

2005

# Teaching content creation with programming

P. McKerrow

*University of Wollongong, [phillip@uow.edu.au](mailto:phillip@uow.edu.au)*

---

## Publication Details

This article was published as: McKerrow, PJ, Teaching content creation with programming, IEEE Multimedia, July-September 2005, 12(3), 36-45. Copyright IEEE 2005

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: [research-pubs@uow.edu.au](mailto:research-pubs@uow.edu.au)

---

# Teaching content creation with programming

## **Abstract**

Multimedia is an art form that uses computers as a means of personal expression. However, much multimedia content is unattractive, and many applications have poor user interfaces because many developers don't have training in both programming and content creation. To counteract this, we teach students QuickTime for Java, which provides a rich library for manipulating media. Our students reported an increased understanding of multimedia programming and developed creative skills.

## **Disciplines**

Physical Sciences and Mathematics

## **Publication Details**

This article was published as: McKerrow, PJ, Teaching content creation with programming, IEEE Multimedia, July-September 2005, 12(3), 36-45. Copyright IEEE 2005

# Teaching Content Creation with Programming

Phillip John McKerrow  
*University of Wollongong, Australia*

**Multimedia is an art form that uses computers as a means of personal expression. However, much multimedia content is unattractive, and many applications have poor user interfaces because many developers don't have training in both programming and content creation. To counteract this, we teach students QuickTime for Java, which provides a rich library for manipulating media. Our students reported an increased understanding of multimedia programming and developed creative skills.**

**H**ave you ever wondered why so many Web sites are unappealing to the eye, lack audio presence, and don't inspire you to linger? Instead of capturing your attention with pleasing aesthetics they bore you with bland information. My colleagues and I hypothesize that they're unattractive because the programmers had no training in creative design when they developed them. As a result, the programmers didn't have the skills to create the media assets and didn't understand the processes involved in media creation. We attempt to address this problem by including content creation in the programming classes we teach.

In 2003, the School of Information Technology and Computer Science at the University of Wollongong introduced a Master of Digital Multimedia degree (see [http://www.uow.edu.au/handbook/yr2005/mast\\_multimedia.html](http://www.uow.edu.au/handbook/yr2005/mast_multimedia.html)). The aim of this degree is to equip computer science graduates to work on multimedia projects with creative professionals. To achieve this goal, the faculty planned their curricula so that it integrated programming and creative design. Creative design and Java programming form the degree's compulsory core. We supplemented these subjects with a range of creative and programming subjects.

We considered this an experiment, and as such we first looked at the nature of multimedia from both an artistic and technological perspective to define a programmer's perspective. Then we looked at the choice of language for teaching

multimedia programming. To determine what to teach, we developed a model of multimedia. From this model we developed a subject structure, and finally we looked at representative assignments and student experiences.

We conducted two classes from 2000 to 2002 as pilot studies: Multimedia Studies and Web Design. In Web Design, we taught the design of dynamic Web sites using Apple WebObjects and Adobe tools. Multimedia Studies consisted of four strands: programming, creative design, professional tools, and assignments. In the programming strand, students programmed multimedia applications using Apple's QuickTime for Java applications programming interface (API). In the creative design strand, they studied how to create digital video and audio. In the professional tools strand, they learned to use professional tools for media production, in particular, Apple's Final Cut Pro.

Students received four assignments: making a video, two QuickTime for Java assignments, and a project to make a multimedia kiosk. In their first assignment, they worked in groups of two to make a 30- to 60-second video. They used this video as media in their two QuickTime for Java assignments. In the final assignment, they worked in groups of four to build a multimedia kiosk.

## Multimedia

The first question you ask when you teach multimedia is, What is multimedia? Definitions of multimedia fall into two groups depending on whether you view it as a technologist or an artist. To the artist, multimedia is a means of expression, a means of communicating ideas to other people. As Packer and Jordan say, "Multimedia is the form that makes the most complete use of the computer's potential for personal expression."<sup>1</sup>

Not all digital media is multimedia. For a production to be considered multimedia, it must include a provision for the user to interact with the content and influence the course of the presentation. At its core, multimedia involves a person interacting with artistic content using a computer. As Tannenbaum notes,

Multimedia is defined as an interactive computer-mediated presentation that includes at least two of the following elements: text, sound, still graphic images, motion graphics, and animation.<sup>2</sup>

Technologists define multimedia by a set of standards that enable media to be acquired, rep-

resented, compressed, delivered, and displayed. To this end, the Moving Picture Experts Group is in charge of developing standards for coded representation of digital audio and video. They're in the process of developing three standards (MPEG-4, -7, and -21) that directly relate to multimedia.

