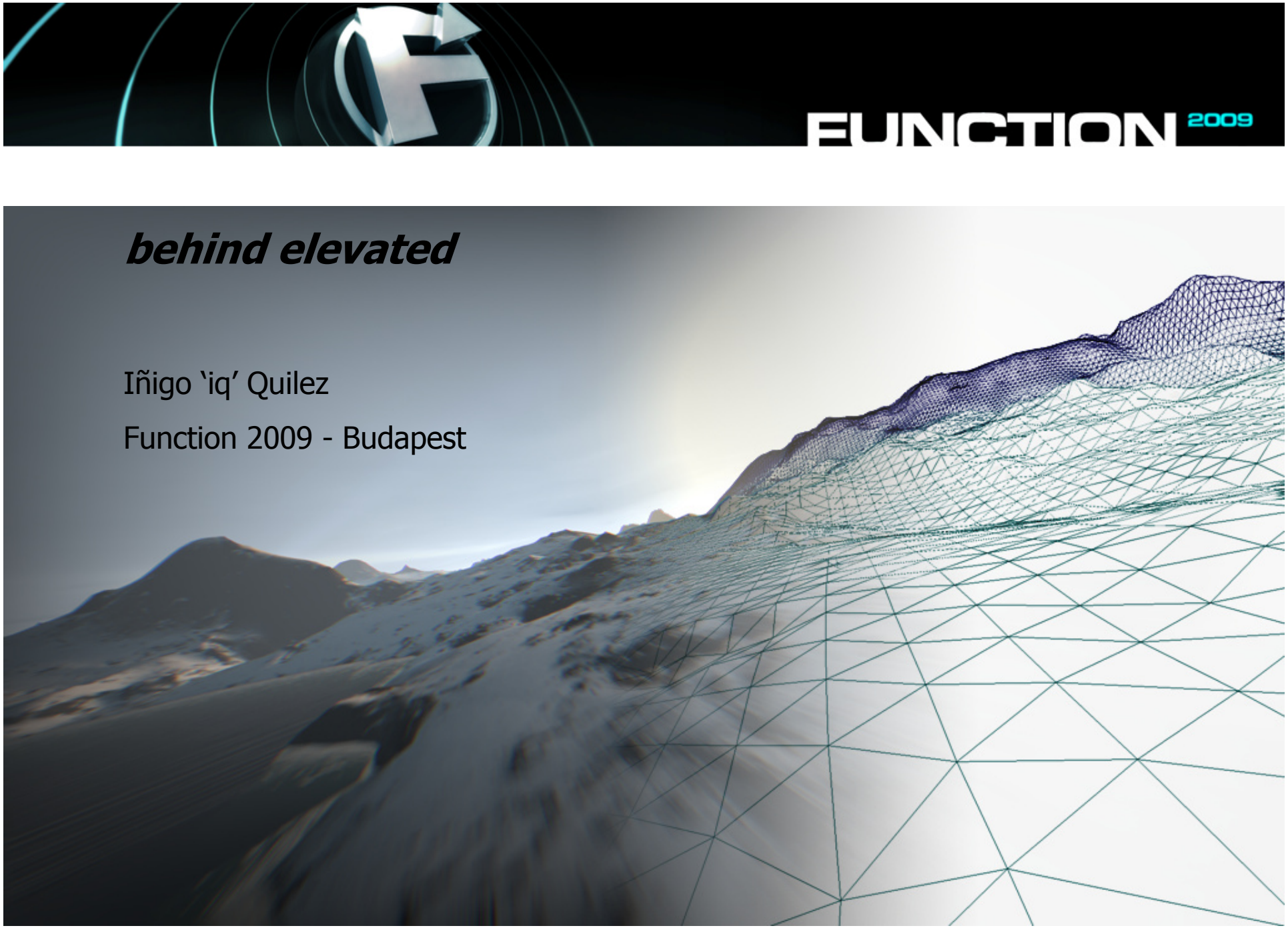
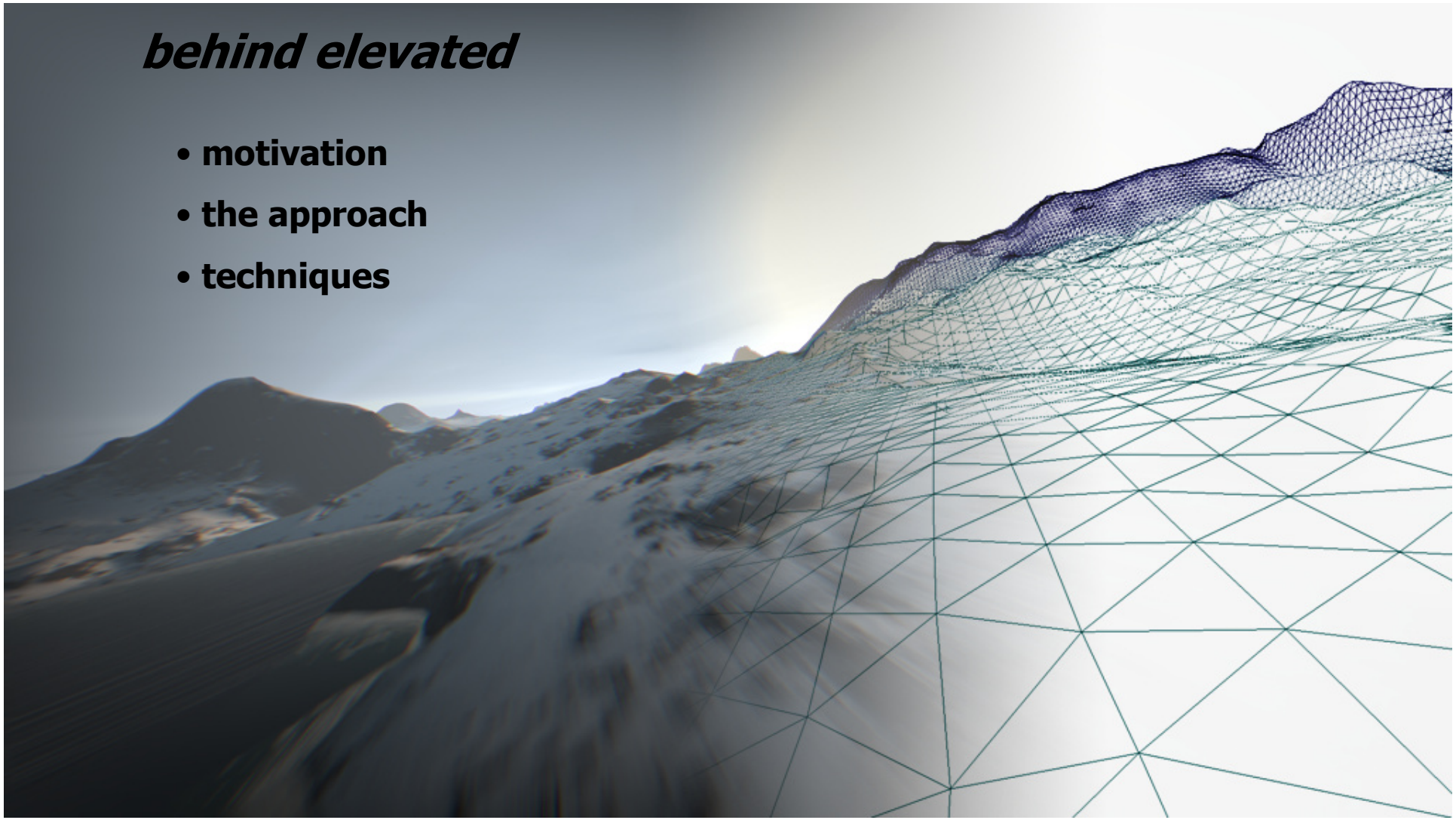
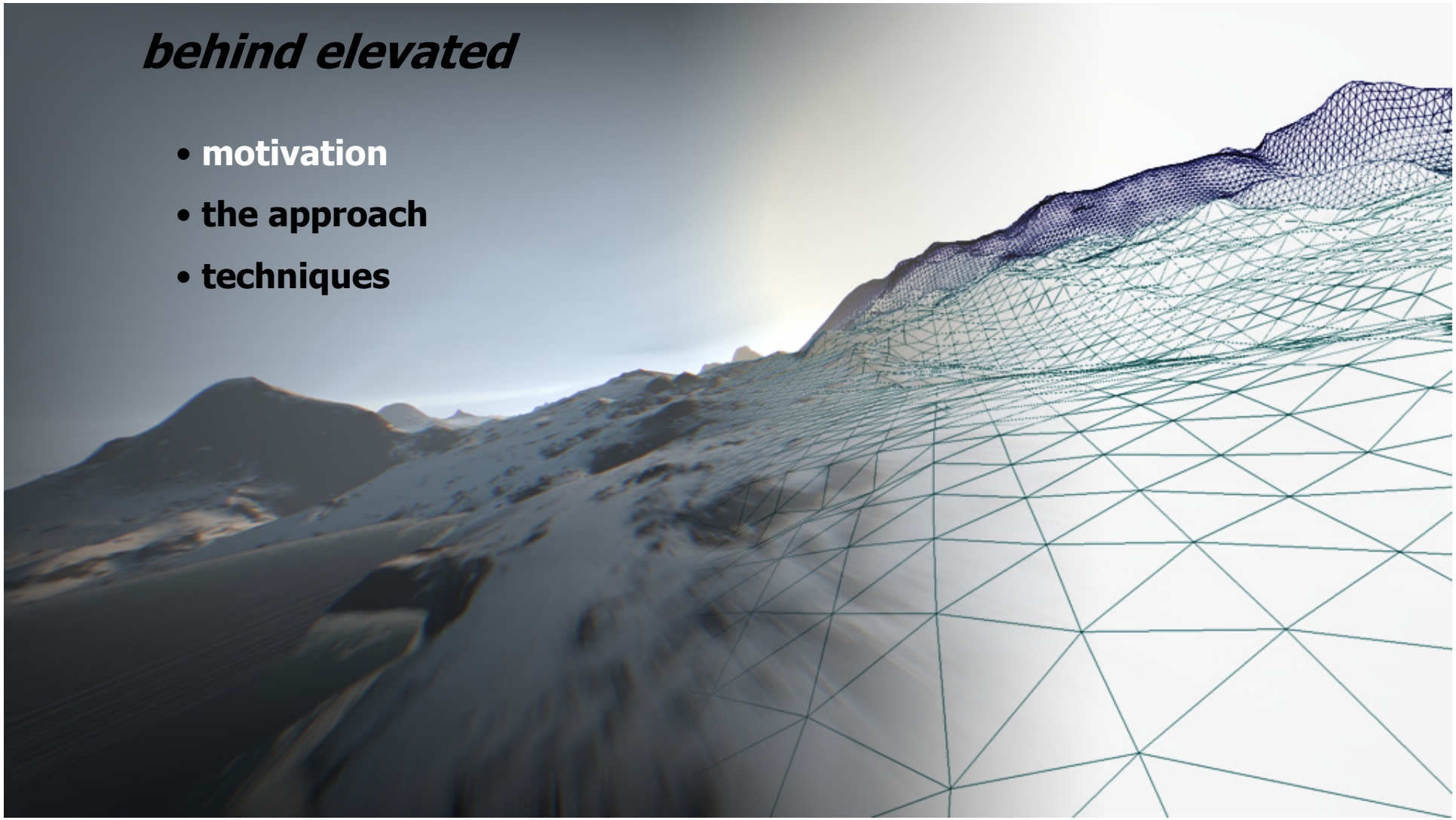

