EXHIBIT H

Trip Report
Win32 Developers Workshop Featuring Chicago
September 9th and 10th, 1993

Note: The information obtained at this preview was obtained under non-disclosure. It was pointed out several times over the two days that Chicago will follow the beta release program of Windows 3.1 and not that of NT (meaning that there will not be a $69 beta CD, nor will there be 80000 beta copies sent out). This also means that Chicago will be under non-disclosure until its release.

Do not discuss this information with anyone outside of WPCorp.

Summary

This workshop was attended by Bruce Brerton, Nolan Larson, Tom Freeman, and Rich Hume. Each of us has a copy of the presentation materials that were made available.

The August beta that we have is referred to internally at Microsoft (and occasionally within this document) as M4. M5 will be the first end user beta of Chicago and is planned for November. M4 is quite stable, much of this report was written using WPWin60 running on Chicago (using a 25mhz 386sl portable w/8meg). If M5 does in fact ship in November, Microsoft will be on track for a mid-'94 release of Chicago (likely to be called Windows 4.0, but that is a marketing decision).

Microsoft is planning a Professional Developer's Conference in December. At that conference more detailed and up-to-date information will be given about Chicago (e.g., what will Chicago support that NT will not at first support). Also, more information will be given on Cairo. The first beta of Cairo is planned for 2Q94.

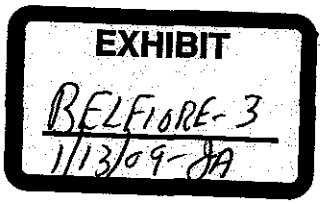
The performance goal for Chicago: all else being equal, Chicago will perform better and require less disk space than Windows 3.1.

The minimum hardware for Chicago: an 80386sx with 4megs of RAM.

A Windows NT interim release is planned for 2Q94. This update will include some support for planned Chicago enhancements to Win32 as well as the 32-bit OLE 2.0.

Microsoft is attempting to refer to Win32 as just Win32 rather than Win32s, Win32c, and Win32. David Weise rather sarcastically (and aptly) pointed out that there is now just one Win32 spec that just happens to have three versions.

Microsoft is shipping copies of Windows at a rate of between 1 and 1.5 million per month. At the time of Chicago's release there should be 40 million (+/- 10 million) installed copies of Windows. They expect an upgrade to Chicago to be a no-brainer. "Those that won't upgrade probably won't upgrade their applications either."



As per usual, the two key points stressed over and over again are the need to go 32bit and the need to support OLE 2.0. It was also interesting the amount of attention given to Plug and Play.

The nine Chicago "Points of Light":

- Win32
- OLE 2.0
- Plug and Play
- Long file names and UNC paths
- New user interface and shell
- Workgroup features
- Mobile platform
- Internationalization
- Robustness and performance

The sections are presented here in the order that they were given (except for the breakout sessions). If your time is limited, I would suggest reading the review of Paul Maritz's talk on System Strategy Overview.

Developer Highlights
Glenn Thompson
Developer Relations Technical Evangelist
Glenn reviewed the mission of the Developer Relations Group within Microsoft. He discussed the various forms of support and communication available today including compuserve, the developer network cd, the developer network newsletter, and the DRG porting lab.

He stressed (as did many of the presenters throughout the two days) the importance of moving to the Win32 API and of implementing OLE 2.0.

The nine "Chicago Points of Light" were presented without much discussion except for the first two which are the most important. Most of the others were discussed in various presentations over the course of the two days. The nine "Chicago Points of Light" are:

- Win32
- OLE 2.0
- Plug and Play
- Long file names and UNC paths
- New user interface and shell
- Workgroup features
- Mobile platform
- Internationalization
- Robustness and performance

In addition to presenting the workshop agenda, one other thing discussed that is of note is the upcoming Win32 Professional Developers Conference. It will be large, but we should register early (the last one they held, in July of '92, had over twice as many people register as they anticipated). The PDC will be held Dec. 13th - 17th in Anaheim California. They will discuss Win32, OLE2.0, Plug and Play, Chicago, and Cairo. They say that many things will have changed with Chicago and that they will be presenting things not covered at this workshop, including a more in depth discussion of Cairo.

System Strategy Overview
Paul Maritz
Senior Vice President, Systems Division
Paul presented the same "Windows family" of products software objective that they have been presenting for two or more years now. There was an added emphasis on the low end reaching down to non-PC devices that was not discussed as recently as the beginning of this year. This new emphasis is presumably to start laying groundwork and gaining mind share for Windows At Work.

Two key system software objectives were presented. The first is to make things simpler and better for the end user through OLE (document centric model), a simplified user interface, and plug & play. Paul claimed that about half of all their support calls

occur during software setup and initialization. The second key objective is to introduce new possibilities though new system features. With Cairo, Microsoft is laying the plumbing (apparently an in vogue word at Microsoft as it was used by a number of presentors) for workgroup and distributed computing. In particular, the new distributed object-oriented storage system was mentioned. Paul emphasized that most of the interesting things that are being done today require a database (e.g., email, document management and forms routing). Microsoft's intent is to make the file system capable of meeting the database needs for these classes of applications.

