The Journal of the Association for Computing Machinery at UIUC:

Banks of the Boneyard

Volume 17, Issue 4

What's inside?

MPEG Basics: *What's actually stored on that DVD?*

by Dan Sachs

All of you probably know that DVDs are the latest and greatest in video storage, allowing over 2 hours of video to be stored digitally on a disc that looks exactly the same as an ordinary CD. Most of you know that a DVD also compresses the video data to cram an entire movie into less than 5 GB.

You may not realize this, but video compression is the technology that enabled the digital storage of video. LaserDiscs, for example, though they use a laser reader and store a digital soundtrack exactly like those found on CDs are actually analog devices, and if you look carefully at a LaserDisc being played back you'll occasionally see noise - little flashes or dropouts - that cannot be corrected due to the analog recording of data. This is because most error correction techniques only work for digital data. Although you can use filters to smooth over analog flaws, there is no good way to correct, or even identify errors in an analog waveform.

DVD's, on the other hand, don't suffer from these analog flaws. This is because the digital data stream is recorded using extensive error correction coding, so that unless you obscure a significant area of the disc, the digital signal the reader sees is the same as the one that was written on the DVD. This also means that you can get away with treating DVD's somewhat more carelessly than LaserDisc, since the reader will be able to reconstruct the original signal around, for example, a random thumbprint.

Unfortunately, even the 4.7GB that can be stored on a single layer of a DVD pales in comparison to the massive size of an uncompressed video data stream. A DVD stores video at a resolution of 720x480. At this resolution, a single frame is about 1 megabyte; therefore, a 30 frame per second uncompressed video stream is about 30 megabytes per second. At this data rate, your DVD would hold about 5 minutes of video on a side. That rate doesn't include the multichannel audio, or other overhead

signaling. Since the DVD player can only read data off the disc at a rate of about 1.4 megabytes a second, there would be no way to get the data to the TV anyway. Clearly, some compression is required.

The compression used by the DVD encodes the video stream along with the audio data and other overhead to a variable bit rate that typically will be around 600 kilobytes per second. This represents a compression ratio of over 500 to 1. These compression ratios are achieved through MPEG-2 video compression, which relies on two major techniques to reduce the number of bits required to represent the video:

- Predictive interframe coding using motion estimation

continued on page 3

ACM@UIUC From the Chair

by Mark Ashton

If there is one thing we have learned in the brief history of the computer industry, it is that, despite the best efforts of those making money by maintaining the status quo, the state of the art changes and improves constantly. Where will the constant innovation in this industry someday lead? Well, as Arthur C. Clarke said, "Any sufficiently advanced technology is indistinguishable from magic." We are already seeing applications for computers that are essentially transparent to the user, and that operate like magic to the uninitiated. A technology is truly mature when users do not realize they are using it.

There are examples of computing applications today that work like magic. Many corporate users daily access and use mid and high-range enterprise servers, such as AS/400s, mainframes and web servers, without knowing it. Most drivers are unaware that their fuel injection is electronically controlled, and many cars today use computers to drive traction control, all-wheel drive, and even to automatically customize shift timings in an automatic transmission. All of this occurs without any knowledge or interference from the user.

Most such uses of "invisible computing" are in the realm of embedded systems. I believe, however, that it is only a matter of time before general purpose computing is just as invisible. Computers are the only devices capable of modeling their own reality, and performing something analogous to human thought. Why interact with such a powerful device with as clunky an interface as keyboard, mouse, and monitor? Today's interfaces require far more knowledge of the actual workings of the machine than is necessary. The average user is mainly concerned with running software packages, and that is all he or she should have to worry about, just as the average driver shouldn't have to worry about adjusting valve timings or tuning a suspension.

One of my favorite far-fetched examples of transparent general-purpose computing is author Neal Stephenson's idea of "mediaglyphic paper." It is a paper-thin computer and display, complete with wireless network access. The user interfaces with it by simply writing on it; natural handwriting recognition turns the input into text, which is interpreted with an AI into user commands. If this sounds farfetched, realize that natural handwriting recognition is already a reality. Remember how impossible the idea of a pocket-sized Personal Digital Assistant once seemed?

Although the form of the device may be different, within ten or twenty years I believe we will have new technology that will make today's PCs seems as clunky and ridiculous as a horse and buggy. Despite the vaunted user-friendliness of the GUI, you can bet it won't be long before someone creates a computer that requires no knowledge of computing to use. The PC will give way to the IA - the Intelligent Assistant. Although some self-proclaimed purists might bemoan a world where one does not need intimate knowledge of computer science to use a computer, I believe the change will be welcome. There will always be room for designers, tinkerers, and hobbyists. The rest of the world needs something with which they can get work done.

The Banks of the Boneyard is a student publication of the UIUC Chapter of the ACM, the First Society in Computing, and does not necessarily represent, in whole or in part, the views of the University of Illinois, the Department of Computer Science, or the International ACM. Partially paid for by SORF and the Department of Computer Science.
Copyright 1998, ACM@UIUC

Editing, Layout, and Design:
Rene Hendrix and Anne Nowinski

The Association for Computing Machinery at UIUC
1225 Digital Computer Lab, MC-258
1304 W. Springfield Avenue
Urbana, IL 61801
(217) 333-5828

For more information, email
acm@uiuc.edu
or visit our homepage at
www.acm.uiuc.edu

The newsletter staff encourages ACM members to submit letters to the editor, articles, photographs and artwork for future issues of *The Banks of the Boneyard.* If you would like to contribute or participate in its creation, please email boneyard@acm.uiuc.edu.

