Software

11

Software

> The programmer, like the poet, works only slightly removed from pure thought-stuff. He builds his castles in the air, from air, creating by exertion of the imagination. Few media of creation are so flexible, so easy to polish and rework, so readily capable of realizing grand conceptual structures. Yet the program construct, unlike the poet's words, is real in the sense that it moves and works, producing visible outputs separate from the construct itself. The magic of myth and legend has come true in our time. One types the correct incantation on a keyboard, and a display screen comes to life, showing things that never were nor could be.
>
> Frederick P. Brooks, *The Mythical Man-Month* (1975)

Software presents a subtly different functionality problem than physical designs do. Software is plainly functional: when run on a computer, it does something. But there are at least two distinct ways in which software might be more than just functional. First, the software's code might contain nonexpressive elements: multiple different programs can do the same thing, so a programmer typically has at least some design choices not dictated by a given function. Second, the thing the software does might be expressive: it might show a movie or play a song. (Recall *Stern*'s treatment of a video game as an audiovisual work.) Complicating matters even more, a program's output might itself be both aesthetic and functional, which is typically the case for user interfaces. Pay attention to how each body of intellectual property law teases out these different aspects of software.

A Trade Secret

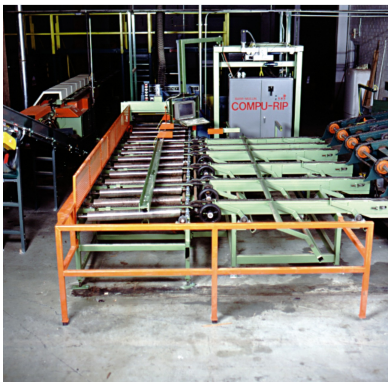
Trade secret protection is obviously available and effective for software used internally within a business. The more interesting question is whether and how a business can make software available to

others while mainting at least some of the software's design as a secret. A later chapter will consider whether contractual restrictions help, but for now, focus on the practical question of what software necessarily discloses to its users.

Barr-Mullin, Inc. v. Browning

424 S.E.2d 226 (N.C. Ct. App. 1993)

According to defendants, the COMPU-RIP software is not a trade secret since it is (1) not subject to reasonable efforts to maintain its secrecy and (2) defendants reverse engineered this software. The COMPU-RIP software is contained in the form of "programmable read-only memory chips" (PROMS) imbedded in the COMPU-RIP machinery. These PROMS contain only the "object code" version of the computer program. This is the version of the computer software which is "read" by the computer's machinery. Computer programmers do not write computer software in object code; rather, the software is written in "source code" and then translated into object code so that the computer can execute the program. Since the COMPU-RIP software was sold in PROM form, the source code was not available to the general public.



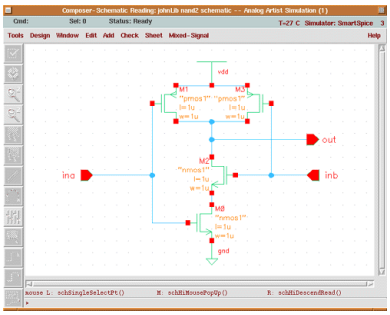
Barr-Mullin Compu-Rip rip saw feeder

The affidavits of Timothy Toombs and Gary Ruggles, who holds a Ph.D. in Electrical Engineering, indicate that because the COMPU-RIP software is distributed in object code form it is practically impossible to make any meaningful changes to the software. This evidence establishes the COMPU-RIP software was subject to reasonable efforts to maintain its secrecy. As to the question of reverse engineering, the affidavits indicate that it is practically impossible to make any modification to the COMPU-RIP software using only the object code contained in the PROMS. We find the evidence presented establishes the COMPU-RIP software was not "readily ascertainable" through reverse engineering.

Silvaco Data Systems v. Intel Corp.

184 Cal. App. 4th 210 (2010)

Silvaco develops and markets computer applications for the electronic design automation (EDA) field, which covers the entire complex process of designing electronic circuits and systems. Among the various subcategories of EDA software are circuit simulators, which permit the designer to create a virtual model of a proposed circuit in order to test its properties before incurring the expense and delay of manufacturing a working prototype.



SmartSpice screenshot

Among Silvaco's software products is SmartSpice, an analog circuit emulator. In December 2000, Silvaco filed a suit against Circuit Semantics, Inc. (CSI), a competing developer of EDA software, alleging that CSI, aided by two former Silvaco employees, had misappropriated trade secrets used in SmartSpice, and had incorporated them in its own product, DynaSpice. Silvaco eventually secured a judg-

ment against CSI. It then brought actions against several purchasers of CSI software, including Intel. It alleged that by using CSI's software, these end users had misappropriated the Silvaco trade secrets assertedly incorporated in that software.

The parties appear to agree, and we may accept for purposes of this opinion, that "source code" describes the text in which computer programs are originally written by their human authors using a high-level programming language.[4]. One who possesses the source code for a program may readily ascertain its underlying design, and may directly incorporate it, or pieces of it, into another program. In order to yield a functioning computer application, however, source code must generally be translated or "compiled" into machine-readable (executable) code. After a program is compiled, it may still be represented as text, but the text is not readily intelligible to human beings, consisting of strings of binary (base 2) or hexadecimal (base 16) numbers.[5] For this reason, the source code for many if not most commercial software products is a secret, and may remain so despite widespread distribution of the executable program.

A. Question Presented

The contention presents a question of law: whether a defendant can be liable for misappropriation of a trade secret which is admittedly embodied in source code, based upon the act of executing, on his own computer, executable code allegedly tainted by the incorporation of design features wrongfully derived from the plaintiff's source

[4]This is an accurate enough description where, as here, the facts conform to what one writer has called the "Standard Scenario." Ed Felten, Source Code and Object Code. In this scenario, human programmers write a program in a high-level programming language; this source code is then processed through software known as a compiler to produce object code that may be executed on a machine. Departures from the standard scenario, however, are increasingly common. Another writer asserts that among programmers the terms "source" and "object" do not really describe "well-defined classes of code" but are "actually relative terms." David S. Touretzky, Source vs. Object Code: A False Dichotomy He continues, "Given a device for transforming programs from one form to another, source code is what goes into the device, and object code (or 'target' code) is what comes out. The target code of one device is frequently the source code of another." The upshot seems to be that the correct use of these terms reflects variables that may not have any particular legal significance. Here, the essential variable for legal purposes is the extent to which the code reveals the underlying design, i.e., the methods and algorithms used by the developer. *See* Felten (suggesting distinctions based on whether code is in "'he form in which programmers customarily read and edit it'" and the extent to which it is "human-readable"). The distinguishing feature for our purposes is that what the parties call "source" code is written by programmers and readily understood by them, whereas object or machine code is compiled by and for machines and does not readily yield its underlying design to human understanding.

[5]For example, the instruction `MOV` for an Intel 32-bit microprocessor may be represented as `10110000 01100001` in binary code or `B0 61` in hexadecimal.

code. It is undisputed that the object code executed by Intel could not disclose the underlying source code or permit the exploitation of its features and design. It could not, in short, impart knowledge of the trade secret. The question is whether, in such circumstances, Intel could be found to have misappropriated Silvaco's trade secrets.

B. Trade Secret

It is critical to any CUTSA cause of action – and any defense – that the information claimed to have been misappropriated be clearly identified. Pursuant to this requirement, Silvaco filed a document, under seal, designating the trade secrets claimed by it. The designation identified trade secrets in six categories. The first five categories referred only to source code.

The sixth category of claimed trade secrets was described as "the ... trade secrets identified in Exhibit B ... and the source code implementing such trade secrets." Exhibit B consists of 22 pages of technical verbiage most of which may be readily intelligible only to those within the EDA field. This much, however, seems reasonably clear: the exhibit does not designate information as such but rather describes various features, functions, and characteristics of the design and operation of Silvaco's software products. Thus the first of the 24 listed subcategories is a "proprietary method" of carrying out a function apparently found in competing programs as well. This asserted secret is also described as "a methodology for" implementing that function, apparently in an unusual way, which "contributes to performance and accuracy improvements." This "trade secret methodology" is "implemented" by two named "modules," also described as "functions," which "represent part" of the critical "algorithm." Three "unique features" of this method are listed: The "integration" of two other operations; a "method" of "changing and controlling" a variable, which "affects the performance of the simulation"; and a mode of "implementation" that produces "efficiency."

Do you understand what the court is talking about here? Did the court? Silvaco's lawyers?

Silvaco's sixth category thus appears to attempt to characterize various aspects of the underlying design as trade secrets. The design may constitute the basis for a trade secret, such that information concerning it could be actionably misappropriated; but it is the information – not the design itself – that must form the basis for the cause of action. And while the finished (compiled) product might have distinctive characteristics resulting from that design – such as improved performance – they cannot constitute trade secrets because they are not secret, but are evident to anyone running the finished program. Indeed, to the extent they tend to disclose the underlying design, it ceases to be a protectable secret for that same reason. The sixth category therefore fails to describe a trade secret other than source code. Since none of the other categories even purport to do so, Intel is quite

correct to premise its argument on the proposition that the only trade secrets at issue are found in Silvaco's source code.

C. Misappropriation

Misappropriation of a trade secret may be achieved through three types of conduct: "acquisition," "disclosure," or "use."

There is no suggestion that Intel ever disclosed Silvaco's source code to anyone, and it is difficult to see how it might have done so since there is no evidence that it ever had the source code to disclose. Silvaco emphasizes that wrongful acquisition of a trade secret may be actionable in itself. But there is no basis to suppose that Intel ever "acquired" the source code constituting the trade secrets.

Silvaco has never explained how any conduct by Intel constituted "use" of its source code. One clearly engages in the "use" of a secret, in the ordinary sense, when one directly exploits it for his own advantage, e.g., by incorporating it into his own manufacturing technique or product. But "use" in the ordinary sense is not present when the conduct consists entirely of possessing, and taking advantage of, something that was made using the secret.

One who bakes a pie from a recipe certainly engages in the "use" of the latter; but one who eats the pie does not, by virtue of that act alone, make "use" of the recipe in any ordinary sense, and this is true even if the baker is accused of stealing the recipe from a competitor, and the diner knows of that accusation. Yet this is substantially the same situation as when one runs software that was compiled from allegedly stolen source code. The source code is the recipe from which the pie (executable program) is baked (compiled). Nor is the analogy weakened by the fact that a diner is not ordinarily said to make "use" of something he eats. His metabolism may be said to do so, or the analogy may be adjusted to replace the pie with an instrument, such as a stopwatch. A coach who employs the latter to time a race certainly makes "use" of it, but only a sophist could bring himself to say that coach "uses" trade secrets involved in the manufacture of the watch.

Intel appears to have been in substantially the same position as the customer in the pie shop who is accused of stealing the secret recipe because he bought a pie with knowledge that a rival baker had accused the seller of using the rival's stolen recipe. The customer does not, by buying or eating the pie, gain knowledge of the recipe used to make it.

Strong considerations of public policy reinforce the commonsense conclusion that using a product does not constitute a "use" of trade secrets employed in its manufacture. If merely running finished software constituted a use of the source code from which it was compiled, then every purchaser of software would be exposed to liability if it

were later alleged that the software was based in part upon purloined source code. This risk could be expected to inhibit software sales and discourage innovation to an extent far beyond the intentions and purpose of CUTSA.

B Patent

CLS Bank is obviously a software-patent case and it may help to review it now. The first two subsections here show how we got to *CLS Bank* and what the Federal Circuit has been doing to apply it. The third gives a look at the state of contemporary software-patent litigation by looking at the obviousness analysis in a recent high-profile case.

1 Pre-*CLS Bank* § 101 Caselaw

How much space is left for software patents after *CLS Bank*? To get a sense, we need to take a look at the Federal Circuit's theory of software patents pre-*CLS Bank*.

In re Bernhart

417 F.2d 1395 (CCPA 1969)

For a discussion of the mathematics involved, see Andrew Chin, *Ghost in the "New Machine"*, 16 N.C. J.L. & Tech. 623 (2015).

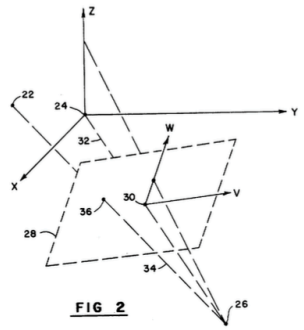


Figure 2 from the application

The patent ultimately issued as U.S. Pat. No. 3,519,997

[Two Boeing employees developed a mathematical technique to display two-dimensional representations of three-dimensional objects. They claimed, for example:]

13 A plotting method ... comprising:

(a) a first step of programming the computer to compute the position of planar Cartesian coordinate axes in the given plane relative to the given set of object points …

(c) the step of applying the computer output to the input of a planar plotting apparatus adapted to provide on a plane a succession of straightline segments that connect between sequential points having positions corresponding to the coordinates computed by the second step.

18 A system for providing a drawing of an object comprising in combination: electronic digital computer means programmed to respond to applied signals (xe,ye,ze) and a series of groups of signals (xi,yi,zi) to provide a corresponding series of pairs of output signals (vi,wi) with the relationship between signals (xi,yi,zi) and (xe,ye,ze) to the signals (vi,wi) being defined as follows …

Looking first at the apparatus claims, these claims recite, and can be infringed only by, a digital computer in a certain physical condition, i. e., electro-mechanically set or programmed to carry out the recited

routine. Accordingly, we think it clear that applicants have not defined as their invention anything in which the human mind could be used as a component. Nor are the "printed matter" cases, cited by the board, supra, controlling as to these apparatus claims either on the facts or in principle. On their facts, those cases dealt with claims defining as the invention certain novel arrangements of printed lines or characters, useful and intelligible only to the human mind. Here the invention as defined by the claims *requires* that the information be processed not by the mind but by a machine, the computer, and that the drawing be done not by a draftsman but by a plotting machine.

There is one further rationale used by both the board and the examiner, namely, that the provision of new signals to be stored by the computer does not make it a new machine, i.e. it is *structurally* the same, no matter how new, useful and unobvious the result. This rationale really goes more to novelty than to statutory subject matter but it appears to be at the heart of the present controversy. To this question we say that if a machine is programmed in a certain new and unobvious way, it is physically different from the machine without that program; its memory elements are differently arranged. The fact that these physical changes are invisible to the eye should not tempt us to conclude that the machine has not been changed. If a new machine has not been invented, certainly a "new and useful improvement" of the unprogrammed machine has been, and Congress has said in § 101 that such improvements are statutory subject matter for a patent. It may well be that the vast majority of newly programmed machines are obvious to those skilled in the art and hence unpatentable under § 103. We are concluding here that such machines are statutory under § 101, and that claims defining them must be judged for patentability in light of the prior art.

In re Alappat

33 F.3d 1526 (Fed. Cir. 1994)

[Alappat's invention dealt with the problem of displaying continuous analog signals on a discrete digital display. Illuminating only the closest pixels to the signal results in jagged, uneven edges. In Alappat's method, all the pixels close to the signal are illuminated, with brightness proportional to how close they are to the signal.] Employing this anti-aliasing technique eliminates any apparent discontinuity, jaggedness, or oscillation in the waveform, thus giving the visual appearance of a smooth continuous waveform. In short, and in lay terms, the invention is an improvement in an oscilloscope comparable to a TV having a clearer picture.Figure 4 from the application

(Rich, J.)

images/alappat.png

Claim 15, the only independent claim in issue, reads:

The patent issued as U.S. Pat. No. 5,440,676

> A rasterizer for converting vector list data representing sample magnitudes of an input waveform into anti-

aliased pixel illumination intensity data to be displayed on a display means comprising:

(a) means for determining the vertical distance between the endpoints of each of the vectors in the data list;

(b) means for determining the elevation of a row of pixels that is spanned by the vector;

(c) means for normalizing the vertical distance and elevation; and

(d) means for outputting illumination intensity data as a predetermined function of the normalized vertical distance and elevation.

