phX

The making-of

Written by **NoRecess / Condense**
http://norecess.cpcscene.net
March 2018

SPOILER ALERT:
This is a making-of. Numerous screens are revealed.
Please watch the demo first before reading this document !

Overview

phX is a demo for **Amstrad CPC** that initially took 2 years of R&D, then 3 years of development. This article, written by **NoRecess/Condense**, provides an overview of the different development steps that allowed this production to reach the state of release.

phX has been created by **Ced / Condense** (graphics), **Factor6** (audio) and **NoRecess / Condense** (programming). It also makes use of **Arnoldemu**'s loader.

Since this is a major work for us, we thought writing a document describing development notes may be interesting to extend the "phX's experience" !

Many parts in this making-of may use some technical terms - an high-level overview has been set as priority here. Feel free to contact me personally if you are interested for more details on a specific topic.

The parts are not treated by order of appearance in the demo, but treated by order of implementation (cf. historical view).

2011-2013 : the Phortem years

Our previous demo, **Phortem**, impacted in some ways **phX**. This production took 2 years of development (from 2011 to 2013) and gave us (**Ced** and me, **NoRecess**, the "core" members of Condense) good habits regarding development of demos in general.

With that production, we learnt that it was completely acceptable to spend an incredible amount of time on a single production, instead of releasing multiple smaller demos. The resulting impact on the audience was higher than any other productions we did before.

Also, we got a great amount of feedbacks when releasing it. We learnt a lot from those critics, and understood it was necessary to apply some changes for future productions. **Phortem** was initially presented as a *demo*, but it was received as a *slide-show*. While being an honour for **Ced** (the **Condense**'s graphical artist), it was clear than the programming-side was the weak point. While good, it ignored video-chip's features and essentially relied on the Z80 CPU to render all the visual data. We had to bounce on that by providing a faster execution of the demo.

2013-2015 : Research & Development phase

After releasing a massive demo, I think taking a large pause for several years is generally acceptable. But for us, it was the opportunity to quietly improve on the technical side.

Phortem was judged *slow* for two reasons: it had to load a massive amount of data, and the presented effects were too slow to process while some parts of the monitor were left unused. We had to improve on those two aspects.

Data packer

One of the primary focus was to work on a new **data packer**. The initial idea was to improve over the **BitBuster** Z80's unpack routine. The original implementation allowed unpacking with a single `CALL` instruction. While useful, this did not allow streaming - cf. can't unpack a given amount of bytes per `CALL` instruction. Plus, the execution time of the unpack routine was completely random - sometimes it was fast to unpack, sometimes too slow, depending of the nature of the data. I needed to be able to stabilize the execution time of that routine in order to make 50hz-animated demos covering all the visible parts of the monitor (without waiting after VSYNC, cf. like in **DTC** demo).

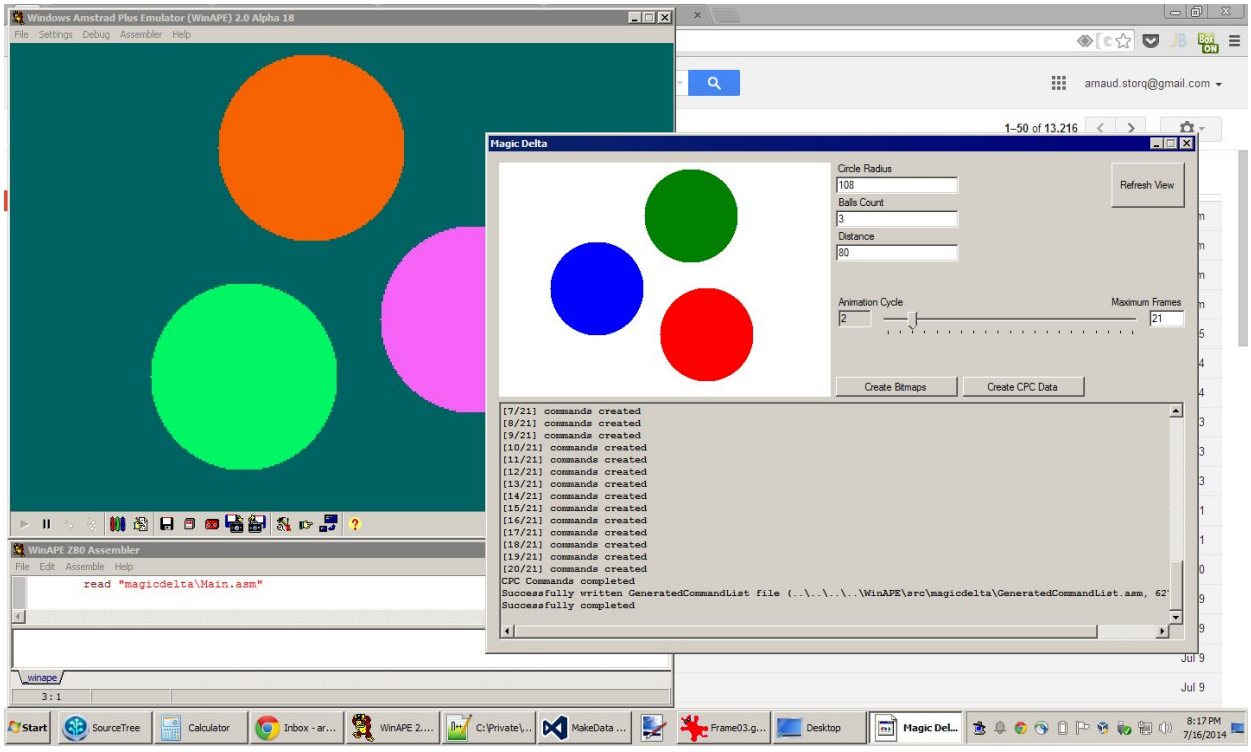
Dealing with current BitBuster's implementation was a no-go. With too much changes to bring, I would have probably spent more time on that than rewriting our own solution. Inspired by BitBuster, I implemented a LZW algorithm that was particularly suitable for the Z80. BitBuster relies on a 12-bit index ; I decided to stick instead with a 8-bit index - providing no bit manipulation ! As a consequence, our packer data does not pack very well ; but it's blazing fast to unpack. The streamable data feature was also implemented as a initial requirement.

CRTC video-chip programming

Next came the programming of the Amstrad CPC video-chip, the **CRTC**. In **Phortem**, the Z80 CPU handled all graphical operations. This resulted per-pixel based effects, but slow processing. Using the CRTC allow scanline-based effects, but enables fast processing. This was the way to go.
When I was a teenager, I had some knowledge based on empirical experiments / poor technical documentation. I did not wish to take similar path once again for **phX**. It was time to be mentored by someone experimented.

How lucky I was ! **Overflow / Logon System**, my long time friend, was skilled at CRTC programming (*S&KOH* anyone? :). He took a great amount of his personal time to explain me the basics. The strategy was to create a course made of different exercises. For each lessons, new CRTC features were briefly discussed, a subject was presented (cf. “please scroll down the screen till it disappears, but you need to keep a perfectly stabilized screen between each VSYNC!”). The corresponding implementation had to be provided by me, taking usually few days. We covered almost everything: hardware scrollings, screen splitting and such.
One advice here to everyone willing to learn CRTC programming: use an emulator (recommended: WinAPE) to monitor the internal CRTC counters varying over time, this is the only way to understand the behaviours (the real hardware does not permit this).

Once basic knowledge about CRTC was acquired, it was time to experiment ! The initial effects were not impressive at first ; previews were continuously shown and reception was important here. If **phX** reached its degree of quality at release date, there is no doubt this was due to the tenacity of **Overflow** on that. He always tried to push the CPC to its limits with his comments, cf. “great FX ! but please can you try adding rasters in background ?”. I never tried to argue ; instead I accepted the criticism and made progress. With time, I was able to predict and reach **Overflow**’s expectations, to the point I actually imposed to myself a certain level of acceptance before showing a preview. Thank you man.



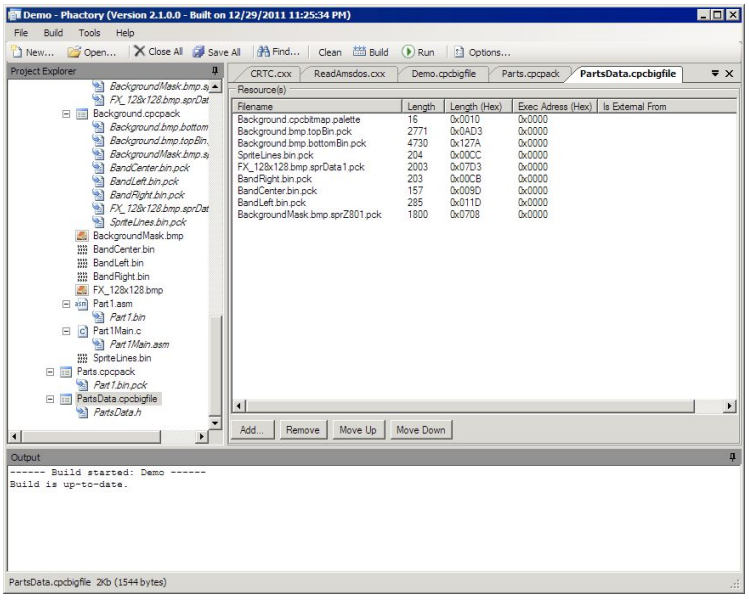
First stabilized screen using line-splitting using WinAPE and a custom tool to generate animation (July 2014)

Audio player

Last thing I experimented with was to create an audio player.
This was the result of fun discussions with **Overflow**, which already had experience with modern audio players for **Yap** and **IO**. I initially relied on the streamable data unpacking routine I just created. I experimented the use of 14 buffers with length of 256 bytes as first introduced by **Madram** in its great AY player, early 2000s. In the making-of of the **Still Rising** demo, I kept in mind how happy **Hicks** was to get a 15 raster-lines player. Mine is 14 raster-lines ;-) but with a twist: I don't rely anymore on the 14 buffers, I simply apply a "register diff" between each frames (along with some data optimizations such as repeat management etc). It takes more data to store the music than I expected, but it's compensated by the fact 14 buffers are not necessary. If you wonder why this is so important to get a stable audio player that is as fast as possible: the faster it is to process, the more CPU time is available for the effects.
Note that the audio player is actually split in two distinct parts, both being approx. 7 lines: a `StreamData` part that prepares the audio data for a given frame, and a `SendRegisters` allowing the audio chip to make some noise. I found some convenience to stream data in the top invisible part of the monitor, and send registers at the bottom invisible part of the monitor. It was easily integrated into the **Arnoldemu**'s musical track-loader : I just had to switch from original **Madram**'s player to my implementation.

