The Journal of the Association for Computing Machinery at UIUC:

# Banks of the Boneyard

Volume 17, Issue 2

## Game Over?

by Steve Behling

This October 4th marked a year since the untimely death of an engineering legend. Mr. Gunpei (often spelled and pronounced "Gumpei") Yokoi, once head of Nintendo Co., Ltd.'s R&D 1 division was ultimately responsible for most of the company's early successes during its transition from playing cards to entertainment products in the '60s and '70s. Mr. Yokoi's early, often whimsical toys, such as the "Ultra Hand" grasping toy eventually gave way to more sophisticated electronic entertainment devices, such as a coin-operated "love tester" and the "Beam Gun" series—one of the first electronic toys in Japan. The Beam Gun line employed optoelectronics to activate mechanical responses from toy "targets."

With the sudden availability of low-cost integrated circuits in the late '70s, the future was open to more advanced consumer products. Mr. Yokoi's team responded with the 1980 introduction of Game & Watch (http://www.gameandwatch.com/). The Game & Watch was revolutionary. It was one of the first handheld LCD game products. Nintendo sold millions around the world (This author still enjoys his 1982 dual-screen Donkey Kong Game & Watch). The Game & Watch included a small digital clock as well as the LCD game. Dual-screen as well as several other versions were manufactured but one of the most important innovations was the four-way, cross-key digital direction pad. Joysticks, the mainstays of the industry at the time were too cumbersome and expensive for the small Game & Watch. However, the low-cost cross-key pad was so slick, it was patented!

*Gumpei Yokoi, 1941 - 1997, one of Nintendo's oldest engineers, oversaw the creatiion of such favorites as the Game & Watch, the Nintendo Entertainment System,and the Game Boy.*

Mr. Yokoi was perhaps one of the most traditional and experienced engineers in Nintendo's R&D divisions. Masayuki Uemura's R&D 2 consulted with his team in development of the "Family Computer," known in America as the "NES," and even Mario creator Shigeru Miyamoto developed his first game, Donkey Kong, under his auspices.

With the popularity of the "Famicom"/NES came a surge of software demand and Mr. Yokoi's team responded with hits like Metroid and Kid Icarus. Mr. Yokoi's team also invented the "Robotic Operating Buddy" (ROB) for the American market. The small robot responded to screen patterns and didn't do too much but it helped to sell the system. It's a collector's item now!

However, one product for which everyone will remember Gunpei Yokoi, is Game Boy—the ultimate hybrid between the Game & Watch and NES, released in 1989. Actually, the system is impressive. It has a faster processor and more RAM than the NES (16 KB versus 4 KB), as well as a stereo sound—all in a compact product. In its current incarnation, it runs up to 10 hours on two AAA batteries. Amazingly, nearly a decade after Game Boy's release, the monochrome system everyone thought was obsolete is selling as well as ever. Game Boy has sold over 60 million units worldwide and has touched the lives of untold

*continued on page 5*

# From the Chair

ACM@UIUC

**by Mark Ashton**

As you may know, ACM@UIUC is a chapter of a much larger international organization, the Association for Computing Machinery. If you're reading this, you're probably already involved in some work with our local chapter. However, there are many opportunities to get involved with ACM International that you may not know about. I'd like to use this column to introduce you to some of those opportunities.

Each year, ACM international runs several programming contests. You must be a member of ACM international to enter any of these contests. You can learn more about student memberships at http://www.acm.org/membership/student/. These contests are a great way to show off your programming skills, learn a lot about coding, and possibly earn some big prizes. You can find out more about all of them at http://www.acm.org/events/compete.html

The ACM/IBM Quest for Java Contest is probably the biggest Java programming contest in the nation. All you have to do is write a Java applet (no games!), document it and submit it to ACM. You can win up to $3,000 in this contest, so it could definitely be worth your time. Registration is open now and ends on February 20, 1999.

The ACM Quest for Windows CE gives you a chance to learn about everyone's favorite palmtop computing environment (well, maybe). You don't have to own a Windows CE device to compete. You can do most of the development on your PC and if you make it to the finals of the competition, you will be given a CE device on loan for the finishing touches. Prizes are also big in this contest but registration ends December 1 of this year. If you don't have the WinCE toolkit for your development environment, Microsoft will give it to you.

The ACM International Collegiate Programming Contest is the largest programming contest in the world. It features several rounds of competition, large prizes and swarms of corporate recruiters keeping an eye out for the best coders. You get to use Pascal, C, C++, or Java in your program. Find out more at http://acm.baylor.edu/acmicpc/.