MPEG-4 (multimedia content representation) is a standard for multimedia content on Web pages. It describes multimedia as follows:

- media objects to represent units of aural, visual, or audiovisual content;
- the composition of these objects to create complex media objects that form audiovisual scenes;
- algorithms to multiplex and synchronize the data associated with media objects, so that we can transport them over network channels providing a quality of service appropriate to the nature of the specific media objects; and
- mechanisms for the user to interact with the audiovisual scene generated on the Web page.

The goal of MPEG-7 (multimedia content description) is to provide a standard for describing audiovisual data content so that we can search for it easily. Audiovisual data content that has MPEG-7 data associated with it may include still pictures, graphics, 3D models, audio, speech, video, and composition information about how these elements are combined in a multimedia presentation (scenarios). A special case of these general data types is facial characteristics.

The goal of MPEG-21 (multimedia framework) is to define the technology needed to support users to exchange, access, consume, trade, and otherwise manipulate digital items in an efficient, transparent, and interoperable way. MPEG bases it on two essential concepts: the definition of a fundamental unit of distribution and transaction (the digital item) and the concept of users interacting with digital items. The digital items represent the "what" of the multimedia framework (such as a video collection or a music album) and the users are the "who."

As the goal of this subject is to combine creative content design with an understanding of the technology in a way that's attractive to computer science students, we wanted to view multimedia from a programmer's perspective. A programmer's task is to develop new multimedia applications.

Programmers need to understand the standards, but usually don't want to write a new codec to implement a standard. Instead they prefer, whenever possible, to use a multimedia API that provides the tools to build their applications. In addition, when programmers design user interfaces, they're concerned with how humans perceive the media and the processes they use to create content.

### **Multimedia characteristics**

To develop a multimedia application, programmers must understand the communication goal of the artists who will develop multimedia content for use with the application. Also, to develop an application for a particular medium, programmers must understand its nature, data management, and the user process involved in creating the media assets. Then they must address each of the medium's characteristics in their designs. Multimedia has five characteristics: integration, interactivity, hypermedia, immersion, and narrativity.<sup>1</sup>

Integration combines artistic forms and technology into a hybrid form of expression. To enable the user to integrate content, programmers must consider the concepts of compositing video data and mixing audio data.

Interactivity is the users' ability to manipulate and affect their experience of media directly, and to communicate with others through media. To achieve interactivity, programmers must consider mechanisms for user interaction, such as the mouse rolling over hot spots.

Hypermedia involves linking separate media elements to one another to create a trail of personal associations. To create this trail, programmers must consider how to navigate between elements of the multimedia presentation, such as Web hyperlinks or DVD chapter-index menus.

Immersion is the experience of entering into the simulation or suggestion of a 3D environment. To immerse users, programmers must consider whether the application requires an immersive environment (such as a virtual world<sup>3</sup>) and whether the users are objective viewers or subjective participants.

Narrativity is the aesthetic and formal strategy that derives from the aforementioned concepts, which results in nonlinear story forms and media presentations. To achieve narrativity, programmers must determine whether the task has a natural flow (which might dictate sequential interaction), or offers an arcade of choices (which might dictate support for arbitrary interaction).

### Media characteristics

Multimedia is hard both for the computer to execute and for the programmer to develop. It's hard because it's time based and data intensive. Loading a single photo image can take anywhere from 20 ms (with one screen refresh) to several seconds and the viewer won't mind. Similarly, there's no point scrolling text faster than the viewer can read, so the computer scrolls it slowly.

In contrast, video is a sequence of images, where the display is updated 25 (phase alternating lines—PAL) or 30 (National TV Standards Committee—NTSC) times per second to create the impression of motion. A computer must read each image from the file or camera, decompress it, and blit it to the display in less than 40 (or 33) ms. The image's size, compression technique, and image display rate determines the data rate. For example, digital video with DV compression has a transfer rate of 100 megabits per second (Mbps) and uses 250 Mbytes of storage every minute.

