



Living characters in 4k intros

beyond just abstract worlds

Iñigo Quilez – iq/rgba

Function 08



rgba

Index

- Context
- Making of some intros
- Conclusions



rgba

Index

- Context
- Making of some intros
- Conclusions



Context

- The vast majority of intros today are abstract.
 - They show some sort of choreography of shapes synchronized to music and camera.
- Many 64k intros however tend to be quite the opposite, they show scenes with recognizable objects.
- Demos also show non abstract stuff very often (cities, characters, animals, architectural structures...)
- So the question is, why do 4k intros mostly show abstract stuff?



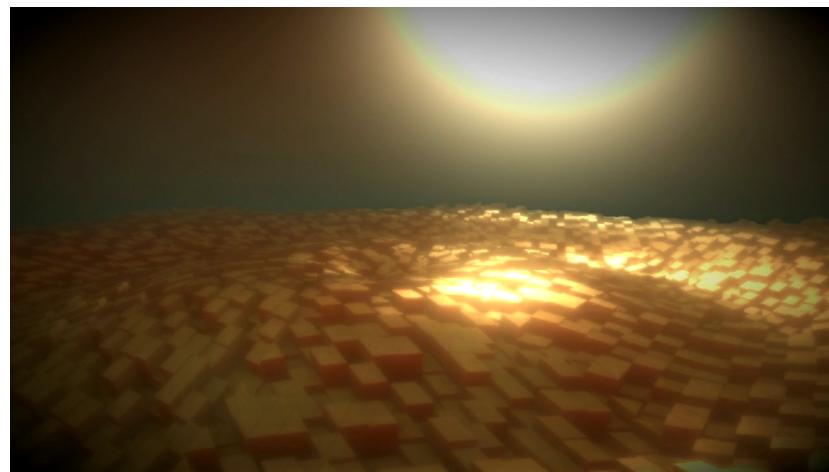
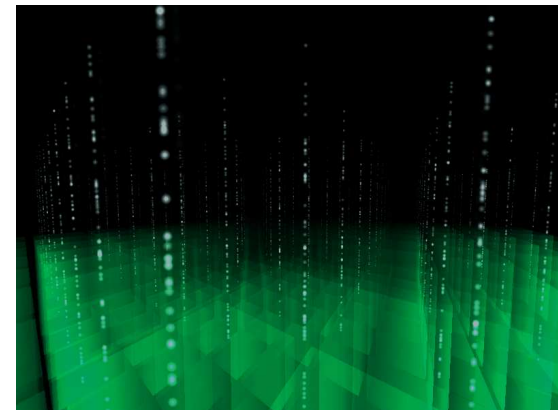
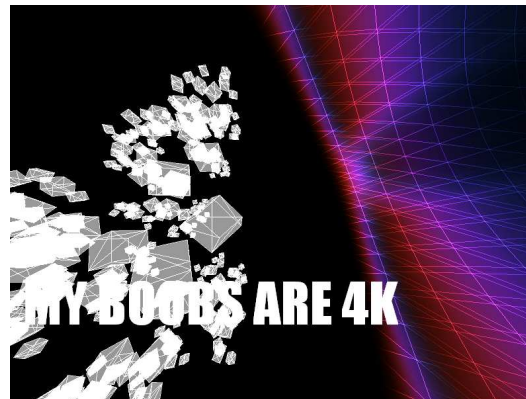
rgba

Context

- 4k intros can be classified in two main categories:
 - Cube based intros
 - The rest of intros

Context

- Intros with cubes

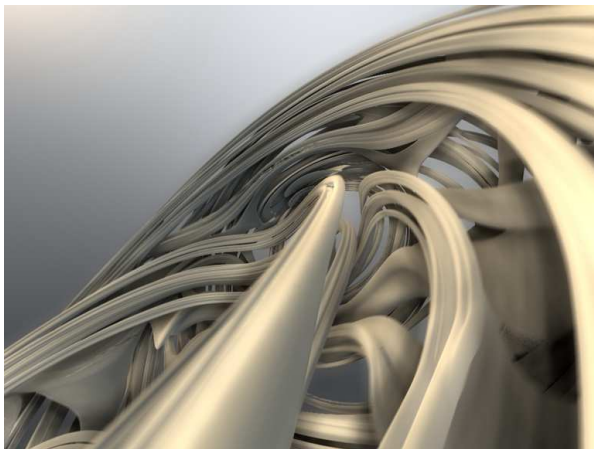




rgba

Context

- Intros without cubes normally show ...
 - fractals, strange shapes, lines, particles...

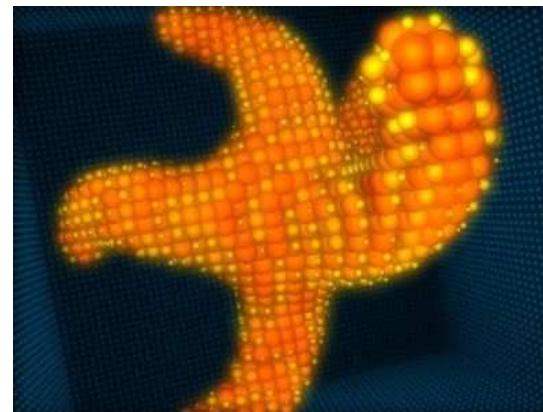
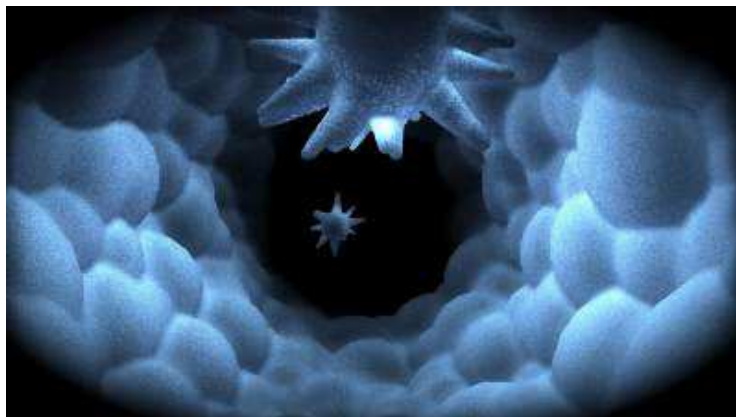
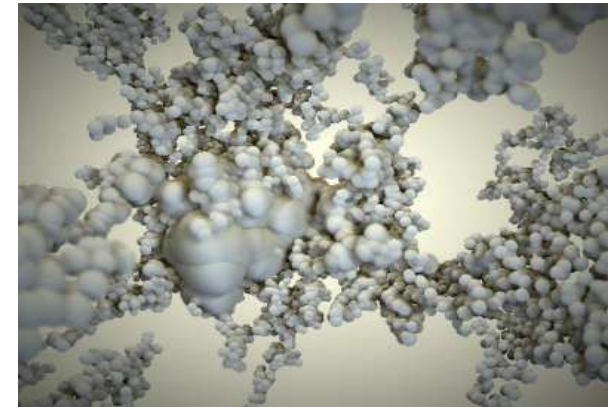
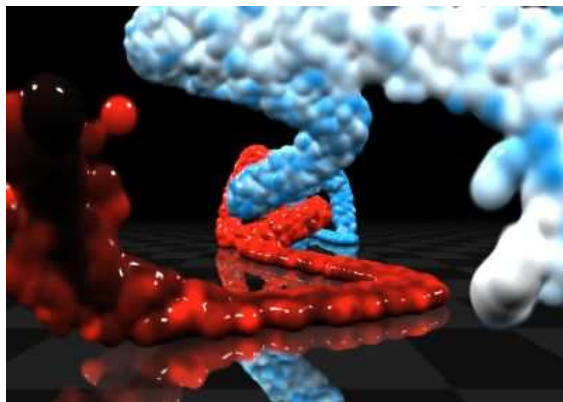




rgba

A message

- Intros without cubes normally show ...
 - ... spheres!





