DEVELOPMENT DIARY OF LUMINAGIA - **AN AMIGA 4K INTRO FOR BREAKPOINT ~~2008~~ 2009**

BY BLUEBERRY OF LOONIES

zine BEHIND THE SCENE

This will be the fifth year in a row where I make an Amiga 4k for Breakpoint. It is becoming more than a tradition. More like a law of nature. This year it seems I will again be up against my favorite 4k-competitor - Scicco/Scarab - he has beaten me twice, but this year it is my turn to win. I hope many more will join the fray.

> **IT OCCURED TO ME AT THAT POINT THAT I WOULD REALLY LIKE TO MAKE A 4K WIH A STRONG FOCUS ON THE MUSIC**

Making a 4k is, for me, always an adventurous journey. I start out in some direction - with some initial ideas - but there is no knowing where I will end up. Often the end result looks quite different from what I originally envisioned. The intro comes to life in the turbulence arising from the struggle between the ideas and the limitations. Both are needed for the creativity to thrive.

Writing a diary about the process along the way will be an interesting experience. I hope it will be just as interesting for you to read.

AUGUST 2007

At the Assembly rave, they had a big screen with abstract patterns moving to the music. Nothing special in these days of media player sound visualizations, but it occured to me at that point, that I would really like to make a 4k wih a strong focus on the music. Where the music drove the intro, and the visuals were perhaps only simple, abstract shapes living out the music. This incident set the direction for the kind of inspiration I would be looking for in the months to come.

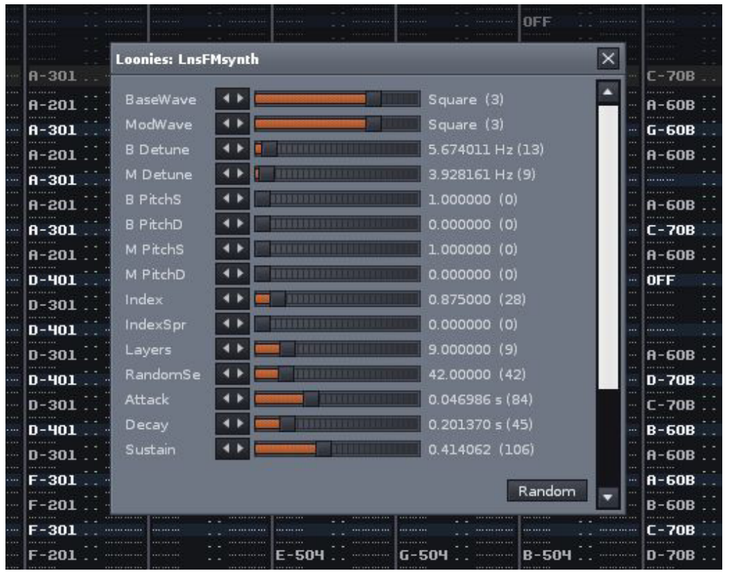
NOVEMBER 2007

I heard an interesting radio program about FM (frequency modulation) synthesis, in celebration of the 40th anniversary of its invention. This sounded like a simple way to make cool instrument sounds, and quite fitting for a 4k synth.

Not long after this, Maytz suggested making some eighties synthpop-style music for a 4k. This was excatly in line with trying out FM synthesis, since the characteristic sound of synthpop, to a large extent, comes from FM synthesizers.

DECEMBER 2007

In order to ease the development of the synth and especially the music itself, I decided to investigate VST plugins. If I could make the synth as a VST instrument, Maytz could make the music in Renoise (with appropriate restrictions), and I could just parse the Renoise song to produce the music data for the intro.



The built-in GUI in Renoise for adjusting VST instrument parameters. By exporting named parameters through the VST interface, we get the GUI almost for free

It turned out that VST plugins are really easy to make, at least when given the example code downloadable from Steinberg's website. I have started restructuring the code to make it easily extensible, and have added support for polyphony. The next step will be to insert a resampling pass to convert from the 28604Hz sampling rate we will be using on the Amiga to the 48kHz sampling rate requested by Renoise. Then I can start experimenting with the actual synth algorithms.

JANUARY 2008

2008 has arrived, and Breakpoint is early this year. The music for the intro is on track (though nothing has been done yet), but I still do not have a good idea for the visuals.

JANUARY 12, 2008

While watching an old Amiga intro - Ride! by Silicon - it occured to me that plain old 2D voxels (or fake-3D if you like) would be a good candidate for the visuals. One of the classic effects which I have never coded myself. Not that I want to make a boring voxel world flyby - the voxels should be organic and pulsing, along the lines of Showbase Shape.

JANUARY 13, 2008

Thinking more about the voxel idea, I started getting ideas on how I could generalize the effect to produce many different voxel effects - blobs, landscapes, tunnels, twisters (mmm, I like twisters) - by varying the parameters. I knew, from the way the ideas for the details started popping up in my mind like flowers, that this was the right idea.

