Walking On Four
The Report
20060701

Nicolas Léveillé <nicolas.leveille@free.fr>

Contents

Part I

Presentation

1 Introduction

In September 2005, even before the release of *Clothing Like A T-Shirt Saying I Wish* I already had set my mind on releasing short intros for the next events of the year. Namely, the Saturne Party 6 in Paris, and the bcnparty101[9] in Barcelona. It turned out the cancellation of the former was actually beneficial to the overall project, since going back to work on an intro immediately after another one proved a bit too optimistic.

After some hectic crunch-time programming for the streamMega[10] party, I needed, or thought I needed a bit of rest. The code base also had suffered through the ordeal and some housekeeping needed to be done to clean up and package independent parts for later works.

So I cleaned up the mess, branched out the code, and started working again on it for the next effort.

In the end, after roughly 67 *hours* of work with a great deal taken by debugging, *Walking On Four* was released in the 64kb intro competition at bcnparty101[9], on 2006.11.05:

> Walking on four
>
> !=
>
> 70051104 @1737 Barcelona / Spain
>
> –
>
> Learn walking on four legs again. Height and surface then becomes all the more interesting.
>
> –
>
> Lorsque la patisserie arriva enfin, elle tronait immense et rustre, Ou plutôt, elle encombrait l'assiette sur laquelle m'était présentée. Pas d'embellissement: on aurait dit le produit d'un blasé de la patisserie, d'un agent ennemi tentant de saboter la discipline. J'échauffais ma machoire, et plongeait néanmoins mes dents dans l'arme de destruction diététique massive. Le retour fut fulgurant. Une vague partant de l'interieur de la bouche alla et revint dans mon corps, comme si j'avais posé la main par mégarde sur une cloture électrifiée. Impossible de savoir si c'était agréable ou désagréable .. pas même après avoir avalé la dernière bouchée.
>
> –

Contact info: tpolm@tpolm.org

Maybe all I needed was a reason to visit the intoxicating city of Barcelona, and its illustrious inhabitants.

2 Description

Starting with a fast moving red loading bar, it then displays the very same effect that ended *Clothing Like A Tshirt saying I Wish*, arrows going up and down diagonally across the screen.

But we notice a slight ghosting of the whole picture.

The soundtrack is distinctively electronic. The pad section is playing a seemingly aimless minor harmony, and the rhythmic section, consisting of just a bass drum and a click, is animated along a long rhythm difficult to apprehend at first.

As the synthetic pads slowly fade-in, stripes, almost calligraphic drawings start appearing and drawing themselves below the frame. The grey on dark drawings do not appear to mean anything, and appear one after another, as pieces of entangled ribbons.

Very quickly the original arrows disappear, and after a short display of the ribbons alone, a distorted pink plane of particles jumps in and out of the screen, locally swelling.

Around the 45 seconds mark, the display spreads and saturate into almost homogeneous whiteness. But slowly ripples back in to display another similar screen: transparent ribbons like in the first half, but on top of which a rotating, block of pink lights, pulsating with the underlying rhythm of the music.

With time, ripples form and emerge from the block of light. Saturated, harsh-looking ripples coming from the center of the screen towards its sides, like a suggested tunnel.

In the end, around the 90 seconds mark, the title "*walking on four* –" appears on a black display, followed in an unusual down to up order, with "'!=" then "BCN 7005" as each new beat of the now almost alone rhythm resonates in.

Part II
Design and Methodology

Let's break it to you immediately: there is no concept behind the creation of *Walking on Four*. It became what it became through a collage of ideas and accidents: it first started as an effort to compensate the dryness of my previous

intro, especially in the visual area. Focusing on what demos are made of, effects.

I wanted to come back to the basics, to accept seeing things from the floor like a child does before trying to walk. Mundane things then become tall and impressive. But more than a regression or process of unlearning, it's about coming back to a wild state of just obeying one's instinct. Creation as the exposure of one's desires[6].

That said, one of the ideas that served my purpose was the reuse of a common core for almost all of the visual effects. This made it possible to implement the intro quickly, giving me enough time to sort out all the technical issues I encountered.