The momentum of Windows was discussed. It is interesting to note that different people gave different numbers for the same statistics. Often the numbers quoted didn't match the slides being presented. Where possible, the numbers presented here indicate who said it or whether it was from the slide. Paul said that windows is selling at a rate of 1.5 million per month mostly through the OEM channel (the slide said 1 million per month). They guess that 30% - 50% of those copies are actually being used. Paul stated that there is a 70% penetration of Windows on PCs (the slide states that new machine penetration is >60% worldwide). Paul predicted that about 40 million copies of Windows will have been sold by the time Chicago releases (the next day, Brad Silverberg gave the number as 50 million). As far as NT is concerned, the statement that "significant corporate and integrator evaluations are underway" was made. There are supposedly about 100 Win32 applications shipping today with 500 due by the end of the year. According to Paul, Nordstroms has a 50 gigabyte database running off of NT. It was stated that NT consists of approximately 5.9 million lines of code.

Microsoft breaks their Windows family framework into three categories: corporate/mission critical; personal; and Non-PC. They see technology migrating downward as the entry level hardware for each category increases. In 1993, the corporate/mission critical category is addressed with Windows NT 3.1 and (based on comments made from several Microsoft people) requires a 486 or RISC processor with 16 MB RAM. In 1994/95, this category will be addressed by Cairo (or Windows NT 2.0) on a 486 or RISC processor with 16+ MB RAM. In 1993, the personal category is addressed by Windows 3.1 and a 386 processor with 4 MB RAM. In 1994/95, they see this category as being addressed by Chicago using the same hardware. In the Non-PC category, they predict the hardware will move from the 1-2 MB RAM/ROM level in 1993 to the 2-4 MB RAM/ROM level in the 1994/95 timeframe.

Regarding releases, they still believe they are on track for a mid-1994 release of Chicago. They are planning an NT update in the second quarter of next year. The first Cairo beta is planned for the second quarter of next year with release of Cairo around mid 1995 (according to Brad although Paul gave a release date of "early '95" for Cairo). Windows for Workgroups 3.11 is planned for release this year. They also listed DOS 7 in the 1994/95 timeframe. Brad said that there is a window of opportunity for Chicago and that they would cut or leave out features rather than miss a mid '94 release.
Note: It's hard to know, but I definately got the feeling that they are serious about the Chicago timeframe and that they are confident that mid-'94 is still possible (Rich).

The key benefits of Chicago that Microsoft is touting are: ease of use; 32-bit multitasking OS; and connectivity (Novell, NFS, and Banyon were mentioned in addition to their own network). It was stated that - all else being equal - a conversion from Window 3.1 to Chicago should provide a net improvement for the user in both speed and disk space.

Microsoft is working with a number of other companies to define the framework for plug and play PCs. When this framework is in place and the hardware and software are available, it will be possible to dock and undock a laptop without rebooting; detect and use IR or RF based resources dynamically; as well as simply plugging in an ISA card and having it work without worry of conflicts. This will be a really big thing if successful. It should leap frog the capability Apple has achieved to date with their Duo machines.

In discussing the new shell and a more intuitive interface, I found it interesting that Paul described the importance of observing how real users use the software (as though Microsoft had recently discovered contextual inquiry).

In presenting the flavors of the Win32 API, it was indicated that NT should provide a subset to Cairo and Chicago should provide a subset of NT. However, Chicago has added a number of things that are not in NT today. The release planned for Q2'94 will address some of these but not all. At the December PDC, Microsoft plans to enumerate what Chicago will support that NT will not (at least at the time of Chicago's release). Based on everything said, it is likely that if we write a "great" Chicago application it will not run on NT unless we special case some of the code.

On the subject of Cairo, Paul said that one of the founders of Banyon, Jim Olichin (sp?), is the "owner" of Cairo at Microsoft (I assume that this is the same as Brad Silverberg being the "owner" of Chicago). He also stated that there are 150 developers working on Cairo. Cairo makes heavy use of OLE. This led into a discussion of OLE which was much like all discussions of OLE. While discussing the document centric approach, Paul emphasized the benefits of allowing the user to customize the environment. While this is a good next step, I would hope that we can pursue a "user-centric" approach where the software customizes itself to the way a particular user works. Although this is a more ambitious goal, it has far greater potential for widespread use.

One of Paul's OLE slides stated that the 32-bit OLE for Windows using Win32s supports full interoperability with 16-bit OLE apps on Windows 3.1. One of the 32-bit OLE developers said that a 16-bit OLE application would have to have a 32-bit helper DLL to connect to a 32-bit OLE application. This helper dll would take care of the translation to and from UNICODE that 32-bit OLE requires.

Paul said that they are guessing that about 2 years after Chicago ships there will be a major down movement of technology from NT to Chicago. They guess that in that timeframe 16 MB RAM will be the entry level machine.

He also stated that there is a "strong effort to release Chicago worldwide simultaneously". They expect to ship the Kanji version of NT 3.1 about 4 months after the US version shipped. Another quote: "The asian market is far and away the fastest growing market".

