Four Kilobyte Art

Markku Reunanen
Lecturer
markku.reunanen@aalto.fi
Aalto University School of Art and Design

First published in Finnish in *WiderScreen* 2–3/2013 as ”Neljän kilotavun taide”.

Abstract

The so-called 4k intros are real-time audiovisual presentations that fit in four kilobytes. They have mostly been created by the demoscene, a technically-oriented community that emerged in the mid-1980s. In this article I study the history and cultural importance of 4k intros as a marginal form of digital art.

Keywords: 4k intros, algorithmic art, demoscene, programming

Introduction

Two to the power of twelve, 4096 bytes, is a tiny amount of data: approximately as much as a row of pixels on a computer screen or a page of text. These days, when applications and media files consume gigabytes of disk space, it might be hard to think that one could fit something meaningful in just four kilobytes, or that there would even be a need for such compression.

4k intros, real-time audiovisual presentations created by the demoscene, a community formed in the mid-1980s, could be called miniatures of the digital age. In this article I discuss them, on the one hand, as cultural-historical artifacts that reflect the changes of the technological landscape and the demo culture itself and, on the other hand, as creative works of art that have let their authors exhibit their wizardry to others. In addition to a historical overview, I will focus on various tools and approaches that have been used to tackle the challenge of four kilobytes. Most of the discussion will revolve around intros created for IBM PC compatible computers due to

their popularity, and because they illustrate the development of computing power and the graphical capabilities of mainstream computers starting from the early 1990s in the best way.

By definition, a 4k intro is an executable file that is at most 4096 bytes in length, including all the code, graphics and audio needed. In spite of their minuscule size, the best 4k intros are remarkably advanced, featuring several visual effects and music that sounds larger than its size, combined into a tightly synchronized audiovisual presentation (see Fig. 1). Two essential keywords that define the content are *generativity* and *compression*: both the graphics and sound are generated algorithmically and, in addition, the size of the code is optimized with special purpose-built tools.

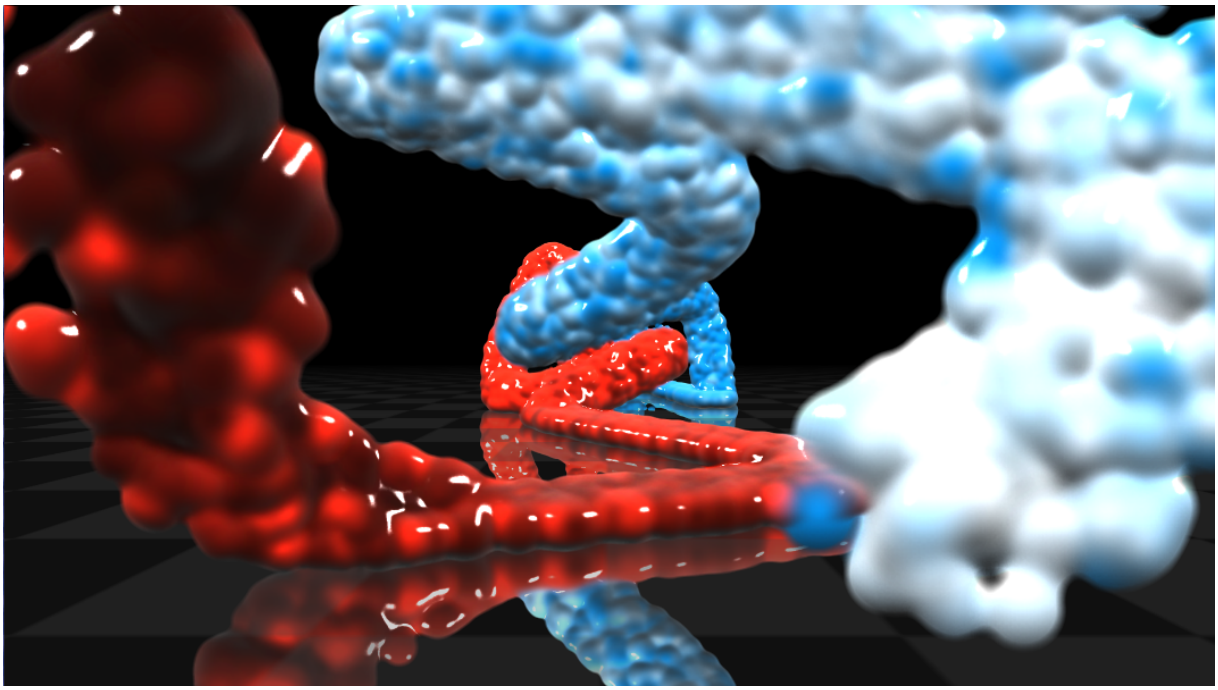

Figure 1. “Nucleophile” by Portal Process and TCB (2008).