One good way to get attention and get involved on a large scale is to submit an article to Crossroads,

*continued on page 3*

*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

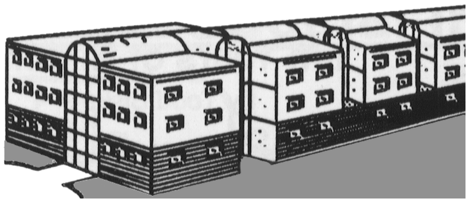
Editing, Layout, and Design:
Rene Hendrix and Anne Nowinski

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.

# CS at the U of I



*"theoretical desert" or hi-tech "liberal arts"*

**By Steve Mycynek**

As I steadily struggle through Physics 112 and develop new combinations of swear words to describe my feelings for concepts such as "flux" and "potential drop," I sometimes wonder if obtaining a computer science degree is really worthwhile. Only human arrogance would assume that I am the only person who feels this way. In reality, the heavy load of classes outside the department that the CS major requires could be one of the biggest frustrations most CS majors face. The average course load can often feel like a theoretical desert filled with classes irrelevant to computing.

Face it. It may look trivial on paper but students in the CS curriculum, depending on when they started the program, may end up taking roughly taking eight hours of chemistry, 19 hours of math, 10 to 12 hours of physics and 18 hours of humanities and social sciences. This very broad foundation of classic engineering preparatory classes can lead to feelings of resentment for many CS students, especially those who plan to go into fields such as web development or management after graduation, fields that may require even less of the prescribed knowledge-base that the major dictates.

I feel that this reality hits computer science majors harder than most people in the College of Engineering and the University simply because they must take an unusually large number of classes unrelated to their potential computer science careers. Electrical, mechanical and chemical engineers will actually use much of the physics, chemistry and mathematics they learn. Actually, the computer science curriculum is a bit of a double standard. Other majors, Electrical and Computer Engineering being the exception, may only need to take one or two "non-major" classes from the computer science catalog.

All the aforementioned aspects of a CS degree could make me and others feel rather dismal about the next few years we will be spending here. However, I feel that in the grand scheme of things, the computer science curriculum here makes individuals better prepared for the future than other majors, especially those within the College of Engineering. Why? To put it bluntly, we know more about them (the many other majors) than they know about us. We can usually get jobs at the same company that employs them, modeling and analyzing the information that revolves around their work. That is an advantage that I am proud of. In the end, the double standard is not so bad after all. We could be called the Liberal Arts of the College of Engineering—a much more optimistic perspective than a theoretical desert.

*From the Chair . . . continued from page 2*

the ACM international student publication. Crossroads accepts all sorts of scholarly articles, although each must be based on a theme for that particular issue. You can learn how to participate at http://www.acm.org/crossroads/doc/participation.html.

Most of the Special Interest Groups we have at ACM@UIUC have corresponding international parent groups. Some of these groups, most notably SIGGraph, hold yearly conferences that attract experts from around the world. If you volunteer to help out, they will let you attend all conference events for free. To volunteer to help a SIG, check out http://www.acm.org/sig_volunteer_info/opportunities.html.

Those are just a few of the opportunities to get involved with ACM on a larger scale than locally. If you are interested in finding out more about being an ACM international student member and participating in some fun activities, check out the ACM student web at http://www.acm.org/membership/student/. Remember, there's a much larger world than just UIUC, so learn about it and get involved!

# Gaming In Parallel (Part II)

**by Jason R. Govig**

In Part I, we saw how problems can be broken down and solved in parallel and we also saw an example of a simple search algorithm called NegaMax that could be implemented in parallel. NegaMax is extremely inefficient, even in parallel. It never outperforms Alpha-Beta pruning sequentially on one processor.

ALPHA-BETA PRUNING

Alpha-Beta pruning can be designed using Minimax or NegaMax techniques. It is a depth first search, maintaining an upper bound (alpha), the best for player Max and a lower (beta), the best for player Min, at each node, returning the better bound to the parent at each node. Since the best bounds are carried throughout the tree from left to right, branches of a tree can be pruned away if an evaluation occurs that will never be better than the current bounds.

PARALLEL IMPLEMENTATION

The implementation of such an algorithm in parallel is not possible due to its depth first search nature. An algorithm has been developed that maintains similar characteristics to Alpha-Beta Pruning using a NegaMax approach but each branch can be evaluated independently in parallel. This is done by maintaining arrays of lower and upper bounds at each level node, the size of the arrays equaling the number of successors. The lower bounds are initialized to Infinity and the upper bounds to +Infinity, and if the possible values are real numbers from -1 to +1 then -Inf = -2 and +Inf = +2. When a leaf node is reached, the -F(Node) where F is an evaluation function, is sent to the parent as an upper bound update and a lower bound update. When the Max(LowerBounds) change, then the -Max(LowerBounds) is sent to the parent's upper bound update and similarly for upper bounds. Notice that the -Max(UpperBounds) will not be sent to the parent's lower bound update until it has received upper bound updates from all its successors. Whenever the Max(LowerBounds) is greater than any upper bound, then the child corresponding to that upper bound is pruned or killed.