This common core, created as a simulation of a simplistic physical system, is reinterpreted through a progression of visual effects: at first, the piece shows up representation of the inner workings of this simulation, – as moving ribbons – then followed by a static result of its operation, – the swelling plane – to finish with a more elaborate use, showing it from "outside". – radiating lights –

Inside, outside.

This use of fake physics as base for visual effects is one of my favourite themes. As far as I remember, I first started playing with it in an unreleased demo for Assembly 2000[11], initially inspired as I was by Golan Levin's Floo[1].

To underline the continuity of work between my two efforts, the very last effect of *Clothing Like A T-Shirt Saying I Wish* was turned into the very first effect of *Walking on Four*.

I won't repeat here what I wrote earlier[5], but again, time was a critical resource to meet the BCN Party's deadline, with around 33 days of work available. ($33\ days = 33 * 24 - 20 * 9 - 33 * 8 - 33 * 2 - 4 * 2 = 274\ hours$) I committed 67 *hours* of work in the end, for a ratio of 0.24 down from my previous ratio of $42/90 = 0.47$. Life is life.

Part III

Implementation

3 Materials

Implementation work consisted of two big parts: First I tried enhancing some of the representations I used previously for music and rhythms. Representation of how one can express temporal relationships in the timeline of a piece. In parallel, I built a few *classic-looking* visual effects.

In an intro, it is always better if we can do more with less. This is precisely

why, under the hood I separated my visual effects in two:

The first part, the simulation, is responsible for creating complexity. The second part interprets or otherwise puts into form the result of the first part.

I liked the idea of making a link between this and the previous intro. Here the reuse of the last effect and presentation from *Clothing Like A T-Shirt Saying I Wish* as initial scene serves to create an illusion of a continuity.

4 Rhythmic patterns

4.1 Or how I sought increasingly complex rhythms from simple rules.

First, what is the simplest rhythm? A regular, repetitive beat. Like one's heart beating. Or somebody clapping.

Clapping in a regular fashion is something perfectly natural to us, and it does not take much effort to do it.

So at the core of our system is the Metronome: a regular beat with a recognisable frequency. It shouldn't have to be very precise or very regular, but computers being what they are, the simplest implementation is very strict.

Rhythm is important in music: it brings about a mental image of movements, of physical actions being performed. And rhythms, as they appear in nature, are rarely fully regular. The song of a bird, the beating of wind in a tree. Water as it breaks on the rocks of a river's bed.

So lets now add more twists to our simplistic rhythm.

If instead of clapping we used our fingers to count down as each regular step comes, clapping only when no finger is left in our counting hand, we gain a flexible way of marking time.

Slowing down our regular beat: 2 1 *clap* 2 1 *clap*.

We will write this rhythm down as:

[3 3]

A clap, then 3 steps later, another clap.

If we were to write an irregular rhythm: 2 1 *clap* 1 *clap*

We would again, write it down as:

[2 3]

The notation represents a rhythmic pattern of n different beats as: $[a_1 \cdots a_n]$. Each a_i represents the number of ticks (steps) each beat lasts. The number of

ticks per second is controlled by a metronome, and defines the overall scale of the rhythm.

We also define the period of a rhythmic pattern in ticks as:

$$\text{Period}([a_1 \cdots a_n]) = \sum_1^n a_i$$

When a rhythmic pattern forms a full rhythm, the notation also implies that:

$$[a_1 \cdots a_n] = [a_1 \cdots a_n \; a_1 \cdots a_n] = [a_1 \cdots a_n \; a_1 \cdots a_n \cdots]$$

Which enables us to simplify [3 3] as [3]

In this notation, rhythms are defined as a series of beats of varying length. This presumes an underlying Metronome, that gives the basic speed of the final rhythm.

