More Questions than Answers

The Rise of Transformative Use to Determine Software Copyright Infringement

By Joseph L. Kish, Veronica A. Bowen, and Patrick F. Sullivan

Google v. Oracle provides little in the way of guidance for lowers courts considering fair use in determining the propriety of software copyrights, instead granting the lower courts wide discretion through its broad transformative use analysis.

On April 5, 2021, the Supreme Court issued a 6–2 decision in the longstanding copyright battle between *Google LLC and Oracle America, Inc. Google LLC v. Oracle America, Inc.*, 574 U.S. 1071 (2021). The decision, which centered around Google's fair use defense that relied primarily on transformative use, ended a fourteen-year battle between the two tech giants. Unfortunately, it provided little insight for future software copyright disputes.

Copyright Basics

To begin to understand how this decision will affect practitioners and companies alike, we must start with a basic understanding of copyright. Copyright laws (*i.e.*, 17 U.S.C. §102, "the Act") protect "original works of authorship fixed in a tangible medium of expression." 17 U.S.C §102 (a). Copyright functions by granting the author the right to exclude others while remaining true to its purpose "to promote the Progress of Science and useful Arts." US Const. art. I, §8, cl. 8. Under Section 102(a), copyright law specifically provides protection for the following categories of works: 1) literary; 2) musical; 3) dramatic; 4) pantomimes and choreographic works; 5) pictorial, graphic, and sculptural works; 6) audio–visual works; 7) sound recordings; 8) derivative works; 9) compilations; and 10) architectural works.

■ Patrick F. Sullivan is an associate in Segal McCambridge's Chicago office, focusing his practice on complex litigation. He is an accomplished litigator with extensive experience in professional liability, products liability, cyber liability and commercial defense. Joseph L. Kish is a shareholder in Segal McCambridge's Chicago office and serves as co-chair of the firm's Technology and Cyber Risk Practice Group and chair of the firm's Pro Bono & Public Service Committee. He devotes a significant portion of his practice to the defense of business litigation, complex commercial, technology, and intellectual property law matters. Veronica A. Bowen is an associate attorney in Segal McCambridge's Chicago office, focusing her practice on toxic tort, intellectual property matters, and employment litigation.



Section 102(b) of the Act specifies the limits of copyright protection. The Act's protection does not "extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery." *Id.* The distinction between Section 102 (a) and Section 102(b) becomes crucial when considering copyrights in relation to computer software.

Copyright and Software

Copyright protection for a computer program extends to all of the copyrightable expression embodied in the program, including "not only the 'literal elements but also …the 'non-literal elements such as code sequence, usage of control structures, and unique or inventive methods of applying normally utilitarian methods, objects, functions, variable of proprietary aspects of a given operating system environment or computer language as an expression of the programmer's original ideas." ***Fair Use and Computer Software*** (2016). Copyright does not protect the more functional aspects of a computer program. ***Copyright Registration of Computer Programs***, U.S. Copyright Office, Cir. 61.

Many courts have considered the functionality debate in terms of the expression versus the idea. All computer code is functional at its most basic level: it tells a machine what to do. But it is how those instructions are expressed that points to the heart of copyright. "An 'expression' may, under some circumstances, be said to be the tangible, fixed form of an idea where the expression's purpose is "to convey information." *Apple Computer Inc. v. Franklin Computer Corp.,* 545. F. Supp. 812, 821 (E.D. Pa. 1982). The Court in *Baker v. Seldon,* 101 U.S. 99, 25 L Ed. 841 (1879), reasoned that "the description of the art in a book, though entitled to the benefit of copyright, lays no foundation for the art itself. The object of one is explanation; the object of the other is use." *Id.* at 105 (holding that an accounting form was not copyrightable, but the explanation of the form was). While the holding in *Baker* came almost 100 years before computer programs were protected under the Copyright Act, the reasoning has been used by modern courts to distinguish between a computer program's expression and its use. *Baker* draws a distinction between Section 102(a), a work "fixed in a tangible medium of expression," and 102(b), a work that primarily functions as the exploitation or use of an idea. 17 U.S.C. §102(a)–(b). According to the House of Representatives Reporter, "Section 102(b) is intended …to make clear that the expression adopted by the programmer is the copyrightable element in a computer program, and that the actual processes or methods embodied in that program are not within the scope of copyright law." H.R. Rep. No. 1476, 94th Cong., 2d Sess. at 56–57, *reprinted* in 1976 U.S. Code Cong. & Admin News at 5670 (general discussion between ideas and expression embodied in section 102(b)).