It basically goes like this: First, trace rays through a height map. At each step along the ray, the height is compared to the highest point so far along the ray, and if it is higher, draw pixels up to the new height. This is the classic voxel tracing algorithm. The voxels are drawn into a voxel buffer which is subsequently mapped onto the screen, either rotated or polar-wrapped (making the voxel rays go radially out from the center of the screen).

Different voxel effects can be achieved by varying three parameters:

» The origins and directions of the rays in the height map

» The way the height is scaled along the ray

» The mapping onto the screen

For instance, to produce a voxel blob, shoot the rays out in all directions from one point, scale the height by a sine function (half a period) and map polar-wrapped. For a landscape, shoot the rays in a fan (starting some way along the rays), scale the height in a perspective fashion and map directly onto the screen (making the rays go vertically up). For a twister, shoot parallel lines out to both sides from a wavy line, scale by the sine and rotate onto the screen. Do the exact same but with polar mapping, and you get a twisting torus. :)

With both the music and visuals on track idea-wise, it is time to start working with more focus on the intro. 67 days to go. This is also the day the diary was started.

JANUARY 28, 2008

Haven't found the time to start working on the intro yet, but I have thought a bit more about the technical side of the voxel effect. I would like to try out a Chunky2Planar technique for 2x2 resolutions where only half the usual amount of data is written to chip memory and then unfolded using the blitter. This should make it possible for the effect to run at 50 fps, provided I can keep the work done per pixel before the c2p below something like 45 cycles. That should be doable with the voxel effect I am envisioning. A bit of a drastic change from the 8-10 fps I presented last year.

This c2p technique will be in just 256 colors, so it could be fun to play around with some clever palette tricks. More about that later, I guess...

FEBRUARY 2, 2008

I implemented the resampling in the VST. The actual synth now operates at a sample rate of 114416Hz - 4 times oversampling of

the Amiga sample rate. FM synthesis, when done in the straightforward way one would implement it from the description, generates lots of aliasing and thus sounds like crap without some oversampling. Or at least that's what the experts out there (hi kb!) say.

The resampling to 48kHz (or whatever sampling rate the VST is told to use) is done using a two-period windowed sinc filter. I have made the cutoff frequency of the filter (expressed as the number of samples covered by the filter) controllable by a parameter, so we can experiment to find an appropriate filter size. In the VST version, there is an unknown, possibly non-integer ratio between the original signal and the result, so I need to store the filter with a high resolution and sample it differently for each output sample. In the intro, this will be much simpler because of the fixed 4x oversampling. There it will just be a list of filter coefficients to multiply onto contiguous samples.

I also started on the actual FM stuff. The synth is pretty basic. Two oscillators, which can each be a sine, sawtooth, square or noise, with one modulating the phase of the other. (So, 'FM' is for 'phase modulation' - oh well, the acronym works in danish, at least.)

FEBRUARY 3, 2008

Implemented multiple layers in the synth. This was a suggestion I got from Loaderror: Calculate the sound many times with slightly varying parameters and mix the results together. This produces a much richer, chorus-like sound. In my synth, there are three parameters that vary from layer to layer: the frequencies of the two waves (detuning - essential for that chorus feeling) and the strength of the modulation. The variation is random, and the random-seed is yet another parameter. Searching for the right random-seed is an essential part of fine-tuning the sound.

Calculate the sound twice with different random values and voila! Nice stereo instruments.

The basic FM synth is relatively complete now. It can create really nice (and most importantly, really eighties-like) sounds. It is somewhat CPU-hungry though, so it will be a bit of a challenge to make this run on the Amiga without spending hours on the precalc. Only thing missing now is a way to make drums. No drums, no synthpop.

FEBRUARY 5, 2008

I got my drums. Each of the oscillators now have an adjustable pitch offset and exponential slide down to the original pitch. This addition, combined with adjustable soft-clipping (poor man's compressor) makes for some pretty good kicks. The synth now has everything needed to create a complete portfolio of instruments for a piece of music. The musician has his tool, and I can move on.

FEBRUARY 10, 2008

Started on the new C2P. It seems the CPU part will be significantly smaller than my usual all-CPU C2P (except for the dead-slow but tiny one I was forced to use in Rapo Diablo last year). But with the blitter code and all the CPU/blitter synchronization logic, it will probably be somewhat larger. The transformation itself only takes about 40-50 cycles for 16 pixels, but it takes more than 150 cycles to write the (intermediate) result to chip memory, so I need to find something to do to the data inside the C2P loop. Some kind of postprocessing that does not need any extra memory accesses. Bumpmapping perhaps...