Chicago Shell Overview
Joe Belfiore
Program Manager, Chicago Shell/UI
Much more information on this topic can be found in the trip report from the Chicago User Interface Design Preview.

Some goals for the Chicago shell are to remove the Progman/Fileman duality, begin supporting document centricity, simplify the user interface, and share a consistent UI with Cairo.

Joe demonstrated an alternative look and use for the tray that is being experimented with. The alternative really turned the tray into a task switcher. This is appealing because of the large number of users who deal with all applications in a maximized state and often shut one down to move to another. It was pointed out that the current tray appears to contain a text entry field in its default configuration. This is likely to cause problems for users.

A general undo capability is planned for the shell.

At the December PDC, there will be more discussion of border styles and graphic design principles used in Chicago. Also, how to apply those same principles to applications.

The MS Mail client will tie into the shell as just another folder. The mail client will be MAPI compliant. Joe stated that there are no plans to allow ISVs to extend the explorer in this way. This was raised as an issue during the presentation. I (Rich) discussed this separately with Dennis Adler (Chicago program manager), Brad Silverberg, and Joe Belfiore. I explained to each that this was an unacceptable situation. They each told me to write down the issue on the workshop review form (which I did). I also plan to complain about this via Compuserve. It may be necessary to elevate this issue if we don't hear anything positive before the November beta. The concern expressed by Brad Silverberg (which I can understand) is that Cairo will be fully extensible. Cairo needs to be compatible with Chicago applications. The MS Mail client is being tied in using some real hacks. They don't want to document these hacks because they don't want to force Cairo to support them. I explained that even if the method is ugly and we know it won't work on Cairo, we need the ability to tie our mail system into the shell replacing the mail system that is shipped with Chicago.

Win32 API
George Moore

Program Manager, Windows Chicago Core
George said that a 2.01 release of OLE providing a 32bit implementation is planned for first quarter '94. It was also stated that the 32bit OLE uses UNICODE for all text.

Win32s was reviewed. It appeared to be the opinion of the four of us that Win32s doesn't seem like a viable API set to be developing to with Chicago as close as it is. Especially since Microsoft itself is saying that Win32 on Chicago "increases the common denominator". A great Chicago application is not going to run on Win32s without a lot of special cased code.

This entire presentation was pretty much an attempt to show Win32 as a standard API supporting a broad range of OSs. In fact they have dropped the Win32c and attempt to talk about Win32 (although they still referred to Win32s). Basically the whole thing is a bit of a sham. There are three distinct API sets and they do not have a strict subset heirarchical relationship.

GDI Module Enhancements
David Weise
Software Design Engineer
The goals for the GDI enhancements were to expand limits, improve performance, improve printing, provide Windows NT compatibility, and to support image color matching (aka device independent color).

To help expand limits, GDI uses a 32-bit heap where the first 64K are maintained in a compatible way with Windows 3.1. Regions and physical objects have been moved out of the 16-bit heap. Owned DCs are still allocated out of the 16-bit heap. This means that all the recommendations regarding the use of owned DCs still apply to Chicago. GDI now tracks ownership and cleans up after applications that don't clean up after themselves. Objects created by a 16-bit application are cleaned up after all 16-bit applications have terminated (because the handle may have been passed to another application).

To improve performance, part of GDI has been rewritten in 32-bit code. There is now a 32-bit True Type rasterizer that fixes a bunch of problems with the current 16-bit rasterizer. It was noted that the 32-bit TrueType rasterizer uses memory mapped files for performance. Chicago will provide the fastest software driver for flat frame buffer video boards as well as hooks for hardware accelerators.

Printing performance has been improved using metafile spooling and a 32-bit printing subsystem. The new printing subsystem also provides better background printing as well as support for bi-directional printing devices. Code changes were made to support re-entrancy in printing.

GDI was enhanced for compatibility with NT. In fact, where possible the NT code was taken and used. Chicago supports paths, beziers, enhanced metafiles, color cursors, and animated cursors. All Win32 path APIs have been implemented however, the only

primitives that get recorded in a path are LineTo, MoveTo and PolyBezier. David pointed this out and almost pleaded with the attendees to complain that this was not acceptable. Apparently this was a scheduling/resource decision that he doesn't particularly agree with. David also pointed out that paths are not pixel perfect with Windows NT.

To support beziers, the Windows NT code was ported to 386 assembly code. David pointed out the beziers are not pixel perfect with Windows NT.

Chicago provides partial support for Windows NT enhanced metafiles. All enhanced metafile APIs are supported, but only partial support of world transforms is provided (scaling only).

The following goals were defined for image color matching: enable color aware applications; allow for color WYSIWYG; allow for device independent color; work across all output devices; allow third parties to add value; ease of use; and performance. Applications do not have to change to take advantage of ICM however APIs are available for those sophisticated applications that want to manipulate the color mappings.