What Is Fair Use and Where Does Transformative Use Fit In?

While copyright law provides protection and exclusive rights to the owners and authors of copyrighted works, those rights are not absolute. 17 U.S.C. §107 provides a fair use defense for the unauthorized use of a copyrighted work. 17 U.S.C. §107. This defense allows for the "use of a copyrighted work, including such use by reproduction in copies or phonorecords or by any other means specified by that section, for the purposes such as criticism, comment, news reporting, teaching, scholarship, or research," without exposing the user to liability for copyright infringement. 17 U.S.C. §106(a). Section 107 lists four factors to be considered in the determination of fair use: 1) the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes; 2) the nature of the copyrighted work; 3) the amount and substantiality of the portion used in relation to the copyrighted work as a whole; and 4) the effect of the use upon the potential market for or value of the copyrighted work. 17 U.S.C. §107 (1)–(4).

The "purpose and character of the use" has become the primary source for the courts' reference to the transformative qualities of a work as a decisive factor. For the purposes of this article this factor is crucial to the understanding of the Supreme Court's decision in *Google v. Oracle*, 574 U.S. 1071 (2021). The term transformative was initially coined by Judge Pierre Level in his 1990 article "Toward a Fair Use Standard," published in the *Harvard Law Review.* Pierre N. Level, *Toward a Fair Use Standard,* 103 Harv. L. Rev. 1105 (1990). In his article he stated that "if the secondary use adds value to the original—if the quoted matter is used as raw material, transformed in the creation of new information new aesthetics, new insights and understandings—this is the very type of activity that the fair use doctrine intends to protect for the enrichment of society." *Id.* at 1111. While Level coined the term, it was the Supreme Court's 1994 decision in *Campbell v. Acuff-Rose Music Inc,* 510 U.S. 569 (1994), that gave rise to transformativeness being a critical fair use factor. The Court stated that "[t]he central purpose of this investigation is to see… whether the new work merely supersedes the objects of the original creation, …or instead adds something new, with further purpose or different character, altering the first with new expression, meaning, or message; it asks, in other words, whether and to what extent the new work is transformative.…" *Id.* at 579. The Court went onto to reason that the "goal of copyright,

All computer code is functional at its most basic level: it tells a machine what to do. But it is how those instructions are expressed that points to the heart of copyright. "An 'expression' may, under some circumstances, be said to be the tangible, fixed form of an idea where the expression's purpose is "to convey information."

to promote science and the arts, is generally furthered by the creation of transformative works." *Id.* According to the Court, "such works… lie at the heart of the fair use doctrine's guarantee of breathing space within the confines of copyright… and the more transformative the new work, the less will be the significance of the other factors." *Id.* Courts have even allowed for verbatim copying "so long as the copy serves a different function than the original work." *Press v. Patton*, 769 F. 3d 1232, 1262 (11th Cir. 2014).

Copyright law has always operated with fluidity instead of relying upon bright-line rules. That fluidity allows for copyright protection to be applied more broadly and to adapt when necessary; however the speed of its evolution pales in comparison to the speed at which new technologies emerge.

How Does Fair Use Work for Software Copyrights?

While fair use has long been used as a defense in most copyright litigation, it was not initially a widely used defense in computer software cases. The first decade of software litigation concentrated on whether protection was afforded to software at all and if so to what extent. Paul Samuelson, *Fair Use For Computer Programs And Other Copyrightable Works In Digital Form: The Implications Of Sony, Galoob And Sega*, 1 J. Intell. Prop. L 49, 51 (1993–1994). The concept of a computer software work being transformative has been discussed even less in case law.