FEBRUARY 22, 2008

C2P finished, in principle, though it doesn't quite work yet. The synchronization mechanisms needed to keep the CPU, blitter and copper going without stepping on each other's toes was trickier than I thought. The copper starts each of the 8 blitter passes, waiting for the blitter to finish between each. The CPU must make sure that the copper only runs when there is data to convert, and itself cannot write into the intermediate buffer until the blitter is done using the data. It all goes well as long as the effect actually runs at full 50fps, but when the CPU is not done rendering the effect in time (which will happen sometimes, at least during development), it locks up, draws something weird and/or crashes.

FEBRUARY 23, 2008

Mini-party with Loonies, TBC, and some others gathered together. Perfect occasion for laying out and discussing plans for Breakpoint (and there are many). Also plenty of internal exposure of my synth: Maytz, Booster, Farfar and Lemmus played with it and were all very excited. Comments like “Are you sure you can get this sound into (or rather out of) an Amiga 4k?” show that I am on the right track here. And yes, I am confident it will work fine (though still a bit worried about the speed of the thing).

FEBRUARY 24, 2008

Finally got the C2P working reliably. I also started on the intro version of the synth. So far it appears it can be relatively compactly done, though it is still far too early to give any size estimates. The rest of the intro so far - intro system, startup, C2P, display code, palette code, and some basic scripting code - weighs in at about 1300 bytes, with no effect code yet, which is a bit more than I had anticipated. This is going to get really crammed...

FEBRUARY 25, 2008

While porting code from the VST to the intro synth, I found a rather serious bug in the random number generator. I had intended to use my usual rotate-dec random generator - rotate the value right by itself and subtract one - which gives a pretty good random for almost no cost. However, the code in the synth rotated 8 rather than by the value itself, resulting in almost the same values being repeated over and over again for every fourth random value. In some sense, this was actually good news, because after I fixed the bug, the synth sounded much better. :-D

DID I MENTION THAT MY SYNTH IS QUITE CPU-HUNGRY? I CAN’T EVEN PLAY THE MUSIC MAYTZ MADE FOR ME IN REALTIME ON MY 1.6GHZ MACHINE

MARCH 1, 2008

Finished the instrument calculation part of the synth (except for the resampling). It is funny how actually writing the code makes the input format fall into place. I had spent some time when writing the VST version to find appropriate quantization levels for each of the parameters to give the parameter a useful range and precision, while making it fit within one byte, and to make it easy and compact for the synth to recreate the value (as an example, all attack/decay/release values are specified as a factor of a power of two samples, so a simple shift will suffice to unfold them). While writing the synth code, I then ordered the parameters within the representation as was most convenient for the code reading them. These five parameters all need to be exponentiated - do that in a loop. These three must be multiplied by a random value - do that in another loop.

MARCH 2, 2008

Maytz has created the first version of the music for the intro, and it sounds great! He has managed to capture that synthpop feeling quite well, I think. He asked how big I think the music he has made will be, and I have no idea. Next to impossible to say before the conversion into the internal intro format is in place.

Did I mention that my synth is quite CPU-hungry? I can’t even play the music Maytz made for me in realtime on my 1.6GHz machine. Maytz has to send it to Booster, who has a registered version of Renoise so he can render it to an mp3 and send it to me so I can hear it. A bit cumbersome, but it works.

MARCH 5, 2008

I am discussing with Maytz which Renoise features he can use. We want to find a reasonable compromise between which features he needs for the music and what can be implemented compactly. These are some of the things we settled on:

» He can use exactly one instrument in each track, and all notes in each track have the same volume. This avoids storing instrument indices for each note. Since there will

not be a limit on the number of tracks, this restriction should not cause any problems.

» The Delay and Stereo Expander effects can be used (they do wonders to the voluminous feel of the music), but all tracks using them have to use both, and all of these tracks have to use exactly the same effect parameters. With this restriction, I can mix all the tracks with the effects on first, apply the effects to the mixed sound, and then mix all the other tracks on top. Saves a lot of juggling around with temporary mixing buffers.

MARCH 8, 2008

Started writing the conversion tool. A Renoise song is simply a zipped XML file, so it is easy to parse. This was a perfect occasion for me to use Xact - a Java transformation library for Java which I was involved in at University. Really convenient for picking out lists of things in the XML file (and in music data there are a lot of lists of things).

MARCH 9, 2008

Finished the conversion tool to a degree where it actually writes out all the instrument parameters and notes.

For each track I store all instrument parameters (remember, only one instrument per track) and then a list of tone/length pairs indicating which notes are played throughout the track. I will just calculate the sound for each of these instrument/tone/length combinations once and then mix these appropriately. This should make the precalculation time for the Amiga version manageable, though I have my fears it will still take a long time. Lots of calculations to do in any case.

For the notes, I store two blocks of data: one with the distances between each note - one byte each with a two-byte encoding of distances larger than 127. And one block with indices into the tone/length table for the track. Each track is stored as single sequences, unfolding the pattern structure of the original Renoise song. This results in a large amount of highly compressible data which is very simple to parse.

