TABLE OF CONTENTS

TABLE OF AUTHORITIES

TABLE OF AUTHORITIES (cont'd)

INTRODUCTION AND INTEREST OF AMICUS CURIAE

In this important computer software copyright case, the District Court held that program elements dictated by compatibility requirements receive copyright protection. By so holding, the District Court's decision is totally at odds with the software copyright law evolving in the other circuits. It appears that the District Court reached its incorrect result by misconstruing the relevant case authority. This amicus brief is submitted to attempt to prevent the District Court's faulty reasoning from becoming precedent in this Circuit and elsewhere.

The American Committee for Interoperable Systems ("ACIS") is an informal organization of companies that develop innovative software and hardware products that interoperate with computer systems developed by other companies.[1] Every member of ACIS believes that computer programs deserve effective intellectual

---

[1] ACIS members include Accolade, Inc., Advanced Micro Devices, Inc., Amdahl Corporation, Broderbund Software, Inc., Bull HN Information Systems, Inc., Chips and Technologies, Inc., Clearpoint Research Corporation, Color Dreams, Inc., Comdisco, Inc., Emulex Corporation, Forecross Corporation, The Fortel Group, Fujitsu Systems Business of America, Inc., Informix Corporation, Integrated Documents Applications Corp., ICTV, Johnson-Laird, Inc., Kapor Enterprises, Inc., Landmark Systems Corporation, LCS/Telegraphics, NCR, Octel Communications Corporation, Phoenix Technologies, Ltd., Plimoth Research Inc., Seagate Technology, Inc., Software Association of Oregon, Software Entrepreneurs Forum, Storage Technology Corporation, Sun Microsystems, Inc., Systems Center, Inc., Tandem Computers, Inc., Unisys Corporation, Western Digital Corporation, and Zenith Data Systems Corporation. (The Software Association of Oregon consists of 430 software development firms, firms in associated industries, and individuals professionally involved in software development. The Software Entrepreneurs Forum consists of over 1,000 software entrepreneurs and developers.)

property protection to give developers sufficient incentive to create new computer programs. At the same time, the members of ACIS are concerned that the improper extension of copyright law will impede innovation and inhibit fair competition in the computer industry. ACIS seeks the application of legal standards that will effectuate copyright law's fundamental aims by ensuring authors "the right to their original expression," but also encouraging competitors "to build freely upon the ideas and information conveyed by a [copyrighted] work." Feist Publications, Inc. v. Rural Tel. Serv. Co., 111 S. Ct. 1282, 1290 (1991).

Neither ACIS nor its members have a direct financial interest in the outcome of this litigation. However, the District Court's decision will have serious anti-competitive consequences to ACIS members and to the computer industry. ACIS is filing this brief in order to address the important intellectual property policy issues raised in this case.

SUMMARY OF ARGUMENT

In this case, the District Court upheld the copyrightability of the utilitarian selection and arrangement of the executable computer operations that comprise the Lotus 1-2-3 command structure. The Lotus command structure functions both as a user interface between spreadsheet users and Lotus 1-2-3, and as a program interface between user-written programs known as "macros" and Lotus 1-2-3. The District Court held that both

of these command structure functions are copyrightable despite the explicit language of 17 U.S.C. §102(b), which expressly prohibits copyright protection for such processes or methods of operation. In particular, the District Court's ruling concerning the command structure in its programming interface function runs directly contrary to recent precedent from other circuits (e.g., Computer Associates Int'l., Inc. v. Altai, Inc., 982 F.2d 693 (2d Cir. 1992) ("Computer Associates")).

ARGUMENT

I. THE LOTUS COMMAND STRUCTURE IN ITS PROGRAM INTERFACE ROLE IS UNCOPYRIGHTABLE UNDER § 102(b).

A. The District Court Protected a Sequence of Utilitarian Operations and Processes -- the Only Elements the Parties' Two Programs Have in Common.

To understand the serious ramifications of the District Court's decisions if they are affirmed, it is instructive to review precisely what the District Court protected in this case. While familiarity with the technical facts is important in every software copyright case, it is particularly important here because of the dual function of the Lotus 1-2-3 command structure.

The District Court's injunction was based upon its four opinions in this case and its opinion in a prior case which was not appealed, Lotus Development Corp. v. Paperback Software

International, 740 F. Supp. 37 (D. Mass. 1990) ("Paperback").[2] Unlike Paperback, where the defendant apparently copied the 1-2-3 visual displays, in this case Borland's product was not visually similar to that of Lotus, and there was no allegation or finding of copying of screen displays. In this case (as in Paperback), it is also undisputed that Borland did not copy any of the source or object code of Lotus 1-2-3. Rather, the District Court found that Borland used a functional element of Lotus 1-2-3: its command structure.