In 2006, the Ninth Circuit did have a chance to address the question of transformativeness when related to computer software. In *Wall Data Inc. v. L.A. County Sheriff's Dep't*, the court considered whether the L.A. County's Sheriff's Department's use of Wall Data Inc.'s software was transformative. 447 F.3d 769 (9th Cir. 2006). The sheriff's department purchased 3,663 licenses to Wall Data's computer software. *Id.* The department then installed the software onto 6,007 computers. The computers were set up so that total number of workstations able to access the installed software never exceeded the total number of licenses purchased. *Id.* The sheriff's department argued that its use was transformative because its use "did not involve exploitation of the intrinsic value of the author's creation." *Wall Data Inc. v. L.A. County Sheriff's* Dep't, 2004 U.S. 9th Cir. Briefs Lexis 402 at 38 (2004). The sheriff's department reasoned that because the copies were inaccessible for the only purpose of the software, the use in question did not involve any use of the inherent copyright and was therefore transformative. *Id.* The court rejected this reasoning and stated that the sheriff's department created exact copies of the software and put those copies to the identical purpose as the original software. *Wall Data Inc.*, 447 F. 3d at 778.

This very limited look at the issue of transformativeness in relation to computer software highlights the lag between the law and technology. Though computer software has been protected since 1980, the caselaw is somewhat lacking when it comes to discussing the full breadth of copyright law, including both copyrightability and fair use.

The Relationship Between Copyright and Technology

It is no secret that the law has always struggled to keep up with technology, and copyright is certainly no different. It has been estimated that the law is at least five years behind developing technologies. JHTL, ***A Losing Game: The Law Is Struggling to Keep Up with Technology***, (2019). While copyright law has been used to protect an author's intellectual creativity since 1790, the law has continuously been forced to evolve to broaden the scope of its protections while allowing for certain fair uses of otherwise protected material. This evolution has caused copyright to cross paths with new emerging technologies, a relationship that has been anything but smooth sailing. Early on, copyright law dealt with the question of protecting perforated music rolls played on gramophones and the player pianos. *White-Smith Musical Publishing Co. v. Apollo Co.*, 209 U.S. 1 (1908). Technological advances from the printing press to the internet continued to provide new tools for expanding forms of creative expression while challenging the constructs of copyright application.

Copyright law has always operated with fluidity instead of relying upon bright-line rules. That fluidity allows for copyright protection to be applied more broadly and to adapt when necessary; however the speed of its evolution pales in comparison to the speed at which new technologies emerge. It has been argued that "technology creates an environment of rapid and unpredictable changes causing legal delay and legal uncertainty." Ben Depoorter, *Technology and Uncertainty the Shaping Effect on Copyright Law.* UC Hastings Scholarship Repository (2009). The U.S. Copyright Office has issued several studies and reports since the 1950s regarding copyright law's application to emerging technologies. Congress has sought to respond to and anticipate changes in technology, including digital communications, software-enabled consumer products, and other internet related issues. New "technologies continue to raise novel questions about how copyright should be applied and whether the law should be revised to fully realize the Founders' goal that copyright 'promote the Progress of Science.'" George Thuronyi. ***Copyright Law and New Technologies: A Long and Complex Relationship*** (2017).

Computer software is no different in this regard. The first attempt to register a computer program with the Copyright Office came in 1961, with the first registration being awarded in 1964. In 1980, the Copyright Act was amended to confirm the copyrightability of computer programs. Congress defined a computer program as "a set of statements or instructions to be used directly or indirectly in a computer in order

to bring about a certain result." Computer programs are generally recognized as copyrightable as literary works because they are "written in English or a language readable by human beings and contain intelligible messages." Richard Stern, **Another Look at Copyright Protection of Software: Did the 1980 Act Do Anything for Object Code?**, 3 Computer L.J. 1, at 4 (1981). While computer programs have been deemed copyrightable, application of the law has proven to be less than straightforward.

Copyright protection for a computer program extends to all of the copyrightable expression embodied in the program, including "not only the 'literal elements but also …the 'non-literal elements such as code sequence, usage of control structures, and unique or inventive methods of applying normally utilitarian methods, objects, functions, variable of proprietary aspects of a given operating system environment or computer language as an expression of the programmer's original ideas." Copyright does not protect the more functional aspects of a computer program. Because computer programs contain both functional and non-functional aspects, the question of copyrightability has been a complex journey. The copyright bar and judiciary has once again had to play catch up in understanding the way that computer software works as well as the way the industry operates in order to adapt copyright law to the field.