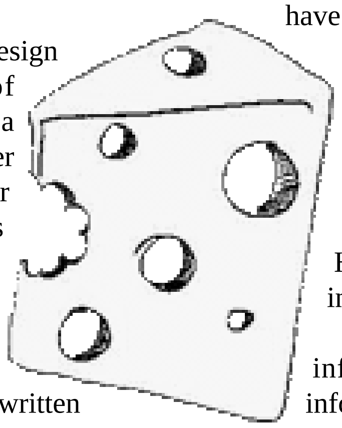
SIG-BIO

Say Cheese!

by Mary Lee

The quest for facial recognition continues as the EOH project of the Special Interest Group for Biocomputing, a pool of people dedicated to applications of computers to the biological sciences. In this case, drafts of the program designs have been made and edited to the first one published last month.

Our main focus of this project is to design and implement a program or set of programs that will use a camera and a video capture card to take images of a user attempting login and then authenticate or reject that person. Methodologies suggested to accomplish this include pixel comparisons or vector geometries verified with those in a database, hash table, or text file.



This program originally designed to written to be objective will probably be broken up into one to add users and another to authenticate. The interface will need camera feedback so that the user can make sure he/she is centered in the photos that will be taken. The stream of data from the video camera sent to the capture card will need the first and last frames removed.

Next, the image will be read and the final decided algorithm will be run to make sure the user is valid on the system. That accomplished, the desired platform will have to be chosen from amongst windows, Solaris, Linux, or a portable PC104.

Additional problems that must be worked around include the size of the eye pupils due to light levels, the amount of light available, individual tolerance levels, blinking, and the emphasis of the eye versus the eye socket. However these will be avoided or taken care of in due time.

If you'd like to help or find out more information, feel free to stop by during our informal meetings at 6:30 PM Tuesday in 1225 DCL. Otherwise you can mail <sigbio@acm.uiuc.edu> or visit http://www.acm.uiuc.edu/sigbio for some outdated information.

```
** LU-9000 version ALPHA - (c)1998 SigSoft@uiuc
** booting....
** Checking CPU.... ok
** Checking RAM.... ok
** Quick integrity check.... passed
** Full integrity check.... FAILED!

*** STATUS DUMP ***
Conceptual Design: done
Game Details: partial
Program Design: partial
Code Completion: minimal

Recommended Fix: Contact developers Juan Custer,
Moshen Chan, Erik Gilling, Jason Govig, Mike
Kenny, Ron Labok, Jason Petrone, Pete Taylor or
visit SigSoft website
http://www.acm.uiuc.edu/sigsoft.

Meeting Time: Wednesdays at 6:00 PM.

*** LU-9000 HALTED ***
```

DVD *continued from page 1*

• DCT (discrete cosine transform) intraframe compression

The first technique is a lot simpler than it sounds. Since in video, much of the change from one frame to the next is simply an object on the frame moving from one place to another (for instance, a car running from left to right across the frame). An image of the object exists in previous or future frames of the video. In this case, instead of sending a new description of the object, we can instead send a tag that says, "pick up the car from the previous frame, shove it to the left ten pixels, and paste it onto this frame." In this way, we can avoid sending another description of the car. Instead, we send only a few bits to describe the location of the car in the previous frame.

continued on page 4

SigUNIX

by Tony Sintes

This month SigUNIX elected Frank Tobin as the new SigUNIX co-chair. Frank will bravely lead SigUNIX into ’99 as the full SigUNIX chairman. While my departure is a significant loss for SigUNIX, it is a victory for mankind. For you see, I, Tony Sintes, am graduating. No longer will the world be denied my creative genius. No longer will it be captive to Banks’ articles, CS classes, and SigUNIX meetings. Oh no, no more will the world be deprived of my greatness, my vision. For I am FREE of the shackles known as UIUC! And you, my dear reader, are not.

At this point I fear that you may be wiping the tears of sorrow from off your cheek. But, dearest reader, do not fret, read on. SigUNIX continues its work on the monitor program. Currently we are working on the marriage between the background plug-ins, the display plug-ins, and the main mother program. Of course, this work is often interrupted with discussion about Unix, the evil Microsoft, and the computing world in general.

So, whether you know Unix inside and out or just want to learn (or maybe just rant about certain software giants), stop on by the next SigUNIX meeting. We meet on Thursday nights at 8 PM in the ACM office. We also hold nearly weekly programming meetings on Sundays. For more information see uiuc.org.acm.sigunix or email me at sintes@uiuc.edu and I’ll add you to the mailing list.

DVD *continued from page 3*

However, without a baseline image to bring the pixels from, motion estimation is useless. Therefore, the MPEG-2 standard specifies that one in every fifteen frames must be stored without references to past or future frames. It also specifies other frames must be stored without references to future frames. These pictures form a baseline from which the remaining frames are predicted. Without these baseline frames, special features such as fast forward and rewind would be impossible. They also provide an occasional “clean slate” to prevent the accumulation of prediction errors as time goes on. MPEG-2’s requirement of a reference frame every 15 frames means that we can randomly access points on the DVD with a resolution of a half second, which is enough for most purposes.