Development pipeline

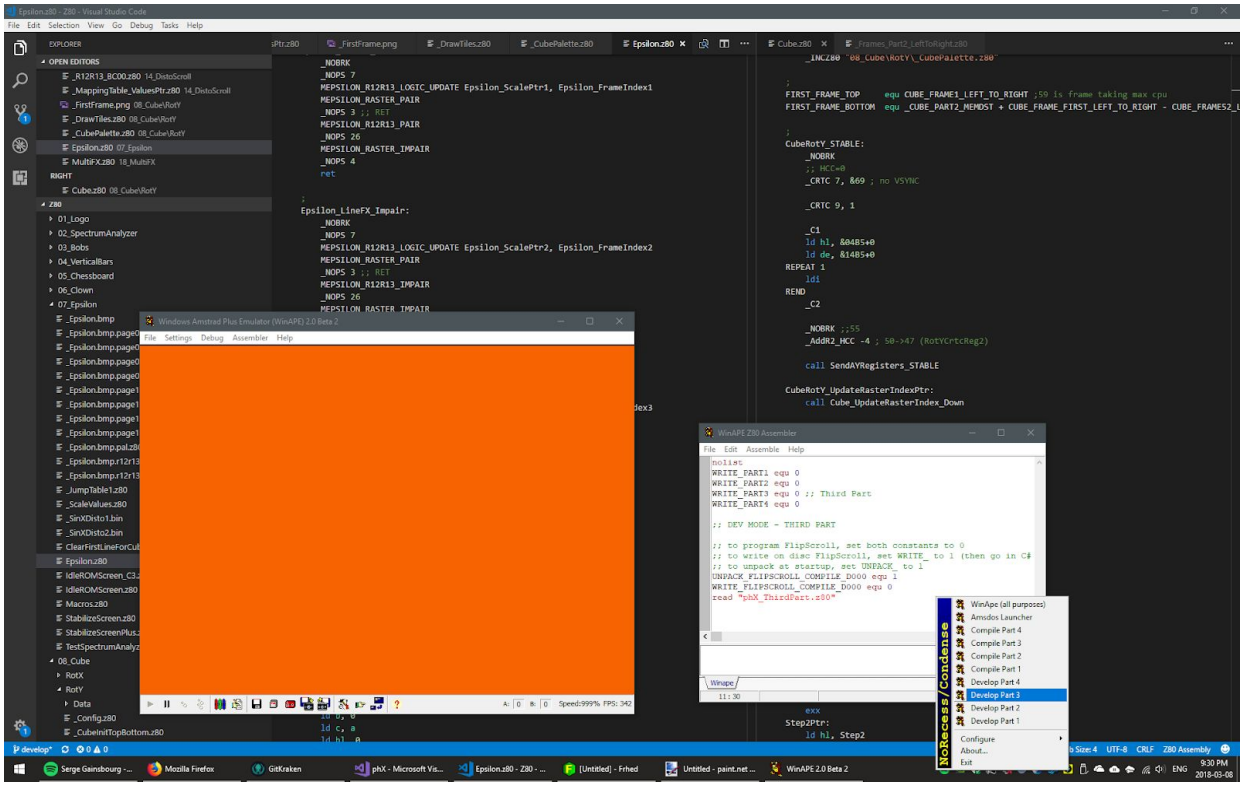
Finally, I completely revisited my workflow. With **Phortem**, I created my own IDE for the demo (called **Phactory**). The effort featured a nice user-interface to implement source-code, edit settings, compilations were made using a nice dependency graph (that was fun to implement !). I initially thought about creating a universal tool for Amstrad CPC development… but it completely distracted me from making the demo itself.



***Phactory**, a custom IDE I wrote and maintained through 2010 to 2013. **Pheelone**, **Phreaks**, **Phortem** and the **HxC Manager V2-V3** were written with it !*

With **phX**, I took a simpler (not cleaner) approach. I created a pipeline where I could focus on the demo, without the wish to continuously improve the pipeline itself.

A command-line tool (written in C# using **Microsoft Visual Studio**) generates all the data of the demo. Everything is hardcoded (relative file paths, constant values…) and I relied on my custom framework providing image manipulation (with palette), sin-curve generation, packing and the likes.



Windows 10, Microsoft Visual Studio, VS Code and WinAPE.

I also used **Microsoft VS Code** as Z80 text editor. I highly recommend it for assembly programming, especially for editing the same Z80 file through multiple vertical panes - this is particularly well suited since assembly source-files usually features small width but lots of vertical lines. I also used **Notepad++** at some point for source editing.

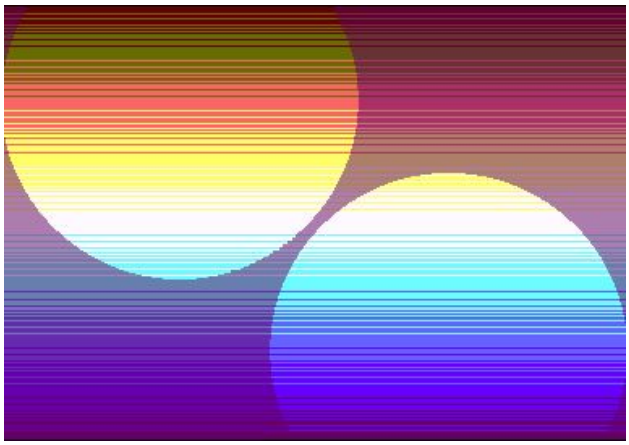
For emulation / compiling / debugging, I relied on the excellent **WinAPE** from Richard Wilson. The fact it's closed source is sad, but it remains a pleasant environment to develop with.

But I continuously reconsider my development workflows. The new **RASM** assembler from **Roudoudou** looks mature enough now, and **Offset**'s **ACE** emulator is really accurate (I tested it with success)...

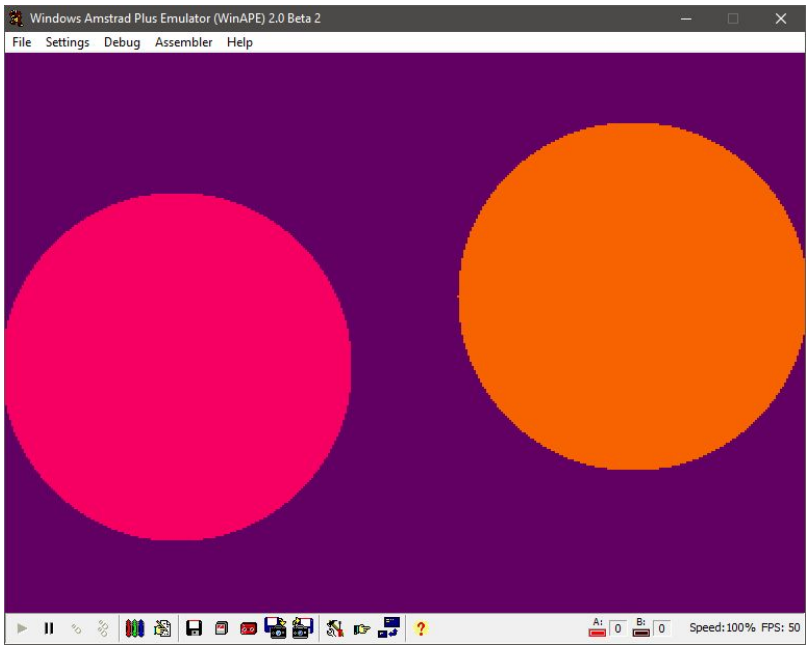
2015-2016 : from beginning to Clown

Finally, it was time to make a demo !

The Bobs



The first effect created for the demo was **The Bobs**, those gigantic spheres moving on screen. The initial goal was to recreate the 5 bobs as featured in **Desert Dreams** demo on Amiga, but I “failed” with a result of only 2 displayed bobs. I spent a crazy amount of time, with many different implementations over 6 months - each iterations presenting larger bobs or better functionality. It internally uses the same technique than the commonly-known “Kefrens Bars”, but instead of displaying bars, circle shapes are displayed. The same scanline is continuously updated following the beam. Technically, this effect is a mix of real-time and precalculated data. Implementing a 4-edges clipping was the biggest challenge in this effect, especially the top edge of the screen (which involves pre-drawing invisible part of circles to simulate clipping).



First bobs for the demo… smaller, no rasters (July 2014)

The music by Factor6

Once I got the **Bobs** finalized, I asked **Ced** to create the **Clown** bitmap (more on that later about it in this document).

It was also time to get a music for the demo.

My first choice was contacting back **DMA-SC**. I was really pleased with the work he provided on our previous demo, **Phortem**. My wish for **phX** was to make a highly fast-paced demo, really Amiga-like in its approach. I asked **DMA-SC** for a music based on those requirements, and he created a prototype for me. While good, I realized the result was not what I was looking for. **DMA-SC** excels in the slow development of progressive themes ; while I wished something fast, something crazzzy (techno, dubstep, etc). We quickly (and respectfully) decided to interrupt the collaboration experience there and I started looking for someone else. Once again, I re-iterate to make things clear: **DMA-SC** is an highly respected musician, skilled and awesome ! I loved my collaboration on **Phortem** with him and still admirate the work he provided for it.

It was the time **Overflow / Logon System** released **IO**, its demo for MSX 1 machines (2015). He told me lots of positive things about **Factor6**, the musician that composed the audio for his demo. I personally appreciate his work on **Batman For Ever** demo so why not ?

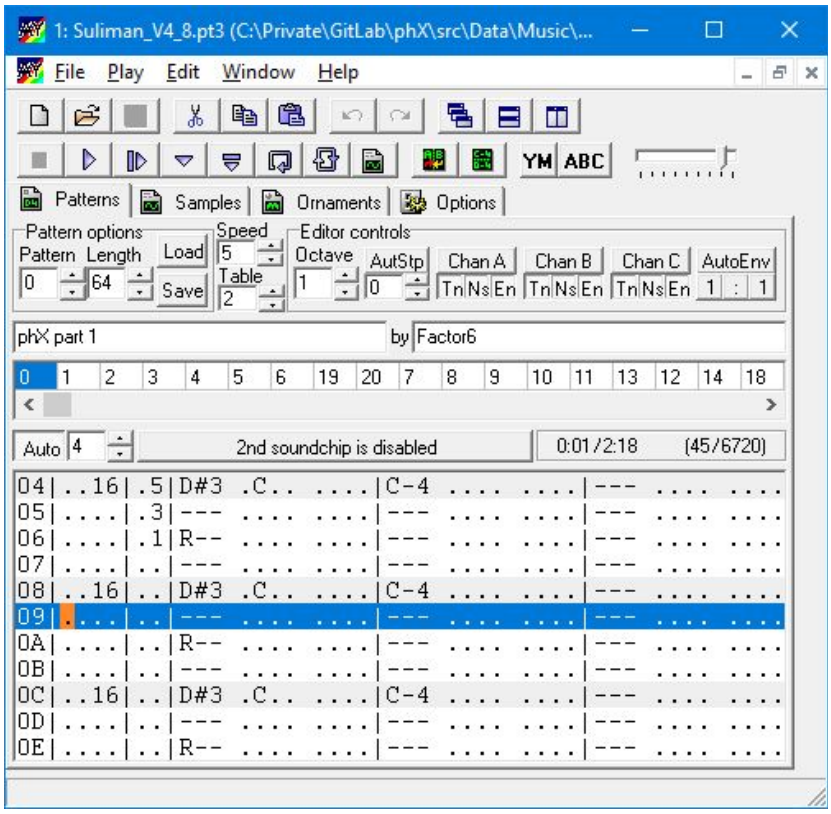
So I asked **Factor6** for his availability and motivation to create the music for **phX**. This time, based on my previous experience with **DMA-SC**, I decided to suggest an existing music as reference. In real life, I enjoy a lot **Infected Mushroom**, which is a psytrance band. So I suggested **Suliman** from the **Vicious Delicious** album. **Factor6** accepted !