The Google v. Oracle Decision

As stated in the introduction above, the U.S. Supreme Court recently attempted to clarify the dividing line of fair use of software copywrites; however, its decision in *Google v. Oracle* creates more questions than answers, leaving software companies to ponder whether the doctrine of transformative use has any limit.

In 2005, Google purchased Android, Inc. with the goal of breaking into the emerging smartphone market. Google's plan was simple: create a smartphone operating system that was "open source"—allowing developers more freedom in app development, marketing, and sales than would be available on a "closed source" system like Apple's competing iPhone's iOS. It was Google's hope that an open-source system would attract app developers to Android, attracting consumers and smartphone manufacturers and expanding Google's profitable advertising platform. Android had the open-source code necessary for this ambitious plan, but Google realized that its open-source system would only attract developers if it used an accessible programming language.

Oracle owned a programming language known as "Java," which is an extremely malleable platform favored by developers for programming on a variety of operating systems. (While the litigation was pursued by Oracle America, Inc., the original Java negotiation proceeded between Google and Oracle's predecessor, Sun Microsystems, Inc. In the interest of clarity and brevity, we just refer to Sun Microsystems and Oracle jointly as "Oracle.") Google saw Java as the key to Android's success and contacted Oracle about licensing Java for Android. Oracle's business model focused on licensing the copyrighted Java language to other companies, such as Microsoft and Apple, for use in designing software, and Oracle and Google engaged in multiple rounds of negotiations on licensing. Unfortunately, negotiations broke down, and Google decided to make its own programming language platform from scratch.

Google assigned over one-hundred computer engineers to design the Android programming platform (eventually known as Android's "SDK"), dedicating three years of time to this project that produced 2.86 million of lines of code. The project reverse engineered Java's code, and rebuilt a similar system using new code to avoid copyright infringement issues. However, Google copied approximately 11,500 lines of higher-level code directly from Java's Application Programming Interface ("API").

API code is the control panel for a developer and is how a developer commands an app to use base level code. A developer uses the API to tell an app how, when, and where to complete certain functions, allowing the developer to make an app by writing a few-hundred lines of commands rather than writing and re-writing tens-of-thousands of lines of individual code for each individual step of a particular function. An effective API is a huge timesaver and makes the extremely complex programs underlying modern apps possible. Oracle's copyrighted Java had the mostly commonly used API in the industry and copying Java's API to Android meant that developers could seamlessly transition between developing in Java and Android by using the same commands even though the two languages used different base level code. In other words, of the millions of lines of code within Java, Google copied the 11,500 that made Java valuable to programmers.

Because computer programs contain both functional and non-functional aspects, the question of copyrightability has been a complex journey. The copyright bar and judiciary has once again had to play catch up in understanding the way that computer software works as well as the way the industry operates in order to adapt copyright law to the field.

The Bigger Picture

Oracle filed suit for copyright infringement, and Google claimed that copying the API fell under fair use. Oracle's claims relied upon traditional copyright principles—it had a copyrighted work in Java that Google infringed for its commercial benefit. However, Google's fair use argument was novel, taking into effect the unconventional nature of software copyrights. Google asserted that its use was fair because API is conceptual and developers' common use of a familiar API ("reimplementation") facilitates programming across operating systems, platforms, and

even generations of technology—allowing different software and apps to have maximum compatibility and freely share data regardless of hardware and operating systems (a phenomenon known as "interoperability"). Google's argument hinged on a slippery slope argument, particularly an end to software innovation due to a lack of interoperability caused by the loss of a common language. In other words, an end to interoperability would spell an end to creativity in software development. Many influential companies and organization in the software development community agreed and contributed amicus curiae briefs in support of Google's position.

In sum, the Court concluded that Google's use of the API was fair under the "nature of the work" factor given the conceptual, organizational nature of its value as a work.