Motion encoding is by far the most computationally expensive procedure used in the DVD encoding processes. However, it is simple to use the motion vectors and therefore is cheap to decode. In the MPEG-2 video compression specification, a motion vector is generated for each 16x16 pixel block in the frame. Ideally, this is done by comparing the block to each 16x16 block in the previous and next reference frame. However, in practice this is impractical and the search is restricted to a small window (typically, the blocks searched are offset by no more than 30 to 60 pixels in any direction from the block we’re encoding). Whichever block in that search window is “least different” from the block being encoded is picked, and the motion vector for that block is encoded into the bit stream. Most encoders use optimizations that allow them to pick the block that is “least different” from the one being encoded without searching all the possibilities, reducing the computational load considerably. However, even with these optimizations, the motion estimation represents over 50% of the total computational burden of the encoder.

Motion estimation works well in most of the time, but there are a few cases in which it fails. Often, these can leave noticeable artifacts on the screen. For instance, consider scrolling text (such as credits) over a complex, low-contrast background. In this case, we have the background (which does not move) and the text (which is moving) in the same 16x16 block. Because the background is low-contrast and the text is high-contrast, the motion encoder will pull the text in from the previous image up to its new location on the screen. However, the background will come along with the text. This results in visible distortion as the background is “carried along” with the scrolling text. Although intraframe encoding is used to combat this effect, eliminating these artifacts will often exceed the “bit budget” available to encoder. In other words, the maximum data rate of the DVD will limit the correction that can be applied. In this case, the “crawling background” effect will be clearly visible.

Intraframe coding simply means coding using only the frame that is currently being encoded. Intraframe coding is used to transmit the “reference” frames; it is also used to code the “error image” that represents the difference between the frame predicted by the motion vectors and the actual frame being transmitted. MPEG-2 uses the DCT (discrete cosine transform) to encode this data.

The DCT is essentially a technique for converting a sequence of numbers into a representation of its frequency content. It is closely related to the Fourier transform. The implementation DCT coding used by MPEG-2 is very similar to the JPEG image compression standard, which is used to compress still images. In fact, the “reference” frames are compressed in essentially the same way as

continued on page 6

ACM WORKSHOPS WRAPUP

by Steve Mycynek

This past semester, ACM sponsored two promotional workshops—"Linux Install-Fest" and "Building Better Web Pages."

The idea behind these workshops was to reach out to the non-ACM and non computer-science communities at U of I to show them what we are all about. Now that the semester is nearly over, we would like your feedback. Did you gain anything from our efforts? Have any of your non-CS peers recently jumped up and dramatically said, "Gee—I thought those ACM people were a bunch of wierdos, but I guess they're pretty cool and have a lot to offer —and they have good taste in soft-drinks, too!" Maybe, with finals approaching, the hope that people are exclaiming anything with glee this time of the year is a stretch, but we hope that anyone who attended can still answer "yes" to the first question.

If you have any comments, questions, or requests for new workshops you would like to see, please email acm@uiuc.edu with your ideas. We would like to make next semester even better—with more opportunities for people to learn about ACM and pick up practical skills at the same time.

SIGOPS

David Wentzlaff

SigOps will be designing a distributed windowing system as their project for this year's Engineering Open House. This cool windowing system will not only allow for remote display, but also migration of the output from one display to another—even across operating systems — without restarting the process. Currently no commercial product does this, and even the X windowing system requires the application to be shutdown to move displays. So come learn about operating systems of all shapes and sizes, and try your hand at writing one with our Roll Your Own Operating System tutorial. SigOps meets Tuesdays at 7 PM.

It WAS Real

by Steve Behling

One of the biggest trends of the early 1990s was CD-ROM-based "interactive multimedia," and some of the earliest attempts at popularizing the idea were from enterprising consumer electronics firms. NEC released a CD drive for its "Turbo Grafx-16," known as the "PC Engine" in Japan. Sega, at the time, was developing the "Sega CD," known in Japan as the "Mega CD." Commodore's R&D department devised the "CD-TV" as a dedicated set-top multimedia box, competing head-to-head with Philips' forthcoming "CD-I."

Conspicuously absent in these early years were Nintendo and Sony. The two had been working in tandem since the earliest days of the Super Famicom/Super NES, yet most people considered this project an exercise in vaporware and hype. In fact, nothing could be further from the truth.

In terms of hardware, the project was grandiose. Nintendo would market the unnamed CD drive and the Super NES as individual components while Sony would market a combination system dubbed "Playstation." At the accessory's heart was a double-speed CD drive (540 MByte maximum capacity at formatting) and, initially, a 16-bit coprocessor with an extra megabyte of RAM for the host system. By August of 1992, these specs had been upgraded to a general-purpose 32-bit RISC coprocessor with one and a half extra megabytes of memory. The discs came in custom, double-door caddies that could also save data to a SRAM chip much like a cartridge.

With abilities including simple 3D graphic effects, extra sound channels, and full screen, full-motion video, hardware certainly wasn't an issue in the accessory's demise. Corporate politics and software were. Sony demanded extra licensing privileges and royalties from all titles developed for the unit. Nintendo balked, but was in an awkward position since Sony also supplied the Super NES' sound chips and development tools. To take their revenge, Nintendo contracted Philips and adopted their CD-I's disc format—"CD-XA" or "Extended Architecture." This was a good move in any case since XA, which allowed interleaved data, could have been potentially more efficient than Sony's format. In the end Sony backed down and several issues were settled. Sony continued to develop the CD accessory, but it shelved its Playstation concept. Sony would receive software royalties only for non-game software while Nintendo controlled game royalties and ultimately directed all licensing. Philips received rights to use Nintendo characters in CD-I titles (e.g. Hotel Mario).