And now for the big revelation: how big is the music? The uncompressed data written out from the converter summed up to around 9k. Including this block of data in the intro and compressing it revealed a compressed size of a mere 600 bytes! Very good!

The control code for the instrument calculation and the mixing of the calculated instruments was quickly done. The synth can now produce the entire music from the data supplied by the converter! Not entirely without errors though. It seems some instances of some of the instruments just contain garbage for some reason...

MARCH 10, 2008

Buffer overflow! The garbage in some of the samples was caused by the instrument calculation buffer (which contains floats) overflowing into the buffer holding the instruments ready for mixing (containing shorts). Floats interpreted as shorts do not sound good, I can tell you. Perhaps I should output the needed size for this buffer from the converter to be sure.

I AM IN DEEP TROUBLE. TODAY I TRIED RUNNING THE MUSIC PRECALCULATION ON MY REAL AMIGA. IT TOOK 11 MINUTES

Implemented track volume in the converter. Only the delay/stereo effects are missing now. The instruments still don't sound completely right, though it is very close now. It sounds like the VST and Amiga versions do not quite agree on the random numbers generated. I will have to inspect that random function more closely. The code looks fine as far as I can tell.

MARCH 11, 2008

I am in deep trouble. Today I tried running the music precalculation on my real Amiga. It took 11 minutes. With the Breakpoint compo rules limiting all productions to 8 minutes including precalc, I have minus 3 minutes for the intro itself. :-/

This is too far above the target to be amended by a bit of optimization and tweaking. I will have to do something drastic. The first thing to try should probably be to reduce the oversampling factor from 4 to 2. Hopefully the degradation in quality will not be noticable in the already fairly noisy Amiga version of the synth.

MARCH 13, 2008

About a week until Breakpoint with a synth that is too slow and still nothing on screen. This will be a hectic end race (as usual).

Thinking about how I can cut some corners in the effect code to get something up and running quick. An obvious shortcut is to reuse the coloring and palette code from Noxie, since it will fit this intro fine anyway. No fancy palette tricks this time. Just render to a 16-bit framebuffer (two 8-bit components) and random-dither down to 8 bits interleaved with the C2P (there is the code to put into the available time while writing to chip ram).

MARCH 15, 2008

Easter holiday has begun! Going to Denmark for five days of intense demo coding marathon with no work or anything else to distract.

Implemented the dithering from 16 to 8 bits in the c2p. The two 8-bit values are treated as 5:3 fixed point with clamping of values larger than 15. This will make it possible to do some additive blending in the effect without worrying too much about overflow. 7 bits of precision for the color components should be fine for what I want to show.

MARCH 17, 2008

I found a stupid bug in the sound calculation which means I calculate much more data than needed. After fixing this, the music precalculation time has dropped to 6 minutes. Much better, but still not quite there.

I have also started working on the actual visuals. It is not that the basic voxel tracing is particularly complicated, but there is still a lot of code to write, and it is not going quite as quickly as I would have liked it to.

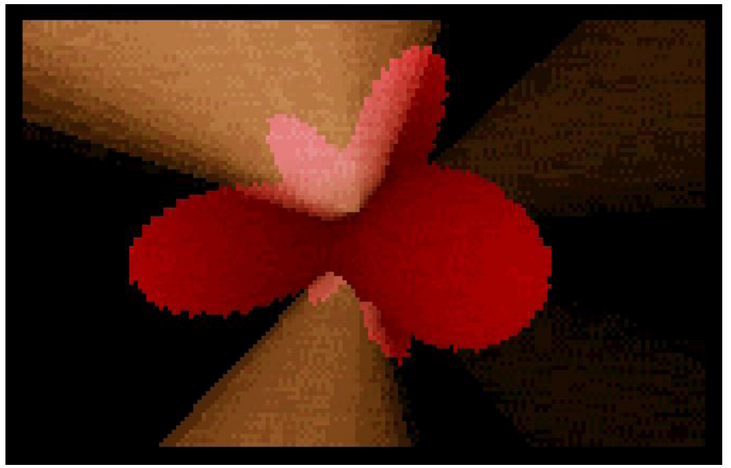
I have joined Lemmus and Mentor for some serious 4k making festivities. Their PC 4k (for which I have made the intro tool - a variant of the tool we used for Candystall) is looking really good. I can't help joining the fun, though I really should concentrate on my own intro...

MARCH 18, 2008

An optimization in the synth - when one instrument is played at the same tone but different lengths, the basic calculation of the sound layers can be shared, and only the attack/decay/sustain/release enveloping and the softclipping need to be performed individually for each length. This costs some space, but it reduces the precalculation time to 4 minutes. This leaves 4 minutes for the intro itself, so it is probably acceptable, though it is still a long time to wait.