The Lotus 1-2-3 command structure has two distinct functions. First, the commands appear on the screen in a logical order to inform the user about the available options at that stage of operation, and the user invokes a sequence of the commands to instruct the program directly to perform certain spreadsheet functions. In this role, the commands act as an interface between the user and the program.

Second, the spreadsheet user can employ the Lotus commands to write his own program known as a "macro". A "macro" is a short keystroke sequence that substitutes for a longer keystroke sequence of menu commands. Borland II, 799 F. Supp. at 206. In other words, a macro is a user-written program (or instruction)

---

[2] See Lotus Development Corp. v. Borland International, Inc., 778 F. Supp. 78 (D. Mass. 1992) ("Borland I"); Lotus Development Corp. v. Borland International, Inc., 799 F. Supp. 203 (D. Mass. 1992) ("Borland II"); Lotus Development Corp. v. Borland International, Inc., 831 F. Supp. 202 (D. Mass. 1993) ("Borland III"); Lotus Development Corp. v. Borland International, Inc., 831 F. Supp. 223 (D. Mass. 1993) ("Borland IV").

that performs a sequence of the spreadsheet operations in a particular order, i.e., using particular process steps. See Paperback, 740 F. Supp. at 64. In this role, the command structure acts as an interface between the spreadsheet program and the user-written programs. It is this program-to-program interface ("program interface") function, and the District Court's rulings concerning this command structure function, that are of primary concern to ACIS.

A consideration of the various kinds of programs that run on modern computers will clarify the functions of the command structure's program interface. All computers must run an "operating system" program, which is the basic program that tells the computer how to operate, how to manage its resources, and how to run other programs. Computer Associates, 982 F.2d at 698; P. Goldstein, Infringement of Copyright in Computer Programs, 47 U. Pitt. L.Rev. 1119, 1126 (1986). Common operating systems include MS-DOS for IBM-compatible personal computers, and the operating system for Apple Computer's Macintosh system. The operating system program serves as a platform for "application programs" such as word processing programs, database programs, video games, or, as in this case, spreadsheet programs. Application programs must be written in a way to conform to the functional requirements of the operating system, or else the applications will not work properly.

Although Lotus 1-2-3, taken as a whole, is an application program, it functions as an operating system with respect to the

user written application programs -- the macros -- that attach to it. In other words, Lotus 1-2-3 serves as a platform upon which the macros run. The syntax and semantics of the communication between a macro and the Lotus platform is the Lotus command structure.

Lotus users have invested substantial time and resources developing libraries of customized macros appropriate to their business needs. Indeed, the Lotus users collectively have invested more resources in the development of their macros than Lotus invested in the development of Lotus 1-2-3. Because of this investment in the macros, the Lotus users are "locked-in" to the Lotus environment: as they expand their operations, they simply will not purchase spreadsheet programs developed by a Lotus competitor such as Borland unless the Borland spreadsheet is compatible with their macros.

The most basic form of macro compatibility requires the Borland platform to have the ability to translate the macro's electronic instructions into electronic instructions intelligible to the Borland platform, and visa versa. Because the set of instructions used by the macro is an electronic version of the Lotus command structure, the Borland platform includes a file that functionally emulates the Lotus 1-2-3 command structure and contains the first letter of the commands. Borland calls its use of this file the "Key Reader."[3]

---

[3] To achieve compatibility, this functional emulation must be performed, regardless of whether it is done "on the fly" (as (continued...)

Complete macro compatibility, however, requires more than just the ability to run existing macros. Users of the Borland platform must be able to correct errors in the macros, and modify the macros to address new problems. Because the macros are constructed out of Lotus commands, the user of the Borland platform can debug or modify the macros only if the platform can display the commands on the terminal and only if the platform can understand the Lotus commands keyed in by the user. Thus, to perform the debugging and modification functions essential for complete macro compatibility, the Borland platform must reproduce the Lotus command structure.

The District Court below ruled that the Lotus command structure was protectable in both its user interface role and its program interface role. Other amici are addressing the District Court's error in protecting the command structure in its user interface role. The balance of this brief explains the District Court's error in protecting the command structure in its program interface role.

