UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

RPX CORPORATION
Petitioner

v.

CEDATECH HOLDINGS, LLC
Patent Owner

INTER PARTES REVIEW OF U.S. PATENT NO. 7,707,591
Case IPR No.: *To Be Assigned*

DECLARATION OF MARK E. CROVELLA, PH.D.

TABLE OF CONTENTS

I, Mark E. Crovella, Ph.D., declare:

1. I have been retained by Wolf, Greenfield & Sacks, P.C., counsel for Petitioner RPX Corporation ("RPX"), to assess U.S. Patent No. 7,707,591 ("the '591 patent). I am being compensated for my time at a rate of $450 per hour, plus actual expenses. My compensation is not dependent in any way upon the outcome of RPX's Petition.

I. PERSONAL AND PROFESSIONAL BACKGROUND

2. I am Professor and Chair of the Department of Computer Science at Boston University. I received an undergraduate degree in Biology from Cornell University in 1982. I received a master's degree in Computer Science from the University of Buffalo in 1989. I received a Ph.D. in Computer Science from the University of Rochester in 1994. The subject of my Ph.D. thesis was "Performance Prediction and Tuning of Parallel Programs." While working toward my Ph.D., I also published several papers concerning operating systems, including R. Bianchini,M. E. Crovella, L. Kontothanassis, and T. J. LeBlanc, "Memory Contention in Scalable Cache-Coherent Multiprocessors," Technical Report 448, U. Rochester Computer Science Department, April 1993; and T. J. LeBlanc, M. L. Scott, B. Marsh, E. Markatos, C. Dubnicki, M. Crovella, and T. Becker, "The Psyche Parallel Operating System," *IEEE TC on Operating Systems Newsletter,* vol. 6, no. 1, 1992, pp. 11-13.

3. From 1982 to 1984 I worked as a computer programmer for the State of Colorado. From 1984 to 1994 I was employed at Calspan Corporation, a research and development firm in Buffalo, NY, where I rose to the level of Senior Computer Scientist. My work at Calspan focused on development of experimental software and large-scale simulation software in support of contracts between Calspan and the U.S. Department of Defense.

4. In 1994, I joined the faculty of Boston University as an Assistant Professor of Computer Science. I was promoted to the rank of Associate Professor in 2000 and became a full Professor in 2006. Since 2013, I have served as Chair of the Department of Computer Science.

5. I am well versed in Microsoft Windows architecture. For example, while working in 1998 and 1999 as co-founder and principal technical officer of Commonwealth Network Technologies, Inc., I co-developed technology that allowed for distributed routing in Microsoft Windows NT and enabled information to be served efficiently from multiple servers to multiple clients. Further, I have taught multiple courses concerning operating systems and operating system architecture.

6. My detailed employment background, professional experience, and list of technical papers and books are contained in my CV. (Ex. 1027).

7. I am familiar with the subject matter of the ‘591 patent, which concerns “integrating [*an*] audio or video program with an application program” (1:6-10). I consider myself to be an expert in the field of computer programming as relevant to the ’591 patent, including the programming technique of “subclassing” that I discuss in detail below. All of the ’591 patent claims require systems that use subclassing to integrate the audio or video program with the application program.

II. MATERIALS REVIEWED AND CONSIDERED

8. In connection with my work on this matter, I have reviewed the ’591 patent (Ex. 1001) as well as the other documents listed on the following list:

EXHIBIT	DESCRIPTION
1001	U.S. Patent No. 7,707,591 (“the ’591 patent”)
1003	U.S. Pat. App. Pub. No. 2002/0090062 (filed Nov. 16, 2001) (“Alhadad”)
1004	U.S. Pat. No. 6,219,047 (filed Sept. 17, 1998, issued Apr. 17, 2001) (“Bell”)
1005	U.S. Pat. No. 5,619,639 (issued Apr. 8, 1997) (“Mast”)
1006	WIPO Pat. App. 2000/68769 (published Nov. 16, 2000) (“Lui”)
1007	Complete File History of U.S. Patent App. No. Appl. No.: 09/683,706 (issued on Apr. 27, 2010 as the ’591 patent)
1008	Steven Roman, Win32 API Programming with Visual Basic (O’Reilly Jan. 2000) (“Roman”)
1009	Dan Appleman, Visual Basic® Programmer’s Guide to Win32 API (Sams 1999) (“Appleman 1999”)
1010	Stephen Teilhet, Subclassing & Hooking with Visual Basic (O’Reilly June 2001) (“Teilhet”)

1011	Jeffrey Richter, Programming Applications for Microsoft Windows (Microsoft Press 4th ed. 1999) ("Richter")
1012	Charles Petzold, Programming Windows (Microsoft Press 5th ed. 1999) ("Petzold")
1013	Dan Appleman, Visual Basic Programmer's Guide to the Windows API (Ziff-Davis 1993) ("Appleman 1994")
1014	Microsoft Computer Dictionary (Microsoft Press 4th ed. 1999)
1015	IBM Dictionary of Computing (McGraw-Hill 1994)
1016	U.S. Pat. No. 6,874,139 (filed May 15, 2001) ("Krueger")
1017	U.S. Pat. No. 6,002,402 (issued Dec. 14, 1999) ("Schacher")
1018	U.S. Pat. No. 6,971,067 (filed Aug. 23, 2000) ("Karson")
1019	Microsoft Support, Differences Between Outlook and Outlook Express, http://support.microsoft.com/kb/257824
1020	MSDN Library, System Requirements for MS Office, https://msdn.microsoft.com/en-us/library/cc749816.aspx
1021	Microsoft, A History of Windows, http://windows.microsoft.com/en-us/windows/history
1022	SubApp.exe Shows How to Append Menu Items to Other Apps, Microsoft Windows Software Development Kit (Dec. 24, 1999), *available at* ftp://ftp.microsoft.com/misc1/DEVELOPR/WIN_DK/KB/Q72/2/22.TXT
1023	INFO: Context-Sensitive Help in a Dialog Box Through F1, Microsoft Windows Software Development Kit (June 10, 1999), *available at* ftp://ftp.microsoft.com/misc1/DEVELOPR/WIN_DK/KB/Q72/2/19.TXT
1024	Microsoft Windows NetMeeting, https://web.archive.org/web/19991013110348/http://microsoft.com/windows/netmeeting/ (archived Oct. 13, 1999)
1025	Intel Email Postcard, https://web.archive.org/web/20000903014913/http://www.intel.com/PCcamera/software/send.htm (archived Sept. 3, 2000)
1026	Picante Mail, https://web.archive.org/web/19981206221024/http://www.picantecorp.com/p-mail-f.html (archived Dec. 6, 1998)

III. LEVEL OF ORDINARY SKILL IN THE ART

9. For purposes of assessing whether prior art references disclose every element of a patent claim (thus "anticipating" the claim) and/or would have rendered the claimed invention obvious, I understand that such prior art references must be assessed from the perspective of a person having ordinary skill in the art ("POSA") to which the patent is related, based on the understanding of that person at the time of the invention date. I understand that a POSA is presumed to be aware of all pertinent prior art and the conventional wisdom in the art, and is a person having ordinary creativity. I have applied this standard throughout my declaration.

10. I have been asked to provide my opinion as to the state of the art in the field of computer programming in the 2001 to 2002 timeframe. Whenever I offer an opinion below about the knowledge of a POSA, the manner in which a POSA would have understood the prior art, or what a POSA would have been led to do based on the prior art, I am referencing this timeframe (i.e., 2001 to 2002). The topics on which I opine in this declaration were well-known and did not change in any significant way in the 2001-02 timeframe. In my opinion, a POSA related to the '591 patent in that timeframe would have had at least a B.S. in Computer Science or the equivalent, along with at least one to two years of computer programming experience including experience with programming on

window-based operating systems ("OS") such as Microsoft Windows ("Window OS"). This person would have been capable of understanding and applying the prior art references discussed herein.

11. By 2001, I was an Associate Professor of Computer Science and I had many years of computer programming experience on window-based operating systems, including Microsoft Windows. Therefore, I was a person of more than ordinary skill in the art during the relevant time period. However, when developing the opinions set forth below, I assumed the perspective of a person having ordinary skill in the art, as set forth above.

IV. THE BASICS OF SUBCLASSING ON WINDOWS OS

12. My opinion discusses the Windows OS operating systems and integrating two programs running on Windows OS. This is relevant to my opinion for several reasons:

a. First, the '591 patent specifically identifies "versions of the Microsoft Windows OS" (Ex. 1001, 3:12-17) as appropriate operating systems on which to implement its purported invention of "integrating" two programs (*Id.* at 10:1-3).

b. Second, the '591 patent's only example of subclassing—a programming technique required by all the claims as a means to integrate two programs—is specific to Windows OS. (*See* Ex. 1001,

6:21-63 (illustrating subclassing with Microsoft Outlook Express, version 5); Ex. 1019 (explaining that Outlook Express was only ever available for Windows and Mac OS); Ex. 1020 (explaining that version 5 of Outlook Express was never available for Mac OS).

c. Third, the programming techniques of subclassing and hooking, as described in the '591 patent, were commonly and primarily associated with Windows OS and were known as appropriate methods to integrate two programs. (Ex. 1008, Roman at 317-39; Ex. 1010, Teilhet at 2-79; Ex. 1009, Appleman 1999 at 37-31; Ex. 1011, Richter at 752-58; Ex. 1012, Petzold at 393).

d. Finally, all four of the prior art references discussed in Section VI below disclose Windows OS as a suitable operating system for their inventions.

13. Parts IV.A though IV.E below review basic concepts of computer programming on Windows OS, including how two programs can be integrated using the programming technique of subclassing and the complementary technique of hooking. To a POSA, as defined in Section III above, these concepts were well-known and would have been well within their capabilities to implement.

A. Graphic User Interfaces and Audio/Video Email

14. Operating systems with graphical user interfaces ("GUI") have been widely used at least since Apple first introduced the Macintosh in 1984.[1] Microsoft released the first version of Windows OS in 1985, which also included a GUI.[2] By the time of the purported invention in the '591 patent, programs for sending audio and/or video messages over the Internet and that used GUIs were commonplace. For example, the face of the '591 patent lists numerous examples:

- Microsoft Windows NetMeeting (Ex. 1024) ("NetMeeting delivers … point-to-point audio and video.") (video conferencing GUI interface right);
- Intel Email Postcard (Ex. 1025) ("Want to use email for more than just messages? … You can add your own video, snapshots, text and sound to fun email postcards.");
- Picante Mail (Ex. 1026) ("Picante Mail transforms text-based e-mail into an eye-catching, multimedia-rich communication tool. … Incorporate bitmaps, video, sound and audio clips….");

(*See* Ex. 1001 at [56]).

[1] *See* Macintosh Version History, http://support.apple.com/kb/TA31885?viewlocale=en_US.

[2] *See* Windows Version History, http://support.microsoft.com/kb/32905.

15. As these examples make clear, the concept of incorporating audio or video into computer-based messaging was well known before the '591 patent's February 2002 filing date. For example, a POSA would have been able to write programs to send audio or video messages over the Internet. The '591 patent is directed to the notion of integrating an audio or video program together with an application program. (Ex. 1001, 1:47-55). The '591 patent touts the purported challenge of using a video camera to record video and then incorporating it into an email. (*Id.* at 1:29-43). I disagree with this premise, which is contradicted by the above examples. In any event, this purported invention "story" is not congruent with the'591 patent claims, none of which require that the audio or video program be capable of recording video. (*Id.* at 10:1-42). Nor do any of the '591 patent's claims limit the application program to email programs. (*Id.*). Instead, the claims relate generally to integrating an audio or video program—even one that is only allows for playing back existing audio or video—with an application program. (*Id.*; *see also* Section V below).

B. Hierarchy of the Windows OS

16. A general understanding of how Windows OS is organized is helpful to understand how two programs running on Windows OS can be integrated together. All of this information would have been well known to a POSA in the 2001 to 2002 timeframe. Windows OS was available in a variety of versions

during the 1990s, including Windows 3.1 (released in 1992), Windows NT (released in 1993), Windows 95 (released in 1995), and Windows 98 (released in 1998). (Ex. 1021). A POSA would have understood each of these versions of Windows OS to be an operating system.

17. The claims of the '591 patent require using a programming technique called subclassing (10:19-23)—discussed further in Sections IV.D and IV.E below—to integrate an audio or video program with a separate application program. Subclassing relies on the Windows OS Application Programming Interface ("API"). The Windows OS API is a set of functions that a computer programmer may utilize to access features of the operating system. (*See* Ex. 1009, Appleman 1999 at 8; Ex. 1008, Roman at 4). Each version of Windows OS noted above provided computer programmers at least one API: Windows 3.x (e.g., version 3.1) provided the Win16 API and Windows NT, 95, and 98 provided the Win32 API.[3] (Ex. 1009, Appleman 1999 at 11-12).

[3] Microsoft Windows NT, 95, and 98 also provided the Win16 API and Microsoft Windows 3.x provided an incomplete version of the Win32 API called Win32s. (Ex. 1009, Appleman 1999 at 11-12). Technically, Microsoft Windows 95 and 98 provided a version of the Win32 API called Win32c, but is functionally identical to Win32 for the purposes of my opinion herein. (Ex. 1008, Roman at 6-7; Ex. 1009, Appleman 1999 at 12; Ex. 1010, Teilhet at xii).

18. A computer programmer uses the functions available in the APIs, along with other instructions written in a programming language such as Visual Basic or C++ (pronounced "See Plus Plus"), to write programs. Programs include sequences of instructions that the computer can execute. (Ex. 1014, Microsoft Computer Dictionary at 359; Ex. 1015, IBM Dictionary of Computing at 535). In general parlance, an application program, or application for short, is simply a program designed to assist a user with a specific task. (Ex. 1014, Microsoft Computer Dictionary at 26-27; Ex. 1015, IBM Dictionary of Computing at 28). With versions of the Windows OS, programs are stored on a computer as executable files and often have names that end in ".EXE". (Ex. 1014, Microsoft Computer Dictionary at 173).

19. When discussing programming on Windows OS, the term 'application' refers to a program and other related files (e.g., images that the application could use in its GUI). (Ex. 1008, Roman at 141). An application running on Windows OS is known as a process. (Ex. 1008, Roman at 141; Ex. 1010, Teilhet at 27). A process will have one or more threads, which are instances of executing computer code from the program. (Ex. 1008, Roman at 141-42; Ex. 1010, Teilhet at 27).

20. As a POSA would have recognized, there is some fluidity in how these terms can be used. The context in which these terms are used is the best

source of their meaning. Although Windows OS programming reference books sometimes distinguish between an application, process, program, and thread, the '591 patent uses the terms "program" or "application program" to refer to them all together.

21. A program is not limited to using the instructions stored in its own files. Rather, as a program runs on a computer system, it can call up instructions stored in central libraries that are available to any other programs. (Ex. 1009, Appleman 1999 at 6-7; Ex. 1008, Roman at 15-18; Ex. 1012, Petzold at 8-9, 1243-44; Ex. 1013, Appleman 1994 at 6-7). Since the links between the program and the external libraries may be established at the time that the program calls for the external instructions, these libraries are called Dynamic-Link Libraries, or "DLLs" for short. (Ex. 1009, Appleman 1999 at 7; Ex. 1013, Appleman 1994 at 6-7). Among the benefits of DLLs is that they may be used to facilitate integration of two programs, as discussed in Section IV.D and IV.E below.

22. Every GUI-based program running on Windows OS is associated with at least one GUI window element. (Ex. 1008, Roman at 263). When programming on Windows OS, the term "window" refers to many more aspects of the GUI window element than a lay person might typically understand. A lay person might understand a window to be the top-level GUI element that appears as a floating rectangle on the screen in which a program appears to be run. While that is indeed

one type of window in the technical sense, each such top-level window typically includes various other GUI elements such as buttons, scrollbars, and other controls—all of which constitute separate windows of their own. (Ex. 1009, Appleman 1999 at 20; Ex. 1010, Teilhet at 27; Ex. 1001, 6:12-20; Ex. 1012, Petzold at 42). Each control window is a child to another window (i.e., a parent window, such as a top-level window), which means that it is overlaid on and attached to the other window. (Ex. 1008, Roman at 264-65; Ex. 1009, Appleman 1999 at 24). To avoid confusion, I use "Windows" or "Windows OS" with a capital "W" to refer to the OS, and I use a lower case 'window' or 'windows' to refer to GUI elements, including both top-level windows and control windows.

23. My approach is consistent with the '591 patent, which explains that "window" includes "top-level" windows along with "any child windows," including "dialog boxes, message boxes, and controls." (Ex. 1001, 6:12-15). The patent notes that this meaning of "window" is "broader . . . than the conventional use of the term." (Ex. 1001, 6:15-18). This is true from the perspective of a layperson. To a POSA, the '591 patent's usage would have been consistent with the typical technical understanding of "window."

24. Each type of window belongs to a particular 'window class.' (Ex. 1008, Roman at 305; Ex. 1010, Teilhet at 33; Ex. 1009, Appleman 1999 at 20-21). For example, "Button" and "Scrollbar" are window classes that Windows OS

provides. (Ex. 1008, Roman at 305; Ex. 1010, Teilhet at 33-34; Ex. 1009, Appleman 1999 at 21-22). A computer programmer can create new window classes, which can be based on existing window classes. (Ex. 1008, Roman at 305-06, 318; Ex. 1010, Teilhet at 222-23). A window class based on an existing window class inherits the properties of the existing window class, creating a hierarchy of related window classes.[4] (Ex. 1008, Roman at 318; Ex. 1010, Teilhet at 222-23). Each window class is associated with a set of computer instructions called a window procedure, which defines how the windows in the class behave in response to user interactions or Windows OS commands. (Ex. 1008, Roman at 306-07; Ex. 1012, Petzold at 42-43, 62).

25. Windows OS creates a window based on a window class when a program uses a Window OS API function (`CreateWindow`). (Ex. 1012, Petzold at 59 ("After the CreateWindow call returns, the window has been created internally in Windows."); Ex. 1008, Roman at 308-09). Windows OS assigns each window a unique name (Ex. 1008, Roman at 308-09; Ex. 1012, Petzold at 59),

[4] Since a new window class based on an existing window class inherits properties of the existing window class, the new window class could be described as a subclass of the existing window class. However, this concept is distinct from the Windows OS-specific programming technique claimed in the '591 patent and discussed in Sections IV.D and E below.

which, as discussed in the next Section, is needed for Windows OS to communicate with the windows.

26. The '591 patent itself emphasizes that "[*a*] window has a window procedure that can receive messages and act on them." (Ex.1001, 6:19-19).

27. Consistent with these disclosures, a POSA would have understood the term "window" to mean a GUI element having a window procedure that can receive messages and act on them. (Ex. 1001, 6:17-20).

C. Messaging on Windows OS

28. As the '591 patent explains, the subclassing programming technique "operates on … the messaging system" of Windows OS (Ex. 1001, 6:1-2). Understanding how subclassing works thus requires knowledge of messaging on Windows OS. Windows OS communicates with windows using messages, which are small pieces of information that inform a window (and therefore the associated program) about some event. (Ex. 1008, Roman at 281-82; Ex. 1010, Teilhet at 3; Ex. 1012, Petzold at 42-43, 61). For example, when a user moves the computer's mouse or hits a key on the keyboard, Windows OS sends messages to the active program indicating that the mouse moved or the key was hit. (Ex. 1008, Roman at 281-82; Ex. 1010, Teilhet at 3; Ex. 1012, Petzold at 43). In addition, programs can send messages to each other, which allows inter-program communication. (Ex. 1008, Roman at 281-82; Ex. 1010, Teilhet at 3).

29. Each messages identifies a particular window as its destination, and Windows OS operates like a the postmaster delivering the messages to the program associated with that window. (Ex. 1008, Roman at 281-82; Ex. 1010, Teilhet at 3). That way, when a user clicks a button (i.e., a window), for example, Windows OS sends the message to the program with the button informing the program that the user clicked that particular button, which is associated with a unique name. Because messages are sent to a window, a program must have a window in order to receive messages. (*See* Ex. 1011, Richter at 71; Ex. 1008, Roman at 281-82). Thus, the fact that a program running in Windows OS receives messages necessarily means that the program has a window and that a POSA would understand the program to "run" in that window per the technical understanding of "window" discussed in paragraph 27 above.

30. What users experience when running GUI-based programs on the Windows OS depends on how each window (e.g., particular buttons and controls) behaves in response to messages. As noted in Section IV.B above, each window's behavior is defined, at least in part, by its window procedure. This window procedure receives and processes the messages sent to the associated window. (Ex. 1012, Petzold at 42-43; Ex. 1008, Roman at 282, 307-08; Ex. 1010, Teilhet at 35-36). For example, when a user resizes a window (e.g., a top-level floating window), Windows OS sends the window a message informing it of the change of

size, and the window's window procedure reviews the message and responds by, for example, making adjustments to the contents of the window. (Ex. 1012, Petzold at 42-43; Ex. 1008, Roman at 307-08; Ex. 1010, Teilhet at 36).

D. Subclassing and Hooking in Windows OS

31. The programming technique of subclassing changes the Windows messaging system. As a POSA would have understood, and as the '591 patent explains, subclassing (1) intercepts a messages before the destination window's normal window procedure can process the message, then (2) redirects the message to a new window procedure. (Ex. 1010, Teilhet at 4, 48; Ex. 1008, Roman at 317-18; Ex. 1001, 6:1-6). The new window procedure can inspect the intercepted message, then either discard it or send the message to the original window procedure. (Ex. 1010, Teilhet at 4; Ex. 1008, Roman at 318). The subclassing programming technique occurs as a program runs.

32. Subclassing was a well-known programming technique in the 2001 to 2002 timeframe and would have been straightforward for a POSA to implement. Numerous prior art programming reference books describe subclassing on Windows OS in detail. (*E.g.*, Ex. 1008, Roman at 317-21 ("Subclassing … is not difficult…."); Ex. 1010, Teilhet (entitled "Subclassing & Hooking"); Ex. 1009, Appleman 1999 at 27-33; Ex. 1012, Petzold at 393; Ex. 1011, Richter at 752-54 ("Let's say you want to subclass an instance of a window…."); Ex. 1013,

Appleman 1994 at 21). The prior art references discussed in Section VI below (i.e., Alhadad, Bell, Mast, and Lui) all use subclassing on Windows OS. (*E.g.*, Ex. 1003 ("[*S*]ubclassing is a well known programming tool….")). In addition, numerous other prior art patents and applications use subclassing, as described in Paragraph 42 below.

33. The programming technique of subclassing a window "is not difficult." (Ex. 1008, Roman at 319). To subclass a window, a computer programmer merely calls a Window OS API function (`SetWindowLong`), specifying the window to subclass and where to find the new window procedure.[5] (Ex. 1008, Roman at 317).

34. Subclassing is similar to another programming technique called hooking. The prior art references discussed in Section VI below use hooking alongside subclassing when integrating different programs together. This is consistent with the '591 patent, which refers to the use of hooking and subclassing. The claims require that the system use subclassing to integrate two programs but do not require that subclassing be the only programming technique for achieving

[5] Other forms of subclassing exist, but understanding those techniques is not necessary to understand the technology at issue in the Petition. (*See, e.g.*, Ex. 1008, Roman at 318-21 (discussing also subclassing a window class using the `SetClassLong` Windows OS API function)).

the claimed invention. Indeed, as discussed further below, hooking is often a useful complement to subclassing when integrating two programs that use the Win32 API included with versions of Microsoft Windows more recent than Microsoft Windows 3.1.

35. Much like subclassing, hooking allows a program to intercept messages bound for a window. (Ex. 1001, 6:7-11; Ex. 1008, Roman at 322; Ex. 1010, Teilhet at 67-68). Hooking also allows programs to "set a hook" and thus watch messages bound for windows associated with another program. (Ex. 1008, Roman at 322; Ex. 1010, Teilhet at 77).

36. Like subclassing, hooking was a well-known programming technique long before the '591 patent. A POSA would have been able to implement each technique alone or in tandem. Many of the same prior art programming reference books discussing subclassing also cover hooking. (*E.g.*, Ex. 1008, Roman at 322-39; Ex. 1010, Teilhet (entitled "Subclassing & Hooking"); Ex. 1011, Richter at 757-74). The prior art references (i.e., Alhadad, Bell, Mast, and Lui) discussed in Section VI below also all use hooking. In addition, numerous prior art patents and applications use hooking in substantially similar circumstances, as described in Paragraph 42 below.

37. Hooking messages is straightforward. To install a hook, a computer programmer calls a Windows OS API function (`SetWindowsHookEx`),

specifying what messages to hook and what procedure to process them (i.e., the hook procedure). (Ex. 1008, Roman at 324; Ex. 1010, Teilhet at 73; Ex. 1023).

38. When hooking messages bound for a window of another program that uses the Win32 API discussed in Paragraph 17 above, computer programmers must take an extra step. Windows 95 (the OS disclosed in most of the prior art references discussed in Section VI below) and other versions of Windows OS that support Win32 compartmentalize Win32 programs. As a result, the computer programmer must put the hook procedure somewhere that the other program can access, such as in a DLL as discussed in Paragraph 21 above. (Ex. 1008, Roman at 322-24; Ex. 1010, Teilhet at 77; Ex. 1011, Richter at 757-58).

E. Integrating Two Programs with Subclassing

39. The programming techniques described above illustrate that a POSA had several ways to integrate two programs using subclassing. The prior art references discussed in Section VI below all disclose using subclassing in conjunction with hooking to integrate two programs. Therefore, I have focused on how both techniques can be used.

40. On its own, the subclassing technique does not allow one Win32 program to subclass a window of another Win32 program. (Ex. 1010, Teilhet at

50-51; *see* Ex. 1011, Richter at 752).[6] However, as a POSA in 2002 would have understood, subclassing can accomplish this result when used in tandem with hooking. A computer programmer can first install a hook on the second program's window and also include the subclassing instructions and new window procedure in the same DLL as the hook procedure. (Ex. 1008, Roman at 340; Ex. 1011, Richter at 757).