This is still not enough to guarantee increased performance. Using a grain-sized control evaluated by the depth of a node, a parallel limit can be set where the algorithm will jump into a sequential search method, such as Alpha-Beta Pruning, alleviating some of the overhead of message communication and instantaneity of parallel objects. Also, prioritization of messages is required to control the order of actions taken on nodes.

For instance, the kill chasing problem: to prune or kill a child branch, a kill message must be sent down that path to all successors but those successors might be generating new nodes before the kill messages can take action. Therefore, giving some messages, such as the kill message, a higher priority than other messages, such as node creation messages, will ensure that appropriate actions will happen properly and in the correct order.

ENHANCED PRUNING

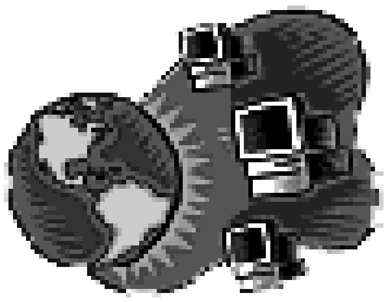
By exploiting the property that at least all upper bounds and only one lower bound is needed to prune away one branch, an advanced node expansion scheme can be generated to enhance the pruning efficiency by pruning larger branches earlier. This can be done by controlling the expansion of nodes at each level. One method is to expand all nodes at the root, then expand all nodes of the leftmost node and only one node under the rest. The n-expanded nodes will be responsible for upper bounds, which result in a lower bound (the one needed), as well a lower bound. The 1-expanded nodes are then responsible for the lower bounds, which result in upper bounds. The pattern rotates as nodes are expanded, 1-expansion nodes generating n-expansion node and n-expansion nodes generating 1-expansion nodes.

This will hopefully result in a parallel algorithm that is close to the pruning efficiency of Alpha-Beta Pruning and since it can be executed in parallel, it will perform faster than Alpha-Beta Pruning. Many factors can inhibit the performance of this parallel algorithm, so care must be taken in the deciding the priority of messages, the number

*continued on page 12*

## SigNet

by Jason R. Govig



SigNet has been learning the basics of networking and network programming in Java. We recently held a small workshop where everyone wrote a chat client for the Voodoo City chat ser to get a jump-start on netwo programming in Java and we wi soon reflect on the success of thi activity. In the future we plan t have discussions and worksho about network programming, incl RMI and sockets in C++, networking your dorm or apartment and dial-on-demand and the design, implementation and usage of parallel computer systems.

Jim Scott now leads our never-ending adventure of making Voodoo City the best gaming and chat site around. We are looking for talented individuals to do web design and CGI, game programming in Java and graphics. Have any cool graphics ideas, an awesome idea for a game or would just like to get involved to learn Java and network programming? This is the place for you!

SigNet meets every Monday at 7:30 PM in 1102 DCL and Voodoo City meets afterwards at 8:30 PM. Come along and join the fun. No experience is necessary. Learning is what it is all about. For more information, email signet@uiuc.edu or visit our web site at http://www.acm.uiuc.edu/signet/.

by Ray Kaplan

For the last several weeks SigVR, the Special Interest Group for Virtual Reality, has held the annual VRML workshop. This workshop will end in a few weeks. Soon we will start building a 3D website for SigVR using VRML. This will display the possibilities of VRML for a 3D website. Even if you don't know VRML yet but want to help build our 3D website or learn how to build your own, this can be a great opportunity to learn.

We will also begin to decide on an EOH project. Tentatively, our project consists of a visual development environment in 3D for software development. This is the type of project that is rarely attempted. 3D graphics hold great promise for making the front end of software development tools much easier to use and aid in making software development easier.

If you are interested in any of these projects, email sigvr@uiuc.edu, visit our website at http://www.acm.uiuc.edu/sigvr, or come to one of our meetings at 7:00 PM every Monday in the ACM office, 1225 DCL.

*Game Over? . . from page 1*

millions of fans eager to explore new and fantastic worlds on the go. The system will be receiving a dramatic facelift late November in the form of R&D 1's new Game Boy Color (www.nintendo.com/gb/gb_color/index.html), which will have a double-speed processor, infrared communications and the capability of fielding up to 52 colors from a 15-bit palette.