While the data rate for audio is much lower, its sample rate is considerably higher. Stereo audio is a pair of analog signals that vary in time to record sound. Audio for digital video is sampled every 20.83 microseconds (at a 48-KHz sample rate) to give a data rate of 192 kilobits per second (Kbps).

Thus, when capturing a movie in DV format, the computer hardware and software has to capture 4.66 Mbytes of video and 48,000 stereo audio samples every second. It must also write this data to a file, display it on the screen, play it through the speakers, and attend to other computational functions required by the operating system.

As a result of these demands on the computer, multimedia software must be tailored specifically to the operating system and processor to work at all, let alone work efficiently. Consequently, multimedia software is difficult to port from one system to another. A mature multimedia-programming environment is the result of many years of work designing, programming, and testing efficient data compression and retrieval algorithms.

### Language choice

Multimedia programming occurs at three levels: library, application, and media presentation. At the library level, programmers write the API and associated codecs. Many of the library routines are heavyweight components that interact with the operating system and hardware.

Programmers usually write them in an efficient, low-level language, such as C.

At the application level, programmers write applications in a high-level language, such as Java, using calls to the API. The applications include media editors, media presenters, and multimedia kiosks.

At the media presentation level, programmers write code to present media. They can program it in a high-level language but often they do it in a specialist media presentation language that's part of a media presentation package, such as Lingo in Macromedia Director.

Because our goal is to teach computer science students to program both at the application and the media presentation levels, we chose Java as the teaching language. We wanted the programming environment to support the learning of multimedia concepts. Java provides standard windowing and graphics APIs that enable rapid development of applications.

Having chosen Java, the next step was choosing a multimedia API. Several factors influenced our choice, including maturity of the multimedia engine, teaching resources, integration with Java, and platform support. Initially, we considered using Sun's Java Media Format,<sup>4</sup> but in 2000 it was immature and with an uncertain future. A quick look at the Web site where it's specified (<http://java.sun.com/products/javamedia/jmf/index.html>) indicated that development is slow.

More recently, we looked at Cocoa for Java.<sup>5</sup> Cocoa is an Objective C framework for developing Macintosh applications. Cocoa for Java is a wrapper class for the Objective C framework that isn't compatible with AWT or Swing. Support for Cocoa for Java is limited to WebObjects applications and Apple doesn't recommend it for general Java development.

While QuickTime for Java<sup>6</sup> sits on top of the QuickTime C library, it's more than a wrapper class. It repackages the library to make it both object oriented and compatible with the standard Java AWT and Swing libraries. Apple programmers put a lot of effort into solving the differences between the languages, shielding the programmer from many of the runtime issues. Development is supported in both Macintosh and Windows environments.

An advantage to the students learning QuickTime is that MPEG-4 draws heavily from QuickTime concepts. Also, a large set of code examples and a book<sup>7</sup> are available, which is helpful for students. The book is more a devel-

oper's reference than a textbook, but Apple is revising it to make it more like a textbook.

### Model of multimedia applications

QuickTime for Java is a complex API with no obvious unifying theme, so it wasn't easy to develop a coherent set of lectures. To aid in this process, and to help understand exactly what a multimedia engine does, we developed a model of multimedia from our experience in using QuickTime for Java. The model consists of three parts: the movie data structure, a media tree, and a data flow diagram.

The key to understanding QuickTime is the movie data structure. It's a software model of a movie derived from film. A movie contains a movie description and one or more tracks (see Figure 1). The movie description contains information about the movie, including its time base, play rate, and volume. Each track contains a track description and either references a media file—including edit points—or the actual media. The track description contains information about the track, including media type, data size, track format, and time base. The movie data structure is a tree of atoms, where an atom is a structure for holding descriptive data.

The medium referenced by or stored in a movie track can be one of three classes: still, space based, and time based (see Figure 2). Still media is composited onto a canvas and written to a display once. The display is only changed in response to a user action that changes the content of the composition, such as a change in camera focal length as it zooms into an image.

Space-based media displays one view of a 3D model or a virtual reality (VR) scene on the canvas. The system updates this view in response to user input. The user creates new views by changing the camera's position relative to the 3D model or panoramic image. Complex algorithms stitch 2D panoramic images into immersive 3D environments.