41. In this manner, programmers can give one program control over a subclassed window associated with a second program. The new subclass procedure can inspect, alter, forward, and/or block messages directed to the subclassed window, as described in Paragraph 31 above. The new subclass procedure can even change the appearance and behavior of the subclassed window. (Ex. 1008, Roman at 319-21 (illustrating how the behavior of a checkbox can be altered with subclassing); Ex. 1010, Teilhet at 4 (noting how subclassing can modify the appearance and behavior of a control or other window); Ex. 1011, Richter at 752 ("[*S*]ubclassing allows you to alter the behavior of a window.")). Because the new subclass procedure can exchange messages with the first program, as described in Paragraph 28 above, the first program can control these

[6] The process is simpler in versions of Windows OS using Win16, since programs are not compartmentalized and a program can simply subclass the window of another program. (Ex. 1022).

changes to the subclassed window in the second program. The first program's control of the subclassed window facilitates the integration of the two programs.

42. The prior art as of 2002 included numerous examples of integrating two programs using subclassing. (*See, e.g.*, Ex. 1011, Richter at 752 ("Let's say that you want to subclass an instance of a window created by another process."); Ex. 1022). The references discussed in Section VI below all use subclassing to achieve such integration. As discussed in Section VI.A below, Alhadad integrates two programs used in call centers. (*E.g.*, Ex. 1003 ¶¶ 123-24 (discussing "hooks … implemented as a Windows dll … file" that determine whether a "Control is subclassed")). As discussed in Section VI.B below, Bell integrates a training agent with a host program. (*E.g.*, Ex. 1004, 6:24-50 ("The agent installs three hooks…. [*T*]he message hook subclasses all windows of the host….")). As discussed in Section VI.C below, Mast integrates an image attachment program into other programs. (*E.g.*, Ex. 1005 6:47-49 ("Attacher.DLL utilizes the shell hook function to decide if an application should be subclassed by the Attacher.DLL subclass procedure.")). As discussed in Section VI.D below, Lui integrates an assistant program with a windows-based application. (*E.g.*, Ex. 1006, 56:20-21 (describing a "Hooking Function, which subclasses windows")). Various other prior art references also illustrate integrating two processes using subclassing:

a. Krueger (Ex. 1016) explains how a program can add file handling functionality to an existing program, such as by adding automated encryption and decryption, with subclassing and hooking. (Ex. 1016, 2:45-50; *id.* at 7:39-66 (disclosing a system that uses a "a system-wide window hook" with an "associated … DLL" that contains a function to perform "sub-classing")).

b. Schacher (Ex. 1017) discloses a program that integrates other programs using subclassing and hooking to convert certain menu items into draggable shortcuts to the menu item (e.g., a user can drag a file listed in a History menu to the Desktop for quick access). (Ex. 1017, 2:23-31; *id.* at 8:44-9:22 (explaining that the system listens for messages indicating a new program has been opened using hooking, and monitors each new program's menus using subclassing); *id.* at 9:4-5 ("Such 'procedure subclassing' is known to those skilled in the art…."); *id.* at 8:48-51 ("Events hooking within GUI environments such as the Windows 3.1™ and Windows 95™ environment is a procedure well known to those skilled in the art….").

c. Karson (Ex. 1018) describes an application that uses subclassing and hooking to add a bar to the top of windows of other programs to provide other functionality including entering information or

launching another program. (Ex. 1018, 2:46-66; *id.* at 4:66-5:2; *id.* at 7:15 (noting that "hooking and subclassing [*are*] well known")).

43. A POSA would have recognized several benefits to integrating two programs using subclassing. First, because subclassing operates through Windows OS API functions, the technique allows programmers to integrate new functionality from one program into a second (target) program even if the target program was not designed to facilitate such integration. Second, because subclassing works by intercepting messages, the program with the subclassed window need not ever know of the integration.

V. SUMMARY OF THE '591 PATENT CLAIMS

44. The '591 patent (Ex. 1001)[7] describes a straightforward application of subclassing and hooking as described in Section IV above. Claim 1 of the '591 patent is the only independent claim and is reproduced below. The bracketed letters are added for the purposes of cross references.

> A system for integrating an audio or video program with an application program comprising:
>
> [*A*] hardware including a processor;
>
> [*B*] an operating system;
>
> [*C*] the application program running on the operating system; and,

[7] Unless otherwise indicated, all citations in Section V are to Ex. 1001.

[*D*] the audio or video program running on the operating system,

[*E1*] the audio or video program separate from [*E2*] but integrated with the application program [*E3*] such that the application program is unaware that the audio or video program has been integrated therewith,

[*F1*] wherein a user of the application program directly interacts with the application program, and [*F2*] the user interacts with the audio or video program as though the audio or video program were part of the application program,

[*G*] wherein the processor executes the operating system, the application program, and the audio or video program,

[*H*] wherein the audio or video program is further integrated with the application program by subclassing into a window of the application program such that the audio or video program detects when an event related to the application program occurs.

45. Elements A, B, C, D, and G recite a typical basic computer system running an operating system and several other programs. (10:1-42). The specification identifies several operating systems, including Windows OS, Apple Mac OS, Linux, and Unix (3:12-17), I focus on Windows OS for the reasons noted in Paragraph 12 above. All of these operating systems were typical in the 1990s. (*See, e.g.*, Paragraph 14 above; Ex. 1021).

46. Specifically, Elements A and G recite "hardware including a processor," which executes the OS and the other programs. As a POSA would

have realized, computer hardware necessarily includes such a processor. In fact, aside from the claims themselves, the '591 patent never even mentions processors.

47. Element D also requires that one of the programs running on the computer system be an "audio or video program." (10:8-12). A POSA would have understood that an "audio or video program" was an application program that can play, record, and/or otherwise process audio and/or video. This understanding is consistent with the '591 patent's description of the audio or video program as a program that "allows for recording and/or playback" of audio and/or video. (3:27). For example, the '591 patent explains that the audio or video program may "control a video camera" (3:31) and "process both audio and video" (3:36), in addition to playing and recording audio or video (3:39-41). Such "audio or video program[*s*]" were common by the end of 1990s. (*See, e.g.*, Paragraph 14 above; Exs. 1024, 1025, 1026).

48. Elements C, E2, and H use the term "application program" to refer to another program running on the computer system, with which the audio or video program is "integrated." (10:6-10). The specification explains that the application program could be an "email program[]," a "spreadsheet program[]," a "word processing program[]," among various examples, or any other program that "processes data for the user." (3:18-25). This discussion of the application program is consistent with how a POSA would have understood the phrase, as

discussed in Paragraph 18 above. In particular, in the context of the '591 patent, a POSA would have understood "application program" to mean "any computer program that is distinct from the Operating System (OS)."

49. Element E1 requires that the audio or video program be "separate from" the application program. A POSA would have understood this language to require that the audio or video program be capable of being executed without the other application program. This is consistent with the discussion in the '591 patent, which illustrates the concept of the audio or video program being "separate from" the application program by depicting each program as a separate box in a diagram. (Ex. 1001, 3:25 & Figure 1).

50. Element F1 specify that users "directly interact with the application program" (10:13-14). A POSA would have understood that this phrase requires an application program that allows the user to interact with one or more user interface elements presented by the application program.

51. Element E3 also require that the application program be "unaware that the audio or video program has been integrated therewith." (10:10-12). As the specification explains, this means that the application program was not "developed a priori" to integrate with the audio or video program "and also does not operate with knowledge that the integration has occurred." (3:58-64).

52. The "detailed description" section of the '591 patent notes several integration techniques—including but not limited to subclassing—that were well known in the 2001-02 timeframe. However, the claims of the '591 patent require that subclassing be used in the course of integrating the audio or video program with the application program. (12:8-23).

53. Subclassing allows programmers to accomplish such integration without having access to the target application's source code or other proprietary information. The target application operates without knowledge of the integration.

54. The "summary of the invention" section of the '591 patent does not mention subclassing or otherwise discuss any particular technique for achieving such integration. The inventors appear to have believed that there was something inventive about the underlying idea of integrating an audio or video program with another application program. I disagree, as previously discussed.

55. While the term subclassing can have different meanings in different contexts, the specification makes clear that the '591 patent refers exclusively to the subclassing technique described in Section IV.D above.[8] For example, '591

[8] The claims of the '591 patent require "subclassing into a window of the application program." The term "subclassing into a window" would not have had any special meaning to a POSA beyond the standard and well-known meaning of subclassing a window, as described in Section IV.D above. To the extent that the

patent's description of subclassing is nearly identical to a description from a prior art reference book:

> "Subclassing specifically *deals with intercepting messages* bound for one or more windows, which are **intercepted before that can reach their destination window**. The intercepted message can be processed, and then *sent to its initial destination*."

Ex. 1001, '591 at 6:2-6.[9]

> "Subclassing techniques *deal with intercepting messages* bound for one or more windows or controls. These messages are **intercepted before they can reach their destination window**. The intercepted message can be left in its original state or modified. Afterward, the message can be *sent to its original destination* or discarded."

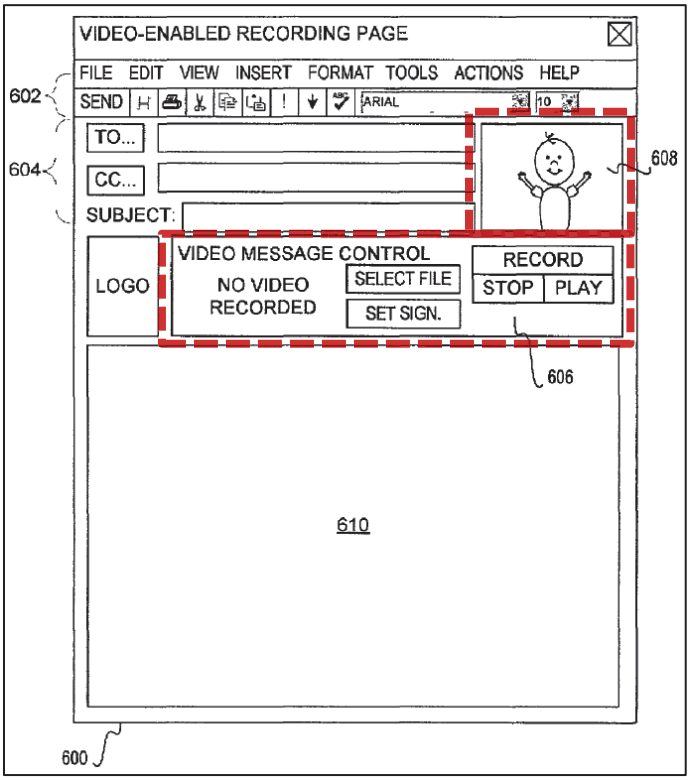
Ex. 1010, Teilhet at 4. These descriptions are consistent with how a POSA would have understood the term subclassing (i.e., a technique that intercepts messages bound for one or more windows, which are intercepted before they can reach their destination window). Notably, this is the tightest form of integration described in the specification.

---

word "into" provides any additional meaning, a POSAwould have understood it as emphasizing that the program performing the subclassing is doing so into a window that the program did not create.

[9] All emphasis added unless other noted.

56. The ’591 patent gives just one example of how subclassing is used to integrate the audio or video program with the application program. (6:21-7:9). The audio or video program is able to subclass Outlook Express Version 5’s windows, overlay the audio/video controls, and adjust the size of the To/From/Subject fields. (*Id.*). The resulting email composition window is depicted in Figure 6 (audio/video controls outlined):



57. Element H requires that the “audio or video program” be “further integrated with the application program by subclassing into a window of the application.” As a POSA would have understood from this language, the claim does require that the audio or video program itself perform the subclassing of the other application’s window. Instead, the claims require only that the window of

the application program be subclassed and that this subclassing serve to integrate the two programs together in some way.

58. However accomplished, integrating an audio or video application into a preexisting application program is necessarily a "system for integrating" two programs as described in the preamble to Claim 1. (10:2-3). Further, integrating such programs using the subclassing technique in the Windows OS environment necessarily involves the audio or video program "detect[*ing*] when an event related to the application program occurs" (as required in Element H). (10:21-23). As discussed in Section IV.C above, such messages are fundamental to how GUI-based applications interact with Microsoft Windows.

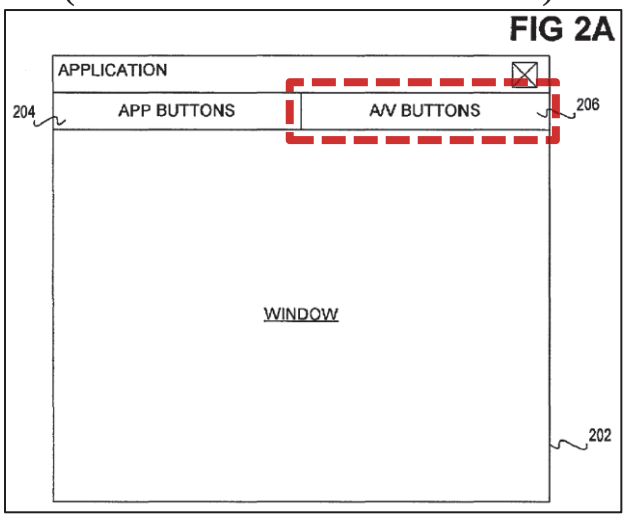
59. Element F2 further requires that the integration allow the user to "interact[] with the audio or video program as though the audio or video program were part of the application program." (10:14-16). A POSA would have understood this limitation to mean that the user interfaces ("UI") for the audio or video program and application program are related. For example, the UI may be related when some aspect of the UI of the application program is used to control the audio or video program, when content from the audio or video program is presented via the UI of the application, and/or or when the audio or video program modifies some aspect of the UI of the application program. (4:56-62). As the '591 patent explains, such a relationship means that the user "may or may not be aware

that the audio or video program is actually external to the application program" (4:1-3).

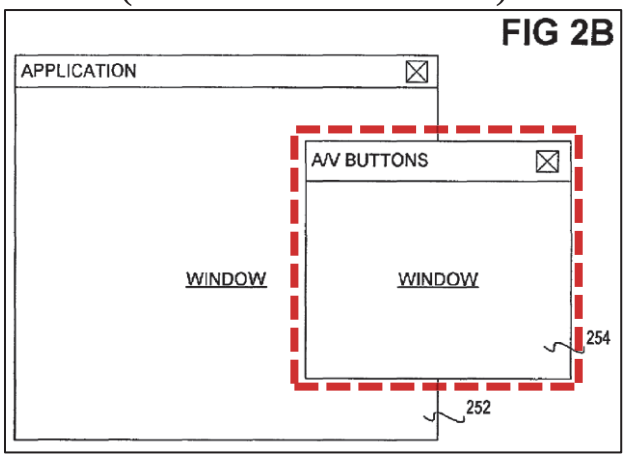
60. If a user can in fact "interact[] with the audio or video program as though the audio or video program were part of the application program" as claimed (10:14-16), a POSA would also necessarily understood the audio or video program as being "integrated with" the application program per the language appearing earlier in '591 claim 1 (10:9-10). Indeed, "integrated with" is a broad and generic phrase. In other words, a POSA would have understood that one application program is "integrated with" another if the UI are related in some way.

61. Claim 2 requires that the "audio or video program modif[*y*] … a window of the application program" (10:24-26), while Claim 3 requires that the "audio or video program runs in a window … related to a window of the application program" (10:27-30). The specification provides examples:

'591 Fig. 2A
(window modifications outlined):



'591 Fig. 2B
(related window outlined)



(Figs. 2A and 2B; *see also* 4:45-49 ("[*T*]he toolbar buttons 206 of FIG. added by the audio or video program 106 to the window 202 can be considered an example of the audio or video program 106 modifying the contents of the window 202 of the application program 104."); 4:53-56 ("The audio-or-video window 254 is related to the application window 252 in that tasks performed in the audio-or-video window 254 relate to the application window 252 in some way.")). A POSA would have understood the term "runs in a window" to refer to a running program with a top-level window GUI element. A POSA would have understood that when the operating system is Windows OS, the operating system creates all the windows, as discussed in Paragraph 25 above.

62. Further, a POSA would have understood the phrase "wherein the audio or video program modifies contents of a window of the application program created through the operating system" (i.e., in claim 2) as being satisfied by any system in which the audio or video program alters the appearance of a GUI window element associated with the application program, or inserts audio or video into the window.

63. In addition, a POSA would have understood the phrase "wherein the audio or video program runs in a window created through the operating system and related to a window of the application program created through the operating system" (i.e., in claim 3) as being satisfied by any system in which one or more

tasks performed in the AV program's window relate to the application program's window in some way. This understanding is consistent with the '591 specification, which explains that the audio or video program window is "related to" the window of the other application program if "tasks performed in the audio-or-video window relate to the application window in some way." (Ex. 1001, 4:53-56).

64. Claim 4 requires that the application program be a certain type of application program, such as an "email program," "a word processing program," or "a spreadsheet program," among others. (10:31-35). Claim 5 requires that the audio or video program comprises one of a "player program," a "recorder program," and a "player-and-recorder program" (10:36-39), which distinguishes the audio or video program from other types of audio or video programs, as discussed in Paragraph 47 above. Claim 6 requires that the audio or video program comprise one of an "audio-only program," a "video-only program," and an "audio-and-video program." (10:41-42).

65. To summarize, I have applied the following claim constructions when analyzing claims 1-6 of the '591 patent:

CLAIM TERM	CLAIM CONSTRUCTION
"application program"	"any computer program that is distinct from the operating system"

CLAIM TERM	CLAIM CONSTRUCTION
"audio or video program"	"an application program that can play, record, and/or otherwise process audio and/or video"
"separate from"	Satisfied if the audio or video program can be executed without the other application program
"integrated with"	Satisfied if the UI of the AV program and the other application program are related in some way.
"such that the application program is unaware that the audio or video program has been integrated therewith"	Satisfied if the application program was not designed or modified to facilitate integration of a separate AV program and does not operate with knowledge that the AV program has been integrated therewith.
"wherein a user of the application program directly interacts with the application program"	Satisfied if the user interacts with one or more user interface elements presented by the application program

CLAIM TERM	CLAIM CONSTRUCTION
"the user interacts with the audio or video program as though the audio or video program were part of the application program"	Satisfied if the UI of the AV program and the other application program are related in some way (e.g., by the UI of the other application being used to control the AV program and/or by content from the AV program being presented via the UI of the other application).
"window"	"a GUI element having a window procedure that can receive messages and act on them"
"subclassing"	"a technique that intercepts messages bound for one or more windows, which are intercepted before they can reach their destination window"
"wherein the audio or video program modifies contents of a window of the application program created through the operating system"	Satisfied by any system that runs in Microsoft Windows and in which in the audio or video program alters the appearance of a GUI window element associated with the application program, or inserts audio or video into the window.

CLAIM TERM	CLAIM CONSTRUCTION
"wherein the audio or video program runs in a window created through the operating system and related to a window of the application program created through the operating system"	Satisfied by any system that runs in Microsoft Windows and in which the AV program's window relate to the application program's window in some way.

VI. CLAIMS 1-6 ARE UNPATENTABLE IN LIGHT OF THE PRIOR ART IDENTIFIED IN RPX'S PETITION

66. I have been asked to provide my opinion concerning whether claims 1-6 of the '591 patent are unpatentable based on the prior art references identified in RPX's Petition. The prior art references I reviewed are:

EXHIBIT	PRIOR ART REFERENCE
1003	U.S. Pat. App. Pub. No. 2002/0090062 (filed Nov. 16, 2001) ("Alhadad")
1004	U.S. Pat. No. 6,219,047 (filed Sept. 17, 1998, issued Apr. 17, 2001) ("Bell")
1005	U.S. Pat. No. 5,619,639 (issued Apr. 8, 1997) ("Mast")
1006	WIPO Pat. App. 2000/68769 (published Nov. 16, 2000) ("Lui")

67. I understand that in an *Inter Partes* Review proceeding, claim terms should be given their broadest reasonable construction consistent with the

specification. In my analysis below, I apply that standard to the words and phrases of the challenged claims—summarized in paragraph 65 above.

68. My opinions on the disclosure of each prior art reference, relative to the limitations of claims 1-6 of the '591 patent, are provided below.

A. Ground 1: Alhadad Discloses Each Limitation of Claims 1-6 of the '591 Patent

69. According to the face of the document, Alhadad (Ex. 1003)[10] is a published U.S. patent application that was filed on November 16, 2001. Alhadad discloses a computer system that uses subclassing to integrate two call center programs. For example, Alhadad describes one program automatically playing audio as a consequence of a user interacting with a GUI of the second program.

70. After reviewing Alhadad and claims 1-6 of the '591 patent, it is my opinion that Alhadad discloses every limitation of the claims. The basis for my opinion and the details of my analysis are below.

1. Claim 1: "A system for integrating an audio or video program with an application program comprising:"

71. Alhadad discloses a system for integrating an audio or video program with an application program. (¶ 47 ("[*T*]he invention resides primarily in the augmentation of the software employed by the call handler's computer workstation, being operative to automatically trigger actions, such as the automatic

[10] Unless otherwise indicated, all citations in Section VI.A are to Ex. 1003.

playback of one or more pre-recorded PRS phrases or other pre-defined actions during a call, in response to the call agent's interactions (such as through a mouse click or keyboard action) with the target software.")). The elements of the preamble are discussed thoroughly in Sections VI.A.1.c through VI.A.1.g below.

a. "[*A*] hardware including a processor;"

72. Alhadad discloses a computer. (¶ 49 ("Attention is initially directed to FIG. 1, which shows the main components of the FAQsoft mechanism of the invention. It is ***comprised of a Personal Computer*** (PC) 10…."); Fig. 1 ("Computer")).

73. Alhadad further explains that the computer runs Windows OS, as discussed in Section VI.A.1.b below. A POSA would have understood Alhadad's disclosure of a "Personal Computer … running Microsoft Windows" to describe hardware including a processor. Any computer must include a processor in order to function and run an operating system such as Windows OS. Further, the processor must be—or run on—hardware.

b. "[*B*] an operating system;"

74. Alhadad discloses Windows OS. (¶ 49 ("Attention is initially directed to FIG. 1, which shows the main components of the FAQsoft mechanism of the invention. It is comprised of a Personal Computer (PC) 10 running ***Microsoft***

Windows Operating System OS (e.g., ***Win95*** and later)…."); Fig. 1 ("Windows OS")).

75. A POSA would have understood "Microsoft Windows Operating System" or "Win95" (i.e., Microsoft Windows 95) to be an operating systems, as discussed Paragraph 16 above. The '591 patent identifies "Microsoft Windows OS" as an example of an operating system that can be used in connection with the purported invention. (Ex. 1001, 3:12-14; *see also* Paragraph 45 above).

c. "[*C*] the application program running on the operating system; and,"

76. Alhadad discloses various application programs running on Windows OS, including what Alhadad calls the "Target Applications." (¶ 50 ("The PC 10 is used to run Target Applications…."); Fig. 1 (showing "Window OS 15" as distinct from "Target Application 14"); *see also* Section VI.A.1.b above (noting that "PC (10)" runs Windows OS)).

77. Alhadad also explains that "Target Applications" may assist the user with specific tasks. (¶ 50 ("Target Applications [*are*] typically used in Call Centers for taking orders or collecting information from a client."); ¶ 4 ("As a non-limiting example, in the case of a catalog order center, a 'target' application program will typically graphically display an order or business 'form' on the call handler's terminal display.")).

78. As one example, Alhadad discloses that the "target application" may be an email program such as Microsoft Outlook. (¶ 151 ("FIG. 24 shows a normal ***Outlook*** Contact window before running the FAQsoft routine and FIG. 25 shows the same window after being configured with FAQsoft and with FAQsoft running."); Fig. 25 (showing Microsoft Outlook with FAQsoft integrated)).

79. A POSA would have understood a "target application," such as an email program, to be an application program given that the target application is a computer program that is distinct from the OS, per the meaning that "application program" would have had to a POSA (paragraph 48 above). In addition, the '591 patent identifies email programs such as Microsoft Outlook and Microsoft Outlook Express as examples of application programs running on the OS that can be used in connection with the purported invention. (Ex. 1001, 2:6-8; 6:21-33; 7:41-51).

d. "[*D*] the audio or video program running on the operating system,"

80. Alhadad discloses a program called "FAQsoft" or the "FAQsoft mechanism," which runs on the operating system. (¶ 57 ("applications, such as FAQsoft"); ¶ 49 ("Attention is initially directed to FIG. 1, which shows the main components of the FAQsoft mechanism of the invention."); ¶ 68 ("FAQsoft is designed to ***run*** silently as a background process and will auto start when the computer is booted up.")).

81. As Alhadad discloses and a POSA would have understood, FAQsoft can record audio. (¶ 68 ("Attention is now directed to FIG. 6, which is a flow chart showing the process of invoking and running the FAQsoft mechanism of the present invention."); ¶ 74 ("If the ***audio files*** are not available locally, the FAQsoft program will retrieve them from the Server or ***have the user record them*** first before continuing in step 180. The Phrase Manager shown in FIG. 11 will be presented if listed phrases have not been recorded. The Phrase Manager ***provides all necessary tools including a Sound Editor to facilitate recording and editing of phrases***."); Fig. 6; Fig. 11).