By this time, however, the market performance of the multimedia machines was less than stunning. Commodore was en route to disaster, finally closing its doors after the failures of CD-TV and AmigaCD32.

continued on page 8

SigArch

Jason Gallicchio

SICArch is making a large LED display. After months of heated debate, the parts were finally ordered. We've decided to make a 21x75 LED array interfaced through a parallel port to a PC. The PC will be running our custom SignServer software that will take e-mails or animations drawn through our web-based JAVA applet, translate the data into a format downloadable to the sign's on-board RAM, and finally communicate through the parallel port to get the animation onto the sign.

Once we finish the basics, we can easily go on to implement algorithms in the on-board programmable logic hardware. The sign could dynamically compute square roots, calculate pi, or play Conway's Game of Life while it's waiting for messages. The possibilities are truly endless. Once we get the parts and have more time, the sign will be ours!

SigBiz

by Nathan Lau

As many of you might know, SigBiz has been revived once again. As a new member and chair of SigBiz, I have found many interesting topics and many new ideas that may be implemented in this newborn sig. Of course, it does not help that in the first two meetings for SigBiz, there have been zero people attending, not including myself. However, it has been my pleasure to have been able to meet with several different people including some past ACM SigBiz members and new ones as well. In those meetings, the idea of starting some seminars in the topics of entrepreneurship and intellectual property were heavily discussed. Hopefully, SigBiz will begin to see some activity in the areas of rounding up some speakers for next semester in those specific topics mentioned earlier. Thanks for all your interests in SigBiz.

The BSDaemons

by Sidney Cammeresi

The BSDaemons is ACM's newest SIG. We meet at 2 PM on Saturdays and discuss generic BSD-related issues, although most of us run OpenBSD. If you use FreeBSD, NetBSD, or OpenBSD, have questions, etc. please come.

DVD *continued from page 4*

JPEG would compress them. In this implementation, the image to be encoded is split up into 8x8 blocks. The DCT is then applied to these blocks. This pushes most of the actual information contained in the block toward the upper-left corner (which encodes the low-frequency terms); the right and bottom parts of the block contain mostly numbers near zero. This block is then quantized, which approximates each entry in the block by a nearby number that can be encoded with fewer bits. This pushes the numbers near zero to zero. The block is then encoded, with run-length compression on the zeros. Due to the properties of the DCT, when this encoded block is decompressed by computing the inverse DCT on the quantized block, recovering an approximation to the original block, it will not be "too different" from the original block.

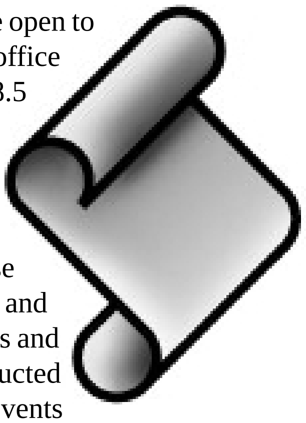
Because some scenes can be predicted more accurately than others, the actual number of bits required to represent the DCT-encoded difference images can vary significantly. If little motion is present, and the motion that is present can be predicted accurately from the past or future frames, relatively few bits are required to encode the frame. However, in a frame that contains much information that cannot be found in past or future frames, the amount of data required to describe the difference image can be large. The variable rate encoding allows the simple scenes to take up less space on the DVD, allowing a longer movie to be encoded without using a second layer (which increases production costs), or requiring the user to flip the disc. Using these techniques, a single-layer, 4.7GB DVD can store an entire 2-hour feature film.

The properties of the DCT encoding can exasperate the limitations of motion estimation. Considering again the case of scrolling text over a complex background, we see that the difference image to be encoded will not contain the text, which will be correctly predicted from past or future frames, but the area occupied by the text will be surrounded by the difference between the background "brought up" with the text and the background associated with the new area. Therefore, there will be a sharp contrast between the correctly predicted text and the incorrectly predicted background. This sharp contrast translates into high frequencies, which ends up in the bottom and right sides of the transformed block. This runs counter

continued on page 10

macWARRIORS

by Rick Roe



MacWarriors is ACM's special interest group for Mac OS users and developers; we're open to all members of the UIUC community and meet every Saturday at 3 PM in the ACM office (1225 DCL). For the past month and a half, we've been focusing on the new Mac OS 8.5 upgrade and its much-improved AppleScript technology in a series of meetings and workshops. Missed out on the workshops? Fear not! Read on, and I'll show you what the big deal's all about... and a few tricks for those who already know the stuff.

The Mac OS uses a system of high-level events—called AppleEvents—to allow running applications to communicate among each other (even on separate machines). These events can reference various objects used by the application and contain both primitive and complex types of data. Like other parts of the OS's resource-based operation, AppleEvents and their data types are represented by four-character codes. Applications which are constructed according to the AppleEvent Object Model can be fully controlled by events, and can use events to report their actions to other parts of the system; other applications can also be controlled via events, but not as completely.

To let users access this functionality, Apple created the Open Scripting Architecture, a means for translating AppleEvent codes into intelligible language-like forms, and AppleScript, a scripting language based on the OSA.