FUNCTION 2009
behind elevated
Iñigo 'iq' Quilez
Function 2009 - Budapest



behind elevated

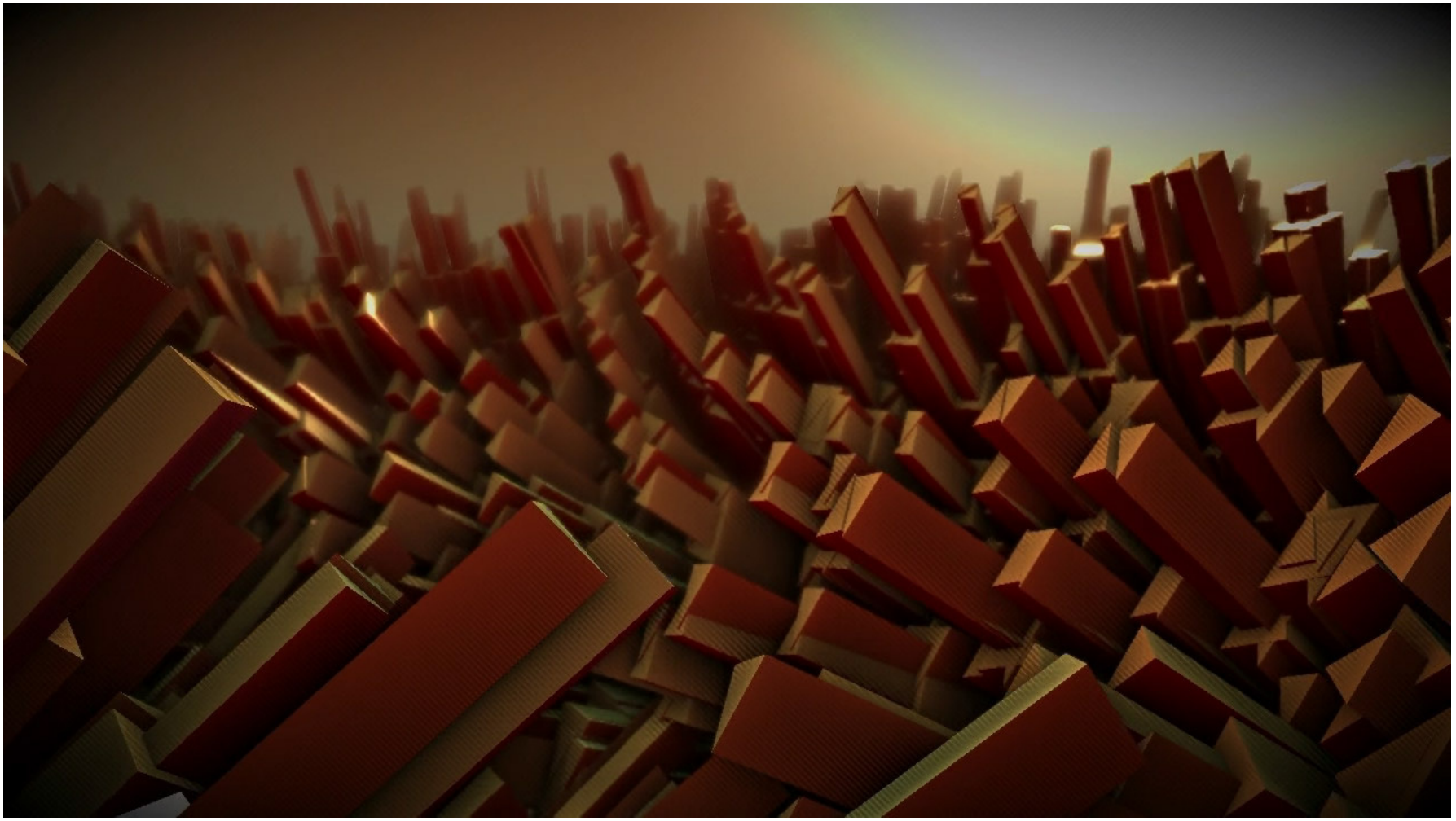
- **motivation**
- **the approach**
- **techniques**



behind elevated

- **motivation**
- **the approach**
- **techniques**

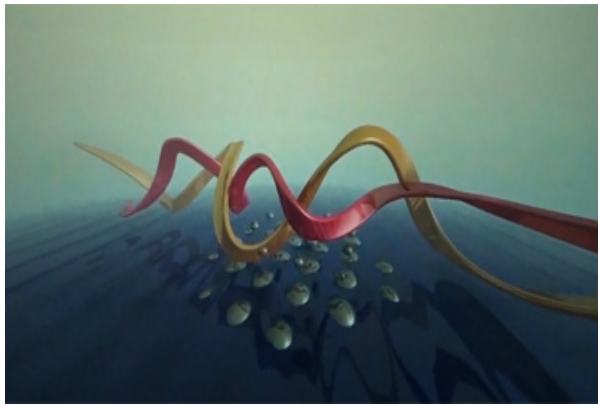
motivation :: competition (1)



Texas / keyboarders – nvscene08 and scene.org awards 4kb intro winner

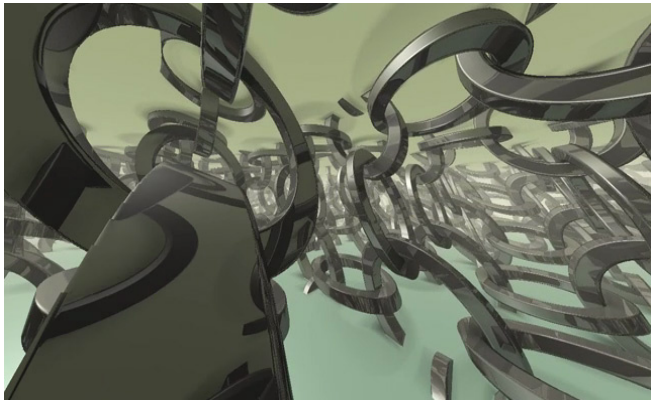
motivation :: technical improvement (2)

- Everybody could smell it was going to be the year of the raymarching
- In fact it happened to be the year of the "AO*Reflection Raymarching"



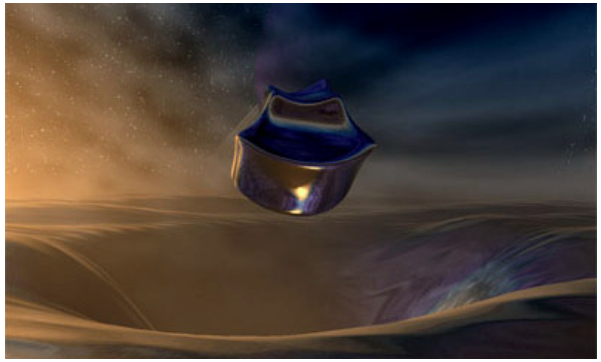
Paradistance, Titan, 2009

Ascension, by Still, 2009



Sult, by Loonies, 2009

Muon Baryon, by YUP+UD+Outracks, 2009



Lunaquatic, by BluFlame, 2009

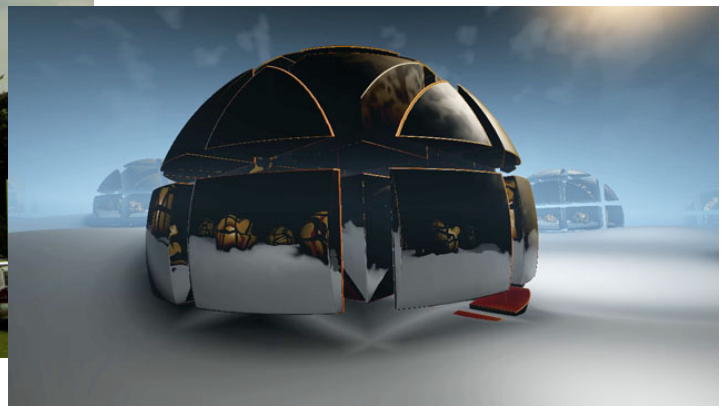
motivation :: technical improvement (2)

- Last year I realized demosceners like cubes
- This year I have discovered sceners like reflective primitives!

motivation :: technical improvement (2)

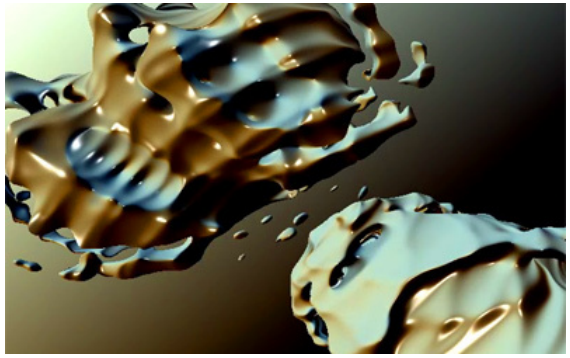
- They should come to Brussels!