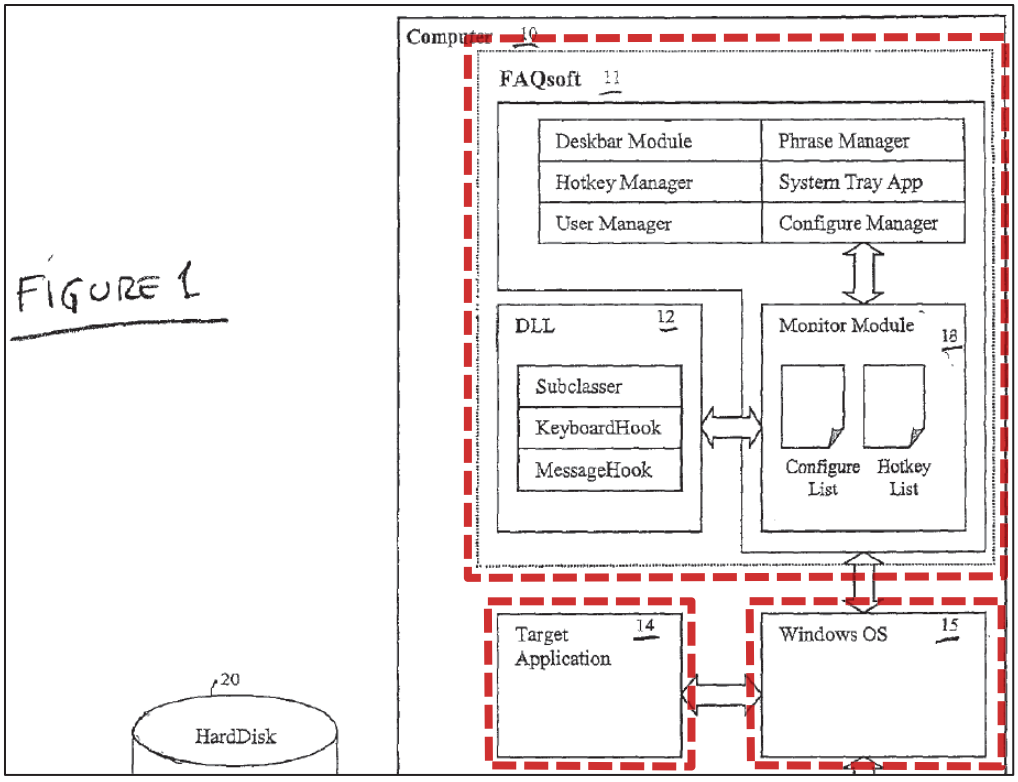
82. Further, as Alhadad discloses and a POSA would have understood, FAQsoft can also play audio . (¶ 13 ("[*T*]he present invention is directed to an enhanced computer workstation-associated call handling mechanism, that may be readily employed by a call handling agent to automatically trigger actions, such as but not limited to the ***automatic playback of one or more pre-recorded*** [*personalized response system*] ***phrases***…."); ¶ 49 ("The system can include a server 26 to archive files of personalized voice recordings (Audio Files) and to store the FAQsoft database. It … may be connected to the sound card I/O ports or other ports for voice communication with a telephone system 29."); ¶ 81 (explaining that FAQsoft can "Play the phrase"); ¶174 ("[*T*]he invention may be readily employed by a call handling agent to automatically trigger various call

handler responses, including ***automatic playback of pre-recorded personal response type phrases***….").

83. Because Alhadad describes FAQsoft as capable of recording and playing back audio, a POSA would have understood Alhadad to disclose an "audio or video program" (i.e., FAQsoft) running on the OS within the meaning of that phrase to a POSA (paragraph 45 above).

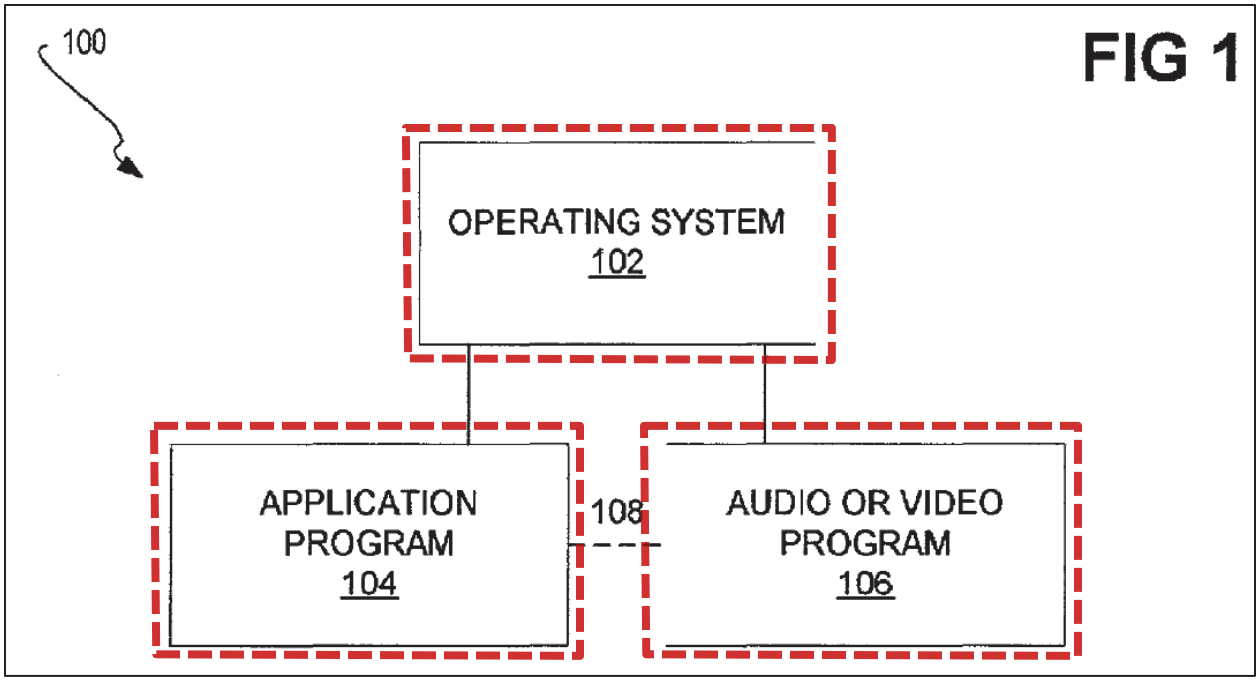
e. "[*E1*] the audio or video program separate from"

84. Alhadad explains that FAQsoft is distinct from the target application. (¶¶ 49-52 ("Attention is initially directed to FIG. 1, which shows the main components of the FAQsoft mechanism of the invention. … The Target Application 14 responds according to its programmed Windows Procedures. … The FAQsoft mechanism 11 monitors messages sent to these Controls and modifies their behavior by substituting or appending another procedure to the Control's original Windows Procedure.")).



(Fig 1 (excerpted) (outlining FAQsoft as distinct from the Target Application and the OS)).

85. Given this disclosure, a POSA would have understood Alhadad as disclosing an audio or video program (i.e., FAQsoft) "separate from" an application program (i.e., the target application) in that FAQsoft could be executed separately from the application program, per the meaning of "separate from" to a POSA (paragraph 49 above). Indeed, Figure 1 in Alhadad mirrors Figure 1 in the '591 patent, which the specification cites when describing the audio or video program as "separate from" the application program (Ex. 1001, 3:25-26):



(Ex. 1001, Fig. 1 (outlining the OS, the application program, and the audio or video program).

f. "[*E2*] but integrated with the application program" & "[*H*] wherein the audio or video program is further integrated with the application program by subclassing into a window of the application program such that the audio or video program detects when an event related to the application program occurs."

86. Alhadad discloses that FAQsoft is integrated with the target application by watching for user interactions with the GUI of the target application that trigger automatic playback of recorded audio, as discussed in this Section. (*See generally* ¶¶ 59-67 of Alhadad).

87. FAQsoft installs a message hook to identify when a user activates a window of the target application. (¶ 123 ("In order to enable the FAQsoft routine

to monitor other applications, hooks are installed (step 510) which trap Windows messages sent to these applications. These hooks are implemented as a Windows dll ([']Dynamic Link Library[']) file 12…. The Message Hook installed monitors only the WM_ACTIVATE message, in order to monitor focus changes occurring in Top Level Windows only.")).

88. FAQsoft examines the active window to identify controls and match them to a "Response." (¶ 124 ("By looking at the WM_ACTIVATE message, the FAQsoft mechanism is informed whenever the user changes the active Top Level Window. When a WM_ACTIVATE message is received (step 560), FAQsoft enumerates the Controls contained within the active Top Level Window (step 570) using Win32 API call EnumChildWindows. It tries to match items in the Configure List with Controls that exist within the currently active Top Level Window (step 580)."); ¶ 59-60 ("The principal requirements of the FAQsoft mechanism of the invention are as follows: Uniquely identify all GUI objects on a Forms based application (Target Application)…."); *see generally* ¶ 109 ("A specific user's Configure Records and Hotkey Records are known as the Configure List and Hotkey List."); ¶ 72 ("The Configure Records include information that associates Controls with one or more actions to execute (a Response).")).

89. When FAQsoft matches a control in the target application's window to a Response, it subclasses the control to monitor user interaction. (¶ 124 ("If a

match is found, that Control is subclassed (step 590), the hwnd field of the matching items in the Configure List (Configure Items) is updated with the control[']s window handle (hwnd) (step 600). User interaction with that control is monitored once it has been subclassed.")).

90. FAQsoft determines whether a user's interaction with the target application's controls causes a Response, which may include playback of an audio recording. (¶ 139 ("Once a Control is subclassed, the FAQsoft routine will be aware of user interactions with that control. For example, a tab into an edit box will cause the edit box to receive focus. WinOS sends a WM_SETFOCUS message to the edit box which is intercepted (via subclass mechanism) and recognized by FAQsoft as one of the monitored message (step 620). The FAQsoft routine then compares the editbox[']s hwnd with hwnds in the Configure List (step 630). If a match is found, the Event parameter is compared to determine a valid match and raises an event (command) for the Tray Application…."); ¶ 148 ("The Tray Application (step 650) executes the Response if the Condition parameter is met (step 201)."); ¶¶ 76-78 (explaining that the Response may include playing the prerecorded audio)).

91. FAQsoft also alters the appearance of the target application's controls to indicate which user interaction(s) with the controls will cause a Response. (¶ 150 ("The FAQsoft routine will draw screen markers on the control in step 670 to

indicate to the user actions that will cause the FAQsoft routine to respond (additional behavior for that Control).”); ¶ 14 (“Screen markers on the Graphical User Interface (GUI) elements of the Target Software make the Call Agent aware of the GUI elements (e.g., field boxes, buttons, pictures, etc. otherwise known by one schooled in the art as [‘]controls[’]) are associated with a pre-defined action (or series of actions) and have been enhanced by the invention.”); ¶ 151 (“FIG. 24 shows a normal Outlook Contact window before running the FAQsoft routine and FIG. 25 shows the same window after being configured with FAQsoft and with FAQsoft running.”)).

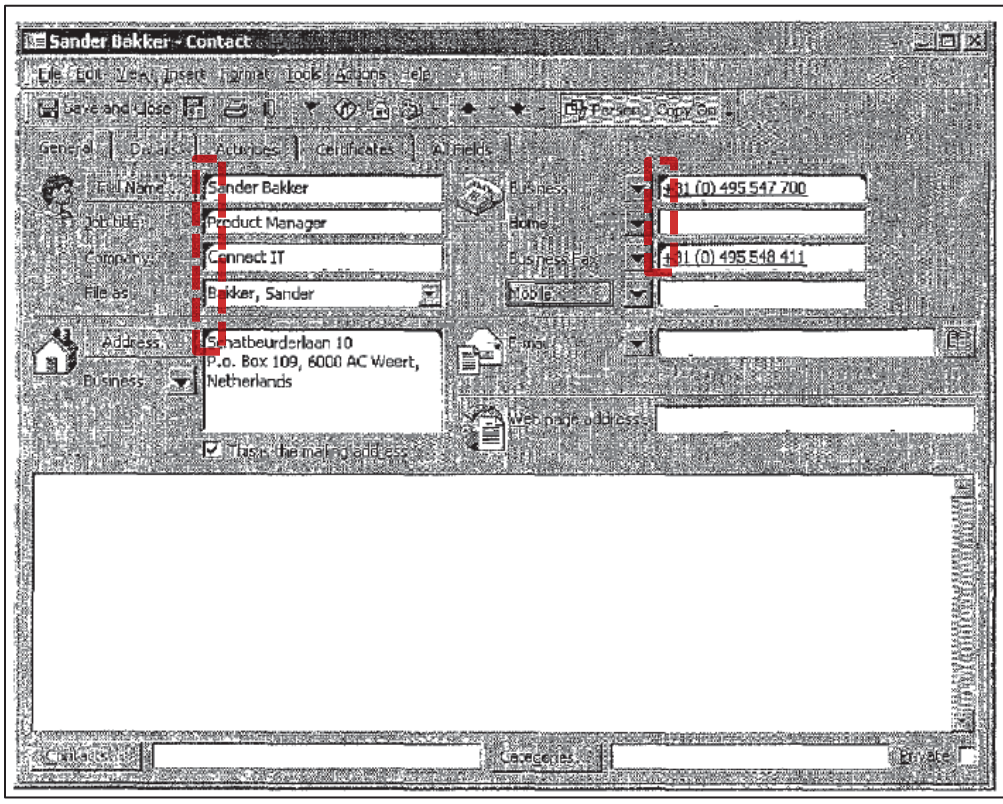
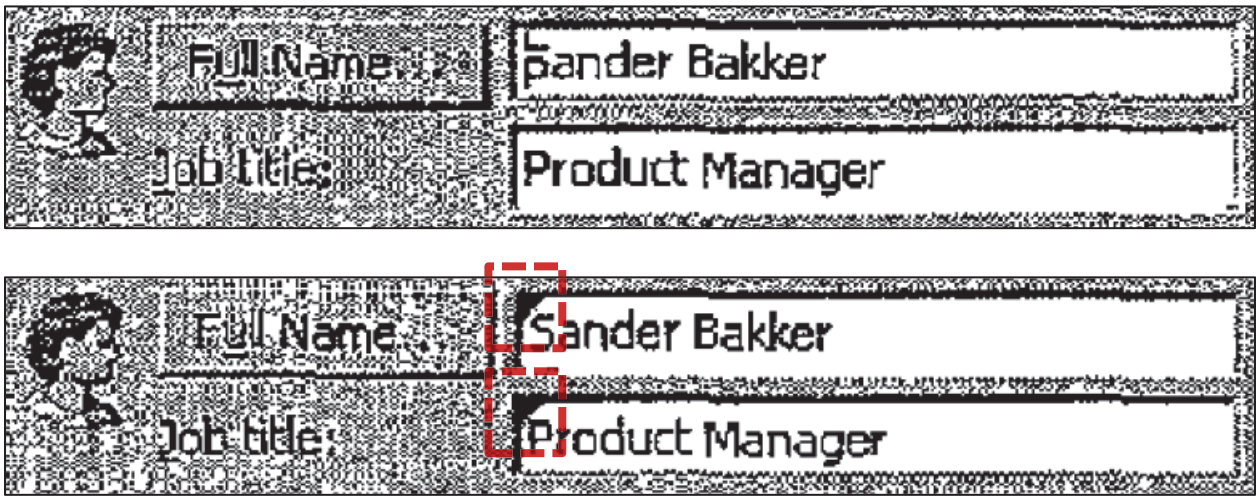


FIGURE 25

(Fig. 25 (showing Microsoft Outlook using FAQsoft, FAQsoft indicators on Microsoft Outlook's controls outlined)). To illustrate the screen markers further, compare Fig. 24 (excerpted below, top) with Fig. 25 (excerpted below, bottom, screen markers outlined)):



92. In certain situations, FAQsoft prevents the target application from receiving certain user input. (¶ 172 ("When the user right-clicks on a monitored control while FAQsoft is in Configure Mode, the Controls context menu is substituted with one supplied by the FAQsoft routine. FIG. 29 shows the original context menu and FIG. 30 shows the substituted one. When the Monitor receives a WM_RBUTTONUP message while in Configure Mode, it raises the ShowContextMenu event (step 891), to inform the Tray Application and does not let that message go to the intended Control.")).

93. Given these extensive disclosures, a POSA would have understood Alhadad as describing the use of subclassing to integrate an audio or video

program (i.e., FAQsoft) with an application program (i.e., the target application, such as Microsoft Outlook). In particular, a POSA would have understood that FAQsoft was "integrated with" the target application given the relationship between the UI of FAQsoft and the UI of the target application. (Paragraph 60 above). Further, given Alhadad's disclosure of subclassing as discussed above, a POSA would have understood that FAQsoft was "further integrated with the application program [*i.e., the target application*] by subclassing into a window of the application program such that the audio or video program detects when an event related to the application program occurs" per the applicable understandings of "subclassing" (Paragraph 55 above), "window" (Paragraph 27 above), and the "further integrated . . ." phrase as a whole (Paragraph 57 above).

g. "[*E3*] such that the application program is unaware that the audio or video program has been integrated therewith,"

94. Alhadad discloses that the target application was not designed a priori to be integrated with FAQsoft. (¶ 17 ("The present invention enhances ***existing*** software…."); ¶ 16 ("A fundamental aspect of the present invention is its ability to respond to user interactions with predefined GUI elements (Controls) in an ***existing*** Target Application, ***without having to modify the source code*** of the Target Software. Instead, hooking and sub-classing techniques made available in WinOS are employed to change the behavior and look of these Target Applications

to provide a higher level of automation for Call Agents, ***without requiring any changes to the Target Application*** itself."); ¶ 174 ("As will be appreciated from the foregoing description, … the FAQsoft mechanism of the present invention ***avoids having to modify the source code of the target software***. This enables the invention to provide a higher level of automation for call agents, without requiring any changes to the target application itself.").

95. Furthermore, a POSA would have understood that the subclassing technique described in Alhadad integrates FAQsoft with the target application without making the target application aware of the integration. As discussed in Paragraph 43 above, nothing about integration with subclassing requires that the target application ever know of the integration. As described in Part VI.A.1.f above, the hooking and subclassing merely monitor and intercept messages. Nothing in Alhadad suggests that the target application would receive any new message that could inform it of the integration—let alone one that it would recognize as such. Although FAQsoft draws screen markers on the subclassed controls of the target application, a POSA would have understood the target application as being unaware of the integration. By the same token, a POSA reviewing the '591 patent's only example of integration by subclassing, in which the audio or video program alters the size of an envelope window in Microsoft Outlook Express (Ex. 1001, 6:36-42), would have understood Microsoft Outlook

Express as being unaware of having been integrated with the audio or video program.

96. Thus, given that a POSA would have understood that the target application was not "developed a priori" to integrate with the audio or video program "and also does not operate with knowledge that the integration has occurred" per the interpretation discussed in paragraph 51 above, a POSA would have understood Alhahad as disclosing a system in which the application program is "unaware that the audio or video program has been integrated therewith."

h. "[*F1*] wherein a user of the application program directly interacts with the application program, and"

97. Alhadad discloses that even after the two programs are integrated, the user interacts with the GUI of the target application. (¶ 16 ("A fundamental aspect of the present invention is its ability to respond to user interactions with predefined GUI elements (Controls) in an existing Target Application…."); ¶¶ 130-38 ("The FAQsoft routine's subclassed Controls … trap[] the messages listed below to detect the following user actions. … user has closed the Top Level Window … user has either tabbed into or clicked on control … user has tabbed out of clicked on another control.")). Allahad also cites "mouse click[s]" and "keyboard action" as examples of a user's "interactions" with the target software. (¶¶ 47 & 50 ("A user interacts with the Target Application[']s Graphical User Interface (GUI) using a keyboard and/or mouse or other input device.")).

98. Alhadad explains that the user enters information into the target application, which runs on the "Personal Computer." (Part VI.A.1.c above). As discussed in Paragraph 50 above, anytime a user enters information into an application program running on the user's computer via a GUI window element of that application, a POSA would have understood the interaction to be direct. Therefore, a POSA would have understood Alhadad as disclosing a system in which a user directly interacts with the application program (i.e., the target application).

i. "[*F2*] the user interacts with the audio or video program as though the audio or video program were part of the application program ,"

99. Alhadad discloses that the user interacts with FAQsoft as though it were part of the target application, because a user's interactions with the GUI of the target application causes a response from FAQsoft. (¶ 2 ("The present invention is operative to automatically trigger actions, such as the automatic playback of one or more pre-recorded PRS phrases or other pre-defined actions at appropriate junctures during a call, in response to the call agent's interactions (via mouse or keyboard) with the target software."); ¶ 16 ("A fundamental aspect of the present invention is its ability to respond to user interactions with predefined GUI elements (Controls) in an existing Target Application…."); ¶¶ 59-63 (describing a

"principal requirement[] of the FAQsoft" is to "Respond to user manipulations or interactions with the selected GUI object" of the target application).

100. Given these descriptions, a POSA would have understood that the UI of the target program controls FAQsoft. As discussed in paragraph 59, a POSA would have understood that a user "interacts with the audio or video program as though the audio or video program were part of the application program" provided that the UI are related in some way. Thus, a POSA would have understood Alhadad as disclosing a system in which a user interacts with an audio or video program (i.e., FAQsoft) as though it were part of the application program (i.e., the target application).

j. "[*G*] wherein the processor executes the operating system, the application program, and the audio or video program,"

101. As discussed in Sections VI.A.1.a though VI.A.1.d above, Alhadad discloses a computer with a processor, an OS running on the computer, an application program referred to as the target application running on the computer, and an audio or video application called FAQsoft running on the computer. (¶ 49 ("a Personal Computer (PC) 10 running Microsoft Windows Operating System"); ¶ 50 ("The PC 10 is used to run Target Applications…."); ¶ 68 ("FAQsoft is designed to run … when the computer is booted up.")). A POSA would have

understood these disclosures to mean that the processor executes (i.e., runs) Windows OS, the target application, and the FAQsoft.

2. Claim 2: "The system of claim 1, wherein the audio or video program modifies contents of a window of the application program created through the operating system."

102. Alhadad explains that FAQsoft modifies the contents a window of the target application. FAQsoft "makes it possible to change the behavior and appearance of selected Controls within the Target Application." (¶ 75). In particular, as described in Section VI.A.1.f above, FAQsoft draws indicators on the target application's controls indicating the FAQsoft functionality added to that control. Thus, a POSA would have understood Alhahad as disclosing a system in which the audio or video program (i.e., FAQsoft) modifies contents of a window of the application program (i.e., the target application).

103. Further, a POSA would have understood from Alhadad's identification of Windows OS (*see* Section VI.A.1.b above) that the OS creates the target application's window. (Paragraph 25 above (noting that Windows OS creates windows when instructed by a program)). By the same token, the '591 patent's only example of integration by subclassing concerns Microsoft Outlook Express and is specific to Windows OS, as discussed in Paragraph 12.b above). A POSA reviewing this example would have understood Windows OS as creating Microsoft Outlook Express's windows.

104. Thus, given how a POSA would have understood "window" (paragraph 27 above) and also given that a POSA would have understood that "the audio or video program modifies contents of a window of the application program created through the operating system" if the programs are running on Microsoft Windows and the audio or video program alters the appearance of a GUI window element associated with the application program (paragraph 62 above), a POSA would have understood Alhadad to disclose a system in which the audio or video program [i.e., FAQsoft] modifies contents of a window of the application program [i.e., the target application] created through the operating system [i.e., a version of Microsoft Windows]."

3. Claim 3: "The system of claim 1, wherein the audio or video program runs in a window created through the operating system and related to a window of the application program created through the operating system."

105. Alhadad discloses that FAQsoft runs in several windows. First, Alhadad explains that FAQsoft's "Monitor Module" communicates that a Response is appropriate following user interaction with the target application's UI by sending FAQsoft's "Tray Application" a "message." (¶ 75). As discussed in Paragraph 29 above, to be the destination for a message on Windows OS, FAQsoft's Tray Application must have a window. A POSA would have understood that the Tray Application's window is related to the window of the target application, because the Tray Application is the destination of the message

indicating the user's interaction with the UI (i.e., either on or itself a window) of the target application.

106. Second, Alhadad explains that FAQsoft has a "Quick Configure Window." (¶ 68 ("Attention is now directed to FIG. 6, which is a flow chart showing the process of invoking and running the FAQsoft mechanism of the present invention."); ¶ 99 ("The Quick Configure step 380 displays the Quick Configure Window (step 390), shown in FIG. 18. This window lists the combination of Phrase items and Response items and takes up relatively little display screen area. It also causes the FAQsoft routine to operate in Configure Mode, so that the items can be dragged and dropped onto controls in Target Applications.")).

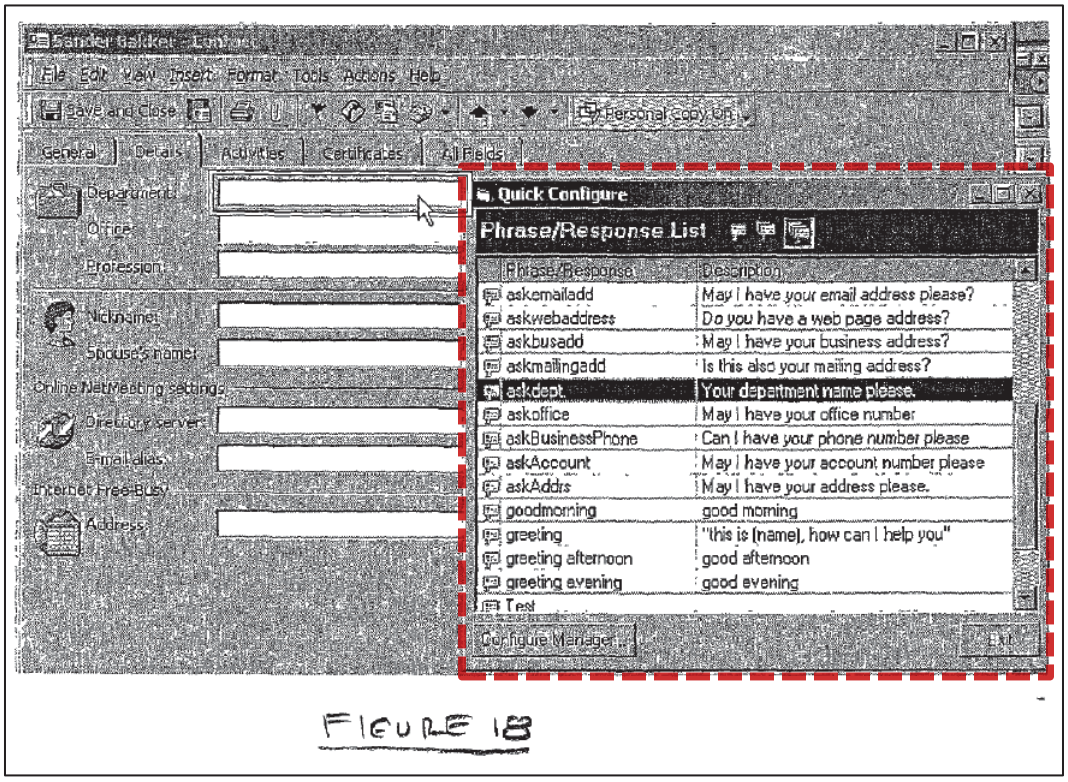


FIGURE 18

(Fig. 18 (Quick Configure Window outlined)). A POSA would have understood that the Quick Configure Window is related to the target application's window, because the user can drag items from the Quick Configure Window onto the target application's window.

107. Thus, a POSA would have understood Alhahad as disclosing a system in which the audio or video program (i.e., FAQsoft) runs in a window (i.e., the Quick Configure Window and Tray Application Window related to a window of the application program (i.e., the target application, such as Microsoft Outlook).