Chicago will support Windows 3.1 display drivers although they will not work as well as new Chicago drivers. There are old drivers that were based on DDK code that had bugs in it. This is being fixed for Chicago. In particular, the ATI drivers will work properly for Chicago.

GDI on NT supports the following which will not be supported on Chicago: wide styled lines; forms; transforms; dithered pens and text; total 32-bit internals; 32-bit coordinates; and fine grained re-entrancy.

Chicago Networking
John Ludwig
Group Manager, Windows Networking, Personal Systems Group
Chicago will provide all the same networking functionality provided by Windows for Workgroups. The goals for Chicago networking include enabling 100% 32-bit protect-mode and providing full compatibility with existing MS-DOS based drivers as well as Windows 3.1 WinNet drivers.

Chicago provides new network APIs that provide support for: concurrent connection to multiple networks; better performance, multi-tasking and robustness; support for new Chicago features like long file names, 'point and print', and 'plug and play'; common code with Windows NT drivers.

The Chicago WNet APIs are derived from Windows NT providing Win32 compatibility. Enumeration APIs are provided for shell browsing. Additional APIs are provided to find network resources.

Essentially, Microsoft is continuing to attempt to promote and enhance network independent APIs for security, printing, etc. It remains to be seen how well this strategy will work with Novell networks. Novell and Banyon were both mentioned as key networks that Microsoft intends to support.

There will be more MAPI discussions at the December PDC.

Chicago Base
Russ Arun
Program Manager
Russ discussed the architecture of Chicago and some of the major internal components. Running at ring 0 are the installable file system (a 32-bit FAT, CDFS, SCSI, and NETWORK redirectors will be provided), and the virtual memory management services (memory manager, scheduler, VxD services, dynamic VxD loader, MS-DOS mgr, and drivers including keyboard, display and communications).

The VMM services are 32-bit running in protected mode. Separate address spaces are provided for Win32 applications. All Win16 applications run in the same address space. The VMM services provide thread based resource tagging which allows the system to clean up after applications. The debug kernel provides parameter validation enabling checking in VxDs.

The VMM services improve performance through 32-bit code and through the use of thread based preemptive scheduling. Memory mapped files are supported and are the recommended method of sharing information across applications. Memory mapped files provide very good performance when dealing with large files.

With Chicago, the swap file is just a regular file. It dynamically grows as needed and shrinks when not needed. This simplifies setup and UI.

VxDs are loaded dynamically as needed reducing the working set size for some applications. New VxDs (written to new APIs) are pagable which minimizes the size of locked code. There are a number of enhancements for VxD writers.

The VMM services have an interrupt latency of less than 200 microseconds. This allows support of 19200 baud communications.

The following describes the system memory map for Chicago:

0 - 1meg	Virtual 8086 - MS-DOS memory (per VM)
1 - 4 meg	mostly open (4meg is minimum program load address) (per VM)
4meg - 1.5gig	Win32 applications live here (per Win32 app)
1.5gig - 2gig	Win16 apps, DLLs, and other shared objects (shared mem)
2gig - 4gig	Ring 0 components (shared mem)

It was noted in the Kernel breakout session that no assumptions about memory layout

should be made. NT memory is not organized this way.

Chicago's installable file system is a 32-bit protected mode system providing a layered file system architecture. Multiple file systems are supported and a protected mode path exists from the API to the media. Features include: Win32 file API, long file name INT 21 API, Win32 applications get >256 handles, and there is an exclusive volume access API for utilities like disk defragmentors.

Chicago provides a 32-bit FAT file system that provides long file names and improved performance through intelligent caching and the use of 32-bit code.

Chicago provides a "single MS-DOS application mode" in which Windows is actually swapped out and the machine is turned over the MS-DOS application. In this mode, long file names will not be supported. Long file names are supported as separate/additional directory entries that use a combination of directory/system/hidden (and maybe other, I don't remember) attributes. This method of supporting long file names affects disk utility programs. Microsoft will be making available a white paper on ISV and end user issues related to long file names.

Win32 applications "get long file names for free". Filenames are all in the ANSI character set unless the application specifically calls an API to switch to the OEM character set. A new INT 21 API is being provided for Win16 and DOS applications. This API uses the OEM character set.

Chicago will provide a CD file system (CDFS) that is a fully 32-bit and compatible with ISO 9660. It provides intelligent caching and takes over MSCDEX if it is used.

A 32-bit network redirector is provided as another file system.

Enhancements for MS-DOS applications include a ShellVMTitle API that allows the application to set the title of the MS-DOS box. There is also a shutdown API that allows an MS-DOS VM to be shut down without UI.

The MS-DOS box has a new toolbar and support for TrueType fonts that scale automatically as the window is resized. With Chicago there is no more PIF editor. You can start Windows applications from the DOS command line. More conventional memory will be made available to DOS applications by replacing some drivers (mouse and MSCDEX) with protected mode equivalents. The DOS box also has better timer support for MS-DOS games. Core DOS utilities like command.com and xcopy will support long file names. There is a new 'start' command that can be used from the DOS box to launch an associated application (e.g., START DOCUMENT.WPD would run WPWin60).

