

The Future of Demo Art: The Demoscene in the 2010s

Ville-Matias Heikkilä

a.k.a. viznut/pwp

2010-09-03

1 Introduction

An end of a decade is often regarded as an end of an era. Around the new year 2009-2010, I was thinking a lot about the future of demo art, which I have been involved with since the mid-nineties. The mental processes that led to this essay were also inspired by various events of the 2010s, such as the last Breakpoint party ever, as well as Markku Reunanen's licenciate thesis on the demoscene.[1]

First of all, I want to make it clear that I'm not going to discuss "the death of the scene". It's not even a valid scenario for me. The demo culture is already 25 years old, and during these years, it has shown its ability to adapt to the changes in its technological and cultural surroundings, so it's not very wise to question this ability. Instead, I want to speculate what kind of changes might be taking place during the next ten years. What is the potential of the artform in the 2010s, and what kind of challenges and opportunities is it going to face?

2 After the nineties

After the nineties

Back in the early nineties, demo art still represented the technological cutting edge in what home computers were able to show.

You couldn't download and playback real-life music or movies, and even if you could, the quality was poor and the file sizes prohibitive. It was possible to scan photographs and paintings, but the quality could still be tremendously improved with some skilled hand-pixelling. Demos frequently showed things that other computer programs, such as video games, did not, and this made them hot currency among masses of computer hobbyists far beyond the actual demoscene. As a result, the subculture experienced

a constant influx of young and enthusiastic newcomers who wanted to become kings of computer art.

After the nineties, the traditional weapons of the demoscene became more or less ineffective. Seeing a demo on a computer screen is no longer a unique experience, as demos have the whole corpus of audiovisual culture to compete with. Programming is no longer a fashionable way of expressing creativity, as there is ready-made software easily available for almost any purpose. The massive, diverse hordes of the Internet make you feel small in comparison; the meaning of life is no longer to become a legend, but to sit in your own subcultural corner with an introvert attitude of “you make it, you watch it”. Young and enthusiastic people interested in arts or programming have hundreds of new paths to choose from, and only a few pick the good, old and thorny path of demomaking.

There are many people who miss the “lost days of glory” of their teens. To them, demos have lost their “glamor” and are now becoming more and more irrelevant. I see the things a little bit differently, however.

Consider an alternative history where the glamor was never lost, and the influx of enthusiastic teenagers always remained constant. Year after year, you would have witnessed masses of newbies doing the same mistakes all over again. You would also have noticed that you are “becoming too old for this shit” and looked for a totally different channel for your creativity. The average career of a demo artist would thus have remained quite short, so there would never have been veteran artists with strong and refined visions and thus no chance for the artform to grow up. Therefore, I don't see it as a bad thing at all that demos are no longer as fashionable as they used to be.

There have been many changes in the demo culture during the last ten years. Most of them can be thought of as adaptations to the changing social and technological surroundings, but you can also think about them as belonging to a growth process. As your testosterone levels have lowered, you are no longer as arrogant about your underground trueness as you used to be. As you have gathered more experience and wisdom about life and the world, you can appreciate the diversity around yourself much better than you used to be. More outreach and less fight, you know.

When thinking about the growth process, one should also consider how the relationship between the demoscene and the technology industry has changed. In the eighties, it was all about piracy. In the nineties, people forgot about the piracy and started to dream about careers in software industry. Today, most sceners already have a job, so they have started to regard their freetime activity as a relief from their career rather than as something that would support it.

Especially those who happen to be coders “on both sides” tend to have an urge to separate the two worlds in some way or another by emphasizing the aspects that differentiate democoding from professional programming. You can't be very creative, independent, experimental or low-level in most programming jobs, so you'll want to be that in your artistic endeavours. You may want to choose totally different platforms, methods and technical approaches so that your leisure activity actually

feels like leisure activity.

Thus, although many demosceners work in the software industry, the two worlds seem to be drifting apart. And it is not just because of the separation of work and freetime, but also because of the changes in the industry and the world in general.

Although the complexity of everything in human culture has been steadily increasing for a couple of centuries already, there has been a very dramatic acceleration during the past few decades, especially in technology. This means, among all, that there are more and more prepackaged blackboxes and less and less room for do-it-yourself activities.

