports a glossary and search through multimedia pages.
- The PREFERENCES button presents a panel of options letting the user adapt the environment according to his or her personal learning style, including turning text boxes or audio on/off, toggling auto-advance or wait for next page, setting the timing rate where there is no audio narration, etc. A user may change these settings at any time during these sessions and they will be recorded locally and on a network drive for the next session.
- The JUST THE FACTS button lets users switch to viewing non-interactive content (text and graphics) presented in HTML pages. From there, one can switch back to rich

media mode via hyperlinks anchored to the corresponding Flash page. There are also links to interactive screens, which remain in Flash.

Our user interface design has been driven by a process of user-driven evaluation, including focus group and usability surveys. This process has helped us to design an interface that accommodates the diverse preferences of students: some enjoying the rich media and others wanting a more straightforward JUST THE FACTS presentation, some liking the redundant text and audio while others found either sound or text annoying. (An anecdotal note: the principal investigator has gotten rave feedback about the rich media from many women.)

The CIMEL framework also includes web-based tracking of all learner interactions to a database on a server. We have used this tracking information to study student usage of the multimedia, noting all interactions with the interface (such as changes of PREFERENCES and switches to JUST THE FACTS mode), all screens visited, and all responses to interactive exercises.

In the fall of 2001, we conducted experiments, surveys and focus groups in two courses to determine whether the alpha version of our multimedia actually improves learning. As reported in [2], we found significant learning effects in a focused study on abstract data types in a graduate level OOSE course and marginal results in the broader study in a CS0/CS1 course. In the spring of 2002, we made improvements to the alpha in response to what we learned from our earlier results, and conducted further experiments in an upper level SE course. We will report the design and results of this study in this paper.

2.0 INTERACTIVITY AND CONSTRUCTIVE EXERCISES

Interactivity is a key aspect of the CIMEL content. Interactive quizzes and constructive exercises help students learn by doing. Personae provide feedback guiding a student through each exercise. In Figure 2, the TA persona changes expression and provides feedback hinting at what is wrong with a user choice in a multiple choice question.

Constructive exercises are much more complex, challenging a learner to build solutions to problems by dragging and dropping pieces of structures into place, incrementally. Figures 3 and 4 are snapshots from a sequence of screens, in which a professor persona walks the learner step by step through the process of building an abstract data type for an Apple. Previous screens presented the components of ADTs, interspersed with simpler interactive questions, and culminating with the simulation of an ADT for Employee. Finally, in this constructive exercise, the learner goes through a series of screens, incrementally building up an ADT specification for class Apple. In figure 3, the learner constructs the signatures section of Apple ADT by first choosing a member function, then in figure 4 selecting that function's arguments. Later on in the exercise, the learner builds the preconditions and postconditions for Apple ADT. Finally, the learner runs simulations running the functions, preconditions and postconditions, testing them for completeness. At each step, feedback helps the learner learn from mistakes as well as correct actions.

The screenshot shows a quiz titled "Inheritance :: Methods Quiz". On the left is a "TRACK LIST" sidebar with items like "Inheritance", "Generalization", "Visualization", "Polymorphism", "Uses in Java", "Dynamic Binding", "Extending a Class", "Keyword Quiz", "Abstract Methods", "Implementing Methods", "Method Calls", "Methods Quiz" (highlighted in red), "Final Methods", "Code Inlining", "Summary", and "Abstract Data Types".

The main question asks: "In superclass Shape, which method declaration should include the keyword 'abstract'?"

Three options are listed:

```
public double area()
public void setName(String name)
public void changeColor( int R, int G, int B )
```

A yellow callout box contains the text: "Wouldn't this method do the same thing in all subclasses? If a superclass can implement a method, it should not be declared abstract." This indicates that the selected option (the first one) is incorrect.

At the bottom, there is a navigation bar with icons for: HOME, THEORY, FIND, TOOLS, PREFERENCES, INFO, JUST THE FACTS, BACK, PAUSE, and NEXT.