The Board erred in its reasoning that claim 15 is unpatentable merely because it "reads on a general purpose digital computer 'means' to perform the various steps under program control." Alappat admits that claim 15 would read on a general purpose computer programmed to carry out the claimed invention, but argues that this alone also does not justify holding claim 15 unpatentable as directed to nonstatutory subject matter. We agree. We have held that such programming creates a new machine, because a general purpose computer in effect becomes a special purpose computer once it is programmed to perform particular functions pursuant to instructions from program software.

Under the Board's reasoning, a programmed general purpose computer could never be viewed as patentable subject matter under § 101. This reasoning is without basis in the law. The Supreme Court has never held that a programmed computer may never be entitled to patent protection. Indeed, the *Benson* court specifically stated that its decision therein did not preclude "a patent for any program servicing a computer." Consequently, a computer operating pursuant to software *may* represent patentable subject matter, provided, of course, that the claimed subject matter meets all of the other requirements of Title 35. In any case, a computer, like a rasterizer, is apparatus not mathematics.

Archer, Chief Judge, dissenting:

I disagree with the majority's conclusion that Alappat's "rasterizer," which is all that is claimed in the claims at issue, constitutes an invention or discovery within 35 U.S.C. § 101. Alappat has arranged known circuit elements to accomplish nothing other than the solving of a particular mathematical equation represented in the mind of the reader of his patent application.

Consider for example the discovery or creation of music, a new song. Music of course is not patentable subject matter; a composer cannot obtain exclusive patent rights for the original creation of a mu-

sical composition. But now suppose the new melody is recorded on a compact disc. In such case, the particular musical composition will define an arrangement of minute pits in the surface of the compact disc material, and therefore will define its specific structure. Alternatively suppose the music is recorded on the rolls of a player piano or a music box.

Through the expedient of putting his music on known structure, can a composer now claim as his invention the structure of a compact disc or player piano roll containing the melody he discovered and obtain a patent therefor? The answer must be no. The composer admittedly has invented or discovered nothing but music. The discovery of music does not become patentable subject matter simply because there is an arbitrary claim to some structure.

And if a claim to a compact disc or piano roll containing a newly discovered song were regarded as a "manufacture" and within § 101 simply because of the specific physical structure of the compact disc, the "practical effect" would be the granting of a patent for a discovery in music. Where the music is new, the precise structure of the disc or roll would be novel under § 102. Because the patent law cannot examine music for "nonobviousness," the Patent and Trademark Office could not make a showing of obviousness under § 103. The result would well be the award of a patent for the discovery of music.

Alappat admits that each of the circuitry elements of the claimed "rasterizer" is old. He says they are merely "form." Thus, they are only a convenient and basic way of electrically representing the mathematical operations to be performed, that is, converting vector data into matrix or raster data. In Alappat's view, it is the new mathematic operation that is the "substance" of the claimed invention or discovery. Claim 15 as a whole thus claims old circuitry elements in an arrangement *defined by a mathematical operation*, which only performs *the very mathematical operation that defines it.* Rather than claiming the mathematics itself, which of course Alappat cannot do, Alappat claims the mathematically defined structure. But as a whole, there is no "application" apart from the mathematical operation that is asserted to be the invention or discovery. What is going on here is a charade.

This is not to say that digital circuitry cannot be an element in an otherwise statutory machine. Under *Diehr*, it can.[26]

Thus unlike the rubber curing process in *Diehr*, the claimed rasterizer here is not an application of mathematics in an otherwise statu-

[26]Likewise, but not present in this case, improved digital circuitry itself, such as faster digital processors, would be statutory subject matter. Unlike the "rasterizer" in this case, they are not simply a claimed arrangement of circuit elements defined by a mathematical operation which does nothing more than solve the operation that defines it.

tory process or product. The rasterizer is simply the mathematical conversion of data. In *Diehr*, the input data were derived by a claimed component of the overall rubber curing process – the press and thermocouple – which fed data to the claimed computer. Here, however, as the specification and claims indicate, the waveform data converted by the claimed rasterizer are not required to come from a particular machine connected up to the rasterizer, and, as Alappat admits, it does not matter how the data are selected. The sets of waveform numbers converted by the claimed rasterizer could come simply from the mind and hand of a person. The end product of the claimed rasterizer is not precisely cured rubber as it was in *Diehr* but rather different data as a mathematical function of the original data. Sure the data have some use. Most data have uses, and that is why people spend time calculating data. But just having some use for data does not make the creation of particular data patentable. Alappat's claimed rasterizer is newly discovered mathematics and not the invention or discovery of a process or product applying it.

Finally, a "general purpose computer" issue has been raised as an aside in this case. Getting back to the music analogy, Alappat is like a composer who claims his song on a compact disc, and then argues that the compact disc is equivalent to a player piano or a music box with the song on a roll or even sheet music because they all represent the same song. The composer is thus clearly asking for (and getting from the majority) a patent for the discovery of a song and a patent covering every physical manifestation of the song.

Thus, a known circuit containing a light bulb, battery, and switch is not a new machine when the switch is opened and closed to recite a new story in Morse code, because the "invention or discovery" is merely a new story, which is nonstatutory subject matter. An old stereo playing a new song on a compact disc is not a new machine because the invention or discovery is merely a new song, which is nonstatutory subject matter. The "perforated rolls [of a player piano] are parts of a machine which, when duly applied and properly operated in connection with the mechanism to which they are adapted, produce musical tones in harmonious combination." *White-Smith*. Yet a player piano playing Chopin's scales does not become a "new machine" when it spins a roll to play Brahms' lullaby.[29]

State Street Bank & Trust Co. v. Signature Financial Group

149 F.3d 1368 (Fed. Cir. 1998)

[29]Of course, a player piano itself could be a new machine, for example in relation to a music box, and, likewise, a player piano capable because of design of improved piano-playing might also be a new machine. In such cases, the invention or discovery is the quality of the structure of the piano – its mode of operation – and not the particular piece of music being played.

The patented invention [U.S. Pat No. 5,193,056] relates generally to a system that allows an administrator to monitor and record the financial information flow and make all calculations necessary for several mutual funds to pool their investment funds into a single portfolio.

Unpatentable mathematical algorithms are identifiable by showing they are merely abstract ideas constituting disembodied concepts or truths that are not "useful." From a practical standpoint, this means that to be patentable an algorithm must be applied in a "useful" way. In *Alappat*, we held that data, transformed by a machine through a series of mathematical calculations to produce a smooth waveform display on a rasterizer monitor, constituted a practical application of an abstract idea (a mathematical algorithm, formula, or calculation), because it produced "a useful, concrete and tangible result" – the smooth waveform.

Similarly, in *Arrhythmia Research Technology Inc. v. Corazonix Corp.*, we held that the transformation of electrocardiograph signals from a patient's heartbeat by a machine through a series of mathematical calculations constituted a practical application of an abstract idea (a mathematical algorithm, formula, or calculation), because it corresponded to a useful, concrete or tangible thing – the condition of a patient's heart.

Today, we hold that the transformation of data, representing discrete dollar amounts, by a machine through a series of mathematical calculations into a final share price, constitutes a practical application of a mathematical algorithm, formula, or calculation, because it produces "a useful, concrete and tangible result" – a final share price momentarily fixed for recording and reporting purposes and even accepted and relied upon by regulatory authorities and in subsequent trades.

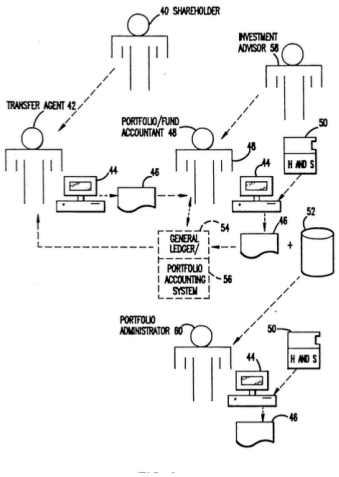


Figure 4 from the patent *Arrythmia*: 958 F.2d 1053 (Fed. Cir. 1992)

2 Post-*CLS Bank* § 101 Caselaw

Bilski and *CLS Bank* obliterated the Federal Circuit's "useful, concrete, or tangible" test and its "new machine" reasoning. But did they leave behind any better foundation for deciding when software inventions are and are not patentable? Consider:

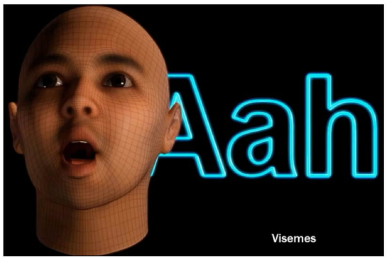
McRO, Inc. v. Bandai Namco Games America Inc.

--- F.3d --- (Fed Cir. Sept. 13, 2016)

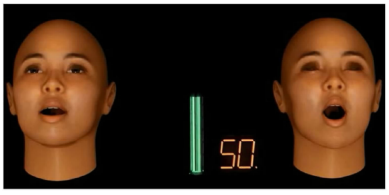
[3D animators depict facial expressions by defining the positions of the vertices for a character's face as it pronounces different phonemes, like "ahh." The set of vertex positions for a phoneme is a "morph target"; the set of positions for a face at rest is the "neutral model." The difference between a morph target and the neutral model is the target's "delta set."Traditionally, animators would identify the "keyframes" for morph targets by working from a "timed transcrpt"

that listed when the character pronounced each phoneme. A computer would then automatically generate intermediate facial expressions between two keyframes by blending the morph targets for the two.]

For example, the face halfway between the neutral model and the "oh" face can be expressed simply by setting the "oh" morph weight to 0.5, i.e., 50%, as shown [in the margin] to the left. The model halfway to the next syllable, in turn, could be expressed by setting both the "oh" morph weight and that for the next syllable each to 0.5, creating a blend of those two delta sets. Because the pronounced phoneme and drawn keyframe corresponded in time, this prior art process synchronized the lips and facial expression of the 3–D character.



Example morph target for the "ahh" phoneme



Two facial expression models

time (sec)	phoneme	word
0		Sil
1.895	h	hello
1.965	eh	
1.995	l	
2.105	o	
2.137	w	
2.165	dh	there
2.235	eh	
2.335	r	
2.435	sil	
2.475	h	how
2.545	a	
2.601	w	
2.635	AA	are
2.66	r	
2.695	y	you
2.835	uw	
2.885	t	today
2.945	ah	
2.985	d	
3.045	e	
3.16	y	
3.225	sil	

Timed transcript of phonemes

The patents [Nos. 6,307,576 and 6,611,278] criticize the preexisting keyframe approach as "very tedious and time consuming, as well as inaccurate due to the large number of keyframes necessary to depict speech." Essentially, the patents aim to automate a 3–D animator's tasks, specifically, determining when to set keyframes and setting those keyframes. This automation is accomplished through rules that are applied to the timed transcript to determine the morph weights. The patents describe many exemplary rule sets that go beyond simply matching single phonemes from the timed transcript with the appropriate morph target. Instead, these rule sets aim to produce more realistic speech by "taking into consideration the differences in mouth positions for similar phonemes based on context."

One exemplary set of rules provided and applied in the specification of the '576 patent is for a character transitioning from silence through saying "hello." This exemplary set of rules provides for inserting a transition starting shortly before the first syllable after a silence. The transition marks when the character begins to transition from silence, shown by the closed-mouthed neutral model, to the morph target for the first syllable, with its open-mouthed shape. That is, the rule automates a character's facial expressions so the character will wait until shortly before it starts speaking to begin opening its mouth. In terms of the prior art method, the effect of this rule is to automatically create a keyframe at a point that no phoneme is being pronounced. If instead no transition were placed at that position, the resulting animation would have an unrealistic quality. The character would open its mouth gradually from the beginning of the sequence through its first utterance as a result of the computer interpolating a continuous transition between those two points. In the prior art system, an animator would have to subjectively identify the problematic sequence and manually fix it by adding an appropriate keyframe. The invention, however, uses rules to automatically set a keyframe at the correct point to depict more realistic speech, achieving results

similar to those previously achieved manually by animators.

Claim 1 of the '576 patent is representative and dispositive of the asserted claims3 for the purposes of appeal:

> A method for automatically animating lip synchronization and facial expression of three-dimensional characters comprising:
>
> - obtaining a first set of rules that define output morph weight set stream as a function of phoneme sequence and time of said phoneme sequence;
> - obtaining a timed data file of phonemes having a plurality of sub-sequences;
> - generating an intermediate stream of output morph weight sets and a plurality of transition parameters between two adjacent morph weight sets by evaluating said plurality of sub-sequences against said first set of rules;
> - generating a final stream of output morph weight sets at a desired frame rate from said intermediate stream of output morph weight sets and said plurality of transition parameters; and
> - applying said final stream of output morph weight sets to a sequence of animated characters to produce lip synchronization and facial expression control of said animated characters.

The defendants are generally video game developers and publishers.

The district court determined that claim 1 of the '567 patent is "drawn to the abstract idea of automated rules-based use of morph targets and delta sets for lip-synchronized three-dimensional animation." We disagree.

Courts must be careful to avoid oversimplifying the claims by looking at them generally and failing to account for the specific requirements of the claims. Here, the claims are limited to rules with specific characteristics. As the district court recognized during claim construction, "the claims themselves set out meaningful requirements for the first set of rules: they define a morph weight set stream as a function of phoneme sequence and times associated with said phoneme sequence." They further require "applying said first set of rules to each sub-sequence of timed phonemes." The specific, claimed features of these rules allow for the improvement realized by the invention.

As the specification confirms, the claimed improvement here is allowing computers to produce "accurate and realistic lip synchronization and facial expressions in animated characters" that previ-

ously could only be produced by human animators. As the district court correctly recognized, this computer automation is realized by improving the prior art through "the use of rules, rather than artists, to set the morph weights and transitions between phonemes." The rules are limiting in that they define morph weight sets as a function of the timing of phoneme sub-sequences. Defendants do not dispute that processes that automate tasks that humans are capable of performing are patent eligible if properly claimed; instead, they argue that the claims here are abstract because they do not claim specific rules. This argument echoes the district court's finding that the claims improperly purport to cover all rules. The claimed rules here, however, are limited to rules with certain common characteristics, i.e., a genus.

Claims to the genus of an invention, rather than a particular species, have long been acknowledged as patentable. Patent law has evolved to place additional requirements on patentees seeking to claim a genus; however, these limits have not been in relation to the abstract idea exception to § 101. Rather they have principally been in terms of whether the patentee has satisfied the tradeoff of broad disclosure for broad claim scope implicit in 35 U.S.C. § 112. It is self-evident that genus claims create a greater risk of preemption, thus implicating the primary concern driving § 101 jurisprudence, but this does not mean they are unpatentable.

Claim 1 is focused on a specific asserted improvement in computer animation, i.e., the automatic use of rules of a particular type. We disagree with Defendants' arguments that the claims simply use a computer as a tool to automate conventional activity. While the rules are embodied in computer software that is processed by general-purpose computers, Defendants provided no evidence that the process previously used by animators is the same as the process required by the claims. Defendants concede an animator's process was driven by subjective determinations rather than specific, limited mathematical rules. McRO states that animators would initially set keyframes at the point a phoneme was pronounced to represent the corresponding morph target as a starting point for further fine tuning. This activity, even if automated by rules, would not be within the scope of the claims because it does not evaluate sub-sequences, generate transition parameters or apply transition parameters to create a final morph weight set. It is the incorporation of the claimed rules, not the use of the computer, that "improved [the] existing technological process" by allowing the automation of further tasks. *CLS Bank* This is unlike *Flook*, *Bilski*, and *Alice*, where the claimed computer-automated process and the prior method were carried out in the same way.