Demo art was born in a cultural environment that advocated hobbyist programming and thorough bitwise understanding of one's gear. The technical ambitions of democoders were in complete harmony with the mainstream philosophy of that era's homecomputing. During the following decades, however, the mainstream philosophy degraded from do-it-yourself into passive consumerism, while the demoscene continued to cultivate its original values and attitudes. So, like it or not, demos are now in a "countercultural" zone.

While demos have less and less appeal to the mainstream industry where the "hard-core" niches are gradually disappearing, they are becoming increasingly interesting to all kinds starving artists, grassroots hippies, radical do-it-yourself guys and other "countercultural" people. And if you want your creative work to make any larger-scale sense in the future world, I guess it might be worthwhile to start hang around with these guys as well.

3 Core Demoscene Activity

The changes during the last ten years have made the demoscene activity somewhat vague. In the nineties, you basically made assembly code, pixel graphics and tracker music, and that was it. The scene was the secret cult that maintained the highest technical standards in all of these "underground" forms of creativity. Nowadays, everyone you know uses computers for creativity, some of them even being better than you, and most computer-aided creativity falls under some valid competition category at demoparties. Almost any deviantART user could submit their work in an average graphics compo, sometimes even win it. As almost anything can be a "demoscene production", being a "demoscener" is no longer about what your creative methods are like, but whom you hang around with.

When talking about demo art, it is far too easy to concentrate on the social background ("the scene") instead of the actual substance of the artform and the kind of activity that makes it unique. For the purposes of this essay, I have therefore attempted to extract and define something that I call "Core Demoscene Activity". It is something I regard as the unique essence of demo art, the pulsating heart that gives it its life. All the other creative activities of demo art stem from the core activity, either directly or indirectly.

When defining “Core Demoscene Activity”, we first need to define what it isn’t. The first things to rule out are the social aspects such as participating in demoscene events. These are important in upholding the social network, but they are not vital for the existence of demos. Making demos is supposed to be the reason for attending parties, not the other way around.

The Core Activity is not just “doing creative things with a computer” either. Everyone does it, even your mother. And not even “making non-interactive realtime animations”, as there are other branches of culture that do the same thing – the VJ and machinima communities, for example. Demos do have their own esthetic sensibilities, yes, but we are now looking for something more profound than that.

What is the most essential thing, in my opinion, is the program code. And not just any tame industry-standard code that fulfills some given specifications, but the wild and experimental code that does something that opens up new and unpredicted possibilities. Possibilities that are simply out of the reach of existing software tools. Although there are other areas of computer culture that practise non-compromising hard-core programming, I think the demoscene approach is unique enough to work as a basis of a wholesome definition.

The core activity of the demoscene is very technical. Exploration and novel exploitation of various possible hardware and software platforms. Experimentation with new algorithms, mathematical formulas and novel technical concepts. Stretching the expressive power of the byte. You can remove musicians, graphicicians and conceptual experimenters, but you cannot remove hardcore experimental programming without destroying the essence of demo art.

The values and preferences of demoscene-style programming are very similar to those of traditional hackers (of the MIT tradition). A major difference, however, seems to be that a traditional hacker determines the hack value of a program primarily by looking at the code, while a demo artist primarily looks at the audiovisual output. An ingenious routine alone is not enough; it must also be presented well, so that non-programmers are also able to appreciate the hack value. A lot of effort is put in presentational tweaking in order to maximize the audiovisual impact. This relationship between code and presentation is another unique thing in demo art.

Here is a short and somewhat idealized definition of “Core Demoscene Activity”:

- Core Demoscene Activity is the activity that leads to the discovery of new techniques to be used in demo art.
- Everything in Core Demoscene Activity needs to directly or indirectly support the discovery of new kind of audiovisual output. Either something not seen on your platform before, or something not seen anywhere before.
- The exploration should ideally concentrate on things that are beyond the reach of existing software tools, libraries or de-facto standard methods. This usually requires a do-it-yourself approach that starts from the lowest available level of abstraction.

- General-purpose solutions or reusable code are never required on this level, so they should not interfere with the research. Rewrite from scratch if necessary.