The Atomiun, in Brussels, 1958 (50 years old)



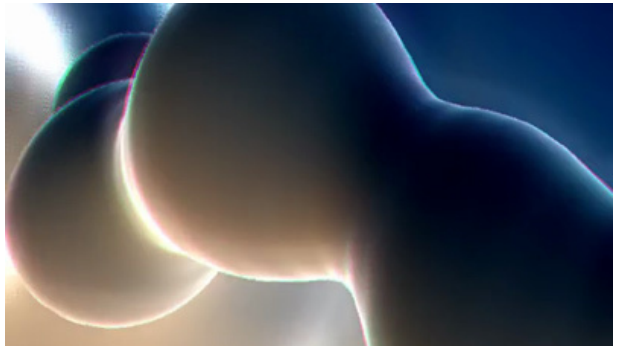
motivation :: technical improvement (2)

- ...it´s all trends in the end... just like... colors

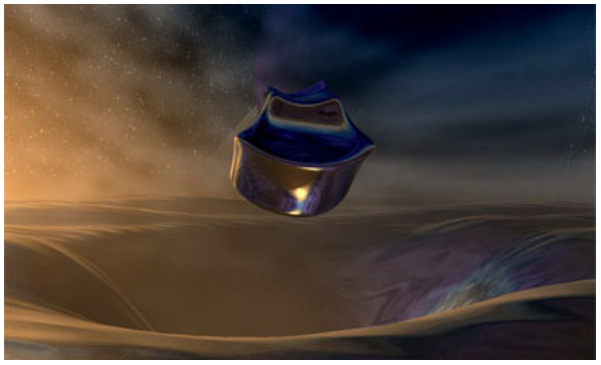


Tracie, by TBC, 2007

Muon Baryon, by YUP+UD+Outracks, 2009



Receptor, by TBC, 2008



Lunaquatic, by BluFlame, 2009

Untraceable, by TBC, 2009

motivation :: technical improvement (2)

- Last year I realized demosceners like cubes
- This year I have discovered sceners like reflective primitives!

motivation :: technical improvement (2)

- Last year I realized demosceners like cubes
- This year I have discovered sceners like reflective primitives!
- Some like both things at a time!

Stargazer, orb + andromeda, 2008 ***(greets, really)***

motivation :: technical improvement (2)

- So need something different than pure raymarch
  - With "trendy" shader in 2 triangles raymarching of procedural fields
    - not easy to produce interesting geometry (beyond twisted cubes)
    - shading is very easy tho (AO, reflections, ...)
    - but no space left for textures, apparently
  - With old triangle meshes you can produce interesting geometry
    - just as we always did
    - more simple shading, no space for "effects" like reflections, ssao, ...
    - not that easy to add textures, it seems

motivation :: technical improvement (2)

- Only 2/24 intros had textures... (a design decision? in all of them? really?)

atrium	tbc :: loonies	1st at Breakpoint	march 2008
texas	keyboarders	1st at NVScene	august 2008
yellow rose of texas	fit :: bandwagon	1st at Assembly	august 2003
mojo dreams		1st at Breakpoint	april 2003
parsec		1st at Breakpoint	march 2005
micropolis	tbc :: mainloop	3rd at Assembly	august 2004
synchroplastikum	calodox	1st at the Ultimate Meeting	december 2005
nucleophile	portal process :: tbc	1st at Assembly	august 2008
fr-057.cns: arancia	conspiracy :: farbrausch	1st at Horde	july 2007
receptor	tbc	2nd at NVScene	august 2008
raptor	mercury	1st at the Ultimate Meeting	december 2006
candystall	pittsburgh stallers :: loonies	1st at Assembly	august 2007
industrial light and magic		1st at Mekka & Symposium	april 2002
muon baryon	yup :: Üd :: outracks	1st at Assembly	august 2009
kindernoiser	rgba	1st at bcnparty	october 2007
sult	loonies	3rd at Breakpoint	april 2009
sprite-o-mat	alcatraz	1st at Breakpoint	april 2007
minidisk	tbc	1st at The Meltdown	october 2007
kindercrasher	rgba	1st at Inspire	may 2008
h4vok	archee	2nd at Breakpoint	march 2008
san angeles observation	armada (pc) :: trauma	1st at Assembly	august 2004
h4vok	archee	2nd at Breakpoint	march 2008
photon race 2	archee	3rd at NVScene	august 2008
jävla kuk fitta	aardbei	2nd at Scene Evε	july 2000

Parsec, 2005

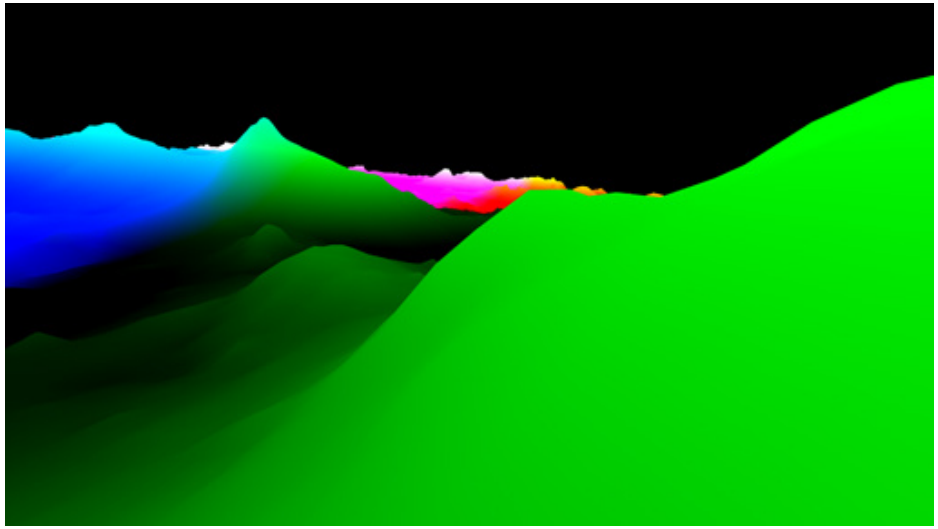
motivation :: technical improvement (2)

- But, do we really need textures?
- If definetly helps to build worlds …
- … beyond brutally obvious CG worlds
    - which is not bad per-se of course
        - but it´s just tiring

motivation :: technical improvement (2)

- Solution: make intros still with 2 triangles…, plus 1.000.000
- Merge both types if intros
  - define **geometry as in the old days**, in a simple explicit way
    - cubes (like 90% of old-trend –pre 2008- intros)
    - terrains (with us since the 80s, never old fashion)
    - DX/GLU primitives (check in4k for a survey by Auld)
    - compressed meshes (*ala* Stiletto and Kinderplomber)
  - do **shading and textures like in 2 triangle raymarched intros**
    - basically like a traditional raymarching, but with primary ray intersections computed by the rasterizer and the zbuffer
    - or if you want to put it in another way, it's like doing defferred shading and lighting as modern games, plus *deferred texturing*

motivation :: technical improvement (2)



- Solve (perhaps only coarsely) the marching with regular rasterization: write z to a buffer, or full xyz (intersection) point as rgb.
  - A 16 bit float format is enough if the data is stored relative to the camera position.

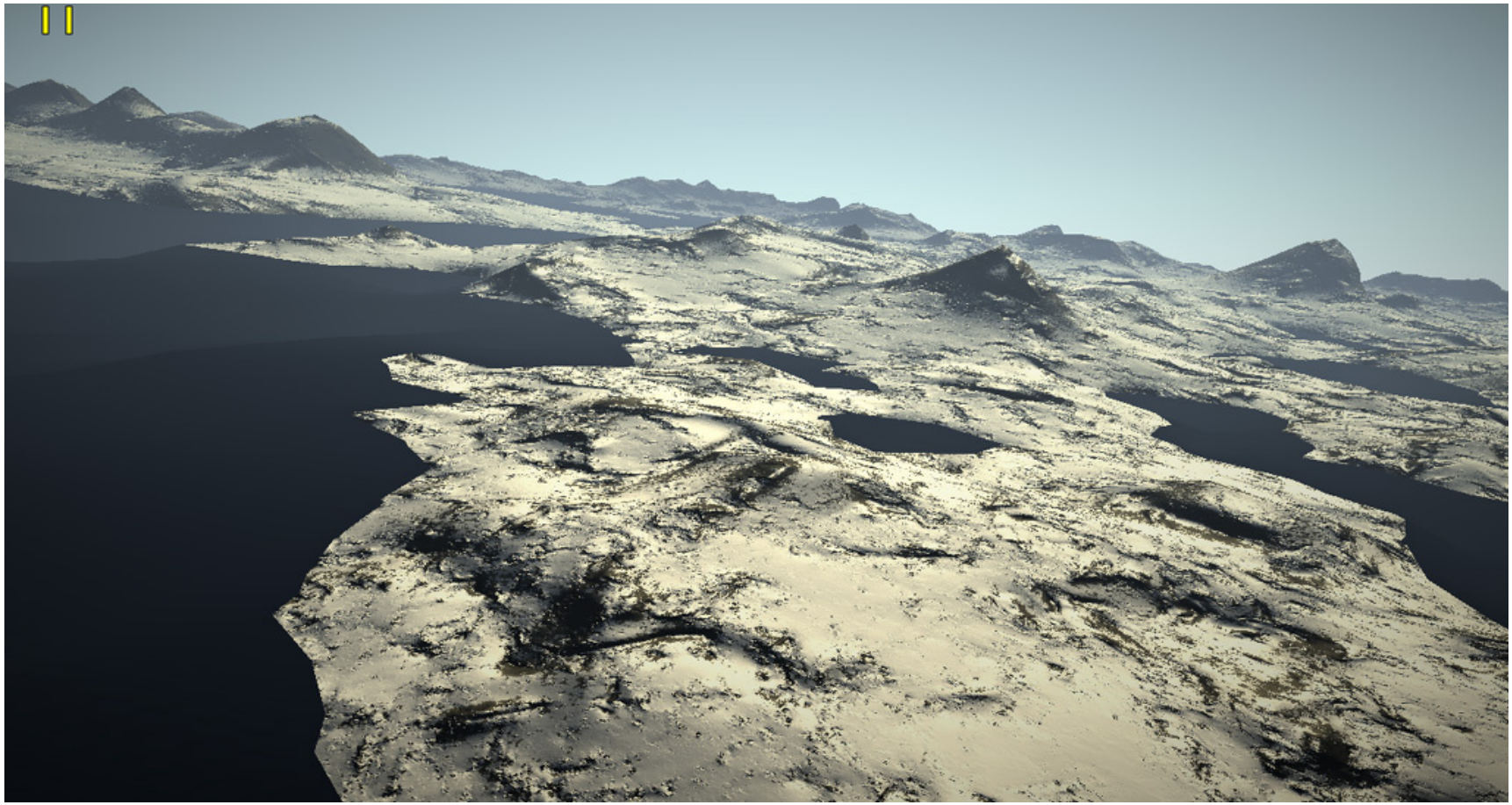
- Optionally resume marching the details in the full screen shader (think on procedural relieve mapping).
- Apply regular procedural texturing and shading in the full screen shader.