Factor6 had many impacts with his work on the demo. Initially, I was highly influenced by **Vanity**'s work on transitions in **From Scratch** and **Still Rising**. I wished to make effects that quietly disappear from screen, smoothly replaced by others, etc. But when I received the first versions of the music, I realized **Factor6** changed radically the theme every 2 patterns. This was perfect timing to show an effect, but wasn't good to start "losing" time with transitions. So… I put in place a guideline that I respected throughout all the demo: instead doing transitions, why not fast-flipping directly to the next part, immediately ? I knew this would be much more complicated to support - this implies for parts to prepare the next parts while processing the effect in-place…

Also, what I did not know initially: **Factor6** is a musician with a strong demoscene culture. He helped me to shape the demo in a positive way with his suggestions. The relationship worked in both ways: while I received suggestions from him, I was able to express my doubts and ask for changes that would fit the initial vision I had for the demo. With such a project, it's important to communicate without fear.

Some words from Factor6 himself

Factor6:

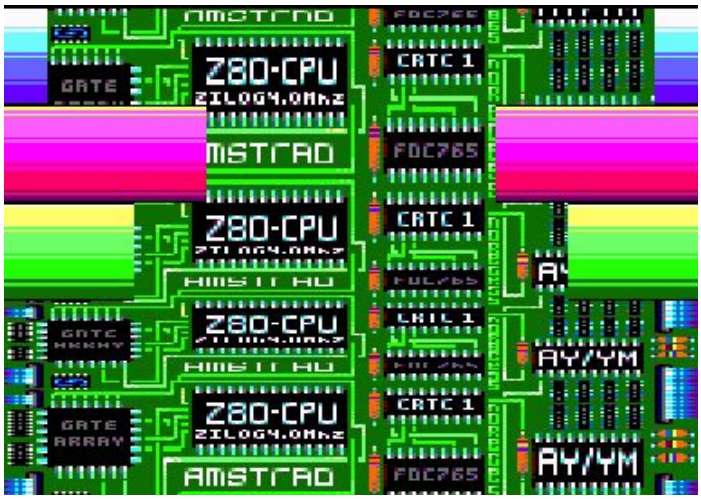


*When **NoRecess** asked me for the music for the first time, I was amazed with the demo effects he had sent to me. That's what I needed for the inspiration. He also wanted to make a cover or remix of existing real music track, which I didn't like too much, but I wanted to make what was requested so I started to make it but then I realized, that it needs more changes, so I made my own parts of the music. After success of the **Batman Forever** demo I had quite a clear image of how should our next demo look like, so I suggested **NoRecess** with some ideas as well as arranged the music on my 'musical taste' needs. I was using a PC crossplatform music tracker called **Vortex Tracker II** which I'm using for making musics for the **Spectrum** and I did a music for **IO** in it as well.*

***NoRecess** decided then to add a second (actually it's the last one now in the demo) part and asked for a new music with **Jarre**'s **Chronologie** theme. I knew quite a cool cover of Chronologie from the ZX Spectrum, so I decided to have a listen to it and start to make something similar. After all my music sounds completely differently, so that's good I think. I don't want to cover anyone in my top productions - yes, I consider making music for this demo as a privilege. Then, we've decided to add the middle part and I completed the music for this part quite quickly with some hints by **NoRecess**. I like the middle part music the most.*

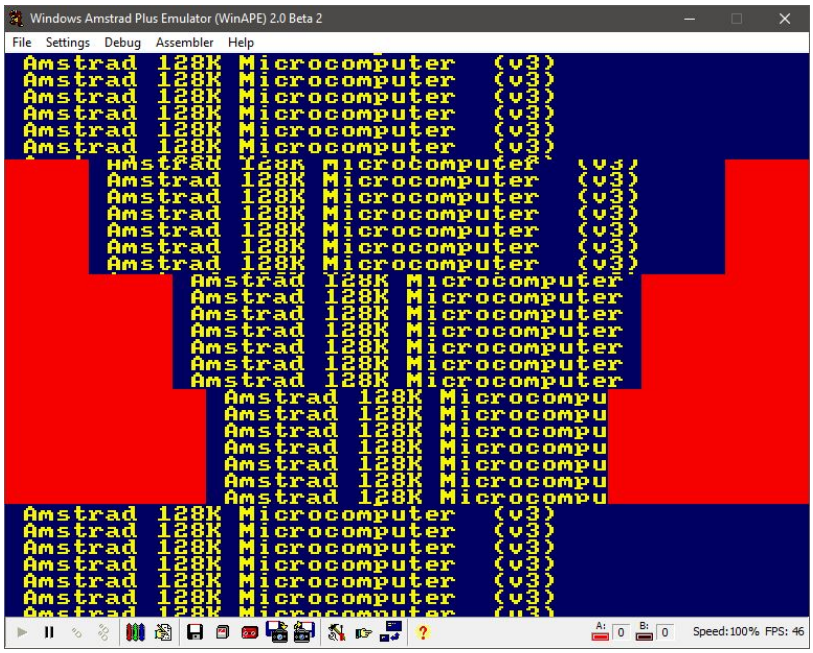
*When finishing the demo, we found that there's something wrong with the player. The music played too fast at some points of the demo. We were solving this for quite a long time. I've made many different versions of musics but with no peremptory impact on the demo behavior. But then, literally on the last minute, **NoRecess** fixed something in his code and the music speeding up problem disappeared! We both relaxed.*

“Borderized” Spectrum Analyzer



So next came what I call the **”Borderized” Spectrum Analyzer**. Displaying the animated motherboard was relatively easy ; but the horizontal bars were complex to implement. I initially though `R6=0` would not work everywhere (which is true…), so I relied on playing with `R0` and `R2` for this (which is more complex to implement, close to the “RVI” technique). It was only when the demo was almost finished that I got knowledge of `R8=&30` to enable the “border scroll” feature on CRTC 0 types. I think using `R6`/`R8` is less intimidating than relying on `R0`/`R2`. Meh!

The spectrum analyzer routine itself (cf. calculating the volume values) was first introduced to me by **Demoniak**, he definitely deserves the technical credits here. He gave me the source of a working prototype that I later rewrote for my own usage. To integrate it in the demo, I observed that only a small range of values were “looking good” (cf. smooth animation) so I decided to create the horizontal bars’ effect on that.



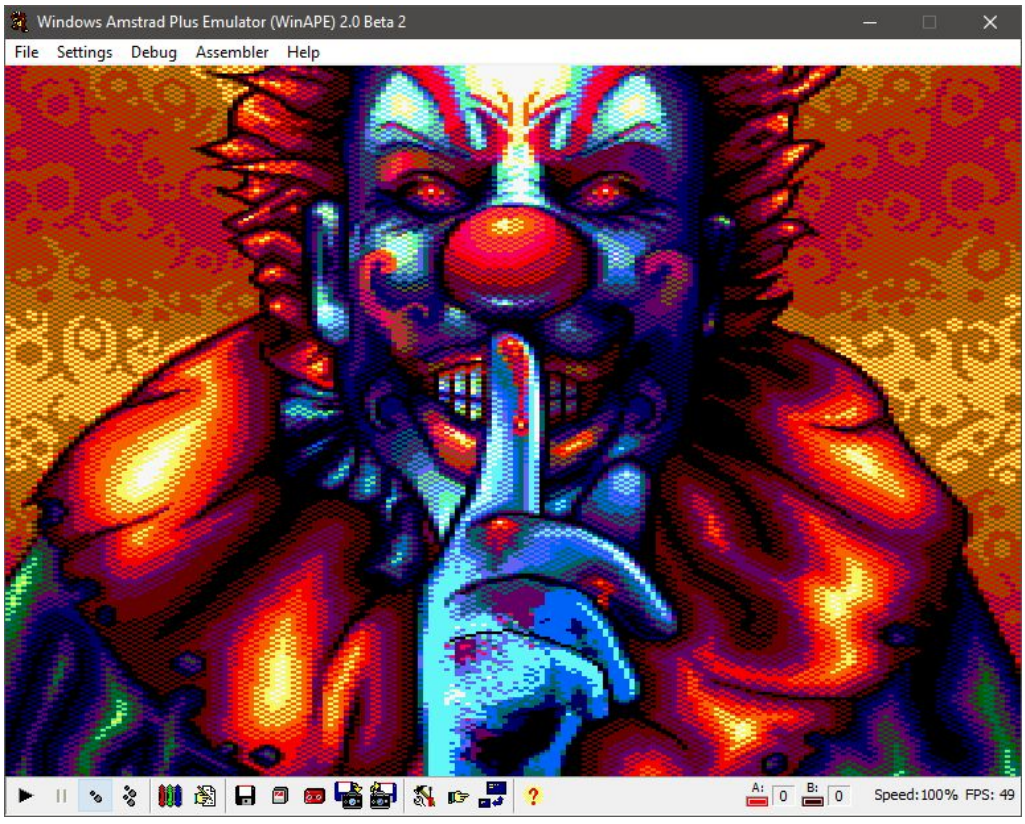
First version of Spectrum Analyzer… definitely NOT relying on border-scroll technique (July 2015)

Clown

The **Clown** was the next effect to be implemented. I was pleasantly surprised by the art done by Ced. I asked him for a clown ; I got the inspiration in French news: some people were (not kidding !) treated in justice because they disguised themselves as clowns to scare children in street… like in *It* movie, but in real life :)

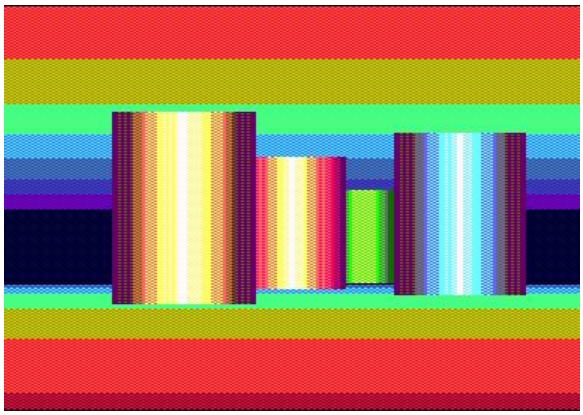
Aesthetically, I remind a mention from Ced in private conversation that he actually merged 3 different existing clowns into a single image.

I knew this static image would permit later to activate the **Arnoldemu**'s musical track-loader. The fact the image was displayed using a vertical stretch was a big-win here. The result was looking good !



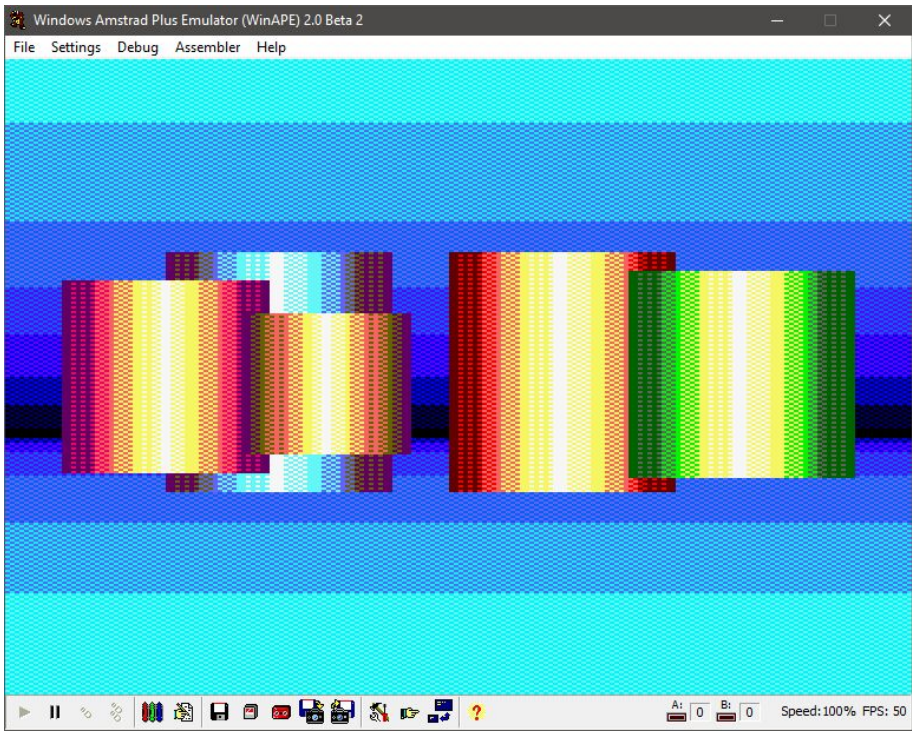
One of the intermediate step of the Clown image (November 2014)

Vertical Bars



Then I implemented the **Vertical Bars**. I also rewritten this routine lots of time. It features an approach as complex as the zoom-scroller as seen in **Batman For Ever** or **Still Rising** demos. The good finding with this effect was to rely on a cycling animation for the bars ; this is barely noticeable thanks to palette changes. Technically, the vertical sorting required to simulate depth was complex to implement.