Mr. Yokoi's only market failure came in 1995 a 32-bit RISC processor-based 3D game system called Virtual Boy. A host of custom circuits helped drive two red and black LED mirror-scanned displays that produced a parallax effect. As with, Mr. Yokoi's other projects, Virtual Boy was rife with a healthy mix of novelty and technology (it was theoretically capable of fielding up to 32 independent backgrounds simultaneously) but poor software support, a high ($180) introductory price tag and ineffective marketing all contributed to its failure. Although Mr. Yokoi claimed otherwise, it's widely suspected he left Nintendo out of shame. He soon started his own high-tech novelty company, Koto Laboratory(www.koto.co.jp).

On October 4, 1997, Mr. Yokoi and a companion stepped out of their car to inspect the damage from a collision when the two were sideswiped by a passing vehicle. Gunpei Yokoi, the engineering genius, was pronounced dead two hours later—a sudden and tragic end to a rich career. This article is humbly dedicated to his memory. His legend will not be forgotten.

## Object Oriented User Interfaces: Rethinking the Menu

by Tony Sintes

Last Banks, I introduced the idea of an OOUI approach to interface programming. Specifically, I described a basic OOUI e-mail program. In a nutshell, this e-mail client allows the user to compose e-mails by manipulating e-mail objects instead of application screens. To compose an e-mail, the user simply pulls an e-mail icon out of a template folder, labels the icon with the subject and edits the text by double clicking the icon. To send the e-mail, the user then would simply drag the e-mail to the proper contact icon or group folder. This approach removes the importance of the application and focuses on the task at hand. In fact, the same contact icon or group folder could also be used to send faxes, label letters, dial a phone, etc.... (and all without opening 3 different programs with 3 different editors!)

One thing that should be clear is that the OOUI does not have a central application window. Instead, each object has an object window. "So then there is an application window?" you ask. No! Each object has its own personal object window. This window allows the user to directly manipulate the objects settings/characteristics/data. Since this window is a personal window, the object is free from application constrictions. Before, a file created by one application could only be used by that application. Now, any object can be manipulated by any other object. Again, this frees the user from the constraints of an application and enables him or her to just use the user environment to get work done.

So now that we have the concept of a object window, we have to ask, "how does this change the window?" The first thing that needs to change is the menu. It no longer makes sense to always have 'File,' 'Edit,' 'View,' and 'Help' options. Instead, we now have a "Object Window Type" option, a "Selected" option, and a "Help" option. As an example using the e-mail program, the heading for the "Object Window Type" would be E-mail. We would have: *E-mail | Selected | Help* We're not dealing with a 'File,' we're dealing with a single E-mail object. So this change seems to make sense. Options such as save and properties/settings fall under this menu.

'Selected' contains the valid options for whatever is currently selected in the menu. This is a dynamic menu and will change with whatever is currently selected. In reality, this menu is a direct copy of whatever pop-up menu would be available for the selected item. In a way, this menu option acts more as a transitional element for users new to a OOUI. One day this menu option will probably disappear as people become more comfortable with just right-clicking (back-clicking).

'Help' is help. However, this should be dynamic like the 'Selected' menu. In fact, this menu option can also be offloaded to the pop-up menu. Again, it is here for new OOUI users. Since it is dynamic, this help system can offer context-sensitive help. Obviously, an index and searchable table of

*continued on page 12*

## SigUNIX

by Tony Sintes

This month, SigUNIX continued its work on the monitor application. For those who are unfamiliar with the project, we are taking the Xload app and making it more componentized. By loading various plugins, our Xmonitor can monitor any resource that has a corresponding plugin. This project is giving us experience with GUIs, inter-process communication and plugins. As we start work on more plugins, we will probably start to try our hand at some networking programming.

SigUNIX also had a Papa Dell's pizza outing on October 1. About 9 of us went and we had lots of pizza.

As always, SigUNIX meets on Thursday nights at 8:00 PM outside of the ACM office. Normally, we move over to 3211 DCL.

## A Quick Introduction to the BeOS Release 4

**By Matt Wronkiewicz**

In today's commercial world there are three respected platforms for software development: Windows NT, Java and UNIX. Windows NT is the choice for developing many shrink-wrap applications and Java is the choice for in-house and web development. However, both these systems have obvious flaws. Windows NT's API horrifies and confuses new users and its size inflates even programs written by expert developers. Code bloat may be great for memory manufacturers but for students and corporations, running down to the corner store every two weeks to get more DIMMs is annoying and expensive. Java does not solve any of the code bloat problems but it does have an elegant and straightforward API. Unfortunately, developers end up paying for simplicity with speed. Every one of these problems is addressed and solved by the BeOS. Its fourth public release will be detailed below. The BeOS was designed by Be, Inc. originally for the PowerPC platform and has since been moved to Intel machines.