Of course, the core activity alone is not enough, as the new discoveries need to be incorporated in actual productions, which also often include a lot of content created with non-programmatical methods. So, here is a four-level scheme that classifies the various creative activities of demo art based on their methodological distance from the “core”. Graphically, this could be presented as nested circles. Note that the scheme is not supposed to be interpreted as a hierarchy of “eliteness” or “trueness”; it is just one possible way of talking about things.

- **First Circle / Core Demoscene Activity:** Hardcore experimental programming. Discovery of new techniques, algorithms, formulas, theories, etc. which are put in use on the Second Circle.
- **Second Circle Activity:** Application-level programming. Demo composition, presentational tweaking of effect code, content creation via programming, development of specialized content creation tools (trackers, demomakers, softsynths), etc.
- **Third Circle Activity:** Content creation with experimental, specialized and “highly non-standard” tools. Musical composition with trackers, custom softsynths or chip music software; pixel and character graphics; custom content creation software (such as demomakers), etc.
- **Fourth Circle Activity:** Content creation with “industry-standard tools” including high-profile software and “real-life” instruments. Most of the bitmap graphics, 3D modelling and music in modern “full-size” demos have been created with fourth-circle techniques. Design/storyboard work also falls in the fourth circle. Blends rather seamlessly with mainstream computer-aided creativity.

It should be noted that the experimental or even “avant-garde” attitude present in the Core Activity can also be found on the other levels. This also makes the Fourth Circle important: while it is possible to do conceptual experimentation on any level, general-purpose industry-standard tools are often the best choices when trying out a random non-technical idea.

The four-circle scheme seems to be applicable to some other forms of digital art as well. In the autumn 2009, the discovery of Mandelbulb, an outstanding 3D variant of the classic Mandelbrot set, inspired me look into the fractal art community. The mathematical experimentation that led to the discovery of the Mandelbulb formula was definitely a kind of “core activity”. Some time later, an “easy-to-use” rendering tool called “Mandelbulber” was released to the community in what I would classify as “second-circle” activity. The availability of such a tool made it possible for the non-programmers of the community to use the newly discovered mathematical structure in their art in activities that would fall on the third and fourth circles.

4 Is it only about demos?

The artistic production central to demo culture is, obviously, the demo. According to the current mainstream definition, a demo is a stand-alone computer program that shows an audiovisual presentation, a couple of minutes long, using real-time rendering. It remains exactly the same from run to run, and you can't interact with it. But is this all? Is there something that demo artists can give to the world besides demos?

I'm asking this for a reason. The whole idea of a demo, defined in this way, sounds somewhat redundant to laymen. What is the point in emphasizing real-time rendering in something that might just as well be a prerendered video? Isn't it kind of wasteful to use a clever technical discovery to only show a fixed set of special cases? In order to let the jewels of Core Demoscene Activity shine in their full splendor, there should be a larger scale of equally glorified ways of demonstrating them. Such as interactive art. Or dynamic non-interactive. Maybe games. Virtual toys. Creative toys or games. Creative tools. Or something in the vast gray areas between the previously-mentioned categories.

The idea of a "non-interactive realtime show" is, of course, tightly knit with the standard format of demoparty competitions. Demos are optimized for a single screening for a large audience, and it is therefore preferable that you can fix as many things as possible beforehand. Realtime rendering wasn't enforced as a rule until video playback capabilities of home computers had become decent enough to be regarded as a threat to the dominance of hardcore program code.

But it's not all about party screenings. There are many other types of venues in the world, and there are, for example, people who still actually bother to download demoscene productions for watching at home. These people may even desire more from their downloaded programs than just a couple of minutes of entertainment. There may be spectators who, for example, would like to create their own art with the methods used in the demo. Of the categories mentioned before, I would therefore like to elevate creative toys and tools to a special position.

It is proven that creative tools originating in the demoscene may give rise to completely new creative subcultures. Take trackers, for example. The PC tracker scene of the nineties was much wider than the demoscene which gave it the tools to work with. In the vast mosaic of today's Internet world, there is room for all kinds of niches. Release a sufficiently interesting creative tool, and with some luck, you'll inspire a bunch of freaks to find their own preferred means of creativity. The freaks may even form a tight-knit community around your tool and raise you to a kind of legend status you can't achieve with demo compo victories alone.