motivation :: realtime ixaleno (3)

- Because everybody was saying "very nice this Ixaleno, yes, but for when realtime?"
- But I *knew* it could be done realtime
    - even thou I couldn't openly tell
        - as I often say that "exe or it didn't happen" myself
- So, I had to try the "2+1.000.000" thing to prove it
    - *et voila*, it just worked!

motivation :: realtime ixaleno (3)

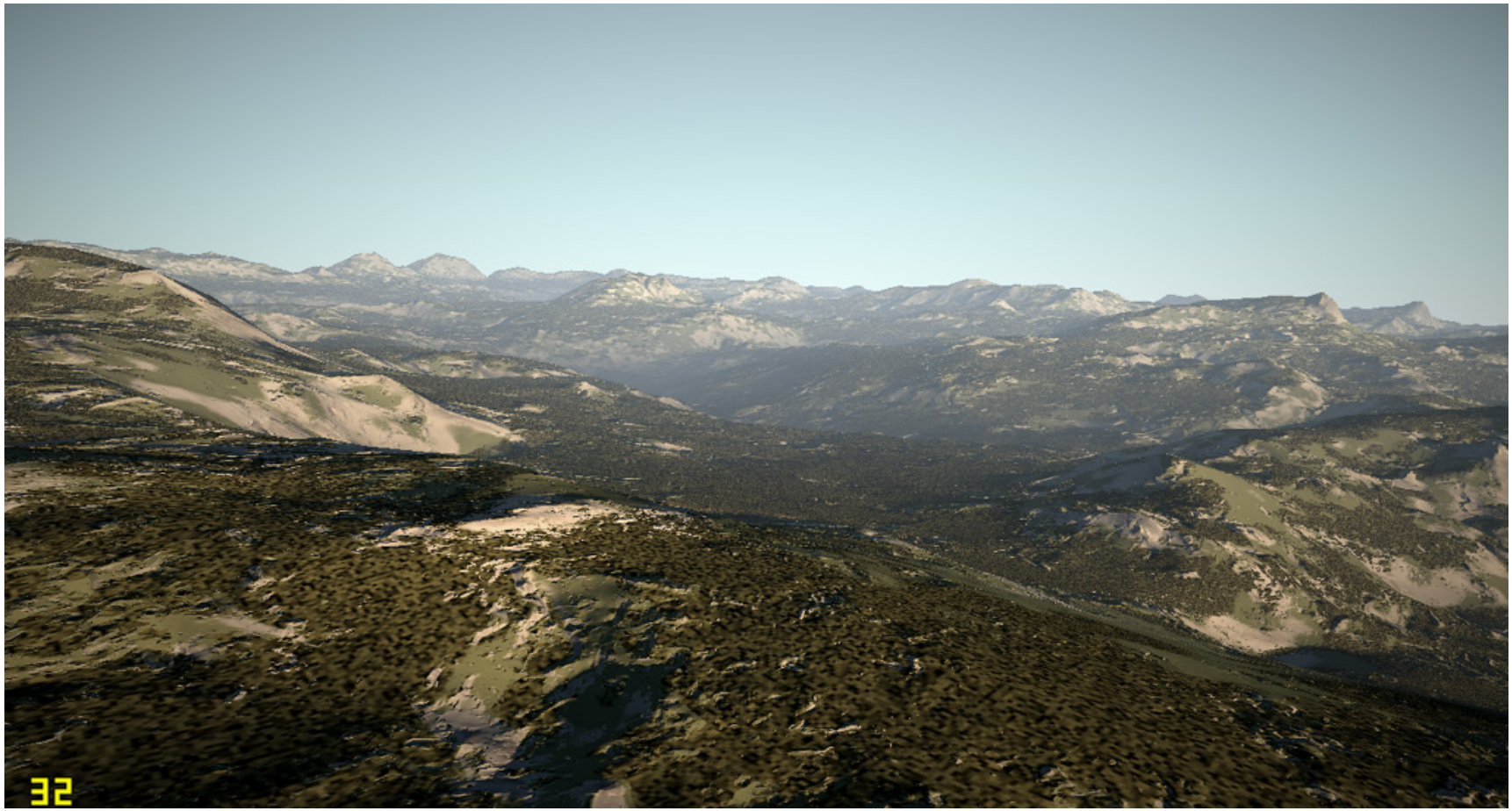
- Tried with uniform grid in clip space



First screenshot taken, not much after Ixaleno,

motivation :: realtime ixaleno (3)

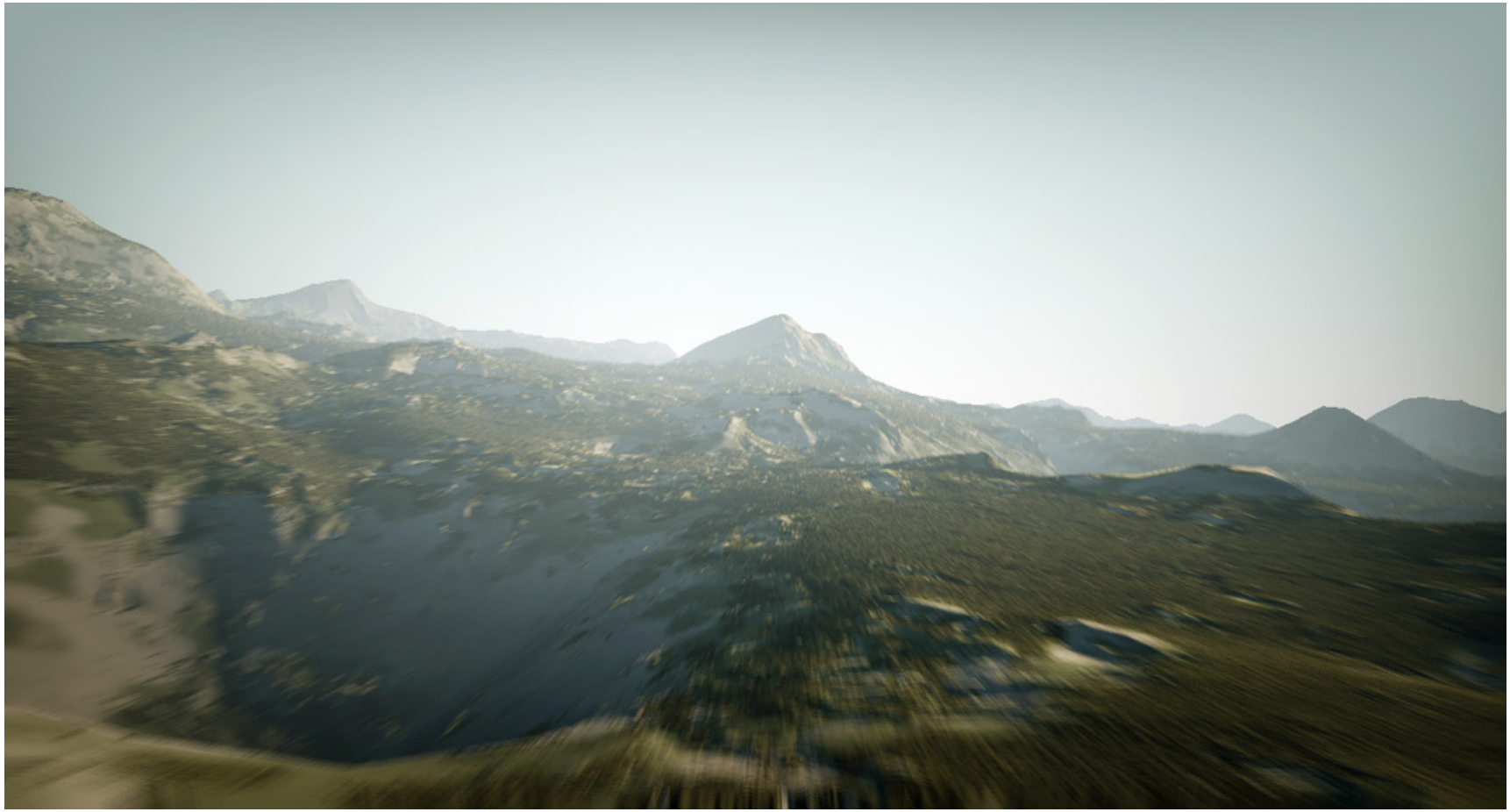
- Changed to camera aligned regular grid (no popping, still inifite terrain support)



Screenshot taken with the final technique, during the experimentation week. Basically realtime Ixaleno in my mobility 8600 GS.

motivation :: realtime ixaleno (3)