Time-based media continuously changes as the system presents a sequence of stills in sync with a computer-generated time base. We can break time-based media into two subclasses: visual and aural, based on the stimulation of human senses. Haptic (touch sense) and smell (scent generation) are only available as experimental outputs at present and you wouldn't normally find these in multimedia. Time-based media involves the regular output of images to a display and sounds to speakers. Most media players force the

### Movie Data Structure

```
Movie [ movie description + tracks [ track descriptions + references OR media ] ]
Movie [ movie atom [ track atoms [ media atoms ] ] ]
```

user to interact sequentially when presenting time-based media.

Event-based media changes in response to user input. The content of event-based media can be any of the three classes in Figure 2. In contrast to controlling a sequential player, user interaction is often a set of arbitrary events—for example, navigating a Web site. A significant challenge in designing multimedia applications is to enable nonlinear user interaction with time-based media. Mostly, we achieve this with event-based random selection from a set of time-based sequential presentations.

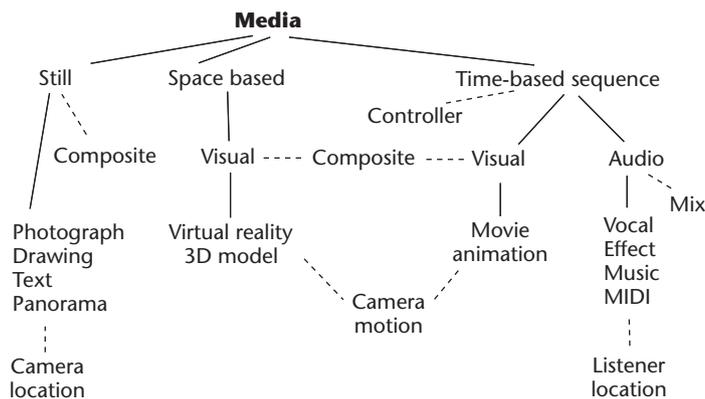
A data flow model (see Figure 3, next page) helps us understand the relationship between the seven processes found in multimedia applications: acquisition, representation, compression, composition, delivery, display, and human perception. We either acquire raw media with an electronic device (such as a camera) or an artist creates it using a software design tool (such as Adobe Photoshop).

This media is stored in files, usually in a compressed form. Every media type can be stored in several media formats, each with its own compression algorithms. Once we acquire the media, we can edit it to produce content for display. A video editor is a multimedia application that fits the data flow model in Figure 3.

After editing, a multimedia application integrates the content with other content and handles user interaction to achieve the desired presentation. Finally, we deliver the presentation using the chosen technology (for example, a

Figure 1. Movie data structure.

Figure 2. Tree-grouping media by class (solid lines) and showing available actions (dashed lines).