MARCH 19, 2008

Yeah! The visuals are working! I now have a beautiful voxel blob rotating on the screen, and with a nice little twist. Since the code is tracing along the voxel object from the center of the screen and radially out, it is a simple matter of marking some voxels as light-emitting and adding a light value along the ray, and you have an illusion of light rays emitted from a light source at the center of the blob through holes in the object. It works pretty well, actually.



The very first fully working visuals - a voxel blob with some simple coloring and light beams

I imagine the light rays can be introduced during a calm portion of the music as vertical beams of light up from a landscape, synchronized to the melody, with different tones sending light up from different holes. Then later, the light beams can be more abundant. A twister with light coming out of holes... Now that's a good thought...

Sitting in the bus on the way to Breakpoint working on the different ways a ray can be traced through the voxels (from one point radially out, from a line perpendicular to the line, etc.) and the different ways the result can be mapped onto the screen (polar or rotated).

MARCH 21, 2008

Oh well, no Amiga 4k from me this year. It became clear as the deadline approached that far too much was missing for this to become an intro. :-(

On the plus side, our PC 4k - Atrium - turned out better than any of us had imagined, and it completely blew away the Breakpoint audience. Yay! :-D

The Amiga 4k will now be put on the shelf to be completed for next year's Breakpoint. With so much done already, and Breakpoint more than a year away, there should be a decent chance that I will finish it this time around. :)

NOVEMBER 2008

Psycho wanted to use my synth for his PC 4k for Kindergarden 2008, so I implemented a PC version of the replayer. It was thrown together in a hurry and turned out somewhat bigger than expected - about 750 bytes for the synth and the same for the music. We managed to find the space in this particular intro, but it will need to be reduced somewhat for future use.

In the compo version of the intro, we accidentally set the resampling filter size way too high, resulting in the loss of all high frequencies in the music, making it sound rather crappy.

FEBRUARY 2009

```
meloBass: E-4:2(33) G-4:1(33) C-5:2(66) E-5:2(33) G-5:2(33) A-5:2(66) C-6:2(66)  longest 0
.514293, total 3.500052
lead1: D-5:12(10) D-5:6(10) D-5:4(10) E-5:6(40) E-5:4(20) F-5:12(30) F-5:10(10) F-5:6(20)
G-5:14(17) G-5:6(13) G-5:4(18) A-5:6(50) A-5:4(20)  longest 1.911189, total 16.045452
lead2: F-5:4(10) G-5:12(10) G-5:6(46) G-5:4(18) A-5:12(28) A-5:10(10) A-5:6(18) B-5:14(15)
 B-5:8(4) B-5:6(9) B-5:4(9) C-6:6(46) C-6:4(27)  longest 1.911189, total 16.245452
lead3: A-5:4(10) B-5:12(10) B-5:6(38) B-5:4(19) C-6:12(29) C-6:10(10) C-6:6(29) D-6:14(16)
 D-6:10(4) D-6:6(9) D-6:4(9) E-6:6(48) E-6:4(28)  longest 1.911189, total 16.445452
hihat: A-4:2(240) A-4:1(240)  longest 0.326347, total 0.552695
hihat delay: A-4:2(240) A-4:1(240)  longest 0.326347, total 0.552695
clap1: E-4:1(160)  longest 0.118950, total 0.118950
drum: C-3:1(214) C-4:1(123)  longest 0.118950, total 0.237900
clap2: E-4:1(160)  longest 0.476942, total 0.476942
bass: F-2:16(3) F-2:2(76) G-2:16(1) G-2:2(76) A-2:16(3) A-2:2(76) C-3:8(1) C-3:2(22) D-3:1
6(3) D-3:2(100) F-3:2(100) G-3:2(100) A-3:2(76) C-4:2(22) D-4:2(100) F-4:2(24) G-4:2(24)
longest 1.663699, total 10.682879
melody: G-6:2(42) A-6:4(17) A-6:2(104) B-6:2(53) C-7:4(2) C-7:2(88) D-7:6(13) D-7:4(7) D-7
:2(30) E-7:2(38)  longest 1.111189, total 8.111886
Max: longest 1.911189, total 16.445452

Wrote file Music.S
```

Output from the converter tool, giving some statistics on note usage in the music

Not exactly a glorious first-time exposure for the synth. Note to self: Don't adjust filter parameters during the gabber music compo.

I have reimplemented the conversion tool in Python, making it much smaller and more maintainable than the Java version. And it is even faster too! So much for our nice Java XML API...

The conversion tool prints a list of tones and lengths used in each track, along with the number of times each combination is used. This information is useful when size-optimizing the music, since if we can identify notes which are only played once and perhaps eliminate them, the number of different tone/length pairs can be reduced, often reducing the size considerably with only minor changes to the music.