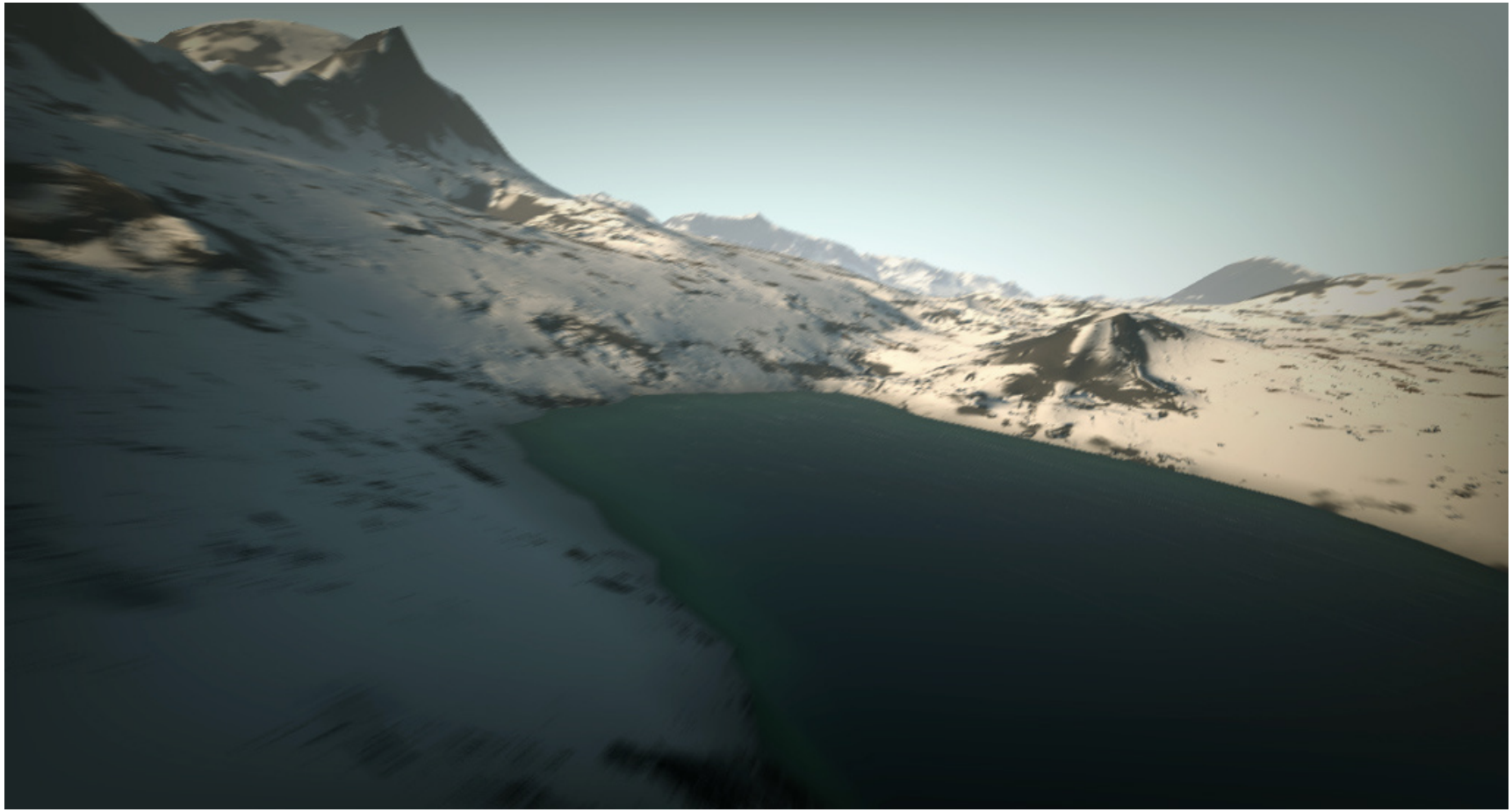
- then added motion blur



Screenshot taken with the final technique, during the experimentation week

motivation :: realtime ixaleno (3)

- and lakes



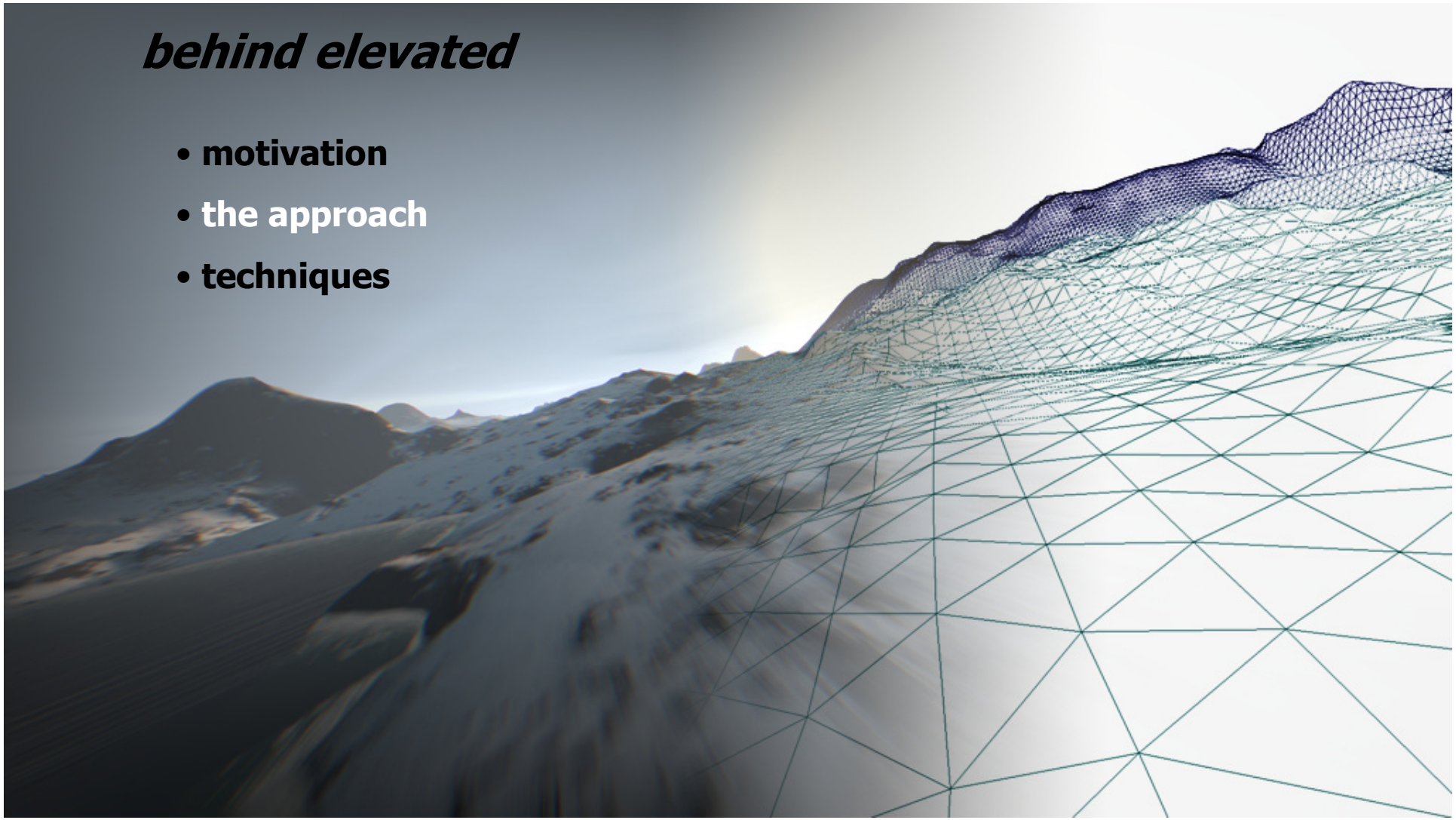
Screenshot taken with the final technique, during the experimentation week

motivation :: realtime ixaleno (3)

- So I knew the 2+1.000.000 was working, that Ixaleno was doable in realtime
  - But it was resting in my disk
  - *"too big for 4k, not good enough for a 64k"*
- Until
  - [Mentor] what´s up? something new?
  - [iq] no, not really. ah, well, yes, i have been trying something
  - [iq] but it´s 3k5 already without mzk or script
  - *iq sends ´realtimeIxalenoScreenshot05.jpg´*
  - [Mentor] hm, looks nice
  - [Mentor] how about making a 4k together
  - [iq] don´t think it´s possible, it´s 3k5, unoptimized, but still 3k5
  - [Mentor] we make it 4k
  - [Mentor] i´m telling puryx
  - [iq] ... ok. wow!

behind elevated

- motivation
- the approach
- techniques

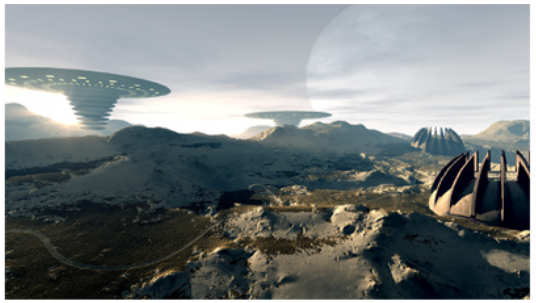


the approach

- The plan was
  - port the intro to DX (my experiments were GL)
  - rewrite the intro in ASM
  - bet on my vision for yet another flyby-over-terrain – “but a good one”
  - use Mentor’s synth
  - apply heavy mentorization to the code and shaders
  - rely on Puryx to produce once more another mzk masterpiece
  - profit (win Breakpoint)


Himalaya / TBC

+


Ixaleno / rgba

=


Elevated / rgba & TBC

the approach :: the elements

- My idea was to make an epic intro
  - With epic music…
    - [iq] …you know, something like The Lord of the Rings
    - [puryx] what?
    - *iq sends ltotr.mp3*
    - [puryx] wtf?
    - [puryx] ok...
    - [iq] after the epic part, we need a "wooaa" moment in the mzk
    - [puryx] ok, I think I can do something with this synth, it´s great. Gimme a couple of days