rgba

Context

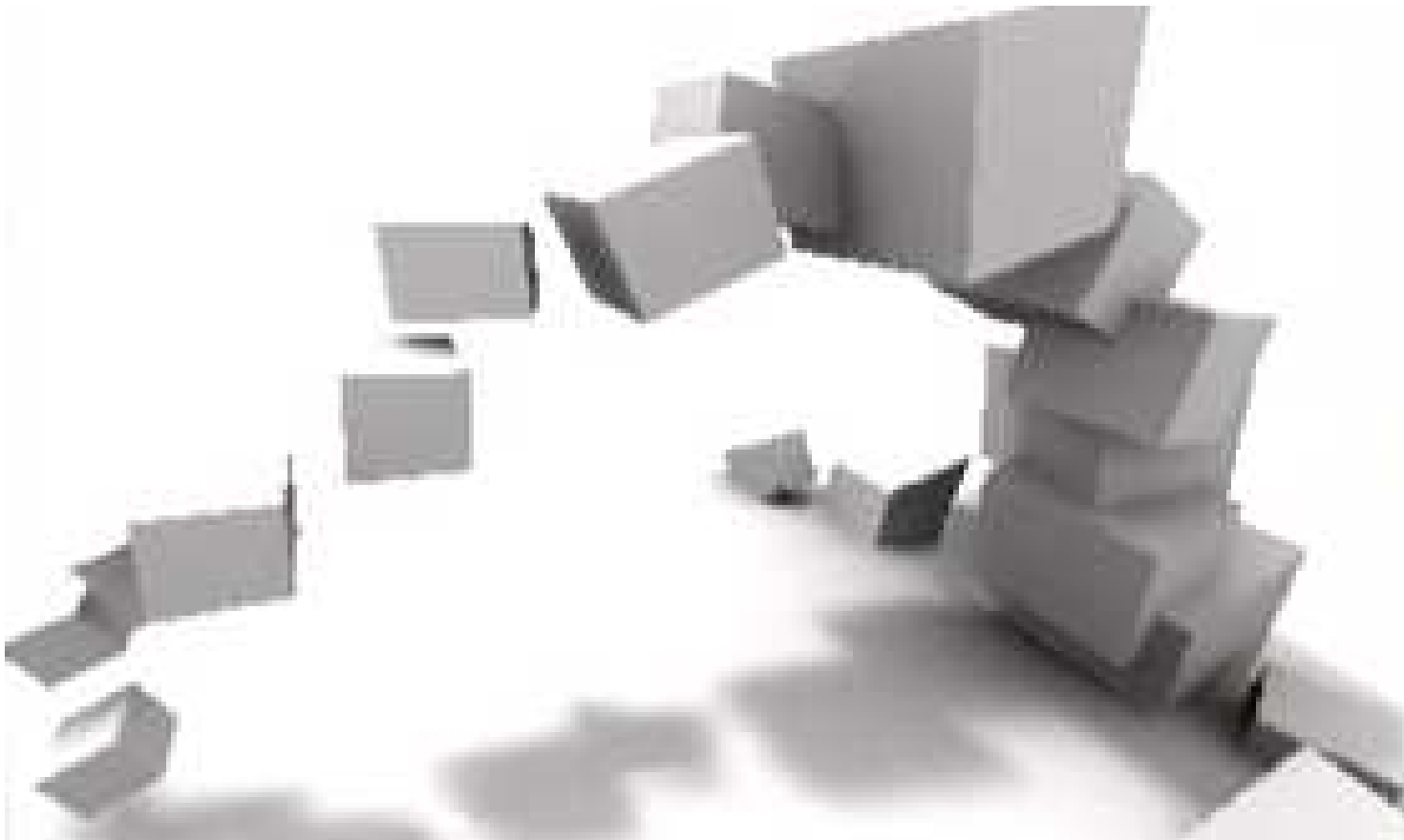
- But spheres are not as cool as cubes
- Neither landscapes or fractals
- For some esoteric reason demosceners love cubes