Full-blown demos have grown in size year by year, whereas tiny intros have rather taken an opposite direction. Traditional 40k and 64k intros have been followed by 4k and even smaller categories, such as 1k intros, where all the content must fit in 1024 bytes. Even smaller powers of two are common: at *Pouet.net* (http://www.pouet.net/), a popular demoscene site, there are also categories for 512, 256, 128, 64, and finally 32 byte intros. These extremely small productions typically feature only one seemingly impossible visual effect that has been

painstakingly hand-optimized byte by byte. These clearly defined categories also highlight how important it is for the demoscene to classify its works. (Reunanen 2010, 52–57.)

All in all, computer demos are a marginal research topic and, judging by the existing publications, 4k intros are in the margin of the margin. As an example, the two largest demoscene books, *Freax* (Polgar 2005) and *Kunst, Code und Maschine* (Botz 2011) hardly mention the topic at all. In my own licentiate thesis (Reunanen 2010, 52–57) there are a few pages dedicated to size-limited intros, but in general there is little research on the intros. One of the most interesting texts published on the topic is the interview of Sebastian "Minas" Gerlach, published in the *SCEEN* magazine, where the author of several high-ranking 4k intros describes his working methods (SCEEN #2 2007, 72–75). In addition to *Pouet.net*, important source material was found in *IN4K* (http://in4k.northerndragons.ca/), a collection of tools and tips for 4k intro programmers.

My personal connection to 4k intro programming dates back to 2003–2005, when I created three intros with Antti Silvast. The first of them, *Yellow Rose of Texas* (2003), clearly required the most hours, whereas the following two, *Je Regrette* (2004) and *Make It 4k* (2005), were experiments built on the already existing foundation. All three were first released for Linux, after which they were ported to other hardware and software platforms. Some of the ports were made by other enthusiasts in the spirit of open source software. These projects provided me with plenty of hands-on experience on the numerous challenges that limited size poses to audiovisual programming and the tools used.

4k Intros Then and Now

Modest technical features, such as minimal processing power and memory, of the 1960s and 1970s computers severely limited the means available to early computer artists and, therefore, digital art of the time is marked by minimalism. In addition to technical reasons, such minimalism was also a conscious choice: according to new media theorist Lev Manovich, the roots of the esthetic can be traced back to the modernism of the late 19th and early 20th century, which steered all visual arts into a minimalist direction. The first overview of the emerging art form, *Computer Graphics – Computer Art*, was written by Herbert W. Franke (1971). The

pioneering works by artists such as Charles Csuri and John Whitney appear, in spite of their age, somehow familiar when compared to 4k intros; mathematically generated graphics and limited resources produced a similar esthetic 25 years later (cf. Saarikoski 2011).

For today's tiny intro programmer the size limit is a completely arbitrary rule set by the community, but similar constraints found in early video game consoles and home computers were due to their technical features. In their book *Racing the Beam* Nick Montfort and Ian Bogost mention familiar figures about the Atari VCS game console: the 6507 CPU could address only eight kilobytes of memory, many games fit in two kilobytes and the maximum size of a standard cartridge was four kilobytes (Montfort & Bogost 2009, 25–26). Just like with tiny intros, we are dealing with powers of two. In addition to their mathematical properties, such numbers are also a way of understanding the complex internal workings of the computer, which leads to repeating them even in contexts where there are few technical reasons for their use.

The demoscene was preceded by a few years by the cracker scene, which was first characterized by so-called crack screens, static images placed in pirated games. Later on, in the mid-1980s, the screens developed into flashy crack intros. An intro, shown before the actual game started, could be described as a business card of the group that had removed the copy protection and distributed the game. Crack intros served multiple social purposes, such as increasing the status of the group, as well as forming and maintaining the social networks of software pirates. (Polgar 2005, 40–70; Reunanen 2010, 22–23.) Limited storage and memory led, again, to a certain minimalism. In addition, cracked games were compressed in order to save precious disk space (Wasiak 2012). Another example of tiny executables is the so-called BBS intros, ads that were circulated in Bulletin Board Systems (Reunanen 2010, 52). Based on this, I argue that size optimization has been at the very core of the demoscene right from the beginning.