    Two days later
    - [puryx] ok, have a look
    - *puryx sends iqtest1.mp3*
    - [iq] MUAHAHAHA! this is an instant win.
    - [iq] you did it!

the approach :: the elements

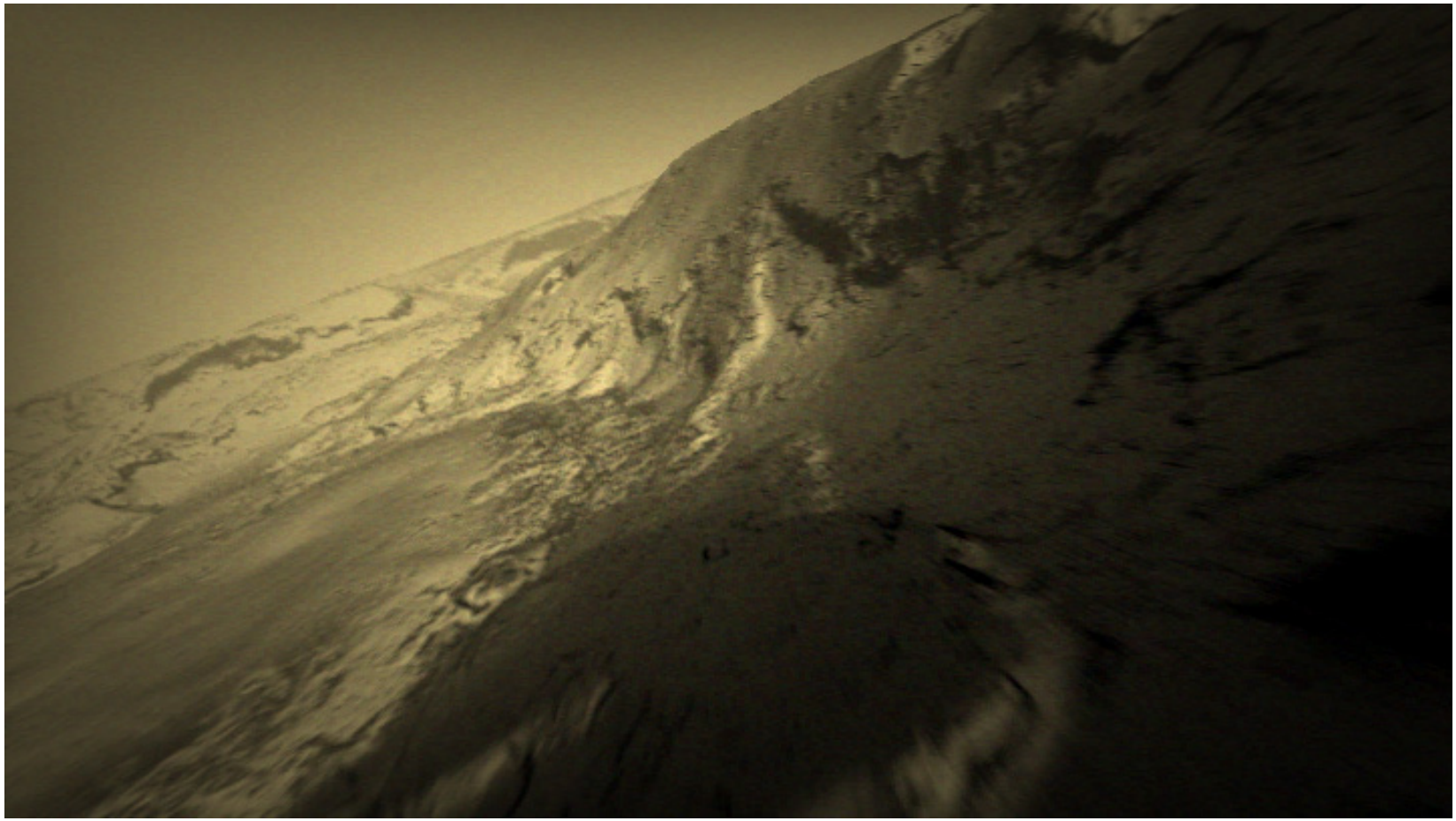
- My idea was to make an epic intro
  - With cool visuals
  - Well, in fact it was an exercise to see where I could go into realism
    - In realtime
    - Without shadows
    - Without AO or GI
    - Without HDR or tonemapping
  - Without using any reference image, working just from my imagination
    - It happened to be a nightmare because my monitors aren´t calibrated
      - looks ok in the latptop -> looks crap in the desktop
      - looks ok in the desktop -> looks crap at work
      - looks ok at work -> looks crap in the laptop

the approach :: the elements

- My idea was to make an epic intro
  - With cinematic look, like shooted with a real camera
    - Image features
      - Image brightness flicker at exactly 25 hz
      - Image grain, at exactly 25 hz
      - Motion blur, at exactly 25 hz (not based on previous frame)
      - Vigneting
      - Chromatic dispersion
      - Dust (removed in the final version)
    - Belivable camera movements
      - Pure sin/cos cameras are too mathematical
      - Real cameras have weight, innertia
      - They shake

the approach :: the elements

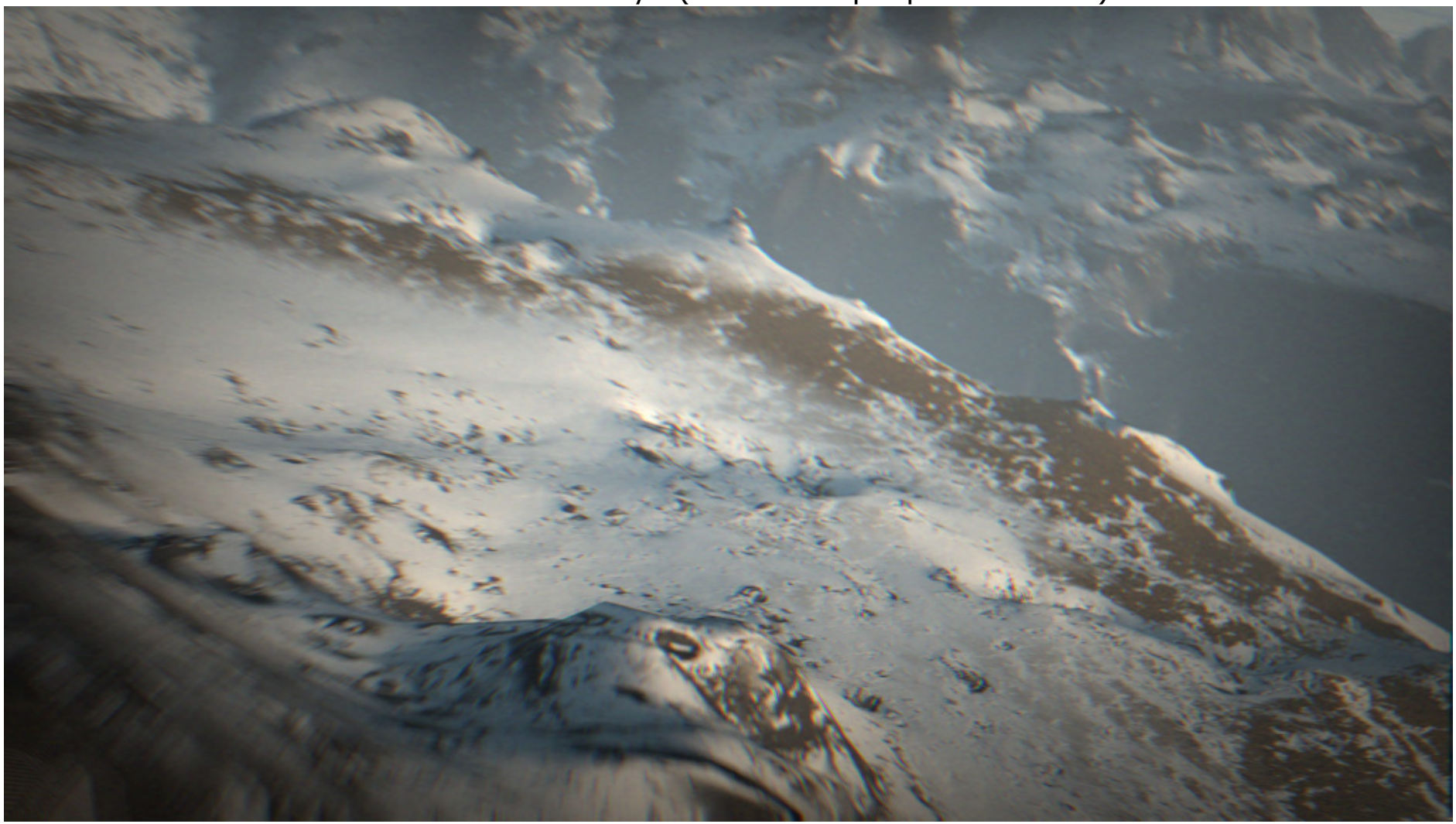
- I experimented with "old movie" look



Early experiments on film look postprocessing shaders – completely discarded after a short discussion about it