Figure 2: TA persona responds to a wrong choice in a multiple-choice question

The screenshot shows a software interface for defining an ADT. On the left is a 'TRACK LIST' with various topics, including 'Names, Sets, Signatures' which is highlighted. The main area is titled 'Abstract Data Types :: Names, Sets, Signatures'. It contains three input fields: 'NAME' with the value 'Apple', 'SET' (empty), and 'SIGNATURES' (empty). To the right of the 'NAME' field is the text 'The name of our ADT is Apple.' Below the input fields are buttons for 'VARIABLES', 'PRECONDITIONS', and 'POSTCONDITIONS'. A green 'Instruction' box says 'Click on Apple's functions below, one at a time.' Below this is a 'Done - no more functions' button and a list of functions: getCenter, setDiameter, print, String, setColor, RealNumber, Apple, size, Symbox, move, red, Point, Color, Integer, Center, green, Diameter. At the bottom, there is a video inset of a man speaking, a speech bubble that says 'Click on Apple's functions, one at a time.', and a toolbar with icons for 'CHALLENGE', 'EXPLORE', 'FIND', 'TOOLS', 'PREFERENCES', 'HELP', 'EXIT THE PAGE', 'BACK', 'PAUSE', and 'ROLL'.

Figure 3: Excerpt from a constructive exercise: choosing functions for ADT Apple

The screenshot displays an educational application window titled "Abstract Data Types :: Names, Sets, Signatures". On the left is a "TRACK LIST" with items like "Inheritance", "Abstract Data Types", "ADT introduction", "ADT sections", "ADT for Collections", "List", "Queue", "Stack", "ADT as contract", "ADT and inheritance", "Apple constructive", "Explanation", "Apple Sections", "Names, Sets, Signatures", "Preconditions", "Constructor", "Function postcond", "Apple simulation", and "ADT summary". The main workspace has fields for "NAME" (filled with "Apple"), "SET", and "SIGNATURES" (containing "getCenter () →"). Below these are sections for "VARIABLES", "PRECONDITIONS", and "POSTCONDITIONS". A red text prompt says "The name of our ADT is Apple." and another says "getCenter() is a full function as it always produces the specified type of output." An instruction in green text says "Instruction : Click on the type that getCenter returns :". A list of types is shown: "getDiameter", "setDiameter", "print", "String", "setColor", "RealNumber", "Apple", "size", "Syntax", "move", "getColor", "Color", "Integer", "red", "Point", "Center", "green", "Diameter". A video inset shows a man speaking, with a speech bubble: "Now, what does getCenter return? Click on that type." At the bottom are navigation icons: "COLLABORATE", "PUBLISH", "FIND", "TOOLS", "PREFERENCES", "END", "JUST THE FACTS", "BACK", "PAUSE", "NEXT".

Figure 4: Choosing the function’s return value for Apple’s ADT signatures.

Another type of constructive exercise involves the review, modification or development of actual programs in a programming environment. The TOOLS button lets the user invoke either of two different available programming environments: BlueJ (a Java programming environment designed for beginners and freely available from www.bluej.org) or JavaEdit (Dick Chase's free, relatively small and easy to use tool). A specially designed ActiveX control lets the Flash page access these tools from a browser securely. In figure 5, the TA explains BlueJ before asking the learner to try out a program. On this screen, the BlueJ button have been overloaded to specifically load the shapes project into the BlueJ IDE. After the learner has experimented inside a programming environment, subsequent screens typically ask questions about the material, then challenge the learner to make changes to the program or create a new program similar to this one.

Inquiry-based exercises facilitate learning by doing research. For example, after studying ADTs for collections, the screen shown in Figure 6 asks the student to investigate the design of similar ADTs in the most recent Java Development Kit (JDK 1.4). The following screen then asks a followup question designed to find out what the student has learned from the inquiry-based exercise.