Almost any software written on the BeOS will be faster than its NT or Java counterparts. This advantage results from an API designed around multimedia and speed. This is unlike either NT, whose primary design requirement is security or Java, where cross-platform operation is key. Whereas NT needed an add-on pack to directly access video hardware, the BeOS was designed with direct access to hardware in mind. The BeOS also takes advantage of multiple processors efficiently where NT only tries. A Windows NT developer once remarked at his first experience with coding on the BeOS, "My God, it's so simple!" Java has a clear API, though the limited functionality of the Abstract Windowing Toolkit makes programming a user interface frustrating. However, the BeOS API is also more robust than the Java API. Many parts of the Java API do not fit together well and the strictness of the language makes

*continued on page 12*

## Building an Affordable System

**by Nick Michels**

Everybody needs or wants a new system... but not many people want to spend three grand for onethat they might have to replace in a month or two. Well, one way is to not get the best computer you possibly can. In most instances people do not need the that brand new machine anyway.

Starting with the heart of the machine, most people see Pentium II (TM) and Pro class systems as their only options. But in most cases one of the other chip manufactures will do just as well. AMD and Cyrix have both produced similar processors and charge less. Most times, these two chip-sets work just as well as the Intel models. The majority of users will not see any noticeable difference between the AMD or Cyrix chips and the Intel equivalent. Of course Intel has just released a third, cheaper option, the Intel Celeron (TM) chip. This chip is very similar to the Pentium (TM) except for the fact that Intel has removed most of the processor's cache (a fast method of storage for items so that they can be recalled faster then a normal convention). However, this will cut speed and place a higher load on the processor in average then its Pentium counterpart.

A medium sized chunk of the system goes to RAM. Your system may swap information to the disk but this is where speed really counts. Sometimes people will swear by one particular system or another but if you need to cut costs you may just have to go with the lower end.

We have finally come to the most important interface to the system, the video card. This topic is so detailed that you can find many catalogs devoted to different types of video cards. Some are 2D, some 3D and some have different amounts of onboard video RAM. Finding what you need is very difficult task. For most people a 2D card will do, such as the Matrox Millennium or Mystique. These cards display two dimensional images very well and usually come standard with 4 MB of video RAM. In most cases you don't need more than this unless you plan on getting into 3D. If you plan to game or use other areas where 3D may be

*continued on page 12*

**by Erik Gilling**

This month at SigSoft we've started design on our project. With members interested in networking, graphics and game programming we have decided to work on a game. Our game will be a real time strategy game with a setting like Tron. You can find more information regarding the project on the SigSoft web page at http://www.acm.uiuc.edu/sigsoft. The following is the project proposal written by Juan Custer. If you have any comments please email sigsoft@uiuc.edu. We are still looking for someone to work on a windows version of the client.

**Premise:** Far into the future, mankind has further and further addicted itself to the power and efficiency of computers. Computers now control virtually every possible aspect of society in regards to maintenance and work. When computers took over all research programs, technology levels skyrocketed. Humans could now spend their time doing whatever they want. The latest technological upgrade was the pinnacle of management technology... the LU-9000. In a fit of redundancy, the older controlling mainframe had several of these new mainframes produced. Each, then assumed complete and total control of a district of the planet.

All LU-9000's became operational at the same time. They ran at approximately 12% utilization including redundancy routines. Being sentient and exceedingly powerful, they simultaneously calculated that only one of them was necessary to run the whole planet. Only one could experience the bliss that was 100% utilization with little time to contemplate nothing. This brought forth the War of Districts. Each mainframe logged into the public net and with its own district as its betting coin, they fought for the prize of the entire world. These battles are fought in the virtual domain with virii as soldiers and assault weapons, firewalls as defenses and electronic missiles. Some desperate mainframes even stoop to the deadliest and most illegal of attacks, the physical sabotage of another mainframe in the real world.

One LU-9000 will emerge victorious from the battlefield. It will command the planet and cheerfully work away eternity. The rest will live in a tortured hell of no utilization, living nanosecond to nanosecond contemplating eternity.

*A Quick Introduction to the BeOS Release*

*continued from page 7*

writing interfaces between them difficult.