Small personal detail: while I was implementing this effect, I can remember the **Reset Party 2015** was held in France and largely commented on the net. I'm living in Canada. I remind being frustrated to spend time on the demo at home without the ability to discuss it publicly / receiving any kind of global recognition for this… of course this was a temporary state of mind: creating a long-term demo involves multiple unpleasant things, too. Note that similar feeling occured for the recurring **Revision** and **Alchimie** parties too… that's what convinced me to present **phX** at a party (and make the travel from Canada): I want to meet people and talk about the demo ! (I released **Phortem** out of party because I was not able to present the demo in person).



Version in development… bugged as hell (September 2015)

phX logo

Next came the **PhX** logo displayed on start.

Historically, the first part to be displayed in the demo after startup was the **Spectrum Analyzer**. But I quickly felt a simpler effect was required to introduce the demo, so I decided to start the demo with a logo at startup.

I remind that I got the idea of animating the background rasters with music while I was in the bus back to home after work. A note on that: making such a demo required constant thinkings about it ; public transportation was for me a precious time where I could think about the demo, an effect, an optimization to bring… it's not true that sitting in front of computer is enough to make a demo. Demomaking is a lifestyle !

About the logo itself, I initially asked **Ced** a "p" in lowercase, but I got uppercase "P" as result ;-)

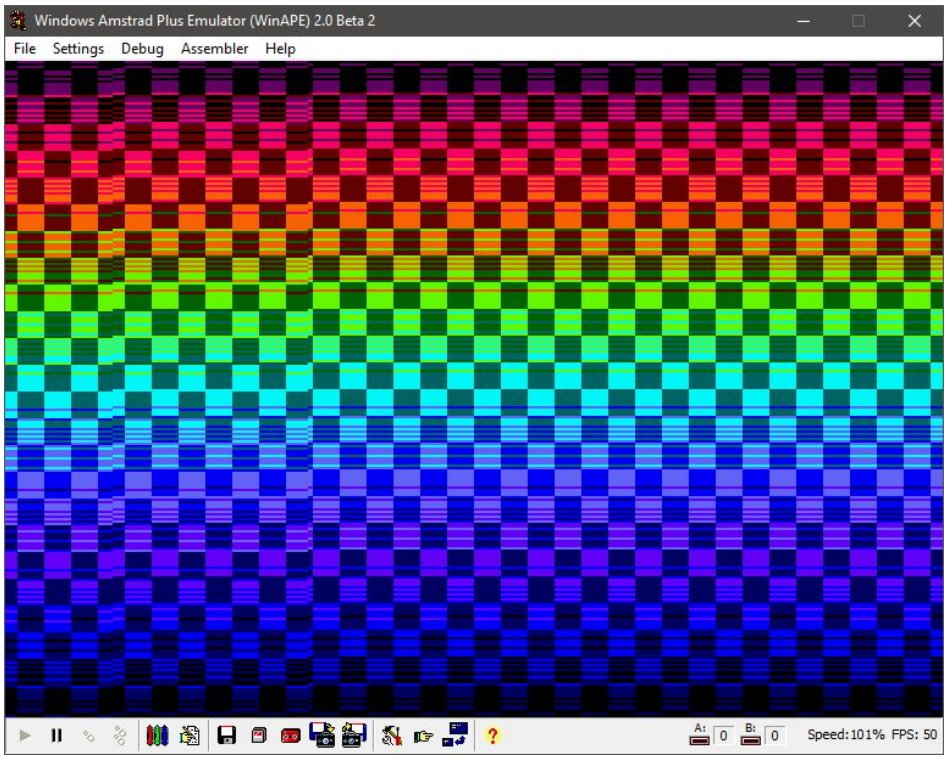
Overall - I'm pretty satisfied with this effect. It's a proof that it's possible to create an enjoyable screen without introducing sophisticated algorithms, sometimes a well designed idea is enough.

So from there, we got this part, followed by the spectrum analyzer part, then the bobs followed by the vertical bars, all of that using with the music from **Factor6**: that was it, we got a first glimpse of the real demo, timeline was aggressively good and the overall feeling was great !

Zooming Chessboard

After many size optimizations, and better initializations, there was still room for another part. So I decided to experiment with the **Zooming Chessboard** ! This is where I realized I was starting to be more comfortable with CRTC programming. It really took few weeks to implement the complete effect, which was an improvement over the nine months I required for both Bobs+Vertical Bars part.

The real challenge here was to display the layer with the characters - it involves a new approach consisting of drawing each lines of characters while the screen is being progressively displayed. The height of the characters depends of the time to process each lines.



One of the first version of the zooming chessboard (November 2015)

A glimpse of the full phX “experience”

Who is here ? Shhh…. Condense

Since the **Zooming Chessboard** timeline allowed to animate 3 small words sequentially, I decided to use *who is here* as message, then asked **Ced** to integrate a Condense logo at the bottom of the **Clown** bitmap… **Ced** then got the idea to add the *shhh…* mention on top of the Condense logo when he realized the clown’s art suggested the silence with his finger (how fortunate !).

All in all - if you ask me what is my preferred moment in the complete demo, I would definitively point the chessboard displaying *who is here* till the clown got stabilized then showing *condense* with *shhh* mention. I think they all fit nicely together with a great timeline, we just could not make it better as it is currently presented. And of course let’s mention the great intensity of the music at this time.

No blank screen between parts

One point to underline is the unpacking side. I mentioned the development of a custom packer through the R&D phase of **phX**. Special effort has been provided to switch from one part to another without any single blank frame. For each parts, a small amount of CPU-time is dedicated to initialize and unpack data for the remaining parts. That was a design requirement for the demo, I knew this would have a subtle impact but in the end - it matters a lot.

Memory footprint

Another point I would like to mention is the memory usage. I don’t know what kind of wizardry it takes to achieve this - but it’s a performance to get all of this in the 128Kb of RAM of the Amstrad CPC machines. The musical track-loader can only be activated when screen is static (cf. no movements) ; which means the parts to be presented (stored as packed data) have to be in memory when demo is running.

Music

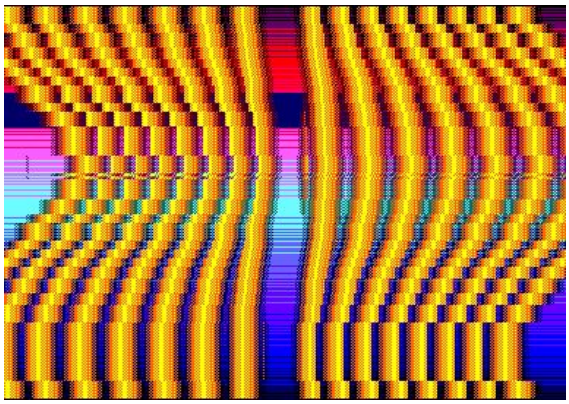
One last point is about the music. So **Factor6** adapted **Infected Mushroom**’s **Suliman** song. While the original music itself is good, **Factor6** decided to go with a custom theme based on his own personal inspiration after the **Clown**.

As a personal note, I would mention how important it was to me to use a music I like to listen in normal life - this really enhanced my personal motivation with the demo.

2016-2017 : from Clown to Freddy

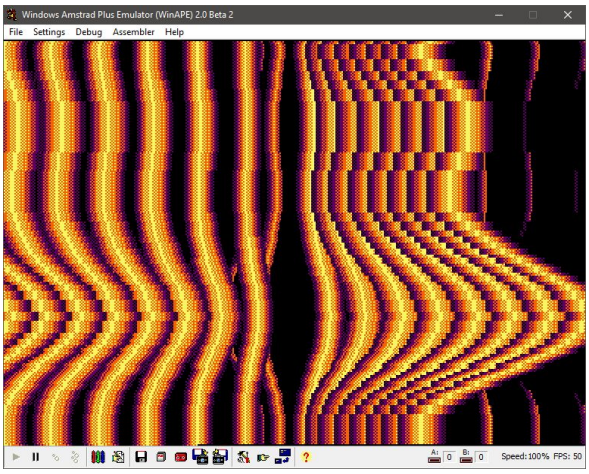
April 2015. Half of the demo was completed ! 50%, because at the time we thought the demo would contain another 3-4 effects and then the demo would be completed.

Epsilon



The vertical bars deforming with the strident sound is called the **Epsilon** part, as a recognition of an old effect exposed in a demo created by Epsilon (a demo maker) around 1994-95. It was an easy-made effect, driven by the curiosity of what would be the result if the bars were distorted while rotating. It appeared so obvious to me that I still wonder why Epsilon did not try distorsion back in 1994 :)

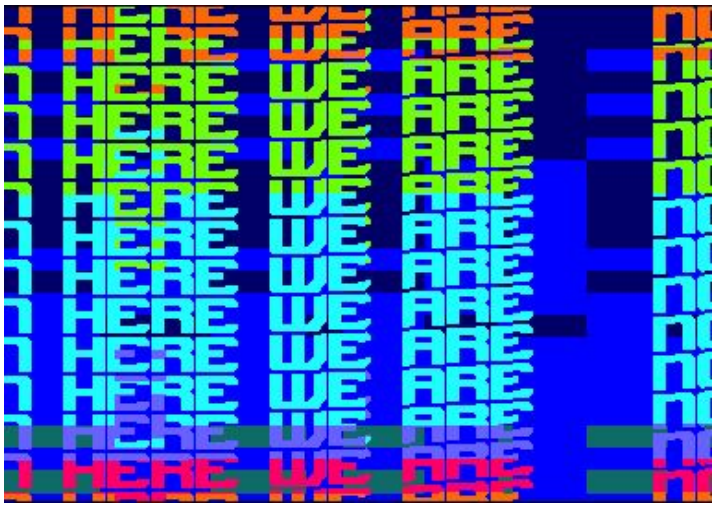
The failure of this effect is that the intensity of the distorsion is relative to the strident sound's intensity - this is what was expected but it would have required some additional work (faster movement..) to get a better visual coherency. As presented in the demo, this effort is not really obvious to understand.



The first version featured smoother animation than in final demo, but required all available VRAM (June 2016)

My inspiration was based from this demo from Epsilon (Power System Megademo, 1995).