Russ pointed out that Microsoft is making an effort to improve accessibility. For example, they're planning aids for people with hearing and motion impairments. This is at least to some degree prompted by the need to conform to the Americans With

Disabilities Act.

Plug and Play
Moshe Lichtman
Senior Program Manager
Microsoft is spearheading an effort to lower the cost of owning and maintaining a personal computer. There is also a great need to more effectively handle mobile machines with and without docking stations. Users should not have to deal with IRQs or DMA addresses etc.

The constraints imposed on the plug and play architecture include: hardware independence and portability across operating systems; compatibility with the installed base; extensibility; reduced complexity; and price. The solution framework centralizes resource and configuration management, keeps the system layout always known, allows for devices to dynamically change their settings, and provides for smart applications and drivers.

The plug and play architecture attempts to accommodate existing hardware devices while taking advantage of new plug and play devices.

Moche went in to quite a bit of detail as to how the hardware information is obtained and maintained. If you are interested, I would suggest getting a copy of the slides presented. There may also be some plug and play documentation with the Chicago beta.

Plug and play will impact all WP applications at least in the area of dealing with open files on a drive that is about to disappear. There will be additional impact of Office as it should be possible for the software to dynamically switch between remote and direct modes.

Remote Network Access
Wassef Haroun
Program Manager
Chicago contains a bunch of new support for remote network access. There is an RNA folder that contains icons for initiating various explicit connections. There are new common dialogs for connecting (including password prompting), file transfer, as well as a connection status dialog.
There is also support for implicit connections (which was presented as likely to be the most common form of connection). Implicit connections refer to RNA connections that are a byproduct of some user-network interaction (e.g., a connection might start when a user invokes a link to a remote net share, printer, or redirected drive).

RNA is part of WOSA. It provides session management APIs for starting, stopping, and managing connections. RNA session management SPIs are defined for adding authentication, routing and driver services.

RNA is based on Chicago TAPI, Win32 communication APIs, and unimodem (a modem independent set of apis).

The Chicago desktop can be a dial in host, with or without dial back capabilities for additional security.

RNA has the capability to resume a suspended session without prompting for authentication. Wassef encouraged feedback on this. Do we want to allow users to resume a suspended session without entering a password, or does this present too much of a security problem?

Microsoft is also considering supporting continuous authentication (like kerberos). Wassef requested feedback on this as well. It was stated the Cairo will support kerberos.

Wassef said that the Chicago team is also seeking input on what type of information would be useful to an application so that it could modify its behavior on a slow connection. Microsoft is planning to put together a set of slow connection awareness guidelines.

File Synchronization
Bill Veghte
Program Manager, Mobile Services/PSG
This was a very short presentation (the presenter had to leave for another meeting) that generated a lot of questions.

The Chicago team is trying to address the problem many people face when they try to keep files in sync across multiple PCs (e.g., a portable, the home machine, and the office machine). The issues they are trying to address are how to store the relationship between files, updating the files, and the UI for establishing these relationships and performing the update.

The model they are proposing to address this is the briefcase. The user can create a briefcase and copy a file into it (there is an AddObjectToBriefcase api). Information is stored in the briefcase about the original file (there is a twin.dat file). The briefcase can then be moved to another machine. At some later point, the user can choose to update the file (an RNA connection, or a direct connection to the original file must be available). The user is presented with information about which files(s) have changed. This is all based on the file's date. If only one file changed, the system can easily allow the user to update. Hooks are provided for an application to be notified that both files have changed. In this case, the application is invoked to merge the files.

This is apparently the machanism that the mail and scheduler applications will use to merge remote mailboxes/calendars with the network versions.

There appear to be a number of issues that have not yet been decided. For instance,

the OS may or may not provide some kind of notification that a particular file has a twin in a briefcase. There are also issues related to file date/time when one travels across timezones (or simply resets the date on their laptop).

The Pen in Chicago
Michael Van Kleeck
Group Program Manager, Mobile Services
It was emphasized that applications should deal with ink whenever possible. Recognition should only be performed if really necessary.

Microsoft is planning 'fat' and 'skinny' versions of pen support. Every box will have the skinny version which will allow a user to read mail messages they receive that contain ink. They have put effort into optimizing real time inking. New support for recognizers has been added. There is also greater integration with USER, including pen support in standard controls and DefWindowProc extensions. The Chicago shell will be pen aware.

Support for the JOT standard will be provided so that ink files can be shared with other pen systems (Apple's Newton for instance).

The targeting of ink is now much more intelligent. This means that you don't have to first put the focus where you want the ink to go.

The boxed edit control has been enhanced and the hedit control has been moved to USER. There is an ink edit control (that is an OLE server) that can be used as a transparency over other controls thus providing ink input into a document for instance. Other controls (listbox, radio buttons, checkboxes) have been 'pen-enabled'.

The basic gesture set has been reduced and simplified. Some gestures like selection and deletion are based on the size of the gesture (e.g., a word can be selected by circling it just as a paragraph can be selected by circling it).