We will now define an agglutination operation (sticking two proto-rhythms together) : +

$$r + [] = r$$
$$[a_1 \cdots a_n] + [b_1 \cdots b_n] = [a_1 \cdots a_n \; b_1 \cdots b_n]$$

a special rhythmic pattern, [] is an utterly useless, beatless rhythm. (Hardly a rhythm at all)

4.2 Elementary patterns

A number of proto-rhythms are used in *Walking on Four*:

$$p = [1]$$
$$pp = [2]$$
$$e5 = [5]$$
$$r7 = [2\ 7]$$
$$lr = [2\ 14]$$
$$t3 = [3\ 2\ 2]$$
$$j4 = [4\ 1\ 2]$$
$$j9 = [9\ 1\ 2]$$
$$m4 = [4\ 2\ 4]$$
$$a5 = [5\ 5\ 2\ 2]$$

To form the following rhythms:

$$\begin{array}{lll} pe5 & = [\text{Period}(e5)] & = [5] \\ pj4 & = [\text{Period}(j4)] & = [7] \\ pm4 & = [\text{Period}(m4)] & = [10] \\ pa5r7 & = [\text{Period}(a5) + \text{Period}(r7)] & = [23] \\ pa5j9 & = [\text{Period}(a5) + \text{Period}(j9)] & = [26] \end{array}$$

t3ppp	= t3 + p + pp	= [3 2 2 1 2]
a5j9a5r7	= a5 + j9 + a5 + r7	= [5 5 2 2 9 1 2 5 5 2 2 2 7]
A5j9A5r7	= pa5j9 + pa5r7	= [26 23]
m4j4e5	= m4 + j4 + e5	= [4 2 4 4 1 2 5]
M4J4E5	= pm4 + pj4 + pe5	= [10 7 5]

Each of those rhythms may be used in different parts, visual or musical ones, and with a different underlying metronome.

We will nevertheless notice that was is missing from this system is a way to emphasise each beat differently: there is no notion of accents, and it must be emulated by hand by layering multiple rhythms together, hence the presence of M4J4E5 together with m4j4e5, or A5j9A5r7 together with a5j9a5t7.

5 Tone

5.1 Modes

Generating notes at random, like I did previously produces feelings of unresolution, aimlessness, but when done completely at random, fails to have any sort of harmonic character. No atmosphere, except that of a sterile, if somewhat dissonant mechanism.

To obtain a degree of control on this, I wanted to reduce the set of notes – to work in a given *mode* – selected by the generation algorithms. The relative, average interval between notes will enable or disable certain consonance or dissonances.

The intro would still feature a random generator as a base for note generation, but this time only selecting among notes from certain sets, while also being tweaked in favour of certain preferred sequences.

Modes are specified as a set of notes indexed by their semi tones interval from a given root:

An example would be : $c, \{0\ 1\ 3\}$ which would represent the following set of notes: c, $c\sharp$ and $d\sharp$.

The octave is not taken into account here.

For reference:

c	$c\sharp$	d	$d\sharp$	e	f	$f\sharp$	g	$g\sharp$	a	$a\sharp$	b
0	1	2	3	4	5	6	7	8	9	10	11

In *Walking on Four* I picked

$$c, 0, 1, 3, 5, 7, 8, 10 \Leftrightarrow (c, c\sharp, d\sharp, f, g, g\sharp, a\sharp)$$

Which would otherwise be known as C Phrygian[7]. It comprises minor chords, giving a slightly dissonant atmosphere.

5.2 Note generator

The note generator was specified in terms of probabilities of notes, but also evolves as time does, going up or down that scale, following the $a5j9a5r7$ (4.2) rhythm. (at frequency: $\frac{120}{16} = 7.5\ \frac{ticks}{second}$).

$$\begin{aligned} i &= 7 + P1 + ((\sigma step + P3)\ mod\ R) \\ midi\_note &= C\ Phrygian(i) \end{aligned}$$

Where $P1$, $P3$, R, $step$, σ, $exprand$ and $prob$ are defined as:
σ alternating between -1 and $+1$ along a [2 14] (lr) rhythm.