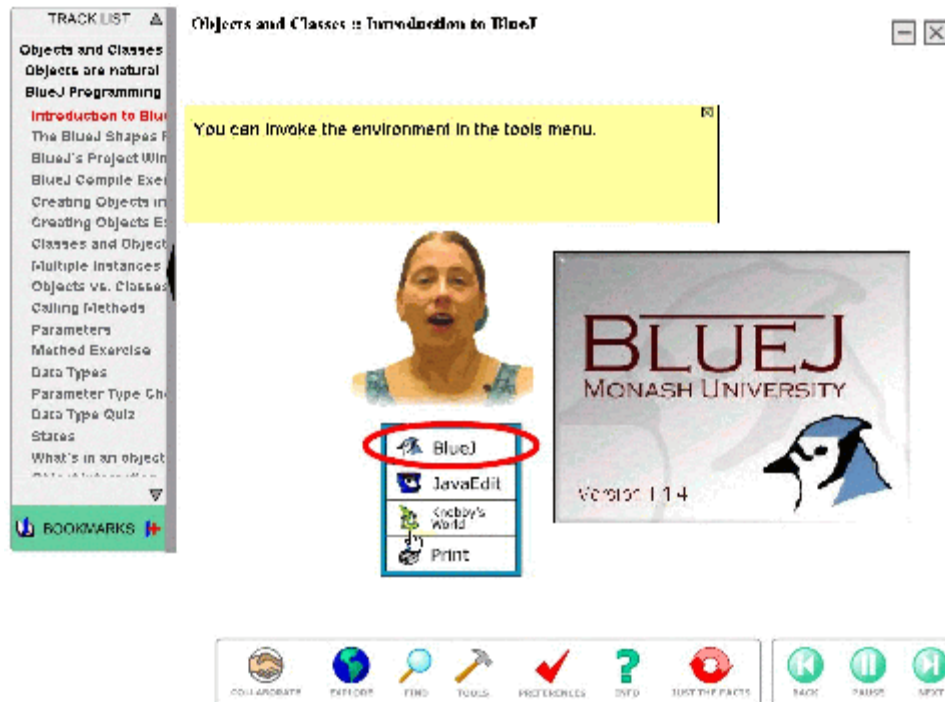


Figure 5: Invoking BlueJ from the TOOLS menu



Figure 6: An inquiry-based exercise researching the Java Development Kit

A text mining and visualization framework for detecting emerging trends facilitates more elaborate inquiry-based learning exercises [2]. An emerging trend is a topic area for which one can trace a growth of interest and utility over time. Detecting emerging trends in a multimedia lesson stimulates inquiry-based learning by providing an avenue of research into key developments in related fields. The CIMEL framework allows the exploration of technology fields closely related to the course work. We conducted a study in which students in an undergraduate programming languages class used a multimedia-supported tutorial and methodology to study emerging trends related to inheritance in object-oriented software. With a confidence of 95%, the precision of students using this methodology to detect emerging trends was significantly higher than students who did not use this methodology [3].

In the background, CIMEL records user interactions in a database, for user interface buttons as well as interactive exercises. Each user's preference settings, tracks and bookmarks are maintained between CIMEL sessions, allowing the user to customize the multimedia environment to their own personal taste. These data are also analyzed for project evaluation, e.g., to zero in on potential usability problems with the system and to study possible correlations between use and learning styles. We foresee that instructors will also use this data to track class or individual performance.

3.0 EVALUATION METHODOLOGY

In the fall of 2001, we conducted a study on abstract data types (ADTs) in a graduate level course on Object-Oriented Software Engineering (OOSE). We designed an experiment to determine whether multimedia actually improves learning, both in terms of objective knowledge and in terms of students' ability to perform a task designing ADTs for a sample problem involving several classes and inheritance. Mean scores on objective tests improved significantly, suggesting that the multimedia does indeed contribute to objective learning of this content [2]. However, the results for the results for *task learning* were less clear. These mixed results may have been due in part to the design of the experiment and in part to the design of the multimedia. Learning from our experience, we have improved both.

