# Patently Absurd, Part 1

WIRED 2.07 Features
Simson Garfinkel
http://simson.net/clips/94.Wired.PatentlyAbsurd.txt

Transmitted: 94-07-19 19:44:48 EDT

How could the Patent Office ever grant a patent to Compton's on its claim to have invented multimedia?

This is how.

Simson L. Garfinkel takes us inside the Patent Office

Something is terribly wrong in Crystal City, and everybody who's thinking about intellectual property knows it.

The problem is rooted in a federal office complex across the Potomac River from Washington, DC, on the fourth floor of Crystal Park 2. That is the home of the US Patent and Trademark Office's Group 2300, the group called "Computer Systems and Computer Applications." Group 2300 is the one that's been carving up computer science and handing out little monopolies to nearly half the people who take the time to file a patent application.

Heather Herndon is a patent examiner in Group 2300. Herndon sits at her desk in an office that's crowded but meticulously organized. At her left is a pile of "file wrappers" – bulging blue folders held together with rubber bands and steel clips – that hold patent applications in progress. At her side is a Dunn 386 PC – a cheap IBM clone – with a low-speed 2400-baud modem. All around her are cabinets and drawers filled with paper files: mostly old patent applications, as well as some particularly important articles that summarize landmark advances in computer science. And on the table right before her is a prime example of what all the fuss is about: a software patent.

The really interesting software patents are the ones that examiners like Herndon and attorneys around the country are fighting over right now – the ones that haven't yet been granted. And under US law, the Patent Office is forbidden from discussing or revealing any application in progress.

Because of the confidentiality rules, Herndon has prepared for this meeting (one of the first granted by the Patent Office between a journalist and an examiner on the subject of software patents) by going into her files and finding a patent that's already been issued. It's Patent Number 5,060,171, "a system and method for superimposing images," issued to Stephen C. Steir, et al.

Steir's patent describes "an image enhancement system and method" that shows people what they might look like with a different head of hair. It uses a simple technique to combine an image of a person's head with a second image of a disembodied hair piece: first you find and smooth the boundary between the two images, then you figure out a new color for each pixel in the boundary by averaging the pixels in the nearby vicinity. The patent describes how you can change the size of the hairstyle image, so that a single hairdo can be placed over tall, thin heads or short, fat ones.

Read through Patent Number 5,060,171 and you'll learn everything about how Steir's invention works. If you are a programmer, you might be tempted to go home and throw together a few hundred lines of C++ to try out the algorithm yourself. You might even demonstrate that program to your bald Uncle Arthur, who happens to be considering a hair transplant.

But if you do, you'll be breaking the law.

That's because Patent Number 5,060,171 gives Clearpoint Research (the company to which Steir et al assigned their patents) an absolute monopoly on the techniques it describes – until October 22, 2008. And that monopoly is backed up by the federal courts, the federal marshals, and ultimately the entire United States Army.

## Ignorance Is Not a Defense

There are three main ways to prevent people from ripping off your intellectual property in the United States: trade secrets, copyrights, and patents. But until recently, few programmers or lawyers bothered to patent the techniques described by computer programs. Many people thought that patenting computer programs was neither ethical nor legal; most of the rest thought that it wasn't necessary.

Programs, after all, were protected by copyright laws. What's fundamentally vexing about software patents is that they can be unknowingly violated by any programmer, even one who has never heard of the patent and who

independently implements the invention. What may seem like a trivial hack to a gifted programmer may already be a patented routine.

Just about the only way programmers can protect themselves from patent infringement is by paying for a patent search – both a time-consuming and an expensive process – which still doesn't conclusively determine whether or not a program is in the clear.

A patent search can cost anywhere from a few hundred dollars to a few thousand. Simply finding the software patents can be a challenge, since the Patent Office doesn't identify them as such. Then there's the sheer number of software patents that the Patent Office has issued: more than 12,000, according to Gregory Aharonian, who runs the Internet Patent News Service, a free Internet mailing list that tracks newly issued patents and developments in intellectual property law. Aharonian has been tracking government-funded software for the last ten years and patents for the last three.

Beyond the Patent Office, complications are born of the nature of software itself. Develop a new drug or a new electronic circuit: seeing whether the invention has ever been patented before is a fairly straightforward task. But after nearly a decade of granting software patents, the Patent Office still hasn't come up with a system for organizing them that's in a language that most programmers can understand. Many software patents – such as Patent Number 5,060,171 – don't even have the words "computer," "software," "program," or "algorithm" in their abstracts, making them harder to locate for those unfamiliar with the terrain. And no easy way exists to take a few lines or a page of computer code and get a list of the patents that it might violate. Identifying all of the patentable inventions inside an entire program is nearly impossible. A typical piece of software might violate three. Then again, it might violate a hundred.