B. Under the Copyright Statute and the Appropriate Case Law, the Lotus Command Structure Is Not Protectable.

Because neither the Lotus computer code itself nor its visual screen displays are used by Borland's programs, the sole feature in common to both Lotus 1-2-3 and Borland's products is

---

[3](...continued)
in Borland's Quattro Pro) or "one time" (as in Microsoft's Excel).

the Lotus command structure, or, as the District Court put it, the "selection and arrangement of the executable operations" in Lotus 1-2-3. See Borland IV, 831 F. Supp. at 231; see also Borland II, 799 F. Supp. at 217-18; Borland III, 831 F. Supp. at 209, 211, 216. As explained above, macro compatibility requires the Borland platform to replicate the Lotus command structure in its program interface role. Recent case law in other circuits makes clear that copyright does not protect elements dictated by compatibility requirements. The District Court acknowledged this authority, but then misinterpreted it and thus misapplied it to this case.

1. Recent Decisions in Other Circuits Make Clear That Elements Dictated by Compatibility Requirements Do Not Receive Copyright Protection.

Ten years ago, the Court of Appeals for the Third Circuit suggested that compatibility was a "commercial and competitive objective which does not enter into the somewhat metaphysical issue of whether particular ideas and expression have merged." Apple v. Franklin, 714 F.2d 1240, 1253 (3d Cir. 1983), cert. denied, 104 S. Ct. 690 (1984). Last year, however, three different circuits rejected this view. In June 1992, the Court of Appeals for the Second Circuit in Computer Associates recognized that, as utilitarian works, computer programs are highly constrained by external factors:

> Professor Nimmer points out that 'in many instances it is virtually impossible to write a program to perform particular functions in a specific computing

> environment without employing standard techniques.' This is a result of the fact that a programmer's freedom of design choice is often circumscribed by extrinsic considerations such as (1) the mechanical specifications of the computer on which a particular program is intended to run; (2) compatibility requirements of other programs with which a program is designed to operate in conjunction; (3) computer manufacturers' design standards; (4) demands of the industry being serviced; and (5) widely accepted programming practices within the computer industry.

982 F.2d at 709-10 (citations omitted). The Second Circuit ruled that copyright does not protect program elements dictated by such external factors. Id.

In September 1992, the Court of Appeals for the Federal Circuit, citing Computer Associates, stated that "the court must filter out as unprotectable . . . expression dictated by external factors (like the computer's mechanical specifications, compatibility with other programs, and demands of the industry served by the program) . . . . " Atari Games Corp. v. Nintendo of America, Inc., 975 F.2d 832, 839 (Fed. Cir. 1992) ("Atari"). In October 1992, the Court of Appeals for the Ninth Circuit expressly recognized that computer programs "contain many logical, structural, and visual display elements that are dictated by . . . external factors such as compatibility requirements and industry demands. In some circumstances, even the exact set of commands used by the programmer is deemed functional rather than creative for purposes of copyright." Sega Enterprises, Ltd. v. Accolade, Inc., 977 F.2d 1510, 1524 (9th Cir. 1993) ("Sega") (citations omitted). Lotus v. Borland presents one of those circumstances in which "the exact set of

commands used by the programmer is deemed functional" because it is indispensable for compatibility between the macro and the platform.

2. The District Court Misinterpreted the Recent Precedent.

Notwithstanding this persuasive authority from three circuits, the District Court ruled that the Lotus command structure in its program interface role constituted protected expression. The District Court reached this erroneous conclusion by misinterpreting Computer Associates' unambiguous language.

The District Court stated that the reasoning of Computer Associates was limited to situations where "what the program was designed to fit was already in existence before the program was designed to fit it." Borland II, 799 F. Supp. at 213. The following example makes the District Court's position more comprehensible. The District Court conceded that the elements of program B1 necessary to achieve compatibility with pre-existing platform A1 were unprotected, and could be replicated by program B2 which also sought compatibility with platform A1. The Court, however, believed that the elements of platform A1 necessary to achieve compatibility with programs B1 and B2 were protected, and that platform A2 could not replicate them to achieve compatibility with programs B1 and B2. The District Court viewed the order of creation as the determining factor.

Platform A1 preceded program B1, so A1 constrained elements of B1, and copyright did not protect these constrained elements; but because A1 preceded B1, B1 did not constrain any elements of A1 at the time of A1's creation, and thus A1's interfaces remained protected, even though they subsequently became necessary to maintain compatibility with B1.