Chicago Professional Developers Kit
Teri Schiele
Program Manager, Chicago Development Kits
Chicago will follow the Win31 beta model. Meaning there won't be a $69 Chicago beta disk like there was for NT. It also means that Chicago will be under non-disclosure until it is released.

Support for the Chicago beta will be through private compuserve forums (WINBTU, WINDEV, and WINDDK).

Microsoft is currently recommending that Chicago development be done on NT. Of course if you are using any of the Win32 enhancements or shell extensions that don't exist on NT you won't be able to run you application on NT.

Command line versions of the tools are available for Chicago. Unfortunately, you can't today build and run an MFC application on Chicago. There were a number of attendees who were really ticked off about this. Especially since Microsoft has been claiming that the best way to stay compatible with future releases (Chicago and Cairo) is to use Visual C++ and MFC. This issue caused enough of a stir that an additional breakout session was added to address the problem. MS will likely make a new version of MFC available on compuserve that will work on Chicago before the M5 release of Chicago in November.

Interesting note: the beta version of the Chicago DDK ships with SoftICE.

Supposedly, the Chicago development team has been self hosted since Sept. 1992.

The following things will definitely change before Chicago ships: long file name internals; the user interface; the UI style guide; support for remaining APIs; Tools.

The input syntax to the resource compiler is the same as for Windows 3.x.

Windows NT and Cairo
Mark Ryland
Senior Program Manager, Advanced Systems
A fairly long presentation on NT. Stated that NT met all of Microsoft's objectives (isn't selective memory a wonderful thing :-)). Microsoft is saying that NT will outsell all UNIXs combined within 1 -2 years. Mark emphasized robustness as justification for NTs large rescurce requirements and poor performance. When queried, the majority of the audience indicated that they are using NT. Mark indicated that the next release of NT would be 1Q94, contradicting what was said earlier about it being 2Q94.

It was interesting that Paul Maritz presented NT as Microsoft's high end OS that will always be pushing the envelope with hardware and technologies. These technologies will migrate down into the mainstream Windows platform as the entry level machine increases in capabilities. Mark Ryland's message was that NT **will be** the mainstream Microsoft OS.

Cairo will be the next generation of Windows NT. You can think of it as NT 2.0. Cairo's relationship to Chicago is similar to NT's relationship to windows 3.1.

A significant feature discussed often is the ability to do content queries across the net without direct knowledge of the net. The queries will be fast and will return information in a table format. This sort of functionality is made possible in part by the distributed OFS (object file system).

"Bottom line: Explorer displays results of multi-gigabyte multi-machine multi-volume query within seconds"

The Cairo user interface and shell are very extensible. These extensions are enabled

through OLE. OFS provides a native implementation of the OLE structured storage (structured storage is the new term for docfiles). OFS maintains an 8.3 name space side by side with the long file name space. Through OLE it will be possible to associate code with a folder to create "smart folders".

There will be more information presented on Cairo at the December PDC. The first beta of Cairo is planned for 2Q94.

How to be a Great App in the Chicago Shell
Joe Belfiore
Program Manager, Chicago Shell/UI

Rich
I did not attend this breakout session. The following information is from the slides.

The guidelines presented are focused toward the UI designer rather than the application architect.

1) Support Long Filenames

2) Support UNC pathnames
The use of UNC paths will enable direct network browsing from the shell.

3) Make sure document and data files are accurately displayed and used in the shell
Need 32x32 and 16x16 document icons. Need to make sure proper registration is performed so that proper icons are shown in shell. Support "print-to" in registration database to enable drag/drop printing to specific printers in shell.

4) Support OLE 2.0 drag & drop and other transfers consistently and extensively
Enables data to be moved to/from desktop, tray, explorer, and other applications. Support non-default drag with mouse button 2. Change menu based transfer model to that of the shell (e.g., paste becomes put <object> here).

5) Use the common dialogs, especially file open
This presents a problem for us. The new common dialogs display links, provide direct browsing of the network, and support shell operations like drag and drop. We have functionality not provided by the common dialogs, but we need the functionality that is provided by the common dialogs.

6) Be careful about multiple instances of app being started too easily at the same time
Restore windows that are open rather than creating additional windows on the same document. Put up a list of currently open app instances if a user double clicks app icon a second time. Follow the style guide for details.

7) Maintain a consistent user interface and object paradigm between application and the shell.

Use quick menus on button 2. Use property sheets. Use wastebasket api.

8) Extend the shell's ability to provide general information about our files (more OLE2.0)
Use the OLE 2.0 general properties like author, thumbnail, etc. so they will be displayed by the shell.

9) Support pen input for pen notebooks and desktop tablets
Use the pen apis to activate pen input. Use ink-edit controls to allow users to scribble notes in app. Enable ink annotations using the OLE 2.0 annotation server. Add natural pen gestures.

10) Support Chicago-style help
Context-sensitive help popups in dialogs ("quick help"). Add help item to quick menus. Revise help menu according to style guide.

11) Miscellaneous
Handle size and color scaleability. Avoid drawing your own title bar. Don't draw your own minimized window. Follow guidelines for installing application.