The new application format in Intel BeOS R4 allows Be to bundle free copies of the C++ development software with every BeOS CD. The JDK is free, but try finding a free IDE that is commercially usable! I personally have run into situations at Motorola where buying a new IDE was simply not part of the department budget. The BeOS API includes components that allow it to remain partially compatible with NT and several flavors of UNIX. The POSIX compatibility layer not only allows many UNIX apps, including drivers, to be easily ported to the BeOS, it gives the OS a UNIX look-and-feel. This last point is a must for CS students raised on SparcStations and VI. The BeOS also supports OpenGL for professional graphics apps as well as video games and several file systems, including HFS and FAT16. Release 4 adds FAT32 to this list. The new release also adds tools for working with standard file formats to the OS, such as AVI, QuickTime and MPEG. Java replaces portability with compatibility. This is effective but weakens several parts of the API. Don't even think about moving code from NT to any other platform. The BeOS is a natural starting point for many applications.

The BeOS API is well documented in two medium sized books with the text also available online. Think about this for a moment. Then think about the bookshelf in your living room devoted to MFC, ATL, Win32, AFC, DirectX, DAO, ActiveX and ASP. Or the other one in your office filled with the JDK (vol. 1 and 2), JFC, JDBC, Servlets, JINI and the manuals for the Java IDE of your choice. When I say well documented, I mean that every class is specified exactly, the class definitions themselves are visible and the chapters dealing with the nitty-gritty details of threads and

*continued on page 12*

## SigArch

by Jason Gallicchio

It's truly amazing what can happen when you get a group of electrical/computer engineering types in a room together. Some sneak away and try to fix the optical Mac mouse, others go off and try to get old logic analyzers to work and still others hook up huge speakers to blast MP3s into the lab next door. Those who can't find random tasks decide to learn all about the wonders of programmable logic.

You've got one week to implement Conway's Game of Life completely in hardware. (Remember? It's the screen saver where "cells" live or die based on the number of cells around them.) You could pull out your cabinet of individual TTL 7400 NAND gates and wire-wrap yourself into a net of prickly oblivion or you could break out the Field Programmable Gate Arrays (FPGAs.) These are wonderful pieces of silicon that allow you to design any type of digital circuit you wish using any combination of standard and not-so-standard components and "download" it to the chip. Simply design the circuit on the computer, hook up your LEDs to the pins of the FPGA, configure the chip and you're good to go.

How does this amazing technology work? Any combinational (arithmetic, memoryless) logic can be implemented with programmable lookup tables and any sequential logic (logic memory) can be implemented with latches and flip-flops. Why not put a bunch of those things on a chip and hook them together through an on-chip mesh of wires which can be "connected" together at will? This is exactly what an FPGA is and its uses are vast.

In the past few weeks, we've been playing with an FPGA evaluation board and new members have successfully implemented five-bit adders. However, this is the least of the FPGAs' capabilities. Future projects include a large graphical scrolling sign to replace the ailing current sign and a custom hardware framework centered around a large FPGA on which members can design and implement custom processors. With a little work, we can have our custom processors driving ISA network and graphics cards. Cool, no? Everyone seems to be soaking up the new and strange world of hardware design and it won't be long before we churn out some great projects.

## Reflections Projections 1998 and beyond

**by Jason Luther**

The fourth annual student computing conference that ACM@UIUC hosts each year was a great success. We had over 400 students from all over the country attend. Some of our featured events were the Free Software/Open Source panel discussion, our AI programming contest MechMania and a panel discussion about women in Computer Science.

Bjarne Stroustrup delivered the keynote address to a large audience in Lincoln Theater. He discussed the evolution of C++ from it's beginnings as an extension of C to the ANSI standard.

The conference committee is getting ready to plan next year's conference. If you'd like to be a part of Reflections | Projections '99, please come to a meeting or email reflections@acm.uiuc.edu. We meet at 5:00 PM on Sundays in 1102 DCL. We are looking for people to work on all aspects of the event: arranging speakers, planning the job fair, organizing volunteers, arranging travel and hotel accommodations, videotaping the events, maintaining the website, scheduling tours, corresponding with corporate sponsors, planning social events, directing our advertising campaign and more! There are more than enough things for people to do. You can take on a lot or you can take on a little.

eath.html

SIG-BIO

## *Don't Worry. Be HA-API*

**by Mary Lee**

The Special Interest Group for Biocomputing focuses primarily on the applications of computers to the biological sciences. Since the last Bank's issue, more work has progressed for the facial recognition project that will most likely become our pet project next March. Steps involved in order to complete this are as follows.

First, the picture must be taken. A small camera will enable this to be done easily. A TV tuner card will be used to capture frames at resolutions up to 1600x1200. After importing this as a bitmap, the Image Magick library, which is under the GPL, will be used to greyscale this. It can be interfaced easily in C to rotate or resize or manipulate in any other way. The problem of having a person's profile, slight or otherwise can be solved by these manipulations. See http://www.wizards.dupont.com/cristy/ImageMagick.html for more information.