So, patent searches for computer programs become expensive propositions. But violating a patent can be more expensive still. According to Stanford University Professor John Barton, patent infringement suits are among the most expensive kind of litigation in the US today, with the average cost of a patent suit being US$500,000 per side per claim. Not surprisingly, the cost of insurance to protect companies against patent infringement is equally steep: $50,000 per product with a $50,000 deductible in the case of multimedia software, says Rob Lippincott, president of the Interactive Multimedia Association, a trade organization for large and small multimedia publishers. "These kinds of numbers are basically intolerable," says Lippincott, adding that the cost of merely defending an infringement will wipe out most small

software houses, whether they win or lose.

But merely defending an infringement suit is peanuts next to losing one. Just ask Microsoft, which was found guilty by a jury of violating STAC Electronics's patent on data compression in February 1994. The cost: $120 million in compensatory damages.

The alternative to a lawsuit, of course, is licensing any software patents that a program might happen to infringe. But then the costs can really swell. For instance, the REFAC Technology Development Corporation of New York holds Patent Number 4,398,249 on natural order recalculation. In 1989, REFAC sued Lotus, demanding 5 percent of all revenues from Lotus 1-2-3. Five years later, the case is still pending in a New York federal court. And last year, when Compton's New Media, a division of Tribune Co., was awarded a far-ranging patent in multimedia databases, the company indicated that it wanted 1-3 percent of net revenue for any potentially infringing multimedia product. License just a few of these patents, and you've licensed away your profit margin.

Such dangers terrify small businesses, shareware authors, and people writing free software. A person writing a simple utility can find himself or herself at the wrong end of a multimillion-dollar lawsuit for lost profits, with treble damages if a court finds that the infringement was "willful." "Software patents turn every decision you make while writing a program into a legal risk," says Richard Stallman, one of the leaders of the opposition to software patents. "They make writing a large program like crossing a minefield. Each step has a small chance of stepping on a patent and blowing you up."

The League for Programming Freedom, which Stallman helped create, has called for the elimination of patents that apply to computer programs. But it may be too late for such sweeping and simplistic reforms.

## Who Are These Guys?

At the heart of the software patent crisis is the Patent and Trademark Office itself. In recent years, the office has issued a steady stream of patents on techniques that nobody would have dreamt patentable just a few years before. The patents can be dauntingly broad. Recently, for example, the Patent Office awarded Patent Number 5,173,051, which describes a system for "curriculum planning and publishing" using a computer and a videodisc player. The patent was filed October 15, 1991, by Optical Data Corporation

of Warren, New Jersey, and issued December 22, 1992 – record time for a patent that, in the words of Harvard researcher Brian Kahin, "makes out millions of teachers to be infringers."

Then there is Patent Number 5,105,184, the so-called "Energizer Bunny Patent," issued in 1992 to Software Advertising Corp. It covers "displaying and integrating commercial advertisements with computer software" – in other words, any advertisement integrated into a screen saver. Or Patent Number 5,263,127, granted last year to an enterprising engineer at Digital Equipment Corp., which patents a technique used in object-oriented programming that can be implemented by using exactly two machine-language instructions.

This year, the Patent Office will issue more than 100,000 patents, 4,000 of which could be classified as software patents, says Gregory Aharonian. Since it typically takes two to four years for the Patent Office to grant a patent, and since newly issued patents apply retroactively to any product that was created after the patent application was filed, companies that want to avoid an infringement suit need to keep track of the new patents – a full-time job for a team of highly trained people.

As a result, few developers realize that they are infringing a patent until "they get the letter from the law firm that they've never heard of in the city they've never been to that says they are infringing US Patent Number so-and-so," says Robert Merges, a professor of patent law at Boston University.

One person who found himself on the wrong side of such an infringement suit is Vern Blanchard, a programmer whose company was destroyed by a patent that wasn't even valid.

Blanchard is president of American Multi-Systems, a San Diego-based company that wrote a program to let professional bingo players play dozens of bingo cards at the same time. The program runs on an IBM PC. Blanchard was ready to start marketing his system, along with custom-built tables and personal computers, to the big-time bingo halls, when one of his competitors filed suit against American Multi-Systems for patent infringement.

The case should have been thrown out of court for two reasons, says Blanchard. For starters, he says, his competitor's patent "covered a hand-held calculator type device," not a general-purpose computer running a program. And the patent couldn't cover general-purpose computers, he says, because programs that play bingo are commonly written by students in introductory computer science courses. There was simply nothing novel or new about the technique that the patent described, and novelty is a basic requirement of

patentability.

Nevertheless, says Blanchard, the company that held the patent was able to convince a judge to grant a preliminary injunction that took American Multi-Systems's product off the market.

Eventually Blanchard discovered a critical piece of "prior art" – concrete evidence that the invention described by the patent had been thought of before by somebody else – and was able to convince the judge to lift the preliminary injunction. Unfortunately, by that time American Multi-Systems had effectively put itself out of business with legal fees.

"Judges are not particularly literate in technical issues," says Blanchard. "When they see a patent they presume that it's valid, as they should. (To them), if the Patent Office says that this is a valid patent, well, of course it's a valid patent."