According to *Pouet.net*, the first actual 4k intros were created in the early 1990s. At STNICC, an Atari ST hobbyist meeting held in 1990, there was a 3.5 kilobyte programming competition with a nostalgic title "VIC Times Revisited" (A Commodore VIC-20 reports 3583 bytes of free memory after starting up). Some of the works were games and some intros, which reveals how the practices had not yet settled at that time. One of the participants was British Jeff Minter, better known for his unique llama and camel games. After the STNICC, the category was

practically forgotten for a few years before it became part of competitions (*compos*) held at demo parties.

4k intros turned from a casual curiosity into a relevant competition largely because of Assembly'94, one of the biggest demo parties of the time. Already at the Bush Party of the same year there had been a 4k compo, but Assembly brought intros to the limelight. As was typical of the era, the competitions were split by the platform and, therefore, the category was called "PC 4k". In the results (*Assembly 94 results*, pouet.net) there are only eleven intros, but there were other participants whose works did not make it past the jury. *Stoned* by the German group Dust came out as the winner. The intro features typical demo effects of the time, such as an image rotator, a tunnel and a Mandelbrot fractal, which was often seen in contemporary music videos, too (Fig. 2). The following year the rest of the big parties, The Gathering and The Party, included 4k intros among their competitions, and they became a permanent part of the demoscene canon.

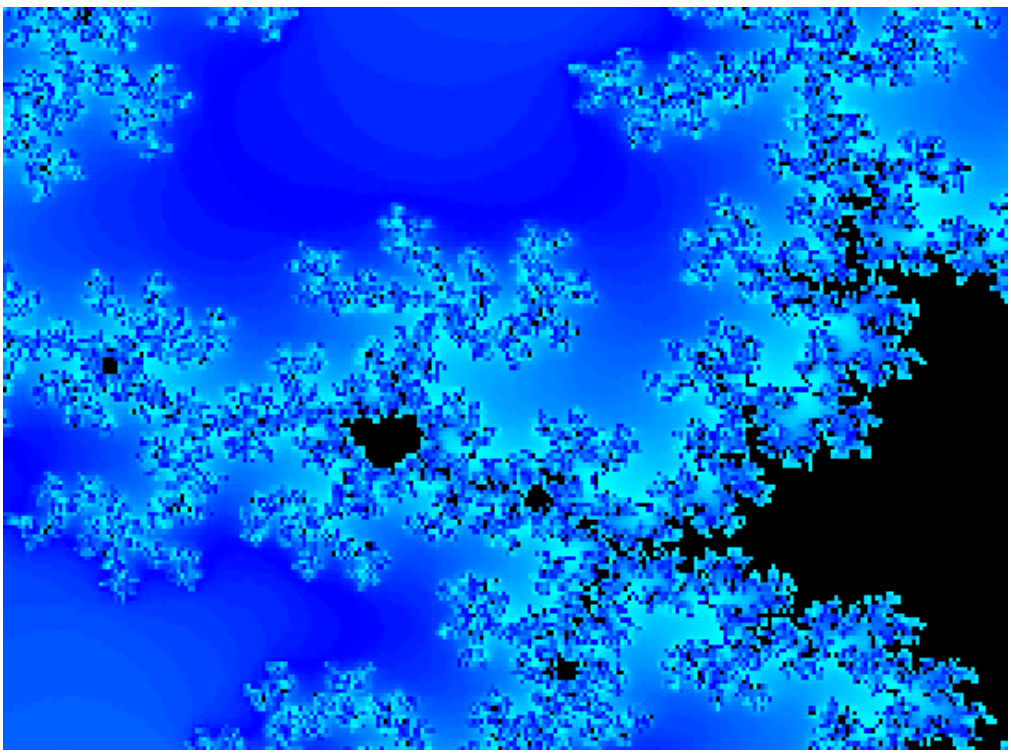

Figure 2. "Stoned" by Dust (1994).

The ever-changing computer market and technological development affect 4k intros, but with a certain delay, since the demoscene does not adopt new technology instantly without criticism.

Reunanen and Silvast (2009) discuss the topic in depth and mention a critical attitude toward mainstream computing and the purported ease brought by new computers as two reasons for the change resistance. Based on the competition results archived at *Pouet.net*, the gradual decay of the Amiga led to the dominance of MS-DOS based intros at big parties toward the end of the 1990s. Separate Amiga and PC competitions were combined into one. At the same time, 4k intros were adopted by retro computer scenes and started appearing for the Commodore 64 and the ZX Spectrum. There had been plenty of small intros for both even earlier, but not as a clearly-defined category.