With respect to the multimedia design, focus groups and usability surveys led us to improve both the user interface and content in many ways. For example, the just the facts mode of presentation of the content now provides an alternative to the Flash media-rich version of the content. Students in both first year and graduate level focus groups were roughly split about whether they liked media-rich techniques such as animated, graphical personae and audio narration in the alpha version. Those who liked the rich media included most of the women and non-majors, while those who found these media a bit annoying and wanted to get right to the nuts and bolts tended to be male and more experienced in computer science. Even students who prefer the media-rich presentation would like a way to review the material in a less media-rich form. This result suggested that it would be useful to provide HTML pages with the text and key graphics from Flash content pages, dropping the use of personae and audio. Interactive exercises remain as is. Finally, a summary with a checklist now appears at the end of the module, summarizing what one should consider in designing ADTs. Each item in the

checklist includes a link back to the relevant screen, for review. Students in the OOSE focus group strongly agreed that a checklist would give them a better idea how ADTs were designed as well as a good review.

With respect to the experimental design, we switched to a 2x2 design, dividing a class of 72 students in an upper level SE course into four groups: One fourth gets neither the multimedia nor lecture, a second gets just the multimedia, a third gets just a lecture, and the fourth gets both the multimedia and lecture. All four groups get the same homework problem description:

An artist wants some software to represent objects he might manipulate in still life paintings. For our prototype, we want to design a system that manipulates different kinds of fruit in a bowl. Fruit generally have the following attributes of interest: they have a color, a size, and a center located in two-dimensional space. To keep the prototype simple, colors will simply be strings, and the only geometric attribute we will consider is size. The artist wants to be able to manipulate these attributes: changing colors, growing (or shrinking) size, moving centers. However, different kinds of fruit have different default colors and sizes. Apples are (by default) red and 2" in diameter, bananas are yellow and 6" in length, grapes are purple and come in clusters of seven grapes. When an apple changes size, its diameter grows (or shrinks) by some specified amount (positive or negative), but bananas grow longer or shorter, and grapes add or remove some whole number of grapes to the cluster. Finally, a bowl is a collection of fruit, to which the artist may add or remove individual fruit (apples, bananas and grapes), access individual fruit (in order to change their attributes), and print out the collection with their current attributes. For the prototype, we will defer all user interface concerns; you may assume that all operations will be demonstrated by a main driver function.

An analyst has already created a UML class diagram for this problem, which you can find in

<http://www.cse.lehigh.edu/~cimel/eval/beta/fruitUML.jpg>. Your job is to flesh out a design of this problem as a set of abstract data types (ADTs). You can find notes explaining the form of ADTs and several examples in <http://www.cse.lehigh.edu/~cimel/eval/beta/sampleADT.htm>.

Hand in: a set of ADTs for the Fruit problem, as a text document.

Following criteria in www.cse.lehigh.edu/~cimel/eval/beta/ADTevalcriteria.htm, the first author graded the assignments blind. After completing the assignment, students in all four groups completed an online post-test of 20 randomly ordered multiple-choice questions.

Our hypotheses were that 1) the multimedia group would perform better than those getting nothing and 2) the multimedia plus lecture would perform better than the lecture only group, on both pretest to post-test improvement and task grades. We also studied the tracking data, demographics and a learning style inventory for correlations between JUST THE FACTS mode, PREFERENCES settings, and learning styles.

We used the C.I.T.E. Learning Styles Inventory by Babich, Allbright and Randol. Though Thomas et al. [8] use the Felder and Silverman model to investigate learning styles in introductory computer science courses, we preferred the C.I.T.E. inventory because it distinguishes more finely between learning styles, notably between visual written and auditory styles, while Felder and Silverman lump written and auditory learning into a single "verbal" style. Our focus groups suggest that we should distinguish these styles, leading to our PREFERENCES settings and JUST THE FACTS mode.

4.0 RESULTS

The results are promising. The students getting multimedia score higher, on both the assignment, $F(1,57)=12.25$, $p<.01$, and the post-test, $F(1,58)=15.39$, $p<.001$. On the other hand, the lecture did *not* have a significant effect on either assignment grades, $F(1,57)=1.11$ or the post-test, $F(1,58)=0.55$, and there was no interaction between groups, for either the assignment, $F(1,57)=0.01$, or the post-test, $F(1,58)=0.43$. Thus the new multimedia has a significant effect on learning, both in terms of the objective knowledge (the post-test) and task knowledge (the assignment).