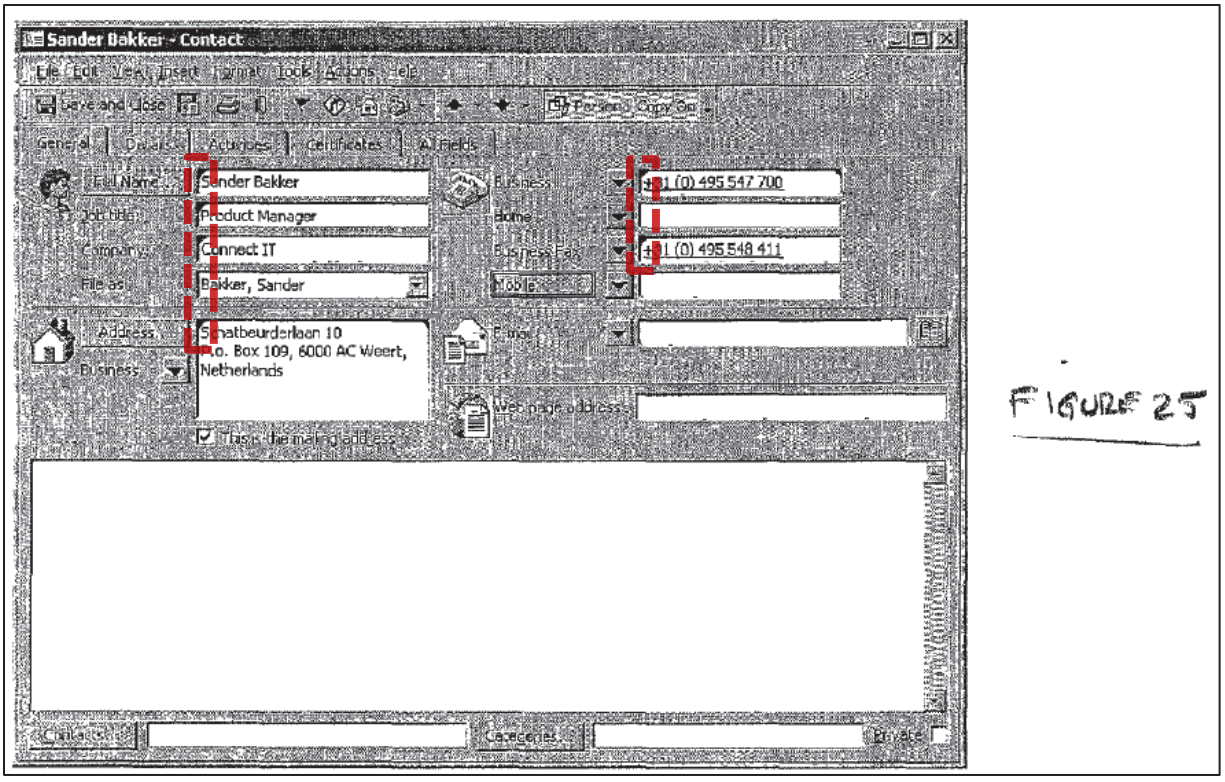
108. Further, a POSA would have understood from Alhadad's identification of Windows OS (*see* Section VI.A.1.b above) that the operating system creates FAQsoft's windows and the target application's windows. See Paragraph 25 above (noting that Windows OS creates windows when instructed by a program)). In addition, since the '591 patent's only example of integration by subclassing is specific to Windows OS (Paragraph 12.b above), Windows OS creates the windows in Alhadad to same extent it creates those in the '591 patent example.

109. Thus, given how a POSA would have understood "window" (paragraph 27 above) and also given that a POSA would have understood that "the audio or video program runs in a window created through the operating system and related to a window of the application program created through the operating

system" if one or more tasks performed in the audio or video program's window relate to the application program's window in some way (paragraph 63 above), a POSA would have understood Alhadad to disclose a system in which "the audio or video program [i.e., FAQsoft] runs in a window created through the operating system [i.e., a version of Microsoft Windows] and related to a window of the application program [i.e., the target application] created through the operating system."

4. Claim 4: "The system of claim 1, wherein the application program comprises one of: an email program, a presentation program, a publishing program, a word processing program, a spreadsheet program, an instant messaging program, a telephony program, and a gaming program."

110. Alhadad discloses using FAQsoft with Microsoft Outlook, which a POSA would have recognized as a popular email program. (¶ 151 ("FIG. 24 shows a normal ***Outlook*** Contact window before running the FAQsoft routine and FIG. 25 shows the same window after being configured with FAQsoft and with FAQsoft running.").



(Fig. 25 (showing Microsoft Outlook using FAQsoft, FAQsoft indicators on Microsoft Outlook's controls outlined)).

5. Claim 5: "The system of claim 1, wherein the audio or video program comprises one of: an audio or video player program, an audio or video recorder program, an audio or video player-and-recorder program."

111. As described in Section VI.A.1.d above, FAQsoft records and plays audio. (¶ 13; ¶ 49; ¶ 68; ¶ 74 ¶¶ 76-81; ¶ 174). Thus, a POSA would have considered FAQsoft to constitute "an audio or video player-and-recorder program" and thus would have understood Alhadad as disclosing such a program.

6. Claim 6: "The system of claim 1, wherein the audio or video program comprises one of: an audio-only program, a video-only program, and an audio-and-video program."

112. As described in Section VI.A.1.d above, FAQsoft records and plays audio, with no discussion of video. (¶ 13; ¶ 49; ¶ 68; ¶ 74 ¶¶ 76-81; ¶ 174). Thus, a POSA would have considered FAQsoft to constitute at least "an audio-only program" and thus would have understood Alhadad as disclosing such a program.

B. Grounds 2 and 3: Bell Discloses Each Limitation of Claims 1-6 of the '591 Patent and Therefore Anticipates Those Claims, or Renders Claims 1-6 of the '591 Patent Obvious

113. According to the face of the document, Bell (Ex. 1004)[11] is a U.S. patent that was filed on September 17, 1998, and issued on April 17, 2001. Bell discloses a training application integrated with a host application such that the training agent automatically presents video and audio tutorial instruction when a user selects certain GUI elements of the host application.

114. After reviewing Bell and claims 1-6 of the '591 patent, it is my opinion that Bell discloses every limitation of the claims. Alternatively, Bell would have rendered every claim 1-6 of the '591 patent obvious to a POSA. The basis for my opinion and the details of my analysis are below.

[11] Unless otherwise indicated, all citations in Section VI.B are to Ex. 1004.

1. Claim 1: "A system for integrating an audio or video program with an application program comprising:"

115. Bell discloses a system for integrating an audio or video program with an application program. *See* Bell 1:38-44 ("The present invention features methods and apparatus for providing tutorial information for a computer program application through a training agent activated by a user of the application. The agent takes control of the application interface and performs actions, such as finding and displaying tutorial information, in response to application user interface commands."). The elements of the preamble are discussed thoroughly in Sections VI.B.1.c through VI.B.1.g below.

a. "[*A*] hardware including a processor;"

116. Bell discloses a computer with a processor. (9:24-33 ("The invention can be implemented in digital electronic circuitry, or in ***computer hardware***, firmware, software, or in combinations of them. Apparatus of the invention can be implemented in a computer program product tangibly embodied in a machine-readable storage device for execution by a ***programmable processor***; and method steps of the invention can be performed by a ***programmable processor*** executing a program of instructions to perform functions of the invention by operating on input data and generating output."); 3:59-4:2 ("As shown in FIG. 1, a computer system 10 includes a training agent 26 in accordance with the invention. The particular implementation of the computer system and training agent that will be described is

based on a ***personal computer*** running the Windows® 95 operating system 22. … The computer system 10 includes a programmable computer 20….”); Fig. 1 (“Computer”)).

117. Bell further explains that the computer runs Windows OS, as discussed in Section VI.B.1.b below. A POSA would have understood Bell’s disclosure of a “personal computer running the Windows® 95” to describe hardware including a processor. Any computer must include a processor in order to function and run an operating system such as Windows OS. Further, the processor must be—or run on—hardware.

b. “[*B*] an operating system;”

118. Bell discloses Windows OS. (3:59-67 (“As shown in FIG. 1, a computer system 10 includes a training agent 26 in accordance with the invention. The particular implementation of the computer system and training agent that will be described is based on a personal computer running the ***Windows® 95 operating system*** 22.”); 3:36 (“Windows NT 4.0 and Windows NT 5.0.”); Fig. 1 (“Operating System”)).

119. A POSA would have understood “Windows® 95 operating system,” “Windows NT 4.0,” and “Windows NT 5.0” to all be operating systems, as discussed Paragraph 16 above. The ’591 patent identifies “Microsoft Windows

OS" as an example of an operating system that can be used in connection with the purported invention. (Ex. 1001, 3:12-14; *see also* Paragraph 45 above).

c. "[*C*] the application program running on the operating system; and,"

120. Bell discloses various application programs running on Windows OS, including what Bell calls the "host application," "host program," or just "host." (3:60-4:11 ("The particular implementation of the computer system and training agent that will be described is based on a personal computer running the Windows® 95 operating system 22. … The system 10 is used to run computer programs, especially ***application programs*** (applications)."); 4:19-21 ("The selected application will be referred to as the host application, or simply as the host, to distinguish it from the training agent, which is also an application.")).

121. As one example, Bell explains that the host may be a word processing program, such as Microsoft Word. (7:58 ("Microsoft Word")). A POSA would have understood a word processing program to be a program that assists the user with specific tasks.

122. A POSA would have understood a "host," such as a word processing program, to be an application program running on the OS given that the host application is a computer program that is distinct from the OS, per the meaning that "application program" would have had to a POSA (paragraph 48 above). In addition, the '591 patent identifies a word processor as an example of an

application program that can be used in connection with the purported invention. (Ex. 1001, 3:18-25).

d. "[*D*] the audio or video program running on the operating system,"

123. Bell discloses a program called the "training agent," which runs on the operating system. (1:58-60 ("The system has a training agent for finding and displaying tutorial information to a user of a computer program application…."); 2:23-24 ("A training agent is provided to ***run*** on the same computer with the application.")).

124. As Bell discloses and a POSA would have understood, the training agent can present video to the user, with or without audio. (2:29-30 ("Tutorial information is provided for the training agent to present to the user…."); 4:17-19 ("The ***training agent 26 provides … tutorial information*** with respect to a selected application."); *id.* 2:12-14 ("The ***tutorial information may have a video clip***, and the video clip ***may include sound***."); 3:38-42 ("Different tutorials can be prepared for users with different levels of experience. A beginner may want to see a ***video*** describing the general purpose of a window…."); 5:15-18 ("Particularly advantageous is training in the form of a mixture of DVD ***video*** clips and hypertext documents, with a limited amount of additional material from the Internet to provide up-to-the-minute information."); 6:14-17 (describing that the agent can "display[] a video clip"); 9:60-10:21 (claims 1-3: "A system … comprising: a

training agent for finding and displaying tutorial information … wherein the tutorial information comprises a video clip … wherein the video clip includes sound.")).

125. One way that the training agent can present audio or video tutorial information is with a "presentation program." (5:4-9 ("The ***action can be performed in a variety of ways, such as by starting a presentation program***. Typically, the presentation program will display some information pertaining to the selected control. By the selection of an appropriate presentation program, information of any form can be displayed, including text, images, ***audio***, ***video***, or hypertext."). Because the training agent is capable of facilitating playback of audio or video, a POSA would have understood the training agent to be an "audio or video program" running on the OS within the meaning of that phrase to a POSA (Paragraph 47 above).

126. Furthermore, Bell also suggests that the training agent could itself playback audio or video without using a separate presentation program. As Bell emphasizes, "starting a presentation program" is just one of a "variety of ways" the training agent can present audio or video. (5:4-9). Elsewhere in the specification, Bell does not limit the training agent to opening a presentation program and rather describes generally that "the agent … will perform some appropriate action, such as displaying a video clip." (6:14-17). Similarly, Bell's claims describe the

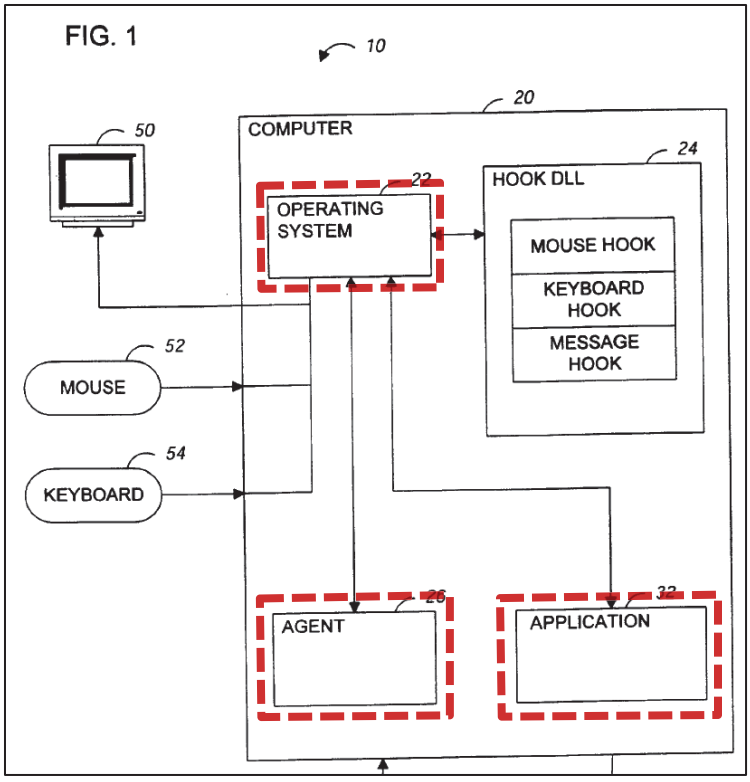
capabilities of the training agent generally as "displaying tutorial information … wherein the tutorial information comprises a video clip." (9:60-10:21 (claims 1-3)). A POSA would have recognized that opening a presentation program is one alternative description, and that another approach that Bell suggests is for the training agent itself to present the audio or video. Incorporating a presentation program's playback functionality into the training agent would have been a straightforward matter of adding audio or video to a window of the agent (Paragraph 152 below)–an enhancement that was well-known in the 2001 to 2002 timeframe (Paragraph 14-15 above; Exs. 1024, 1025, 1026). As a POSA would have recognized, incorporating the presentation program's playback functionality directly into the training agent that Bell describes would have been beneficial because it ensures that computers running the training agent will be able to display the myriad of training material types that Bell describes (5:7-18), regardless of whether users have installed particular presentation programs. Further, integrating different types of playback functionality directly into the training agent would have allowed developers to enhance the impact of the training by providing various types of materials—such as "a mixture of DVD video clips and hypertext documents,"[12] which Bell highlights as a "[p]articularly advantageous" approach to training (5:15-18)—together in a single top-level GUI element. Therefore, even if

[12] A webpage is a common type of "hypertext document."

Bell were not understood to describe playing audio or video without a presentation program, a POSA would have understood configuring the training agent to present the audio or video itself to be an obvious alternative to using a presentation program.

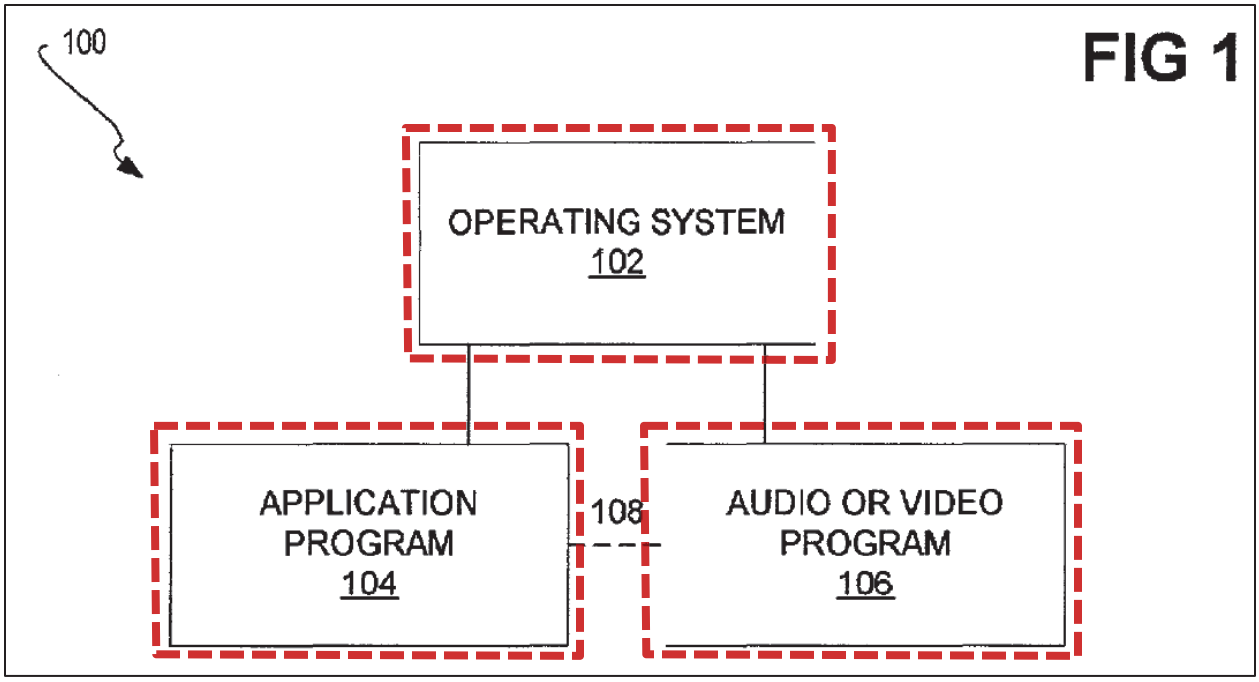
e. "[*E1*] the audio or video program separate from"

127. Bell explains that the training agent is distinct from the host. (1:61-62 ("The training agent does not include any portion of the computer program application."); 3:17-19 ("The invention can be implemented without affecting, or requiring any reprogramming of, the computer program."); 2:22-23 ("A training agent is provided to run on the same computer with the application."); 9:60-65 (claim 1: "A system … comprising: a training agent …, the training agent including no portion of the computer program application…."); 4:15-21("An application user interface associates user commands with actions to be performed by an application 32. The training agent 26 provides, in response to user actions, tutorial information with respect to a selected application. The selected application will be referred to as the host application, or simply as the host, to distinguish it from the training agent, which is also an application.")).



(Fig. 1 (excerpted above) (outlining the training agent as distinct from the host and the OS)).

128. Given this disclosure, a POSA would have understood Bell as disclosing an audio or video program (i.e., the training agent) separate from an application program (i.e., the host) in that the training agent could be executed separately from any given host application (of which Bell discloses various examples), per the meaning of "separate from" to a POSA (paragraph 49 above). Indeed, Figure 1 in Bell mirrors Figure 1 in the '591 patent, which the specification cites when describing the audio or video program as "separate from" the application program (Ex. 1001, 3:25-26):



(Ex. 1001, Fig. 1 (outlining the OS, the application program, and the audio or video program)).

f. "[*E2*] but integrated with the application program" & "[*H*] wherein the audio or video program is further integrated with the application program by subclassing into a window of the application program such that the audio or video program detects when an event related to the application program occurs."

129. Bell discloses that the training agent is integrated with the host by taking control of aspects of the host's GUI and presenting tutorial videos and audio in response to user interaction with the host's GUI elements. (1:38-44 ("The present invention features methods and apparatus for providing tutorial information for a computer program application through a training agent activated by a user of the application. The agent takes control of the application interface and performs

actions, such as finding and displaying tutorial information, in response to application user interface commands.")).

130. The training agent runs in the background until activated. (4:39-41 ("After it is started, the agent runs silently as a background process until the user activates it by pressing a predefined hotkey."); 5:63-66 ("Windows will activate the agent when the hotkey is pressed, even if another application is active. The agent runs as a background process until Windows reactivates it with a hotkey message."); 6:4-8 ("When the agent's hotkey is pressed, Windows sends a WM_HOTKEY message to the agent's Window procedure (step 130).")).

131. The training agent detects the active program, and if it recognizes the program, the training agent installs several hooks. (6:8-10 ("The agent detects the active application using the Windows GetForegroundWindowcall."); 6:6-8 ("If the agent recognizes the active application, the agent proceeds to take control of the otherwise active application, making it a host application (step 140)."); 6:18-28 ("To enable it to take control of the host, the agent installs hooks 24, which are implemented in a Windows DLL file. … The agent installs three hooks using the SetWindowsHookEx call. A message hook receives all messages directed at the host program; a mouse hook receives the host program's mouse messages; and a keyboard hook receives the host program's keyboard messages.")).

132. The mouse and keyboard hooks detect user interaction with the host application's interface. (6:18-19; 7:1-9 ("The keyboard hook watches for keyboard messages (step 180). … [*T*]he key activation and release messages are processed to determine whether they represent recognized user interface navigation shortcuts: if so, the corresponding control is recognized…."); 7:11-36 ("The mouse hook watches several different mouse messages (step 190). … A WM_LBUTTONUP message selects the window [*e.g., button*] under the mouse. … The mouse hook procedure then relinquishes control of the host program and sends a WM_USER message to the agent with the handle of the selected window.")).

133. The training agent uses subclassing to detect when a user selects a menu item (e.g., "Open" in the "File" menu, or "Copy" in the "Edit" menu). (6:49-52 ("On the first call (step 160), the message hook subclasses all windows of the host, using the EnumThreadWindows and EnumChildWindows calls to find the windows (step 170)."); 7:39-42 ("The subclass procedure described above watches WM_COMMAND messages, which are sent when a menu item is selected. When a WM_COMMAND message is received, the subclass procedure relinquishes control of the host program and sends a WM_USER message to the agent with the identifier of the selected menu item (step 200)."); Fig. 2 (Step 170: "Subclass The Host's Windows")). The subclassing also prevents the host from receiving certain user interaction when the training agent is active. (Bell 6:55-57 ("The function of

the subclass procedure is to stop messages from reaching the host's windows. Without messages, the host is cut off and the agent has control of the host.")).

134. The training agent receives messages from the mouse hook and subclass procedures describing user interactions with the host. (Paragraphs 132 and 133 above). The training agent matches the user's interaction with available tutorial information. (7:43-8:39 ("Having received information describing the selected control, the agent identifies, in the agent's data file for the host application, a control that is equivalent to the selected control (step 210). … Finding the action associated with a control is straightforward: the agent searches the list until a control is found that matches the selected control (step 220). Once the action has been found, it must be performed (step 230).")).

135. The training agent visually signals the change by altering the appearance of the mouse pointer when the host application is active. (4:43-49 ("The agent intercepts commands directed to the host application's user interface…. To signal the change of behavior, the agent changes the shape of the cursor that would otherwise be displayed when the host application is active.")).

136. Given these extensive disclosures, a POSA would have understood Bell as describing the use of subclassing to integrate an audio or video program (i.e., the training agent) with an application program (i.e., the host, such as Microsoft Word). In particular, a POSA would have understood that the training

agent was "integrated with" the host application given the relationship between the UI of FAQsoft and the UI of the target application. (Paragraph 60 above). Further, given Bell's disclosure of subclassing as discussed above, a POSA would have understood that the training agent was "further integrated with the application program [i.e., the host application] by subclassing into a window of the application program such that the audio or video program detects when an event related to the application program occurs" per the applicable understandings of "subclassing" (Paragraph 55 above), "window" (Paragraph 27 above), and the "further integrated . . ." phrase as a whole (Paragraph 57 above).

g. "[*E3*] such that the application program is unaware that the audio or video program has been integrated therewith,"

137. Bell discloses that the host was not designed a priori to be integrated with the training agent. (3:17-19 ("The invention can be implemented ***without affecting, or requiring any reprogramming, of the computer program***."); 2:19-22 (Summary of the Invention: "In general, in another aspect, the invention is directed to a method of adding help functionality to a computer program application having a user interface ***without modifying the application***."); 10:34-36 (Claim 8: "A method of adding help functionality to a computer program application having a user interface ***without modifying the computer program application***, comprising….")).

138. Furthermore, a POSA would have understood that the subclassing technique described in Bell integrates the training agent with the host without making the host aware of the integration. As discussed in Paragraph 43 above, nothing about integration with subclassing requires that the host ever know of the integration. As described in Part VI.B.1.f above, the hooking and subclassing merely monitor and intercept messages. Nothing in Bell suggests that the host would receive any new message that could inform it of the integration—let alone one that it would recognize as such. (2:13-15 ("The training agent may act without affecting the running of the application and without affecting the behavior of the application.").

139. Thus, given that a POSA would have understood that the host application was not "developed a priori" to integrate with the training agent "and also does not operate with knowledge that the integration has occurred" per the interpretation discussed in paragraph 51 above, a POSA would have understood Bell as disclosing a system in which the application program is "unaware that the audio or video program has been integrated therewith.

h. "[*F1*] wherein a user of the application program directly interacts with the application program, and"

140. Bell explains that the training agent does not affect any application program until activated. (4:39-41 ("After it is started, the agent runs silently as a background process until the user activates it by pressing a predefined hotkey.");

5:64-6:3 (“The agent runs as a background process until Windows reactivates it with a hotkey message. The presence of the agent does not noticeably degrade system performance. The agent does not monitor any windows or processes, relying instead upon the hotkey mechanism to activate or reactivate it. The agent also occupies a small amount of memory, less than 500 kilobytes.”)).

141. Bell discloses that even after the two programs are integrated, the user interacts with the GUI of the host. (4:10-19 (“The system 10 is used to run computer programs, especially application programs (applications). ***A user interacts with an application*** through a graphical user interface. Manipulations by the user of the user input devices, such as the keyboard and mouse, become commands that are sent to the application. An application user interface associates user commands with actions to be performed by an application 32. The ***training agent 26 provides, in response to user actions, tutorial information*** with respect to a selected application.”)).