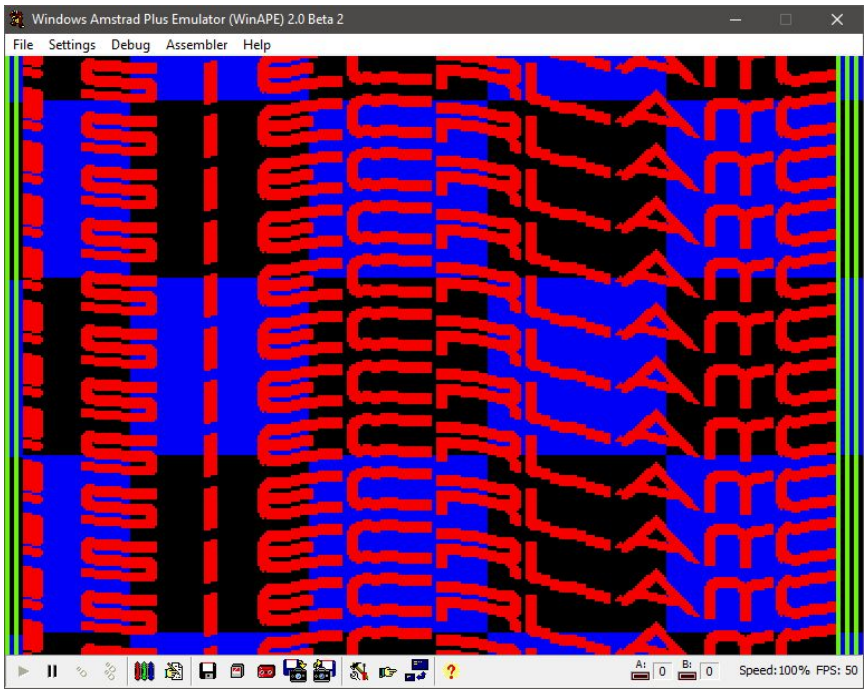
Sine Scrollers



Then came the **Sine Scrollers**. Initial version was looking better ! I had to degrade the effect to be able to fit into memory : the font itself packs very well, but is hardly visible. As for the background animation, which is pre-calculated while displaying the Epsilon effect, or even the sine curve - pretty basic. But better animations would have required more memory for pre-calculations - which is a rare resource :)

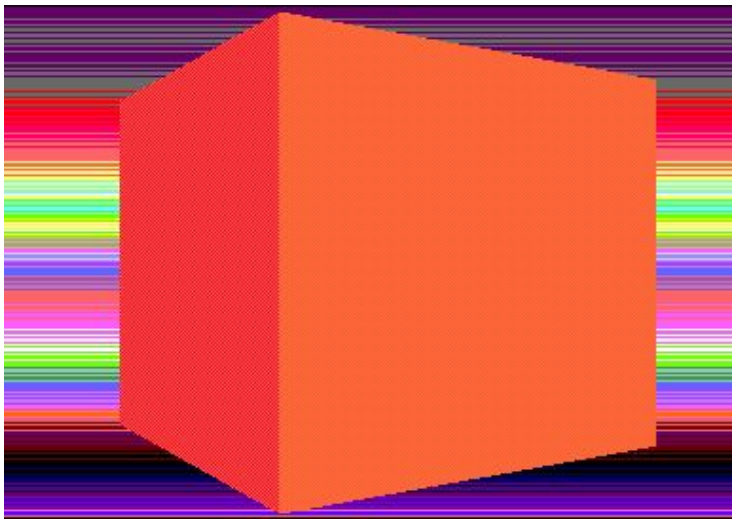
The real challenge here was managing a soft scroller while presenting a distinct layer in both background and foreground. This is achieved through a subtle horizontal background colour change (mixing both in data + rasters) for each vertical slices of character.

This part got mixed reception while sending previews. **TotO** disliked the background animation, while **Overflow** praised the effect. Personally, I think it's great technically but visually it does not reach the quality of the other parts.



First version of sine scroller… featuring a chessboard in background (June 2016)

3D cube



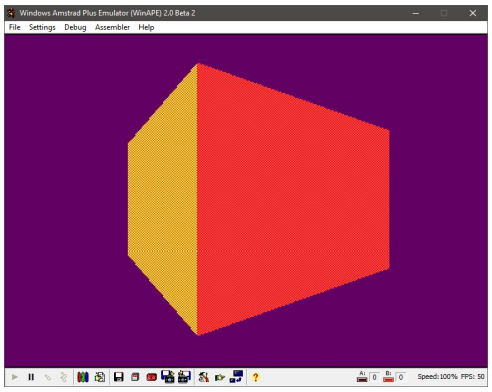
The **Cube** was the following effect to be added into the demo. That one was a real challenger to implement ! The effect is actually made of 2 separate parts: what people can see while rotating on X-axis is a complete different routine than the one used to rotate on Y-axis.

The rotation on X-axis is made of classic line splitting with edges updated dynamically on the side for each scan-lines (similar to the rotating Batman sign in **Batman For Ever**).

The rotation on Y-axis is more complex to implement: the top part of the cube is implemented using delta-diff (only the pixels that require changes between 2 frames are updated); the bottom part is re-using the exact same data but reversed using a different tileset. The middle part is managed using a line splitting.

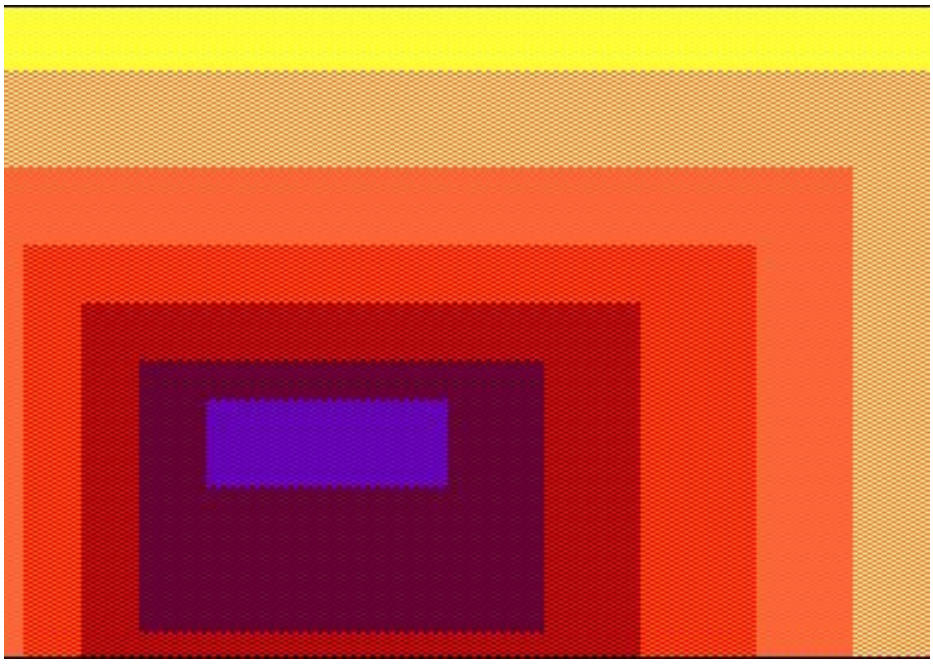
Sometimes, for both implementations, there are frames that require special cases. Also, the cube rotating on Y-axis does not respect the aspect ratio in order to match the full-screen height of the screen ; it's noticeable when observing closely the rotating cube.

The cube changes colors while rotating. It was the result of a technical research from **Ced** to simulate shading with Amstrad CPC's palette.
It could have been possible to support Z-axis rotation, but memory was lacking.



First version of the cube (September 2016)

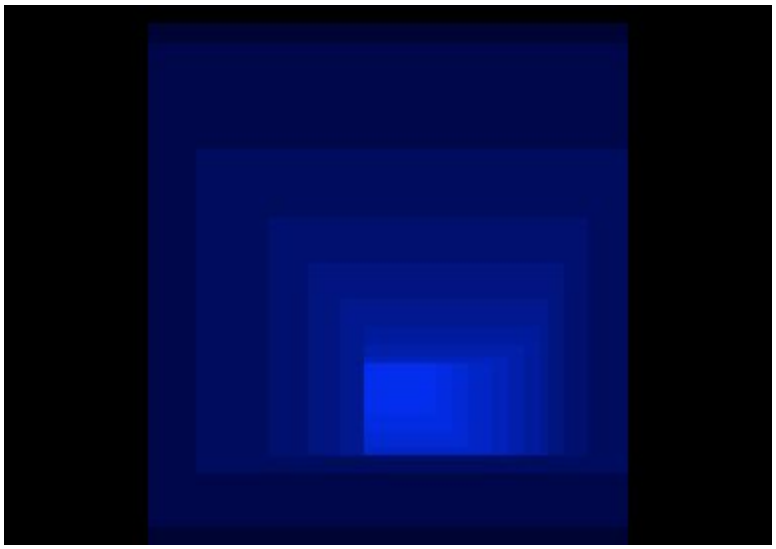
Tunnel



Finally, the **Tunnel** part was implemented in February 2017.

Here it's a remake of the most impressive effect available in the old **Phat** demo (the Amstrad CPC demo made during my comeback in 2008). I have been curious for years about an accelerated version using the CRTC, so I did it !

With this effect, I particularly enjoy the proper introduction around the screen on the first frames as also the ending to dark screen.

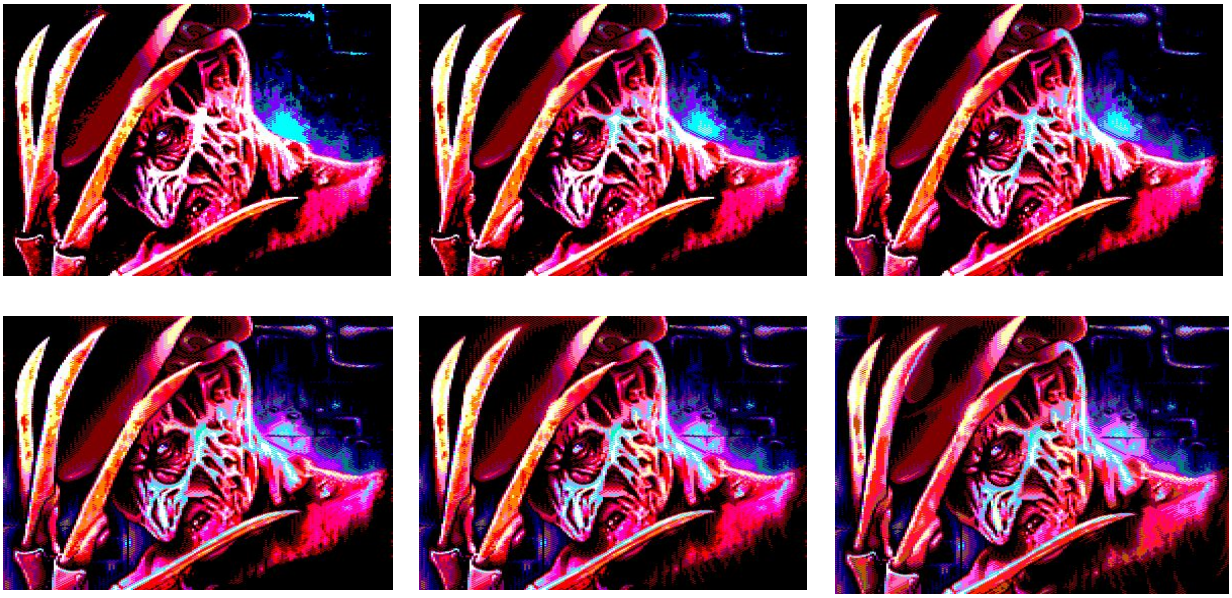


Tunnel in Phat demo on Amstrad Plus, was approx. 7-10 frames/sec and half-screen only (2008)

Freddy

The **Freddy** image was added later into the demo, when I managed to make data optimizations to reduce overall memory footprint.