One of the most fundamental paradigm shifts in the history of the demoscene is the transition from low-level "hardware banging" to system-friendly programming around the beginning of the millennium, fueled by the popularity of Microsoft Windows that did not allow for direct hardware access anymore. The shift was by no means painless and it required several years of accommodation. One important factor was the rise of affordable 3D accelerator cards, since their interesting features could only be used through the programming interfaces provided by new operating systems. (Reunanen 2010, 92–96.) Likewise, other services, such as audio APIs (Application Programming Interface), changed the essence of tiny intros dramatically, as there was no longer a need to program everything from scratch if a suitable component was already offered by the system.

As of this writing, in 2013, 4k intros are still relevant for the demoscene. According to the statistics gathered from *Pouet.net* by Bent Stamnes (2013), approximately a hundred new intros are released every year, and so it seems that they will not disappear anytime soon. The audiovisual quality of 4k intros has climbed so high – "4k is the new 64k" – that the focus seems to be shifting towards the next, even more challenging category, 1k intros, that have not yet been explored in the same depth. For example, at Assembly 2012 there were significantly more works in the 1k competition than in the 4k (*Assembly 2012 results*, pouet.net). In addition to the joy of discovery, it seems likely that the workload involved in the creation of a full-scale 4k intro has become discouraging for authors. Another recent competition category, "procedural graphics", where the aim is to generate an impressive static image in little space, can be seen as another example of the demoscene going back to its roots.

Tiny Effects

It is practically impossible to include media files, such as digital video, sound clips, pre-made 3D models or even static images, in four kilobytes, and so the typical working methods of the new media field do not suit this context. The most important means of creating impressive visual effects is to *generate* them instead: objects, patterns and movement can be created with the creative use of mathematical functions, random numbers and fractals. In the heart of this kind of process is the programmer, whose creativity and skill mostly dictate the quality of the outcome.

When creating the aforementioned 4k intros, we quickly discovered another strategy that was refined already when some of the effects were used for visualizing music at concerts and clubs a few years earlier. The strategy was *parametrization*. In the context of concert visuals parametrization refers to altering the same content progressively so that it is possible to prolong the performance without additional material. Likewise, in the case of 4k intros precious data and code can be reused in multiple ways to provide the viewer an illusion of multiple effects. Typical means for parametrization are, for example, changing the color palette, mirroring the graphics, modifying and copying 3D objects using mathematical formulas, changing the camera angle and filtering the output in various ways. In other words, the content is not hard coded, but its parameters are left open and modified during the execution of the program.

Computational generation of graphics produces a distinctive, recognizable esthetic that is marked by abstraction: organic figures, such as lifelike human models, are tedious to generate. Thus, it is not surprising that most 4k intros do not even aim at replicating real-world objects, but are rather based on abstract forms instead. As a counterexample we could consider the numerous 3D landscapes that may appear very convincing at their best. Other real-world objects and phenomena that lend themselves to algorithmic generation are, for example, regular plants (Prusinkiewicz & Lindenmayer 1990), waves, clouds, buildings and mechanical machines, which have all appeared in both tiny intros and procedural graphics.

In the history of computer graphics there are numerous examples of similar approaches to creating photorealistic images, especially since the 1980s (see Goodman 1987, 102–164; Foley

et al. 1996, color plates 12–41). Karl Sims' animations, such as *Particle Dreams* (1988) and *Panspermia* (1990) are based on algorithmically generated imagery, and their at least indirect effect on demos is easy to notice. Another well-known case of algorithmic art is the works by Laurent Mignonneau and Christa Sommerer, who have brought art and technology together since the early 1990s. The typical division of labor between a programmer and an artist is alien to Mignonneau, Sommerer and the demoscene: the role of code is not hidden, but it is considered as an experimental and creative tool on its own (cf. Mignonneau & Sommerer 2006).

Currently, the most popular 4k intro at *Pouet.net* is *Elevated*, released in 2009 by TBC and rgba. The intro serves as a good example of the algorithmic generation and parametrization of real-time visuals. A virtual camera pans in a believable snowy landscape shown from multiple angles (Fig. 3). There is water in the valleys, clouds in the sky, mist in the air and the sun reflects from the surfaces – all typical means of increasing realism in computer-generated imagery, but this time implemented in a minimal amount of bytes. Music plays in the background and supports the illusion with its echo and wind sounds. Iñigo "iq" Quiles, the other programmer of *Elevated*, discussed the technology and production process of the intro in his presentation at the Function demo event in 2009 (Quilez 2009).