rgba

Context

- Cubes are the main ingredient for success

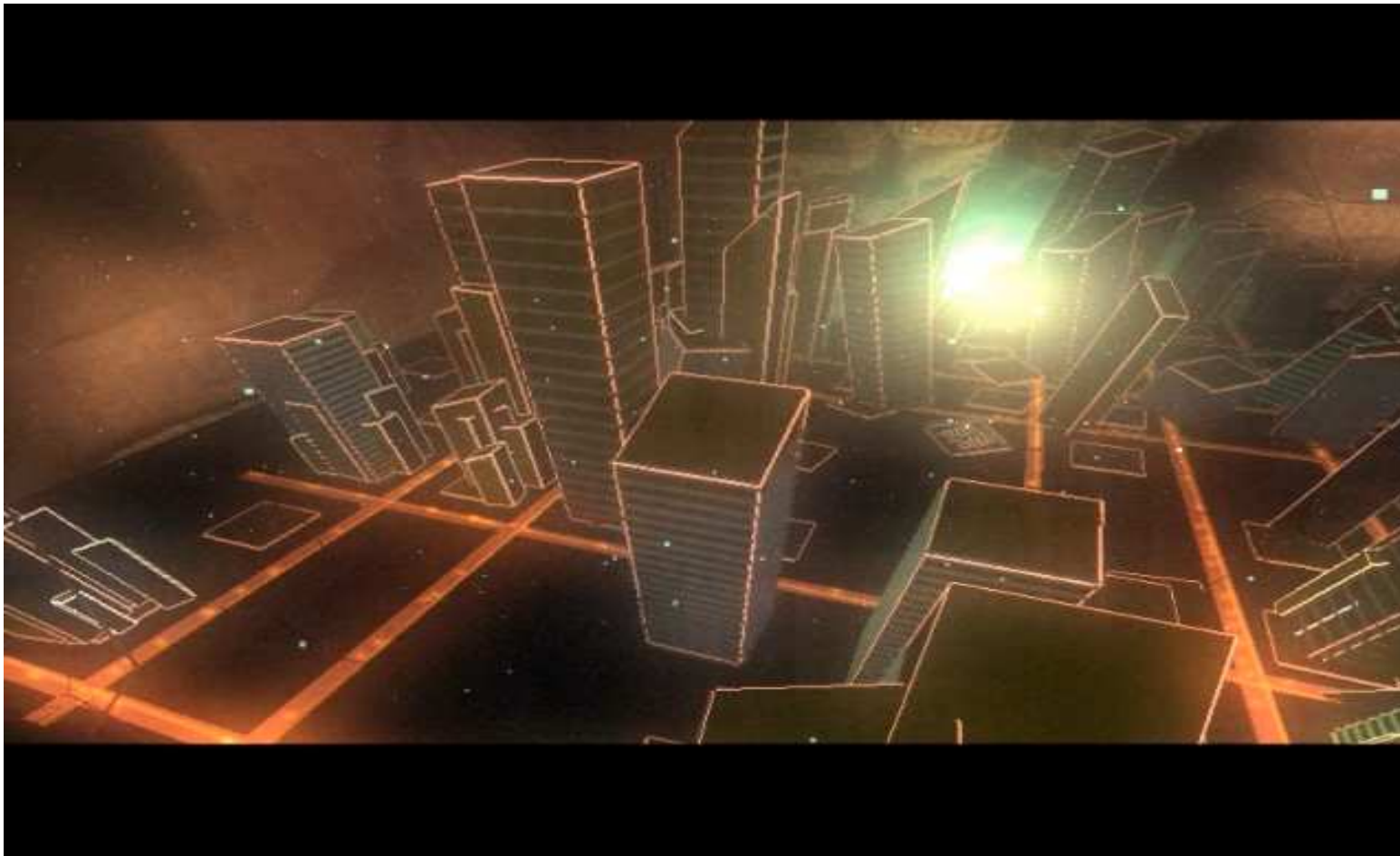




rgba

Context

- Cubes are the main ingredient for success





rgba

Context

- Cubes are the main ingredient for success





rgba

Context

- Cubes are the main ingredient for success





rgba

Context

- Cubes are the main ingredient for success





rgba

Context

- Cubes are the main ingredient for success





rgba

Context

- Cubes are the main ingredient for success





rgba

Context

- Cubes are the main ingredient for success

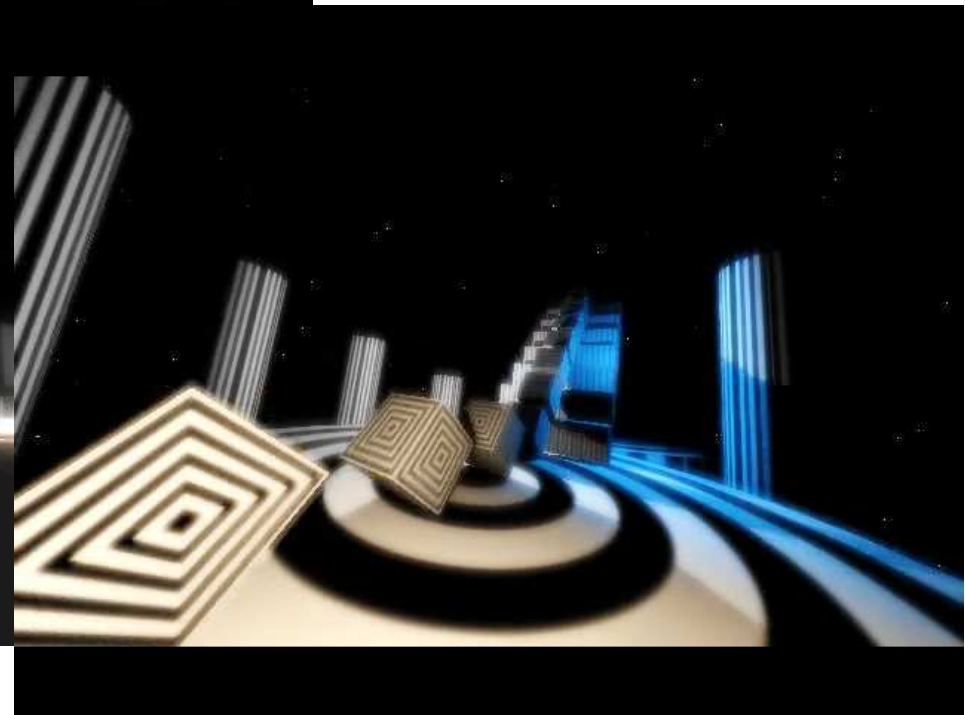
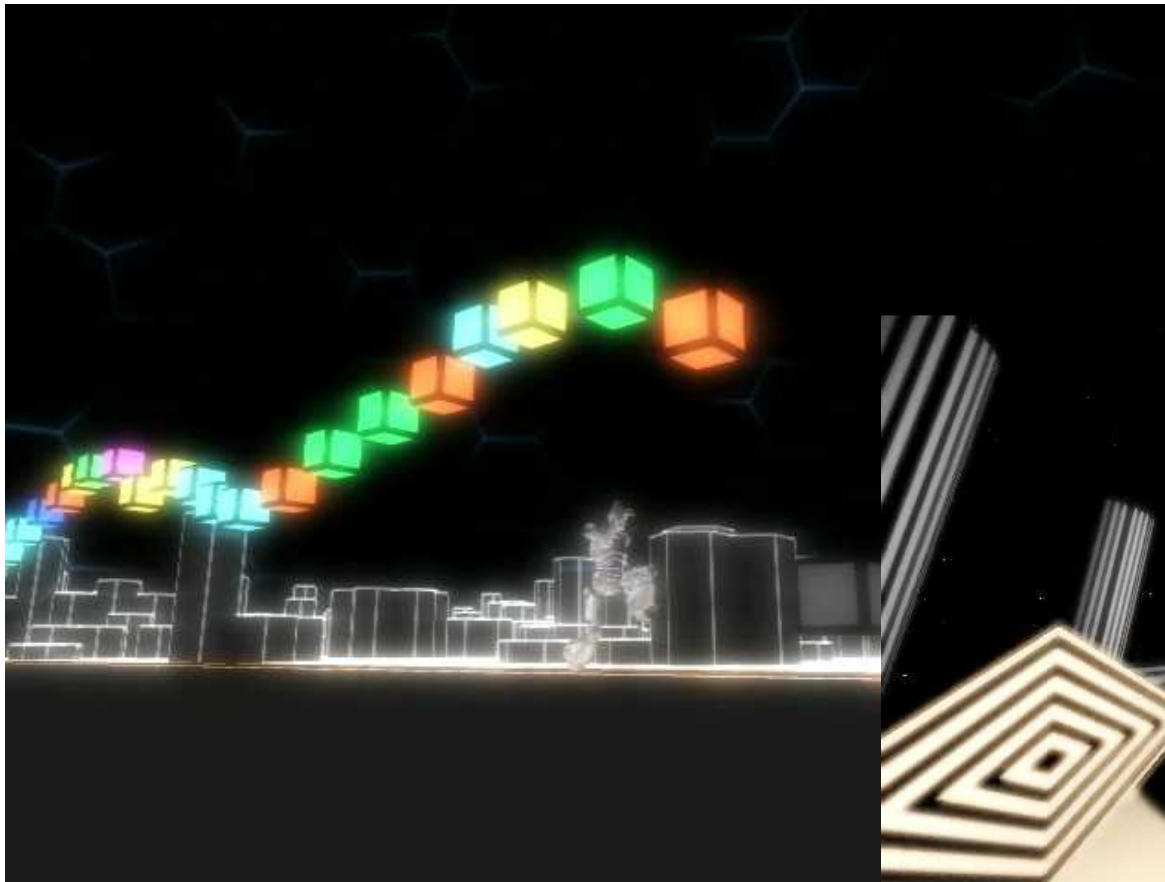




rgba

Context

- Cubes are the main ingredient for success





Context

- In anyway, cubes involved or not, 75% of the intros show abstract stuff
- And 99.9% of the intros don't (cannot) show non-abstract organic shapes.
- I would classify like this:

	<i>abstrasct</i>	<i>non abstrasct</i>	
<i>organic</i>	20%	0.1% ?	
<i>not organic</i>	60%	20%	

The red area represents the amount of cube-intros

- I have tried the four already



rgba

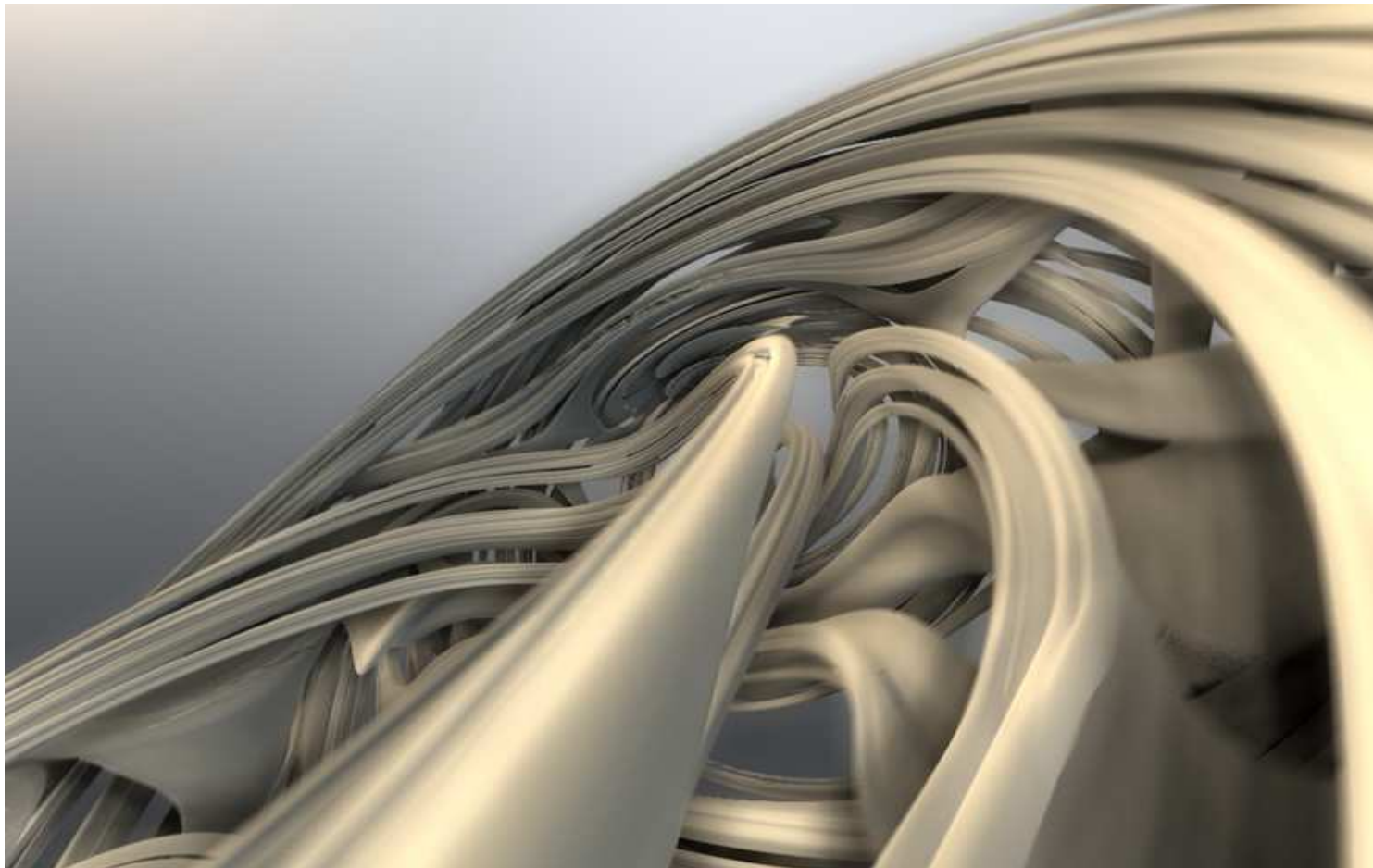
Index

- Context
- Making of some intros
- Conclusions



Making of "*Kindernoiser*"

- Classification: organic & abstract shit





Making of "*Kindernoiser*"

- Classification: organic & abstract shit
- Easy to make:
 - 1. Get some interesting effect(s)
 - 2. Play randomly with all it's parameters to get use to them
 - 3. Think about something interesting to do with it (find an idea?)
 - 4. Synchronize it to the mzk
 - 5. People will think there is a concept behind
 - 6. Profit



Making of "*Kindernoiser*"

- Classification: organic & abstract shit
- In this case the effect(s) where
 - Rendering of a Julia Set with distance fields raymarching
 - Screen Space Ambient Occlusion
- The idea was to make an organic shape fighting against it's embedding space

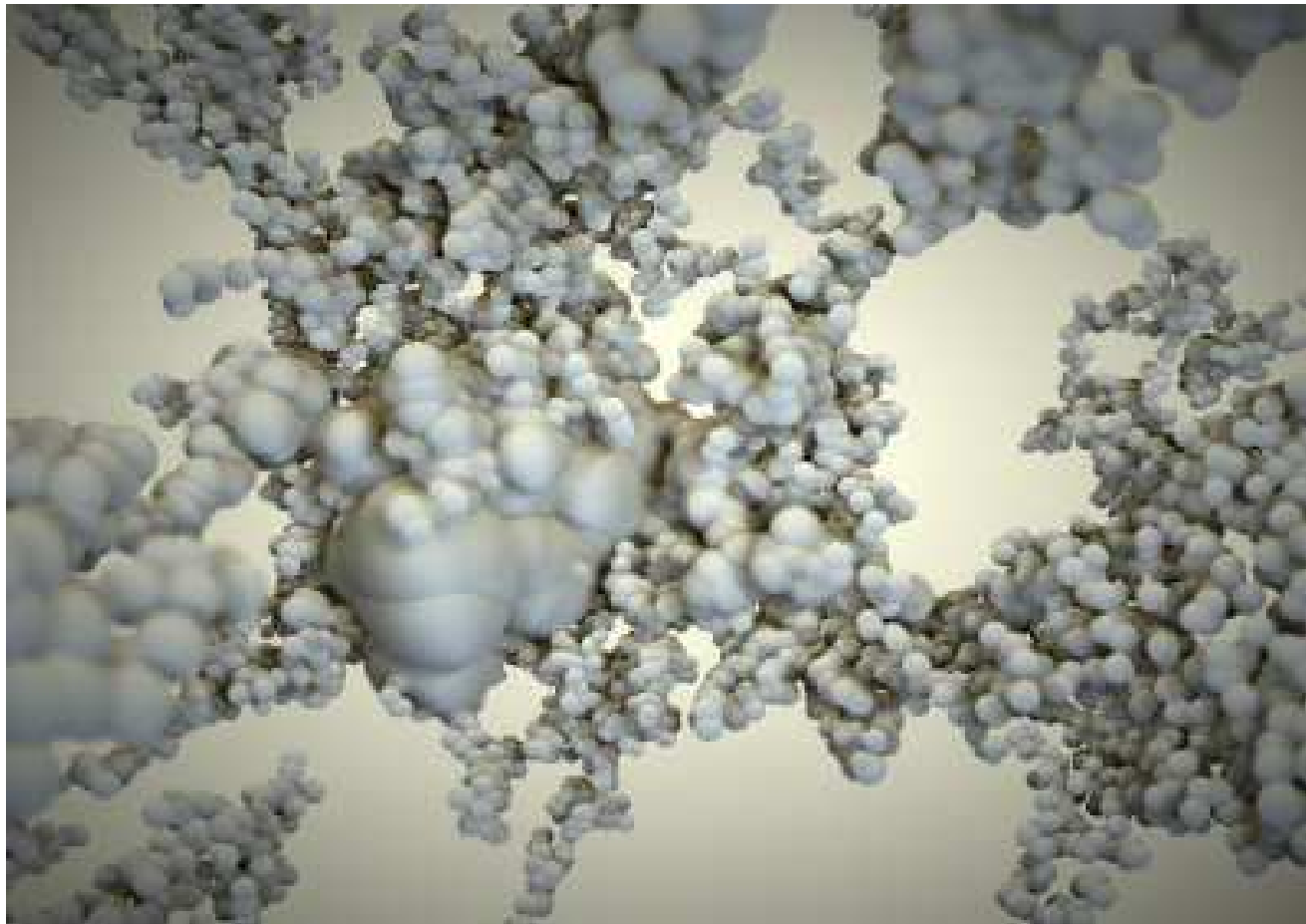
- All the production process was very easy