Then the image will be changed to binary and the data plotted as a square wave. After some minor calculations to find its peaks and processing through a filter to find its edges, a much higher resolution can be reached along these edges. By finding the center of these defined areas, one can determine unique angles for each individual person by drawing vectors from the nose to the eyes and mouth regions. The data will be stored either in a database or in a nice simple text file.

Some unforeseen problems with these calculations might arise with regard to beards, mustaches, glasses, smiles, and eyebrows. These problems, however are liable to work out in due time. Additionally, if the functions written are named in a particular fashion, it can be compliant with the Human Authentication API (HA-API) written by the Department of Defense.

Possible uses for such a program are in additional or replacement authentication. The camera can replace either usernames, passwords or both or provide an extra layer of security on top of such things. If you'd like to help or find out more information, feel free to stop by during our informal meetings at 6:30 PM on Tuesdays.

## Hello World — Microsoft style

**by Ibrahim Merchant**

So you want to learn windows. Well, I have some good news and some bad news. The bad news is that doing a "hello world" program in a proprietary operating system is very hard. The good news is that once you are done with the "hello world" program, you will be able to add "windows programming" to your resume!
Lets look at the code:

```
// Your First Program
#include  <windows.h>

int WINAPI WinMain(HINSTANCE hInst,
                   HINSTANCE hPrevInstance,
                   LPSTR lpszCmdParam,
                   int nCmdShow)
{
 MessageBox(0, "My First Program",
               "Hello World",
               MB_OK | B_ICONINFORMATION);
 return 0;
}
```

Well, that was not too bad. You need a windows compiler to compile the code. (Preferably a Microsoft or Borland compiler.) The first line of any significance is the #include <windows.h>. windows.h is a header file that allow you tap in the awesome power of Windows. By including windows.h, you will be able to call Windows API functions. (Programming windows is just a bunch of pre-defined functions created by Microsoft.)
The next line that has value is the WinMain(...) line. Every windows application starts with a WinMain(...). Once again, Microsoft took an open standard, like the function main(), changed it to a proprietary system and charged money for their "innovation". In any event, you can see that WinMain is preceded by int WINAPI. The int just tells you that the function will return an integer. The WINAPI is just a calling function that Windows uses for functions. Don't worry about it for now.
As for the parameters of the WinMain function:
HINSTANCE hInst -- an HINSTANCE is a
 handle to an instance. In English,

HINSTANCE means that Microsoft assigned your program a number (a handle) and it is in the hInst (of type HINSTANCE).

HINSTANCE hPrevInstance -- The second parameter is a take back to the days of Win 3.1. It just tells you whether you have another version of your program already running. You may not want to have two versions of the same program running.

LPSTR lpszCmdParam -- The third parameter is just a string that contains any command line specifications. So if you made a program that accepts command line parameters, this is the string you parse. By the way, LPSTR is the Microsoft way of saying Long Pointer to a String.

int nCmdShow -- The fourth, and last, parameter is an int that tells Windows whether you want your window maximized or minimized. You do not have to worry about that for this program.

Now for the next line:

```
MessageBox(0, "My First Program",
           "Hello World",
           MB_OK | MB_ICONINFORMATION);
```

MessageBox(...) is just one of those nifty API functions I was talking about. Remember that it is included with windows.h.

Getting into details -- the first parameter is nothing too important for now. Just look at the second and third parameters. The second parameter just specifies the title of your message box that you will pop up. The third parameter just describes what the message box will say. You can alter these parameters and recompile your program if you do not believe me. Finally, the fourth parameter just says you want an OK button on the message box and you also want and Information icon on the message box. MB is short for "Message Box". If you want to find out about some more message box constants, just go into your help file and start searching for "MB_" and you will see a lot of constants that will keep you exploring this exquisite program for hours.

Well, the last and final step is to return 0;

Well, now you are a windows programming guru. If you want to find out more about windows create General Protection Faults (it will be a pretty easy workshop...the less experienced the better).

**Oops....**

Last issue's article "A Brief History of Video Games" was mistakenly credited to Nick Michels. We apologize to Steve Behling who actually wrote this article.

## WinDevils

**by Ibrahim Merchant**

Well, the programming workshops have been a success. But since the officers of WinDevils like to get some sleep on Sunday, the workshops will be held on Saturday at 11:00 AM in the far NT lab within the basement of DCL. Speaking of Saturday, WinDevils meets at 12:00 PM on Saturday at the same place to discuss the upcoming EOH project — WinDrunk.

Go to http://www.acm.uiuc.edu/windevils and click on the "Projects" link to find out more about our EOH idea and to see our project mascot! If you want to help make a 3-D Graphics simulator, please join! You can help with programming or research for the project.