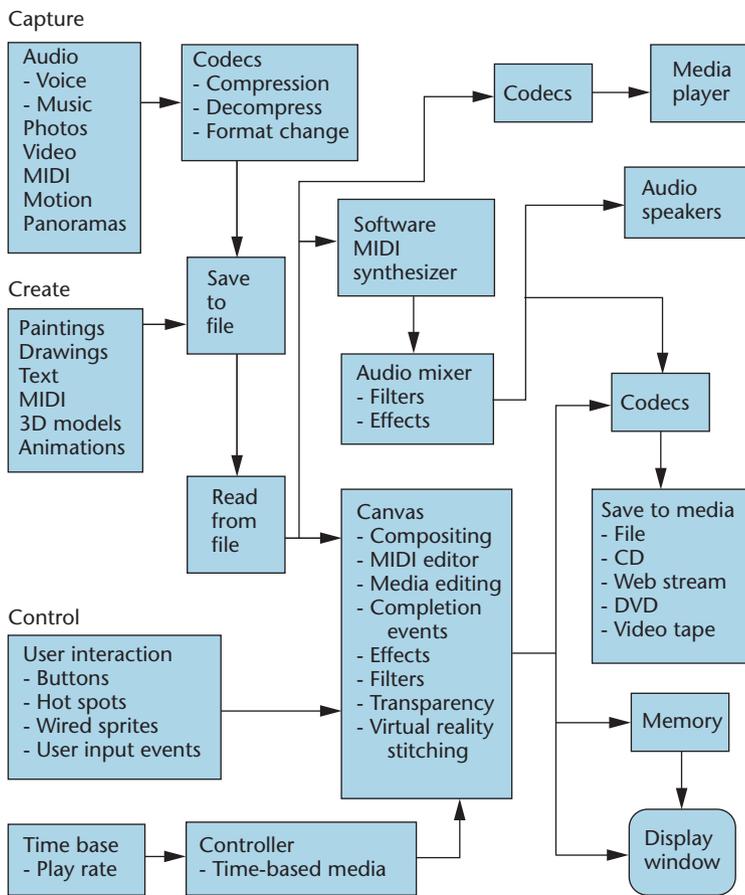


Figure 3. Data flow model of multimedia applications.

DVD). Each delivery technology has a physical medium for storage, transport, display, and interaction with the media. So each delivery technology has its own formats and each format has its own compression algorithms.

### Creative process

While the data flow model is helpful in that it shows the flow of data through the computer, it's not the complete picture. When you develop a multimedia application you must also consider the process that the creative artist will follow when using your application to create or edit media assets.

For example, each media type is edited in a different way. If a person edits visual material on a canvas, then it displays in a window for the editor to view the results. If someone edits audio material with a mixer, it plays through speakers for the editor to listen to the results.

Designing and creating multimedia is an interactive process. Artists may spend a lot of time writing down or sketching ideas. They

discuss and revise these ideas with other members of the creative team. When they start the creation process they have a good idea how to express the idea in their minds with the chosen medium. However, they may make many attempts before they feel satisfied with the result.

Consider the editing example again. Editing a video occurs in stages. Each stage is a finer edit. The first stage captures the clips into the computer. The editor filters through which takes to use for each scene. More potential content gets edited out at this stage than any other.

Then the editors link the clips together in a rough sequence. They review the rough edit to see whether the video will work artistically. Several editing stages follow to finely tune the video transitions, correct color imbalances, sweeten the audio, and add effects and titles. We can only understand and appreciate these creative processes by learning them and producing creative content.

Properly managing the media during the creative process has a significant impact on both the content creators' productivity and their ability to achieve the desired artistic outcome. For example, a video might contain hundreds of clips, with some linked in sequences and some composited together to form individual scenes. The artist must be able to access and group these easily and quickly.

Multimedia editors and composers typically include browsers so that artists can arrange the media in a way that suits their creation style. Often this arrangement bears little relationship to the directory and file structure of the disk on which they store the content.

### Subject content

Our desire to teach a programmer's view of multimedia governed how we combined the characteristics of multimedia with the model of multimedia and the functionality of QuickTime for Java. Our approach is to present the characteristics (integration, interactivity, hypermedia, immersion, and narrativity) as problems to be solved by the processes (acquisition, representation, compression, composition, delivery, display, and human perception) and implemented with calls to QuickTime for Java functions.

As a result, we devised a list of topics that cover the knowledge needed to understand and program multimedia applications. This knowledge includes

- media types—image, audio, MIDI, movie, text, and animation;
- media representation—data and file structure;
- creative process flow—how to turn an idea into a media asset and content creation;
- capturing external media—snapping images, grabbing sequences of images, and recording audio;
- media formats for each media type—format description, compression, conversion, and tradeoffs when choosing compressors for different applications;
- media players—loading and controlling the presentation of different media;
- editing media—clipping, sequencing, and filtering;
- mixing audio—balance, filters, effects, and level control;
- compositing visual media—layers, transparency, effects, and transitions;
- animation—cell based, sprites, timelines, and event sequences;
- creating immersive environments—panoramas and QuickTime Virtual Reality;
- designing multimedia applications—integrating media components with user interaction to support creative process flow and media management;
- designing user interaction—navigation, roll overs, hot spots, and dragging;
- synchronizing media presentation—controllers, time bases, completion events, and user input events;
- output of media—display of canvas, blitting, playing audio mix, and saving to files; and
- distributing media—DVD, formats, and compressors.

When presenting these topics, it's a challenge

to integrate the presentation of artistic and technical aspects into one subject. We took two approaches to integration: immersive and problem solving.

In the immersive approach, we immersed students in both aspects by getting them to create media (a video) and then write programs to dissect the media and recombine the media elements in a different way. In the problem-solving approach, we illustrated the creative concepts to show students how to solve the problems of media capture, storage, editing, and presentation.