Further, the automation goes beyond merely organizing existing information into a new form or carrying out a fundamental economic

practice. The claimed process uses a combined order of specific rules that renders information into a specific format that is then used and applied to create desired results: a sequence of synchronized, animated characters. While the result may not be tangible, there is nothing that requires a method be tied to a machine or transform an article to be patentable. The concern underlying the exceptions to § 101 is not tangibility, but preemption.

The limitations in claim 1 prevent preemption of all processes for achieving automated lip-synchronization of 3–D characters. McRO has demonstrated that motion capture animation provides an alternative process for automatically animating lip synchronization and facial expressions. Even so, the absence of complete preemption does not demonstrate patent eligibility. The narrower concern here is whether the claimed genus of rules preempts all techniques for automating 3–D animation that rely on rules. Claim 1 requires that the rules be rendered in a specific way: as a relationship between sub-sequences of phonemes, timing, and the weight to which each phoneme is expressed visually at a particular timing (as represented by the morph weight set). The specific structure of the claimed rules would prevent broad preemption of all rules-based means of automating lip synchronization, unless the limits of the rules themselves are broad enough to cover all possible approaches. There has been no showing that any rules-based lip-synchronization process must use rules with the specifically claimed characteristics.

Defendants' attorney's argument that any rules-based lip-synchronization process must use the claimed type of rules has appeal, but no record evidence supports this conclusion. Defendants again rely only on the patents' description of one type of rules, but the description of one set of rules does not mean that there exists only one set of rules, and does not support the view that other possible types of rules with different characteristics do not exist. The only information cited to this court about the relationship between speech and face shape points to the conclusion that there are many other possible approaches to automating lip synchronization using rules. For example, Amicus Public Knowledge cites Kiyoshi Honda, *Physiological Processes of Speech Processing*, as support for the proposition that the claimed rules reflect natural laws. Honda shows, however, that the interaction between vocalization and facial expression is very complex, and there are relationships present other than those required by the claimed rules. This complex interaction permits development of alternative rules-based methods of animating lip synchronization and facial expressions of three-dimensional characters, such as simulating the muscle action underlying characters' facial expressions. Under these circumstances, therefore, we need not assume that future alternative discoveries are foreclosed.

In *Springer Handbook of Speech Production* (Jacob Benesty et al. eds., 2008)

Because we find that claim 1 is not directed to ineligible subject matter, we do not reach Alice step two.

--- F.3d --- (Fed. Cir. Oct. 17, 2016)

Synopsys, Inc. v. Mentor Graphics Corp.

Synopsys, Inc. appeals the District Court for the Northern District of California's grant of summary judgment invalidating certain claims of U.S. Patent Nos. 5,530,841, 5,680,318, and 5,748,488 under 35 U.S.C. § 101.

Background

In the early days of logic circuits, a designer was required to specify his design in great detail. He would do so in the form of a schematic diagram that identified individual hardware components and the interconnections between them or via a set of Boolean logic equations that specified the precise functionality of the design. A fabrication facility would then build the corresponding physical circuit based on the architecture presented in the detailed design.

Over time, logic circuits became more and more complex. As complexity increased, many designers began to focus on the higher-level functionality of their designs. [New computer languages known as "hardware description languages"] allowed designers to describe only the desired operation of the logic circuit rather than having to specify the actual individual components and interconnections of the logic circuit.

The introduction of HDLs necessitated the development of computerized design tools that could translate the functional description of the logic circuit into a detailed design for fabrication. Early computerized design tools, however, could only recognize and translate simple circuit elements.

The Gregory Patents describe constructs known as "control flow graphs," and "assignment conditions," that provide a scheme to translate HDL-based functional descriptions of logic circuits into hardware component descriptions of those same circuits without requiring the designer to instantiate any individual hardware components.

Representative claim 1 and the associated portion of the specification of U.S. Pat. No. 5,530,841 detail the method of using assignment conditions to translate from a functional description of a level sensitive latch into a hardware component description of that same latch. Claim 1 reads:

> A method for converting a hardware independent user description of a logic circuit, that includes flow control statements including an IF statement and a GOTO statement, and directive statements that define levels of logic signals, into logic circuit hardware components comprising:

converting the flow control statements and directive statements in the user description for a logic signal Q into an assignment condition AL(Q) for an asynchronous load function AL( ) and an assignment condition AD(Q) for an asynchronous data function AD( ); and

generating a level sensitive latch when both said assignment condition AL(Q) and said assignment condition AD(Q) are nonconstant;

wherein said assignment condition AD(Q) is a signal on a data input line of said flow through latch;

said assignment condition AL(Q) is a signal on a latch gate line of said flow through latch; and

an output signal of said flow through latch is said logic signal Q.

A level sensitive latch is a basic form of memory. It is a hardware component that stores a binary input (i.e., the value ”1” or ”0”), but only when a specified condition is true. A level sensitive latch can be described functionally using HDL code as follows:

TABLE 8

```
If (COND)
   Q: = D
else
endif
```

Here, ”D” represents the input to the latch and ”Q” the output. The relationship between input D and output Q is dictated by the ”flow control statement” defined by the line of code ”If(COND).” In this example, when condition ”COND” is true (i.e., has the value ”1”), the code flows to the immediately following line of code – i.e., ”Q: = D” – and output Q is assigned the value of input D. In contrast, when condition COND is false (i.e., has the value ”0”), the code skips the directive statement ”Q: = D” and flows directly to the line of code ”else.” In this example, no instructions follow ”else.” The value of output Q therefore remains unchanged. In sum, when condition COND is true, output Q is assigned the value of input D; when condition COND is false, output Q retains its prior value regardless of whether the value of input D remains the same or changes.

The claimed method takes the functional description of the latch as an input. It then converts the functional description into an equivalent description in the form of (1) an asynchronous load function; and (2) an asynchronous data function. Here, the asynchronous load function for output Q is COND because output Q is assigned a new value (i.e., it is ”loaded”) whenever condition COND is true. The asynchronous data function for Q is ”COND*D” because output Q

is assigned the value "1" if, and only if, both condition COND and input D are true.

The assignment conditions associated with the functional description of the latch are summarized in the table below:

TABLE 9

Assignment Conditions

Variable	AL()	AD()	SL()	SD()	DC()	Z()
Q	COND	COND*D	0	0	0	0

Claim 1 specifies that where, as here, the asynchronous load function and the asynchronous data function are non-constant, the claimed method generates a level sensitive latch. A hardware component description of the level sensitive latch is shown below:

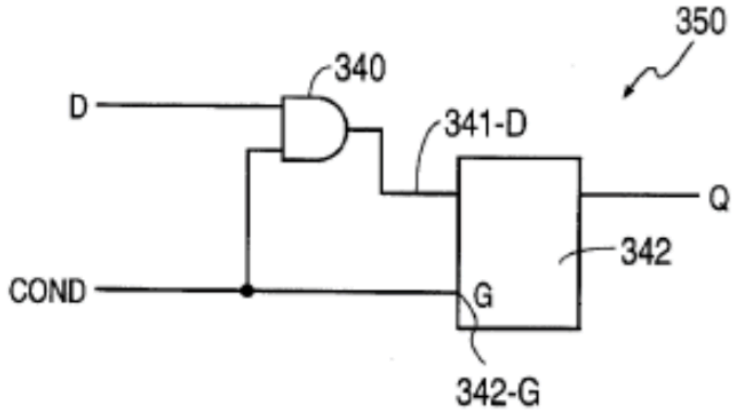


Figure 8A

Importantly, the Gregory Patents make clear that HDL code existed in the prior art. The HDL code for the level sensitive latch shown in Table 8 was already well known by the time the claimed inventions of the Gregory Patents were conceived. The same is true of the circuit diagram for a level sensitive latch shown in Figure 8A; circuit diagrams like this existed long before the Gregory Patents. What Gregory instead claims to have invented is a process for interpreting the HDL code in Table 8 that uses the assignment conditions of Table 9 to identify the circuit diagram of Figure 8A as the hardware that performs the function recited in the HDL code. At bottom, the information provided in Table 8 (code), Table 9 (assignment conditions), and Figure 8A (circuit diagram) are all equivalent representations of the same thing: a level sensitive latch.

I. Alice Step 1: Are the Asserted Claims directed to an abstract idea?

CyberSource: 654 F.3d 1366 (Fed. Cir. 2011)

The district court based its Alice Step 1 analysis on a basic premise: "the claims are directed to a mental process." We held in *CyberSource Corp. v. Retail Decisions, Inc.* that mental processes are "a subcategory of unpatentable abstract ideas." As we explained:

> Methods which can be performed entirely in the human mind are unpatentable not because there is anything wrong with claiming mental method steps as part of a process containing non-mental steps, but rather because computational methods which can be performed entirely in the human mind are the types of methods that embody the basic tools of scientific and technological work that are free to all men and reserved exclusively to none.

While the Supreme Court has altered the § 101 analysis since *CyberSource* in cases like *Mayo* and *Alice*, we continue to treat analyzing information by steps people go through in their minds, or by mathematical algorithms, without more, as essentially mental processes within the abstract-idea category.

Although the Asserted Claims are devoid of any reference to a computer or any other physical component, Synopsys suggests that the "complexity" of the claimed methods would make it implausible – if not impossible – for a skilled logic circuit designer to perform the methods mentally or with pencil and paper.

Synopsys' argument is belied by the actual claims at issue. Claim 1 recites a method of changing one description of a level sensitive latch (i.e., a functional description) into another description of the level sensitive latch (i.e., a hardware component description) by way of a third description of that very same level sensitive latch (i.e., assignment conditions). As demonstrated above and in the patent specification itself the method can be performed mentally or with pencil and paper. The skilled artisan must simply analyze a four-line snippet of HDL code and translate this short piece of code into assignment conditions and further translate those two assignment conditions into a schematic representation of a level sensitive latch.

I.e., turn Table 8 into Table 9 and then into Figure 8A

Although an understanding of logic circuit design is certainly required to perform the steps, the limited, straightforward nature of the steps involved in the claimed method make evident that a skilled artisan could perform the steps mentally. The inventors of the Gregory Patents confirmed this point when they admitted to performing the steps mentally themselves.

Synopsys next argues that even if the Asserted Claims could be performed mentally they would, in practice, be performed on a computer. In support of this argument, counsel for Synopsys during oral argument pointed to the "200 pages of code" attached to the specifications of the Gregory Patents that he contended reveal the "true novelty" of the Asserted Claims.

While Synopsys may be correct that the inventions of the Gregory Patents were intended to be used in conjunction with computer-based design tools, the Asserted Claims are not confined to that conception.

On their face, the claims do not call for any form of computer implementation of the claimed methods. Because the Asserted Claims make no mention of employing a computer or any other physical device, they are so broad as to read on an individual performing the claimed steps mentally or with pencil and paper.

For this reason, we need not decide whether a computer-implemented version of the invention would not be "directed to" an abstract idea. And, for the same reasons, Synopsys cannot rely on our decisions in *Enfish, LLC v. Microsoft Corp.* and *McRO* to support the patentability of the Asserted Claims. In *Enfish*, we held that claims directed to a specific improvement to the way computers operate to store and retrieve data were not unpatentably abstract. In *McRO*, we similarly held that claims that recited a specific asserted improvement in computer animation were not directed to an unpatentable abstract idea. By their terms and the district court's unchallenged constructions, the Asserted Claims do not involve the use of a computer in any way. The Asserted Claims thus cannot be characterized as an improvement in computer technology.

Enfish: 822 F.3d 1327 (Fed. Cir. 2016)

Synopsys' argument that "the asserted claims do not preempt all conversions" from functional descriptions of logic circuits to hardware component descriptions of logic circuits likewise misses the mark. While preemption may signal patent ineligible subject matter, the absence of complete preemption does not demonstrate patent eligibility.

We recognize that defining the precise abstract idea of patent claims in many cases is far from a straightforward exercise. But, here, the Asserted Claims are drawn to the abstract idea of: translating a functional description of a logic circuit into a hardware component description of the logic circuit. As detailed above, this translation is a mental process.

II. Alice Step 2: Do the Asserted Claims include an inventive concept?

Synopsys equates the inventive concept inquiry with novelty and contends that the Asserted Claims contain an inventive concept because they were not shown to have been anticipated by or obvious over the prior art. That position misstates the law. A claim for a new abstract idea is still an abstract idea. The search for a § 101 inventive concept is thus distinct from demonstrating § 102 novelty.

That being said, the contours of what constitutes an inventive concept are far from precise.

DDR Holdings: 773 F.3d 1245 (Fed. Cir. 2014)

In *DDR Holdings, LLC v. Hotels.com, L.P.*, we held that claims "directed to systems and methods of generating a composite web page that combines certain visual elements of a 'host' website with content of a third-party merchant" contained the requisite inventive concept.

We explained that the claims at issue involved a technological solution that overcame a specific challenge unique to the Internet.

BASCOM: 827 F.3d 1341 (Fed. Cir. 2016)

In *BASCOM Global Internet Servs., Inc. v. AT&T Mobility LLC*, we likewise held that claims "directed to filtering content on the Internet" contained an inventive concept. We recognized that "the limitations of the claims, taken individually, recite generic computer, network and Internet components, none of which is inventive by itself." We explained, however, that "an inventive concept can be found in the non-conventional and non-generic arrangement of known, conventional pieces." We found that the claims at issue contained just such an inventive arrangement through "the installation of a filtering tool at a specific location, remote from the end-users, with customizable filtering features specific to each end user." The claimed custom filter could be located remotely from the user because the invention exploited the ability of Internet service providers to associate a search request with a particular individual account. This technical solution overcame defects in prior art embodiments and elevated an otherwise abstract idea to a patentable invention.

Are *McRO* and *Synopsys* compatible?

The Asserted Claims contain no such technical solution. To the extent the Asserted Claims add anything to the abstract idea, it is the use of assignment conditions as an intermediate step in the translation process. But, given that the claims are for a mental process, assignment conditions, which merely aid in mental translation as opposed to computer efficacy, are not an inventive concept that takes the Asserted Claims beyond their abstract idea. The Asserted Claims do not introduce a technical advance or improvement. They contain nothing that amounts to significantly more than a patent upon the abstract idea itself.

3 Obviousness

Apple Inc. v. Samsung Electronics Co., Ltd.

--- F.3d ---

The jury found claim 8 of [Apple's] U.S. Patent No. 8,046,721 infringed and not invalid. The district court later denied Samsung's requested JMOL and entered judgment accordingly. On February 26, 2016, a panel of this court reversed the denial of JMOL with regard to the jury verdict of non-obviousness as to the '721 patent. We agree with the district court that the jury's fact findings supported the conclusion that claim 8 would have been obvious.

The '721 patent discloses a portable device with a touch-sensitive display that can be "unlocked via gestures" performed on the screen. The patent teaches that a "problem associated with using touch screens on portable devices is the unintentional activation or deactivation of functions due to unintentional contact with the touch screen."

"Unintentional activation or deactivation of functions due to unintentional contact with the touch screen" is commonly referred to as "pocket dialing." Greg Christie, an inventor of the '721 patent, described the problem he and his colleagues set out to solve:

> We were worried about accidental use, pocket dialing, the phone getting shut down accidentally, or since we were going to have all these features on the phone, like e-mail and messaging, we were worried that, you know, mail could be sent accidentally or deleted accidentally or the phone would answer itself simply because the touch surface—you know, if it was like, like, the touch surface against your leg in your pocket, we were worried that just, like, you know, jostling around, moving around would trigger things on the screen.