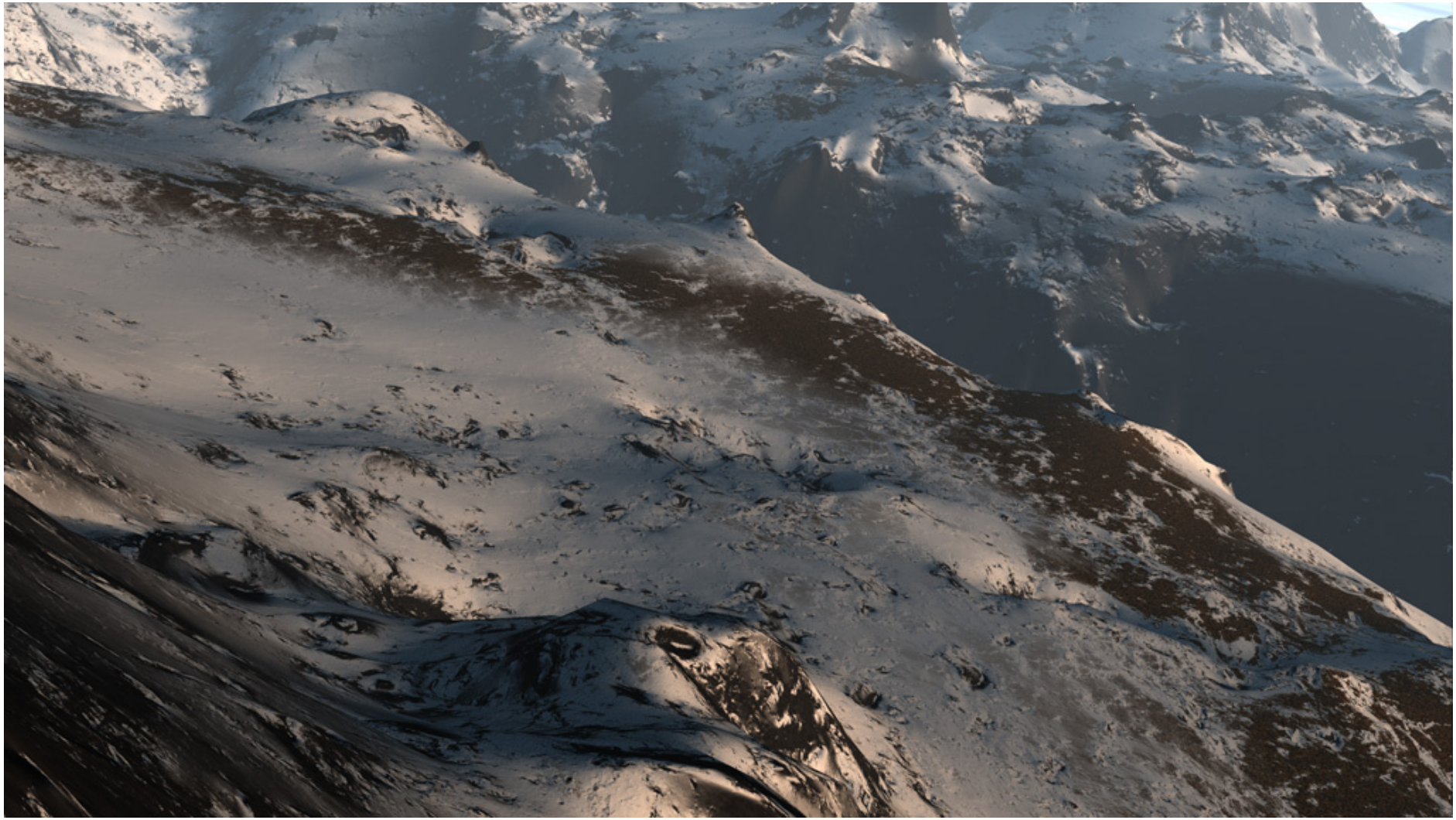
the approach :: the elements

- In the end we went for a more 70s camera style (result of the postprocess shader)



the approach :: the elements

- Before postprocessing, for comparison

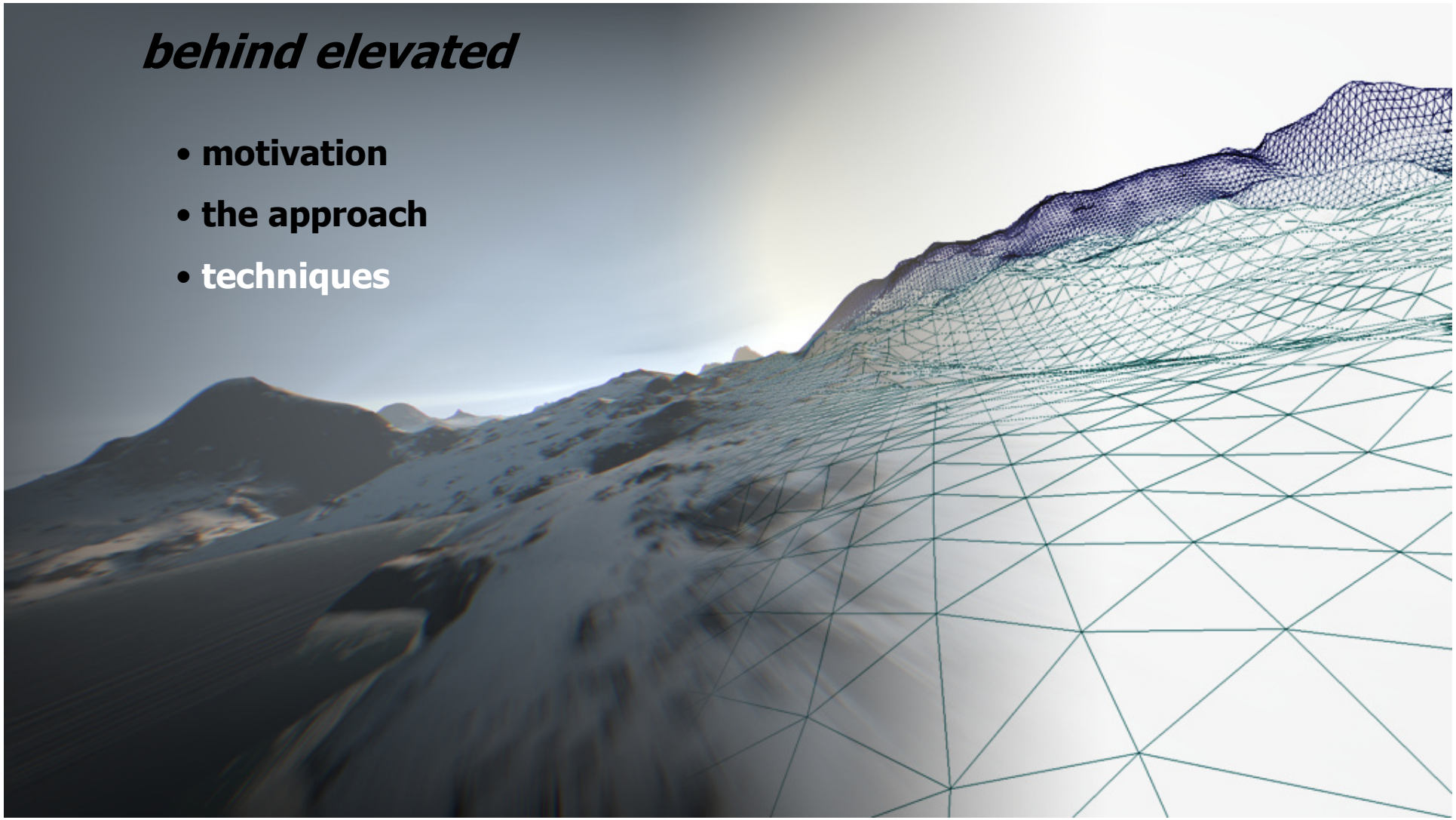


the approach :: the elements

- My idea was to make an epic intro
  - With cinematic look, like taken with a real camera. We had some dissagreements here
    - I absolutelly wanted to avoid the CG look - the danish part of the team wanted something sharp and shinny
    - I wanted a hand-held TV camera - they wanted a sts04 like smooth lovely cameras
    - I wanted a realistic scenery - they wanted more action in the scene...
      - [Mentor] ... like flashing rays in the sky
      - [iq] wait, like... what? what?
      - [Mentor] hey, we kept the grain, light flicker and camera shakes...
      - [iq] very true. ok yeah, give it a try
      - [Mentor] I tried already... ☺
      - *mentor sends bp09_h.rar*
      - [iq] damn, this can work indeed!

behind elevated

- motivation
- the approach
- techniques



the techniques :: modeling

- A shader that displaces vertically the vertices of a subdivided flat plane.
- The same displacement function is used to compute surface features so they seamless and naturally follow the geometry.
- The displacement function is used to do camera collisions too
    - Therefore the camera movements HAD to be done in a shader too
- In the GL experiments the mesh was moving with the camera, making the terrain infinite in a true way.
    - For size reasons, in final intro the mesh is static and centered at 0,0,0
- Low tesselation, no space for making a perspective distortion to extend the view distance (even if it´s just few bytes)

the techniques :: modeling

- Analytic (value)noise derivatives
  - Faster (no need to evaluate 4 times for central differences method)
  - Useful for approximating local neighborhood (Taylor series)
    - For erosion?

- Reminder of regular value noise: given one of the grid cells with corner random values *a*, *b*, *c* and *d*, for a point (x,y) in 0..1 within the cell, the noise n(x,y) is the bilinear interpolation of the four corner values thru *u* and *v*.

