Fabrice Bellard

Andy Gocke
and
Nick Pizzolato

There are some computer scientists whose names are immediately recognized in the field: Alan Turing, Donald Knuth, Edsger Dijkstra. These people have become names and personalities larger than their (groundbreaking) achievements. Alan Turing was so influential as to have his name permanently attached to the most prestigious award from the Association for Computing Machinery (ACM), arguably the most prestigious in computer science.[Turing1] Aside from winning said award, Knuth and Dijkstra are also known for revolutions in algorithms and data structures. These computer scientists have become personalities as well as authors, earning universal respect and sometimes a cult following.[Munroe] There are others who have similarly impressive and influential work, but don't acquire this level of personal fame. Fabrice Bellard has taken a different path: he is one of the brightest and most influential programmers of the past 20 years and yet has managed to stay below the radar despite his contributions.

Fabrice was born in Grenoble, France in 1972. He grew up, however, in Montpellier in Southern France. Many computer scientists show exceptional intelligence and interest in related fields, like mathematics or technology, at a young age. Edsger Dijkstra, for example, was the son of a chemist and mathematician. This background had a powerful influence on him. He was interested from an early age in mathematics and physics and proceeded to follow a career in this field.[Dijkstra1] In this case, Fabrice Bellard is no different. While Dijkstra had a partiality to physics and mathematics, Bellard was drawn to electronic devices. His first word was "magnétophone" (tape recorder). It is this interest in electronics that he credits for his dedication to computer science.

As a young child, Fabrice Bellard was able to express his passion for electronic systems through the technology and knowledge available to him as a part of an upper-middle class family. At the age of 9 he began experimenting with programming by writing programs for his TI-59 scientific calculator, one of the first programmable calculators created by Texas Instruments two years prior. The TI-59 featured a Turing complete language, but due to its 10-digit display and alphanumeric-only character set, the language had certain nontrivial restrictions. For example, because there were no symbols or indentation possible, writing C style programs straight on the calculator was very difficult — one had to plan out their program on paper ("code books") before entering them into the TI-59. Nonetheless, the language was far simpler than straight assembly code, featuring more intuitive concepts such as loops. There is no doubt that the difficulty of this language compared to modern languages fostered an early excellence in low-level code, which would appear as a theme in his later work.

Then, when he was eleven, his family purchased their first home computer, the TI-99/4A, also produced by Texas Instruments. The TI-99/4A was the first 16-bit personal computer, although certain limitations at Texas Instruments caused most processes to run through a 16-to-8-bit multiplexer, losing most functionality of the additional bits. The computer included an interpreter for TI BASIC, a language utilized only on the TI-99/4A. Although the interpreter was incompatible with Microsoft BASIC, the popular language of the time, which contributed to Texas Instruments leaving the market for home computer, TI BASIC was simple to learn. It gave the user easy access to commonly used functions, making basic programs easier to write. Each line could only contain one command, similar to the TI-59 Bellard had written programs for previously. This provided an easy transition from programming on a calculator to a full-fledged computer.

While writing in TI BASIC allowed Bellard to develop basic programming skills, the fact that it only saw implementation into that specific machine would have ultimately limited his development. At the age of fifteen, Bellard received his own personal computer: the Amstrad PC1512. More powerful specs, a full qwerty keyboard, and a computer to call his own fueled his passion even more.

Bellard developed what would unexpectedly become his first major success while programming for the PC1512. Due to the limited hard disk space available to him at the time, Bellard saw a need for a method of efficient compression. The result of his effort was LZEXE, the first executable file compression method for personal computers. It allowed executable files to be compressed and subsequently launched without having to explicitly decompress them.