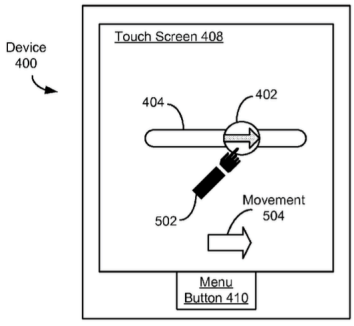
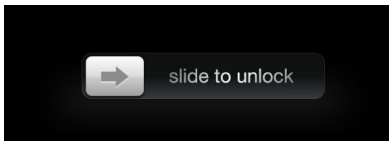


Figure 5B from the '721 patent



iPhone lock screen

The '721 patent also describes the importance of making phone activation as "user-friendly" and "efficient" as possible. It teaches:

> Accordingly, there is a need for more efficient, user-friendly procedures for unlocking such devices, touch screens, and/or applications. More generally, there is a need for more efficient, user-friendly procedures for transitioning such devices, touch screens, and/or applications between user interface states (e.g., from a user interface state for a first application to a user interface state for a second application, between user interface states in the same application, or between locked and unlocked states). In addition, there is a need for sensory feedback to the user regarding progress towards satisfaction of a user input condition that is required for the transition to occur.

Mr. Christie testified that the ease of the user interface was a central design consideration when developing the slide to unlock feature:

> We thought to introduce some sort of definite gesture. We knew we wanted to have some instruction. We knew we wanted the interface to be obvious to the customer. It would be possibly the first experience even in a retail environment. They're deciding whether they want to buy it. They pick up this iPhone, you know, it would be very bad if they looked at the phone that they had heard so much about and they look at it and say "I can't figure out how to use this. I don't know how to unlock it. It's locked." At the same time, we knew that people would be unlocking their phone, you know, tens or hundreds of times a day, so we didn't want the instruction to be, you know, insult-

> ing or talk down to the customer. We didn't want it to be cumbersome, something that they would grow tired of after a while.

Apple's expert, Dr. Cockburn, explained that there was a tension between preventing pocket dialing and ease of use: "It has to work. It has to succeed in preventing accidental activation by mistake. But yet it needs to be something that's easy to do, but not so easy that it can occur by accident, and it succeeds in that."

Apple asserted claim 8, which depends from claim 7, against several Samsung devices. These claims recite:

7. A portable electronic device, comprising:

   a touch-sensitive display;

   memory;

   one or more processors; and

   one or more modules stored in the memory and configured for execution by the one or more processors, the one or more modules including instructions:

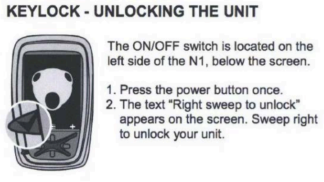
   to detect a contact with the touch-sensitive display at a first predefined location corresponding to an unlock image;

   to continuously move the unlock image on the touch-sensitive display in accordance with the movement of the detected contact while continuous contact with the touch-sensitive display is maintained, wherein the unlock image is a graphical, interactive user-interface object with which a user interacts in order to unlock the device; and

   to unlock the hand-held electronic device if the unlock image is moved from the first predefined location on the touch screen to a predefined unlock region on the touch-sensitive display.

8. The device of claim 7, further comprising instructions to display visual cues to communicate a direction of movement of the unlock image required to unlock the device.

Samsung argues claim 8 would have been obvious in light of the combination of Neonode and Plaisant. "Neonode" refers to the Neonode N1 Quickstart Guide. Neonode discloses a mobile device with a touch-sensitive screen. It explains that a user may unlock the device by pressing the power button. After the user presses the power button, text appears instructing the user to "Right sweep to unlock." Sweeping right then unlocks the unit.



Neonode N1 Quickstart Guide illustration

"Plaisant" refers to a video and corresponding two-page paper

published in 1992 titled "Touchscreen Toggle Design" by Catherine Plaisant and Daniel Wallace. The authors of the paper conducted an experiment to determine which controls ("toggles") users prefer on wall-mounted controllers for "entertainment, security, and climate control systems." These controllers were intended to be installed "flushmounted into the wall or the cabinetry." The authors presented six alternative unlocking mechanisms to a group of fifteen undergraduate students, including a "slider toggle" where a user could activate the controller by "grab[bing] the pointer and slid[ing] it to the other side." The students preferred "toggles that are pushed" over "toggles that slide," and generally ranked the slider fifth of the six alternatives. The paper also notes that sliders "were not preferred," "sliding is a more complex task than simply touching," and that "sliders are more difficult to implement than buttons."

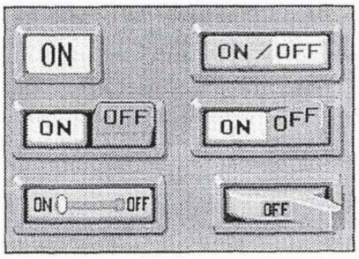


Illustration from Plaisant paper

On appeal, Apple does not contest that, together, Neonode and Plaisant disclose all the elements of claim 8. Rather, the parties dispute whether a person of ordinary skill in the art would have been motivated to combine one of the unlocking mechanisms disclosed in Plaisant with Neonode. Samsung argues "there was no evidence of any kind suggesting that Plaisant's application to a wall-mounted device would lead inventors not to combine Plaisant with Neonode." Its expert, Dr. Greenberg, testified that a person of ordinary skill "would be highly interested" in both references because "they both deal with touch base systems, they both deal with user interfaces." Dr. Greenberg testified that "a person looking at this would just think it natural to combine these two, as well taking the ideas in Plaisant, the slider, and putting them on the Neonode is, is just a very routine thing to think about in terms of interaction design." Samsung points to the Plaisant reference which states that sliding movement "is less likely to be done inadvertently."

Apple counters that a skilled artisan designing a mobile phone would not have been motivated to turn to a wall-mounted air conditioning controller to solve the pocket dialing problem. Its expert, Dr. Cockburn, testified that a person of ordinary skill would not have been naturally motivated to combine Neonode and Dr. Cockburn testified that the way the Plaisant controllers "were intended to be used was the touch screen would be mounted into a wall or into cabinetry and it would be used to control, for remote control, office or home appliances, like air conditioning units or heaters." He also explained to the jury that Plaisant itself discloses that sliding toggles were less preferred than the other switches disclosed. Apple points to Plaisant's teachings that "sliders were not preferred," "sliding is a more complex task," and "sliders are more difficult to implement." Apple argues there was substantial evidence for the jury to conclude that there would not have been a motivation to combine Plaisant and

Neonode to arrive at the claimed invention.

The district court determined that a reasonable jury could have found that a person of ordinary skill would not have been motivated to combine Plaisant and Neonode:

> A reasonable jury could infer from [Dr. Cockburn's] testimony that an ordinary artisan would not have been motivated to combine elements from a wall-mounted touchscreen for home appliances and a smartphone, particularly in view of the "pocket dialing" problem specific to mobile devices that Apple's invention sought to address.

Additionally, Dr. Cockburn explained that Plaisant "teach[es] away from the use of sliding," because it "tells you not to use the sliding [toggle] mechanism." We agree with the district court that on this record, the jury's implicit fact findings that Plaisant would not have provided a skilled artisan with a motivation to combine its slider toggle switch with Neonode is supported by substantial evidence. In addition to the statements in Plaisant, the court explained:

> Dr. Cockburn testified, contrary to Dr. Greenberg, that a person of ordinary skill in the art would not have been motivated to combine the Neonode and Plaisant in such a way as to invent claim 8. He provided two reasons. First, Plaisant described "toggle designs" intended to be used with a "touch screen [that] would be mounted into a wall or into cabinetry" for controlling "office or home appliances, like air conditioning units or heaters." A reasonable jury could infer from this testimony that an ordinary artisan would not have been motivated to combine elements from a wall-mounted touchscreen for home appliances and a smartphone, particularly in view of the "pocket dialing" problem specific to mobile devices that Apple's invention sought to address.

We agree with the district court's analysis. Because the jury found the issue of validity in favor of Apple, we presume it resolved the conflicting expert testimony and found that a skilled artisan would not have been motivated to combine the slider toggle in Plaisant with the cell phone disclosed in Neonode. The question for our review is whether substantial evidence supports this implied fact finding. We conclude that it does. Neonode discloses a mobile phone. Plaisant discloses a wall-mounted air conditioning controller. The jury had both references before it. Although Samsung presents arguments for combining the two references, these arguments were before the jury. We agree with the district court: "A reasonable jury could infer from this testimony that an ordinary artisan would not have been motivated to

combine elements from a wall-mounted touchscreen for home appliances and a smartphone, particularly in view of the 'pocket dialing' problem specific to mobile devices that Apple's invention sought to address."

The record includes Plaisant and Neonode and all that these references teach, including Plaisant's reference to inadvertent activation, complexity, difficult implementability, and that users do not prefer sliders. Though the prior art references each relate to touchscreens, the totality of the evidence supports the conclusion that it would not have been obvious for a skilled artisan, seeking an unlock mechanism that would be both intuitive to use and solve the pocket dialing problem for cell phones, to look to a wall-mounted controller for an air conditioner. The two-page Plaisant paper published in 1992 reported the results of a user-preference survey of fifteen undergraduates on six different computer-based switches. That a skilled artisan would look to the Plaisant paper directed to a wall-mounted interface screen for appliances and then choose the slider toggle, which the study found rated fifth out of six options in usability, to fulfill a need for an intuitive unlock mechanism that solves the pocket dialing problem for cell phones seems far from obvious.

[The *en banc* court also discussed secondary considerations. Among other things, it noted:

- *Industry praise*. – When Steve Jobs unveiled the slide to unlock feature, the audience burst into cheers.
- *Copying*. – Internal Samsung documents praised the iPhone's slide to unlock feature as "a creative way of solving UI complexity" and concluded that Samsung's designs would be better if they were more like the iPhone's.
- *Commercial success*. – Apple wanted a vivid easily comprehensible demonstration of how its iPhone touch screens worked to make customers comfortable with it. Slide to unlock was the first feature shown in Apple's first iPhone commercial. These facts were evidence that the iPhone's commercial success was due in part to slide to unlock.
- *Long-felt need*. – Apple's experts testified to the deficiencies of numerous previous "frustrating" solutions to the pocket-dialing problem, and internal Samsung documents showed that it had tried four other alternatives and concluded they were all inferior to slide to unlock.]

C Copyright

Software copyright has been intensely controversial.

1 Subject Matter

Some of the most important software subject-matter holdings arose in the context of similarity analyses. In order to explain whether the defendant's allegedly infringing software, courts must first filter out the unprotectable aspects of plaintiff's software. Thus, *Whelan* and *Altai* are included here rather in a section on similarity because of their importance to subsequent caselaw.

Apple Computer, Inc. v. Franklin Computer Corp.

714 F.2d 1240 (3d Cir. 1983)

Apple, one of the computer industry leaders, manufactures and markets personal computers (microcomputers), related peripheral equipment such as disk drives (peripherals), and computer programs (software). It presently manufactures Apple II computers and distributes over 150 programs. Apple has sold over 400,000 Apple II computers, employs approximately 3,000 people, and had annual sales of $335,000,000 for fiscal year 1981.

Programs sold by Franklin in conjunction with its ACE 100 computer were virtually identical with those covered by fourteen Apple copyrights. The variations that did exist were minor, consisting merely of such things as deletion of reference to Apple or its copyright notice.[5] Franklin did not dispute that it copied the Apple programs. Its witness admitted copying each of the works in suit from the Apple programs.

Franklin's principal defense is that the Apple operating system programs are not capable of copyright protection.

A. Copyrightability of a Computer Program Expressed in Object Code

Certain statements by the district court suggest that programs expressed in object code, as distinguished from source code, may not be the proper subject of copyright. We find no basis in the statute for any such concern. The district court here questioned whether copyright was to be limited to works designed to be "read" by a human reader as distinguished from read by an expert with a microscope and patience. The suggestion that copyrightability depends on a communicative function to individuals stems from the early decision of *White-Smith*, which held a piano roll was not a copy of the musical composition because it was not in a form others, except perhaps for a very expert few, could perceive. However, it is clear from the language of the 1976 Act and its legislative history that it was intended to obliterate distinctions engendered by *White-Smith*.

Apple II

Franklin ACE 100

[5]For example, 8 bytes of memory were altered in the Autostart ROM program so that when the computer is turned on "ACE 100" appears on the screen rather than "Apple II." The Franklin DOS 3.3 program also had 16 bytes (out of 9000) that allowed use of upper and lower case.

17 U.S.C. § 102

17 U.S.C. § 101

Under the statute, copyright extends to works in any tangible means of expression "*from which they can be perceived*, reproduced, or otherwise communicated, either directly or *with the aid of a machine or device*." Further, the definition of "computer program" adopted by Congress in the 1980 amendments is "sets of statements or instructions to be used *directly or indirectly* in a computer in order to bring about a certain result." As source code instructions must be translated into object code before the computer can act upon them, only instructions expressed in object code can be used "directly" by the computer.

The district court also expressed uncertainty as to whether a computer program in object code could be classified as a "literary work." However, the category of "literary works", one of the seven copyrightable categories, is not confined to literature in the nature of Hemingway's *For Whom the Bell Tolls*. The definition of "literary works" in section 101 includes expression not only in words but also "numbers, or other ... numerical symbols or indicia", thereby expanding the common usage of "literary works." *Cf. Harcourt, Brace & World, Inc. v. Graphic Controls Corp.* (the symbols designating questions or response spaces on exam answer sheets held to be copyrightable "writings" under 1909 Act); *Reiss v. National Quotation Bureau, Inc.* (code book of coined words designed for cable use copyrightable). Thus a computer program, whether in object code or source code, is a "literary work" and is protected from unauthorized copying, whether from its object or source code version.

Harcourt, Brace: 329 F. Supp. 517 (S.D.N.Y. 1971)

Reiss: 276 F. 717 (S.D.N.Y. 1921)

C. Copyrightability of Computer Operating System Programs

We turn to the heart of Franklin's position on appeal which is that computer operating system programs, as distinguished from application programs, are not the proper subject of copyright. Franklin argues that an operating system program is either a "process", "system", or "method of operation" and hence uncopyrightable.

Franklin's attack on operating system programs as "methods" or "processes" seems inconsistent with its concession that application programs are an appropriate subject of copyright. Both types of programs instruct the computer to do something. Therefore, it should make no difference whether these instructions tell the computer to help prepare an income tax return (the task of an application program) or to translate a high level language program from source code into its binary language object code form (the task of an operating system program such as "Applesoft"). Since it is only the instructions which are protected, a "process" is no more involved because the instructions in an operating system program may be used to activate the operation of the computer than it would be if instructions were written in ordinary English in a manual which described the

Wait! Is Applesoft an operating system or an application?

necessary steps to activate an intricate complicated machine. There is, therefore, no reason to afford any less copyright protection to the instructions in an operating system program than to the instructions in an application program.

Franklin's argument that an operating system program is part of a machine mistakenly focuses on the physical characteristics of the instructions. But the medium is not the message. We have already considered and rejected aspects of this contention in the discussion of object code and ROM. The mere fact that the operating system program may be etched on a ROM does not make the program either a machine, part of a machine or its equivalent. Furthermore, as one of Franklin's witnesses testified, an operating system does not have to be permanently in the machine in ROM, but it may be on some other medium, such as a diskette or magnetic tape, where it could be readily transferred into the temporary memory space of the computer. In fact, some of the operating systems at issue were on diskette.