rgba

Making of "*Kindercrasher*"

- Classification: non organic & abstract shit





Making of "*Kindercrasher*"

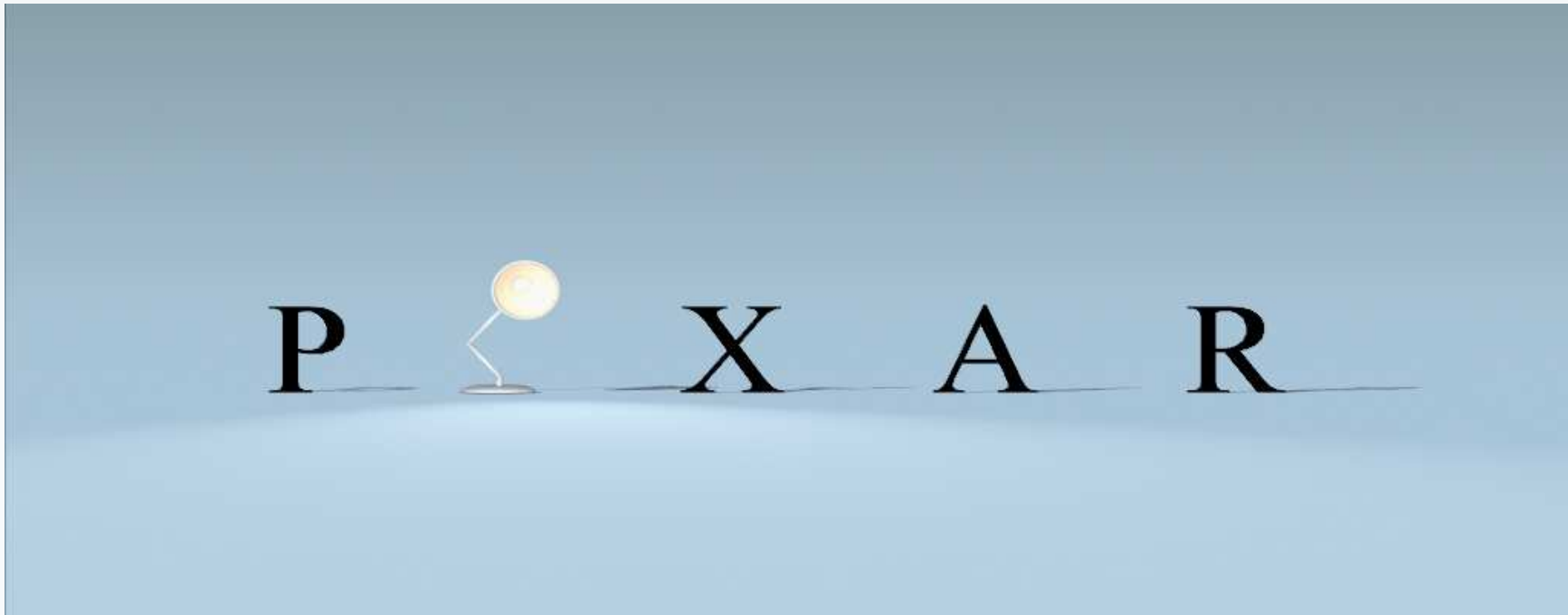
- Classification: non organic & abstract shit
- In this case the effect(s) where
 - Fast rendering of millions of spheres (not used in the end!)
 - Screen Space Ambient Occlusion
 - FFT synchronization
- The idea, to make something with lots of energy
- Rushed for being in time for Inspire 2008 (Spain), not satisfied with the result, but was just good enough.
- Mega easy to get something "just noto very bad" : synch, synch, ...



rgba

Making of "*Luxo 4k*"

- Classification: non organic & non abstract shit





Making of "*Luxo 4k*"

- Started here at Function 2007
- Collaboration with Gargaj again
- Features
 - Simple geometry (revolution profiles)
 - Planar shadows (render with a flatenning matrix - raycast)
 - Animation!