### Learning outcomes

These topics represent the knowledge that a multimedia programmer should understand. However, a course should do more than impart understanding. It should change the way students act, think, and feel about the topic. We specified this desired change in behavior in the course objectives. Bloom<sup>8</sup> defined six classes of learning objectives: knowledge, comprehension, application, analysis, synthesis, and evaluation.

The knowledge outcome was to understand the material taught: QuickTime for Java and how to create multimedia content. The comprehension outcome was to create a digital video using Final Cut Pro and to write a multimedia application in QuickTime for Java. The application outcome was to abstract from the skills learned to understand how to develop multimedia.

The analysis outcome was to break a movie down into parts and understand the relationship between these parts both in the movie data structure and in the content. The synthesis outcome was to combine programming and content creation to produce a multimedia presentation. The evaluation outcome was to make judgments about the quality of the multimedia content.

To achieve these learning outcomes, we divided the subject into four sections: lectures on content creation, lectures on QuickTime for Java, laboratory tutorials on professional tools (such as Apple's Final Cut Pro, LiveType, Logic, and SoundTrack and Adobe's Photoshop and Illustrator) and assignments. Initially, we presented these in parallel.

However, we found it easier for the students to achieve the learning goals when we presented the lectures in blocks. We determined which lectures to present in each block by two criteria. First, we put the material relevant to the current assignment into the block that immediately preceded its due date. Second, the blocks built on one



*Figure 4. Creating a thumbnail index (right) for a movie (left).*

another to achieve a holistic view of the subject.

Assignment 1 (see the “Programming assignments” section) was to produce a video that’s 30 to 60 seconds long. The assignment was due in week 5, and the block was three weeks long with 2 hours of lecture and 1 hour of laboratory tutorials each week. The lecture topics included video basics, photography, videography, lighting, script writing, and storyboards. The laboratory tutorials covered video capture, nonlinear editing, and audio sweetening with Final Cut Pro.

Assignments 2 and 3 involved writing programs. We presented the lectures in a six-week block and covered programming in QuickTime for Java. The laboratory tutorials covered the programming environment, motion and effects in Final Cut Pro, and creating titles with LiveType. In week 6, the students presented their videos from assignment 1 to their laboratory class.

Assignment 4 was a group project to integrate the skills learned to produce a multimedia kiosk. A three-week block of lectures covered audio recording, mixing, storytelling, reading aloud, music, MIDI, color correction, communication, and QuickTime Virtual Reality. The laboratory tutorials covered creating music with SoundTrack, editing sound and MIDI files with Logic, and preparing graphics and images for video with Illustrator and Photoshop. The students presented their multimedia kiosks to the class in the last 2-hour lecture (week 13).

### **Programming assignments**

The most common example of a multimedia application is the player, such as Apple QuickTime Player, Real Player, and Windows

Media Player. But media players are just one example of a multimedia application. The QuickTime for Java API enables the development of a wide range of applications. We developed four types of applications in assignments: sequential media players, accessing movie internals, interactive media presenters, and projects.

In the sequential media player assignments, students created their own media players by providing control buttons on a Java window. They essentially modified the standard QuickTime media players, image presenters, and audio sequencers. An example of this involves reading a set of images from a file and making a slide show or a storyboard. The student can present the slide show on the canvas or write it to a movie file. This application supports editing slides (such as deletion or changing order) and playing transition effects between slides.

To learn about the internals of the movie data structure we used two types of assignments. In the first type the task is to read the movie- and track-description data from the movie and display it. The second type is to capture images from a movie and save these to a file for making a slide show or a storyboard, or to create a thumbnail index into the movie (see Figure 4).

Developing an interactive media player causes the student to think about parallel presentation of media and random user interaction. An example is to divide a movie into several smaller movies, display them onto the same canvas, and use mouse rollovers to select which movie to play. Alternatively, the student can make an interactive movie by compositing wired sprites onto a sprite track to create buttons for user navigation.

Multimedia projects include multimedia kiosks that present a variety of multimedia content in one window and games that change the media content in response to user interaction. Figure 5a shows a multimedia kiosk for a Kendo club. A film strip representing a menu scrolls to the left across the window. When the mouse rolls over an image in the filmstrip, a text panel appears to describe the associated movie. Double clicking on the image causes the filmstrip to become transparent and stop playing. Then the application brings the movie controller to the front (see Figure 5b).