Does this reasoning make the SCPA pointless? Should conductor mask designs be registered as literary works because they are equivalent to computer programs? For that matter, what does this reasoning suggest about the Federal Circuit's "new machine" justification for software patents?

Adobe Systems Inc. v. Southern Software Inc.

45 U.S.P.Q.2d 1827 (ND Cal Feb. 2, 1998)

This description of digital typefaces is taken from Blake Fry, *Why Typefaces Proliferate Without Copyright Protection*, 8 J. Telecomm. & High Tech. L. 425 (2010)

[Computer outline fonts are a set of points, selected by the font's designer, describing the outside of a letter. The advantage of outlined computer fonts is that since only the outline of the letter is described, a character can be enlarged or shrunk by simply increasing or decreasing the distance between the points. For displaying or printing, software connects these lines, and shades in the letter.]

Computer programs are protectable literary works. Typeface designs are not copyrightable. *Eltra Corp. v. Ringer*. A computer program is not rendered unprotectable merely because its output is not protectable. Thus, the fact that a computer program produces unprotectable typefaces does not make the computer program itself unprotectable.

Eltra: 579 F.2d 294 (4th Cir. 1978)

In a 1988 Copyright Office Policy Decision, the Copyright Office determined that digitized typefaces were not copyrightable because they were not computer programs and required little selection or arrangement beyond that dictated by the uncopyrightable typeface design. However, in 1992 the Copyright Office issued a final regulation regarding the registrability of "computer programs that generate typefaces" which appears to back off the 1988 policy decision. The 1992 Regulation states:

> The Copyright Office is persuaded that creating scalable typefonts using already digitized typeface represents a significant change in the industry since our previous Policy Decision. We are also persuaded that computer programs designed for generating typeface in conjunction with printing devices may involve original computer in-

> structions entitled to protection under the Copyright Act. For example, the creation of scalable font output programs to produce harmonious fonts consisting of hundreds of characters typically involves many decisions in drafting the instructions that drive the printer. The expression of these decisions is neither limited by the unprotectible shape of the letters nor functionally mandated. This expression, assuming it meets the usual standard of authorship, is thus registrable as a computer program.

Defendants state that "merely manipulating an unprotectable font image to create another, slightly different (but still unprotectable) font image cannot possibly give rise to protectable expression" Defendants assert that no matter what points are selected by the Adobe editor performing the process, they correspond directly to, and hence are determined by, the unprotectable font shape. Therefore, because the output is not protected and there cannot be any creativity in what the editor does to obtain the output, nothing is protectable.

The evidence presented shows that there is some creativity in designing the font software programs. While the glyph dictates to a certain extent what points the editor must choose, it does not dictate every point that must be chosen. Adobe has shown that font editors make creative choices as to what points to select based on the image in front of them on the computer screen. The code is determined directly from the selection of the points. That some creativity is involved is illustrated by the fact that two independently working programmers using the same data and same tools can produce an indistinguishable output but will have few points in common. Accordingly, the court finds that the Adobe font software programs are protectable original works of authorship. Thus, any copying of the points is copying of literal expression, that is, in essence, copying of the computer code itself.

Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc.

797 F.2d 1222 (3d Cir. 1986)

How are DENTALAB and DENTCOM as trademarks?

[Jaslow, a dental laboratory, hired Whelan, a software development firm, to develop dental laboratory bookkeeping software called Dentalab. Whelan kept the copyright; Jaslow marketed the software to other dental firms and received a 35% commission. One of Jaslow's owner/officers allegedly used his access to the Dentalab software to develop a competing dental-firm bookkeeping program, Dentcom, which was written in a different programming language: BASIC rather than EDL.]

The *Arnstein* ordinary observer test, which was developed in cases involving novels, plays, and paintings, is of doubtful value in cases involving computer programs on account of the programs' complexity

and unfamiliarity to most members of the public. We therefore join the growing number of courts which do not apply the ordinary observer test in copyright cases involving exceptionally difficult materials, like computer programs, but instead adopt a single substantial similarity inquiry according to which both lay and expert testimony would be admissible.

It is well, though recently, established that copyright protection extends to a program's source and object codes. In this case, however, the district court did not find any copying of the source or object codes, nor did the plaintiff allege such copying. Rather, the district court held that the Dentalab copyright was infringed because the overall structure of Dentcom was substantially similar to the overall structure of Dentalab. The question therefore arises whether mere similarity in the overall structure of programs can be the basis for a copyright infringement, or, put differently, whether a program's copyright protection covers the structure of the program or only the program's literal elements, i.e., its source and object codes.

The copyrights of other literary works can be infringed even when there is no substantial similarity between the works' literal elements. One can violate the copyright of a play or book by copying its plot or plot devices. *See, e.g., Twentieth Century-Fox Film Corp. v. MCA, Inc.* (13 alleged distinctive plot similarities between *Battlestar Galactica* and *Star Wars* may be basis for a finding of copyright violation); *Sid & Marty Krofft Television v. McDonald's Corp.* (similarities between McDonaldland characters and H.R. Pufnstuf characters can be established by "total concept and feel" of the two productions. By analogy to other literary works, it would thus appear that the copyrights of computer programs can be infringed even absent copying of the literal elements of the program. Defendants contend, however, that what is true of other literary works is not true of computer programs. They assert two principal reasons, which we consider in turn.

Fox v. MCA: 715 F.2d 1327, 1329 (9th Cir. 1983)

Krofft: 562 F.2d 1157 (9th Cir. 1977)

Defendants argue that the structure of a computer program is, by definition, the idea and not the expression of the idea, and therefore that the structure cannot be protected by the program copyright.

Just as *Baker* focused on the end sought to be achieved by Selden's book, the line between idea and expression may be drawn with reference to the end sought to be achieved by the work in question. In other words, the purpose or function of a utilitarian work would be the work's idea, and everything that is not necessary to that purpose or function would be part of the expression of the idea. Where there are various means of achieving the desired purpose, then the particular means chosen is not necessary to the purpose; hence, there is expression, not idea.

It is clear that the purpose of the utilitarian Dentalab program was to aid in the business operations of a dental laboratory. It is equally

clear that the structure of the program was not essential to that task: there are other programs on the market, competitors of Dentalab and Dentcom, that perform the same functions but have different structures and designs. The conclusion is thus inescapable that the detailed structure of the Dentalab program is part of the expression, not the idea, of that program.

Computer Associates Intern., Inc. v. Altai, Inc.

982 F. 2d 693 (1992)

The essentially utilitarian nature of a computer program complicates the task of distilling its idea from its expression. In order to describe both computational processes and abstract ideas, its content combines creative and technical expression. The variations of expression found in purely creative compositions, as opposed to those contained in utilitarian works, are not directed towards practical application. For example, a narration of Humpty Dumpty's demise, which would clearly be a creative composition, does not serve the same ends as, say, a recipe for scrambled eggs – which is a more process oriented text. Thus, compared to aesthetic works, computer programs hover even more closely to the elusive boundary line described in § 102(b).

To the extent that an accounting text and a computer program are both a set of statements or instructions to bring about a certain result, they are roughly analogous. In the former case, the processes are ultimately conducted by human agency; in the latter, by electronic means. In either case, as already stated, the processes themselves are not protectable. But the holding in *Baker* goes farther. The Court concluded that those aspects of a work, which "must necessarily be used as incident to" the idea, system or process that the work describes, are also not copyrightable. Selden's ledger sheets, therefore, enjoyed no copyright protection because they were "necessary incidents to" the system of accounting that he described. From this reasoning, we conclude that those elements of a computer program that are necessarily incidental to its function are similarly unprotectable.

We think that *Whelan*'s approach to separating idea from expression in computer programs relies too heavily on metaphysical distinctions and does not place enough emphasis on practical considerations.

We think that district courts would be well-advised to undertake a three-step procedure in order to determine whether the non-literal elements of two or more computer programs are substantially similar. This approach breaks no new ground; rather, it draws on such familiar copyright doctrines as merger, scenes a faire, and public domain.

In ascertaining substantial similarity under this approach, a court would first break down the allegedly infringed program into its constituent structural parts. Then, by examining each of these parts for

such things as incorporated ideas, expression that is necessarily incidental to those ideas, and elements that are taken from the public domain, a court would then be able to sift out all non-protectable material. Left with a kernel, or possible kernels, of creative expression after following this process of elimination, the court's last step would be to compare this material with the structure of an allegedly infringing program. The result of this comparison will determine whether the protectable elements of the programs at issue are substantially similar so as to warrant a finding of infringement.

Step One: Abstraction

As applied to computer programs, the abstractions test will comprise the first step in the examination for substantial similarity. Initially, in a manner that resembles reverse engineering on a theoretical plane, a court should dissect the allegedly copied program's structure and isolate each level of abstraction contained within it. This process begins with the code and ends with an articulation of the program's ultimate function. As an anatomical guide to this procedure, the following description is helpful:

> At the lowest level of abstraction, a computer program may be thought of in its entirety as a set of individual instructions organized into a hierarchy of modules. At a higher level of abstraction, the instructions in the lowest-level modules may be replaced conceptually by the functions of those modules. At progressively higher levels of abstraction, the functions of higher-level modules conceptually replace the implementations of those modules in terms of lower-level modules and instructions, until finally, one is left with nothing but the ultimate function of the program. A program has structure at every level of abstraction at which it is viewed. At low levels of abstraction, a program's structure may be quite complex; at the highest level it is trivial.

Steven R. Englund, Note, *Idea, Process, or Protected Expression?: Determining the Scope of Copyright Protection of the Structure of Computer Programs*, 88 Mich. L. Rev. 866 (1990)

Step Two: Filtration

Once the program's abstraction levels have been discovered, the substantial similarity inquiry moves from the conceptual to the concrete. We endorse, a successive filtering method for separating protectable expression from non-protectable material. This process entails examining the structural components at each level of abstraction to determine whether their particular inclusion at that level was "idea" or was dictated by considerations of efficiency, so as to be necessarily incidental to that idea; required by factors external to the program itself; or taken from the public domain and hence is nonprotectable

expression. The structure of any given program may reflect some, all, or none of these considerations.

Strictly speaking, this filtration serves the purpose of defining the scope of plaintiff's copyright. By applying well developed doctrines of copyright law, it may ultimately leave behind a core of protectable material. Further explication of this second step may be helpful.

(a) Elements Dictated by Efficiency

In the context of computer program design, the concept of efficiency is akin to deriving the most concise logical proof or formulating the most succinct mathematical computation. Thus, the more efficient a set of modules are, the more closely they approximate the idea or process embodied in that particular aspect of the program's structure.

While, hypothetically, there might be a myriad of ways in which a programmer may effectuate certain functions within a program, – i.e., express the idea embodied in a given subroutine – efficiency concerns may so narrow the practical range of choice as to make only one or two forms of expression workable options. Of course, not all program structure is informed by efficiency concerns. It follows that in order to determine whether the merger doctrine precludes copyright protection to an aspect of a program's structure that is so oriented, a court must inquire whether the use of this particular set of modules is necessary efficiently to implement that part of the program's process being implemented. If the answer is yes, then the expression represented by the programmer's choice of a specific module or group of modules has merged with their underlying idea and is unprotected.

Efficiency is an industry-wide goal. Since, as we have already noted, there may be only a limited number of efficient implementations for any given program task, it is quite possible that multiple programmers, working independently, will design the identical method employed in the allegedly infringed work. Of course, if this is the case, there is no copyright infringement.

(b) Elements Dictated By External Factors

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.

Q-Co: 625 F. Supp. 608 (S.D.N.Y. 1985)

The district court in *Q-Co Industries, Inc. v. Hoffman* rested its holding on what, perhaps, most closely approximates a traditional scenes a faire rationale. There, the court denied copyright protection to four program modules employed in a teleprompter program. This decision was ultimately based upon the court's finding that "the same modules would be an inherent part of any prompting program."

(c) Elements taken From the Public Domain

Material found in the public domain is free for the taking and cannot be appropriated by a single author even though it is included in a copyrighted work. We see no reason to make an exception to this rule for elements of a computer program that have entered the public domain by virtue of freely accessible program exchanges and the like. Thus, a court must also filter out this material from the allegedly infringed program before it makes the final inquiry in its substantial similarity analysis.

Step Three: Comparison

The third and final step of the test for substantial similarity that we believe appropriate for non-literal program components entails a comparison. Once a court has sifted out all elements of the allegedly infringed program which are "ideas" or are dictated by efficiency or external factors, or taken from the public domain, there may remain a core of protectable expression. In terms of a work's copyright value, this is the golden nugget. At this point, the court's substantial similarity inquiry focuses on whether the defendant copied any aspect of this protected expression, as well as an assessment of the copied portion's relative importance with respect to the plaintiff's overall program.

Evidentiary Analysis

The district court took the first step in the analysis set forth in this opinion when it separated the program by levels of abstraction. The district court stated:

> As applied to computer software programs, this abstractions test would progress in order of "increasing generality" from object code, to source code, to parameter lists, to services required, to general outline. In discussing the particular similarities, therefore, we shall focus on these levels.

Moving to the district court's evaluation of OSCAR 3.5's structural components, we agree with Judge Pratt's systematic exclusion of nonprotectable expression. With respect to code, the district court observed that after the rewrite of OSCAR 3.4 to OSCAR 3.5, "there re-

mained virtually no lines of code that were identical to ADAPTER." Accordingly, the court found that the code "present[ed] no similarity at all."

Next, Judge Pratt addressed the issue of similarity between the two programs' parameter lists and macros. He concluded that, viewing the conflicting evidence most favorably to CA, it demonstrated that "only a few of the lists and macros were similar to protected elements in ADAPTER; the others were either in the public domain or dictated by the functional demands of the program." As discussed above, functional elements and elements taken from the public domain do not qualify for copyright protection. With respect to the few remaining parameter lists and macros, the district court could reasonably conclude that they did not warrant a finding of infringement given their relative contribution to the overall program. In any event, the district court reasonably found that, for lack of persuasive evidence, CA failed to meet its burden of proof on whether the macros and parameter lists at issue were substantially similar.

The district court also found that the overlap exhibited between the list of services required for both ADAPTER and OSCAR 3.5 was "determined by the demands of the operating system and of the applications program to which it was to be linked through ADAPTER or OSCAR" In other words, this aspect of the program's structure was dictated by the nature of other programs with which it was designed to interact and, thus, is not protected by copyright.

Finally, in his infringement analysis, Judge Pratt accorded no weight to the similarities between the two programs' organizational charts, "because the charts were so simple and obvious to anyone exposed to the operation of the programs." CA argues that the district court's action in this regard "is not consistent with copyright law" — that "obvious" expression is protected, and that the district court erroneously failed to realize this. However, to say that elements of a work are "obvious," in the manner in which the district court used the word, is to say that they "follow naturally from the work's theme rather than from the author's creativity." This is but one formulation of the scenes a faire doctrine, which we have already

Oracle America, Inc. v. Google Inc.

872 F. Supp. 2d 974 (N.D. Cal. 2012)

This action was the first of the so-called "smartphone war" cases tried to a jury. This order includes the findings of fact and conclusions of law on a central question tried simultaneously to the judge, namely the extent to which, if at all, certain replicated elements of the structure, sequence and organization of the Java application programming interface are protected by copyright.

The facts here are drawn primarily from the Court of Appeals' opinion, which immediately follows.

Sun Microsystems, Inc. ("Sun") developed the Java "platform" for

computer programming and released it in 1996.[1] The aim was to relieve programmers from the burden of writing different versions of their computer programs for different operating systems or devices. The Java platform, through the use of a virtual machine, enabled software developers to write programs that were able to run on different types of computer hardware without having to rewrite them for each different type. With Java, a software programmer could "write once, run anywhere."