$$P1 = \text{Prob}\left\langle \begin{array}{rr} 43.55\% : & -1 \\ 33\% : & 7 \\ 23.45\% : & 0 \end{array} \right\rangle$$

$$P3 = \text{Prob}\left\langle \begin{array}{rr} 62\% : & \sigma * 5 \\ 38\% : & 21 * exprand \end{array} \right\rangle$$

$$R = \text{Prob}\left\langle \begin{array}{rr} 51\% : & 7 \\ 49\% : & 21 \end{array} \right\rangle$$

$step \in [0\ 1 \cdots + \infty]$ increases with each beat of the underlying rhythm.
$exprand$ produces an exponential distribution of random numbers.
$\text{Prob}\langle percentage : value \cdots \rangle$ returns a value with a given probability.

The idea is to generate a pattern of notes with a globally raising (but sometimes descending) motif, with notes wrapped around an alternating register R of 7 to 21 semi-tones.

The atonal instruments like the click and bassdrums are controlled directly by the following rhythms:

clicks	t3ppp
bassdrum	a5j9a5r7 accented via A5j9A5r7

6 Video filters

I've always loved feedback-based effect, and the patterns that emerge from their often chaotic behaviour. I tried to capture this via the implementation of an IIR filter in video.

This is a transposition in video of the simple audio low pass filter I use in the synthesiser. But with one problem: the sampling rate of video is quite small (50-60hz) and non homogeneous as well. Both factors make the results of the filter quite difficult to control. Nevertheless, we get interesting results.

Frames are saved, rendered into six different textures, three capturing the input frames, while the three others capture the output frames. In turn, if one texture stands for the current frame at one point in time, during the next frame that texture will represent the past. When the texture is too old, it now again becomes the present time frame.

Using `OpenGL` it comes down to using glCopyTexSubImage2D to copy the currently rendered picture to a texture.

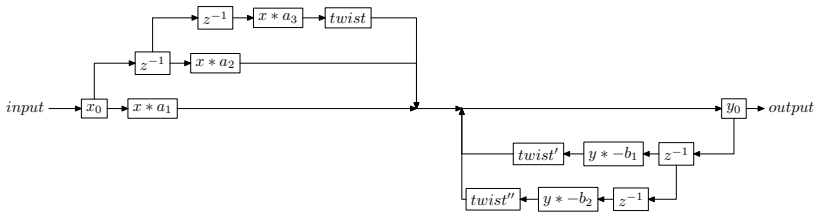

Figure 1: Video low-pass filter. z^{-1} delays a video frame by one frame.

Then multiplication and addition of frames is achieved through blending, using the filter's parameters as alpha channels for each frame, with:

```
glBlendFunc (GL_SRC_ALPHA, GL_ONE);
```

For the output stage, an `OpenGL` extension, EXT_blend_subtract is used to apply negative alpha channels:

```
glBlendEquationEXT (GL_FUNC_REVERSE_SUBTRACT);
```

The filter is also combined with a rotating and zooming stage to give it a small twist: the most dramatic effect is achieved towards the end of the intro, when the filter produces patterns resembling a tunnel.

7 A Simulation

I am always following many different things at a time, and I especially like to look around to see what others are doing. So, watching out for pieces made with

the processing[3] software package, I stumbled on Quasimondo's portfolio[4] and one example especially attracted my eyes: his caustics simulator, *Raycoaster*[2]. Caustics are the subtle light effects one can experience in a swimming pool, when the refracted light rays accumulates in certain areas and thus produce almost geometrical patterns.

The original idea was kept: a simulation where light is represented by particles, grains projected inside a chamber with varying properties, and whose resulting trajectories form a final image. This had to form a source material for the actual visual effects.

My simulation is formed of two subparts. The chamber, where the simulation is performed, which ends with a projection screen. And the particle emitter, which defines how grains of lights are injected in the chamber, and their initial conditions.

7.1 Chamber

The chamber was designed as a stack of different media, each with their different properties. Each medium is composed of elements that pulls, pushes or otherwise affect the trajectory of grains inside them, until the grain exits to the next media.