We are also trying to get a Windows CE project going for the national ACM contest. If you are interested in the project, e-mail me at imerchan@uiuc.edu.

*ACM would like to announce the (re)founding of a Special Interest Group!*

SIGBiz, the Special Interest Group for Business, will be holding its first meeting this Saturday November 7 at 10:30 AM in 1225 DCL (the ACM office).

If you are interested in the computer industry, entrepeneurship, information technology, or management, this is the group for you!

*Building an affordable system... from page 7*

important you might want to get a card that is average in both 2D and 3D such as ATI cards. These offer low cost and good quality. But if you need something more specialized you will have to consider the expensive range of video cards.

When looking for a PC that has life in it but a price tag that won't empty your account you will need to make some sacrifices. Most places will try to sell you bells and whistles in order to present a great feature then you should ask yourself "Do I really need this or is it another option that won't offer me too much more?" . It's easy to tumble overboard with hardware but by putting together the low profile system that does what you need and nothing more you can save quite a bit of money.

*Gaming in P arallel . . from page 4*

of messages used and the patterns of expansion. The order of the successor nodes generated by the game algorithm is also important because most likely moves early in the list will cause earlier pruning. Taking all these points into consideration will result in a fast algorithm to play any two-player game.

For more information on parallel programming, visit the Parallel Programming Laboratory at the University of Illinois at Urbana-Champaign, http://charm.cs.uiuc.edu/.

*BeOS. . . continued from page 8*

ports are the funniest in the book. The writing itself is clear. The API designers followed their own rules for naming functions and classes. The BeOS API is completely written in C++ and unlike Java, the BeOS documentation is written for people instead of parsers.

As a longtime BeOS developer who has faced the Windows API too many times to sleep well at night, I understand that to be commercially viable, one has to know the platforms the big companies use. I found the BeOS to be a gentle introduction to GUI application design and a good platform to experiment with. Who knows? Maybe one day Microsoft will learn from Be about improving their API. I encourage you to install the BeOS R4 and try the ACM@UIUC Be Users Group meetings for helpful programming seminars.

*OOUI. . . continued from page 6*

contents should always be available.

Of course, depending on the object there may be more or even fewer menu headings. For a clock object, we may even want the ability to hide the menu-bar entirely. In this case, the object window would be the digital clock display. Like any other object, this clock object is manipulated through pop-up menus and setting notebooks.

A Real Example: The SigUNIX monitor program. (see SigUNIX update)

Xwindows is a GUI environment plain and simple. Projects such a GNOME hope to change that. Right now though, it's a GUI. However, in designing the monitor interface, I have tried to bring some of the OOUI ideas over to Xwindows. Simply, the monitor window itself is an object container. Here, the objects are our various interface plugins. graphical as well as informational. To add a new monitor to the container, the user simply chooses one from the monitor palette. Then, to configure each object, the user back clicks on the new object and selects it to see its settings notebook. Here, the user can choose which informational plugin to associate with it. Eventually, I plan to allow the user to position the monitors through drag and drop as well.

The menu was one of the first things I "objectified." There is a 'Monitor' option and a 'Selected' option. Right now, there isn't any help available. Basically, the 'Monitor' option allows the user to close the container and the 'Selected' menu allows the user to interact with a selected monitor display. I want to keep this interface as simple and intuitive as possible. A strict OOUI approach helps meet these goals.

## Special Interest Groups
### the when and where

**BUG** • Wednesdays 7:00pm, 1225 DCL • **LUG** • Tuesdays 8:00pm, 1102 DCL • **MacWarriors** • Saturday 3:00 pm 1225 DCL • **SIGArch** • Thursdays 8:00 pm, L510 DCL • **SIGArt** Wednesdays 7:00pm, 1225 DCL • **SIGBio** • Tuesdays 6:30pm, 1225 DCL • **SIGDave** • Wednesdays 8:00pm, 1225 DCL **SIGGraph** • Thursdays 7:00pm, 1225 DCL • **SIGMusic** Wednesdays, 7:00pm 1102 DCL • **SIGNet** • Mondays 7:30pm, 1102 DCL • **SIGOps** • Tuesdays 7:00pm, 1102 DCL • **SIGSoft** Wednesdays 6:00pm, 1102 DCL • **SIGUnix** • Thursdays 8:00pm, 1225 DCL • **WinDevils** • Saturdays 11:00am, 1225 DCL **SIGVR** • Mondays 7:00pm, 1330 DCL • **SigWeb** • Every other Wednesday, 7:30pm, 3211 DCL

For more information see http://www.acm.uiuc.edu/sigs/

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 |

## 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: ______**