142. Further, the training agent allows the user to use the host in standard ways. For example, clicking a menu passes a message to the host to open the menu. (7:19-21 (“If the mouse is located inside a menu, the button down message is passed to the subclass procedure to pull down the menu.”); Section VI.B.1.f above (discussing menus and the subclass procedure)). In addition, pressing a key on the keyboard or clicking outside any recognized control can deactivate the

training agent. (7:6-10 ("[*T*]he key activation and release messages are processed to determine whether they represent recognized user interface navigation shortcuts: … if not, the agent relinquishes control of the host program[.]"); 7:12-15 ("Any button messages outside of the expected places (such as outside the client area or in the title bar) cause the agent to relinquish control of the host program (step 240).")). Even while the training agent waits for the user to select a topic for instruction, the host continues to run. (2:13-15 ("The training agent may act without affecting the running of the application and without affecting the behavior of the application.")).

143. Given that these interactions involve the host application's GUI, a POSA would have understood them to be "direct." As discussed in Paragraph 50 above, anytime a user enters information into an application program running on the user's computer via a GUI window element of that application, a POSA would have understood the interaction to be direct.

i. "[*F2*] the user interacts with the audio or video program as though the audio or video program were part of the application program ,"

144. Bell discloses that the user interacts with the training agent as though it were part of the host, because the user's interactions with the GUI of the host cause a response from the training agent. (4:42-46 ("The ***agent intercepts commands directed to the host application's user interface***, finds tutorial

information associated with the intercepted commands, and presents the tutorial information to the user."); 2:51-54 ("The user can select an aspect, function or feature of the host application by clicking the mouse on a user interface control. In response to the control selection, the agent performs an action."); 6:12-17 ("For example, if the File | Save menu item is selected while the agent is in control, the host will not save a file. Instead, the agent will receive the message and look up the File | Save menu item in its database, and will perform some appropriate action, such as displaying a video clip on saving files.")).

145. Given these descriptions, a POSA would have understood that the UI of the host controls the training agent. As discussed in paragraph 59, a POSA would have understood that a user "interacts with the audio or video program as though the audio or video program were part of the application program" provided that the UI are related in some way. Thus, a POSA would have understood Bell as disclosing a system in which a user interacts with an audio or video program (i.e., the training agent) as though it were part of the application program (i.e., the host).

j. "[*G*] wherein the processor executes the operating system, the application program, and the audio or video program,"

146. As discussed in Sections VI.B.1.a though VI.B.1.d above, Bell discloses a computer with a processor, an OS running on the computer, an application program referred to as the host running on the computer, and an audio

or video application called the training agent running on the computer. (3:63 ("running the Windows® 95 operating system"); 4:10-20 ("run computer program[] … referred to as the host"); 2:23-24 ("training agent … run on the same computer")). A POSA would have understood these disclosures to mean that the processor executes (i.e., runs) the operating system, the host, and the training agent.

2. Claim 2: "The system of claim 1, wherein the audio or video program modifies contents of a window of the application program created through the operating system."

147. Bell explains that the training agent modifies the contents of a window of the host by changing the window's behavior. (6:4-17 ("When the agent's hotkey is pressed, Windows sends a WM_HOTKEY message to the agent's window procedure (step 130). If the agent recognizes the active application, the agent proceeds to take control of the otherwise active application, making it a host application (step 140). … When the agent has control of the host, the host receives no mouse or keyboard messages, and will therefore not behave normally. For example, if the File | Save menu item is selected while the agent is in control, the host will not save a file. Instead, the agent will receive the message and look up the File | Save menu item in its database, and will perform some appropriate action, such as displaying a video clip on saving files."); 6:55-57 ("The function of the subclass procedure is to stop messages from reaching the host's

Windows. Without messages, the host is cut off and the agent has control of the host.")).

148. A POSA would have recognized that abnormal behavior in response to user interactions will modify a window of the host relative to its appearance had the host behaved normally. For example, clicking the "File | Save" menu item in Microsoft Word would normally result in Microsoft Word displaying a saved document (e.g., changing the title of the window to reflect the file name). However, if the training agent is active, clicking the "File | Save" menu item "will not save a file" (6:14), and instead the training agent will perform some action. Microsoft Word's window has been modified.

149. Thus, given how a POSA would have understood "window" (paragraph 27 above) and also given that a POSA would have understood that "the audio or video program modifies contents of a window of the application program created through the operating system" if the programs are running on Microsoft Windows and the audio or video program alters the appearance of a GUI window element associated with the application program (paragraph 62above), a POSA would have understood Bell to disclose a system in which the audio or video program [i.e., the training agent] modifies contents of a window of the application program [i.e., the host application] created through the operating system [i.e., a version of Microsoft Windows]."

150. Furthermore, Bell also suggests other ways to modify a window of the host. For example, when choosing where to display the video tutorial content, a POSA implementing Bell would have two choices: either display the video within a window of the host or display the video in some other window. Bell's example of using a presentation program (5:4-5) illustrates a way to do the latter. However, just as Bell recognizes that tutorial information can be "part of the application program" (1:8-10), a POSA would have recognized that the tutorial information could be presented as part of the host's window. In particular, a POSA would have understood this as a possibility given Bell's emphasis that "the invention is directed to a method of adding help functionality to a computer program application having a user interface…." (2:20-22). In addition, a POSA would have recognized the desirability of presenting tutorial information as part of the host's window given that incorporating video to the host GUI element would have had the benefit of increasing the proximity between the tutorial information and the user's selection. Presenting the tutorial information (e.g., a "video clip on saving files" (6:16-17)) alongside the selection (e.g., the place onscreen where the user selected the "File/Save menu item" (6:13-14)) would have ensured Bell's goal of creating an "intuitive mechanism of providing tutorial information." (3:13-17). In particular, incorporating tutorial information into the host's window ensures that the tutorial information remains "related to the user interface." (3:30). A POSA

would have recognized that presenting video tutorials as part of the host window is a particularly advantageous feature in the case of training agents that, as Bell discloses, "provide training for multiple applications simultaneously." (2:16-17). By presenting videos as part of the host window, one would avoid any confusion as to which application a given tutorial relates. Thus, given the finite number of options, the benefit of presenting the tutorial information in a window of the host, and the known means to accomplish doing so (*see* Section IV.E above), a POSA would have understood from Bell that adding video to a window of the host was an obvious choice.

151. A POSA would have understood from Bell's identification of Windows OS (*see* Section VI.B.1.b above) that the OS creates the host's windows. (Paragraph 25 above (noting that Windows OS creates windows when instructed by a program)). The '591 patent's only example of integration by subclassing concerns Microsoft Outlook Express and is specific to Windows OS, as discussed in Paragraph 12.b above). A POSA reviewing this example would have understood Windows OS as creating Microsoft Outlook Express's windows.

3. Claim 3: "The system of claim 1, wherein the audio or video program runs in a window created through the operating system and related to a window of the application program created through the operating system."

152. Bell discloses that the training agent runs in at least one window. When the mouse hook or subclass procedure detect user interaction with the host's

GUI that matches available tutorial information, that procedure sends a message (as discussed in Part IV.C above) to the agent describing the user interaction. (7:34-42 ("The ***mouse hook procedure*** then … ***sends a WM_USER message to the agent***.... [*T*]he ***subclass procedure … sends a WM_USER message to the agent***…."); 6:40-47). The agent receives the message. (7:43-46; 6:40-47). As discussed in Paragraph 29 above, to be the destination for a message on Windows OS, the training agent must have a window, even if it is hidden from view. A POSA would have understood that the training agent's window is related to the window of the host, because the training agent's window is the destination of the message indicating the user's interaction with the UI (i.e., either on or itself a window) of the host.

153. Thus, given how a POSA would have understood "window" (paragraph 27 above) and also given that a POSA would have understood that "the audio or video program runs in a window created through the operating system and related to a window of the application program created through the operating system" if one or more tasks performed in the audio or video program's window relate to the application program's window in some way (paragraph 63 above), a POSA would have understood Bell to disclose a system in which "the audio or video program [i.e., the training agent] runs in a window created through the operating system [i.e., a version of Microsoft Windows] and related to a window of

the application program [i.e., the host application, such as Microsoft Word] created through the operating system."

154. Furthermore, Bell also suggests other ways that the training agent could run in a window. For example, as discussed in Paragraph 126 above, a POSA would have understood that configuring the training agent to present the video tutorial content itself to be a straightforward and advantageous alternative to using a presentation program. A POSA configuring the agent to present the video tutorial content itself would have recognized two readily apparent options for achieving this desirable result: First, the agent could present the video tutorial content in a window GUI element of the host. This option is discussed in Paragraph 150 above. Second, the agent could present the video tutorial content in its own window GUI element that the training agent runs in. (*See* Paragraph 22 above). The agent creates the window GUI element by simply invoking an Window OS API instruction (`CreateWindow`), as discussed in Paragraph 25 above. A POSA would have recognized that this second option mirrors the presentation program's functionality of opening a new window GUI element in which the video tutorial content is presented (*see* 2:3-9; 8:31-56), while still securing the benefits of the agent presenting the video tutorial content itself (Paragraph 126 above). Further, as a POSA would have appreciated, there are circumstances under which it is desirable to present tutorial content in a separate

window from the host application—particularly in the case of host applications that have intricate GUIs that could be obscured by a tutorial video, thus reducing the value of the tutorial to the user. Indeed, there are situations in which it is advantageous to allow users to shift their attention back and forth between the tutorial and the host application's unobscured interface. Given these advantages of both approaches (discussed herein and, in paragraph, 150 above), a POSA would have seen the choice between presenting the video tutorial content in the window GUI element of the host or in a new window GUI element of agent as a simple matter of design choice. Further, as previously discussed, both options would have been routine for a POSA to implement. Therefore, a POSA would have understood that configuring the training agent to run its own window to be an obvious alternative.

155. A POSA would have understood from Bell's identification of Windows OS (*see* Section VI.B.1.b above) that the OS creates the windows of the training agent and the host. (Paragraph 25 above (noting that Windows OS creates windows when instructed by a program)). Furthermore, since the '591 patent's only example of integration by subclassing is specific to Windows OS (Paragraph 12.b above), Windows OS creates the windows in Bell to same extent it creates those in the '591 patent example.

4. Claim 4: "The system of claim 1, wherein the application program comprises one of: an email program, a presentation program, a publishing program, a word processing program, a spreadsheet program, an instant messaging program, a telephony program, and a gaming program."

156. Bell discloses using the training agent with Microsoft Word, which a POSA would have recognized as a popular word processing program. (7:58).

5. Claim 5: "The system of claim 1, wherein the audio or video program comprises one of: an audio or video player program, an audio or video recorder program, an audio or video player-and-recorder program."

157. As described in Section VI.B.1.d above, the training agent presents video, with or without audio, and is therefore either "an audio or video player program" or "an audio or video player-and-recorder program." (2:12-14; 2:29-30; 3:38-42; 4:17-19; 5:15-18; 6:14-17; 9:60-10:21). Thus, a POSA would have considered the training agent to constitute "an audio or video player program" or "an audio or video player-and-recorder program" and thus would have understood Bell as disclosing such a program.

6. Claim 6: "The system of claim 1, wherein the audio or video program comprises one of: an audio-only program, a video-only program, and an audio-and-video program."

158. As described in Section VI.B.1.d above, the training agent plays video, with or without audio. (2:12-14; 2:29-30; 3:38-42; 4:17-19; 5:15-18; 6:14-17; 9:60-10:21). Thus, a POSA would have considered the training agent to

constitute at least "a video-only program" or "an audio-and-video program" and thus would have understood Alhadad as disclosing such a program.

C. Ground 4: Mast Discloses Each Limitation of Claims 1-6 of the '591 Patent

159. According to the face of the document, Mast (Ex. 1005)[13] is a U.S. patent issued on April 8, 1997. Mast discloses a way that one program can use subclassing to attached a small video to the window of another program, such as word processing program or spreadsheet program.

160. After reviewing Mast and claims 1-6 of the '591 patent, it is my opinion that Mast discloses every limitation of the claims. The basis for my opinion and the details of my analysis are below.

1. Claim 1: "A system for integrating an audio or video program with an application program comprising:"

161. Mast discloses a system for integrating an audio or video program with an application program. (4:14-16 ("A method and apparatus for associating an image display area with an application display area in a computer system are described."); *id.* Abstract ("A method for adding a photograph or other still or animated image to any application program in its own window….")). The elements of the preamble are discussed thoroughly in Sections VI.C.1.c to VI.C.1.g below.

[13] Unless otherwise indicated, all citations in Section VI.C are to Ex. 1005.

a. "[*A*] hardware including a processor;"

162. Mast discloses a computer. (13:5-6 ("***a computer*** implementing the present invention"); 281:2-4 (claim 1: "Apparatus … comprising … ***processing means***…."); 4:15 (A[*n*] … apparatus for associating … in a computer system are described."); 1:11 ("Computer systems … are commonly used in business and personal settings.")).

163. Mast further explains that the computer runs the Windows OS operating system, as discussed in Section VI.C.1.b below. A POSA would have understood Mast's disclosure of a computer running Windows OS to describe hardware including a processor. Any computer must include a processor in order to function and run an operating system such as Windows OS. Further, the processor must be—or run on—hardware.

b. "[*B*] an operating system;"

164. Mast discloses the Windows OS operating system. (Abstract ("A method ... in a windowed computer operating environment such as ***Microsoft Windows 3.1*** is presented."); 1:11-12 ("Computer systems using multi-window operating systems are commonly used in business and personal settings."); 5:18-20 ("FIG. 1 illustrates one possible embodiment of the present invention in a windowing environment such as ***Microsoft Windows***™ ('Windows')."")).

165. A POSA would have understood "Microsoft Windows 3.1" to be an operating system, as discussed in Paragraph 16 above. The '591 patent identifies "Microsoft Windows OS" as an example of an operating system that can be used in connection with the purported invention. (Ex. 1001, 3:12-14; *see also* Paragraph 45 above).

c. "[*C*] the application program running on the operating system; and,"

166. Mast discloses various programs running on Windows OS, including what Mast calls the "window application" or the "subclassed application," depending on whether the subclassing described in Section VI.C.1.f below has occurred. (2:54-57 ("The present invention relates to … ***any application program*** running in its own window in a windowed computer operating environment."); 5:27 ("Virtual Machine VM1 contains ***Window Applications***…."); 7:39-44 ("As described above, when a window application is opened, the application is subclassed by the shell hook filter functions … ***the subclassed application***."); 281:2-8 (claim 1: "Apparatus … comprising … at least one application program ***running*** in said windowed operating environment…."); 1:12-14 ("These multi-window operating systems permit more than one application to be ***running*** in a system at one time….").

167. As examples, Mast explains that "subclassed application" may be a word processing program or a spreadsheet program. (5:4-9 ("FIG. 14B illustrates a

sample windowed display of one embodiment of the present invention wherein each application window has an image attachment window attached to its caption bar. All elements of FIG. 14B are identical to those of FIG. 14A except for the addition of attachment window images 1470 and 1475."); 1:20-22 ("FIG. 14A shows a typical prior art display screen 1400 containing two windows 1410 and 1420. Window 1410 is associated with a ***word-processing program***."); 12:51-52 (disclosing that "the application" may be a "***spreadsheet***[] or text editor[]")).

168. A POSA would have understood a windows application or subclassed application, such as a word processing program or a spreadsheet program, to all be application programs running on the OS given that each is a computer program that is distinct from the OS, per the meaning that "application program" would have had to a POSA (paragraph 48 above). In addition, the '591 patent identifies word processing and spreadsheet programs as examples of application programs that can be used in connection with the purported invention. (Ex. 1001, 2:6-8; 6:21-33; 7:41-51).

d. "[*D*] the audio or video program running on the operating system,"

169. Mast discloses a program called "Attacher.EXE," or the "image attachment application," which runs on the operating system and operates in conjunction with "Attacher.DLL." (6:5-22 ("Attacher.DLL is installed by Windows…. A 'LibMain' function is called when the DLL is loaded. … Once

loaded, the library performs an initialization function. … [*T*]he initialization function starts the ***image attachment application, Attacher.EXE***."); 281:38-50 (claim 6: "The apparatus … comprising an attachment application running in said windowed operating environment…."); Fig. 2 ("Start Attacher.EXE"); 6:22-26 ("Attacher.EXE provides the user interface to Attacher.DLL and performs all image file processing on behalf of the Windows-based application with which an image attachment window is associated")).

170. As discussed in paragraph 18 above, application programs that run on the Windows OS are stored on a computer as executable files and often have a name that ends in ".EXE". A POSA would have understood Attacher.EXE to refer to a program.

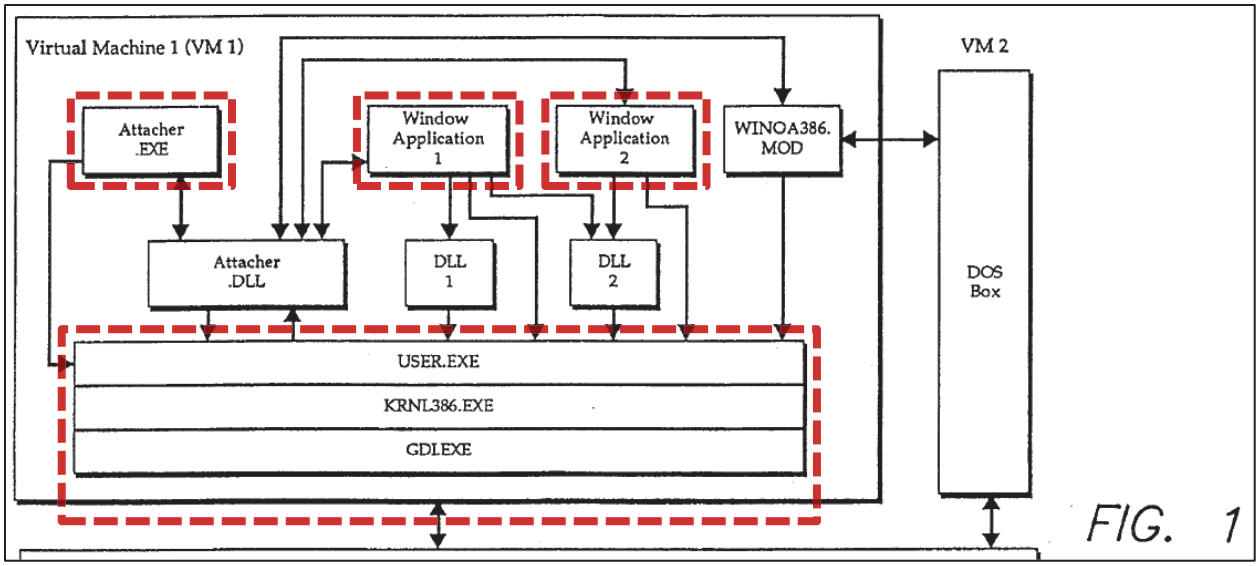
171. Mast explains that Attacher.EXE plays video. (281:38-50 (claims 6 & 9: "The apparatus … comprising an attachment application … providing an interface … to select an object to be displayed …, said object being selected from a plurality of displayable objects … wherein said ***displayable objects comprise video files***."); Abstract ("A method for adding a photograph or other still or ***animated image*** to any application program…."); 2:54-56 ("The present invention relates to a method for adding a photograph or other still or ***animated image*** to any application program running in its own window…."); 3:22-24 ("[*T*]he ***present invention can be used to display a video file*** concurrently with an application

program in the application program's window. The ***user activates the brief motion video*** presentation at the user's personal convenience by such means as pressing a designated mouse button when the cursor (or pointer) is positioned over the image.")). A POSA would have understood an "animated image" to be a video.

172. Because Mast describes Attacher.EXE as capable of playing back video, a POSA would have understood Mast to disclose an audio or video program (i.e., Attacher.EXE) running on the OS within the meaning of "audio or video program" to a POSA (Paragraph 47 above).

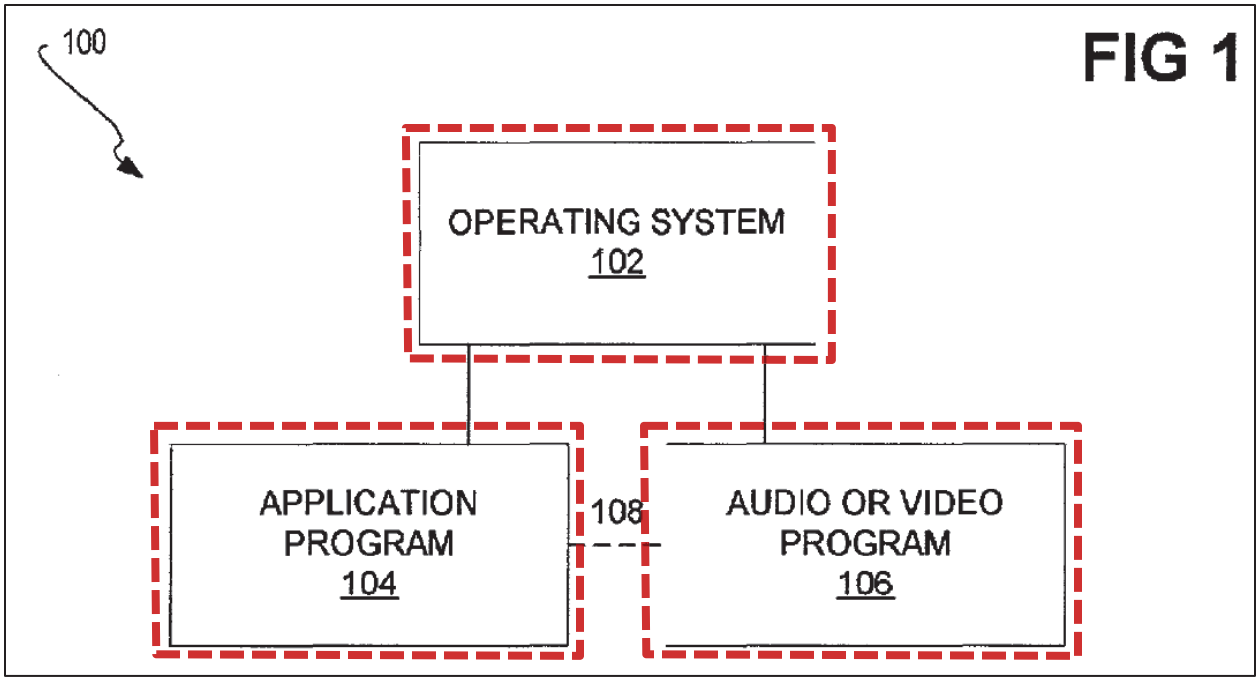
e. "[*E1*] the audio or video program separate from"

173. Mast explains that Attacher.EXE is distinct from the "subclassed application." (5:18-31 ("FIG. 1 illustrates one possible embodiment of the present invention…. Virtual Machine VM1 contains Window Applications 1-2 [*e.g., the "subclassed application"*]…. An executable file ..., Attacher.EXE … [*is*] also present to provide the capabilities of the present invention."); see also Paragraph 18 above (explaining that programs are stored on Windows OS as executable files).



(Fig. 1 (excerpted) (highlighting Attacher.EXE as distinct from Window Applications 1 and 2 and the OS—part of which is located in lower box)).

174. Given this disclosure, a POSA would have understood Mast as disclosing an audio or video program (i.e., Attacher.EXE) separate from an application program (i.e., the window application or the subclassed application) in that Attacher.EXE could be executed separately from the application program (of which Mast discloses various examples), per the meaning of "separate from" to a POSA (paragraph 49 above). Indeed, Figure 1 in Mast mirrors Figure 1 in the '591 patent, which the specification cites when describing the audio or video program as "separate from" the application program (Ex. 1001, 3:25-26):



(Ex. 1001, Fig. 1 (outlining the OS, the application program, and the audio or video program).

f. "[*E2*] but integrated with the application program"& "[*H*] wherein the audio or video program is further integrated with the application program by subclassing into a window of the application program such that the audio or video program detects when an event related to the application program occurs."

175. Mast discloses that the audio or video program is integrated with the application program by attaching an "image attachment window" to a window of the application program. (*See generally* 4:14-50).

176. Attacher.EXE runs before Windows OS opens any programs. (6:5-11 ("Attacher.DLL is installed by Windows prior to Windows loading and running any Windows-based application … [*and*] performs an initialization procedure.");

6:21-26 ("[*T*]he initialization [*procedure*] starts the image attachment application, Attacher.EXE. Attacher.EXE provides the user interface to Attacher.DLL and performs all image file processing on behalf of the Windows based application with which an image attachment window is associated.").

177. Attacher.DLL installs a hook to monitor the creation of new windows. (6:15-19 ("Block 202 follows with the installation of a shell hook into the operating environment so that Attacher.DLL can be made aware of the creation of top level windows."); 6:43-47 ("The shell hook installed at step 202 in FIG. 2 allows tasks to be notified when top level windows are being created. 'Hooks' are resources that install filter functions. These filter functions process 'hooked' function calls before the functions that have been hooked are called.").

178. Attacher.DLL uses the hook to determine if a window application's window is eligible for an "image attachment window." (7:1-14 ("In block 401, Attacher.DLL determines whether a top level window is being created for the starting application. … In this embodiment of the present invention, no image is attached to popup windows, therefore, the filter functions permit the application startup to resume if the window being opened is a popup window.")).

179. Attacher.DLL then subclasses windows that are eligible for an image attachment window. (7:11-14 ("If the window being created is not a popup window, in block 403, the Attacher.DLL subclass procedure subclasses the

application for which the window is being created…."); 6:49-52 ("Subclassing allows Attacher.DLL to screen messages to a subclassed application before those messages are received by the subclassed application."). A POSA would have understood subclassing the "application" to refer to subclassing a window of the application, as described in Part IV.D above.

180. Once Attacher.DLL subclasses the "subclassed window," Attacher.EXE creates an "image attachment window." (7:15-36 ("Once the application has been subclassed, … a message is posted to Attacher.EXE to select and open an appropriate image file. In block 406, the image attachment window is created for the starting application.")).

181. The "image attachment window" is attached to the window of the subclassed application. (Mast 2:54-64 ("The present invention relates to a method for adding a photograph or other still or animated image to any application program running in its own window in a windowed computer operating environment. The present invention allows any bitmapped or other image to be "attached" to any application program capable of running in a windowed environment, without any modification to such application program, such that when the application program is started and its window is opened, the image is displayed at a predetermined location with respect to the application window."); 4:35-41 ("When the window for the application is re-sized or moved, the position

of the floating window is changed to correspond to the new size and/or position of the application program window, such that when the application program window is displayed in its new size and/or position, it appears that the floating window was fixed to the application program window and moved along with the application program window."); 5:4-7 ("FIG. 14B illustrates a sample windowed display of one embodiment of the present invention wherein each application window has an image attachment window attached to its caption bar.")).