Let X, Y, Z be three axis. X, Y form planes parallel to the final projection screen. The corresponding coordinates of a grain of light are x, y and z.

One medium is defined by:

- an elevation ($h \in]0; +\infty[$) : this tells us where in the stack it will be, and consequently, how large it will be. (Coordinate in the Z axis)
- a "speed of light" ($c \in]-\infty; +\infty[$) : this controls the z coordinate of each grain of light: $z(t) = z0 + c * |\langle x(t), y(t)\rangle|$ note that the z coordinate is controlled by the length of the path of the particle in the plane.
- a number of force fields in the X, Y plane. Each force contributes to the instantaneous acceleration of the grain depending on its position, following Newton's formula: $\frac{d^2\langle x,y\rangle}{dt} = \frac{1}{m} * \sum F_i(x, y)$ with $m = 1$

The force fields are of two main kind:

Let $\langle x', y'\rangle$ be the position in the field's referential. The force field is here centered around $\langle 0, 0\rangle$ with $f \in]-\infty; +\infty[$

- One of them is the attraction force

$$F(x', y') = f * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} * \frac{\langle x', y'\rangle}{|\langle x', y'\rangle|^2}$$

- The other type is the rotation force

$$F(x', y') = f * \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} * \frac{\langle x', y' \rangle}{|\langle x', y' \rangle|^2}$$

  Which tends to rotate the grain around its center.

When f is below zero, the forces appear to be attractive, when above zero repulsive.

By then, if you have any elementary knowledge of physics you will have realized this is not a sound physical simulation of a known, real phenomenom. Indeed, very rightly so. Modelling photons as if they were balls on a pool table would not be acceptable. But we are not interested here in modelling reality, but to produce material of a certain degree of complexity while being easy to manipulate, and staying continuous.

It is very easy to create different force fields, and even animate them by animating the sources of rotational and attractive forces, or changing their magnitudes.

7.2 The ribbon scene

Contrary to the other two ones, this scene displays the inside of the simulation. Instead of taking its result, we are following the trajectory of grains sent inside the chamber. Here, their inertia / speed is being tweaked so that each force fields affect their trajectory in a much stronger way than in the actual simulation, but the principles and parameters for each force fields are the same.

Splines aren't even used! We get second order continuity for free, by virtue of integrating the Newtonian equations, with continuously varying forces.

Particles are injected in sequence from starting point around the screen via a simple sine based equation.

Every effect starts with a sine after all.

As they get modified by the simulation the grains record their own trajectory: the history of their position, speed and acceleration.

To display them, we thus sampling the path of each grain. At each step in the trajectory, the path is drawn by going through through this history, connecting each point via a quadrilateral, slightly slanted from the horizontal to give the drawing a calligraphic look. The width of each quad of light is also mapped on the velocity of the particle at this position, the faster it was, the thinner the ribbon appears around this part.

7.3 Generating animated textures with `OpenGL`

Throwing all my grains of light through a weird simulation of medium/light interaction was initially devised as a way to generate an animated texture. A texture which would be later used in other effects throughout the intro.

At the origin of any grain, we have a particle emitter. It generates grains over a whole area, spaced out regularly. But covering the screen in a natural way is not so straightforward. I first used a regularly spaced grid, but then the grid would show up in the resulting texture as persistently rectangular features. In order to fix it, a Fibonacci-lattice[8] was used instead.

A Fibonacci-lattice of rank 1 is a two dimensional sequence defined from the Fibonacci sequence F as:

$$FL1_i = \begin{cases} \begin{bmatrix} i\frac{1}{F_k} \\ i\frac{F_{k-1}}{F_k} \end{bmatrix} & i \in [0 \cdots F_k] \\ \textit{wrapped inside the rectangle} : \ \langle 0,0\rangle \cdots \langle 1,1\rangle \end{cases}$$

With F:

$$F_i = \begin{cases} 0 & i = 0 \\ 1 & i = 1 \\ F_{i-1} + F_{i-2} & i > 1 \end{cases}$$

Choosing k here means choosing the number of points we can generate inside the plane. I chose $k = 9$, for $55 + 1$ points.