Although the Computer Associates fact pattern did involve similarities between the Computer Associates product (B1) and the Altai product (B2) necessary to achieve compatibility with the pre-existing IBM operating system (A1), nothing in the language of the opinion suggests that the Second Circuit's exclusion of the protectability of elements necessary for compatibility should be construed as narrowly as by the District Court below. Moreover, two traditional copyright principles and one reality of the computer industry weigh heavily against such a narrow interpretation.

First, the set of rules or commands permitting communication between two programs -- the program interface -- is clearly a "system" or "method of operation" unprotectable under 17 U.S.C. § 102(b). The fact that one program preceded the other does not diminish the inherently functional quality of the program interface regardless of whether it is implemented in the pre-existing program (A1) or the subsequent program (B1).

Second, the District Court's reasoning is inconsistent with the well-established copyright doctrine of scenes a faire. That doctrine provides that "where 'it is virtually impossible to

write about a particular historical era or fictional theme without employing certain "stock" or standard literary devices,' such expression is not copyrightable." Computer Associates, 982 F.2d at 709 (quoting Hoehling v. Universal City Studios, Inc., 618 F.2d 972, 979 (2d Cir. 1980)). During the course of writing novel A1 which begins a genre, the author feels few constraints limiting her range of expression. Nonetheless, subsequent authors seeking to conform their novels A2 and A3 to that genre are free to borrow from novel A1 those elements that define the genre. So long as constraints on the range of expression exist at the time the subsequent authors create their work, the constrained expression is unprotectable, regardless of whether those constraints existed at the time of the creation of the first work. See P. Goldstein, Copyright Principles, Law and Practice, 1993 supplement at 17-18.

The District Court also based its reasoning on a false assumption about Lotus 1-2-3 and computer programs generally. The District Court suggests that Lotus 1-2-3 was a stand-alone program that users later employed as a platform for their macros. In fact, Lotus 1-2-3 was designed to serve as a platform for the macros; Lotus intentionally included in its product the facility for constructing compatible macros. Similarly, the developer of an operating system designs it not as a stand-alone program, but as a platform for application software. The operating system includes "hooks" -- program interfaces -- to which the applications can attach. Given the

operating system developer's intention to create an environment in which the operating system and applications can interoperate, it makes absolutely no sense to say that "eyelets" in the applications receive no copyright protection because they are constrained by the pre-existing hooks in the operating system, but the hooks in the operating system do receive copyright protection because they were created before the eyelets. Stated differently, when the operating system developer designs the operating system's hooks, it in effect also designs the application program's eyelets, even though the application may actually be written at a much later date. If the eyelets do not receive protection, the hooks should not receive protection either.

3. Computer Compatibility Is Not About Chickens or Eggs; It Is About the Need For Avoiding Monopolies on Unprotectable Aspects of Computer Programs.

The District Court reduced the compatibility-related issues in this case to a "familiar childhood riddle" -- that of "the chicken or the egg." Borland II, 799 F. Supp. at 212. Borland had argued that since the spreadsheet users had to use the Lotus command structure in order to run, modify and debug the macros on the Borland platform, then the 1-2-3 command structure should not receive copyright protection. The District Court rejected this important point as a "chicken and egg" argument, supposedly because Lotus 1-2-3 was written first, before any user could write macros designed to work with it. Id. at 213. ACIS firmly

believes that neither this case, nor computer interoperability in general, are about chickens and eggs; rather, interoperability is about the need to avoid monopolies in computer systems.

The District Court's approach would let the first developer of a computer operating system "'lock up' basic programming techniques as implemented in programs to perform particular tasks." Computer Associates, 982 F.2d at 712 (citing P. Menell, An Analysis of the Scope of Copyright Protection for Application Programs, 41 Stanford L. Rev. 1045, 1082 (1989)). Under the District Court's reasoning, it would be permissible to have two entirely different application programs designed to run with the same pre-existing operating system, but not permissible for a second programmer to write a new operating system program compatible with the two application programs.