The Java virtual machine ("JVM") plays a central role in the overall Java platform. The Java programming language itself – which includes words, symbols, and other units, together with syntax rules for using them to create instructions – is the language in which a Java programmer writes source code, the version of a program that is in a human-readable language. For the instructions to be executed, they must be converted (or compiled) into binary machine code (object code) consisting of 0s and Is understandable by the particular computing device. In the Java system, source code is first converted into "bytecode," an intermediate form, before it is then converted into binary machine code by the Java virtual machine that has been designed for that device.

Sun wrote a number of ready-to-use Java programs to perform common computer functions and organized those programs into groups it called "packages." These packages, which are the application programming interfaces at issue in this appeal, allow programmers to use the prewritten code to build certain functions into their own programs, rather than write their own code to perform those functions from scratch. They are shortcuts. Sun called the code for a specific operation (function) a "method." It defined "classes" so that each class consists of specified methods plus variables and other elements on which the methods operate. To organize the classes for users, then, it grouped classes (along with certain related "interfaces") into "packages." The parties have not disputed the district court's analogy: Oracle's collection of API packages is like a library, each package is like a bookshelf in the library, each class is like a book on the shelf, and each method is like a how-to chapter in a book.

The original Java Standard Edition Platform ("Java SE") included eight packages of pre-written programs. The district court found, and Oracle concedes to some extent, that three of those packages – java.lang.java.io, and java.util – were "core" packages, meaning that programmers using the Java language had to use them in order to make any worthwhile use of the language. By 2008, the Java platform had more than 6,000 methods making up more than 600 classes grouped into 166 API packages. There are 37 Java API packages at

[1] Oracle acquired Sun in 2010.

issue in this appeal, three of which are the core packages identified by the district court. These packages contain thousands of individual elements, including classes, subclasses, methods, and interfaces.

Every package consists of two types of source code — what the parties call (1) declaring code; and (2) implementing code. Declaring code is the expression that identifies the prewritten function and is sometimes referred to as the "declaration" or "header." As the district court explained, the "main point is that this header line of code introduces the method body and specifies very precisely the inputs, name and other functionality." The expressions used by the programmer from the declaring code command the computer to execute the associated implementing code, which gives the computer the step-by-step instructions for carrying out the declared function.

For example, one of the Java API packages at issue is "java. lang." Within that package is a class called "math," and within "math" there are several methods, including one that is designed to find the larger of two numbers: "max." To invoke this method from another program (or class), the following call could be included in the program:

```
int a = java.lang.Math.max (2, 3);
```

Upon reaching this statement, the computer would go and find the max method under the Math class in the java.lang package, input '2' and '3' as arguments, and then return a '3,' which would then be set as the value of 'a.' The declaration for the "max" method, as defined for integers, is

```
public static int max (int x, int y) {
```

The word "public" means that other programs can call on it. (If this instead says "private," then it can only be accessed by other methods inside the same class.) The word "static" means that the method can be invoked without creating an instance of the class. (If this instead is an instance method, then it would always be invoked with respect to an object.) The word "int" means that an integer is returned by the method. (Other alternatives are "boolean," "char," and "String" which respectively mean "true/false," "single character," and "character string.") Each of these three parameters is drawn from a short menu of possibilities, each possibility corresponding to a very specific functionality. The word "max" is a name and any name (other than a reserved word [i.e, a few names like "int" and "public" that have specific and rigidly defined roles in the Java language]) could have been used.. The phrase "(int x, int y)" identifies the arguments that must be passed into the method, stating that they will be in integer form. The "x" and the "y" could be "a" and "b" or "arg1" and "arg2," so there is a degree of creativity in naming the arguments.

Finally, "{" is the beginning marker that tells the compiler that the method body is about to follow. The marker is mandatory.

Although Oracle owns the copyright on Java SE and the API packages, it offers three different licenses to those who want to make use of them. The first is the General Public License, which is free of charge and provides that the licensee can use the packages – both the declaring and implementing code – but must "contribute back" its innovations to the public. This arrangement is referred to as an "open source" license. The second option is the Specification License, which provides that the licensee can use the declaring code and organization of Oracle's API packages but must write its own implementing code. The third option is the Commercial License, which is for businesses that want to use and customize the full Java code in their commercial products and keep their code secret. Oracle offers the Commercial License in exchange for royalties. To maintain Java's "write once, run anywhere" motto, the Specification and Commercial Licenses require that the licensees' programs pass certain tests to ensure compatibility with the Java platform.

The accused product is Android, a software platform that was designed for mobile devices and competes with Java in that market. Google acquired Android, Inc. in 2005 as part of a plan to develop a smartphone platform. Later that same year, Google and Sun began discussing the possibility of Google taking a license to use and to adapt the entire Java platform for mobile devices. They also discussed a possible co-development partnership deal with Sun under which Java technology would become an open-source part of the Android platform, adapted for mobile devices. The parties negotiated for months but were unable to reach an agreement. The point of contention between the parties was Google's refusal to make the implementation of its programs compatible with the Java virtual machine or interoperable with other Java programs. Because Sun/Oracle found that position to be anathema to the "write once, run anywhere" philosophy, it did not grant Google a license to use the Java API packages.

When the parties' negotiations reached an impasse, Google decided to use the Java programming language to design its own virtual machine – the Dalvik virtual machine ("Dalvik VM") – and "to write its own implementations for the functions in the Java API that were key to mobile devices." Google developed the Android platform, which grew to include 168 API packages – 37 of which correspond to the Java API packages at issue in this appeal.

With respect to the 37 packages at issue, "Google believed Java application programmers would want to find the same 37 sets of functionalities in the new Android system callable by the same names as used in Java." To achieve this result, Google copied the declaring

source code from the 37 Java API packages verbatim, inserting that code into parts of its Android software. In doing so, Google copied the elaborately organized taxonomy of all the names of methods, classes, interfaces, and packages – the "overall system of organized names – covering 37 packages, with over six hundred classes, with over six thousand methods. The parties and district court referred to this taxonomy of expressions as the "structure, sequence, and organization" or "SSO" of the 37 packages. It is undisputed, however, that Google wrote its own implementing code [with some exceptions not here relevant].

Google released the Android platform in 2007, and the first Android phones went on sale the following year. Oracle indicated at oral argument that Android phones contain copies of the accused portions of the Android software. Android smartphones rapidly grew in popularity and now comprise a large share of the United States market. Google provides the Android platform free of charge to smartphone manufacturers and receives revenue when customers use particular functions on the Android phone. Although Android uses the Java programming language, it is undisputed that Android is not generally Java compatible. As Oracle explains, "Google ultimately designed Android to be *incompatible* with the Java platform, so that apps written for one will not work on the other."

Application of Controlling Law to Controlling Facts

Wait. *Why* is everyone free to program in Java? *Is* everyone free to program in Java?

All agree that everyone was and remains free to program in the Java language itself. All agree that Google was free to use the Java language to write its own API. While Google took care to provide fresh line-by-line implementations (the 97 percent), it generally replicated the overall name organization and functionality of 37 packages in the Java API (the three percent). The main issue addressed herein is whether this violated the Copyright Act and more fundamentally whether the replicated elements were copyrightable in the first place.

This leads to the first holding central to this order and it concerns the method level. As long as the specific code written to implement a method is different, anyone is free under the Copyright Act to write his or her own method to carry out exactly the same function or specification of any and all methods used in the Java API. Contrary to Oracle, copyright law does not confer ownership over any and all ways to implement a function or specification, no matter how creative the copyrighted implementation or specification may be. The Act confers ownership only over the specific way in which the author wrote out his version. Others are free to write their own implementation to accomplish the identical function, for, importantly, ideas, concepts and functions cannot be monopolized by copyright.

To return to our example, one method in the Java API carries out

the function of comparing two numbers and returning the greater. Google – and everyone else in the world – was and remains free to write its own code to carry out the identical function so long as the implementing code in the method body is different from the copyrighted implementation. This is a simple example, but even if a method resembles higher mathematics, everyone is still free to try their hand at writing a different implementation, meaning that they are free to use the same inputs to derive the same outputs so long as the implementation in between is their own.

Much of Oracle's evidence at trial went to show that the design of methods in an API was a creative endeavor. Of course, that is true. Inventing a new method to deliver a new output can be creative, even inventive, including the choices of inputs needed and outputs returned. But such inventions – at the concept and functionality level – are protectable only under the Patent Act. Under the Copyright Act, no matter how creative or imaginative a Java method specification may be, the entire world is entitled to use the same method specification (inputs, outputs, parameters) so long as the line-by-line implementations are different. To repeat the Second Circuit's phrasing, "there might be a myriad of ways in which a programmer may express the idea embodied in a given subroutine." *Altai* The method specification is the *idea*. The method implementation is the *expression*. No one may monopolize the *idea*.

To carry out any given function, the method specification as set forth in the declaration *must be identical* under the Java rules (save only for the choices of argument names). Any other declaration would carry out some *other* function. The declaration requires precision. Significantly, when there is only one way to write something, the merger doctrine bars anyone from claiming exclusive copyright ownership of that expression. Therefore, there can be no copyright violation in using the identical declarations. Nor can there be any copyright violation due to the *name* given to the method (or to the arguments), for under the law, names and short phrases cannot be copyrighted.

In sum, Google and the public were and remain free to write their own implementations to carry out exactly the same functions of all methods in question, using exactly the same method specifications and names. Therefore, at the method level – the level where the heavy lifting is done – Google has violated no copyright, it being undisputed that Google's implementations are different.

* * *

Even so, the second major copyright question is whether Google was and remains free to group its methods in the same way as in Java, that is, to organize its Android methods under the same class and

package scheme as in Java. For example, the Math classes in both systems have a method that returns a cosine, another method that returns the larger of two numbers, and yet another method that returns logarithmic values, and so on. As Oracle notes, the rules of Java did not insist that these methods be grouped together in any particular class. Google could have placed its trigonometric function (or any other function) under a class other than Math class. Oracle is entirely correct that the rules of the Java language did not require that the same grouping pattern (or even that they be grouped at all, for each method could have been placed in a stand-alone class). There was nothing in the rules of the Java language that required that Google replicate the same groupings even if Google was free to replicate the same functionality.

The main answer to this argument is that the overall scheme of file name organization is also a command structure for a system or method of operation of the application programming interface. The commands are (and must be) in the form

```
java.package.Class.method()
```

and each calls into action a pre-assigned function.

That a system or method of operation has thousands of commands arranged in a creative taxonomy does not change its character as a method of operation. Yes, it is creative. Yes, it is original. But it is nevertheless a command structure, a system or method of operation – a long hierarchy of over six thousand commands to carry out pre-assigned functions. For that reason, it cannot receive copyright protection.

* * *

Interoperability sheds further light on the character of the command structure as a system or method of operation. Surely, millions of lines of code had been written in Java before Android arrived. These programs necessarily used the java.package.Class.method() command format. These programs called on all or some of the specific 37 packages at issue and necessarily used the command structure of names at issue. Such code was owned by the developers themselves, not by Oracle. *In order for at least some of this code to run on Android, Google was required to provide the same java.package.Class.method() command system using the same names with the same "taxonomy" and with the same functional specifications.* Google replicated what was necessary to achieve a degree of interoperability – but no more, taking care, as said before, to provide its own implementations.

That interoperability is at the heart of the command structure is illustrated by Oracle's preoccupation with what it calls "fragmentation," meaning the problem of having imperfect interoperability

among platforms. When this occurs, Java-based applications may not run on the incompatible platforms. For example, Java-based code using the replicated parts of the 37 API packages will run on Android but will not if a 38th package is needed. Such imperfect interoperability leads to a "fragmentation" – a Balkanization – of platforms, a circumstance which Sun and Oracle have tried to curb via their licensing programs. In this litigation, Oracle has made much of this problem, at times almost leaving the impression that if only Google had replicated all 166 Java API packages, Oracle would not have sued. While fragmentation is a legitimate business consideration, it begs the question whether or not a license was required in the first place to replicate some or all of the command structure. (This is especially so inasmuch as Android has not carried the Java trademark, and Google has not held out Android as fully compatible.) The immediate point is this: fragmentation, imperfect interoperability, and Oracle's angst over it illustrate the character of the command structure as a functional system or method of operation.

Oracle America, Inc. v. Google Inc.

750 F.3d 1339 (Fed. Cir. 2014)

We are mindful that the application of copyright law in the computer context is often a difficult task. On this record, however, we find that the district court failed to distinguish between the threshold question of what is copyrightable – which presents a low bar – and the scope of conduct that constitutes infringing activity. The court also erred by importing fair use principles, including interoperability concerns, into its copyrightability analysis. For the reasons that follow, we conclude that the declaring code and the structure, sequence, and organization of the 37 Java API packages are entitled to copyright protection.

Why the Federal Circuit? Because the case originally included patent claims, giving the Federal Circuit appellate jurisdiction. Note that the Federal Circuit here is technically applying Ninth Circuit law.

1. Declaring Source Code

Under the merger doctrine, a court will not protect a copyrighted work from infringement if the idea contained therein can be expressed in only one way. For computer programs, this means that when specific parts of the code, even though previously copyrighted, are the only and essential means of accomplishing a given task, their later use by another will not amount to infringement.

Atari: 975 F. 2d 832 (Fed. Cir. 1992)

In *Atari Games Corp. v. Nintendo of America Inc.*, for example, Nintendo designed a program – the 10NES – to prevent its video game system from accepting unauthorized game cartridges. Nintendo "chose arbitrary programming instructions and arranged them in a unique sequence to create a purely arbitrary data stream" which "serves as the key to unlock the NES." Because Nintendo produced expert testimony "showing a multitude of different ways to generate

a data stream which unlocks the NES console," we concluded that Nintendo's specific choice of code did not merge with the process.

The evidence showed that Oracle had unlimited options as to the selection and arrangement of the 7000 lines Google copied. Using the district court's "java.lang. Math.max" example, Oracle explains that the developers could have called it any number of things, including "Math.maximum" or "Arith.larger." This was not a situation where Oracle was selecting among preordained names and phrases to create its packages. As the district court recognized, moreover, "the Android method and class names could have been different from the names of their counterparts in Java and still have worked." Because alternative expressions were available, there is no merger.

We further find that the district court erred in focusing its merger analysis on the options available to Google at the time of copying. It is well-established that copyrightability and the scope of protectable activity are to be evaluated at the time of creation, not at the time of infringement. The focus is, therefore, on the options that were available to Sun/Oracle at the time it created the API packages. Of course, once Sun/Oracle created "java.lang.Math.max," programmers who want to use that particular package have to call it by that name. But nothing prevented Google from writing its own declaring code, along with its own implementing code, to achieve the same result. In such circumstances, the chosen expression simply does not merge with the idea being expressed.[7]

"It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to Heaven, we were all going direct the other way -- in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only." Charles Dickens, *A Tale of Two Cities*

"*Mr. Burns*: This is a thousand monkeys working at a thousand typewriters. Soon, they'll have written the greatest novel known to man. Let's see ... 'It was the best of times, it was the blurst of times!' You stupid monkey!" *The Simpsons*, "Last Exit to Springfield" (episode 9F15).

The district court is correct that words and short phrases such as names, titles, and slogans are not subject to copyright protection. The court failed to recognize, however, that the relevant question for copyrightability purposes is not whether the work at issue contains short phrases – as literary works often do – but, rather, whether those phrases are creative. And, by dissecting the individual lines of declaring code at issue into short phrases, the district court further failed to recognize that an original combination of elements can be copyrightable.