Now, each grain is injected in the chamber, and its movements calculated as it goes through it. As grains encounter medium changes, or the final screen they release a new particle, called a `ParticleImpact`, which shows up as an additive Gaussian grain of red/pink light.

And as we change the simulation parameters with time (for example by moving force fields around) we can produce an animation.

Now we have to capture the resulting frames to standard `OpenGL` textures.

An extension enables precisely that: the EXT_framebuffer_object extension, also abbreviated as FBO extension. It was introduced in 2005 to replace the platform specific pbuffer extension. Pbuffers have a crufty, slightly different interface for each operating system.

This extension appeared rather late in `OpenGL` drivers, first for `NVIDIA` cards then later in the year for `ATI` cards. Support for older card from other manufacturers is unfortunately absent.

So, running the simulation multiple times with slightly varying forces then capturing the result in a texture resulted in an animation of 6 textures, that I would later use in the visual effects.

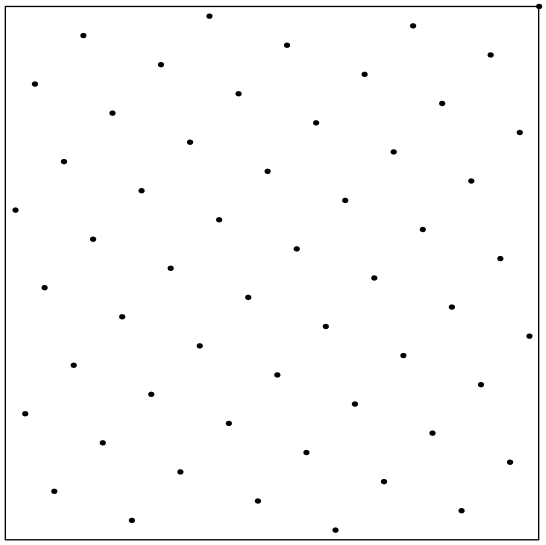

Figure 2: Fibonacci Lattice k = 9.

Of course a great deal of time was lost tracking down a bug that led me defining non-power-of-two textures, which were silently accepted by one driver while leaving the other drivers crawling to a halt. I also did experiment a bit with texture compression, but it would reliably crash as the sets of frames in the animated texture grew. Scaling down the animation to a reasonable number was in the end considered more compatible.

7.4 Morph-grid

This is one of the big classics of demo effects. It is the basis of tunnel effects, and most of the 2d transformations seen in demos of the mid 1990s.

The design involves only one component, but remains very flexible:

A grid of cells is laid out over the screen. Cells are fixed in the screen, and composed of four vertices. Each vertex holds a set of input coordinates. A typical meaning for input coordinates is to represent the coordinates in an input texture. Thus, each quadrilateral maps over the screen a certain area from the input texture. The whole grid enables us, by using varied coordinates in space and time to deform an input texture.

This grid forms a sampling of a more general, pre-calculated function.

This texture can be animated as I did here: the input texture comes from a number of frames rendered from the simulation.

Animating the grid itself is a useful tool. The method I adopted is a very simple

cycle:

1. one optional step of deformation.
2. one step of relaxation.

The deformation is customised, but here corresponded with an expansion of the grid's input coordinates around a continuously moving center. To avoid too great of a change, this deformation is only applied at certain points in time, here with a frequency of $\frac{120}{8} = 15\ \frac{ticks}{seconds}$. One can compose deformations rather freely, and another step was added, zooming the input texture in and out in alternation, at a frequency of $\frac{120}{96} = 1.25\frac{ticks}{seconds}$.

The relaxation gradually brings the grid to its resting state: a plain grid mapping each point of the screen to its corresponding point in the texture.

All of this results in two turgescences that appear to rotate over the screen, as we apply two expansion deformations around two different, rotating centers. Each deformation is rotating in opposite directions around the center, about 2/3 away from the screen's center. The full rotations take about 12 seconds each to go around the screen.