MARCH 7, 2009

Time to start coding again! (Really, really!) Breakpoint is now only 5 weeks away. I have taken a brief look through the code to refresh my memory. Most of the effect code is there (except for the background effect), but it is pretty big at the moment - just 100 bytes shy of 4 kilobytes, which is far too much considering what is still missing. It will need some thorough rewriting in a more careful manner than the panicked oh-no-Breakpoint-

is-tomorrow code I wrote last year.

I also have a couple of new ideas for the effect, mainly to do with lighting and shadowing of the voxel forms. I envision that this is going to be really delicious. :)

MARCH 9, 2009

First sacrifice: stereo. The stereo effect obtained by using separate random sequences for each channel is quite cool, but it has a high cost both in code space and calculation time, and neither is exactly plentiful.

To estimate how much the sound quality would be affected by a reduction to mono, I changed the playback code to mix the stereo signal into mono whenever the right mouse button was pressed, so I could switch quickly between them throughout the music. In headphones, the difference was quite dramatic, but on speakers it was much less noticeable, especially when many tracks were playing. As the intro is primarily meant to be enjoyed on speakers, the conclusion is that stereo can be left out.

I removed the stereo loop and did some code simplifications which was made possible by the change, and saved a whopping 144 bytes on the compressed size! A couple more of those kinds of optimizations and I might even stop panicking about available space.

As a very nice side effect, the precalculation time has been cut in half - down to two minutes, which is definitely acceptable.

The synth mixes the music in 16-bit precision, and the result is converted to 8 bits for each of the 4 hardware channels during playback, using different rounding for each, giving effectively 10 bits of output precision. I tried to simplify this to producing only one output channel and playing this in all 4 channels, corresponding to 8 bits of output precision, but that sounded like crap. Bad idea.

MARCH 17, 2009

Code cleanup commencing... I had three versions of the voxel tracer: one for mapping from the left edge to the right edge of the screen (well, not exactly the screen, but the voxel rendering buffer which is later mapped onto the screen), one for mapping from the center to the left edge, and one for mapping from the center to the right edge. I have now collapsed these into one function with some extra arguments (a bit more tricky than it sounds when you are already out of registers). This saved some space and, most importantly, made the code much more maintainable.

MARCH 18, 2009

Took a closer look at the scripting code. The purpose of this code is to control the values of all parameters throughout the intro. Even though the particular scripting strategy used was invented in a panic shortly before the deadline last year, I think it will work quite well, and I will use it mostly unmodified.

It goes like this: the timeline is split into blocks of variable length. In each block, each parameter has a start value, an end value, a trigger instrument, and a trigger strength. The scripting code interpolates between the start and end values and adds a wobble-shaped pulse (sine times exponential decay) each time the particular instrument is played. Should work well for making lively, pulsing voxels.

The voxel tracer traces through two height maps simultaneously, multiplying their heights by different parameters. Changing these parameters independently will give an impression of the voxels changing shape, rather than just changing size. I particularly imagine this giving a nice effect on the landscape: when the bass reenters the music, the two height maps will pulse alternately, giving the impression that the landscape is wiggling to the music.

MARCH 22, 2009

We are meeting for a whole weekend of Loonies demo action. We have been working pretty efficiently (that varies a lot between such meetings), Psycho working on his PC 4k intro, me working on the Amiga 4k, Booster making music for Psycho (using the very same synth). I have worked through the in-

tro visuals, having now well-working versions of several components:

» The background effect. The voxel tracer simply continues along its route in the height map, but instead of drawing voxels, it uses the height to distort a color gradient. This way, the background will sort of move with the foreground voxels. The distortion strength is controlled be a parameter, so the background can twitch to the music.

» The twister. It is twisting and everything! Though it turns out that the light beams don't work well with the twister - it looks weird when they cross over the middle. We will just have light beams in the landscape and the blob, then.

» The landscape. Finding the right perspective parameters to make the landscape look interesting on screen required a lot of tweaking.

A major thing still missing is interesting height maps. One of the challenges here is to create height maps with holes in them with some kind of identifiers on the holes so the light can come out of different holes on different tones. Running through the generated height map to identify the hole regions and put identifiers on them is likely to require far too much code.

MARCH 25, 2009

Back home, it was time to try out the effect on my real Amiga to make sure it was running oneframe. I did this, and it was - some of the time. It is quite sensitive to the direction in memory in which the voxel tracing and the screen mapping are reading. For instance, the twister runs fine as long as its axis is not close to horizontal. That is no problem of course. We will just make sure it does not come close to horizontal in the final intro.

The twister with light beams. It looks a bit weird that the beams switch direction abruptly when the holes cross the middle (and even go both ways for a brief instant)

Still, I took the opportunity to optimize several things. The screen area was made slightly smaller (that was easy), the height maps were interleaved so that tracing in two maps in parallel is more memory bandwidth efficient, and a few other things.