So, what's all this good for? AppleScript lets you automate your Mac.

Yeah, Unix users have been doing it for years with their shell-scripts and awks and pythons and various other creatures, and Windows types could kinda do it with DOS command-line programs and batch files, but AppleScript does it on your Mac... and do a number of things that Unix and batch scripting can't.

Here's an example you can write in Script Editor (which should have come with your Mac OS install... look in Apple Extras):

```
on open (fileList)
	set archiveLoc to (choose folder)
	checkFolder(archiveLoc)
	tell application "Finder"
		set labelList to name of every label of Finder preferences
		repeat with theFile in fileList
			move theFile to folder (item (label index of theFile) ¬
				of labelList) of archiveLoc
		end repeat
	end tell
end open

on checkFolder(theFolder)
	tell application "Finder"
		set labelList to name of every label of Finder preferences
		repeat with i from 1 to (number of items in labelList)
			set theLabel to item i of labelList
			if not (exists folder theLabel of theFolder) then
				make new folder at theFolder with properties ¬
					{name:theLabel, label index:i}
				set view of window of theFolder to modification date
			end if
		end repeat
	end tell
end checkFolder
```

Save that as an application and drop some icons onto it, and they will be sorted into different folders by label. Pretty keen, huh? It's just the beginning of what AppleScript can do: for example, you could expand on this script to separate files by type, then send text files to BBEdit to have their spelling and HTML checked. Or you could use one of Mac OS 8.5's Folder Actions handlers to do the same thing whenever you add files to a folder, making sure that

continued on page 9

SigVR

by Ray Kaplan

With the Engineering Open House coming up in a few months, SigVR, the Special Interest Group for Virtual Reality, has been working on its EOH project, a 3D Integrated Development Environment. The 3D IDE is being created with VRML, the Virtual Reality Modeling Language, and Platinum Technologies WorldView for Developers Active X control. We currently have identified the major structures in the C programming language, and have decided how to represent them in the IDE. Support for C++ is planned right now.

SigVR is also creating a VRML website. This will be the VRML equivalent to our regular web page, as well as a showcase of different aspects of VRML. If you are interested in either of these projects or anything else VRML or virtual reality related take a look at our website, http://www.acm.uiuc.edu/sigvr, email sigvr@uiuc.edu, or come to one of our meetings held every Monday at 7:00 PM in 1225 DCL, the ACM office.

SigGraph

by Brian T. Klamik

SigGraph is diligently working on our EOH project. The project is automatically built every night at midnight and the sources and binaries are zipped up and placed on an ftp server. The project is still a true physics based hovercraft game where racing and battle are the primary tasks. Our meetings have become a place to resolve component issues. Also, Sounds and Visions is approaching. This is where SigGraph writes graphics code to run at a concert against music. SigMusic should have obtained all the music pieces by now, so we'll listen to them soon.

The Costly PC Upgrade Cycle

by Brian T. Klamik

Face it, PC gamers are the majority of the people who demand high quality PC components. They always need better, faster, and more stuff to power their gaming experience. Your average PC user can easily settle for a sub $1000 PC to run Office and AOL or the like. For those that don't try to keep up with the gaming market, here's the typical experience:

First, one buys a PC that is close to top of the line at the current time.

It's packed with the latest video and audio boards as well as other speedy subsystems. When shopping around for games, one need not even look at the system requirements as not even the recommended specs come close to touching the newly purchased PC. Then after only a year, a couple hot new games are released which put the PC between the required and recommended specs. More and more games are released, pushing the PC's performance level dangerously close to the required spec. Upgrading the audio and video boards might boost the lifetime of the PC to 2 to 4 years. Eventually, a new PC is purchased to keep up with the new game releases. However, all the old games usually play just as they did before. The apparent benefit of the new PC is just the ability to play the new games. Personally, I like to think that I'll also be able to play my old games better or faster to justify the massive cost. Without this benefit, gamers on a budget will most likely turn to console systems which last between 1 and 2 times as long as a PC with upgrading, but cost one tenth the price.

The PC upgrade cycle exists because software companies justifiably target a minimum PC platform for which their software will run. This typical software

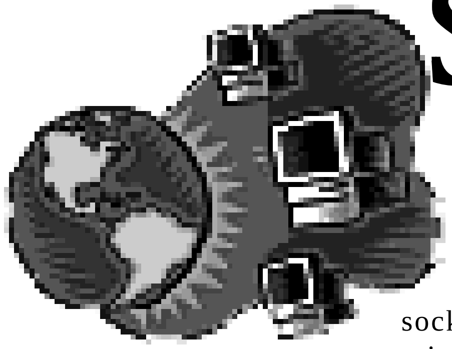
continued on page 9

It WAS Real *continued from page 5*

The 16-bit CD-I went nowhere. Pioneer's thousand-dollar LaserActive never took off, either. Many early adopters of the $700 3DO Interactive Multiplayer were disappointed by the early titles. One of the better performing systems, the Sega CD, managed to attain a loyal, but small following—some figures estimated it as one twentieth the size of the 8-bit NES market. The key issue, besides cost, was software quality. Low-budget video on top of standard gameplay just didn't impress the target audience. As Nintendo saw such events unfold, it shied away from releasing the finished CD accessory.