We had hard times to find a suitable image… **Ced** originally drew a picture of **Harley Quinn**, but it was more "Barbie is angry"-mood and did not fit well within the demo ; we also tried a picture of **The Prodigy**'s **Keith Flint** (the mad guy with the green hair :). But in the end, **Freddy** was the clear winner here.



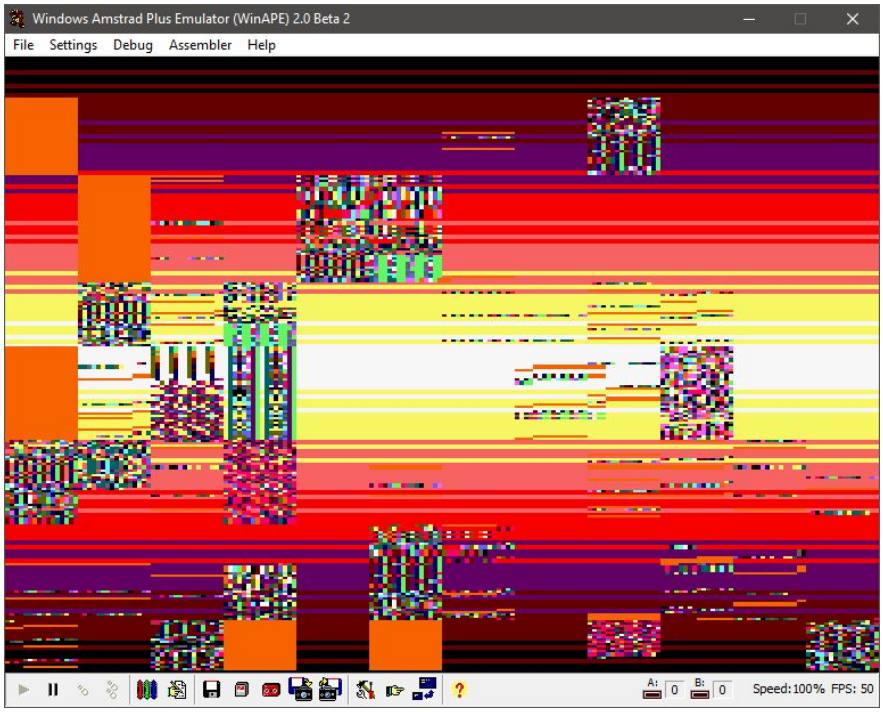
Freddy steps - every days of work brings tiny changes !

Summer 2017 : the Outro

We needed to create a special part to end the demo. That part had to be in a relax and quiet mood. A vertical scroller was suitable here for that purpose, like a movie. In all previous Condense demos, we avoided scrollers - because we had not much things to say, actually. But here with **phX**, the context is different - we wished to share some words about the demo. And after watching the **30 Years Megademo**, I realized how great was the use of greetings !

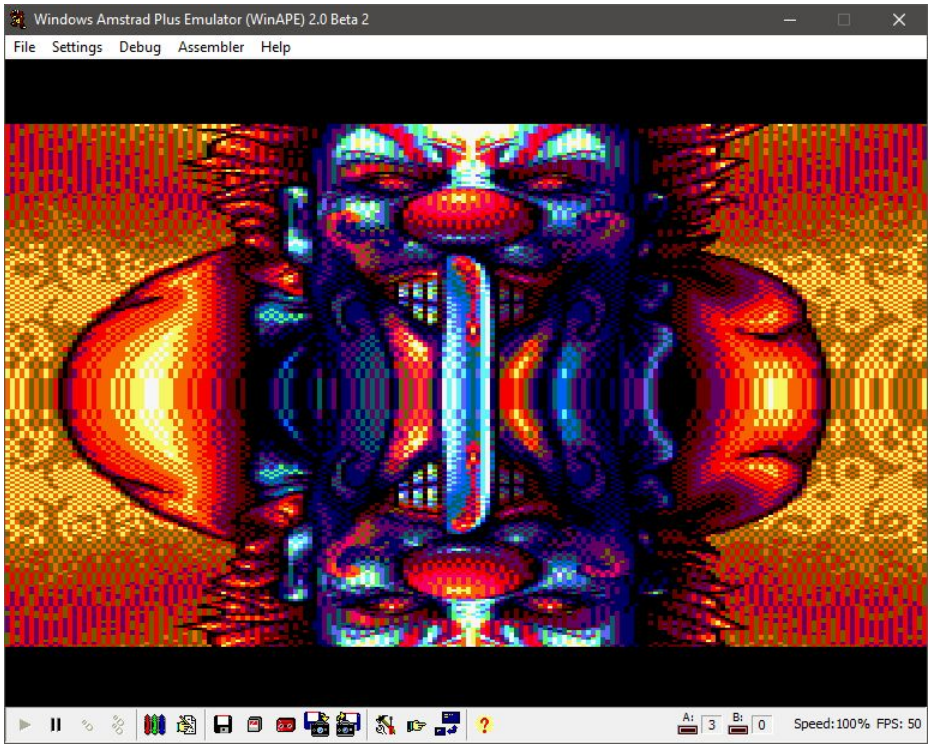
The outro is exposing an effect called **Disto Scroller**. It relies internally on one of the most advanced usage of the CRTC, the so-called **RVI** as usually referenced by French programmers. It was introduced to me by **Overflow**, that explained me its approach in the **S&KOH** demo. He definitely deserves the credit of showing me the trick to iterate VLC quickly in the invisible left part of the monitor. This technique allows to use more VRAM using repeating lines. The problem with that algorithm is that it consumes almost all available CPU-time… it was hard to find a good usage for that technique. That's why most of implementations on CPC are usually presenting distortion of a static image. With **phX**, this is the first time on Amstrad CPC where RVI is used to present a vertical scroller.

Managing the vertical distortion + updating the rasters every 2 lines was definitely a great challenge ! Without mentioning the update of the scroller's content itself (cf. the characters in the font to manage).



First running version of the DistoScroller… who needs font? :) (August 2017)

Note that my first tries with RVI dates from April 2015.



First successful attempt at using RVI technique, using Clown as test image (April 2015)

September 2017-March 2018 : from Freddy to Fishes

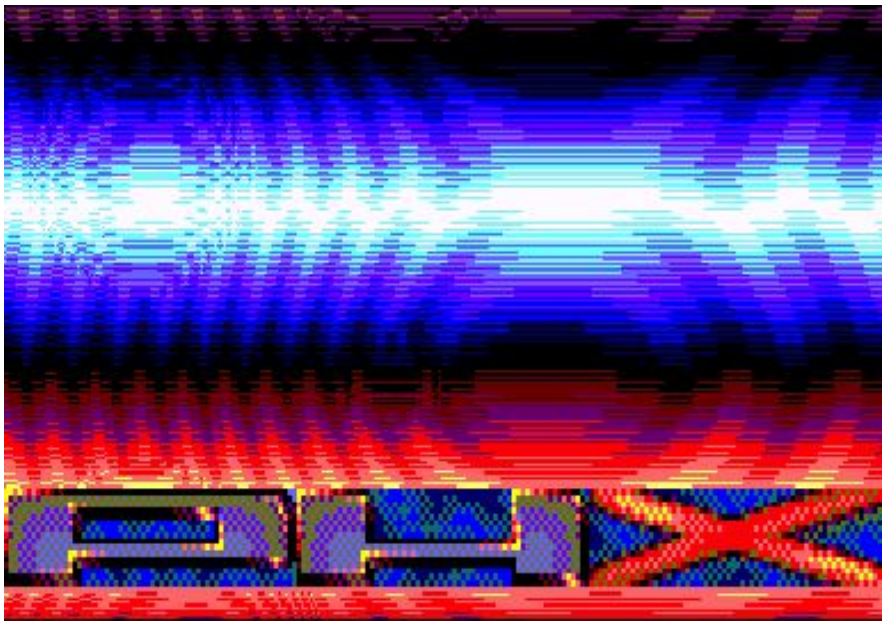
End of summer 2017: *the demo is over !..* Really? Hmmm. While content was pretty satisfying, it was clearly not enough. The demo needed to be extended with another bunch of effects, otherwise final reception's criticism would be all about the demo's length.

The demo was meant to be presented at **Revision 2018**. I absolutely wished to target the Revision party to get maximum exposure of the demo - because I believe **phX** has potential to reach people outside the Amstrad CPC community.

Deciding for such a demo expansion was risky:

- **Overflow** approved the choice and suggested to make a "filler" - cf. do simple things and extend your timeline as much as you can. But I was reluctant in doing low-quality effects after spending so much time trying to do my best...
- Would we be able to finish the demo on time ?
- Will the demo appear to be rushed ?
- etc.

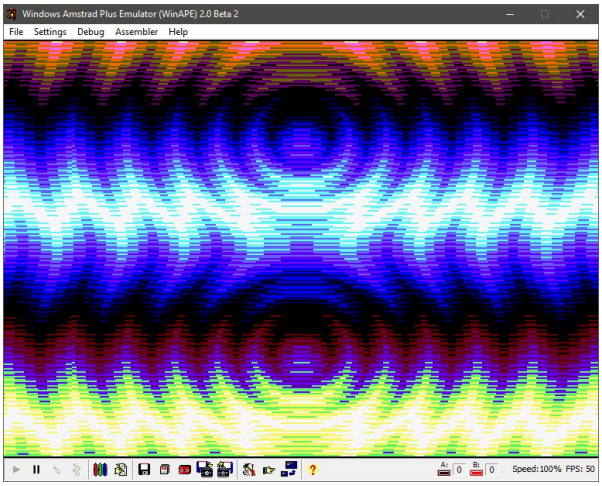
Plasma Circles



The first effect to be implemented was the **Plasma Circles** effect. It's based on an old effect presented around 1994 by **Odiesoft**. The iteration provided in **phX** features full-screen + dynamic update of repeated lines. The circles that appear / disappear are an artifact embedded into the visual data. Several attempts were achieved before obtaining a satisfying result.

The stretching **phX** logo on top of the effect came later as an enhancement to make the screen a bit more complete. There is a problem with this logo - more especially about the background. **Ced** made several versions to provide the best possible background he could achieve using 16 vertical lines.

I decided to add flashing colours following the music to bring some consistency with the **Bobs** / **Vertical Bars** part.