This matter underwent an extensive and complex procedural history, including two separate trials and appeals to the Federal Circuit. While the jury and Federal Circuit court were convinced that the API was copyrightable, there was a great deal of conflict over whether Google's use could be considered fair. The U.S. Supreme Court assumed that the API was copyrightable, a fact with which the dissent wholly agreed, and focused its analysis on whether Google's use of the API constituted a fair use. While the Supreme Court's analysis considered all four factors in turn, it relied almost exclusively on the doctrine of transformative use in reaching its decision.

Reliance on Transformative Use

Justice Steven Breyer authored the majority decision, starting with an analysis of the "nature of the work" factor, noting that Java's copyright inextricably bound clearly copyrightable programming code with the non-copyrightable, high-level organizational methods and concepts. This meant that while the entirety of Java's language is a creative work entitled to copyright protection, the nature of its creativity and extent of protection available would vary depending upon what part of the code is being considered. Base level implementing code is the unique language that forms the foundation of a software program, making it inherently creative and requiring the strongest protections and limiting the application of fair use despite its functional element. Meanwhile, API language derives its creativity from how intuitive and accessible it is for developers, broadening the application of fair use because intuitive design is more conceptual and organizational. The Court illustrated the conceptual and organizational nature of the API by pointing out that its value derived not from the intuitiveness of the API alone, but from programmers' willingness to learn and adopt the API—with the Court considering the API as less of an independently creative work and more of a gateway to the creative work that is the base level implementing code. In sum, the Court concluded that Google's use of the API was fair under the "nature of the work" factor given the conceptual, organizational nature of its value as a work.

However, in his dissenting opinion, Justice Clarence Thomas pointed out a major flaw with the Court's analysis on the nature of Oracle's copyright—the Copyright Act does not distinguish between levels of copyrighted code. Per the Act, a code can be copyrighted in its entirety—something on which both the Court and dissent agreed. However, the Court created a sliding scale of protections that depended upon the level at which the code is copied. Justice Thomas noted that this presented a substantial change to copyright, completely altering the extent to which copyright's protections actually protect copyrighted code.

Having found that the "nature of the work" factor favored fair use, the Court turned to the "purpose" factor and the impact of transformative use on a software copyright. The Court considered Google's use of the API as transformative under the fair use doctrine based upon its conceptual nature and value to developers. In short, Google created its own base level implementing code and only copied Java's API to foster developers' creativity on the Android platform. Thus, Google's infringement "was consistent with [] creative progress" and was a transformative fair use.

While the Supreme Court recognized that the purpose factor must also consider the "commerciality" and "good faith" of the alleged fair use, it quickly dispensed with these factors as being heavily outweighed by the transformative nature of the use—even suggesting that these factors are of limited importance in a fair use analysis. On their face, these factors should have been very problematic for Google and were at the core of Oracle's infringement argument; after all, Google copied the Java API to ensure the commercial success of Android after Oracle refused to license its copyrighted Java language according to Google's terms. However, the Supreme Court noted that most fair uses have some commercial element, holding that infringing a copyright for a commercial purpose can still constitute fair use. The Court also expressed its skepticism as to whether bad faith should even be considered relative to fair use before concluding that bad faith would not counterbalance the value of the transformative use here. More than just simply finding transformative use to outweigh commerciality and bad faith, the Court's decision effectively questions whether commerciality and bad faith are even relevant to the consideration of fair use—drastically changing this doctrine relative to software copyrights.

The Court then quickly addressed the last two factors by again relying on transformative use. Interestingly, it found that the copying of arguably the most-important 11,500 lines of code was insubstantial not because of its volume compared to the millions of lines of uncopied code but, rather, because the use was transformative. The Court found that the marketability factor also favored Google because (1) Oracle's Java had not significantly broken into the smartphone market before Android, (2) Android's success led to greater use of Java in the smartphone marketplace, favoring Oracle, and (3) finding infringement would stymie creativity of develop-

ers by "locking" familiar programming languages behind copyright.