The accessory's unofficial death knell was the August 23, 1993, signing of the "Project Reality" contract with Silicon Graphics, from which the Nintendo64 was born. Shortly thereafter, Sony resurrected the Playstation, but completely revised it. Today's Playstation is, in fact, NOT equivalent to the early one.

Even the best hardware is nothing without good software to run on it, and as consumer technology spirals into the stratosphere, the early multimedia systems should serve as a reminder that software quality should match the pace of the industry.



SigNet

by Jason R. Govig

SigNet has had a great year. We learned networking basics and sockets programming using Java and C++. Our current project is a UNIX socket API that will make sockets programming in C++ as easy as Java. We will also be looking at other topics such as parallel processing. We are also looking for a new chair. You really don't need to know a lot about networking, just the ability to find people and other resources, as well as manage projects and activities. If you are interested, email signet@uiuc.edu.

Voodoo City, our web-based, Java-powered gaming site, has been looking at some redesign possibilities. There are some interesting ideas and new directions that Voodoo City may take this year. If you have design ideas, like writing HTML or CGI programs, or want to create a new game, email signet@uiuc.edu or come to a meeting.

SigNet meets every Monday at 7:30 PM in 1102 DCL and Voodoo City meets after that around 8:30 PM. For more information, email signet@uiuc.edu or visit or web page at http://www.acm.uiuc.edu/signet.

Upgrade continued from page 8

architecture could be changed to yield a more direct relationship between cost of system versus quality of PC experience and extend the life of games and PC systems.

The main feature of new games that is needed is the idea of a speed versus realism type setting. Many current games provide such an idea by allowing users to turn on and off rendering options and choosing a rendering depth. Choosing a resolution which to run in is another popular idea. Too many games, however, use resolution as the only means to trade off speed for better quality. What do people do who want to use TV tuners which max out at 800x600 when they want to reclaim some wasted processing time for more realism?

The best game I've seen with this theme was Nascar racing. It had

continued on page 11

MacWarriors continued from page 7

image and link tags in any HTML files it contains are correct.

In the new Mac OS 8.5, there are a couple of new features that can only be configured by AppleScript. The new Application Switcher palette is one of them. Here's a short script that sets it up in a useful format:

```
tell application "Application Switcher"
       set oldSettings to properties of palette
       set visible of palette to true
       set icon size of palette to small
       set names visible of palette to false
       set button ordering of palette to alphabetical
       set anchor point of palette to upper left
       set frame visible of palette to false
       set position of palette to upper right
       tell me to display dialog "Keep these settings?"
       if button returned of the result is "Cancel" then
              set properties of palette to oldSettings
       end if
end tell
```

Having fun yet? There's plenty more. You can check out online versions of our workshops at http://www.acm.uiuc.edu/macwarriors/applescript/, and there are many more scripting resources to be found at these websites:

- http://www.apple.com/applescript/
- http://developer.apple.com/techpubs/mac/AppleScriptLang/AppleScriptLang-2.html
- http://developer.apple.com/techpubs/mac/scriptingadditions/ScriptAdditions-2.html
- http://developer.apple.com/techpubs/mac/AppleScriptFind/AppleScriptFind-2.html
- http://www.scriptweb.com/

DVD *continued from page 6*

to the JPEG encoder's expectation that most of the energy will be in the upper-right corner, and tends to introduce large errors in the output. These errors show up as "ringing," which appears as lines of light and dark surrounding the sharp transitions. Since the DCT-based coder cannot effectively compress this type of data, combating this ringing requires raising the data rate significantly. This often demands a data rate that exceeds the rate at which data is actually read off the DVD. For an example of what these ringing artifacts look like, take a look at the images on the web page http://www.netswitcher.com/secure_web.htm

Despite the problems I've described, the compression used by DVD players is highly effective, and allows huge video compression ratios with minimal loss. It is often difficult to see distortion even in pathological cases like the one I've described above. DVDs have a picture quality that rivals LaserDisc, and can be made cheaply using the same processes that are used to press CDs. Players are also considerably smaller and less expensive than LaserDisc players, and the discs can hold an entire feature length movie on one side. Often three disc sides are required to store a movie on a LaserDisc. The DVD standard also has features that Hollywood likes, such as copy protection and region coding.

An interesting article covering many of the technical details of the DVD standard is available at the following web site: http://www.icdia.org/dvdfaq02.html Although this article is old and somewhat outdated, the information it contains is still valid, and it also contains several links to sites with other relevant information about the DVD and the MPEG standards.

SigDave

by Nick Michels

SigDave is the special interest group dedicated to find a new level of boredom. Each week an elite group of special agents search to reach our goal, whatever it may be. In the past few weeks we have decided to play with distributed processing in some form or another. Although several have fallen on our mission, we will proceed onward. The special 'Dave' agents are encouraged to bring any other tasks to the collective, for assimilation into the soon to be DaveNet online database and resource center.

If you wish to join the growing Dave collective stop on by the ACM office at 8 PM on Wednesday...for there is Dave in all of us.