He drew inspiration for the compression method from LZSS, a lossless data compression algorithm implemented in various free, open-source software distributed by Okumura at the time. Utilizing his previous experience writing in low-level languages, he rewrote LZSS from the ground up in 8086 assembly, the machine code of the PC1512, reworking the structure of the program to allow for very fast decompression. Writing in machine code was essential to the solution as it allowed Bellard to keep the decompresser incredibly small, ensuring that it would take up less space than what was gained from compressing the executable file in the first place. Distributing LZEXE to several friends and posting it on various BBS's (the predecessor to modern web forums) LZEXE saw instant success. Having showcased his creativity and programming ability, Bellard continued to develop his skills in preparation for study at the world-renowned French institute, École Polytechnique.

Those familiar with only American-style education will find French higher education to be a striking departure from the norm. French students attend middle school for four years rather than three, and then attend Lycée, high school, for three years. Graduation from high school requires passing the Baccalauréat, a long test similar to the SATs and ACTs. Following graduation from Lycée, students attend one of two schools of higher education: a university or a Grandes école. Unlike American universities, which range from anywhere between 1000 and 50,000 undergraduates, French universities are traditionally very small, with even medium size cities having multiple universities specializing in different fields. Liberal arts schools are uncommon in France; universities typically focus on a single field of study. Just like for most American students, enrollment in universities happens directly after graduation of secondary education.

However, if the student is among the top of his class, he may attempt to enter a grandes écoles instead, a prestigious group of universities that rank highest in France and are among the top universities in the world.[Ecole des Mines de Paris] Prior to enrollment in a grandes écoles, a student must spend between two to three years in a preparation course in order to pass the rigorous entrance examinations, known as Classe Préparatories aux Grandes Écoles (CPGE).

The Grandes Écoles are traditionally incredibly selective, with class sizes ranging from three to five hundred students. Of these, École Polytechnique is viewed as the most prestigious. Once enrolled in X, a common nickname for École Polytechnique, a student spends five years obtaining an Engineer's degree, equivalent to a Masters of Science from American universities.[Ecole]. X further deviates from American universities in that it is a military academy that specializes in engineering. All students must complete a minimum of one year of military service during their education. Unlike American universities that traditionally have students begin to deviate into a specialization by their second year, X stresses a curriculum of breadth, rather than depth. Although the university specializes in engineering, students are required to take classes in sports and humanities. Students spend the first four years studying a wide undergraduate curriculum before their year of military service and then spend the remaining year exclusively studying their specific major. Bellard found the program liberating since the school is highly prominent in technical fields, has large amounts of resources for students, and provides students with a generous stipend when they attend. He also notes that the military service aspect of X was unimportant as military service was required for all French men at the time.

The goal of the X curriculum is to develop critical thinking skills rather than preparation for an engineering occupation. The result, as seen through Fabrice Bellard, is a person who excels in a variety of differing fields. Bellard's contributions to the field of computer science have spanned a vast range of unrelated topics: from digital signals processing to processor emulation to mathematical innovation and everything in between.

His early education in programming and his education at École Polytechnique had a large impact on his overall outlook on the field of computer science. He feels the two most important aspects of computer science are the study of how computers work and the different ways to efficiently use them through the development of languages as well as the study of computation itself.[Gocke and Pizzolato] The first has developed from his original experiences, beginning by programming in a very machine code-like language and slowly expanding to increasingly higher-level languages. His care for the theory of computation stems from his prestigious education. To this day, he feels it is essential that aspiring computer scientists have a deep understanding of the way computers work through assembly language and computer hardware.

A very prominent aspect of Bellard's work is mathematics, particularly that of digital signals processing. In 1995, he made his first foray into the world of numerical algorithms, writing an implementation of Pollard's FFT fast multiplication method in C.

FFT is an acronym for "Fast Fourier Transform," a very common and efficient algorithm in digital signals processing, a recurring theme in Bellard's work. The FFT was originally used as an extremely fast way to compute the Discrete Fourier Transform. The Discrete Fourier Transform is a method of transforming a sampled complex function into a *frequency domain representation.* This representation describes the frequencies that are in the original function, with the precondition that the Discrete Fourier Transform operates on discrete (finite) points.[Weinsstein] As one can immediately see, the implications of this mathematical theory are great — computers cannot operate on continuous functions, so functions such as these, like video and audio, are discretized in a process known as "sampling." The Nyquist-Shannon sampling theorem states that a sampling rate of twice the frequency of the input function results in *perfect information transfer.*[Nyquist 1928; Shannon 1949] This means that computers can accurately represent continuous functions (such as sound), despite having the limitations of having finite data points. Consequently, the Discrete Fourier Transform can use this discrete points to gather information about the original function, such as decomposing the wave into its composite frequencies — information which is invaluable in digital signals processing.