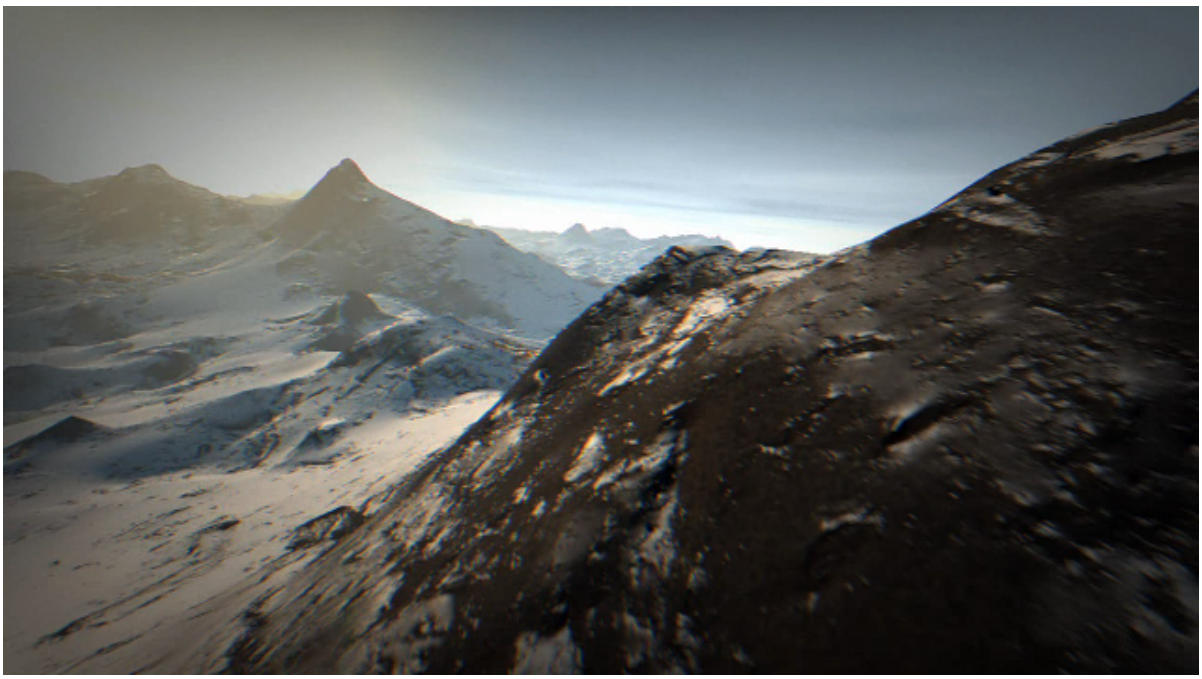

Figure 3. "Elevated" by TBC and rgb (2009).

The development of mainstream hardware and software has undeniably affected the amount and types of content that can be included in 4k intros. Traditional MS-DOS and Amiga intros had to be self-contained, whereas modern PC operating systems are bundled with multiple useful components, such as libraries, fonts and compression software, which take some load off the programmer. As an example, in the case of 3D graphics the difference is radical: back in the day, everything had to be created from scratch, whereas these days equal or better functionality is available through standard API calls. The use of external components has transformed 4k intros and, at times, led to bitter arguments between the traditional do-it-yourself ethic and pushing the genre's limits (*Some thoughts on 4k competition rules*, pouet.net).

Tiny Audio

Most 4k intros were silent until the end of the 1990s and, thus, rather ascetic compared to other demos; the tightly-knit interplay between visuals and audio is the main point of many productions. Still in the Assembly'98 competition rules (*Assembly'98 Official Invitation Text*, ftp.scene.org) the controversial situation was justified as follows:

> NO MUSIC or other sound is allowed (this is because this a coders' competition, not musicians')

Behind this arbitrary constraint was apparently the idea that programmers, musicians and graphic artists should each have one dedicated individual competition, as full-blown demos are usually created by teams. After 2000 music finally started becoming an integral part of 4k intros, which increased the already high requirements of the category even further: in addition to visual effects you had to fit in a tune and a sound player routine.

Creating music in tight space heavily depends on the underlying software and hardware platform. Home computers of the early 1980s typically contained a sound chip with a few channels and different waveforms that would produce recognizable, characteristic sound. In the case of the Commodore Amiga and modern PC compatible computers, sound consists of digital samples, meaning that the waveforms have to be somehow generated first. Anders Carlsson (2010) deals with different sound chips from a composer point-of-view in his MA thesis. Here I will focus

especially on software sound synthesis, even though it is not the only option: some 4k intros have utilized the speech synthesizer or MIDI playback provided by the operating system which, however, often leads to space-efficient but easily recognizable and plain sound.