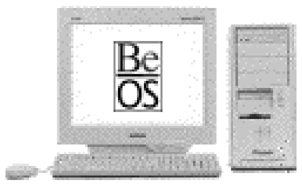
SigArt

by Jon Galownia

Under the fearless leadership of Misha Voloshin, SIGArt is building a mechanical arm whose sole goal in life is to recognize, pick up, and deposit objects into a container. The arm will replicate the human arm as closely as possible. Object recognition will take place via a k-means algorithm through a camera interface, which will be located atop the arm. The brain of the robot will be composed mostly of genetic algorithms focused on planning and learning goals. Planning skills will have to be demonstrated by the robot so that it doesn't knock over or push away its goal. Just for fun, we will be putting a large kill switch within the robot's reach, so that it can learn not to turn itself off. Construction plans are mostly completed, and work on coding has also begun.

To find out more about the project, visit our newly revamped website, http://www.acm.uiuc.edu/sigart. As they are completed, you will be able to see plans for the robot, and learn more about the algorithms being used.

BUG



by Vikram Kulkarni

We at the Be Users Group do all things Be. The BeOS is a new operating system aimed at the "high bandwidth" user. It runs on x86 and PPC hardware. Visit the Be web page at http://www.be.com to learn more about the BeOS.

This month we will be continuing our Be Programming workshops. We plan to cover basic multi-threaded programming, and basic GUI programming. We are also going to start working on our EOH project. We will be writing realtime mapping software using a GPS. BUG meets every Wednesday at 7 PM in 1225 DCL. Visit our web page at http://www.acm.uiuc.edu/bug/ or email bug@acm.uiuc.edu for more information.

WinDevils

by Ibrahim Merchant

WinDevils is still working on the 3D Drunk Driving Simulator for EOH. If worse comes to worse, we will at least have a Windows-based tic-tac-toe game. The members of WinDevils are also gearing up to have more Windows programming workshops for next semester. So stay tuned!

Improve Your Web Page in Six Easy Steps

by Valerie Franek

You've learned HTML, you've even experimented a bit with graphics in PhotoShop, and you finally have a web page of your own. It's not perfect and far from complete, so you wonder how you can improve what you have. By paying attention to a few elements of your web page, you can do just that:

(1) Layout - When you created your page, you probably made it to fit the monitor you were viewing it on. But what about people who don't have monitors as large as yours or with as high a resolution? Not everyone will know or want to scroll over, so it's often a good idea to create a page based on the 640x480 resolution. This way someone viewing your site will be able to view your entire page in one glance, no matter what their resolution.

Another common problem with layout is the length of one's pages. Most people will not look beyond the first or second Page Down, so it's a good idea to break a long page up into several shorter pages. You can split the page up by subject and link to each new page. This way there's a greater chance of people continuing to browse your page.

(2) Navigation - If a person viewing your site can't figure out how to maneuver through your pages, then they probably won't look at it long. Making sure they can find what information they're interested in and view those pages can be done with a logical navigation system. By using a consistent menu of list throughout your pages, you can provide such a system. Graphical buttons are a common solution, but it's always a good idea to include text links as an alternative.

You can help navigation by creating separate directories for various related pages when organizing the files of your web pages, . By naming the base page of each directory index.htm or index.html this can help you when linking pages and users when browsing your page. If they want to access a certain directory of your page

continued on page 12

Uppgrade continued from page 9

sliders which allowed the user to ensure a certain number of frames per second by shutting off detail levels. There was also a manual override for each detail feature. This idea is directly to the point. Users require a minimum of 15 frames per second to experience animation. However, each person has their own preference of whether 60 frames per second is worth more than realistic lens flare. This idea needs to be implemented more drastically. Many games have such a small range of performance capabilities that users end up with everything on or off depending on the hardware purchased. Game programmers need to expand their game's range of performance. When starting a project, they should gauge the average system currently as the slowest supported. Then, they should use Moore's Law to approximate the level of performance for average systems at the time of release. This should be the target platform. Then, double or triple that amount of time, and use Moore's Law to estimate the highest performance setting supported. A game should be able to bring the current systems to a crawl and still be able to work on systems a year old. That is the desired goal.

The real problem is how to achieve this broad spectrum of results. Again, supporting a full range of resolutions and color depths provides a good primary choice of performance level. Current level-of-detail techniques in games can be extended to provide higher detailed meshes and textures. The game could ship with very high detailed textures and complex meshes. When the game starts, the performance level is then determined or chosen and the correct texture resolution and mesh are used as base level. Current research by Hugues Hoppe provides a promising progressive mesh technology. It allows a mesh to be represented in the same amount of space, but described in such a way as to specify a wide range of detail levels. It was originally designed to allow progressive loading of meshes in a networked environment, much like progressive gifs. Only one model would ship with the game (highly detailed) and then the game could generate the actual detail level. It will remain to be seen whether or not technology works well with textures. Textures can easily be filtered down to resolutions which fit into texture memory.

Another technology which might aid this theme is the Fahrenheit Scene Graph API, which is yet to be released. This API is designed with extensibility as a primary goal. It enforces a distinct separation of scene graph data and the functions which use that data to display something on the screen. These functions can be replaced by functions provided by a hardware vendor, allowing unpredictable new hardware features to automatically be used. However, this requires hardware vendors to probably spend more on driver developers and decreases stability. It remains to be seen whether or not game developers will use this slower API to gain long term benefits. Such features won't boost product's initial profits greatly, as the increase in sales opportunity will come from opening the market to people with low end systems. After a while, however, sales might not drop off as sharply as usual. Will scalability become a priority in games?

Better Web pages in 6 Steps

continued from page 11

and know your Web address, they don't necessarily need to remember the exact file name.