However, researchers soon realized that the FFT might have applications for things other than the discrete Fourier transform. John Pollard, a European mathematician, discovered that a process similar to the one used in FFT might be used for efficient multiplication. The result from his work were algorithms for efficiently multiplying large integer sequences and also for factoring large composite numbers. However, while the mathematical theory was sound, there were few if any practical implementations of Pollard's work in complete code. This is where Bellard came in — in 1995 he completed his work on a practical implementation of Pollard's work in C. He demonstrated his success by calculating the few million digits of π with Salamin's quadratic method, further cementing his status as a master programmer with an agile and adept mathematical mind.[Salamin 1976]

The topic occurs again and again in his work, but may be most notable in one of his first ground breaking successes. On January 20th, 1997, Fabrice Bellard published the fastest known formula to compute the nth binary digit of π on a computer. The algorithm is a variant of the previously fastest known algorithm form 1995 known as the Bailey-Borwein-Plouffe formula. The Bailey-Borwein-Plouffe computed binary digits of pi in $O\left(n^3 (\log n)^3\right)$. Fabrice Bellard improved this algorithm to $O\left(n^2\right)$, thus speeding binary calculation by an impressive 43% and earning him new respect in the mathematical community.

Over the years, many of Bellard's student projects served as the foundation for larger things. For example, in 1998 Bellard started a student project called VReng (Virtual Reality Engine), which is a "distributed 3D application allowing navigation in virtual worlds connected over the Internet using Multicast technology." After building VReng, Bellard noticed that the OpenGL backing available that was software based was far slower than necessary. Instead of modifying the Mesa project (an open source OpenGL implementation that is cross platform), Bellard decided to write a much smaller and faster 3D rasterizer using code from VReng. In 2002, Bellard released TinyGL, a small implementation of a subset of OpenGL. TinyGL is far faster than Mesa or Solaris OpenWin OpenGL, is platform independent and is orders of magnitude smaller than either of them, clocking in at 40kB instead of tens or hundreds of MB. Once again, Bellard demonstrated his considerable skill with writing efficient C code and doing it with far fewer resources than his competitors (Mesa 3D is now maintained by the company Tungsten Graphics, Inc, which has recently been acquired by virtualization leader VMware).

In the year 2000, Fabrice Bellard began one of his most important and widely recognized projects, FFMPEG. FFmpeg was started by Bellard under the pseudonym “Gerard Lantau” and finally brought Bellard’s expertise in telecommunications and digital signals processing to the forefront. FFmpeg is a veritable “Swiss Army knife” of digital video and audio, allowing you to record, stream, and convert between many different formats. Split into several sections, FFmpeg is made up of libavcodec and libavformat, each of which are published under free software libraries. Libavcodec is a collection of audio and video codec libraries, while libavformat provides an audio and video container mux and demux library. Together, these two modules provide a method of parsing and converting digital media between formats.

Description of Bellard’s design first requires a description of the format of digital video and audio. While video and audio are simply data, the digital representation of this data can vary. We call this difference in representation the format of the media. For raw media streams, this is mainly a difference in layout. For example, PCM (Pulse Code Modulation) is a standard digital representation of raw audio data which is sampled, quantized, and then encoded into PCM according to the standard. One may wonder why there are different formats for the same media. The answer mainly lies in the cost of storing and playing certain formats. For example, PCM is usually an “uncompressed” format, meaning that it literally stores all of the input data that it is given to it. For audio and video, this is very costly, especially in terms of data size. In order to reduce the disk space that digital media takes up, a compression algorithm is applied in the form of a different media format, which performs a trade off between disk space and media quality and/or required computational power. The process of writing data into one of these formats is called “encoding,” while the reversal is called “decoding.” Thus, the name of a file or program which contains information on how to do these two things is known as a “codec” (enCODe dECode) for that particular format. However, sometimes many different pieces of media, or “streams,” must each be combined. For example, a movie is comprised of at least two streams — a video stream and an audio stream. The process of mixing many inputs to produce one output is known as “multiplexing,” while the reversal is known as “demultiplexing.” In digital media, multiplexed streams are written into a new format, known as a “container” or “wrapper” format, which contains information only on how the data are stored, but not how they are encoded.