I experimented a bit with the coloring of the voxels. Making the voxels darker further along the ray looked good (and is necessary for the landscape and tunnel anyway, in order to make the place where the voxels pop up hidden in the shadowy mist). But apart from that, I haven't yet found a satisfactory formula. Coloring should somehow be based on the height maps (having a separate color texture is out of the question), but at the same time, it should not just be the plain height coloring seen in so many classical voxel landscapes.

MARCH 28, 2009

More coloring experiments. I would like to have some kind of lighting on the voxels, but it is not obvious how to do so since

the voxels don't have normals - not even depth.

I tried to fake a normal calculation by subtracting contiguous voxel heights, and that sort of worked. The lighting was there alright, but there was a lot of noise in it because of the low precision of the method. Certainly looks better without it.

MARCH 31, 2009

Now I know how to make height maps! Simply add together a lot of bumps at random positions with random sizes and heights. By specifying the ranges of sizes and heights and trying out different randomseeds, I should be able to find some good-looking ones. The bump height can be negative, and if that happens to make the total height go negative, meaning it punched a hole in the height map, the index of the bump is used as an identifier for the hole. Simple simple.

I will need a tool for quickly previewing the height maps. I might be a bit oldschool in my thinking here, but I find the most suitable language for such a preview tool to be AMOS. So I made a height map calculator in AMOS, making sure it would calculate the exact same height maps as the asm code (which I have yet to write).

A height map as shown by the height map preview tool. When viewed like this, the individual bumps that make up the height map are clearly visible

APRIL 2, 2009

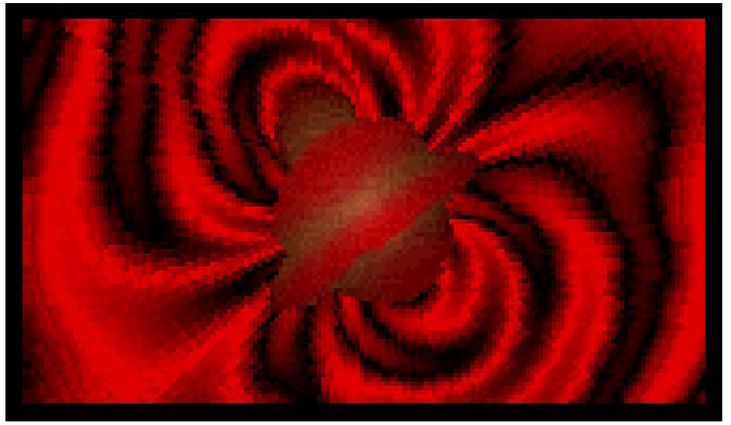
Finally a coloring formula I am satisfied with. Sometimes the simple solutions are the best: Make the two color components different linear combinations of the two height maps and ping-pong wrap them into the valid interval. You are probably thinking this sounds like a horrible coder colors recipe, and you would be right if it wasn't for the fact that the actual colors are controlled by a palette afterwards.

APRIL 3, 2009

Going to Denmark for some pre-easter family visits (and intro making, of course). Implemented the height map generator in the train. It is not exactly tiny - the intro is starting to get a bit crammed again...

APRIL 7, 2009

Met with Maytz to get the look and form of the intro into place. We did some adjustments to the music and, most importantly, to the background effect: as Maytz pointed out, the existing background effect was much too detailed and hectic. It took away the attention from the foreground effect, making the overall impression a bit chaotic. We replaced it by a simple color gradient (still based on the trace coordinates so it moves with the voxels) and also added a parameter to make it darker (rather than making it twitch). The end result was much better balanced, and the code was smaller and faster as well. More of that, please!



The original background effect. It looks cool, but you barely notice the voxel effect in the middle which is supposed to be the center of attention

APRIL 8, 2009

The last two planned effects - the tunnel and the torus twister - are working now. Just scripting and tweaking (and size optimizations as necessary) left now. I think this is the best situation we have ever been in for a 4k before the start of the party. :)

APRIL 9, 2009

Axel is bugging us for the name of the intro, since he needs it for the Zine headlines demo for Breakpoint.