In practice, these simple rules have created headaches for the 150 patent examiners of Group 2300.

In order to be a patent examiner, you've got to have a college degree in science or engineering, including at least 30 credit-hours of specific science-related courses, before applying for the job and taking an exam. Once you are hired, the Patent Office's in-house training takes over.

Upon hiring, every examiner gets an intensive two-week course that teaches the basics of patent examination – the standards of patentability, where to look to find prior art, and how to evaluate patent documents.

That course is followed by four more two-week sessions over the next eight months. But the real training for a new patent examiner comes on the job. Each new examiner's supervisor has the ultimate signature authority over the newcomer's patent applications. Trainees don't get to sign their own patents until they've been at the Patent Office between three and five years.

The Patent Office has gotten quite good at recruitment and training. It has to be: few examiners stay on for more than a few years. This despite the benefits that being a patent examiner carries: high pay by federal standards ($38,000 to $75,000 a year, before overtime), flex-time arrangements, and tremendous job satisfaction.

At the same time, the job invites burnout. Each examiner has a quarterly quota that must be filled. As a result, examiners frequently put in overtime: up to 40 hours every two weeks. Since applications must be held in the utmost secrecy, examiners are forbidden to work at home, which makes long hours even less attractive.

Then there is the honey pot of private practice.

The Patent Office encourages its examiners to take courses at nearby law schools so that they can improve the job they do. Unfortunately, the idea frequently backfires: soon after examiners earn their law degrees, they frequently leave the government for a lucrative job in private practice on the other side of the patent bar.

"I know somebody who recently left the office who had been in there approximately three or four years. Their take-home pay doubled when they left the office. And they expect increases in salary of $15,000 to $20,000 over the next year," said David Clark, a patent examiner dropout.

## Idea vs. Implementation

No matter how good it is, you can't patent an idea in the United States: patents are only awarded in this country for usable inventions that accomplish tasks.

Computer programs, occupying that strange world between mathematical ideas and applied engineering, have posed a problem for the courts since the 1960s. On the one hand, the public has traditionally viewed computer programs as mathematical. On the other hand, programs can solve real problems.

In 1972, the US Supreme Court spoke for the first time on the subject of software patents. In the case of Gottschalk v. Benson, the court denied a patent on a system for converting binary-coded-decimal numbers into decimal numbers. The Court's decision was based on the notion that code was preeminently mathematical. The Court's decision said, in part, that if patents on algorithms were allowed, "the patent would wholly preempt the mathematical formula" for other uses.

Faced with the Court's ruling, patent attorneys simply bypassed the problem by framing the language of their patent applications so that software inventions seemed like hardware devices. For example, in July 1973, AT&T filed for a patent on the fundamental technique used by the Unix operating system to enforce computer security (this eventually came to be known as the SUID Patent). But instead of describing the invention as a software code, AT&T's attorney, Stephen Phillips, described the invention with a circuit diagram containing 11 chips connected by more than 40 wires. (The patent was granted in January 1979 but was dedicated to the public domain just ten months later.)

Disguising software patents with hardware implementations was a common trick, says Rick Jordan, patent counsel at Thinking Machines, a maker of supercomputers. "Instead of being up-front about the fact that their product was embodied in software, people would go through subterfuge to make it look like a product was embodied in hardware," and thus make it eligible for a patent as a device which performed a process or embodied a technique.

But Phillips needn't have bothered with his subterfuge. Just five years after the Benson case, the US Court of Customs and Patent Appeals reinterpreted the Supreme Court's decision. In a case called In re Freeman, the appellate court upheld a patent on a system for typesetting mathematical equations, arguing that the Supreme Court really meant that the Benson case only forbid patents on "mathematical algorithms." Unfortunately, the appellate court neglected to define the term "mathematical algorithm."

The search for the proper definition of the term "software patent" may be moot, argues Joe Dixon, a supervisory patent examiner who runs Art Unit 2312, or "Storage and Retrieval." He adds: "We do not grant 'software patents.' " What the Patent Office grants, insists Dixon, are patents on methods or processes that can be embodied in computer programs. This is the reason that words like "software," "program," and "algorithm" don't usually appear in abstracts of most so-called software patents: the law is indifferent as to whether the invention is built with a program or with a bunch of integrated circuits and wires.

Obtaining software patents got easier in 1981, when the Supreme Court ruled in favor of a patent applicant in Diamond v. Diehr. In that case, Diehr had applied for a patent on a system for vulcanizing rubber that used a computer program to control the temperature of the rubber mold. The Court ruled that the patent application's inclusion of a computer program didn't render the application unpatentable. In essence, the 1981 decision gave the clear impression to the lower courts and the Patent Office that the Court felt software patents were OK. Since then, the number of software patents granted each year has nearly doubled.

(story continued, see *Patently Absurd, Part 2*)

Simson L. Garfinkel (simsong@mit.edu) is a computer consultant and science writer.