FFMPEG provides a system which combines these two in a single package. Libavcodec is a collection of codecs (hundreds, now) for a variety of multimedia formats, while libavformat provides mux/demux libraries for a variety of different container formats. Together, it can convert from any supported format to an efficient intermediate format devised by Bellard, and then write out the corresponding supported format. Bellard's structure has made the project extremely flexible and extensible, which has allowed it to be incorporated into a variety of different projects across the web. For example, FFMPEG currently include tens, maybe hundreds, of multimedia utilities and players including VLC, which has over a hundred million downloads from the home page alone and is currently being downloaded 5.5 times per second.[vlc]

In 2000 and 2001, shortly after founding the FFMPEG project, Bellard submitted two entries to the International Obfuscated C Code Contest (IOCCC). The IOCCC is a contest for the most creatively obfuscated C. The contest was started in 1984 by Landon Curt Noll and Larry Bassel. Over the years, winning in the IOCCC has become a selective badge of honor among C programmers, as emphasized by the pride many take in their accomplishment.[De Smet] Bellard won the contest twice.

First, for implementing a strict subset C compiler in under 4KB. This entry not only won him the 2000 contest, but also served as the starting point for Bellard's TinyCC project, an ANSI C99 compiler that is orders of magnitude smaller than most other C compilers (e.g. gcc) and still used by many 9 years later. As a demonstration Bellard wrote a utility that uses TinyCC to compile and launch a Linux kernel in less than 15 seconds.[Bellard]

His second winning entry was in 2001 for a 475-byte program which computes and prints the largest known prime number ($2^{6972593} - 1$, which has millions of digits). Conventional methods of calculating prime numbers would be far to slow for Bellard's submission, so he instead adapted a familar algorithm, the Integer Fast Fourier Transform, to print the answer in mere minutes.

Lastly, in 2005 Fabrice Bellard published arguably his most important project yet: QEMU. QEMU is a processor emulator, meaning that it simulates different processor architectures (ISAs) in software, allowing binary programs compiled for one specific architecture to be run on another processor through software emulation. The reason why Bellard was able to this is the Church-Turing thesis, which states indirectly that any Turing-complete language in conjunction with a Universal Turing Machine can emulate any other Turing machine. While a substantial accomplishment on its own, QEMU is not simply a processor emulator, it uses *dynamic translation* to improve performance. As explained in the Usenix paper [Bellard 2005], QEMU uses a novel approach to ISA translation. Instead of translating one instruction at a time, QEMU gathers many instructions together in a process called "chunking," and then translates this chunk as a whole. QEMU then remembers these chunks. Many times there are certain chunks which will occur many times in the code of a program. Instead of taking the time to translate them all separately, QEMU stores the chunks and their native translation, next time simply executing the native translation instead of doing translation a second time. Thus, Bellard invented the first processor emulator that could achieve near native performance in certain instances.

This was not, however, the end of QEMU. In recent years virtualization has become a huge part of computer science. At the moment, some of the largest hardware and software vendors (Sun and Microsoft) have expended great resources to develop their own Virtualization engines (VirtualBox and Hyper-V, respectively), while marketshare leader VMware earned over a $1 billion in revenue in 2008.[VMware, Inc. 2009] In addition to processor emulation, QEMU also supports a series of device interfaces. In this sense, QEMU can be viewed as a hosted virtual machine monitor, wherein it allows the emulation of entire systems inside other operating systems. To this end, because of its speed, it has been included in a variety of other major virtualization technologies, including VirtualBox, Xen, and the Linux Kernel-based Virtual Machine (KVM). As of 2009, Fabrice Bellard is still the lead developer on QEMU.