My own approach in 2003 was also software sound synthesis. *Syna*, a minimal synth written in assembly language, took 1.5 kilobytes with a tune after compression, which was borderline acceptable, since plenty of other content had to be fit in as well. At its core *Syna* features four typical waveforms (square, saw, sine and noise) that can be played back at different frequencies, which is enough for simple beeps. The timbre is augmented by using envelopes that mimic real instrument behavior and a low-pass filter that smoothens the sound (cf. Tolonen et al. 1998). The music produced using these means still appears somewhat flat, so the output is fed to a delay loop echo that creates an illusion of space.

From a musician's point of view, using *Syna* requires technical skill, patience and careful planning, since the composing is done by typing notes to a text file. Having said that, the modest feature set encourages creative problem-solving to overcome the limitations, as demonstrated by musicians that have created music with *Syna*. For example, adding distortion by increasing the volume or recycling the same melody with different instruments were not features that I had consciously implemented, but which emerged from real use and its needs instead.

As of now, the most popular tool used for 4k music creation is *4klang* that was developed by the demogroup Alcatraz. Compared to *Syna*, it features multiple technical improvements, such as waveforms that can be modified and filtered more freely. One of the fundamental principles of *4klang* is that simple building blocks, such as oscillators and filters, can be combined into complex instruments – a similar approach is used in a number of other so-called modular synthesizers. Instead of relying on a monolithic codebase that might contain unneeded features, *4klang* outputs a playback routine that is optimized for the tune at hand and can be directly used by a programmer in an intro. Musicians' workload has been reduced by creating a VSTi plug-in (Fig. 4) that can be used together with practically any common sequencer software. (Zine #14, 2010.)

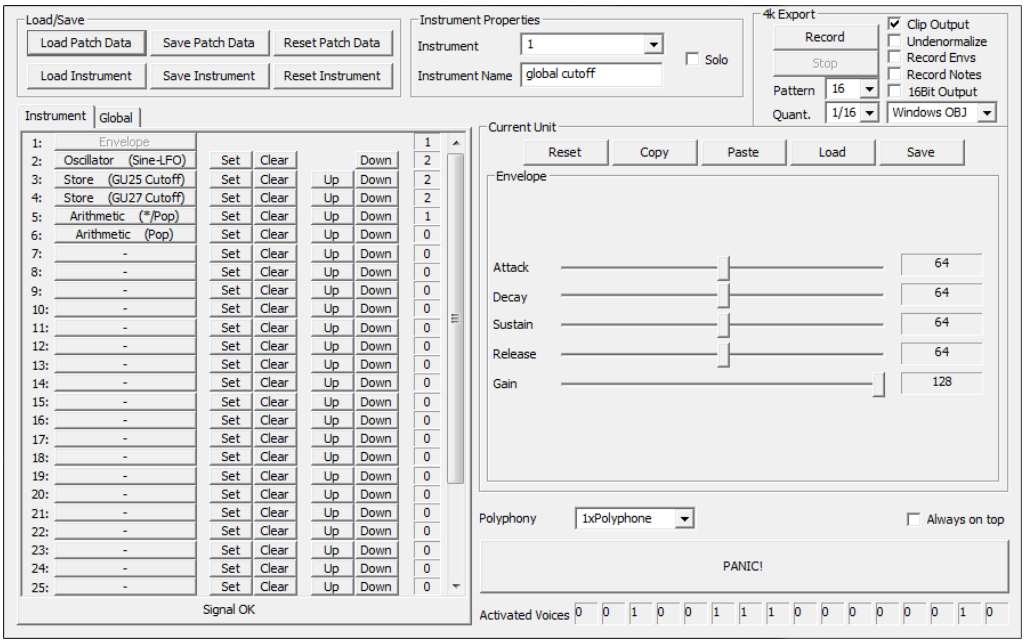

Figure 4. The graphical user interface of 4klang.

The gradual improvement of tools and increasing standards have led to a situation where the best 4k intro tunes sound – as they should – rather massive in spite of their minuscule size of one or two kilobytes. In addition, visual effects and music are tightly coupled in order to create a carefully crafted show for the viewers. The algorithmic generation of instruments and limited space together restrict the realistically possible music genres to different types of repetitive electronic music that tend to be popular in full-scale demos, too. At times there is no music at all, but an ambient soundscape that supports the spatial illusion created by visual means.

Space Optimization Tools

In addition to their audiovisual content, 4k intros have their purely technical side, which needs to be understood when dealing with limited space. The correct use of tools, such as compilers and compression software, saves precious bytes for the actual content and reduces the programmer's workload. Tools and approaches have improved gradually over time; the discoveries made by one programmer have been followed by others, and at the moment the workflow of creating 4k intros is already highly streamlined with its special methods and utilities. Ready-made examples help newcomers to get started by offering a platform on which own experiments can be built.