Good as these results are, closer examination of the task results indicated that further improvement of the multimedia was possible. With or without the multimedia, most students missed an important component of the assignment—reusing a given ADT (List) to construct the semantics of a new ADT (Bowl). This observation led to a redesign of the multimedia to make this point clearer: rearranging materials on ADT for collections as a separate chapter and adding another exercise that explicitly gets learners to solve problems from existing ADTs. Further improvements have also been made in response to feedback to student comments on a usability survey. We plan to conduct another study in the fall of 2002 to determine whether these changes improve performance, particularly with respect to applying constructive semantics. Thus, evaluation is helping to drive our development process.

With respect to learning styles, there was no statistical significance between scores achieved by students on the assignment or the post-test, who were visual, auditory, or combination learners. Additionally, there was no significant difference between scores for students who were group, individual, or combination learners. Finally, there was no significant difference between scores for students who demonstrated a preference for oral or written expressiveness. The multimedia did not provide a learning advantage for any particular cognitive or social learning style, or expression mode. We plan to conduct further studies about learning styles in a class of first year students.

5.0 RELATED WORK

There is a rich and growing literature of multimedia-based educational material. The CIMEL framework seeks to provide a framework for curriculum development in computer science and to support multiple learning styles and tracks. In this section we compare our approach to a selection of comparable research efforts and systems.

The ProgramLive application [5] is a rich multimedia tutorial of the Java programming language. ProgramLive's interface represents a notebook, within a browser. There are tabs to the side of the notebook display that can be used for the navigation of the material, as well as pop-up explanation of key terms. The CIMEL user interface also plays through a browser, but avoids mixing interface metaphors by eliminating all of the usual buttons of a browser. The interface thus immerses the learner in an environment uniquely associated with the material. Another difference between ProgramLive and CIMEL is in their approach to feedback. CIMEL provides an explanation of each of the wrong answers as the student makes these mistakes rather than just providing the correct answer. This helps the student to gain a better understanding of the thought process involved in solving problems.

The TILE (Technology Integrated Learning Environment) project [5] is developing and evaluating an integrated system for web-based education. This system uses web-based delivery of course material including interactive multimedia presentations, problem solving and simulation environments in which students learn by doing. Like CIMEL, TILE provides students with an interactive multimedia environment, and developers with a framework for managing, authoring, monitoring and evaluating multimedia. The most salient differences are that the CIMEL framework lets students go beyond the lessons through collaboration with experts (e.g., instructors, TAs, research librarians and other student) as well as through tools that allow the student to explore current research trends in course-related literature.

The Interactive Learning Modules (ILM) presents web-based multimedia tutorials, created with the Director authoring environment [6]. ILM provides a mechanism for the creation of supplementary material for lectures, and collaborative problem solving environments. The system is highly modular to encourage the usage of parts of the lesson material in different courses. Similarly, the CIMEL multimedia framework is being developed with modularity in mind, where each screen is a separate Flash movie, and screens are organized hierarchically into sections and chapters. (We chose Flash instead of Director or Authorware because its vector-based graphics lets us easily use the whole screen, regardless of resolution.) The CIMEL dynamic tracks interface will let instructors and students create and traverse their own learning track, corresponding to their unique requirements.

Another system that provides course material in a student-based manner is MLE (Multimedia Learning Environment), a networked educational application [7]. MLE provides a virtual learning environment, through a client application, where multimedia educational material is structured in Adaptive Hypermedia from which sets of hypermedia pages are dynamically retrieved and presented to the student by tailoring both contents and presentation style to the student's needs. In addition to this dynamic content, the MLE system provides a mechanism for real-time audio communication between a student and an instructor. CIMEL also uses a client-server model to present multimedia through a high-speed network.