Bellard believes that FFMPEG and QEMU are the most important projects that he has worked on [Gocke and Pizzolato] — but not for the conventional reason of money. Indeed, one of the most important things to note about Fabrice Bellard is that he has published every one of these major projects under the umbrella of a free software license. What this means is that not only is anyone able to freely download his programs, but they are also able to freely download and modify the source code. The full philosophy behind free software (especially copyleft software) is complex, however Bellard's reasons are not. Bellard is not interested or money or fame (his pseudonyms are evidence of this), but simply in working on projects which are both interesting and useful to him. When asked why he decided to work on such a variety of different subjects, Bellard responded, "It is not a decision. It is just that I tend to be bored by doing always the same thing, so I try to switch of project from time to time. . . ." When he is working on these projects, Bellard wishes to share his work with the world, in the hope that it will be helpful or useful to others. This also factors into his disdain of administrative and communicative tasks. Fabrice Bellard is an accomplished and renowned programmer and he loves to do what he is good at and interested in.

In his view, computer science can be split into two sections. The first is a practical exploration of computers and their uses. The second part is theoretical — the mathematical foundations of computation and what does it mean to be able to "compute" a problem and how fast can we compute that problem. It is not a stretch to say that Fabrice Bellard has pushed the limits of both of these fields and made truly noteworthy success on an astounding range of topics. He emphasizes this in his advice to future computer scientists: gain a grounding in the fundamentals of computer science. For theory he recommends D.E. Knuth, who authored the seminal work in algorithms and data structures, The Art of Computer Programming. In regards to computer systems, he believes every computer scientist should have comprehensive knowledge of the hardware and the low-level software that runs on it. From there, a computer scientists has all the tools he needs to learn and build whatever he wishes.

He also has no intention of stopping yet. His current project is aimed at efficiently using multicore processors in digital signals processing, specifically Software Defined Radios. With his history it will be exciting to see what he produces next.

REFERENCES

Bellard, F. 2005. QEMU, a Fast and Portable Dynamic Translator. *USENIX 2005 Annual Technical Conference, FREENIX Track*, 41–46.

Bellard, F. 2007. Fabrice Bellard's Project Page. `http://bellard.org`.

De Smet, A. Alan's ioccc entry. `http://www.highprogrammer.com/alan/ioccc/`.

Dijkstra1 2003. Edsger Wybe Dijkstra 1920-2002. `http://web.archive.org/web/20041206193322/www.digidome.nl/edsger_wybe_dijkstra.htm`.

Ecole. École Polytechnique. `http://www.polytechnique.edu/page.php?MID=25`.

Ecole des Mines de Paris. Worldwide ranking of universities. `http://www.mines-paristech.fr/Actualites/PR/EMP-ranking.html#7`.

Gocke, A. and Pizzolato, N. An Interview with Fabrice Bellard.

Munroe, R. Donald Knuth. `http://xkcd.com/163/`.

Nyquist, H. Apr. 1928. Certain topics in telegraph transmission theory. *Trans. AIEE 47*, 617–644.

Salamin, E. 1976. Computation of π Using Arithmetic-Geometric Mean. *Math. Comput. 30*, 565–570.

Shannon, C. E. Jan. 1949. Communication in the presence of noise. *Proc Institute of Radio Engineers 37,* 1, 10–21.

Turing1. ACM A.M. Turing Award. `http://awards.acm.org/homepage.cfm?awd=140`.

vlc May 8, 2009. Vlc media player. `http://www.videolan.org/vlc/`.

VMware, Inc. 2009. Company fact sheet. `http://www.vmware.com/files/pdf/fact_sheet_new.pdf`.

Weinsstein, E. W. Discrete fourier transform. From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/DiscreteFourierTransform.html`.