(3) Graphics - Perhaps your site already incorporates graphics into its appearance. But it's often suggested to consider what type of file you're using as well as the sizes of these files.

First, web graphics are either JPEG's (Joint Photographic Experts Group) or GIF's (Graphics Interchange Format). JPEG's are typically used for higher quality images, like photographs. GIF's are used for animations and graphics with fewer colors (256 or less).

When it comes to file sizes, it's a good idea to keep files as small as possible, so they will load as quickly as possible. By using fewer colors in your graphics you can achieve smaller file sizes. This is often done by limiting the color palette used for making these graphics, in programs like PhotoShop or PaintShop Pro.

(4) Aesthetics - Graphics can definitely add a nice touch to a page, but it's a good idea to consider the number and intensity of them, as well as how they're used. Various animated graphics may be nice on their own, but when included on a page next to several others, a page can look too busy. Also, by using very extreme or bright background images, this can make pages hard to read. By softening or fading these graphics, you can produce better versions.

One way to include graphics in a page is to wrap text around them. This can add to the information you're presenting, and looks cleaner than just having text and then an image in separate sections. If you're linking to other pages with images, it can help to eliminate the borders around them. This can be done by including the BORDER=0 attribute in your <IMG> tags.

(5) Compatibility - Before telling others about your site, it's important to test your pages in various ways. By keeping in mind how people may be viewing your pages and how compatible they appear, you can make it accessible to a larger number of people.

Considering other browsers is a good place to start testing. Often there are slight differences between how browsers will show a web page, like between Netscape and Internet Explorer. And since not everyone will be viewing your page with the same browser you use, this can give you a sense of how your page may appear differently.

But not everyone not everyone is using the latest version of a browser. The older browsers didn't always support all technologies that the new ones do, like DHTML or JavaScript. So for those people who aren't viewing your page with these technologies, it's a good idea to have alternates. For examples, if you use a Java applet for navigation, it's a good idea to include an alternate text menu for those without Java-enabled browsers.

There can also be differences between what machine someone is viewing your page on. Sometimes plug-ins may be specific for certain platforms. Avoiding these or having alternate information is a good idea. Another example is with using certain fonts on your page. It may not be a font on everyone's machine, perhaps because it's PC specific. By listing extra fonts in the FACE attribute of <FONT> tags, you can provide an alternate. (FACE="times new roman, arial, helvetica" instead of just FACE="times new roman")

(6) Publicizing Your Site - Once you've finished your pages, there are several ways to get more people to view them. You can submit them to search engines and directories. There are a few sites that will submit your page to several engines and sites, like Submit-It or Position-It. If you're willing to create a banner graphic for your site, you can join a banner exchange program. You'll need to create a banner to submit and include a banner on your page but it's a good way to advertise. Similarly, if you don't mind adding a graphic to your page, you can join a web ring. Using this graphic, you can link to pages about topics similar to those on your own.

Examples of this were discussed in the recent ACM Web Improvement Workshop and can be viewed at http://www.acm.uiuc.edu/sigweb/workshop/

---

by Jason Gallicchio

Since the dawn of time, there has been one motivating factor in the evolution of humanity - laziness. The wheel was invented because people were too lazy to drag each other around by their hair. The industrial revolution took place because people were too lazy to make their own shoes. Calculus was invented because people were too lazy to add up an infinite number of small things. The telephone was invented because people were too lazy to shout at each other across the street. Radio came about because people were too lazy to get out of their houses and listen to speeches in person. Television was invented because people were too lazy to use their imaginations when listening to the radio. And as the pinnacle of humanity, the computer was invented as the culmination of all these lazy desires. No more would people have to leave their house to look at "art." No longer would people have to pick up the phone to order pizza. No longer would people have to turn on tune knobs to listen to National Public Radio. No longer would people have even leave their chairs to make friends. Truly, people who design the heart and soul (hardware and software) of modern computers must surely be some of the laziest people the world has ever known. We are the few, the proud, the University of Illinois ACM members.

acm

student chapter
university of
illinois urbana-
champaign

membership form

Return or mail this form to:
1225 Digital Computer Lab, MC-258
1304 W. Springfield Ave.
Urbana, IL 61801

name:

campus address:

campus phone:

home address:

home phone:

electronic mail:

curriculum:

UNIVERSITY STATUS

- ☐ freshman
- ☐ sophomore
- ☐ junior
- ☐ senior
- ☐ m.a. / m.s.
- ☐ ph.D.
- ☐ faculty / staff
- ☐ postdoc
- ☐ alumni
- ☐ other

special interest groups

lug	linux users group
bug	be users group
sigarch	architecture
sigart	artificial intelligence
sigbio	biocomputing
sigbiz	enterpreneurship
sigcas	computers and society
sigdave	short-term distractions
siggraph	graphics
sigir	information retrieval
sigmicro	microcomputers
sigmusic	music
signet	networking and security
sigops	operating systems
sigsoft	software development
sigunix	unix programming
sigvr	virtual reality
webmonkeys	www development

MEMBERSHIP TYPES
return form with check or money order payable to the ACM at UIUC

- ☐ $40 for eight semesters
- ☐ $22 for four semesters
- ☐ $12 for two semesters

ACM NATIONAL MEMBER

- ☐ yes — #
- ☐ no
- ☐ currently applying

for internal use

Received Date: _____ by:_____
Chk #_______ $_______

Enter Date:_______ by: ______