6.0 CONCLUSIONS

CIMEL is a multimedia framework for Constructive and collaborative, Inquiry-based E-Learning supplementing computer science courses. Within the CIMEL framework, we have developed and evaluated prototypes of materials for upper level courses in computer science. This results of our studies are promising. We plan future studies, including one looking at the effectiveness of multimedia for learning Java with BlueJ; we hope to present the results of this study at the conference.

After analyzing the results of the current study and usability surveys, we plan to make further refinements to the user interface, content and development environment for CIMEL content. This summer, we plan to investigate whether CIMEL multimedia can be adapted to target minority college and high school students. A student from a historically minority institution, a secondary teacher from a local school district, and several high school students

from the Students That Are Ready (S.T.A.R.) Academies will join the CIMEL development team. They will:

- Conduct focus groups and web-based surveys of students and teachers at a historically black college and at local high schools, showing them the current CIMEL user interface and content, to determine effectiveness for these audiences.
- Propose and design possible changes to the user interface, that might make it more suitable for black or Hispanic undergraduate students. For example, we are developing a professor persona who is female and black (from Trinidad).
- Implement proposed changes to user interface and content.

Once the user interface stabilizes, our efforts will shift to developing material for a CS0 course, which can be used either with a new textbook, *The Universal Computer: Introducing Computer Science with Multimedia*, or as a stand-alone product. Since we are designing material for a wide range of students, we will develop a dynamic TRACKS interface that will let instructors and students manipulate what gets presented. For example, much of the ADT material may be too advanced for first year students; an introductory track on software engineering will include an excerpt from this material; the TRACKS interface will let instructors and students select alternative materials supplied by the developers.

Additional documentation about the CIMEL project is available at www.cse.lehigh.edu/~cimel/documentation.html. The prototype is available at www.cse.lehigh.edu/~cimel/prototype.html. We welcome beta testers and inquiries!

ACKNOWLEDGEMENTS

The three principal investigators wish to thank God for inspiring us to begin this project and for blessing the work of our hands. We also wish to thank Edwin J. Kay for his help with experimental design and analysis, Morgan Jennings and Debra Dirksen of the Metropolitan State College of Denver for their work on learning styles evaluation, as well as all the students who have worked on software design and development, including David Gevry, David Goldfeder, Jeffrey Heigl, Martin Herr, Harriet Jaffe, Sumit Jain, Chris Janneck, Adam Kinnear, Jeffrey Lutz, Jonathan Lutz, Andrew Mall, David Servas, Aaron Sherrick and Qiang Wang.

REFERENCES

1. G. D. Blank and R. Barnes, *The Universal Machine: A Multimedia Introduction to Computing* (WCB/McGraw-Hill, 1998).
2. G. D. Blank, W. M. Pottenger, G. D. Kessler, S. Roy, D. Gevry, J. Heigl, S. Sahasrabudhe and Q. Wang. Design and Evaluation of Multimedia to Teach Java and Object Oriented Software Engineering. *American Society for Engineering Education* (Montreal, June, 2002).
3. Soma Roy, David Gevry and William M. Pottenger, Methodologies for Trend Detection in Textual Data Mining, *Proceedings of the Textmine '02 Workshop, Second SIAM International Conference on Data Mining*, April 2002.

4. D. Gries and P. Gries. *ProgramLive - Master Java Programming in a Self-paced Learning Environment*. New York, John Wiley and Sons, Inc. 2002.
5. C. Jesshope, E. Heinrich and Kinshuk. *On-line Education using Technology Integrated Learning Environments*. Massey University, New Zealand. www-file.massey.ac.nz/publicns.html.
6. D. M. Millard, Interactive Learning Modules for Electrical Engineering Education and Training. In *Proceedings of the American Society for Engineering Education*, 1999.
7. M. Rocchetti and P. Salomoni. A Web-based Synchronized Multimedia System for Distance Education. In *Proceedings of the 16th ACM Symposium on Applied Computing*, 2001, pp. 94 – 98.
8. L. Thomas, M. Ratcliffe, J. Woodbury and E. Jarman, Learning Styles and Performance in the Introductory Programming Sequence. In *Proceedings of SIGCSE 2002*, pp. 33-37.