New Controls, Dialogs, and Shell Integration
Joe Belfiore
Program Manager, Chicago Shell/UI

Rich
I did not attend this breakout session. The following information is from the slides.

New controls:
- toolbar
- status bar
- column heading
- slider
- spin buttons
- progress indicator
- tabs
- property sheet
- list view
- tree view
- rich text control

Common Dialogs:
- file open and save as
- print
- print setup (choose printer)
- page setup
- find & replace

font
color
OLE 2.0 dialogs
    insert object
    change icon
    put here as (was paste special)
    convert
    links (link properties)

Chicago UI Design and UI Guidelines
Tandy Trower
UI Design Guide

Rich
I did not attend this breakout session. The slides presented were the same as the Chicago UI design review. Please see the trip report from that visit for more information.

USER Enhancements For Chicago
George Moore
Program Manager, Chicago Core
There are quite a few new USER features in Chicago. They fall into the following categories: increased system capacity; robustness additions; asynchronous input model; Win32 API support; new Win32 APIs to support new USER features; scaleable window metrics; and hardware change notifications.

Increased System Capacity
Each task can now have up to 32K of menu and window handles.
The number of timers is limited only by available memory.
The number of COM and LPT ports is not limited.
Listboxes can hold up to 32K items with data limited by available memory.

Robustness Additions
More robust parameter validation.
Each 32 or 16-bit process or thread ID is used for object ownership or Ring 3 objects (allowing USER to clean up after tasks).

Asynchronous input model
"Quite compatible with NT".
Each thread can have its own message queue. The queue is created by the system when it's needed. It's size is dynamic.

Win32 USER api support
Debugging:
    SetDebugErrorLevel api
Dialog Boxes:

MessageBoxEx allows one resource to have multiple language strings
MessageBoxIndirect (includes help ID #)

Icons:
CreateIconFromResource
CreateIconIndirect
GetIconInfo
LookupIconIDFromDirectory

Keyboard Accelerators:
CopyAcceleratorTable
CreateAcceleratorTable
DestroyAcceleratorTable

KeyboardInput:
ActivateKeyboardLayout (unlike NT, Chicago can have one keyboard active at a time)
GetKeyboardLayoutName
LoadKeyboardLayout
UnloadKeyboardLayout

Window Properties:
EnumPropsEx - enumerates one at a time

Messages and Message Queues:
PostThreadMessage
SendMessageCallback
SendMessageTimeout
SendNotifyMessage

Multiple Document Interface:
CreateMDIWidow

Painting and Drawing:
WindowFromDC

Processes and Threads:
AttachThreadInput
WaitForInputIdle

String Manipulation:
FormatMessage

Synchronization:
MsgWaitForMultipleObjects

System Information:
GetThreadDesktop
SetThreadDesktop

Windows:
GetForegroundWindow
SetForegroundWindow
Mini-caption windows using WS_EX_SMCAPTION
New minimized window look
Enhanced control capabilities (bitmap pushbuttons, thumbsize of scrollbars, etc.)
APIs for drawing 3-D windows and frame controls
Enhanced support for context sensitive help and message boxes

Enhanced menu support (default items, radio items, bitmap/icon items and extended popup menus)
Dialog manager support for multiple font styles in RC template
ChildWindowFromPointEx (can skip invisible/transparent child windows)
DrawCaption
DrawEdge
DrawFrameControl
DrawAnimatedRects
GetScrollInfo

Menus:

AppendMenuItem
GetMenuItemInfo
InsertMenuItem
SetMenuItemInfo
There is a new MENUITEMINFO structure that stores, among other things, checked and unchecked bitmaps
FindMenuDefaultID
GetMenuDefaultItem
SetMenuDefaultItem
TrackPopupMenu
TrackPopupMenuEx

Fonts and Text:

DrawTextEx (adds new margin controls, tab sizes and ellipses if text won't fit)

Applications/driver notification:

BroadcastSystemMessage (can be used for VxD notification)

Scaleable Window Metrics

All USER visual components are scaleable (captions size and font, menu size and font, frame buttons, and scroll bars)

All are accessible via SystemParametersInfo API

All USER visual components have a new 3D look (VGA is minimum resolution supported by Chicago)

Use the new COLOR_3D colors are returned from GetSysColor API
Don't hard code button colors to grey
Old BTN_FACE, BTN_HIGHLIGHT and BTN_SHADOW are mapped to new structures

Hardware Change Notification

Support for dynamic screen resolution changes

Standardized mechanism for tilt screen devices and hot-docking stations
WM_DISPLAYCHANGED message is sent twice: one before and one after the resolution change. wParam contains change flag. lParam contains old/new resolution values.

General purpose PnP change notification
New DBT_DEVNODES_CHANGED message is sent after the new configuration is valid.
lParam points to the root devnode of the hardware tree.
Uses new BroadcastSystemMessage API.