The application will now play the selected movie. After viewing the movie, the user can return to the menu by clicking on the bar at the top. A white line cuts the window, like a slash from a light saber, and the window reverts to the menu display in Figure 5a.

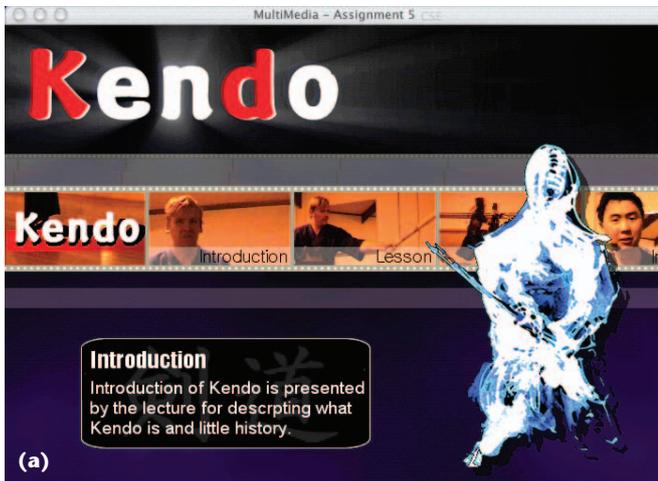


Figure 5. Multimedia kiosk for a Kendo club. (a) Each image in the scrolling film is a selectable menu item. (b) The movie controller comes to the front of the compositor in response to the user's selection from the menu.

As another example, in the game shown in Figure 6, the user must pilot the spacecraft around a course defined by pairs of rotating moons. When the craft touches a moon, it explodes with the appropriate visual and audio effects and returns to the start. The game includes audio messages, such as “prepare to launch” when the craft first takes off. It also has special effects, such as engine noise when the rockets are fired. The student can integrate the images, movies, sprites, speech, sound effects, and user interaction via the keyboard into one application using QuickTime for Java functions.



Figure 6. Space navigation game—fly the ship around the course defined by pairs of rotating moons.

### Student response

Many students commented that having a creative component in a computer science course is refreshing and that it stimulates their interest in multimedia. Most enjoy making their videos and are excited when they present them to others. The excitement and tension peaks when they present their final project to the whole class. Most feel proud of what they achieved.

To determine whether the students achieved our learning objectives, we surveyed the 2004 class during the last week of the course. Of the 76 students enrolled, 50 completed the survey. The survey consisted of a set of statements with a request to rate their level of agreement with each statement on a five-point scale: not at all, low, medium, high, and very high.

We give a subset of the survey relevant to this article in Table 1 (next page), in particular the statements that relate to Bloom's<sup>8</sup> taxonomy of learning objectives (statements 1 through 9). The

percentage of students who rated their level of agreement with each statement as either high or very high ranged from 58 percent for statement 11 to 94 percent for statement 3.

Statements 1 and 2 in Table 1 measure the students' perception of the amount of knowledge they learned. The third statement measures the student's level of comprehension, showing that 94 percent of students felt that they achieved the goal of understanding how to create media. The fourth statement measures the abstraction objective, indicating that a surprising 90 percent of students felt that their ability to create multimedia improved.

Statement 5 measures the analysis objective, showing that a high percentage of students believe that they sufficiently understand the relationship between content creation and program-

**Table 1. Survey results. Percentage of students indicating their level of agreement with each of the statements.**

Statement	Not at All (%)	Low (%)	Medium (%)	High (%)	Very High (%)
1. Learning QuickTime for Java has increased my understanding of multimedia programming.	—	16	14	42	28
2. Learning Final Cut Pro has increased my understanding of multimedia content creation.	—	2	10	46	40
3. This subject increased my understanding of how to create multimedia content.	—	2	4	54	40
4. This subject improved my ability to create multimedia content.	—	—	10	50	40
5. Studying programming and media creation in the one subject has helped me understand the relationship between multimedia content creation and multimedia programming.	—	4	20	54	22
6. In this subject, I have learned how to synthesize programming skills with creative skills to produce multimedia.	—	6	24	48	22
7. Including creative activities in a programming subject has given me a better understanding of multimedia programming.	—	4	20	52	24
8. Programming multimedia applications in QuickTime for Java has helped me develop creative skills.	4	10	32	42	12
9. This subject has increased my ability to evaluate the quality of multimedia.	—	—	28	48	24
10. As a result of doing this subject, I believe that I will be better able to relate to creative professionals.	—	—	38	40	22
11. Through doing this subject, my attitude toward creative people has become much more positive.	—	10	32	40	18
12. This subject is hard work.	—	6	26	46	22
13. I enjoyed this subject because it's fun.	—	—	12	50	38