Back in the testosterone-filled days, you frowned upon those who used certain creative tools without understanding their deep technicalities. But nowadays, you may already realize the importance of "laymen" exploring the expressive possibilities of your ingenious routine or engine. If you are turned off by the fact that "everyone" is able to (ab)use your technical idea, you should move on and invent an even bet-

ter one. The Core Activity is about continuous pushing of boundaries, not about jealously dwelling in your invention as long as you can.

Now, is there a risk that the demoscene will “bland out” if “non-demo productions” will receive as much praise and glory as the “actual” demos? I don’t think so. To me, what defines the demoscene is the Core Activity and not the “realtime non-interactive production”. As long as you nurture the hardcore spirit, it manifests itself in all kinds of things you produce, regardless of how static, realtime, bouncy or cubistic they are.

5 Parties and social networks

An important staple in keeping demo culture alive is the demoparty. It both strengthens the social bonds and motivates the people involved to create and release new material. Of course, extensive remote communication has always been there, but flesh-and-blood meetings are the ones that strengthen the relationships to span years and decades.

As there are so many people who have deeply dedicated themselves to demo art for so many years, I am convinced that there will be demoscene parties in 2020 as well. Only a global disaster of an apocalyptic scale can stop them from taking place.

While pure insider parties may be enough for keeping the demoscene alive, they are not enough for keeping it strong and vital. There is a need for fruitful contacts between demo artists and other relevant people, such as other kinds of artists and potential newcomers. High-profile mainstream computer parties, such as Assembly, have been succesful in establishing these contacts in the past, but much of the potential for success has faded out during the last decade, as an average demo artist has less and less in common with an average Assembly visitor.

I think it is increasingly vital for demo artists to actively establish connections with other islets of creative culture they can relate to. The other high-profile Finnish demoparty, Alternative Party, has been very adventurous in this area. Street and museum exhibitions that bring demo art to “random” people may be fruitful as well, even in surprising ways. When looking for contacts, restricting oneself to “geeky subcultures” is not very relevant anymore, as everyone uses computers and digital storage formats nowadays, and being creative with them – even in ways relevant to demo art – does not require unusual levels of technological obsession.

Crosscultural contacts, in general, have the potential of giving demosceners more room to breath. While a typical demoparty environment strongly encourages a specific type of artwork (i.e. demos), other cultural contexts may inspire demo artists to create totally different kinds of artifacts. I’m also sure that many experimental artists would be happy to try out some unique creative tools that the demo community may be able to give to them, so the collaboration may work well in both directions. Real and virtual platforms

The relationship between demo artists and computing platforms has changed dra-

matically during the past ten years. Back in the nineties, you had a limited number of supported platforms with separate scenes and competitions. Nowadays, you can choose nearly any hardware or software platform you like, and different platforms often share the same competitions. Due to the existence of decent emulators and easy video captures, the scene is no longer divided by gear ownership. Anyone can watch demos from any platform or even try to develop for almost any platform without owning the real hardware. Also, as the average age of demosceners has risen, platform fanboyism is now far less common.

The freedom is not as full as it could be, however. There are people who build their own demo hardware and they are praised for this, but what about creating your own entirely software-based “virtual platforms”? Most demo artists don’t even think about this idea. Of course, there are many coders who have created ad-hoc integrated virtual machines in order to, for example, improve the code density in 4K demos, but “actual” platforms are still something that need to be defined by the industry. In the past, it even required quite a tedious process until a new hardware platform became accepted by the community.

So, why would we need virtual platforms in the first place? Let’s talk about the expressive power of chip music, for example. There are various historical soundchips that have different sets of features and limitations, and after using several of them, a musician may not be completely satisfied by any single chip. Instead, he or she may imagine a “perfect soundchip” that has the exact combination of features and limitations that inspires him/her in the best possible way. It may be a slight improvement of a favorite chip or a completely new design. Still, someone who composes for a virtual chip rather than an authentic historical chip may not be regarded as very “true”. There is still certain history-fetishism that discourages this kind of activity. In my earlier essay about Computationally Minimal Art, however, I expressed my belief that the historical timeline will lose its meaning in the near future.[2] This will make “non-historical experimentation” more acceptable.