7.5 Radiating lights

This effect aims to imitate movings lights as they would appear through a dense, fog like atmosphere. It uses the animation created during the simulation in a different way, here in three dimension.

At the center of the screen, a small mapped cube is drawn. This cube moves on its own, pushed regularly by forces in opposite directions, like somebody was punching it from side to side. The rhythm was already mentioned: $m4j4e5$ (4.2), with a frequency of $\frac{120}{64} = 1.875\frac{ticks}{seconds}$). Another rhythm is used to control a greater impact of the "punch" each time the accenting rhythm $M4J4E5$ (4.2) produces its beats.

The cube is mapped via `OpenGL` 's spherical mapping, which is normally used for environment mapping. It is then recopied many times at greater and greater scales, transparently.

The bigger and bigger cubes, accumulated further on the screen, and combined with a low pass filter (6) give the illusion of rays of light coming from the origin of the initial cube, towards the extremities of the screen.

The transparent clones of the base cube are themselves expanded and contracted as the accenting rhythm $M4J4E5$ kicks in.

Part IV

Conclusion

Delivering the intro in Barcelona was difficult, but a great pleasure, and I learned a bit while creating it. Physical simulations remain a very interesting model for creating complex content.

The color palette seemed to have irked quite a few people, although that might not be surprising as I intended it as slightly aggressive. Nevertheless, I'd like to experiment a bit more with representing colors and their relations, a quite fascinating topic all in itself.

Connections between *Walking on Four* and the old 90s intros also didn't go unnoticed, which was nice to read.

I took a few months out of coding, to concentrate a bit on music making. In contrast with my work in trying to represent timelines algorithmically, I also became interested in taking the counterpoint: interactive control of a predefined set of effects.[12]

References

[1] In the interactive visual instrument *Floo* (1999), Golan Levin displays entangled tendrils of lights. The interactions are produced by a pseudo-newtonian simulation combining repulsive and rotative force fields.
`http://acg.media.mit.edu/people/golan/sketches/index.html`

[2] *Raycoaster* (2005), by Quasimondo is a simple java applet made in the Processing multimedia coding environment. It tries to display caustics: patterns of lights as they go through a serie of material. A mondane example: the shimmering lights seen at the bottom of a swimming pool
`http://incubator.quasimondo.com/processing/raycoaster.php`

[3] Processing, a java-based language for audio visual programming.
`http://www.processing.org/`

[4] `http://incubator.quasimondo.com/`

[5] *Clothing Like A T-Shirt Saying I Wish* (2006)

[6] *Melancholia* (1996) by Ryu Murakami
His character, Yazaki, rants:

> Why? Why do I hate talking about myself? Sincerely I hate it. It's like conceptualising one's desires. Don't you find one must be authentically pretentious to employ those words! Conceptualising one's desires! I mean that I hate all those guys who pretend to be *authors*. A sequence of images, a bit of music, and there, here we are. Nothing's cruder than this fashion of using and showing one's very own desires all by oneself. And I'm not talking to you about masturbation. The most incredible thing is, people have the utmost respect for these guys.

[7] `http://en.wikipedia.org/wiki/Phrygian_mode`

[8] *To trace or not to trace - that is the question.* Seminar at BP2005 by Prof. Dr. rer. nat. Alexander Keller / Universität Ulm

[9] `ftp://ftp.scene.org/pub/parties/2005/bcnparty05/`

[10] `ftp://ftp.scene.org/pub/parties/2005/stream05/`

[11] `ftp://ftp.scene.org/pub/parties/2000/assembly00/`

[12] Lazy Sunday Radio: Video premiere, held on 2006.03.19. Performing as neq together with sushi brother.

TPOLM lazy sunday radio was born in the year 7000, as live music sessions broadcasted through the internet by electronic musicians in helsinki, finland. Over the years the lazy sunday sessions have evolved into an internet festival, in which musicians from several countries and continents stream music to a global audience from their homes. This session on 19 march, is the first time visual artists will also stream live video to accompany the music.

`http://www.tpolm.com/archive/lsv200603/index.html`