ming to analyze multimedia. Statements 6, 7, and 8 all measure aspects of the synthesis objective. This is a pleasing result because it indicates that they believe that combining media creation and programming in one subject works for them.

Statement 9 measures the evaluation objective, indicating that students believe that they have learned the skills to evaluate the quality of multimedia. One of the goals of the subject is to equip computer science graduates to work with creative professionals. The high level of agreement with statements 10 and 11 indicate that we achieved this goal.

Students attempting this subject have a lot to learn. Those who have a weak background in Java commented that learning QuickTime for Java is difficult. Most have no experience with professional media creation tools, such as Final Cut Pro. As a result, they have to learn the processes involved in media creation at an abstract level as well as the use of specific tools at a concrete level.

Our multimedia laboratory contains 28 Macintosh G4 computers and a Macintosh server with 1 Tbyte of hard-drive storage. Many students have no experience with OS X as a development environment and have to learn it. Students are unaccustomed to dealing with the large files required to store video and can get frustrated at the time it takes to move Gigabytes of video files around a 100-Mbit network.

One concern we had when creating the class involved combining two paradigms (creative and programming) in one subject. We worried that we might overload the students either intellectually or with the amount of work. The students' response to statement 12 indicates that the subject is hard but not too hard. When asked whether they enjoyed the subject because it was creative, 92 percent agreed to a high or very high level, and 88 percent agreed that they enjoyed it because it is fun (statement 13).

### Future avenues

Students are excited by the opportunity to create something artistic and then write a program to present it. To capitalize on the motivation this produces, we have to integrate programming and creativity into multimedia subjects. A multidisciplinary approach with teachers from both the creative arts and computer science provides a role model of people from separate disciplines working together.

However, to provide a role model that integrates the two disciplines requires a cross-disciplinary approach with teachers who can work in both disciplines. Finding academic staff with relevant experience in both computer programming and media creation is the biggest challenge we face when developing new multimedia subjects. **MM**

### References

1. R. Packer and K. Jordan, *Multimedia—From Wagner to Virtual Reality*, W.W. Norton, 2001, p. xxxi.
2. R.S. Tannenbaum, *Theoretical Foundations of Multimedia*, W.H. Freeman, 1998, p. 4.
3. R.R. Bartle, *Designing Virtual Worlds*, New Riders Publishing, 2003.
4. R. Gordon and S. Talley, *Essential JMF: Java Media Framework*, Prentice Hall, 1999.

5. A. Hillegass, *Cocoa Programming for Mac OS X*, Addison Wesley, 2002.
6. C. Adamson, *QuickTime for Java, A Developer's Notebook*, O'Reilly, 2005.
7. T. Maremma and W. Stewart, *QuickTime for Java: A Programmer's Guide to Building Multimedia Applications with Java*, 2nd ed., Morgan Kaufmann, 2005.
8. B.S. Bloom, ed., *Taxonomy of Educational Objectives*, David McKay Co., 1956.



**Phillip John McKerrow** is an associate professor in computer science at the University of Wollongong, Australia. His research interests include perception of the environment with ultrasonic sensing and mobile robot navigation. McKerrow graduated from the University of New South Wales with a BE (Hons) in electrical engineering, and from the University of Wollongong with an ME in electrical engineering and a PhD in computer science.

Readers may contact Phillip John McKerrow at the School of Information Technology and Computer Science, Univ. of Wollongong, Wollongong, NSW 2522, Australia; phillip@uow.edu.au.

## Visit the IEEE Computer Society's all-new Software Engineering online resource



unbiased and trusted  
peer-reviewed  
in-depth and topical  
practical and timely  
free technical content

**Go online today**

**SEONLINE**   
SOFTWARE ENGINEERING ONLINE

[www.computer.org/seonline](http://www.computer.org/seonline)