It is already relatively acceptable to run demos with emulators instead of real hardware, even in competitions, so I think it’s only a matter of time that completely virtual platforms (or “fake emulators”) become common. For many, this will be a blessing. Artists will be happier and more productive working with instruments that lack the design imperfections they used to hate, and the audience will be happier as it gets new kinds of esthetic forms to appreciate.

Virtual platforms may also introduce new problems, however. One of them is that none of the achieved technical feats can be appreciated if the platform is not well-understood by the audience: if you participate in a 256-byte competition with a demo written for your own separate virtual machine, it is always relevant for the spectator to assume that you have cheated by transferring logic from the demo code into the virtual machine implementation. You could, for example, put an entire music synthesizer in your virtual machine and just use a couple of bytes in the demo code to drive it. If you want your technical feats appreciated, the platform needs to pass some kind of a community acceptance process beforehand.

On the other hand, virtual platforms may eventually become mandatory for certain

technical feats. It is already difficult in modern operating systems, for example, to create very small executables that access the graphics and sound hardware. As the platforms “improve”, it may eventually become impossible to do certain things from within, say, a four-kilobyte executable. In cases like this, the community may need to solve the problem with a commonly accepted “virtual platform”, i.e. a loader that allows running executables given in a format that has less overhead. Such a loader may also be used for fixing various compatibility problems that are certain to arise when new versions of operating systems come out.

Within some years from now, we may have a plethora of virtual machines attempting to represent “the ultimate demo platform”. There will be a need for classifying these machines and deciding about their validity in various technical competitions. Despite all the possible problems and controversies they are going to introduce, I’m going to embrace their arrival.

But what about actual hardware platforms, then? I guess that there won’t be as much difference by 2020 anymore. FPGA implementations of classic hardware have already been available for several years, and I assume it won’t take long until it will be common to synthesize both emulators and physical hardware from the same source code. Once we reach the point that it is easy for anyone to use a printer-type device to produce a piece of hardware from a downloadable file, I don’t think it’ll really matter so much to anyone whether something is running virtually or physically.

Regarding the next decade of the mainstream hardware industry, I think the infamous Moore’s law makes it all quite predictable and obvious: things that were not previously possible in real time will be easy to do in real time. There will be smaller video projectors and all that. Mobile platforms will be as powerful as today’s high-end PCs, so you won’t be able to get “oldschool kicks” from them anymore. If you want such kicks from an emerging technology, you won’t have many niches left; conductive ink may be one of the last possibilities. Before 2020, your local grocery store will probably be selling milk in packages that have ink-based circuits displaying animations, and before that happens, I’m sure that the demoscene will be having lots of fun with the technology.

6 Paths of initiation

It is already a commonly accepted view that the demoscene needs newcomers to remain vital, and that they need to be actively recruited since the influx is no longer as overwhelming as it used to be. This view represents a dramatic change from the underground-elitist attitudes of the nineties, when potential newcomers were often forced thru a tight social filter that was supposed to separate the gifted individuals from the “lamers”. Requiring guidance was a definitive sign of weakness; if you couldn’t figure out the path of initiation on your own, no one was going to help you. You simply got stuck in the filter and never got in.

According to my experiences, it is not very difficult to get people interested in demo

art as long as you manage to pull the right strings. It is also relatively easy to get them participate in demoscene events. But getting them involved in the various creative activities is a much more complex task, especially when talking about the inner-circle activities that require programming. It is not about a lack of will or determination but more like about uncertainty about how to get started.

A lot of consideration should be put in the paths of initiation during the following decade. Instead of generalizing from their own past experiences, recruiters should listen to the stories of the recent newcomers. What kind of paths have they taken? What kind of niches have they found relevant? What have been the most difficult challenges in getting involved? Success stories and failure stories should both be listened to.

I'm now going to present some of my own ideas and observations about how democoder initiation works in today's world and how it does not. These are all based on my personal experiences with recent newcomers and not on any objective research, so feel free to disagree.