The executable files of the early 1980s home computers were simple: for example, a COM type

program used in MS-DOS contains pure code with no extra headers, which has ensured its continuous popularity in the smallest intro categories. The primitive COM format dates back to the 1970s, when it was used in the CP/M operating system (see Digital Research 1983). In contrast, modern-day executables are considerably more complex and include various headers that, from an intro programmer's point of view, can be considered as unnecessary overhead. One way of *bumming* bytes is, therefore, reducing the headers to a minimum. *A Whirlwind Tutorial on Creating Really Teensy ELF Executables for Linux* (http://www.muppetlabs.com/~breadbox/software/tiny/teensy.html) by Brian Raiter is an illustrative example of not just saving space, but also the effort that enthusiasts invest in their hobby.

My personal experience with system and code level optimization dates back to 2003, when Linux 4k intros were still in their infancy, much like their Windows counterparts. The beginning of the millennium was a transitional period, when intros had just started appearing on modern PC operating systems, as opposed to the traditional MS-DOS. A significant portion of the effort involved in the creation of *Yellow Rose of Texas* went to pure engineering, such as fine-tuning of compiler parameters, optimizing external library use, and finding suitable methods for code compression. With the two other intros the platform and workflow were already there, so we could mainly focus on the actual content production.

Perhaps the most mystic and hardest to control factor in byte bumming is dealing with compressed code. 4k and 1k intros typically consist of a small stub followed by compressed executable code that is decompressed and run by the stub. Thus, the original uncompressed program can be considerably larger than four kilobytes – for instance, in the case of *Yellow Rose*, 7632 bytes. Already the earliest intros created for the MS-DOS utilized *PKLITE* that can easily be recognized by looking at the beginning of the file. Because of its mathematical nature, the gains obtained by compression are hard to predict and the effect of small changes in the program code affect the size in an almost random manner: changing an individual number or even removing code lines, which would seem like a natural thing to do, may increase the size of the end result. In practice even small differences start to matter when approaching the hard limit of 4096 bytes.

The *IN4K* website (http://in4k.northerndragons.ca/) is a collection of tips and tools suitable for 4k intro programming. *Crinkler*, originally created in 2005 by Rune L.H. Stubbe and Aske Simon Christensen for Windows, serves here as an example of an advanced tool created for the needs of tiny intros. In addition to compression, *Crinkler* optimizes intros in multiple other ways, such as by loading the necessary libraries in a space-efficient manner. (*The Crinkler executable file compressor*, http://www.crinkler.net/.) The version history of the utility also reveals how system-dependent extreme size optimization is: new operating system versions and updates may render current methods useless, which results in unwelcome incompatibility when trying to view old productions.

Conclusion

The study of 4k intros brought up phenomena that are also relevant outside the demoscene. Tiny intros are an example of how early technical limitations have over time turned into a practice and tradition, which is only relevant to the community itself. There is no practical reason to limit executable files to four kilobytes on modern computers – the size counts because of the rules created and maintained by the community. The demo culture is strongly marked by the appreciation of technical skill and creativity, and to fit an impressive production in a few bytes requires both.

The demoscene creates tools, such as *Crinkler* or *4klang*, for its own purposes in the do-it-yourself spirit. Painstakingly crafted utilities are often released for free, so that other members of the community may benefit from them. At the same time, the creation of advanced tools is one more opportunity to show one's skills, and such development has taken the genre forward by allowing for more content in the same space. Another community-oriented trait is the publication of example programs that help others to get started. All in all, we may observe how the development at large has led to the automation of several trivial or tiresome steps, letting the programmer focus on what counts, namely the creative problem-solving tasks involved with audiovisual content production.

The two-decade history of 4k intros mirrors the evolution of computer hardware, software, and the practices of the community during the same period. In the historical perspective, new

operating systems and hardware platforms have been adopted relatively slowly, often with considerable criticism. Especially in the case of old iconic platforms, demosceners have tried to "push them to the limit" and, on the other hand, have not been willing to lose the social capital they have gained – skills and a familiar community. Large demo events, in particular the Finnish Assembly, have had an active role in the construction of practices through their competition rules: which computers and operating systems are allowed, what kind of content is allowed and how a 4k intro is defined in the first place.

In the big picture, tiny intro programming can be compared to other art forms, such as miniature paintings, haiku poems, limericks or ships in a bottle. Their strict rules, which may at first appear arbitrary, require similar problem-solving and focus on the essential – wizardry and creativity often spring from limitations, rather than from a complete freedom of expressive means.