However, as the dissent noted, Google's infringement caused Android to lose its mobile phone market share, worth tens-of-billions of dollars. Moreover, Google replaced the infringing code with a new API in 2014 without detriment to its majority market share, and Google's main competitor, Apple, created its own highly marketable and successful API for its smartphone. These facts, considered together, effectively undermine the "locking" argument relied upon by the Court as support for its transformative use analysis. The dissent thus concluded that three of the four fair use factors favored Oracle, with only the "nature of the work" factor possibly, but not conclusively, favoring Google. Nonetheless, the majority disagreed, largely because its consideration of transformative use took precedence over every other factor and element for consideration of fair use.

After concluding that all four factors confirmed a fair use, the Supreme Court cautioned that its opinion did not overturn prior decisions on copyright and fair use; rather, it implemented the traditional fair use framework to a "different kind of copyrighted work"—suggesting that the fair use analysis is ill suited to a software copyright. The Court explained that copyrighted computer programs are protected as creative works but are also inherently functional, making them very distinguishable from traditional copyrighted works and necessitating a unique approach to fair use. The Court's ultimate conclusion was that Google took only what was needed to allow developers to use their talents for new, creative applications on smartphones, establishing a transformative fair use.

"Transformativeness:" The New Standard?

Admittedly, the *Google v. Oracle* Court faced a square-peg-round-hole scenario in trying to fit a software copyright into the artistically oriented fair use doctrine, but its analysis fosters confusion by failing to establish a workable framework for considering fair use in software development. While the four fair use factors may not be easily applicable to the world of software development, the Court's decision essentially simplifies the fair use analysis down a single factor: "transformativeness." Transformativness is seemingly based upon a case-by-case analysis of immeasurable, intangible concepts such as intuitiveness, familiarity, and promotion of creativity. Moreover, the transformativeness analysis turns transformative value into the *uber*-factor, allowing a court to use the transformative nature of the work to supplant the more tangible, traditional economic and legal analyses underlying the "commerciality," "bad faith," and "substantiality" considerations. The *Google v. Oracle* decision erodes the limited value of the fair use framework in software copyright cases, leaving in its stead an extremely malleable "transformativeness" analysis that could be used by a district court as tool to conform the facts of a software copyright infringement case arbitrarily to whatever decision that Court would like to make.

Despite its overall focus on the transformative nature of Google's infringement, the Supreme Court did provide a more complete economic analysis on the marketability factor. However, this analysis also establishes a concerning precedent for software copyrights. The Court's analysis relies upon the fact that Oracle did not significantly break into the smartphone market until after Google infringed on the Java API, suggesting that this fact pattern favored Google's infringement as a more marketable fair use. However, the dissent pointed out the primary legal issue undercutting this analysis—copyright protects the holder's intellectual property regardless of whether the holder personally uses the copyright in every available market. Using the example of an author licensing his work for a film adaptation, the dissent asserted that copyright gives the holder the right to develop a market for the intellectual property or to license it to another to develop a market. That was Oracle's business plan, suggesting that Google's infringement was anything but fair.

The Supreme Court's opinion validates the argument that a software copyright can be infringed so long as the infringer makes a better economic use of the software product. While not exactly a new argument in copyright analysis, the substantiation of a "better use" argument here may prove problematic for companies like Oracle whose business relies on licensing copyrighted works. Oracle designed Java to be malleable and intuitive so it would become the platform of interoperability as a part of a business plan to sell Java licenses. If a company like Oracle is unable to protect its copyright because of the same interoperability and malleability that makes its copyright valuable, then there is no economic incentive to develop highly functional, interoperable systems and creativity is stymied.

The Court explained that copyrighted computer programs are protected as creative works but are also inherently functional, making them very distinguishable from traditional copyrighted works and necessitating a unique approach to fair use.

Conclusion

Google v. Oracle provides little in the way of guidance for lowers courts considering fair use in determining the propriety of software copyrights, instead granting the lower courts wide discretion through its broad transformative use analysis. For software companies who have protected their intellectual property by copyright, the Supreme Court's decision is concerning to say the least, allowing an infringer to use copyrighted programming language for commercial purposes with seeming impunity based on the argument that the copyrighted program is too just too valuable to restrict. Those opposing a similar result in subsequent litigation will likely characterize the circumstances in Google as unique or otherwise inapplicable. In any event, expect an increased emphasis on transformative use in software copyright disputes. FD