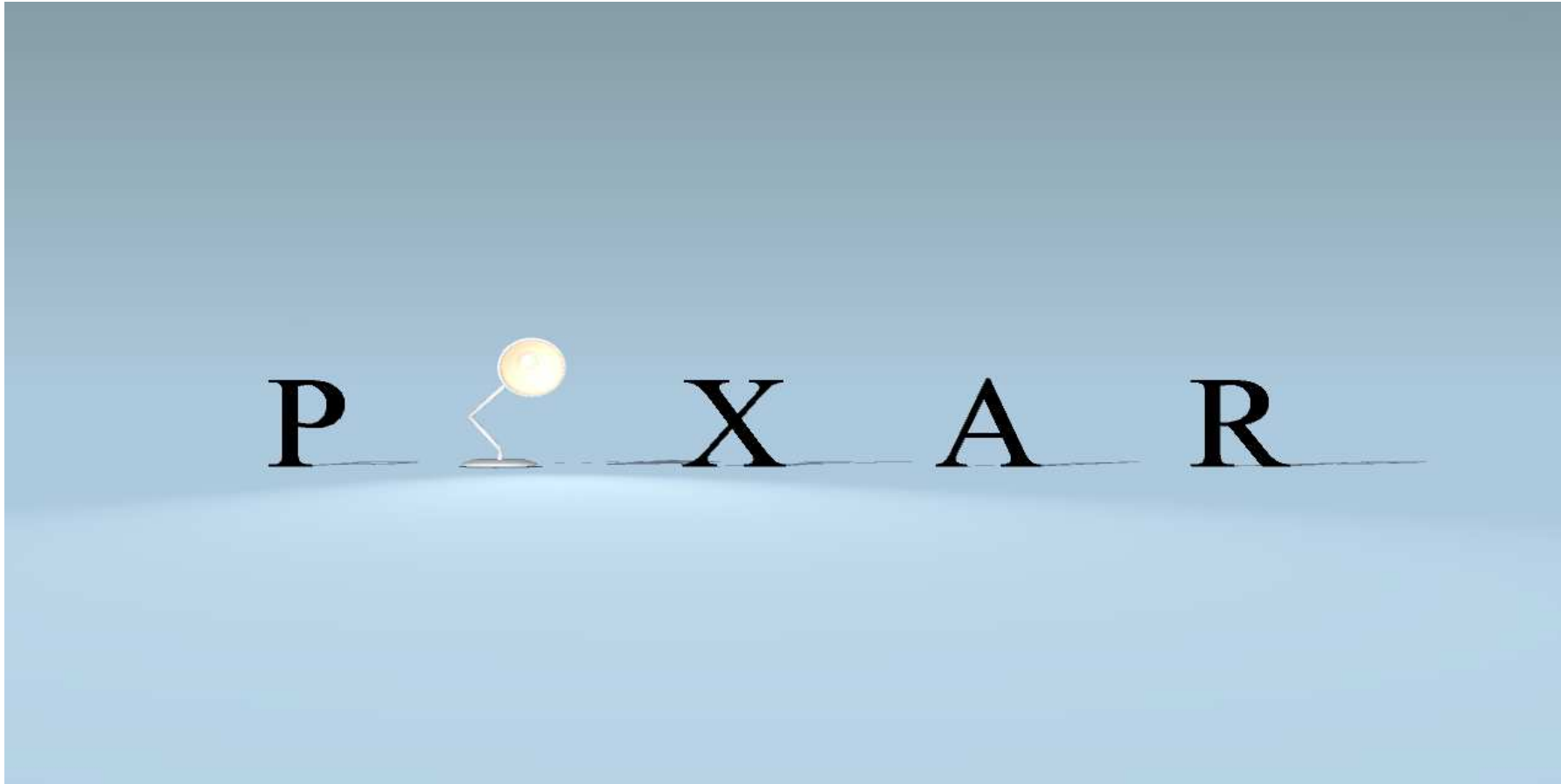


P I X A R



rgba

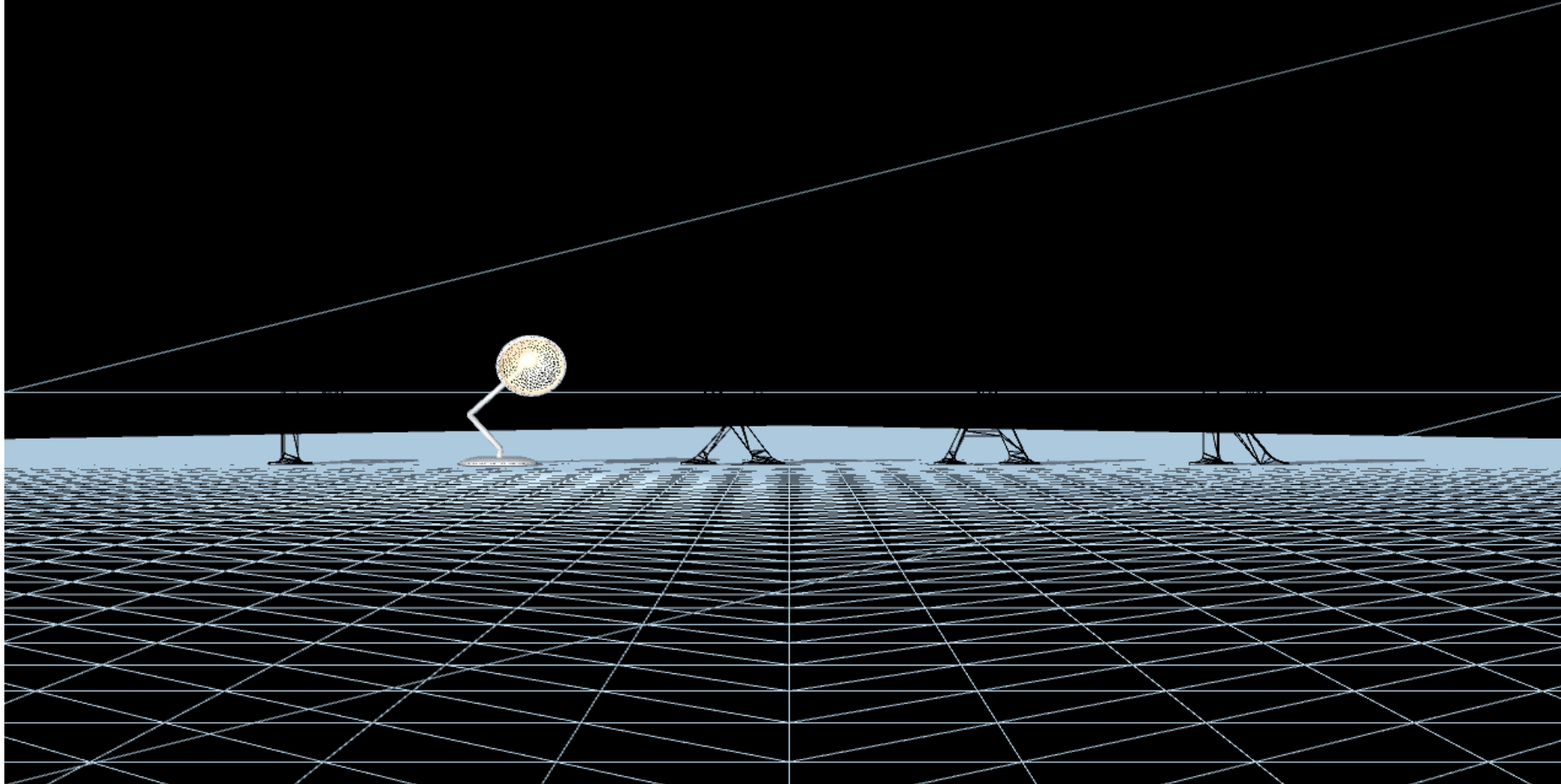
Making of "*Luxo 4k*"





rgba

Making of "*Luxo 4k*"





Making of "*Luxo 4k*"

Several options for the animation

- Define animation with math functions (ala Stiletto or Mircropolis)
 - Ok when there is no animation script in front
 - You adapt the intro to what you can get procedurally
- Store the animation curves (compressed)
 - Need for compression of course. Can this be done in 4k?
 - The "only" solution when the animation is predefined.
 - Much more easy than replicating movements by formulamination...