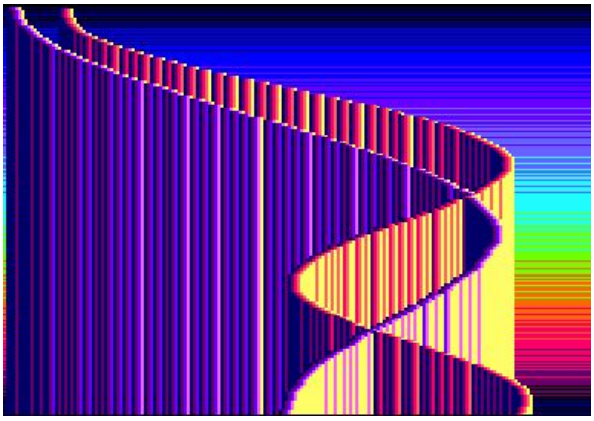
First version (November 2016)

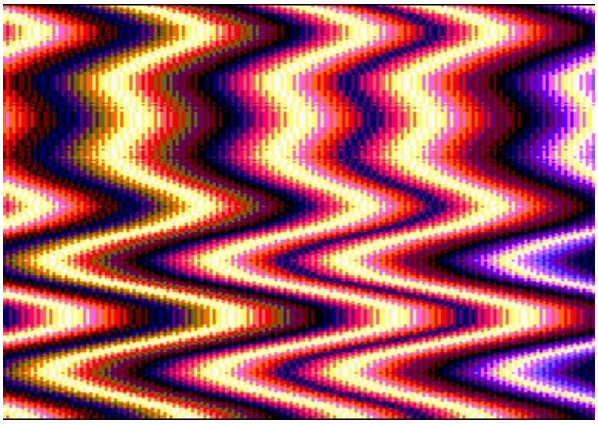
Odiesoft's Circles (1993)

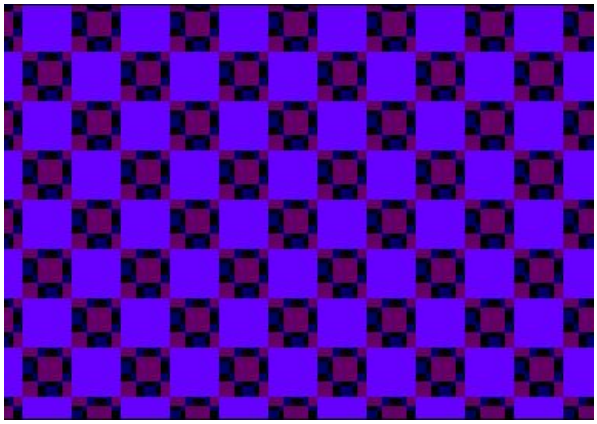
Multi-FX

Then came the **Multi FX** effect, in December 2017. The initial goal with this effect was to mix easy-made effects together, scrolling down quickly. The plasma, the Kefrens bars and the multi-Condense effects were quickly implemented ; but the 3-layers chessboard was a really complex one. It took me days of thinking before having a valid idea of its implementation ! **Gozeur** first implemented this in 1994 and I'm still impressed by his performance.

The part may look simple, but internally it's crazy as hell to manage. The part is always evolving, cf. the plasma does not use the same palette than the Kefrens bars effect. Plus, all effects together use more VRAM than the "line splitting" CRTC technic permits ; dynamic copy of data had to be implemented, depending of which parts is on-screen. And… since the **Plasma Circles** (next effect to be presented after the Multi-FX) also use all possible VRAM, the fast scrolling back to bottom in the Multi-FX also quickly prepare the next part (which is why the Kefrens bars effect is displayed till the very last moment - it uses a minimum amount of VRAM).







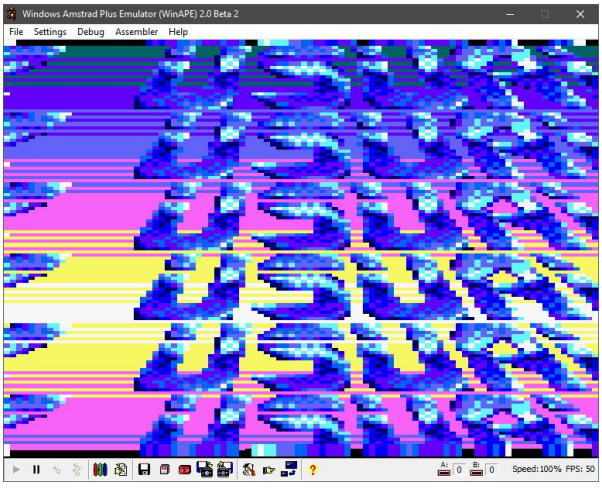
Flipscroll

Then the **FlipScroll** effect was implemented. That one was directly inspired by a demo from **CRT** (C64 programmer), a colleague of mine in my previous job.

Another inspiration was the zoom-scrollers as exposed in **Batman For Ever** and **Still Rising**. I tried to mix all of that and the result was convincing enough.

But this part really shined when the “repeating words” effect + rasters in background was added.

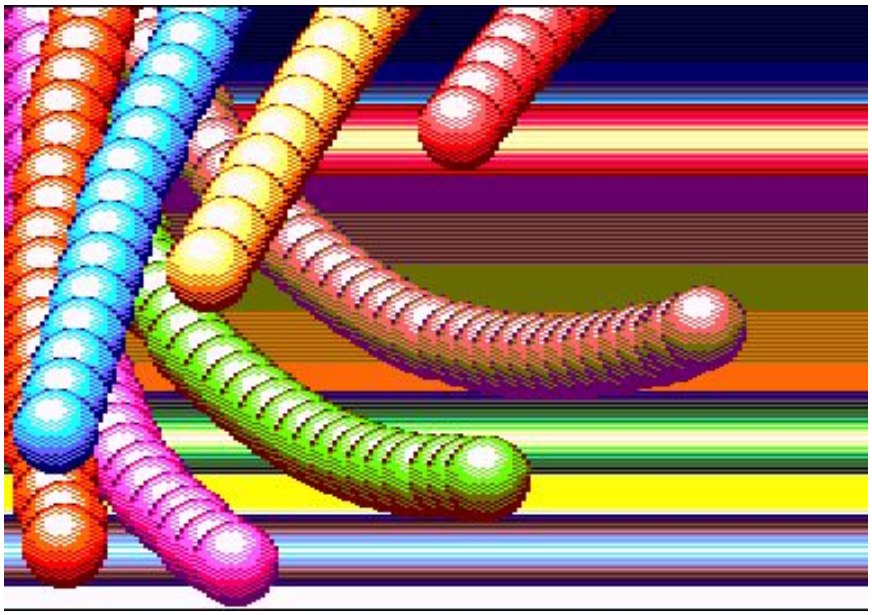
2 weeks only of work was required to implement that part !



The first version was repeating the effect several times (December 2017)

Flipscroll by CRT in Wonderland XI (C64, 2012)

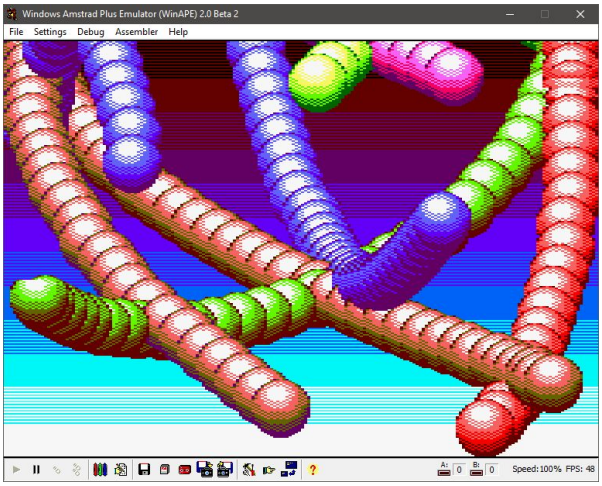
Balls



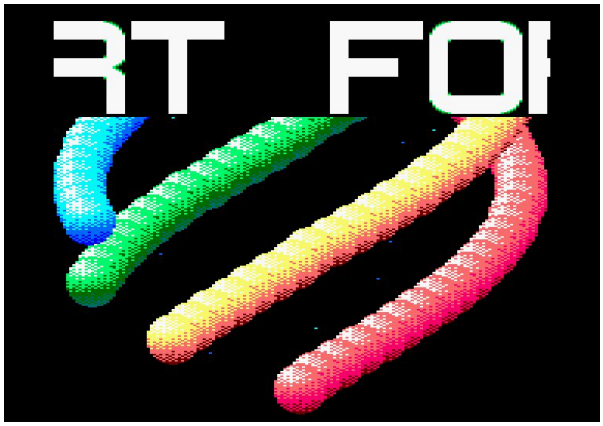
And finally came the **Balls** part. Once again, it's a remake of an existing effect made by **Face Hugger** in 1993. His original implementation featured 4 balls, while the attempt in **phX** features more balls in real fullscreen with highly animated background. The overall result is colourful, original enough to impress people for 2-3 seconds (as mentioned by **Overflow** when he first watched the preview). Variations in movement and background have been introduced later in order to avoid the part to be boring.

Like the **FlipScroll**, only 2 weeks of work was necessary to implement this part too !

Like **Phortem** demo with the **Batman** part, **phX** also features dubstep ! :) It's definitely a risky choice, especially since **Factor6** was reluctant to apply this kind of sounds in the demo. In the end, I believe the effect fits very well with the music.



First version featured 8 balls (instead of 7 in final) but did not have moving rasters in background (January 2018)



Face Hugger's Space Balls (1993)

Fishes

Then **Fishes** image is displayed. When first implemented, both **Fishes** and **Freddy** were displayed without care - I had no choice to unpack images before displaying it, resulting an unwished black screen for 2-3 seconds. **Factor6** mentioned the missed opportunity to display something original, like the effect applied with the Clown.

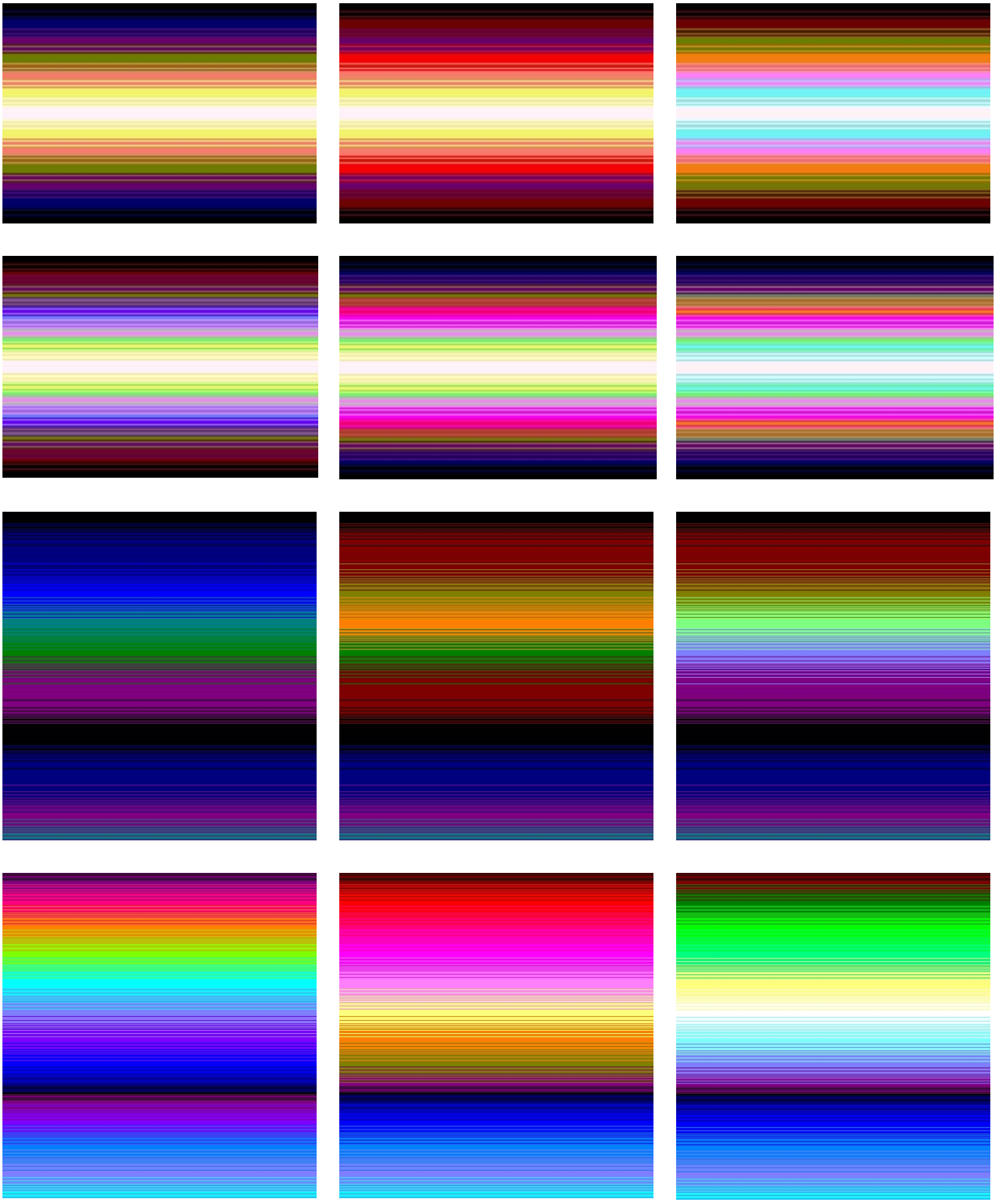
Based on the fact I had to rely on unpacking data, I came with the idea to dynamically unpack sub-divisions of images. That's how the **Freddy** part got displayed using 3 columns, and the same approach was then re-used to display the Fishes screen.