By analogy, the opening of Charles Dickens' *A Tale of Two Cities* is nothing but a string of short phrases. Yet no one could contend that this portion of Dickens' work is unworthy of copyright protection because it can be broken into those shorter constituent components. The question is not whether a short phrase or series of short phrases can be extracted from the work, but whether the manner in which they are used or strung together exhibits creativity.

Although the district court apparently focused on individual lines

[7]he district court did not find merger with respect to the structure, sequence, and organization of Oracle's Java API packages. Nor could it, given the court's recognition that there were myriad ways in which the API packages could have been organized.

of code, Oracle is not seeking copyright protection for a specific short phrase or word. Instead, the portion of declaring code at issue is 7,000 lines, and Google's own "Java guru" conceded that there can be "creativity and artistry even in a single method declaration." Because Oracle exercised creativity in the selection and arrangement of the method declarations when it created the API packages and wrote the relevant declaring code, they contain protectable expression that is entitled to copyright protection.

2. The Structure, Sequence, and Organization of the API Packages

The district court found that the SSO of the Java API packages is creative and original, but nevertheless held that it is a system or method of operation and, therefore, cannot be copyrighted. In reaching this conclusion, the district court seems to have relied upon language contained in a First Circuit decision, *Lotus Development Corp. v. Borland International, Inc.*.

Lotus: 49 F.3d 807 (1st Cir. 1995), *aff'd without opinion by equally divided court*, 516 U.S. 233 (1996)

In *Lotus*, it was undisputed that the defendant copied the menu command hierarchy and interface from Lotus 1-2-3, a computer spreadsheet program that enables users to perform accounting functions electronically on a computer. The menu command hierarchy referred to a series of commands – such as "Copy," "Print," and "Quit" – which were arranged into more than 50 menus and submenus. Although the defendant did not copy any Lotus source code, it copied the menu command hierarchy into its rival program. The question before the court was whether a computer menu command hierarchy is copyrightable subject matter.

Although it accepted the district court's finding that Lotus developers made some expressive choices in selecting and arranging the command terms, the First Circuit found that the command hierarchy was not copyrightable because, among other things, it was a "method of operation" under Section 102(b). In reaching this conclusion, the court defined a "method of operation" as "the means by which a person operates something, whether it be a car, a food processor, or a computer." Because the Lotus menu command hierarchy provided "the means by which users control and operate Lotus 1-2-3," it was deemed unprotectable. For example, if users wanted to copy material, they would use the "Copy" command and the command terms would tell the computer what to do. According to the Lotus court, the "fact that Lotus developers could have designed the Lotus menu command hierarchy differently is immaterial to the question of whether it is a 'method of operation.' The court further indicated that, "[i]f specific words are essential to operating something, then they are part of a 'method of operation' and, as such, are unprotectable."

On appeal, Oracle argues that the district court's reliance on *Lotus* is misplaced because it is distinguishable on its facts and is inconsis-

tent with Ninth Circuit law. We agree. First, while the defendant in Lotus did not copy any of the underlying code, Google concedes that it copied portions of Oracle's declaring source code verbatim. Second, the *Lotus* court found that the commands at issue there (copy, print, etc.) were not creative, but it is undisputed here that the declaring code and the structure and organization of the API packages are both creative and original. Finally, while the court in *Lotus* found the commands at issue were "essential to operating" the system, it is undisputed that – other than perhaps as to the three core packages – Google did not need to copy the structure, sequence, and organization of the Java API packages to write programs in the Java language.

More importantly, however, the Ninth Circuit has not adopted the court's "method of operation" reasoning in *Lotus*, and we conclude that it is inconsistent with binding precedent. Specifically, we find that *Lotus* is inconsistent with Ninth Circuit case law recognizing that the structure, sequence, and organization of a computer program is eligible for copyright protection where it qualifies as an expression of an idea, rather than the idea itself. We find, moreover, that the hard and fast rule set down in *Lotus* and employed by the district court here – i.e., that elements which perform a function can never be copyrightable – is at odds with the Ninth Circuit's endorsement of the abstraction-filtration-comparison analysis.

Here, the district court recognized that the SSO "resembles a taxonomy," but found that "it is nevertheless a command structure, a system or method of operation — a long hierarchy of over six thousand commands to carry out pre-assigned functions."[12] In other words, the court concluded that, although the SSO is expressive, it is not copyrightable because it is also functional. The problem with the district court's approach is that computer programs are by definition functional – they are all designed to accomplish some task. If we were to accept the district court's suggestion that a computer program is uncopyrightable simply because it "carr[ies] out pre-assigned functions," no computer program is protectable. That result contradicts Congress's express intent to provide copyright protection to computer programs, as well as binding Ninth Circuit case law finding computer programs copyrightable, despite their utilitarian or functional purpose.

On appeal, Oracle does not — and concedes that it cannot — claim copyright in the idea of organizing functions of a computer program or in the "package-class-method" organizational structure in the abstract. Instead, Oracle claims copyright protection only in its particular way of naming and organizing each of the 37 Java API packages.

[12]This analogy by the district court is meaningful because taxonomies, in varying forms, have generally been deemed copyrightable. *See, e.g., Practice Management*

Oracle recognizes, for example, that it "cannot copyright the idea of programs that open an internet connection," but "it can copyright the precise strings of code used to do so, at least so long as other language is available to achieve the same function." Thus, Oracle concedes that Google and others could employ the Java language – much like anyone could employ the English language to write a paragraph without violating the copyrights of other English language writers. And, that Google may employ the "package-class-method" structure much like authors can employ the same rules of grammar chosen by other authors without fear of infringement. What Oracle contends is that, beyond that point, Google, like any author, is not permitted to employ the precise phrasing or precise structure chosen by Oracle to flesh out the substance of its packages — the details and arrangement of the prose.

3. Google's Interoperability Arguments are Irrelevant to Copyrightability

Oracle also argues that the district court erred in invoking interoperability in its copyrightability analysis. Specifically, Oracle argues that Google's interoperability arguments are only relevant, if at all, to fair use – not to the question of whether the API packages are copyrightable. We agree.

Connectix: 203 F.3d 596 (9th Cir. 2000)

The district court characterized *Accolade* and *Sony Computer Entertainment v. Connectix Corp.* as "close analogies" to this case. According to the court, both decisions "held that interface procedures that were necessary to duplicate in order to achieve interoperability were functional aspects not copyrightable under Section 102(b)." The district court's reliance on *Accolade* and *Connectix* in the copyrightability context is misplaced, however. Both cases were focused on fair use, not copyrightability. In *Accolade*, for example, the only question was whether Accolade's intermediate copying was fair use. The court never addressed the question of whether Sega's software code, which had functional elements, also contained separable creative expression entitled to protection.

This is not a case where Google reverse-engineered Oracle's Java packages to gain access to unprotected functional elements contained therein. Had Google reverse engineered the programming packages to figure out the ideas and functionality of the original, and then created its own structure and its own literal code, Oracle would have no remedy under copyright whatsoever. Instead, Google chose to copy both the declaring code and the overall SSO of the 37 Java API packages at issue.

We disagree with Google's suggestion that*Accolade* and *Connectix* created an "interoperability exception" to copyrightability. Because copyrightability is focused on the choices available to the plaintiff at the time the computer program was created, the relevant compati-

bility inquiry asks whether the plaintiff's choices were dictated by a need to ensure that its program worked with existing third-party programs. Whether a defendant later seeks to make its program interoperable with the plaintiff's program has no bearing on whether the software the plaintiff created had any design limitations dictated by external factors. Stated differently, the focus is on the compatibility needs and programming choices of the party claiming copyright protection – not the choices the defendant made to achieve compatibility with the plaintiff's program. Consistent with this approach, courts have recognized that, once the plaintiff creates a copyrightable work, a defendant's desire "to achieve total compatibility... is a commercial and competitive objective which does not enter into the ... issue of whether particular ideas and expressions have merged." *Apple v. Franklin*

Whether Google's software is "interoperable" in some sense with any aspect of the Java platform (although as Google concedes, certainly not with the JVM) has no bearing on the threshold question of whether Oracle's software is copyrightable. It is the interoperability and other needs of Oracle – not those of Google – that apply in the copyrightability context, and there is no evidence that when Oracle created the Java API packages at issue it did so to meet compatibility requirements of other pre-existing programs.

Google maintains on appeal that its use of the "Java class and method names and declarations was 'the only and essential means' of achieving a degree of interoperability with existing programs written in the Java language." Indeed, given the record evidence that Google designed Android so that it would *not* be compatible with the Java platform, or the JVM specifically, we find Google's interoperability argument confusing. While Google repeatedly cites to the district court's finding that Google had to copy the packages so that an app written in Java could run on Android, it cites to no evidence in the record that any such app exists and points to no Java apps that either pre-dated or post-dated Android that could run on the Android platform. The compatibility Google sought to foster was not with Oracle's Java platform or with the JVM central to that platform. Instead, Google wanted to capitalize on the fact that software developers were already trained and experienced in using the Java API packages at issue. Google's interest was in accelerating its development process by leveraging Java for its existing base of developers. Although this competitive objective might be relevant to the fair use inquiry, we conclude that it is irrelevant to the copyrightability of Oracle's declaring code and organization of the API packages.

Finally, to the extent Google suggests that it was entitled to copy the Java API packages because they had become the effective industry standard, we are unpersuaded. Google cites no authority for its

suggestion that copyrighted works lose protection when they become popular, and we have found none. In fact, the Ninth Circuit has rejected the argument that a work that later becomes the industry standard is uncopyrightable. *See Practice Management. Info. Corp. v. Am. Med. Ass'n* (noting that the district court found plaintiff's medical coding system entitled to copyright protection, and that, although the system had become the industry standard, plaintiff's copyright did not prevent competitors "from developing comparative or better coding systems and lobbying the federal government and private actors to adopt them. It simply prevents wholesale copying of an existing system."). Google was free to develop its own API packages and to lobby programmers to adopt them. Instead, it chose to copy Oracle's declaring code and the SSO to capitalize on the preexisting community of programmers who were accustomed to using the Java API packages. That desire has nothing to do with copyrightability. For these reasons, we find that Google's industry standard argument has no bearing on the copyrightability of Oracle's work.

Practice Management: 121 F.3d 516, (9th Cir. 1997)

[On remand, the jury found that Google's copying was a fair use. The District Court declined to set aside the verdict. The case is now on appeal again on a variety of grounds.]

Tetris Problem

Your client, Thoth Software, would like to create and sell a version of Tetris for the Digix gaming console. What aspects of the game can Thoth imitate without fear of liability? The name? Falling blocks? The shapes of the blocks? Their colors? Lines that disappear when completely filled in? The music? The graphics around the play field?

Based on *Tetris Holding, LLC v. Xio Interactive, Inc.*, 863 F. Supp. 2d 394 (D.N.J. 2012) and *Spry Fox LLC v. LOLApps Inc.*, 104 U.S.P.Q.2d 1299 (W. D. Wash 2012)

2 Defenses

When in 1980 Congress confirmed that software was copyrightable as a literary work, it rewrote Section 117 to codify a kind of limited exhaustion rule for software. *Accolade* shows that the courts have also used fair use to create a fairly robust reverse engineering defense to software copyright. Although it arises in the DMCA context, *Corley* shows how the courts have applied the First Amendment to software copyright.

Copyright Act

(a) *Making of Additional Copy or Adaptation by Owner of Copy.* – Notwithstanding the provisions of section 106, it is not an infringement for the owner of a copy of a computer program to make or authorize the making of another copy or adaptation of that computer program provided:

17 U.S.C. § 117
Limitations on exclusive rights: Computer programs

(1) that such a new copy or adaptation is created as an essential step in the utilization of the computer program in conjunction with a machine and that it is used in no other manner, or

(2) that such new copy or adaptation is for archival purposes only and that all archival copies are destroyed in the event that continued possession of the computer program should cease to be rightful.

(b) *Lease, Sale, or Other Transfer of Additional Copy or Adaptation*. – Any exact copies prepared in accordance with the provisions of this section may be leased, sold, or otherwise transferred, along with the copy from which such copies were prepared, only as part of the lease, sale, or other transfer of all rights in the program. Adaptations so prepared may be transferred only with the authorization of the copyright owner.

(c) *Machine Maintenance or Repair*. – Notwithstanding the provisions of section 106, it is not an infringement for the owner or lessee of a machine to make or authorize the making of a copy of a computer program if such copy is made solely by virtue of the activation of a machine that lawfully contains an authorized copy of the computer program, for purposes only of maintenance or repair of that machine, if –

(1) such new copy is used in no other manner and is destroyed immediately after the maintenance or repair is completed; and

(2) with respect to any computer program or part thereof that is not necessary for that machine to be activated, such program or part thereof is not accessed or used other than to make such new copy by virtue of the activation of the machine.

Sega Enterprises Ltd. v. Accolade, Inc.

977 F.2d 1510 (9th Cir. 1992)

Sega develops and market video entertainment systems, including the "Genesis" console and video game cartridges. Sega licenses its copyrighted computer code and its "SEGA" trademark to a number of independent developers of computer game software. Those licensees develop and sell Genesis-compatible video games in competition with Sega. Accolade is not and never has been a licensee of Sega.

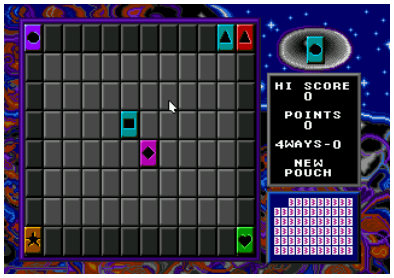
Sega Genesis console

Accolade reverse engineered Sega's video game programs in order to discover the requirements for compatibility with the Genesis console. As part of the reverse engineering process, Accolade transformed the machine-readable object code contained in Sega's game

cartridges into human-readable source code using a process called "disassembly" or "decompilation"[2] Accolade purchased a Genesis console and three Sega game cartridges, wired a decompiler into the console circuitry, and generated printouts of the resulting source code. Accolade engineers studied and annotated the printouts in order to identify areas of commonality among the three game programs. They then loaded the disassembled code back into a computer, and experimented to discover the interface specifications for the Genesis console by modifying the programs and studying the results. At the end of the reverse engineering process, Accolade created a development manual that incorporated the information it had discovered about the requirements for a Genesis-compatible game. The manual contained only functional descriptions of the interface requirements and did not include any of Sega's code.

What about cases like *Barr-Mullin* and *Silvaco*, which asserted that reverse engineering object code was impossible?

[In creating its own games for the Genesis, Accolade] did not copy Sega's programs, but relied only on the information concerning interface specifications for the Genesis that was contained in its development manual. With the exception of the interface specifications, none of the code in Accolade's games is derived in any way from its examination of Sega's code. In 1990, Accolade released "Ishido", a game which it had originally developed and released for use with the Macintosh and IBM personal computer systems, for use with the Genesis console.



Ishido screenshot

Accolade contends that its disassembly of copyrighted object code as a necessary step in its examination of the unprotected ideas and functional concepts embodied in the code is a fair use. Because, in the case before us, disassembly is the only means of gaining access to those unprotected aspects of the program, and because Accolade has a legitimate interest in gaining such access (in order to determine how to make its cartridges compatible with the Genesis console), we agree with Accolade. Where there is good reason for studying or examining the unprotected aspects of a copyrighted computer program, disassembly for purposes of such study or examination constitutes a fair use.

There is no evidence in the record that Accolade sought to avoid performing its own creative work. Indeed, most of the games that Accolade released for use with the Genesis console were originally developed for other hardware systems.