ACIS believes that the District Court's reasoning would effectively eliminate competition in operating systems, or any software product that functions as a platform for other software products. Such a broad monopoly would have serious implications for consumer welfare.[4] In the absence of competition, the first

---

[4] See Peter B. Menell, An Analysis of the Scope of Copyright Protection for Application Programs, 41 Stan. L. Rev. 1045, 1082, 1097 n.281 (1989); Peter G. Spivak, Does Form Follow Function? The Idea/Expression Dichotomy in Copyright Protection of Computer Software, 35 U.C.L.A. L. Rev. 723, 765 (1988). See also Computer Associates, 982 F.2d at 711-12; Richard Moreno, Note, "Look and Feel" as A Copyrightable Element: The Legacy of Whelan v. Jaslow? Or, Can Equity in Computer Program Infringement Cases Be Found Instead By the Proper Allocation of Burden of Persuasion, 51 U. La. L. Rev. 177, 206 (1990).

developer would have little incentive to develop more innovative and less costly products. These negative consequences would be compounded by the fact that the personal computer revolution, and recent major improvements in communications technology, have produced an overwhelming need for interconnection between different elements of computer systems. Within a given company, literally thousands of personal computers and workstations scattered across the globe need to interact with each other and with the company's mainframes or supercomputers. Moreover, different users need to exchange vast quantities of data with one another through their computers. The District Court's decision would lock users in to a particular operating system or network software environment, and would inhibit the transfer of data between users with different operating systems.

Interoperability is increasingly important in complex business and technical environments. Interoperability is purely utilitarian. Neither this case nor ACIS' concern is directed to the protection of artistic or fanciful aspects of computer-generated screen displays. Interoperability does not implicate the fanciful or artistic aspects of works. Nor does interoperability implicate whatever creative process goes into the designing of fanciful screen displays or the creation of computer code. Interoperability deals with the functional aspects of a computer program (in this case, a command structure) that are necessary to work with another program. Interoperability is a purely utilitarian function.

4. Given the Need for Compatibility, the Scope of Software Copyright is Sufficiently Broad Without Protecting Program Interfaces.

This Court need not be concerned that copyright protection will be insufficient and inadequate if copyright does not extend to program interfaces. Copyright protects a great deal without extending to the protection of utilitarian functions. For instance, copyright protects at least the following:

(1) It protects against computer piracy, such as illegal wholesale copying of a company's products on computer disks and reselling those disks.

(2) It protects against users who make unauthorized copies of programs for free use by friends or associates.

(3) It protects against disgruntled employees or competitors stealing the source code of a program directly, as well as indirect misappropriation of source code, by paraphrasing it or translating it to another computer language.[5]

(4) It protects against the wrongful copying of fanciful screen displays to the extent they are not functionally dictated, and have sufficient pictorial content to qualify for copyright.

(5) It protects internal elements of the program, to the extent that such elements survive the filtration test of Computer Associates.

---

[5] For instance, copyright would protect a mere translation of a computer program from one computer language (such as BASIC) to another (such as C).

Thus, it makes no sense to say that copyright law will be emasculated if it does not protect program interfaces. Moreover, even if the examples listed above do not adequately protect all aspects of a computer program, that does not mean that copyright protection should extend to program interfaces, as the District Court held here. Program interfaces constitute software processes that in some circumstances may qualify for protection under the patent laws, if the statutory requirements for patentability are met. Compare 35 U.S.C. §101, which explicitly permits patents on processes, with 17 U.S.C. §102(b), which precludes copyrights on processes. See also Atari, 975 F.2d at 839; Arrhythmia Research Technology v. Corazonix Corp., 958 F.2d 1053 (Fed. Cir. 1992). It is not necessary to extend copyright protection to technological features that Congress has already protected under the patent laws.

ACIS members rely on copyright protection for the aspects listed above, among others. We consider copyright protection important, but also consider it adequate for its intended purposes. ACIS members frequently obtain and rely upon patent protection where a higher level of protection of computer processes is needed. This includes computer hardware and, where patentable, computer software processes such as the "selection and arrangement of executable operations." ACIS members are deeply concerned, however, that a software company might obtain patent-like protection for aspects of computer technology that

do not meet the rigorous standards of patentability, merely by claiming a copyright.

II. CONCLUSION

In order to achieve compatibility with macros written by the third-party users, it was necessary for Borland to use the words of the 1-2-3 command structure, and to display those words on the computer screen. The creators of computer programs such as Borland should have the right to use as much of another program's utilitarian operations as is necessary to enable such third-party programs to work. This is needed to ensure interoperability across computer networks.

The District Court erred by failing to recognize these principles. ACIS believes that this Court should follow the reasoning of courts such as Computer Associates and Sega, and hold that program interfaces such as the Lotus 1-2-3 command structure are not copyrightable.

Dated: December 10, 1993

Respectfully submitted,

By _____

Peter M.C. Choy, Esq.
AMERICAN COMMITTEE FOR
INTEROPERABLE SYSTEMS
2550 Garcia Avenue
Mail Stop PAL 1-521
Mountain View, CA 94043
(415) 336-2482

Attorney for Amicus