Acknowledgements

I would like to thank the Kone Foundation for supporting the *Kotitietokoneiden aika ja teknologisen harrastuskulttuurin perintö* (Home Computer Era and the Heritage of Technological Hobbyist Culture) project, and Yrjö Fager, Anna Haverinen, Petri Isomäki, Antti Silvast and Mikko Heinonen for their comments.

References

All online sources checked on March 3, 2013.

Magazines

SCEEN #2, 2007.

Zine #14, 2012, disk magazine.

Web Pages

"Assembly 94 results." http://www.pouet.net/results.php?which=7&when=94.

"Assembly'98 Official Invitation Text." July 20 1998. ftp://ftp.scene.org/pub/parties/1998/assembly98/info/asm98inv.txt.

"Assembly 2012 results." http://www.pouet.net/results.php?which=7&when=12.

"Main Page – IN4K." http://in4k.northerndragons.ca/.

"Pouet.net :: your online demoscene resource." http://www.pouet.net/.

Quilez, Iñigo. 2000. "Behind Elevated." http://www.iquilezles.org/www/material/function2009/function2009.pdf.

Raiter, Brian. "A Whirlwind Tutorial on Creating Really Teensy ELF Executables for Linux." http://www.muppetlabs.com/~breadbox/software/tiny/teensy.html.

"Some thoughts on 4k competition rules." http://www.pouet.net/topic.php?which=9093.

Stamnes, Bent. 2013. "(NEW) State of the demoscene: 2012." http://blog.subsquare.com/state-of-the-demoscene-1991-2012-new.

"The Crinkler executable file compressor." http://www.crinkler.net/.

Literature

Botz, Daniel. 2011. *Kunst, Code und Maschine – Die Ästhetik der Computer-Demoszene (Art, Code and Machine – The Aesthetics of the Computer Demoscene)*. Bielefeld: Transcript Verlag.

Carlsson, Anders. 2010. *Power Users and Retro Puppets. A Critical Study of the Methods and Motivations in Chipmusic*. MA thesis. Lund: University of Lund.

Digital Research. 1983. *CP/M Operating System Manual*. Pacific Grove: Digital Research. 3rd edition.

Foley, James D., Andries van Dam, Steven K. Feiner, John F. Hughes, and Richard L. Phillips.

1996/1990. *Introduction to Computer Graphics*. Reading (Mass.): Addison-Wesley.

Franke, Herbert W. 1971. *Computer Graphics – Computer Art*. London: Phaidon Press.

Goodman, Cynthia. 1987. *Digital Visions. Computers and Art*. New York: Harry N. Abrams.

Manovich, Lev. 2007. "Abstraction and Complexity." In *Media Art Histories*, edited by Oliver Grau, 339–354. Cambridge (Mass.): MIT Press.

Mignonneau, Laurent & Christa Sommerer. 2006. "From the Poesy of Programming to Research as Art Form." In *Aesthetic Computing*, edited by Paul A. Fishwick, 169–183. Cambridge (Mass.): MIT Press.

Montfort, Nick, and Ian Bogost. 2009. *Racing the Beam: The Atari Video Computer System*. Cambridge (Mass.): MIT Press.

Polgar, Tamas. 2005. *Freax. The Brief History of the Demoscene. Volume 1*. Winnenden: CSW Verlag.

Prusinkiewicz, Przemyslaw, and Aristid Lindenmayer. 1990. *The Algorithmic Beauty of Plants*. Berlin: Springer.

Reunanen, Markku, and Antti Silvast. 2009. "Demoscene Platforms: A Case Study on the Adoption of Home Computers." In *History of Nordic Computing 2*, edited by John Impagliazzo, Timo Järvi, and Petri Paju, 289–301. Berlin: Springer.

Reunanen, Markku. 2010. *Computer Demos – What Makes Them Tick?* Licentiate thesis. Espoo: Aalto University.

Saarikoski, Petri. 2011. "Kasarisukupolven teknoanimaation perintö (Heritage of the Eighties' Techno Animation)." *WiderScreen* 1–2/2011. http://www.widerscreen.fi/2011-1-2/kasarisukupolven-teknoanimaation-perinto/.

Tolonen, Tero, Vesa Välimäki, and Matti Karjalainen. 1998. *Evaluation of Modern Sound Synthesis Methods*. Report 8. Espoo: Helsinki University of Technology.

Wasiak, Patryk (2012). "'Illegal Guys'. A History of Digital Subcultures in Europe during the 1980s." *Zeithistorische Forschungen/Studies in Contemporary History* 9/2012. http://www.zeithistorische-forschungen.de/site/40209282/default.aspx.