Kernel Architecture
Jon Thomason
Development Lead, Chicago Kernel/Win32

16-bit Tasking
Cooperative tasking, same as Windows 3.1
Each Win16 app runs as a thread, providing for resource tracking in Ring0 (VxDs) and Ring3 (GDI & USER)
Synchronization between apps occurs via messages

32-bit Tasking
All 32-bit applications and threads are fully preemptive
Any thread can call any API
Fault handler on a separate thread for robustness
Compatible with Windows NT model

The Scheduler
A completely new scheduler is being written right now. It may be in M5, but they didn't talk like that was for sure.

It will be compatible with the Windows NT model: 32 priority levels supported.
It will also be compatible with existing Win16 apps, VDDs and VxDs.
It will provide dynamic priority boosting with timed decay as well as priority inversion boosting.

Thunking
The thunking layer has been completely rewritten (it is NOT using the Win32s code). All thunks live in 32bit land. Each thunk is performed with 7 selector loads (at 9 clocks/selector load) - very fast! This code will be available in M5 (Nov. beta).

Memory Maps
"Don't depend on memory map!"

The minimum virtual alloc size is 4K (on NT it is 64K).

Chicago is different than Win32s which is different than NT.

In Chicago, shared DLLs will reuse pages if the addresses are available in each app. If not, the code will be reloaded for each app. We will want to make sure we address this issue with ALL WP apps and the shared code so that the shared code can truly be

shared.

Key differences between Chicago and Win32s:
- Private address space for all 32-bit applications
- 4MB load line - apps that request below this will have fixups performed (NOT GOOD FOR PERFORMANCE)

Key differences between Windows NT and Chicago:
- No shared address space
- Memory mapped files are not necessarily visible at the same address in all application's spaces

Chicago Extensible Thunks
The universal thunk supported in Win32s will not be supported in Chicago.
Chicago's internal kernel thunking APIs will be documented in a future release.
Very similar to Windows NT generic thunks.
16/32bit code mixing is NOT recommended. These thunks are provided only for special cases

Windows User Assistance
Marion Hogan, Gayle Picken and Ralph Walden
Chicago User Education
Based on the slides, it appears that nothing new was discussed beyond what was discussed at the Chicago UI design preview. Please refer to that trip report for more information on changes to the help system.

I would like to recommend that we adopt "Quick Help" as the menu item on our "Quick menus" for getting context sensitive help. Office currently has "Help" on most (if not all) quick menus.

Multilingual Windows
Tom Grate
Program Manager

Rich
I did not attend this breakout session. The following information is from the slides.

Chicago will support three platforms:
1) Americas, Western and Eastern Europe, Greece, and Turkey
2) Middle East
3) Far East

After installation, Chicago will have one fixed language for UI elements and help files.
It will have multiple languages/character sets for fonts and keyboards.

A multilingual goal of Chicago is to support document portability.

Exchange documents with any other system of the same Chicago platform.
Enter, display, and manipulate data in different character sets.
Quickly change keyboard layout and fonts.

They also want to be able to add on to an installed system everything needed for a new set of languages in a document.

Apparently, they are planning something called a Language Pack (LPK), the contents of which have not been decided. One slide said the LPK might contain: keyboard layouts; input methods; fonts; sorting tables; link break / word break info; normalizer; and writing system specific layout.

There will be system hooks and notifications for keyboard and character set changes.

National Language Support in Chicago
Steve Reddoch
Program Manager

Rich
I did not attend this breakout session. The following information is from the slides.

Chicago will provide NLS APIs that are identical to those provided by Windows NT.

After reading through the slides, my guess is that this support is not something that our applications will be using. If that is not true, or if more information is needed, I would suggest looking at the Windows NT SDK.

File System APIs
Russ Arun
Program Manager, Chicago Base
The Chicago FAT file system can maintain a "last accessed date" that is stored in a reserved entry. This support is not turned on by default. If a utility clears the entry, the file will be obliterated. There is a new file open api that does not set the last accessed date (useful for utility programs).

Russ expressed surprise that they have not heard from ISVs or customers regarding problems with long file name support that exist in NT and in Chicago. Some operations on long file names will change the 8.3 name of the file (e.g., copying LongFileName to a directory that contains LongFileName2 can cause 8.3 name of LONGFI~2 to change to LONGFI~1).

The 8.3 autogeneration algorithm cannot be changed by an application.

Some operations on 8.3 file names will lose the long file name.

Searches apply to both LFN and 8.3 name spaces (e.g., find first returns either LFN or

8.3 match, this means that the user may see a long file name that doesn't match the search because the 8.3 name did match the search).

Down level systems only see the 8.3 name.

Reserved directory entries are used to store the long file names. It is possible that existing DOS utilities might think that the disk is corrupt.

Valid characters in 8.3 names have not changed. Valid characters in LFNs are the same as Win32. Russ said that they are still debating storing the LFNs as ANSI or UNICODE.

There is a new GetVolumeInformation API that may be used for local drives. It gives lots of interesting information without accessing the disk. Some of the information is max filename length, max pathname length, file system name, and whether the volume is compressed or not.

Russ said that they are planning a Canonical API for comparing filepaths (there are 2**n possibilities for a path with n components since each component can be an LFN or an 8.3).