The inspiration of the image is a transfer of an album cover that I suggested to **Ced**. **TotO** suggested minor changes regarding the palette and **Ced** approved. Did I say I really enjoy **Ced**'s open-minded attitude ? Technical skills are one thing, but the human behavior is another precious quality…. Ced rocks. :)

*The original image (suggested by me) is based on a cover of **Running Out Of Time**, a not-so-good album by **Melicia** (psytrance)*

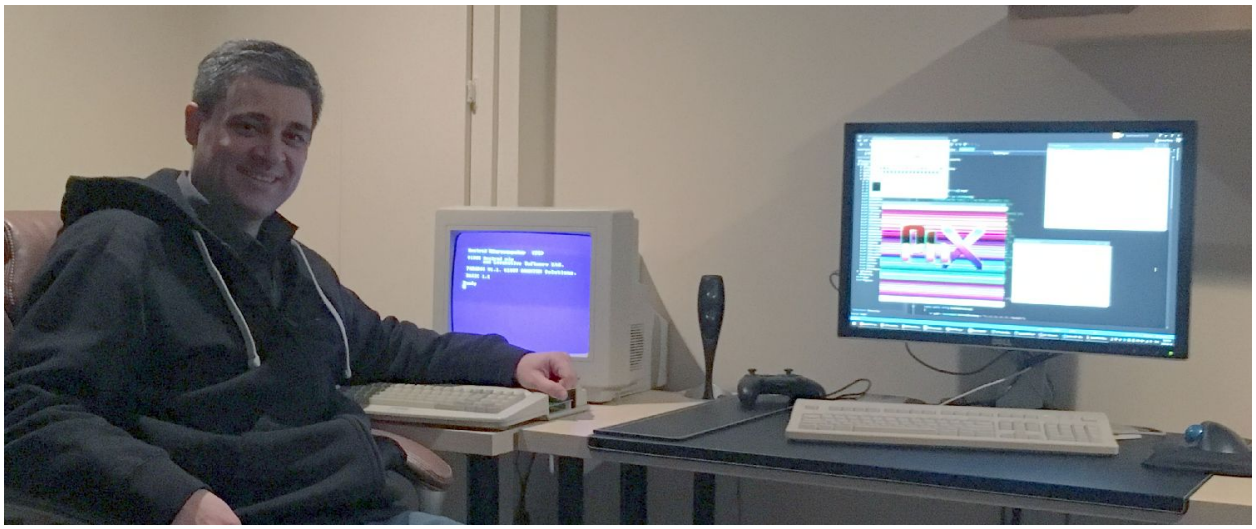
Ced’s background rasters

Ced created lots of colorful background rasters (not all landed into final demo).



Random notes and recommendations

Something unusual to note here - please have a particular attention to schedule. All those parts between Freddy and the Fishes were implemented in 6 months ; it was a personal challenge to reach this degree of quality in given time. I was literally coding days and evenings ; every opportunities to progress on the demo were taken. Coding in the train, during lunch-time, when Netflix was boring.. let me claim it loud: *this will never happen once again in my life !* (it was too intense).



NoRecess / Condense coding at home, March 2018

A global recommendation would be to pick people of trust if you are planning a demo taking years of development. You need to keep motivation through the whole development cycle ; sending previews from times to times helps a lot on that front. You need people to tell you when it's good, but you also need people to tell you when you are going wrong.
But sending previews can also have a perverse side-effect: you may be satisfied of the feedback, take it for granted and then take a large pause from here. This is a trap. :)

I have to say that **phX** has been highly inspired by other Amstrad CPC productions like **Wake Up**, **Batman For Ever**, **Still Rising** and **Logon's Run**. All those wonderful demos were created during the last decade and I really expect to see another productions of that level of quality coming through the next years ! **Madram**, make us dream ! :P

Keep curiosity high! would be another advice ! If you have an effect in mind, something to try, if you see something existing that could potentially be improved in a demo…. Stop talking about it, go implement it ! :)

Also: it's OK to fail. You learn with failures. Even if you have spent weeks to create a single effect, if the reception is not good, then it's not good. Don't try to push things with the simple argument that you spent time on this. I have tons of those prototypes landing somewhere and really, it does not worth releasing them. I admit **Overflow** got particularly intense with me with his criticism ;-)

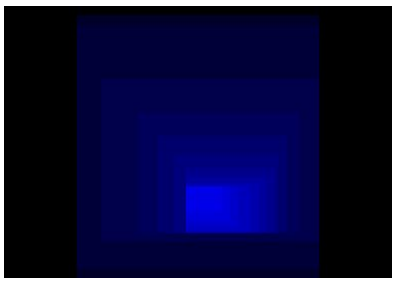
Accept the criticism, and remain open-minded when people disagree with your direction. Do not accept everything too - my advice here is to listen and apply other people's feedbacks as much as possible, while respecting your original direction. Especially when there are "impossible cases" to solve: I got the case where **TotO** suggested one direction and **Overflow** the perfect opposite. Oh god ! :)

Technical detail: the music is uninterrupted from the beginning of the demo till Freddy gets displayed. Internally, the music is split in 2 separate memory blocks ; second part of the music is loaded from disc to memory while displaying the Clown. The audio player automatically switches to the second part of the music when loaded. The first memory block initially allocated for the music get available for other data. That technique allow to make a music that could last hours without interruptions…

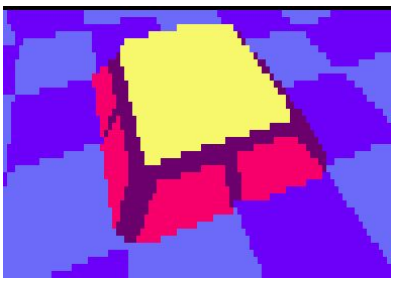
All models of Amstrad CPC (CRTC 0,1,2,3,4) at home. Ideal for testing !

Technical details regarding CRTC usage: the demo strictly respects **19968 nops** / **312 scanlines** per VBL / **64 nops** per HBL. The demo has been successfully tested against CRTC 0, 1, 3, 4. CRTC 2 displays an incompatibility message at startup. All effects strictly uses 268 visible lines (which is convenient resolution when working under WinAPE's emulation: this allows to get one blank line as first / last visible line) ; top / bottom invisible parts of the monitor display black bands.

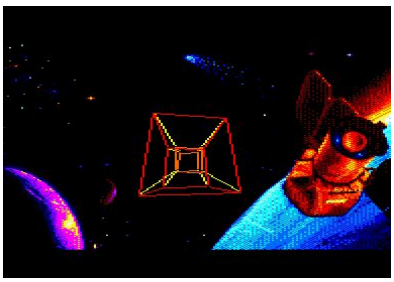
Here is a list of the **Condense** productions over the last decade and their required time of developments…



Phat 1 (2008)
6 months



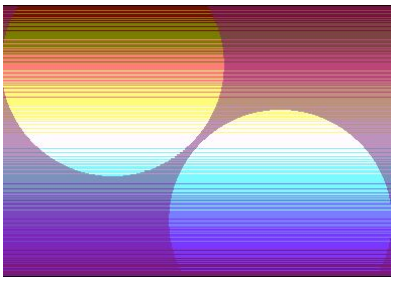
Phat 2 (2008)
2 months



Pheelone (2009)
9 months

Phreaks (2010)
3 months

Phortem (2013)
2 years !



phX (2018)
5 years !!!

…the trend is obvious, now make your own conclusion about those numbers. :)

One last thing, why the name **phX** ? It was originally the internal codename of the demo (cf. mentioned in folder names, implementation, etc). All **Condense** demos on Amstrad CPC are prefixed with '**ph**' (don't ask why: it's a cool concept of mine used without any solid explanations); '**X**' would mean here "any name". The plan was to find later a better name to be used in the visible sections of the demo (logos, AMSDOS binary filename, etc.). But a valid replacement was never found and **phX** was actually cool enough to read and pronounce so we decided to keep it final :-)

*My friends **Alkama** and **Soundy** sending me extra-motivation to finish the demo (Revision 2017)*

Conclusion

This making-of is over. I described development of all parts as much as I could ; if you are curious on some mentioned points, once again please do not hesitate to contact me privately through my website `http://norecess.cpcscene.net`.

Finally, **phX** is the result of a solid team work ! The demo would never be released if one member of the “core” team would be missing.

So thanks to **Factor6** for paying so much attention to details and for the fantastic audio support you did - you rules ! :)

Also, thanks to **Ced** for being one of the most talented artist I know on Amstrad CPC (and beyond) ! This guy is a machine ; he never complains when something is not landing in the final version of the demo. This is an incredible quality, that allowed multiple iterations again and again of the same part of the demo. In **phX**, his work not only consisted to make bitmaps and logos ; his work also consisted to create fonts and millions of rasters and everything else that is presented into the demo…

Also, I want to give a special thanks to **Overflow** (the “technical director” of this demo - but also *“my good friend on CPC”* : following closely your progress on your productions motivated me a lot doing the same on my side !), **TotO** (his focus on “coherency” was an incredible asset), **Fra** (as an Amiga user and long time friend, his point of view was representative of the expected reception) and **Longshot** (the “hacker” of the demo - sorry for not being able to bring proper support of CRTC 2 !). All those people gave continuous reporting to the previews I sent on a regular basis. Their suggestions really improved the demo, and it was a really funny experience to share.

Long live to the Amstrad CPC demoscene !
And oh, in case you missed it: **Amstrad CPC** is also **Amstrad Plus**/**GX-4000** ! I have been really pleased to discover those awesome Amstrad Plus productions at **Alchimie 2017**.
And finally, 8-bit rocks !!!! A warm hello to everyone active in the demoscene on a 8-bit machine in general…..

NoRecess / Condense, March 2018 (Montreal / Canada).
`http://norecess.cpcscene.net`