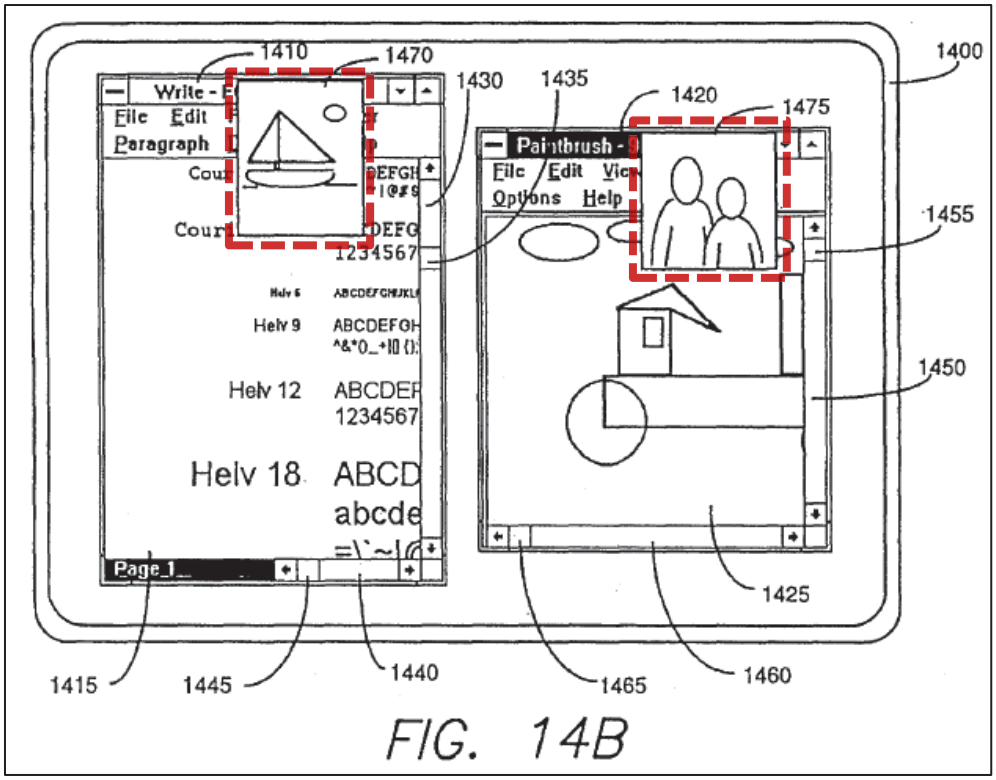
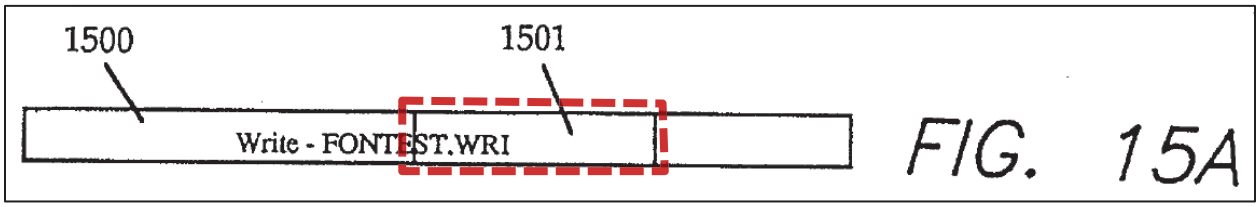


FIG. 14B

(Fig. 14B (images in the "image attachment windows" outlined)).

182. The image attachment window may also provide a button in the application program's window to show and hide the image and/or video. (Mast 4:51-65 ("In one embodiment, the present invention also provides a method for a

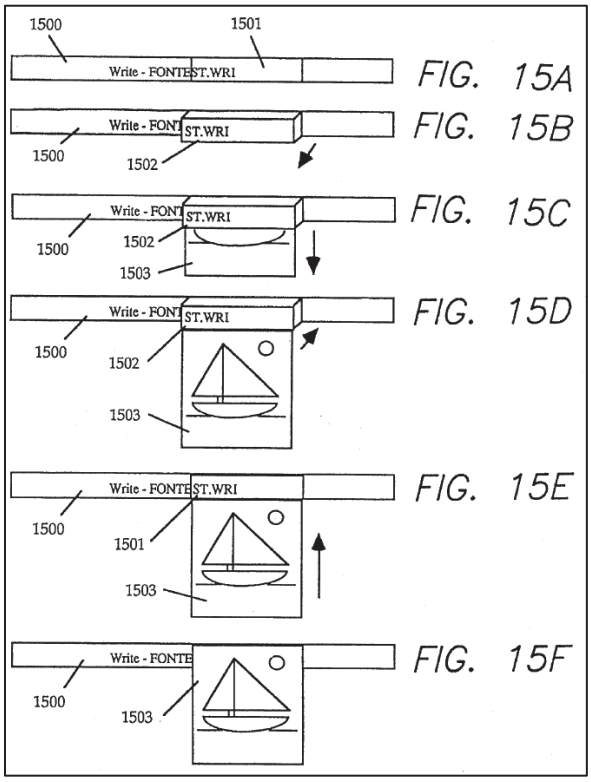
user to remove the image 'attached' to an application from the display by positioning a cursor (also referred to as a pointer) over the attached image and 'clicking.' … To re-display the image, the cursor is positioned over the shaded portion of the caption bar and 'clicked,' and the drawer opens and the image emerges."); 9:55-58 ("FIG. 15A shows the caption bar 1500 with caption drawer 1501 in closed position. The caption drawer can be opened by such means as the user placing the mouse pointer on the caption drawer and clicking the left mouse button.")).



(Fig. 15A (closed caption bar drawer susceptible to clicking outlined)).

183. Mast explains that the image may appear to roll up into the "caption drawer" using an animated sequence. (4:55-63 ("When this is done, this embodiment displays an animated sequence that creates the illusion of the portion of the caption bar to which the image is attached opening like a drawer. The image 'rolls up' into the opened drawer, and the drawer closes. The portion of the caption bar that forms the drawer (which is slightly wider in width than the width of the image) is displayed in a different shading from the remainder of the caption

bar to identify the position of the drawer."); 9:45-46 ("FIGS. 15A-F illustrate one embodiment of the caption drawer animation sequence as outlined above.")).



(Fig. 15A-F (showing the animated sequence)).

184. The user may also arrange the "image attachment window" horizontally along the caption bar. (4:66-5:1 ("In one embodiment, the image may be dragged by the user to any desired location horizontally along the caption bar, although it is restrained from moving vertically."); 13:16-17 (noting that if the "image attachment window" obscures something the user wants to see, the "[*t*]he image attachment window can be moved out of the way.")).

185. The "image attachment window" may also be configured to automatically hide in certain circumstances. (13:7-25 ("If, during the course of a

user session, the cursor in the application window passes beneath the image attachment window (e.g., the cursor advances due to typing in a text editor), the application remains active though the active application cursor may be visually obscured by the image attachment window. To be as unobtrusive as possible, the present invention allows this visual obstruction of the underlying application window to be handled in a variety of ways. For example: … The user can place the mouse pointer over the image attachment window and not press any mouse buttons. Eventually, the window will roll up or snap up (e.g., the image attachment window can be configured to roll up or snap up after about three quarters of a second).")).

186. Attacher.DLL uses subclassing to monitor user interactions with the image attachment window. (6:49-52 ("Subclassing allows Attacher.DLL to screen messages to a subclassed application before those messages are received by the subclassed application."); 7:41-44 ("Therefore, messages sent to these applications are first sent to Attacher.DLL. The subclass procedure is then able to inspect the messages prior to their reaching the subclassed application."); 7:47-56 ("Image File Loaded and Roll Up/Roll Down Timer messages are initially posted to the subclassed application with which the relevant attachment window is associated although they are really intended for Attacher.DLL. By posting them as messages to the subclassed application, however, Attacher.DLL is able to use the handle in

the message indicating the subclassed application to identify the appropriate image attachment window to which the message relates.")).

187. Attacher.DLL sorts messages directed to a "subclassed application" based on whether they pertain to an "image attachment window," and forwards on those that do not. (7:44-49 ("Messages that are not of interest to the subclass procedure are passed along to the subclassed application. Otherwise, the message is processed by the subclass procedure. In some cases, messages will not be passed to the subclassed application. For instance, Image File Loaded and Roll Up/Roll Down Timer messages…."); 7:60-8:8 ("In FIG. 5, the subclass procedure is initiated by interception of a message directed to a subclassed application in block 500. … If the message is of import to Attacher.DLL, the message is processed by Attacher.DLL in block 503. If the message was originally intended for the subclassed application, the message is then passed to the subclassed application.")).

188. Based on what it receives, Attacher.DLL sends instructions to Attacher.EXE to alter the "image attachment window." (Mast 8:9-18 ("In FIG. 6, the procedure for handling an 'Image File Loaded' message within the subclass procedure is illustrated. The Image File Loaded message notifies the subclassed application that Attacher.EXE has opened an image file and that the image file has been prepared to be displayed. … In block 600, the Image File Loaded message is

received by the subclass procedure and selected as a message of interest. … In block 605, the width and position of the existing image attachment window are changed to match the new image file. Then, in block 606, a message is posted to the subclassed application to roll down the window."). Attacher.DLL would similarly intercept the roll down message directed to the "subclassed application" and then send instructions to Attacher.EXE to perform the roll down animation. (8:58-9:25).

189. Given these disclosures, a POSA would have understood Mast as describing the use of subclassing to integrate an audio or video program (i.e., Attacher.EXE) with an application program (i.e., the window application or subclassed application). In particular, a POSA would have understood that Attacher.EXE was "integrated with" the window application given the relationship between the UI of Attacher.EXE and the UI of the window application. (Paragraph 60 above). Further, given Bell's disclosure of subclassing as discussed above, a POSA would have understood that Attacher.EXE was "further integrated with the application program [i.e., the window application] by subclassing into a window of the application program [i.e., what Mast specifically calls the "subclassed application] such that the audio or video program detects when an event related to the application program occurs" per the applicable understandings of "subclassing"

(Paragraph 55 above), "window" (Paragraph 27 above), and the "further integrated . . ." phrase as a whole (Paragraph 57 above).

g. "[*E3*] such that the application program is unaware that the audio or video program has been integrated therewith,"

190. Mast discloses that the "subclassed application" was not designed a priori to be integrated with Attacher.EXE. (1:46-50 ("[*P*]rior art systems do not allow images to be 'attached' by a user to any application program without modification of the application program and without requiring any special capabilities of the application program."); 2:57-61 ("The present invention allows any bitmapped or other image to be 'attached' to any application program capable of running in a windowed environment, without any modification to such application program….")).

191. Furthermore, a POSA would have understood that the subclassing and hooking techniques described in Mast integrate Attacher.EXE with the application program without making the application program aware of the integration. As discussed in Paragraph 43 above, nothing about integration with subclassing requires that the host ever know of the integration. Further, as described in Section VI.C.1.f, the hooking and subclassing merely monitor and intercept messages. Nothing in Mast suggests that the application program would receive any new message that could inform it of the integration—let alone one that it would

recognize any such message as indicative of integration of another application. In addition, Mast emphasizes that "[i]n some cases, messages will not be passed to the subclassed application. For instance, Image File Loaded and Roll Up/Roll Down Timer messages…." (7:47-49).

192. Thus, given that a POSA would have understood that the subclassed application was not "developed a priori" to integrate with Attacher.EXE "and also does not operate with knowledge that the integration has occurred" per the interpretation discussed in paragraph 51 above, a POSA would have understood Mast as disclosing a system in which the application program is "unaware that the audio or video program has been integrated therewith.

h. "[*F1*] wherein a user of the application program directly interacts with the application program, and"

193. Mast discloses that even after the two programs are integrated, the user interacts directly with the "subclassed application" by giving it commands or entering text through a window GUI element of the subclassed application. For example, the user can move and resize the window of the "subclassed application." (4:34 ("When the window for the application is re-sized or moved….")). Likewise, the user can minimize the window of the "subclassed application." (7:63-64). In addition, the user can enter text in the "subclassed application." (13:7-10 ("If, during the course of a user session, … the cursor advances due to typing in a text editor….")). Further, the user can close the "subclassed

application." (10:20-26 (noting that Attacher.DLL "releases the subclassed application")). As a POSA would have understood, all of these actions are taken in response to a user interacting with the application via a GUI window element of that application. As discussed in Paragraph 50 above, anytime a user enters information into an application program running on the user's computer via a GUI window element of that application, a POSA would have understood the interaction to be direct.

194. Mast also emphasizes that the content attached by Attacher.EXE does not interfere with the application program's operation. (2:67-3:3 ("The displayed image, though 'attached' to the application program's window, does not interfere to any significant extent with the operation of the application program.")).

195. In light of these disclosures, a POSA would have understood Mast to disclose a system in which users of the application program directly interact with the application program. This direct interaction is facilitated by the operation of Attacher.DLL in passing along messages for the subclassed application that do not relate to the "image attachment window." (7:41-46 ("Therefore, messages sent to these applications are first sent to Attacher.DLL. … Messages that are not of interest to the subclass procedure are passed along to the subclassed application."); *id*. 8:6-8 ("If the message was originally intended for the subclassed application, the message is then passed to the subclassed application."); 10:28-41

("Attacher.DLL receives the Left Mouse Button Down in Non-Client Area message intended for the subclassed application. … If the mouse pointer is not currently over the appropriate portion of the caption bar, Attacher.DLL assumes it is a message for the subclassed application window and permits the subclassed application to interpret the message.")).

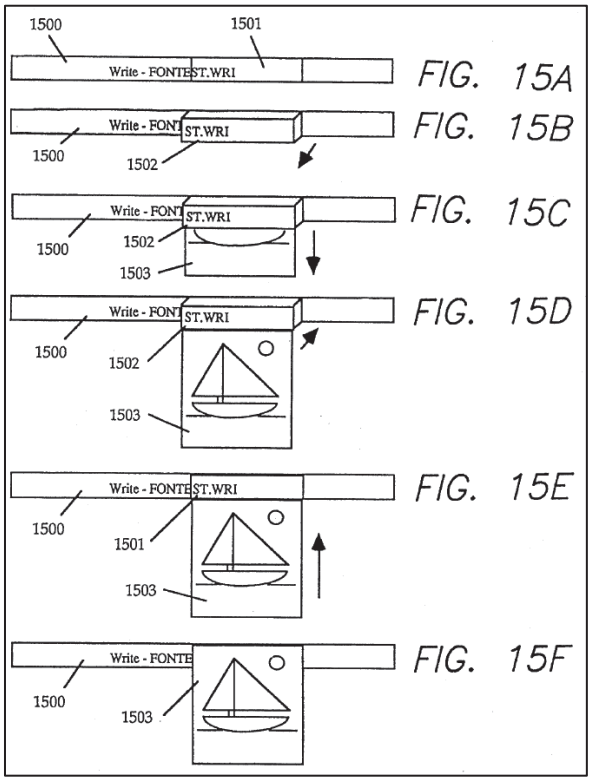
i. "[*F2*] the user interacts with the audio or video program as though the audio or video program were part of the application program,"

196. Mast discloses that the user interacts with Attacher.EXE (and Attacher.DLL) as though the "image attachment window" were part of the "subclassed application" under the applicable understanding to a POSA (discussed in paragraph 59 above) because the "image attachment window" adds functionality to the "subclassed application" that is available to the user at the same time as the user uses the preexisting functionality of the "subclassed application." (5:4-17; 7:35-37). In this way, video can be "attached to" the application program, allowing the application to be "customized and personalized." (Abstract; 2:57-60).

197. The "image attachment window" supplements the "subclassed application." (2:54-59 ("The present invention relates to a method for adding a photograph or other still or animated image to any application program running in its own window in a windowed computer operating environment. The present invention allows any bitmapped or other image to be "attached" to any application

program capable…."); 2:67-3:3 ("The displayed image, though "attached" to the application program's window, does not interfere to any significant extent with the operation of the application program.")).

198. Further, the user can interact with the "image attachment window" while using the "subclassed application." As discussed in Section VI.C.1.f above, the user instructs the "image attachment window" to open, close, and move directly in the window of the "subclassed application." For example, the user can trigger the animated sequence while using the "subclassed application." Figures 15A-F in Mast depict the animation sequence:



See also Paragraphs 184 and 185 above (discussing that the user may move and hide the "image attachment window" while using the "subclassed application").

199. Furthermore, Mast describes how certain interactions with the window of the "subclassed application" simultaneously operate on the "image attachment window." (4:34-41 ("When the window for the application is re-sized or moved, the position of the floating window is changed to correspond to the new size and/or position of the application program window, such that when the application program window is displayed in its new size and/or position, it appears that the floating window was fixed to the application program window and moved along with the application program window."); 12:21-45 (discussing how the "image attachment window" automatically adjusts to a window minimizing)).

200. Given these disclosures, a POSA would have understood that the UI of the audio or video program (i.e., Attacher.EXE) is related to the UI of the application program and thus that a user interacts with the audio or video program as though the audio or video program were part of the application program. As discussed in paragraph 59 above, a POSA would have understood that a user "interacts with the audio or video program as though the audio or video program were part of the application program" provided that the UI are related in some way.

j. "[*G*] wherein the processor executes the operating system, the application program, and the audio or video program,"

201. As discussed in Sections VI.C.1.a through VI.C.1.d above, Mast discloses a computer with a processor, an operating system running on the

computer, an application program referred to as the "subclassed application" running on the operating system, and an audio or video program called Attacher.EXE running on the operating system. (Mast 5:19-20 ("Microsoft Windows™"); 2:55-56 ("any application program running"); 6:21-22 ("initialization function starts … Attacher.EXE")). A POSA would have understood these disclosures to teach that the processor executes (i.e., runs) the operating system, the "subclassed program," and Attacher.EXE.

2. Claim 2: "The system of claim 1, wherein the audio or video program modifies contents of a window of the application program created through the operating system."

202. Mast explains that Attacher.EXE modifies the contents of a window of the "subclassed application" by adding the "image attachment window." As described in Section VI.C.1.f above, Mast discloses how the "image attachment window" adds a drawer button to the caption bar. (4:51-65).

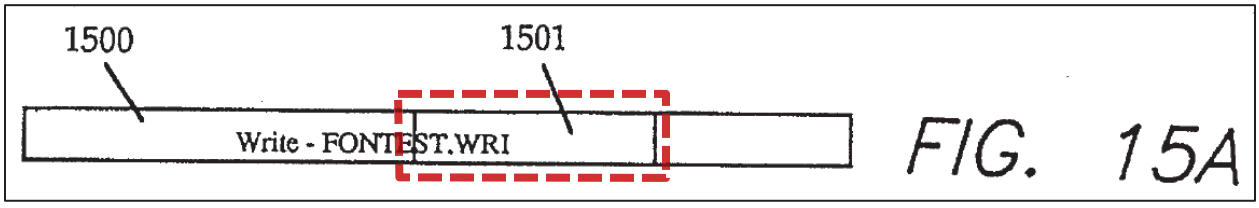


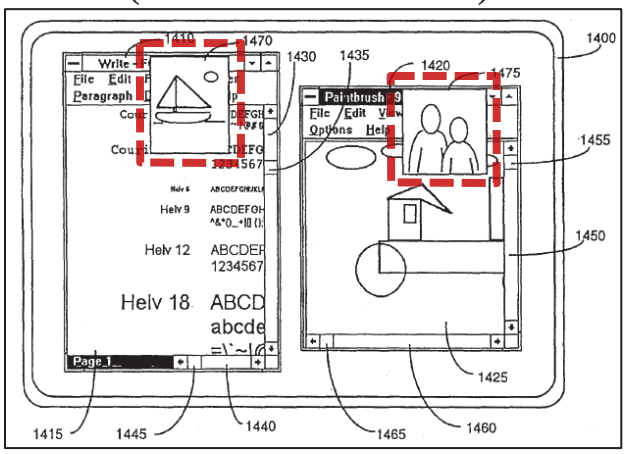
FIG. 15A

(Fig. 15A (closed caption bar drawer susceptible to clicking outlined)).

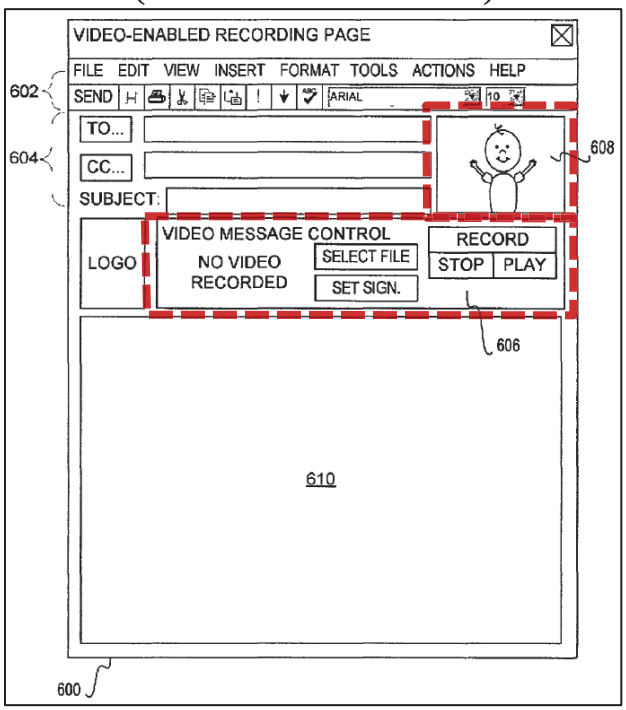
203. Further, in addition to incorporating the drawer button and thus modifying the visual appearance of the application program, the "image attachment window" is itself can be integrated as a child of the "subclassed application"

window. (5:4-7; 12:13-17 ("In order to provide more seamless operation, the image attachment window can be temporarily changed from a popup window to a child window during a move so that it is moved in the same operation that moves the application window.")). There is a comparable disclosure in the '591 patent, as can be seen by comparing Figure 14B of Mast with Figure 6 of the '591 patent:

Mast Fig. 14B
(modifications outlined):



'591 Fig. 6
(modifications outlined)



Based on the foregoing, a POSA would have understood Mast as disclosing a system in which the audio or video program (i.e., Attacher.EXE) modifies contents of a window of the application program (i.e., the subclassed application).

204. A POSA would have understood from Mast's identification of Windows OS (*see* Section VI.C.1.b above) that the OS creates the subclassed application's windows. (Paragraph 25 above (noting that Windows OS creates

windows when instructed by a program)). Furthermore, the '591 patent's only example of integration by subclassing concerns Microsoft Outlook Express and is specific to Windows OS, as discussed in Paragraph 12.b above). A POSA reviewing this example would have understood Windows OS as creating Microsoft Outlook Express's windows.

205. Thus, given how a POSA would have understood "window" (paragraph 27 above) and also given that a POSA would have understood that "the audio or video program modifies contents of a window of the application program created through the operating system" if the programs are running on Microsoft Windows and the audio or video program alters the appearance of a GUI window element associated with the application program (paragraph 62above), a POSA would have understood Mast to disclose a system in which the audio or video program [i.e., Attacher.EXE] modifies contents of a window of the application program [i.e., the subclassed appliction] created through the operating system [i.e., a version of Microsoft Windows]."

3. Claim 3: "The system of claim 1, wherein the audio or video program runs in a window created through the operating system and related to a window of the application program created through the operating system."

206. Mast explains that Attacher.EXE runs in at least one window—namely the image attachment window. As discussed in Section VI.C.1.f above, once Attacher.DLL subclasses a window, Attacher.EXE creates an image

attachment window. (7:15-36 ("Once the application has been subclassed, … a message is posted to Attacher.EXE to select and open an appropriate image file. In block 406, the image attachment window is created for the starting application.")). Attacher. EXE, also known as the image attachment application, "provides the user interface to Attacher.DLL and performs all image file processing on behalf of the Windows based application with which an image attachment window is associated." (6:21-26).

207. Attacher.EXE's image attachment window is related to window of the subclassed application. The image attachment window modifies the window of the subclassed application by adding the button the caption bar and inserting video, as described in Section VI.C.1.f above. The image attachment window is further related because it can be configured as a child window to subclassed application's window (12:6-23), meaning that the image attachment window can be overlaid on and attached to the subclassed application's window (IV.B above). In view of this disclosure, a POSA would have understood Attacher.EXE to run in a window that is related to a window of the subclassed program within the meaning discussed in paragraph 64 given that tasks performed in the AV program's window (i.e., the image attachment window of Attacher.EXE) relate to the application program's window in some way

208. A POSA would have understood from Mast's identification of Windows OS (*see* Section VI.C.1.b above) that the OS creates the windows of the Attacher.EXE and the subclassed application. (Paragraph 25 above (noting that Windows OS creates windows when instructed by a program)). Furthermore, since the '591 patent's only example of integration by subclassing is specific to Windows OS (Paragraph 12.b above), Windows OS creates the windows in Mast to same extent it creates those in the '591 patent example.

4. Claim 4: "The system of claim 1, wherein the application program comprises one of: an email program, a presentation program, a publishing program, a word processing program, a spreadsheet program, an instant messaging program, a telephony program, and a gaming program."

209. Mast discloses that the "subclassed application" may be a **word processing program** or a **spreadsheet program**. (1:20-22 ("FIG. 14A shows a typical prior art display screen 1400 containing two windows 1410 and 1420. Window 1410 is associated with a word-processing program."); 5:4-10 ("FIG. 14B illustrates a sample windowed display of one embodiment of the present invention wherein each application window has an image attachment window attached to its caption bar. All elements of FIG. 14B are identical to those of FIG. 14A except for the addition of attachment window images 1470 and 1475. Image 1470 is attached to the caption bar of application window 1410."); 12:51-52 (disclosing that "the application" may be a "spreadsheet[] or text editor[]").

5. Claim 5: "The system of claim 1, wherein the audio or video program comprises one of: an audio or video player program, an audio or video recorder program, an audio or video player-and-recorder program."

210. As discussed in Section VI.C.1.d above, Attacher.EXE plays video. (281:38-50; 2:54-56; 3:22-24). Thus, a POSA would have considered Attahcer.EXE to constitute an audio or video player program.

6. Claim 6: "The system of claim 1, wherein the audio or video program comprises one of: an audio-only program, a video-only program, and an audio-and-video program."