Making of "*Luxo 4k*"

How to create the animation curves

- 1. "somehow" get a Pixar movie at the higher resolution you can
- 2. rip the Luxo sequence, (and of course delete the movie)
- 3. extract all the frames of the animation (245)
- 4. hack a tool that loads the frames and lets you draw pixels on top
- 5. rotoscope every joint for every frame
- 6. given the points, extract angles relative to the bone parent
- 7. compress all the angular info curves
- 8. export to an array of data



rgba

Making of "Luxo 4k"

Rotoscoping and compression "tool"

- I hate developing tools, just implemented the bare minimum (in C#)

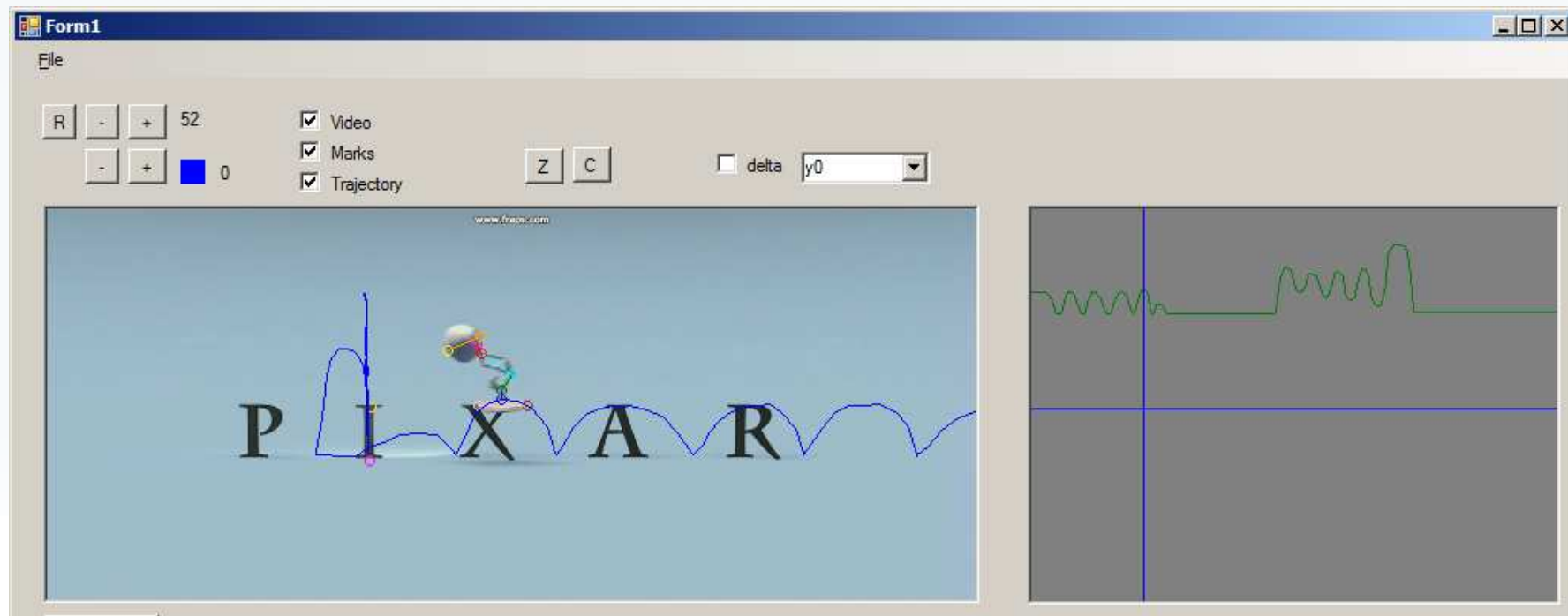




Making of "Luxo 4k"

Rotoscoping and compression "tool"

- Can add, remove and clone join points
- Visual inspection of absolute, relative, and relative+delta rotation curves





Making of "*Luxo 4k*"

Rotoscoping and compression "tool"

- It worked fine because the basic animation was mostly 2D, so I could "see" very well all the angles
- However Luxo rotates in 3D before jumping to the "I"
 - Had to manually create that animation curve
 - Directly in the delta coded C array of data (a nightmare, I)
- The compression of the curves introduced lot of noise
 - Went for manual low pass filtering of the signals
 - Directly in the delta coded C array of data (a nightmare, II)



Making of "*Luxo 4k*"

Compressing

- 11 signals to store, 246 samples each (more than 10 seconds at 24 fps)
- Curves were delta encoded
 - Prediction of smooth curves at high framerates (24 fps) works very well of course, competitive with splines, but easier to work with 😊
- Less precision required as going down the hierarchy of bones \o/
 - First nodes compressed to 3.5 bits per frame
 - Last nodes compressed to 1.7 bits per frame
- In the end, arround 100 bytes per curve for the complete animation
 - Arround 979 bytes of total animation data
- At replay time, linear interpolation for subframe accuracy.



Making of "*Luxo 4k*"

- In the end, with animation curves (979), animation player (220), mesh data (96), mesh generation (147), hierarchical object animation, lighting, plus shadows, materials, font and graphics and sound setup, it went to 3500.
- So, basically Gargaj had 600 bytes to generate some music!
 - Without gm.dls, nor dx tricks, nor other cheats.
- The day Gargaj showed it at Pixar the intro was 100 bytes over the limit.
- Worked almost two weeks more to get it down to 4096.



Making of "*Luxo 4k*"

- Conclusion were:
 - Curve based animations in 4k are not impossible, but would need better compression for real cases (intros longer than 20 seconds).
 - Making an intro when the target is precisely set in advance is MUCH more difficult than just making an intro where the storyboard can be adapted to the capabilities of the techniques, tricks and code.
 - Didn't impress much to the average demoscener, not even to intro coders 😞



rgba

Making of "*Stiletto*"

- Classification: organic & non abstract shit
- An attempt to code pr0n (push limits, a human in 4k?)