Accolade copied Sega's software solely in order to discover the functional requirements for compatibility with the Genesis console – aspects of Sega's programs that are not protected by copyright. To the extent that there are many possible ways of accomplishing

[2]Disassembly devices are commercially available and are widely used within the software industry.

a given task or fulfilling a particular market demand, the programmer's choice of program structure and design may be highly creative and idiosyncratic. However, computer programs are, in essence, utilitarian articles – articles that accomplish tasks. As such, they contain many logical, structural, and visual display elements that are dictated by the function to be performed, by considerations of efficiency, or by external factors such as compatibility requirements and industry demands. *Altai*

Sega argues that even if many elements of its video game programs are properly characterized as functional and therefore not protected by copyright, Accolade copied protected expression. Sega is correct. The record makes clear that disassembly is wholesale copying. Because computer programs are also unique among copyrighted works in the form in which they are distributed for public use, however, Sega's observation does not bring us much closer to a resolution of the dispute.

The unprotected aspects of most functional works are readily accessible to the human eye. The systems described in accounting textbooks or the basic structural concepts embodied in architectural plans, to give two examples, can be easily copied without also copying any of the protected, expressive aspects of the original works. Computer programs, however, are typically distributed for public use in object code form, embedded in a silicon chip or on a floppy disk. For that reason, humans often cannot gain access to the unprotected ideas and functional concepts contained in object code without disassembling that code – i.e., making copies. If disassembly of copyrighted object code is per se an unfair use, the owner of the copyright gains a de facto monopoly over the functional aspects of his work – aspects that were expressly denied copyright protection by Congress.

Universal City Studios, Inc. v. Corley

273 F.3d 429 (2001)

Our case concerns a security device, CSS computer code, that prevents access by unauthorized persons to DVD movies. The CSS code is embedded in the DVD movie. Access to the movie cannot be obtained unless a person has a device, a licensed DVD player, equipped with computer code capable of decrypting the CSS encryption code. In its basic function, CSS is like a lock on a homeowner's door, a combination of a safe, or a security device attached to a store's products.

DeCSS is computer code that can decrypt CSS. In its basic function, it is like a skeleton key that can open a locked door, a combination that can open a safe, or a device that can neutralize the security device attached to a store's products. DeCSS enables anyone to gain access to a DVD movie without using a DVD player.

The initial use of DeCSS to gain access to a DVD movie creates no

loss to movie producers because the initial user must purchase the DVD. However, once the DVD is purchased, DeCSS enables the initial user to copy the movie in digital form and transmit it instantly in virtually limitless quantity, thereby depriving the movie producer of sales. The advent of the Internet creates the potential for instantaneous worldwide distribution of the copied material.

[The District Court found that DeCSS was a "technology" that was "primarily designed" to circumvent CSS in violation of § 1201(a)(2) of the DMCA. It enjoined the defendant-appellants from posting the DeCSS code to the Internet.]

At first glance, one might think that Congress has as much authority to regulate the distribution of computer code to decrypt DVD movies as it has to regulate distribution of skeleton keys, combinations to safes, or devices to neutralize store product security devices. However, despite the evident legitimacy of protection against unauthorized access to DVD movies, just like any other property, regulation of decryption code like DeCSS is challenged in this case because DeCSS differs from a skeleton key in one important respect: it not only is capable of performing the function of unlocking the encrypted DVD movie, it also is a form of communication, albeit written in a language not understood by the general public. As a communication, the DeCSS code has a claim to being "speech," and as "speech," it has a claim to being protected by the First Amendment.

Computer programs are not exempted from the category of First Amendment speech simply because their instructions require use of a computer. A recipe is no less "speech" because it calls for the use of an oven, and a musical score is no less "speech" because it specifies performance on an electric guitar. Arguably distinguishing computer programs from conventional language instructions is the fact that programs are executable on a computer. But the fact that a program has the capacity to direct the functioning of a computer does not mean that it lacks the additional capacity to convey information, and it is the conveying of information that renders instructions "speech" for purposes of the First Amendment. The information conveyed by most "instructions" is how to perform a task.

The Appellants vigorously reject the idea that computer code can be regulated according to any different standard than that applicable to pure speech, i.e., speech that lacks a nonspeech component. Although recognizing that code is a series of instructions to a computer, they argue that code is no different, for First Amendment purposes, than blueprints that instruct an engineer or recipes that instruct a cook. We disagree. Unlike a blueprint or a recipe, which cannot yield any functional result without human comprehension of its content, human decision-making, and human action, computer code can instantly cause a computer to accomplish tasks and instantly render

The District Court noted in a footnote, "Professor Touretzky of Carnegie Mellon University convincingly demonstrated that computer source and object code convey the same ideas as various other modes of expression, including spoken language descriptions of the algorithm embodied in the code. He drew from this the conclusion that the preliminary injunction irrationally distinguished between the code, which was enjoined, and other modes of expression that convey the same idea, which were not,, although of course he had no reason to be aware that the injunction drew that line only because that was the limit of the relief plaintiffs sought. With commendable candor, he readily admitted that the implication of his view that the spoken language and computer code versions were substantially similar was not necessarily that the preliminary injunction was too broad; rather, the logic of his position was that it was either too broad or too narrow. Once again, the question of a substantially broader injunction need not be addressed here, as plaintiffs have not sought broader relief."

the results of those tasks available throughout the world via the Internet. The only human action required to achieve these results can be as limited and instantaneous as a single click of a mouse. These realities of what code is and what its normal functions are require a First Amendment analysis that treats code as combining nonspeech and speech elements, i.e., functional and expressive elements.

We recognize that the functional capability of computer code cannot yield a result until a human being decides to insert the disk containing the code into a computer and causes it to perform its function (or programs a computer to cause the code to perform its function). Nevertheless, this momentary intercession of human action does not diminish the nonspeech component of code, nor render code entirely speech, like a blueprint or a recipe.

Neither the DMCA nor the posting prohibition is concerned with whatever capacity DeCSS might have for conveying information to a human being, and that capacity, as previously explained, is what arguably creates a speech component of the decryption code. The DMCA and the posting prohibition are applied to DeCSS solely because of its capacity to instruct a computer to decrypt CSS. That functional capability is not speech within the meaning of the First Amendment. The Government seeks to justify both the application of the DMCA and the posting prohibition to the Appellants solely on the basis of the functional capability of DeCSS to instruct a computer to decrypt CSS, i.e., without reference to the content of the regulated speech. This type of regulation is therefore content-neutral, just as would be a restriction on trafficking in skeleton keys identified because of their capacity to unlock jail cells, even though some of the keys happened to bear a slogan or other legend that qualified as a speech component.

As a content-neutral regulation with an incidental effect on a speech component, the regulation must serve a substantial governmental interest, the interest must be unrelated to the suppression of free expression, and the incidental restriction on speech must not burden substantially more speech than is necessary to further that interest. The Government's interest in preventing unauthorized access to encrypted copyrighted material is unquestionably substantial, and the regulation of DeCSS by the posting prohibition plainly serves that interest. Moreover, that interest is unrelated to the suppression of free expression.

Posting DeCSS on the Appellants' web site makes it instantly available at the click of a mouse to any person in the world with access to the Internet, and such person can then instantly transmit DeCSS to anyone else with Internet access. Although the prohibition on posting prevents the Appellants from conveying to others the speech component of DeCSS, the Appellants have not suggested, much less

shown, any technique for barring them from making this instantaneous worldwide distribution of a decryption code that makes a lesser restriction on the code's speech component.

D Trademark

While trademark and trade dress law are excellent for protecting trademarks *for* software, they don't do particularly well protecting software *qua* software.

Apple Inc. v. Samsung Electronics Co., Ltd.

786 F.3d 983 (Fed. Cir. 2015)

Apple sued Samsung in April 2011. On August 24, 2012, the first jury reached a verdict that numerous Samsung smartphones infringed and diluted Apple's patents and trade dresses in various combinations. The diluted trade dresses are Trademark Registration No. 3,470,983 and an unregistered trade dress defined in terms of certain elements in the configuration of the iPhone.

The '983 trade dress is a federally registered trademark. The '983 trade dress claims the design details in each of the sixteen icons on the iPhone's home screen framed by the iPhone's rounded-rectangular shape with silver edges and a black background:

> The first icon depicts the letters "SMS" in green inside a white speech bubble on a green background; ...
> the seventh icon depicts a map with yellow and orange roads, a pin with a red head, and a red-and-blue road sign with the numeral "280" in white; ...
> the sixteenth icon depicts the distinctive configuration of applicant's media player device in white over an orange background.

Apple's '983 trade dress registration

It is clear that individual elements claimed by the '983 trade dress are functional. For example, there is no dispute that the claimed details such as "the seventh icon depicts a map with yellow and orange roads, a pin with a red head, and a red-and-blue road sign with the numeral '280' in white" are functional. Apple's user interface expert testified on how icon designs promote usability. This expert agreed that "the whole point of an icon on a smartphone is to communicate to the consumer using that product, that if they hit that icon, certain functionality will occur on the phone." The expert further explained that icons are "visual shorthand for something" and that "rectangular containers" for icons provide "more real estate" to accommodate the icon design. Apple rebuts none of this evidence.

Apple contends instead that Samsung improperly disaggregates the '983 trade dress into individual elements to argue functionality. But Apple fails to explain how the total combination of the sixteen

icon designs in the context of iPhone's screen-dominated rounded-rectangular shape – all part of the iPhone's "easy to use" design theme – somehow negates the undisputed usability function of the individual elements. Apple's own brief even relies on its expert's testimony about the "instant recognizability due to highly intuitive icon usage" on "the home screen of the iPhone." Apple's expert was discussing an analysis of the iPhone's overall combination of icon designs that allowed a user to recognize quickly particular applications to use. The iPhone's usability advantage from the combination of its icon designs shows that the '983 trade dress viewed as a whole "s nothing other than the assemblage of functional parts. There is no separate overall appearance which is non-functional. The undisputed facts thus demonstrate the functionality of the '983 trade dress.

E Design Patent

The Apple-Samsung smartphone litigation has put design patents on the map in a good way. We saw in the previous chapter that they provided a natural solution to the problem of protecting three-dimensional designs. How well do they fit software, and what aspects of software might they cover?

17 Stan. Tech. Rev. 53 (2013)

Michael Risch
Functionality and Graphical User Interface Design Patents

Today, design patents cover only images displayed on a screen. As many claims are written, merely viewing an image on a blog page or in a PDF file associated with this Article (which includes some patented images) would constitute infringement. From a theoretical point of view, something seems off about that: viewing an image on a display screen can hardly be considered an article of manufacture, yet the law outlaws precisely such use, *even if one is simply viewing the patent itself on a computer*!

Thus, the second threshold question is whether an ephemeral image, viewable anywhere and in any context, can be considered an "article of manufacture" under the statute. The guidelines, issued in 1996, give surprisingly little attention to this question. Courts have long held that "surface ornamentation" constitutes an article of manufacture, and displayed images are part of a surface. Thus, the guidelines only ask whether the image is part of a display, not whether an ephemeral image is they type of thing that should *ever* be protected.

Even if one accepts that copyrightable works should be protected by design patent, this does not mean that all copyrightable expression qualifies for patent protection. Consider, for example, protection of structure, sequence, and organization of factual information.

This may well be protected by copyright, but does not fall under the design patent umbrella.

Protection of user interfaces essentially merges the copyright law's fixation requirement with patent law's "article of manufacture" requirement. Fixation is the cornerstone of copyright: no work can be protected if it is not fixed in a tangible medium. But fixation is far from permanent; loading a file into computer memory is sufficiently fixed, even if the computer could be turned off or the memory changed.

The question, then, is whether any image present in computer memory – fixed, to be sure – becomes an "article of manufacture," even if it is not displayed on the screen at all times. Thus far, the PTO has said yes, and courts have not asked the question, assuming that if a patent is issued, then it must be an article of manufacture. Indeed, design patents now protect "animations," which are a series of images that move in sequence, such as a spinning icon or a simulated folding of a page to emulate a book on a display screen.

Protection of displays appears to rest on two seminal cases issued by that Court of Customs & Patent Appeals, the precursor to the Federal Circuit. The first case is *Hruby*, which held that the shape of water moving in a fountain could be a patented design, even though the water was moving and could be turned off. The second precedent is *In re Zahn* in which the C.C.P.A. ruled that a portion of a manufacture could be separated by a "broken line" to separate the new, protectable design from the preexisting remainder of the article. Though the notion that a portion of an article could be patented is more than 140 years old, the Zahn court's broken line rule leads to the near ubiquity of broken lines in graphical displays that separate the image from the rest of the display.

Zahn: 617 F.2d 261 (C.C.P.A. 1980)

Despite the apparent reasonableness of *Hruby* and *Zahn* with respect to the facts of those cases, their extension to user interfaces is troubling. Courts and commentators have simply not asked the difficult questions. Collapsing fixation, animation, and display screen into an "article of manufacture" leaves design patents on a very slippery slope.

For example, there is no theoretical bar to protecting every displayed copyrightable work with a design patent. Every television show and movie is theoretically a novel and non-obvious design to be incorporated into display screens everywhere. Indeed, every photograph captured and displayed on every mobile device might be protected. Every doodle on an electronic Etch-a-Sketch could be patented. Any use of the material would be infringing, without any consideration of fair use, the ideas represented by the work, or even the First Amendment.

Further, and perhaps more unsettling, the only apparent reason

why such claims have not been made before is that nobody thought to do so, because there is no body of law to avoid such an outcome. The PTO has almost no tools to reject small, or even large, snippets of movies. While only a single inventive design may be covered by a patent, multiple patents might be filed on different—but important—segments of audiovisual works, sufficient to block all downstream use with no fair use defense. A design patent protecting four or five screen captures from the famous Hitler *Downfall* movie scene would eliminate all claims to fair use of that short but endlessly entertaining parody clip. The PTO has no track record of rigorously examining images to determine whether they are novel or obvious. Even if it did, the exact combination of images in that screen is unlikely to appear elsewhere.

This parade of horribles might be solved in two ways. First, courts could recognize that an article of manufacture (or portion thereof) requires more than copyright fixation. Instead, an article of manufacture requires permanence at the point of manufacture, display, and use. To be sure, many elements might be hidden at one point or another, such as bottoms of drawers, collapsible devices, folding elements that become hidden, or even water fountains. But each of these examples is different in kind from the ephemeral images on a display screen. Ephemeral images can be configured by moving bits in memory, and as such, they are not ornamental articles of manufacture; instead, they are displays of information. In short, the PTO's 1996 concern about patenting images was well founded, but the solution was not to add "on a display screen" to patent claims. Instead, the solution was to recognize that images divorced from manufacture do not qualify as articles – they can be shown on any article, any screen, and any device, and that is not what design patents are meant to protect.

A second solution might recognize that modern commercial products live under a big tent. As such, there may be times when the design of the product includes designs on the screen. However, protection for displayed surface designs should be limited in a number of ways to ensure that the design is an ornamental article of manufacture, rather than an ephemeral image.

Smartphone Problem

Based on *Apple Inc. v. Samsung Electronics Co., Ltd.*, 786 F.3d 983 (Fed. Cir. 2015)

Here are Figure 1 from Design Patent 604,305 (left), assigned to Apple, and a picture of the Samsung Galaxy S (right). Does the Galaxy S infringe the '305 pagent?


AT&T
4:00 PM
SMS
Text
Calendar
Photos
Camera
YouTube
Stocks
Maps
Weather
Clock
Calculator
Notes
Settings
Phone
Mail
Safari
iPod


SAMSUNG
1:28 AM
Calculator
Calendar
Camera
Clock
Daily Briefing
E-book
Email
FM Radio
Gallery
Gmail
Internet
Maps
Market
Memo
Mini diary
Music player
Phone
Cantacts
Message
Home