First, I want to outline my own theory about programming pedagogy. This is something I regard as a meaningful “hands-on” path for hobbyist programmers in general, not only for aspiring democoders. Lazy academic students (whose minds get “mutilated beyond recovery” by a careless choice of first language) may prefer a more theoretical route, but this three-phase model is something I have witnessed to work even for the young and the practical-minded, from one decade to another.

- First phase: Toy Language. It should have an easy learning curve and reward your efforts as soon as possible. It should encourage you to experiment and gradually give you the first hints of a programming mindset. Languages such as BASIC and HTML+PHP have been popular in this phase among actual hobbyists.
- Second phase: Assembly Language. While your toy language had a lot of different building blocks, you now have to get along with a limited selection. This immerses you into a “virtual world” where every individual choice you make has a tangible meaning. You may even start counting bytes or clock cycles, especially if you chose a somewhat restricted platform.
- Third phase: High Level Language. After working on the lowest level of abstraction, you now have the capacity for understanding the higher ones. The structures you see in C or Java code are abstractions of the kind of structures you built from your “Lego blocks” during the previous phase. You now understand why abstractions are important, and you may also eventually begin to understand the purposes of different higher-level programming techniques and conventions.

Based on this theory, I think it is a horrible mistake to recommend the modern PC platform (with Win32, DirectX/OpenGL, C++ and so on) to an aspiring democoder who doesn't have in-depth prior knowledge about programming. Even though it

might be easy to get “outstanding” visual results with a relative ease, the programmer may become frustrated by his or her vague understanding of how and why their programs work.

The new democoders I know, even the youngest ones, have almost invariably tried out assembly programming in a constrained environment at some point of their path, even if they have eventually chosen another niche. 8-bit platforms such as C-64 or NES, usually via emulator, have been popular choices for “first hardcore coding”. Sizecoding on MS-DOS has also been quite common.

Not everyone has the mindset for learning an actual “oldschool platform” on their own, however. I therefore think it might be useful to develop an “educational demoscene platform” that is easy to learn, simple in structure, fun to experiment with and “hardcore” enough for promoting a proper attitude. It might even be worthwhile to incorporate the platform in some kind of a game that motivates the player to go thru varying “challenges”. Putting the game online and binding it to a social networking site may also motivate some people quite a lot and give the project some additional visibility.

7 Conclusion

We have now covered many different aspects of the future of demo art in the 2010s, and it is now the time to summarize. If we crystallize the prognosis to a single word, “diversity” might be a good choice.

It indeed seems that the diversity in what demo artists produce will continue to increase in all areas. There will be more platforms available, many of them designed by the artists themselves. There will be more alternatives to the traditional realtime non-interactive demo, especially via the various “new” venues provided by “crosscultural contacts”. And I’m sure that the range of conceptual and esthetic experimentation will broaden as well.

Back in the nineties, most demo artists were “playing the same game”, with the same rules with relatively similar goals. After that, the challenges became much more individual, with different artists finding their very own niches to operate in. There are still “major categories” today, but as the new decade continues, they will have less and less meaning compared to the more individual quests. This may also reduce the competitive aspect of the demo culture: as everyone is playing their own separate game, it is no longer possible to compare the players. Perhaps, at some point of time, someone will even question the validity of the traditional compo format.

Another keyword for the next decade could be “openness”. It will show both in the increased outreach and “crossculturality”. There will be an increasing amount of demo artists who operate in other contexts besides the good old demoscene, and perhaps there will also be more and more “outsiders” who want to try out the “demoscene way” for a chance, without intentions of becoming more integral members of the subculture.

In the nineties, many in the scene were dreaming about careers in the video game industry. After that, there have been similar dreams about the art world: gaining acceptance, perhaps even becoming professional artists. The dreams about the video game industry came true for many, so I'm convinced that the dreams about the art world will come true as well.

References

- [1] Markku Reunanen. *Computer Demos—What Makes Them Tick?* Helsinki: Aalto University School of Science and Technology, 2010-04-23. <http://www.kameli.net/demoresearch2/reunanen-licthesis.pdf>
- [2] Ville-Matias Heikkilä a.k.a. Viznut/PWP. *Defining Computationally Minimal Art*. Online essay, 2010-03-15. <http://www.pelulamu.net/countercomplex/computationally-minimal-art/>