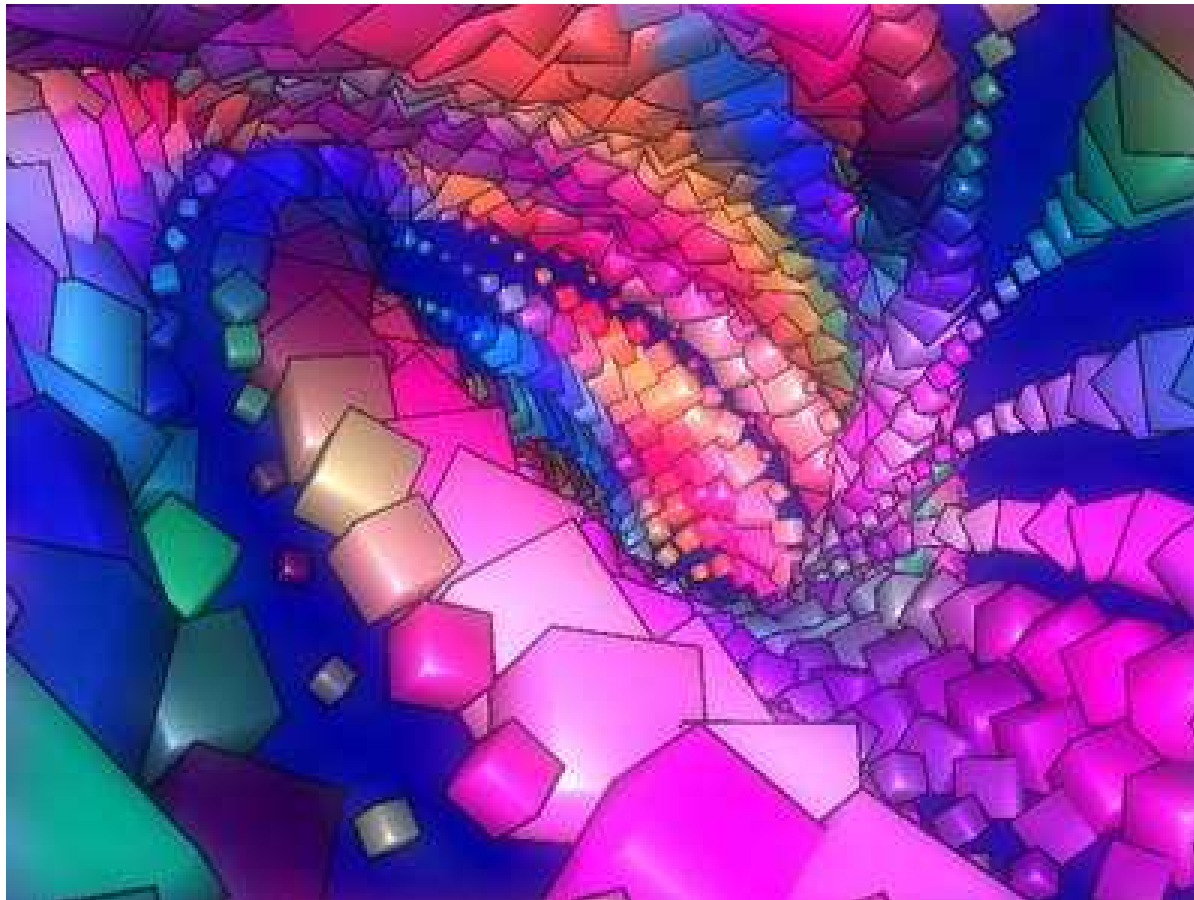




rgba

Making of "*Stiletto*"

But no codepr0n will beat god music and cubes!!!!!!



Candystall, by , st at Assembly 2007





Making of "*Stiletto*"

- An attempt to code pr0n (push limits, a human in 4k?)
- Features (almost like Luxo4k...)
 - Complex geometry
 - Compressed
 - Subdivided
 - Planar shadows (render again with a flatenning matrix - raycast)
 - Animation!
- Stiletto aspired to impress
 - Geometry-wise
 - Animation-wise



Making of "*Stiletto*"

Geometry

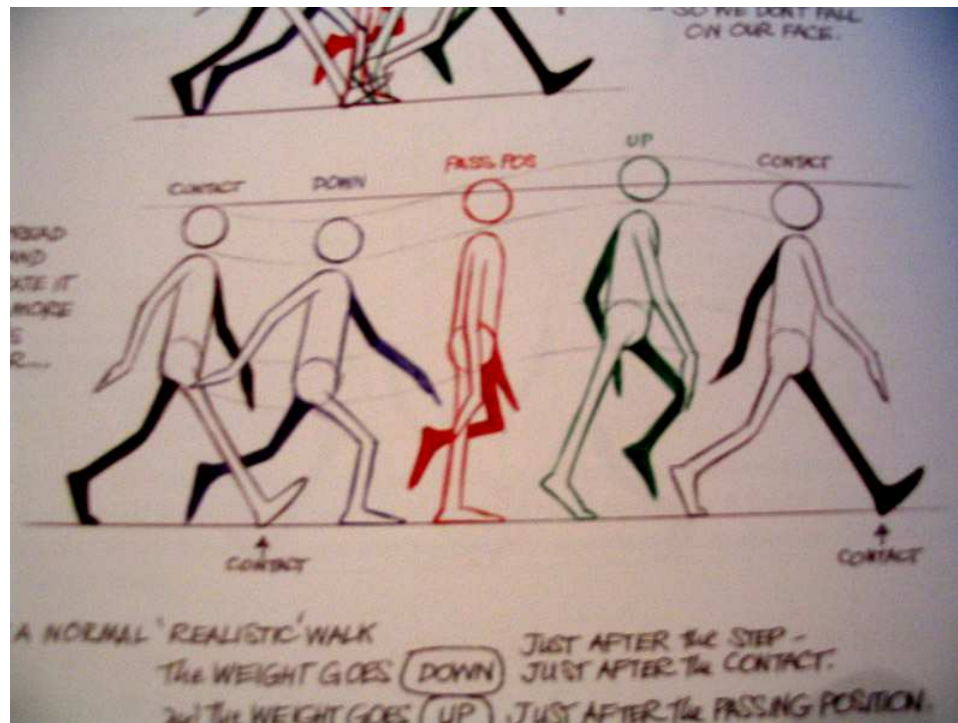
- The idea was to be the first to put some organic and non trivial geometry (not terrains or architectural structures, nor abstract shit of course)
- Used: modeling in Maya, compression and subdivision.
- More details on subdivision and compression just here:
<http://www.rgba.org/iq/divulgation/bcn2007/bcn2007.htm>
- I used a simplified version of the automatic vertex weighting algorithm (for skinning) I described here:
<http://www.rgba.org/iq/divulgation/breakpoint2007/breakpoint2007.htm>



Making of "Stiletto"

Animation

- Once I knew I could put an interesting mesh, I had to animate it.
- Googled for "animation cycle", found this nice image...



...and thought I could make a walk cycle with few cosinus applied to the rotation angles of the bones (again, it's almost a 2d animation)



Making of "*Stiletto*"

Animation

- Very little effort compared to curve compression
 - No tools to develop
 - No compression to investigate
- More difficult to get to work...
 - Really?
- Much smaller... 300 bytes?

```
static void animateCharacter( const float t )
{
    float f, a1;
    // body translation
    mats[13] = -p0d05*cosf( 2.0f*PFreq*t - 1.5f );
    mats[14] =  p0d10*cosf( 2.0f*PFreq*t );

    // heaps and torso rotation
    a1 = cosf( PFreq*t );
    quatsw[0] = -p0d10*a1;
    quatsw[1] =  p0d25*a1;

    // one leg
    a1 = cosf( PFreq*t );
    quatsw[2] = p0d05*a1 - 0.5f;
    quatsw[3] = p0d30*a1;
    f = 0.5f + 0.5f*cosf( PFreq*t - 2.15625f );
    quatsw[4] = -p0d05+p1d20*f*f;

    // the other leg (same as before, but pi radians later)
    a1 = cosf( PFreq*t + pi );
    quatsw[6] = p0d05*a1 - 0.5f;
    quatsw[7] = p0d30*a1;
    f = 0.5f + 0.5f*cosf( PFreq*t - 2.15625f + pi );
    quatsw[8] = -p0d05+p1d20*f*f;
}
```



Making of "*Stiletto*"

Conclusions

- Other than the mesh, features were almost identical to Luxo4k (shadows, flat background, animation. Actually Stiletto had far more complex geometry.
- However, it was much simpler to do.
 - I was free to make any animation I could
 - And adapt all my cameras, intro and design for it
 - While Luxo it was the other way around, first have an idea, and then develop whatever possible to get it to work.
- Regarding technique, go for formanimation over curve compression when possible.



Conclusions

- Non abstract shit requires MUCH more work
- But most people will not appreciate
 - Just as intros are not that much appreciated by demo coders, non abstract intro coders are not that much appreciated even by intro coders. Basically only very few people can value it (6 or so)
- Anyway demosceners prefer abstract stuff to regular animations.
- Anyway demosceners prefer CUBES to any other type of abstract stuff.

- However, coders should forget about instant success and feel challenged to try non abstract shit.



rgba

?