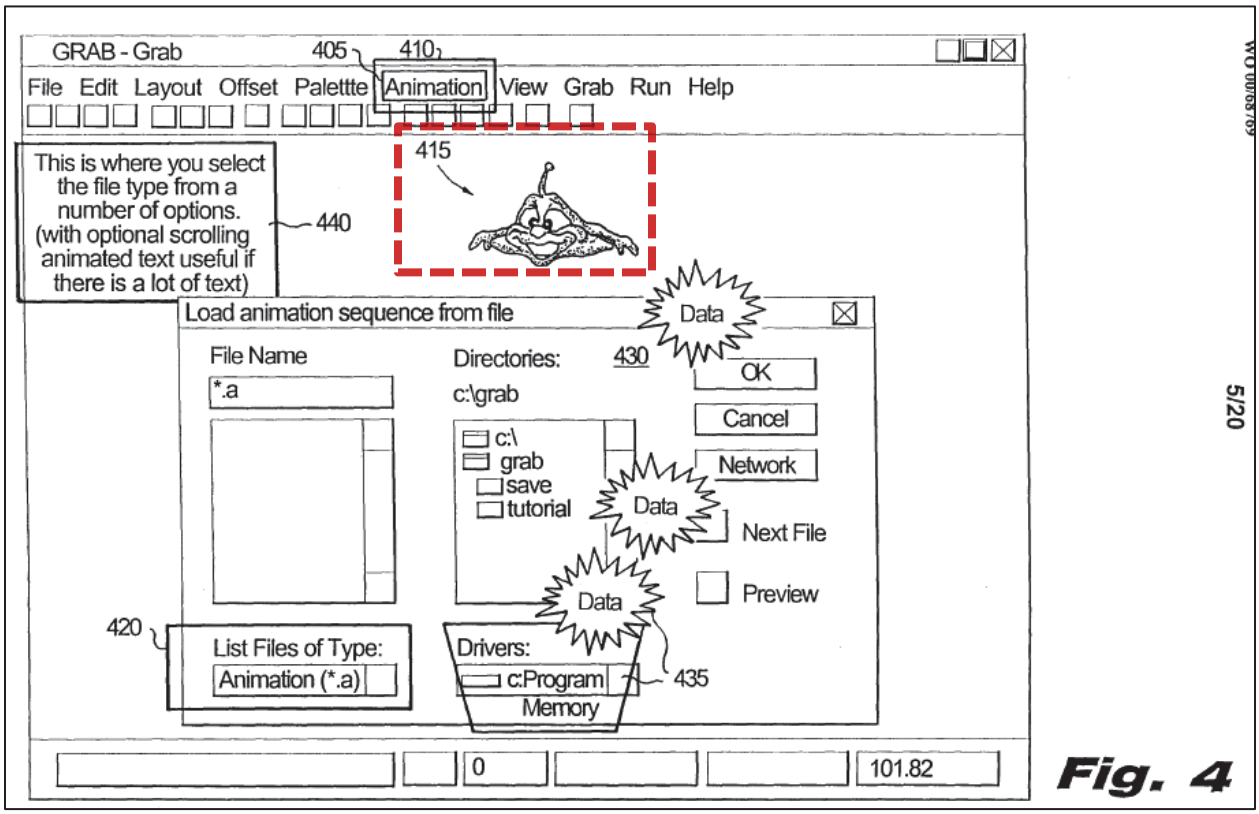
211. As discussed in Section VI.C.1.d above, Attacher.EXE plays video. (281:38-50; 2:54-56; 3:22-24). Mast does not specify whether the video files include audio. A POSA would have considered Attacher.EXE to constitute a video-only program or alternatively (in the event that a particular video included audio), an audio-and-video program.

D. Ground 5: Lui Discloses Each Limitation of Claims 1-6 of the '591 Patent

212. According to the face of the document, Lui (Ex. 1006)[14] is a PCT patent application that was published on November 16, 2000—more than one year before the '591 patent's February 5, 2002 filing date. Lui describes a "Cooperative Help Assistance" (CHA) system that uses subclassing to monitor user interactions with a "host application" and in turn provide "intelligent assistance" based on

[14] Unless otherwise indicated, all citations in Section VI.D are to Ex. 1006.

those interactions. (Abstract; 56:20-21). Lui's CHA system may provide such assistance via an animated "Guide Character," which is outlined below and can provide oral feedback (e.g., exclaiming "Good!") when users take particular actions in the host application. (11:26-12:6, 16:23-24, 43:27-31).



(Fig. 4 (Guide Character outlined)).

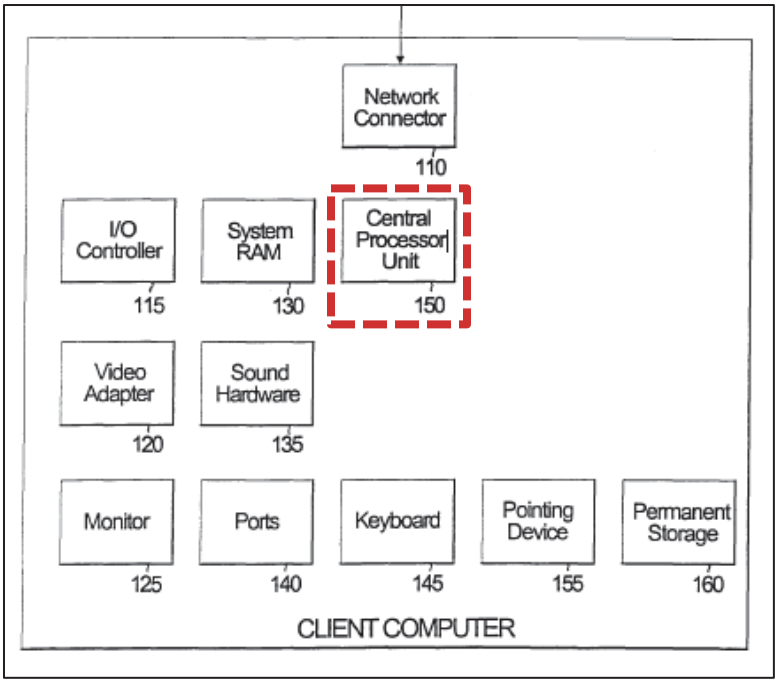
213. After reviewing Lui and claims 1-6 of the '591 patent, it is my opinion that a POSA would have understood Lui to disclose every limitation of the claims. The basis for my opinion and the details of my analysis follow below.

1. Claim 1: "A system for integrating an audio or video program with an application program comprising:"

214. Lui discloses a system that integrates an audio or video program (in the form of the CHA system) with an application program, which Lui describes as a "host application." (*See generally* 3:4-18). The elements of the preamble are discussed thoroughly in Sections VI.D.1.c through VI.D.1.j below.

a. "[*A*] hardware including a processor;"

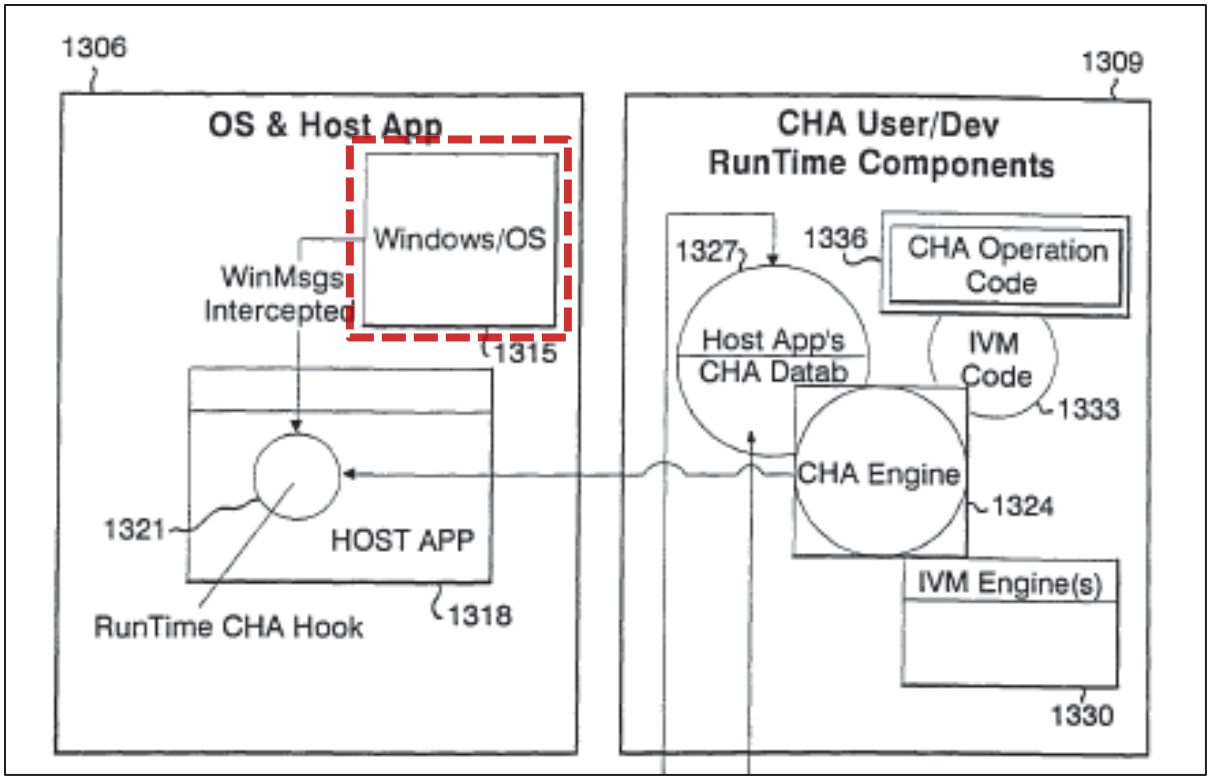
215. Lui discloses implementing the system "on an 'INTEL PENTIUM"-based system" with "known input/output devices, including a display, a keyboard, a mouse, etc." (6:11-14). As a POSA would have understood, the Intel Pentium was a well-known computer hardware processor. Further, Figure 1 in Lui illustrates an "example of a computer system including hardware and software" for implementing the invention (4:4-5); it depicts a "client computer" with a "Central Processor Unit."



(Fig. 1 (processor outlined)). In light of these disclosures, a POSA would have understood Lui to disclose hardware including a processor.

b. "[*B*] an operating system;"

216. Lui discloses implementing the system on the above-noted "'INTEL PENTIUM"-based system" running "'Microsoft Windows NT' to provide the GUI." (6:12-14). Lui also illustrates "Windows/OS" in Figure 1A, which provides "greater detail" concerning Figure 1. (4:6-8).



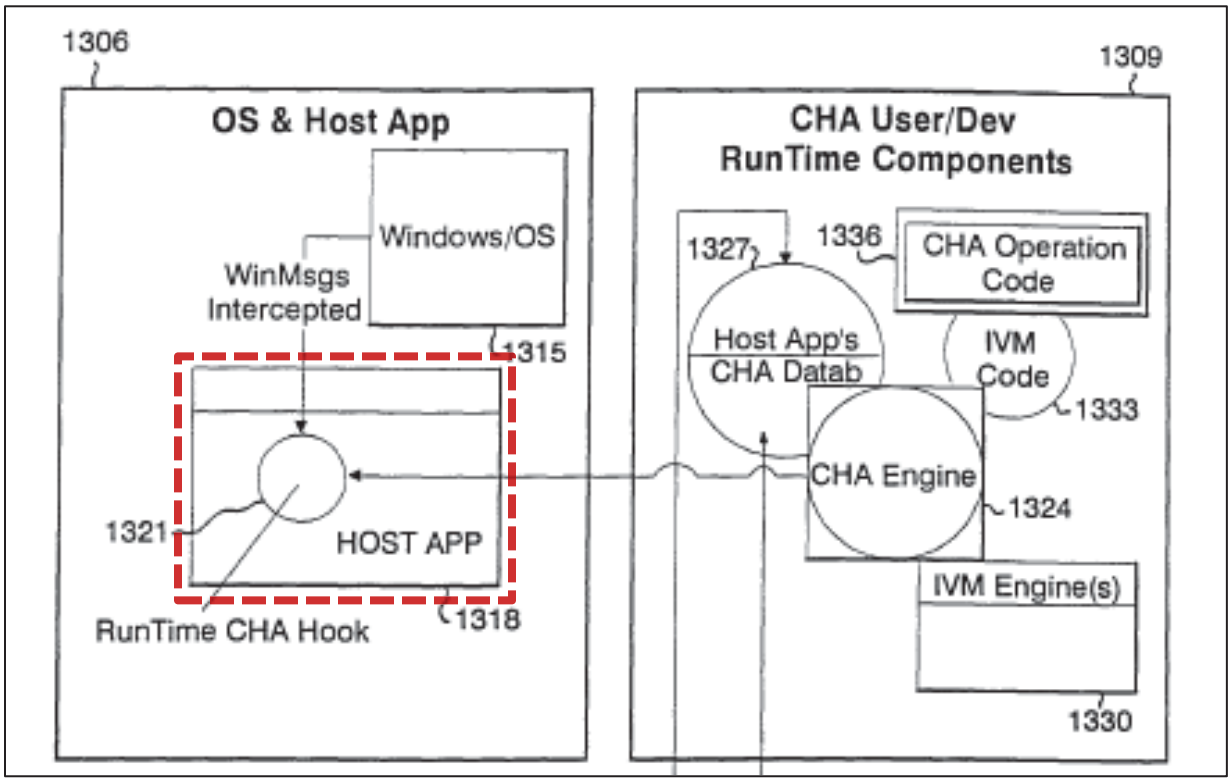
(Fig. 1A (excepted) (OS outlined)).

217. As discussed in paragraph 16 above, a POSA would have understood Microsoft Windows NT and other versions of Microsoft Windows to constitute an operating system. The '591 patent identifies "Microsoft Windows OS" as an example of an operating system that can be used in connection with the purported invention. (Ex. 1001, 3:12-14; see also Paragraph 43 above).

218. In light of these disclosures, a POSA would have understood Lui to disclose an operating system.

c. "[*C*] the application program running on the operating system; and,"

219. Lui describes the CHA system as enhancing a "host application" (6:11) and notes that such host applications can include "stand-alone application[s] such as a word processor or a spreadsheet program" (18:10-11). Figure 1A depicts the "HOST APP" running on a Microsoft Windows operating system.



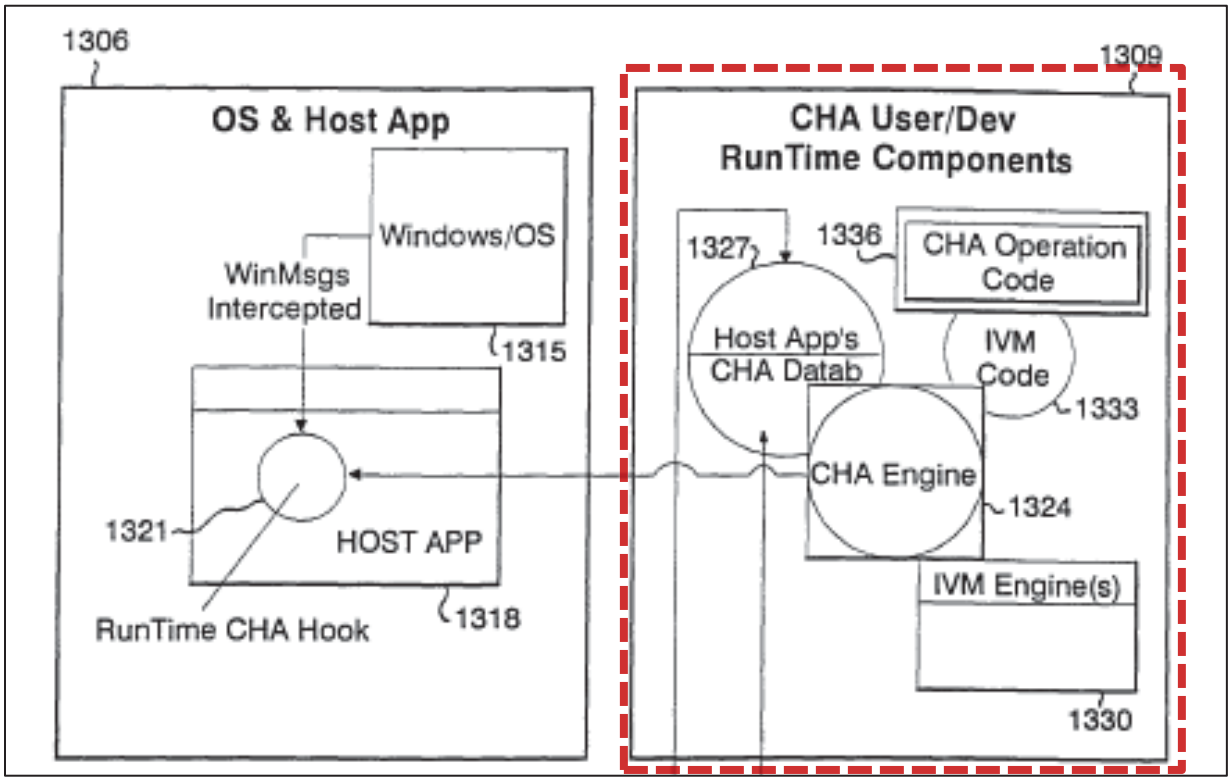
(Fig. 1A (excepted) (Host App outlined)).

220. In light of these disclosures, a POSA would have understood Lui to disclose an application program (i.e., the host application) running on the OS given that the host application is distinct from the OS, per the meaning that "application program" would have had to a POSA (paragraph 48 above). In addition, the '591

patent identifies word processing and spreadsheet programs as examples of application programs that can be used in connection with the purported invention. (Ex. 1001, 2:6-8; 6:21-33; 7:41-51).

d. "[*D*] the audio or video program running on the operating system,"

221. Lui discloses that the "CHA system" can take the form of an "application program or software" running on the Windows operating system. (5:4-5, 5:24-27, 19:29-31). For example, Figure 1A depicts the CHA system 1309 running in a "computing environment . . . in conjunction with [the] Host Application 1318" and the "Microsoft Windows" operating system. (5:24-27).



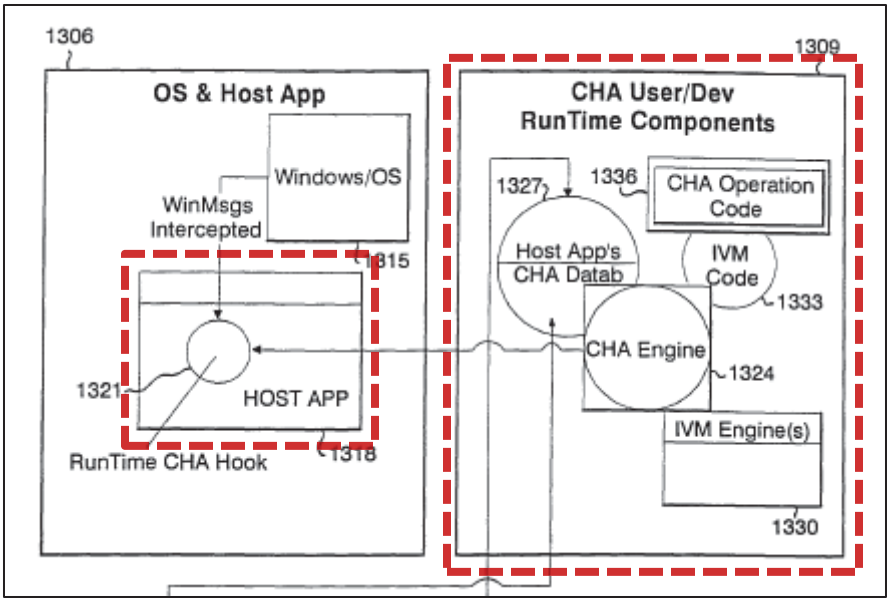
(Fig. 1A (excepted) (CHA system outlined)).

222. Further, Lui discloses that the CHA system can present feedback to users via the above-noted "Guide Character," which can provide "positive reinforcement expressed as digitized speech." (16:23-24). The Guide Character can "be displayed in conjunction with the processing of a sound or a voice file to aurally convey information using an audio-based mechanism." (12:4-7). In particular, Lui also discloses that the CHA system includes an "Expression Engine 2170" to "play sound." (51:9-11). Further, Lui describes the CHA system as being capable of various modes that provide "animation" (9:13) or other types of "video" (13:24-28).

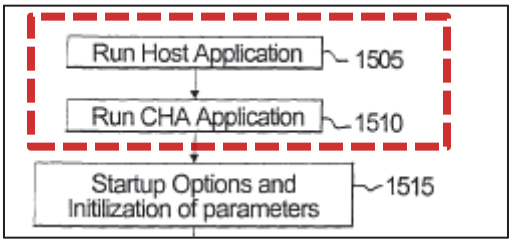
223. In light of these disclosures, a POSA would have understood Lui to disclose an audio or video program (i.e., the CHA system) running on the OS given the CHA system's capability to present audio and/or video, per the meaning of "audio or video program" to a POSA (Paragraph 47 above).

e. "[*E1*] the audio or video program separate from"

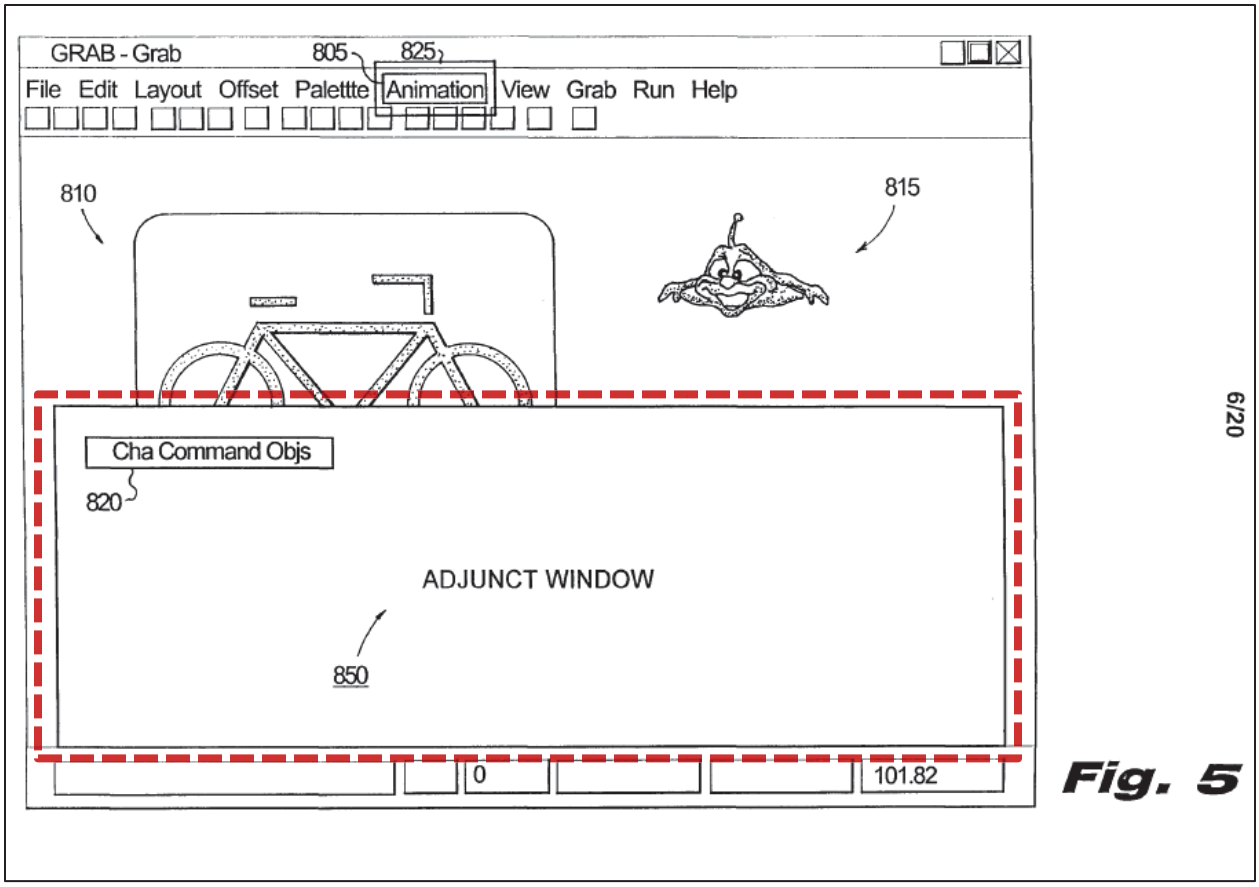
224. Figure 1A in Lui illustrates the CHA system as separate from the host application:



(Fig. 1A (Host App and CHA system outlined)). Likewise, Lui describes the host application and CHA system launching separately, in steps 1505 and 1510:



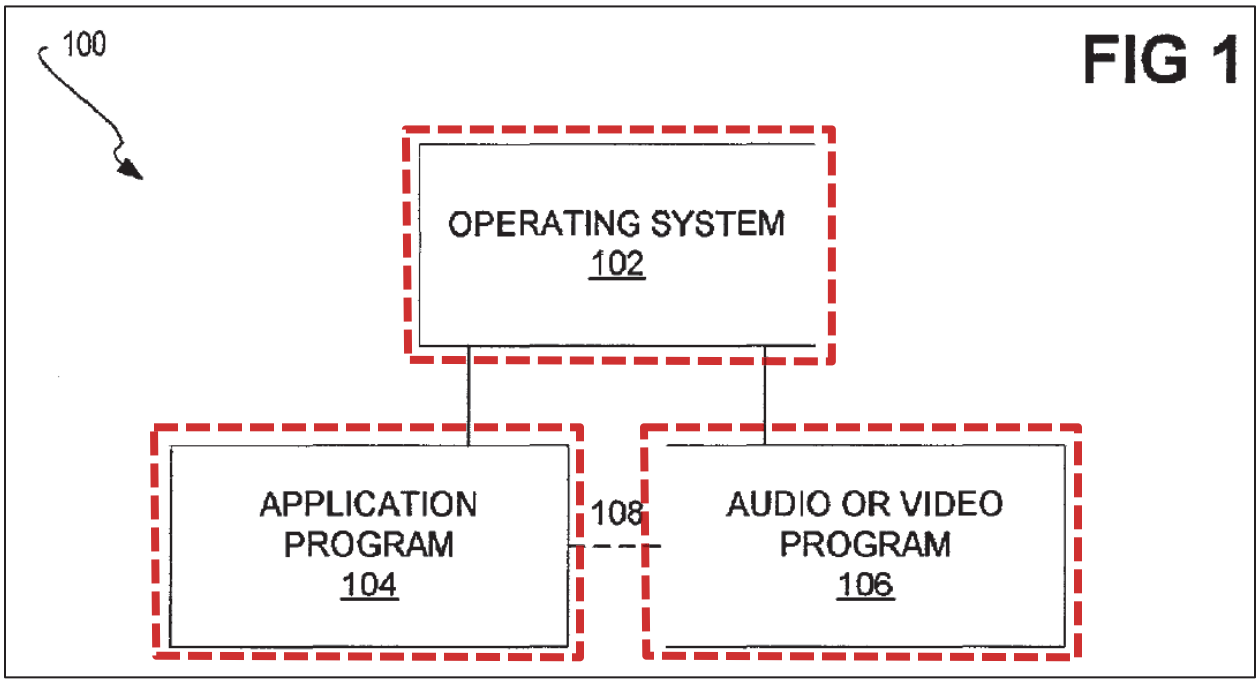
Further, Figure 5 in Lui illustrates the "CHA Interface Overlay with an Adjunct Window," in which "animated interactive activities run independently and communicate bi-directionally with [*the*] Host Application." (4:15-17)



(Fig. 5 (Adjunct Window outlined)). The Adjunct Window 850 "run[s] on its own process [and] provides an independent method for handling any type of data display in any format such as graphics, video, and text." (13:27-28). Further, Lui describes how the CHA system can be developed "without access to the internal code of the application program, except through public OS interfaces." (17:9-11).