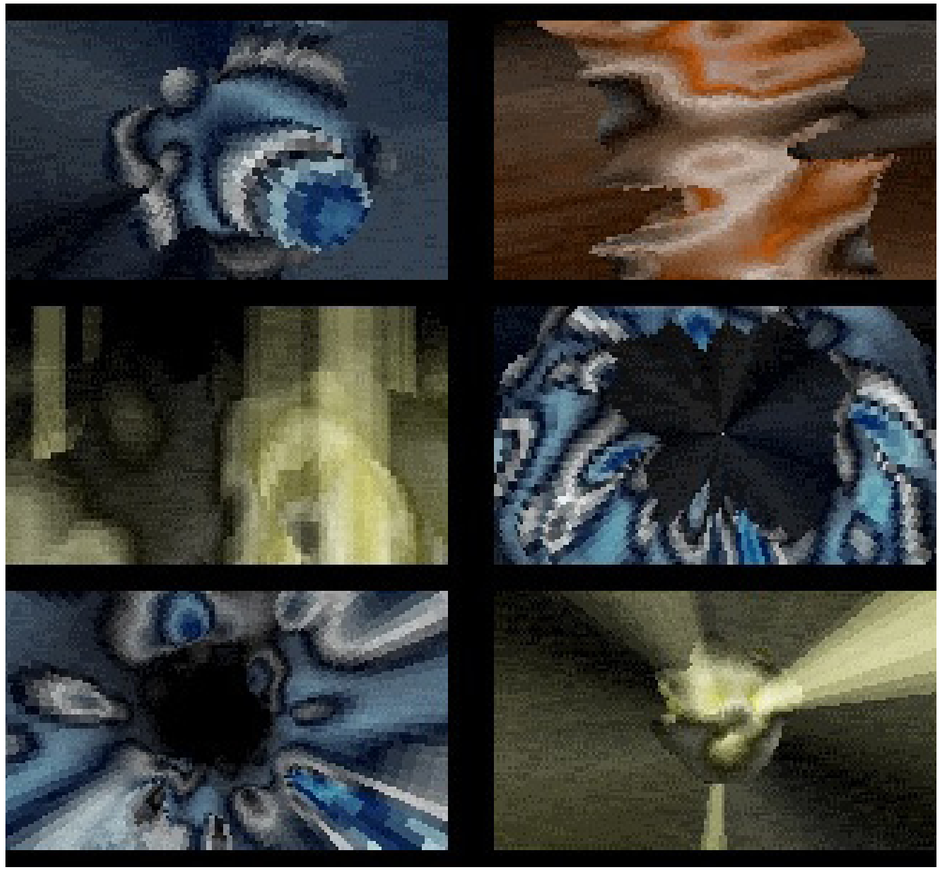
My original idea for the name was “Luminaris”. Tasting of the word for some more months changed it to “Luminaria”. I googled a bit yesterday and figured that a luminaria is a small paper lamp - somewhat, but not exactly, fitting to the look of the intro. The ideal would be some word that did not exist before. Something that gave zero hits on Google. After a few tries (and more tasting) I found it: “Luminagia”. There is a bit of a magic feeling about it I think. Plus you can spell “Amiga” from some of the letters. ;)

INTRO DONE AND DELIVERED. ALL SET FOR THE COMPO

I asked Maytz what he thought about it, and his reply was: “Sounds like Viagra.” Well, if it also works like Viagra, we have at least done something right...

APRIL 10, 2009

After the first day of party coding (intermixed with bits of the real party - wow, best Breakpoint weather ever), we have a draft script and decent height maps. I also went through all the code and inlined everything that could be inlined (macros ftw). This saved quite some space and we are now ever so slightly below 4k. This is going to be an act of balance adding the rest while staying under 4k, but it is in no way critical.



Screenshots from the final intro

Only colors and parameter tweaking missing now. Plenty of time for that before the “deadline” (that is, the real deadline) tomorrow. For the first time ever, I will actually be able to get something akin to a full night’s sleep friday night on Breakpoint! Mission accomplished.

APRIL 11, 2009

Another relaxed day of party coding, and the intro is finished, almost an hour before the “deadline”.

As we went to the recording room to hand in the executable, it was not Charlie (the Amiga compo organizer) opening the door for us. In the recording room we found Loaderror, sitting by himself, coding a last-minute 4k for the compo. It turned out there were only two entries handed in - one less than the minimum for the compo to be a reality. Hence, Loaderror had taken it upon himself to save the compo by quickly throwing together a... well, something.

Intro done and delivered. All set for the compo.

Looking back at the process of this intro coming into existence, it occurs to me that it has been a bit atypical as such 4k intros go. Even though the intro has been more or less underway for more than one and a half years, the end result turned out quite close to my original ideas. Also, contrary to the norm, there wasn't really any major feature that had to be left out this time (well, except the stereo, but that didn't really matter).

Though the intro was in no way finished as we arrived at the party, the party coding part was quite relaxed. It was simply a matter of finishing the intro and then handing it in. There was no scarcity of time either.

I wonder what was different this time. Maybe I am getting older... ;)

The reception of the intro was very positive, which is always nice. Still, it feels like there is nothing really remarkable about this intro. It is simply very good in all areas. But fairly traditional overall.

Next year - oh well, now that is the question: will we be back again next year? It sometimes feels like this is my last Amiga 4k intro, but on the other hand, I don't yet see anything that could cure me of my addiction, so of course we will be back next year! Anyway, next year we will see if we can throw together something a bit more unconventional, and we are just as curious as to what that might be as you are. Let the journey begin...