225. In light of these disclosures, a POSA would have understood Lui to disclose an audio or video program (i.e., the CHA system) separate from the application program (i.e., the host application) in that the CHA system could be executed separately from any given host application (of which Lui discloses

various examples), per the meaning of "separate from" to a POSA (paragraph 49 above). Indeed, Figure 1 in Lui mirrors Figure 1 in the '591 patent, which the specification cites when describing the audio or video program as "separate from" the application program (Ex. 1001, 3:25-26):



(Ex. 1001, Fig. 1 (outlining the OS, the application program, and the audio or video program).

f. "[*E2*] but integrated with the application program" & "[*H*] wherein the audio or video program is further integrated with the application program by subclassing into a window of the application program such that the audio or video program detects when an event related to the application program occurs."

226. Lui discloses using subclassing and hooking techniques to integrate the CHA system (i.e., what a POSA would have understood to be an "audio or

video program") with a host application (i.e., the "application program). For example, "Customized application hook-interfaces and subclassing components are used in the operating system with a live Host Application program without access to the internal code of the application program, except through public OS interfaces." (17:9-11).

227. In particular, the CHA system includes a "user interception function," which may "involve 'MICROSOFT WINDOWS'-based subclassing and uses Hooking Functions such as SetWindowsHook, UnhookWindowsHook, and CallNextHookEx, and is responsible for intercepting all operating system (OS) messages before and after such messages are processed by the Host Application." (18:5-9). Lui also describes how this "integration function" may work differently depending on the nature of the host application. (18:9-10). For example, if the host application "is a stand-alone application such as a word processor or a spreadsheet program, then hooking may be performed on the 'WIN32' API level where message interception is performed on a relatively low level." (18:9-12). Paragraphs 17 and 40 above discusses the Win32 API and its role in implementing the hooking and subclassing techniques.

228. Figure 12 in Lui also illustrates a series of "Guide Sequence Processing Modules" within the CHA system (4:25), including a "Host App Subclassing Object" and related "CHA Message Mapping" module, which "handle

the interception of Host Application operating system messages and the filtering and mapping to CHA Events which are then passed to the Main Sequence Processor 2050 to coordinate and synchronize user input actions on the Host Application with internal processes and sub-processes being managed for the presentation of InfoObjs" (35:5-9). In other words, by using subclassing to intercept these messages, the CHA system monitors how users are interacting with the host application and in turn determines what content to present in response. In particular, Lui discloses that the CHA system includes a "Monitor Mode," which uses hooking and subclassing to "intercept[] and monitor[] all user actions with the Host App." (56:20-28).

229. Further, Lui describes subclassing as a mechanism within Microsoft Windows that "allow[s] for the establishment of an intercepting routine or function to monitor and process messages before or after such messages are passed on to the Host Application," with such "interception functions permit[ting] control of the behavior of the Host Application and the adding of additional functionality which may not have originally existed in the Host App." (19:19-24).

230. In light of these disclosures, a POSA would have understood Lui to disclose the use of subclassing to integrate an audio or video program (i.e., the CHA system) with an application program (i.e., the host application). In particular, a POSA would have understood that the CHA system was "integrated with" the

host application given the relationship between the UI of the CHA system and the UI of the host application. (Paragraph 60 above). Further, given Lui's disclosure of subclassing as discussed above, a POSA would have understood that the CHA system was "further integrated with the application program [i.e., the host application] by subclassing into a window of the application program such that the audio or video program detects when an event related to the application program occurs" per the applicable understandings of "subclassing" (Paragraph 55 above), "window" (Paragraph 27 above), and the "further integrated . . ." phrase as a whole (Paragraph 57 above).

g. "[*E3*] such that the application program is unaware that the audio or video program has been integrated therewith,"

231. Lui emphasizes that the CHA system works "independently from the original source code or development environment of the target Host Application" (3:7-9) and, in particular, does not require "access to the internal code of the application program of interest." (17:9-11). Further, Lui notes that the host application can be a "stand-alone application such as a word processor or a spreadsheet program." (18:10-12).

232. In light of these disclosures and the nature of the hooking and subclassing techniques that Lui uses to achieve the integration, a POSA would have understood that Lui's integration is accomplished such that the application

program (i.e., the host application) is unaware that the audio or video program has been integrated with it. As discussed in Paragraph 43 above, nothing about integration with subclassing requires that the host application ever know of the integration. As described in Part VI.D.1.f above, the hooking and subclassing merely monitor and intercept messages. Nothing in Lui suggests that the host application would receive any new message that could inform it of the integration—let alone one that it would recognize as such.

233. Thus, given that a POSA would have understood that the host application was not "developed a priori" to integrate with the CHA system "and also does not operate with knowledge that the integration has occurred" per the interpretation discussed in paragraph 51 above, a POSA would have understood Lui as disclosing a system in which the application program is "unaware that the audio or video program has been integrated therewith.

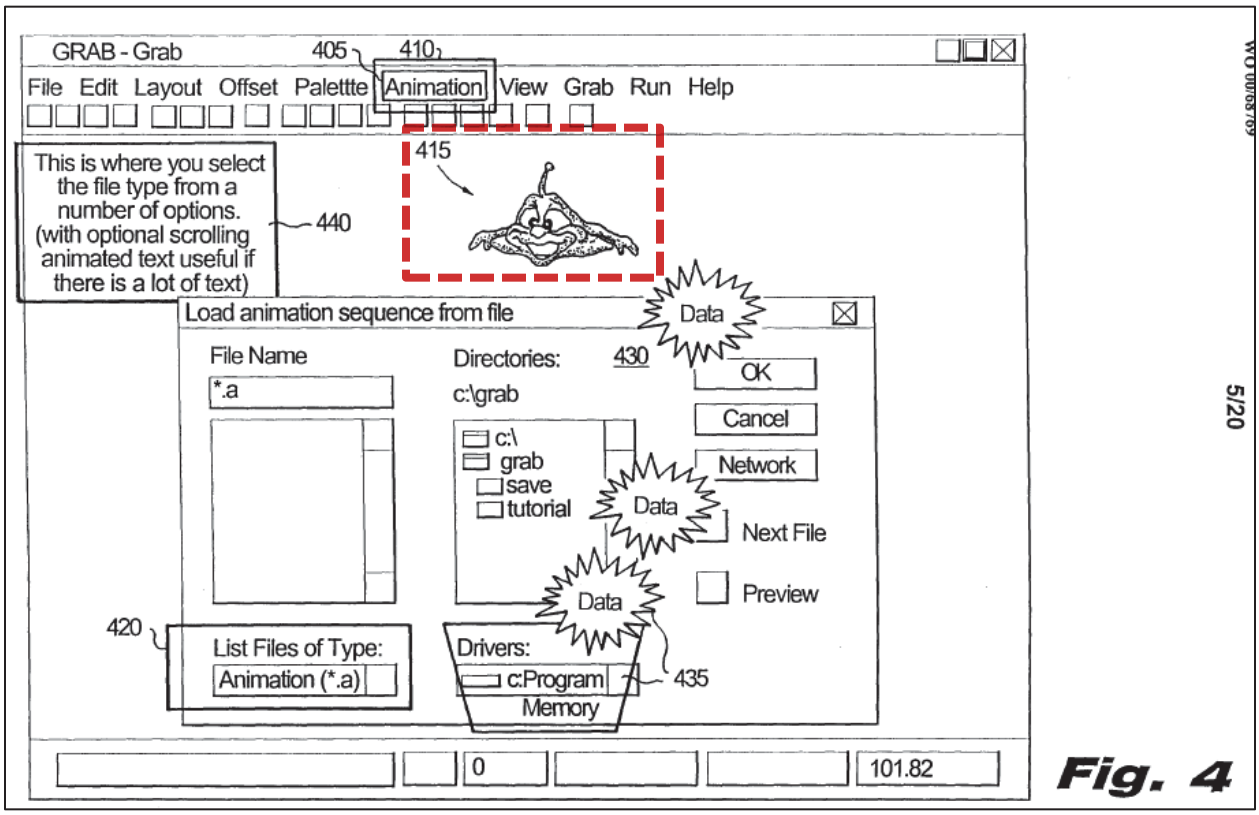
h. "[*F1*] wherein a user of the application program directly interacts with the application program, and"

234. Lui expressly notes that users may "interact" with the host application "directly." (32:15-16). Consistent with this disclosure, Lui describes host applications in which a user inputs commands using a GUI interface. For example, Figure 2 "shows an example of a typical application and interface for opening a file via a dialog box." (4:9-10). As discussed in Paragraph 50 above, anytime a user enters information into an application program running on the user's computer

via a GUI window element of that application, a POSA would have understood the interaction to be direct.

i. "[*F2*] the user interacts with the audio or video program as though the audio or video program were part of the application program ,"

235. Lui describes the GUI of the CHA system (i.e., the audio or video program) overlaying the GUI of the host application (i.e., the application program). For example, Figure 4 depicts the "CHA Interface Overlay," including the "Guide Character" 415, which "provide[s] a friendly personification or image to describe the Host application's logic or operations" and thus "eases the user into the complexities of the Host Application program." (4:13-14, 11:26-30, 12:23-24).



(Fig. 4 (Guide Character outlined)). Lui emphasizes that the Guide Character "overlap[s] or cover[s] portions of the live application." (12:19-22).

236. Further, Lui describes how the user interacts with the Guide Character (or alternatively multiple guide characters) that provide "encouragement or discouragement, in a manner similar to a live teacher, with the particular responses dependent upon the "user's actions" in working through "any procedure of a Host Application." (8:29-9:9). These actions are monitored by the CHA engine. (9:4-9). In this manner, a user dictates the behavior of the Guide Character—and thus interacts with the CHA system—by virtue of the direct interactions the user has with the host application, and in particular, the host application's user interface. Performing the desired procedure in the application program properly could prompt "encouragement" from the Guide Character, whereas incorrect steps could prompt "discouragement." In other words, a user's interaction with the host application dictates the behavior of the Guide Character and thus the CHA System. In addition, Lui describes how "***events . . . performed in the Host Application*** can also generate a message to an object inside the Adjunct Window" of the CHA System and "activate an animation that illustrates an ***underlying process occurring in the Host Application***, thus "complement[ing] the Host Application and augment[ing] the expression output to the user for the explaining of a Host Application's organizational concepts or principles and details. (10:22-29).

237. In light of these disclosures and, in particular, the intersection of the GUI elements along with the role of the host application's user interface when a user interacts with the CHA system, a POSA would have understood Lui to disclose a system in which users interact with the audio or video program (i.e., the CHA system) as though the audio or video program were part of the application program (i.e., the host application). As discussed in paragraph 59 above, a POSA would have understood that a user "interacts with the audio or video program as though the audio or video program were part of the application program" provided that the UI are related in some way, for example when some aspect of the UI of the application program is used to control the audio or video program. Per the examples discussed above, Lui describes situations in which the content presented by the CHA System (e.g., the encouragement or discouragement from the Guide Character and/or the nature of the animation presented in the Adjunct Window) depends on the nature of the user's interactions with the UI of the application program.

j. "[*G*] wherein the processor executes the operating system, the application program, and the audio or video program,"

238. Lui describes a "computer system running the Host Application program and the CHA system." (12:7). Figure 8 depicts the host application having been launched and "running in the operating system," after which the

"CHA Application" is likewise "launch[ed]" and "establishes links with existing operating system services" (19:25-31). In light of these disclosures, a POSA would have understood Lui as describing a system in which the processor noted in Section VI.D.1.a above executes the operating system, the application program (i.e., the host application program), and the audio or video program (i.e., the CHA application). A processor must necessarily execute the operating system, the application program, and the audio or video program in order for these programs to run.

2. Claim 2: "The system of claim 1, wherein the audio or video program modifies contents of a window of the application program created through the operating system."

239. Lui describes various ways in which the CHA system (i.e., the audio or video program) modifies the contents of a window of the host application. For example, Lui discloses that the previously-discussed Guide Character 415 is "presented as overlapping or covering portions of the live application," thereby "eas[ing] the user into the complexities of the Host Application program." (12:25-31). Further, the CHA system can provide a "command indicator" that "visually indicat[es] to a user a portion of a display of the computer program" corresponding to a particular command of interest. (69:2-4). For example, Figure 4 depicts an "example application" (i.e., the host application—the "application program)) that

has been enhanced with a "Highlighting Effect 405" associated with a particular menu item. (11:26-31):

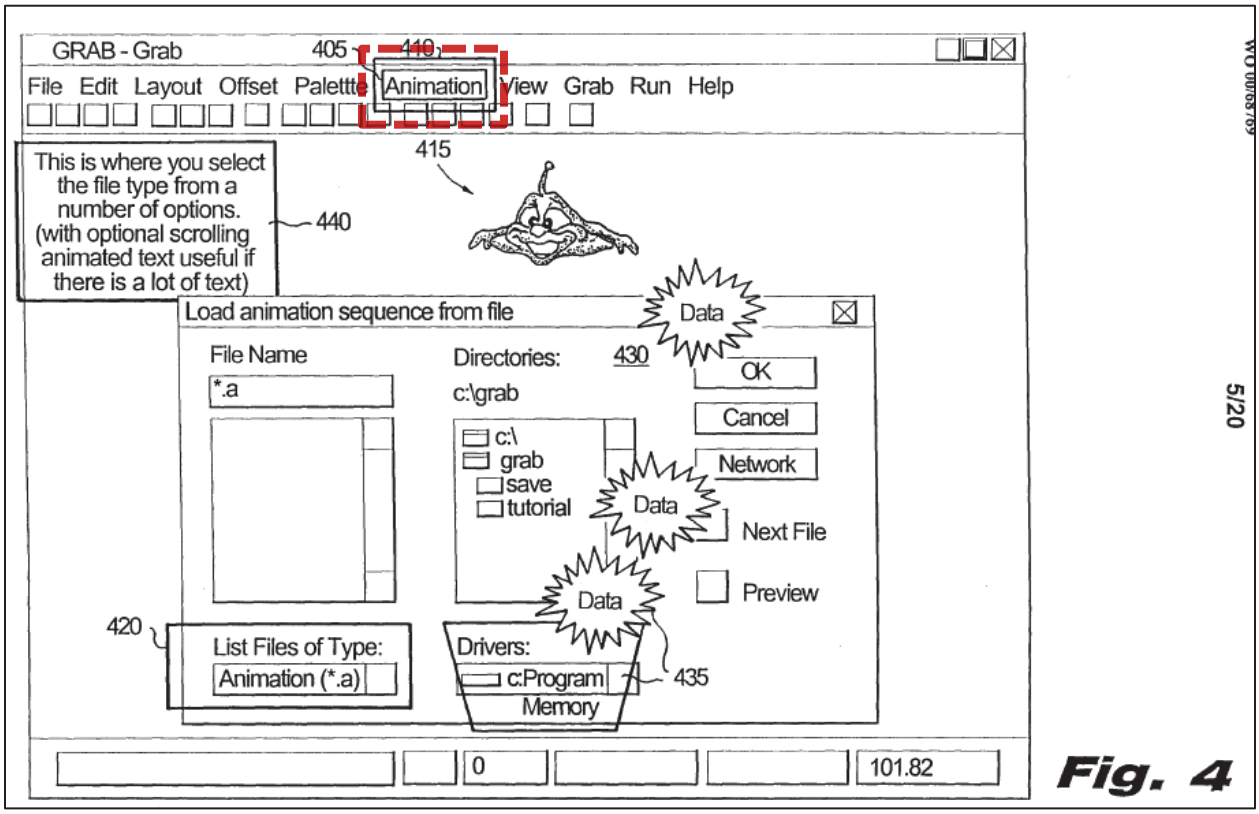


Fig. 4

(Fig. 4 (animation outlined)).

240. A POSA would have understood each of these as examples of the CHA system (i.e., the audio or video program) modifying the contents of a window of the Host Application program. As discussed in paragraph 62 above, a POSA would have understood the phrase "wherein the audio or video program modifies contents of a window of the application program created through the operating system" as being satisfied by any system in which the audio or video program

alters the appearance of a GUI window element associated with the application program, or inserts audio or video into the window.

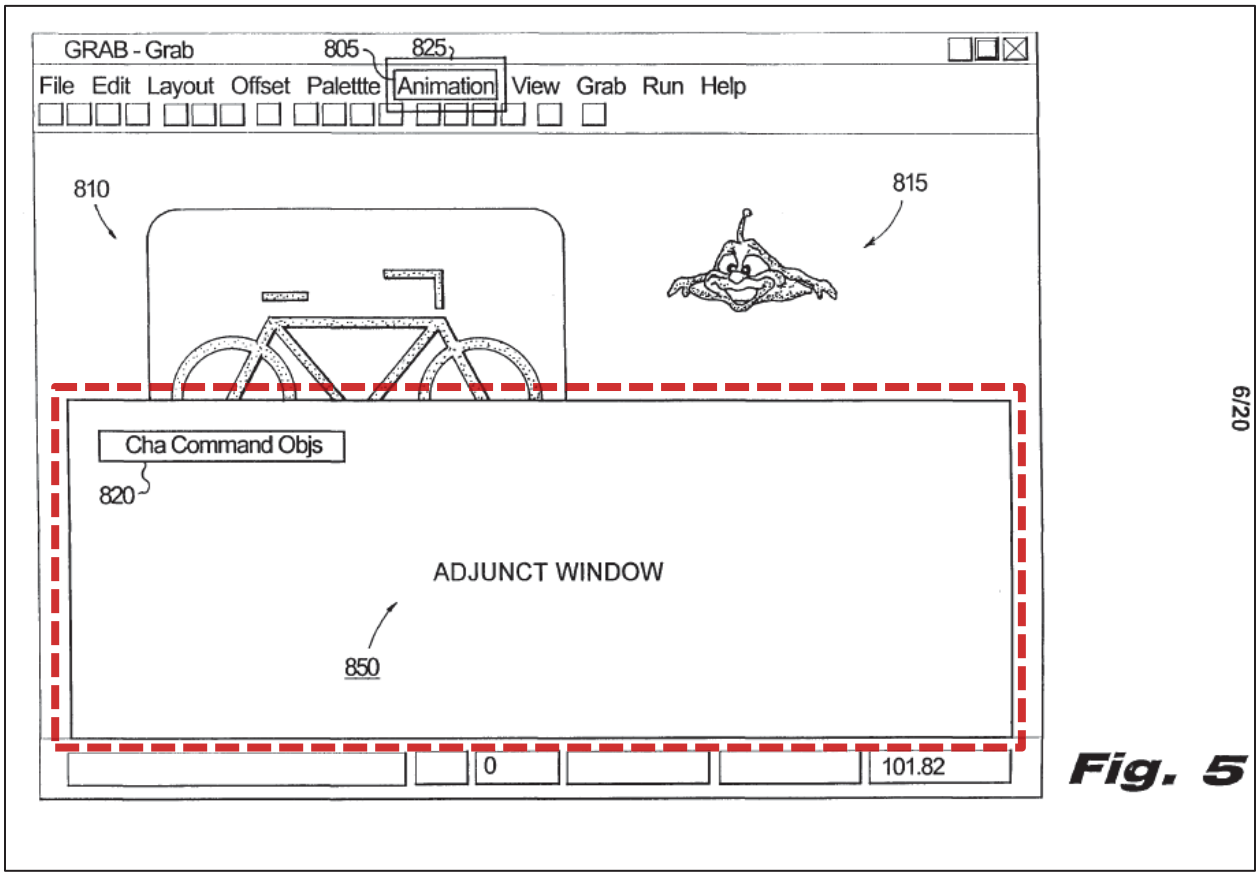
241. Further, a POSA would have understood the phrase "wherein the audio or video program modifies contents of a window of the application program created through the operating system" (i.e., in claim 2) as being satisfied by any system in which the audio or video program alters the appearance of a GUI window element associated with the application program, or inserts audio or video into the window. The CHA system alters the appearance of the application program's GUI window element by incorporating the highlighting and the Guide Character, which from the perspective of a typical user would have appeared as an integral part of the host application itself.

242. Further, a POSA would have understood from Lui's identification of Windows OS (*see* Section VI.D.1.b above) that the operating system creates the target application's window. (Paragraph 25 above (noting that Windows OS creates windows when instructed by a program)). The '591 patent's only example of integration by subclassing concerns Microsoft Outlook Express and is specific to Windows OS, as discussed in Paragraph 12.b above). A POSA reviewing this example would have understood Windows OS as creating Microsoft Outlook Express's windows.

3. Claim 3: "The system of claim 1, wherein the audio or video program runs in a window created through the operating system and related to a window of the application program created through the operating system."

243. Lui also describes various ways in which a window of the CHA program is related to a window of the Host Application per the construction discussed in paragraph 62, under which the two windows are related if one or more tasks performed in the audio or video program's window relate to the application program's window in some way. For example, Lui describes how the CHA Interface Overlay noted above can include an "Adjunct Window" in which "animated interactive activities run independently and communicate bi-directionally with a Host Application" (4:15-17). For example, in one mode of operation, the activities running in the Adjunct Window "can communicate directly with the interface of a Host Application." (8:9-11). In another mode of operation, "objects in the Adjunct Window are linked or associated with the live application." (10:6). For example, "events occurring or being performed in the Host Application can . . . generate a message to an object inside the Adjunct Window application to activate an animation that illustrates an underlying process occurring in the Host Application." (10:22-24). In this manner, the Adjunct Window "complements the Host Application and augments the expression output to the user." (10:24-26). Similarly, Figure 5 illustrates a mode in which the Adjunct Window 850 "may be used to present an overview map of the entire Host App's interface to display in

context where the user is currently interacting." (13:29-30). The Host Application and the Adjunct Window 850 can be closely synchronized through the CHA's System's internal messaging." (14:4-6). In other words, tasks performed in the CHA System's window (e.g., the Adjunct Window), such as presenting the "overview map" relate to the host application's window per the understanding discussed in paragraph 62 above given that the nature of the content displayed in the Adjunct Window depends on what is currently transpiring in the host application's window.



(Fig. 5 (Adjunct Window outlined)).

244. Separately, Lui describes how a user can "actuate[] a button or icon," thus prompting the CHA system to provide an "additional window" that "displays animations" concerning relevant aspects of the Host Application "that may not be obvious to the user." (14:24-30). Again, the nature of the task in the "additional window" (i.e., the audio or video program's window) depends on the nature of the button or icon that the user has selected in the application program.

245. Given these disclosures, a POSA would have understood Lui to disclose a system in which a window of the audio or video program (i.e., the CHA system) is related to a window of the application program (i.e., the host application) per the construction discussed in paragraph 62, under which the two windows are related if one or more tasks performed in the audio or video program's window relate to the application program's window in some way.

246. Further, given Lui's disclosure that the host application and the CHA system run in Windows OS (*see* Section VI.A.1.b above), a POSA would have understood that Windows OS creates the CHA system's windows and the host application's windows. See Paragraph 25 above (noting that Windows OS creates windows when instructed by a program)). Furthermore, Windows OS creates the windows in Lui to same extent it creates those in the '591 patent. (Paragraph 12.b above).

4. Claim 4: "The system of claim 1, wherein the application program comprises one of: an email program, a presentation program, a publishing program, a word processing program, a spreadsheet program, an instant messaging program, a telephony program, and a gaming program."

247. As previously noted in Section VI.D.1.c above, Lui discloses that the host application may be a "stand-alone application such as a word processor or a spreadsheet program." (18:10-11). Thus, a POSA would have understood Lui as disclosing a system with an application program (i.e., the host application) in which the application program "comprises one of: an email program, a presentation program, a publishing program, ***a word processing program, a spreadsheet program***, an instant messaging program, a telephony program, and a gaming program" (emphasis added).

5. Claim 5: "The system of claim 1, wherein the audio or video program comprises one of: an audio or video player program, an audio or video recorder program, an audio or video player-and-recorder program."

248. As discussed in Section VI.D.1.d above, Lui discloses that the CHA system can play both audio and video. In particular, it can "play sound" (51:10) and also present "video." (12:17). Thus, a POSA would have understood Lui to disclose an "audio or video player program."

6. Claim 6: "The system of claim 1, wherein the audio or video program comprises one of: an audio-only program, a video-only program, and an audio-and-video program."

249. Given that Lui describes the CHA system as capable of presenting both audio and video (VI.D.1.d above), a POSA would have understood Lui to disclose an "audio-and-video program."

VII. SIGNATURE

250. I understand and have been warned that willful false statements and the like are punishable by fine or imprisonment, or both (18 U.S.C. § 1001). I declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and further, that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under § 1001 of title 18 of the United States Code.

251. I declare under penalty of perjury that the foregoing is true and correct.

Executed on 2/11/15 in Boston, Massachusetts.

Mark E. Crovella, Ph.D.