- More info on derivatives of value-noise: http://iquilezles.org/www/articles/morenoise/morenoise.htm

$$k_0 = a$$
$$k_1 = b - a$$
$$k_2 = c - a$$
$$k_3 = a - b - c + d$$

$$u = 6x^5 - 15x^4 + 10x^3$$
$$v = 6y^5 - 15y^4 + 10y^3$$

$$u' = 30x^4 - 60x^3 + 30x^2$$
$$v' = 30y^4 - 60y^3 + 30y^2$$

$$n(x, y) = k_0 + k_1 \cdot u + k_2 \cdot v + k_3 \cdot u \cdot v$$

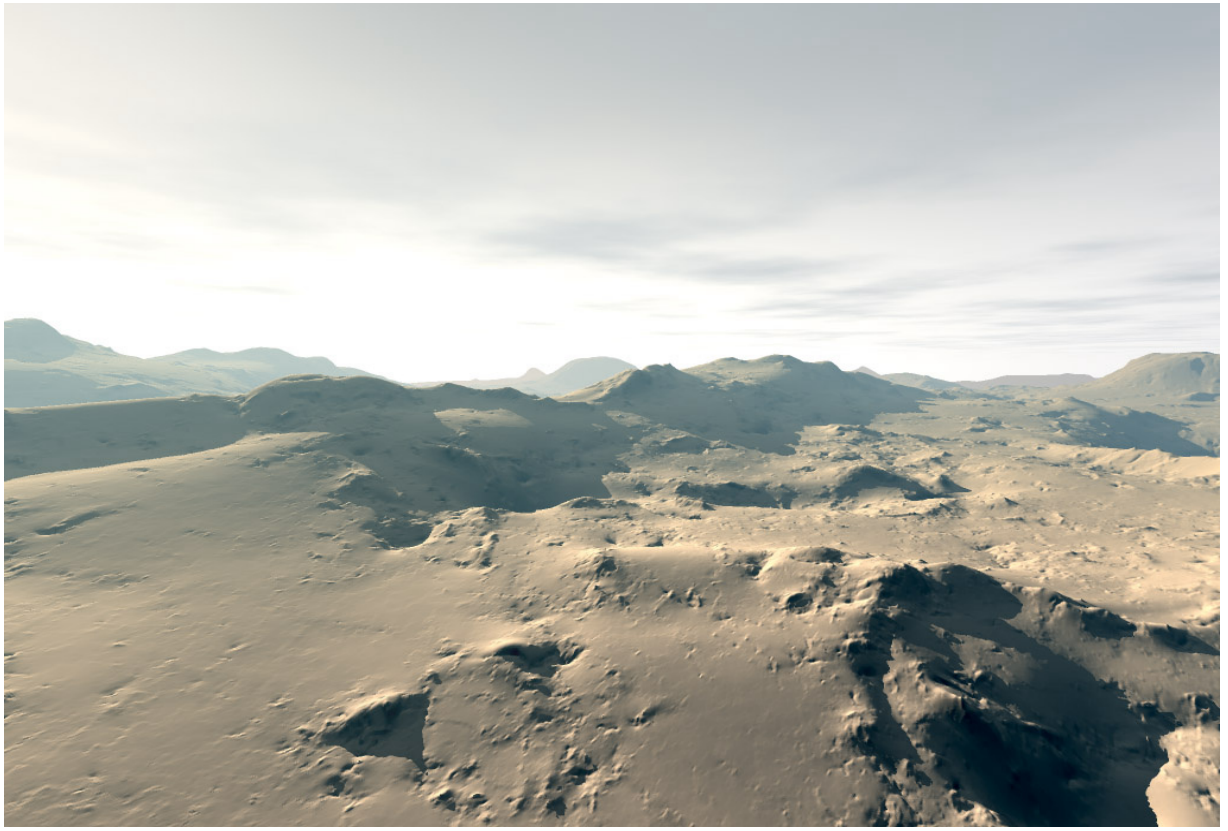
$$\frac{dn}{dx} = (k_1 + k_3 \cdot v) \cdot u'$$
$$\frac{dn}{dy} = (k_2 + k_3 \cdot u) \cdot v'$$

the techniques :: modeling

- A (failed) (and quick) attempt to erosion
  - But at least better than pure old "fractal" terrain (with both smooth and rought parts)

```
float terrain( vec2 p )
{
    float a = 0.0;
    float b = 1.0;
    vec2 c = vec2(0.0);
    for(int i=0; i<16; i++ )
    {
      vec3 n = noise2f(p);
      c += n.yz;
      a += b*n.x / (1.0+dot(c,c));
      b *= 0.5;
      p = mat2x2(1.6,-1.2,1.2,1.6)*p;
    }
    return a;
}
```

- Without the red part, the code reduces to the traditional fbm construction.



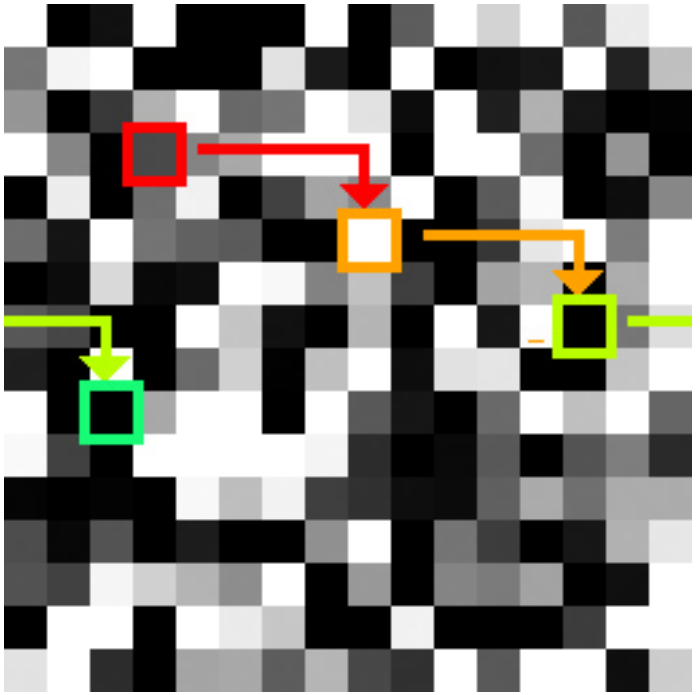
the techniques :: camera

- Took quite long to decide for a camera system.
  - Random camera paths can give cool shots, BUT are difficult to find!...
  - Manual cameras (splines) are controlable, but require lot of space
- So we did a mix. Random cameras, plus tunneable parameters

the techniques :: camera

- X and Z are simply two octaves of cosinus functions. Frequencies and phases define different cameras. These were chosen randomly from a "random" texture. This texture was in fact the same one used for noise() ;)
  - Therefore, *only* 256x256 = 65536 cameras possible
  - We only explored two rows (512 cameras)

```
f1 = randomtexture[ texel+=k ]
f2 = randomtexture[ texel+=k ]
f3 = randomtexture[ texel+=k ]
f4 = randomtexture[ texel+=k ]
x(t) = 16*cos(f1*t+f2) + 8*cos(2*f3*t+f4)
y(t) = 16*cos(f4*t+f3) + 8*cos(2*f2*t+f1)
```



- Only need 16 bit (the uv of the texel) to define a complete camera path

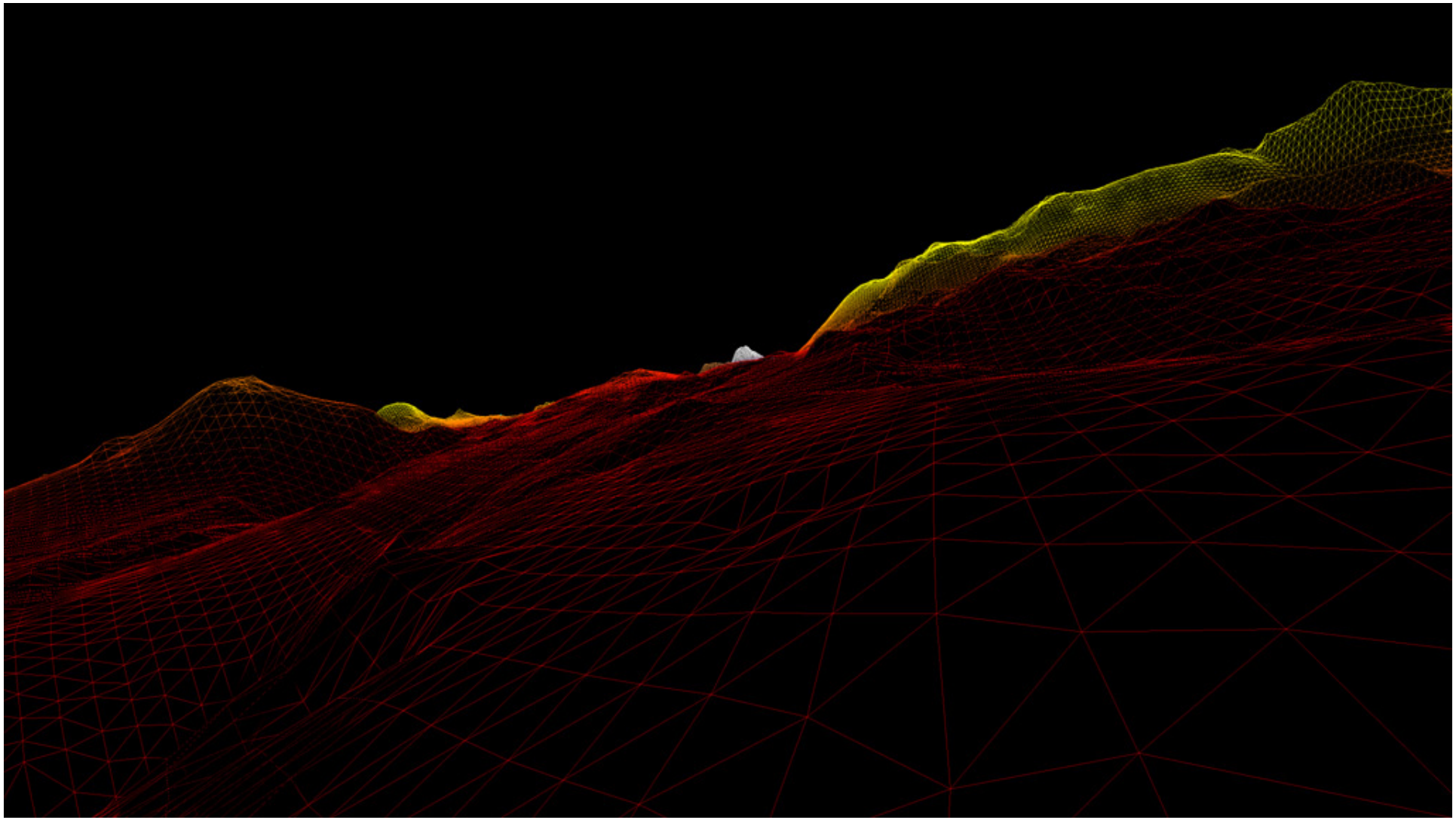
the techniques :: camera

- Y is computed from terrain altitude (collision detection) …
  - Helps to keep the camera attached to the ground and IN the world, like a real camera and not just an external flying entity
- … PLUS a user controled offset
  - to make terrestial or aereal cameras (when applied to the position)
  - to control the view direction too (when applied to the target)

the techniques :: camera

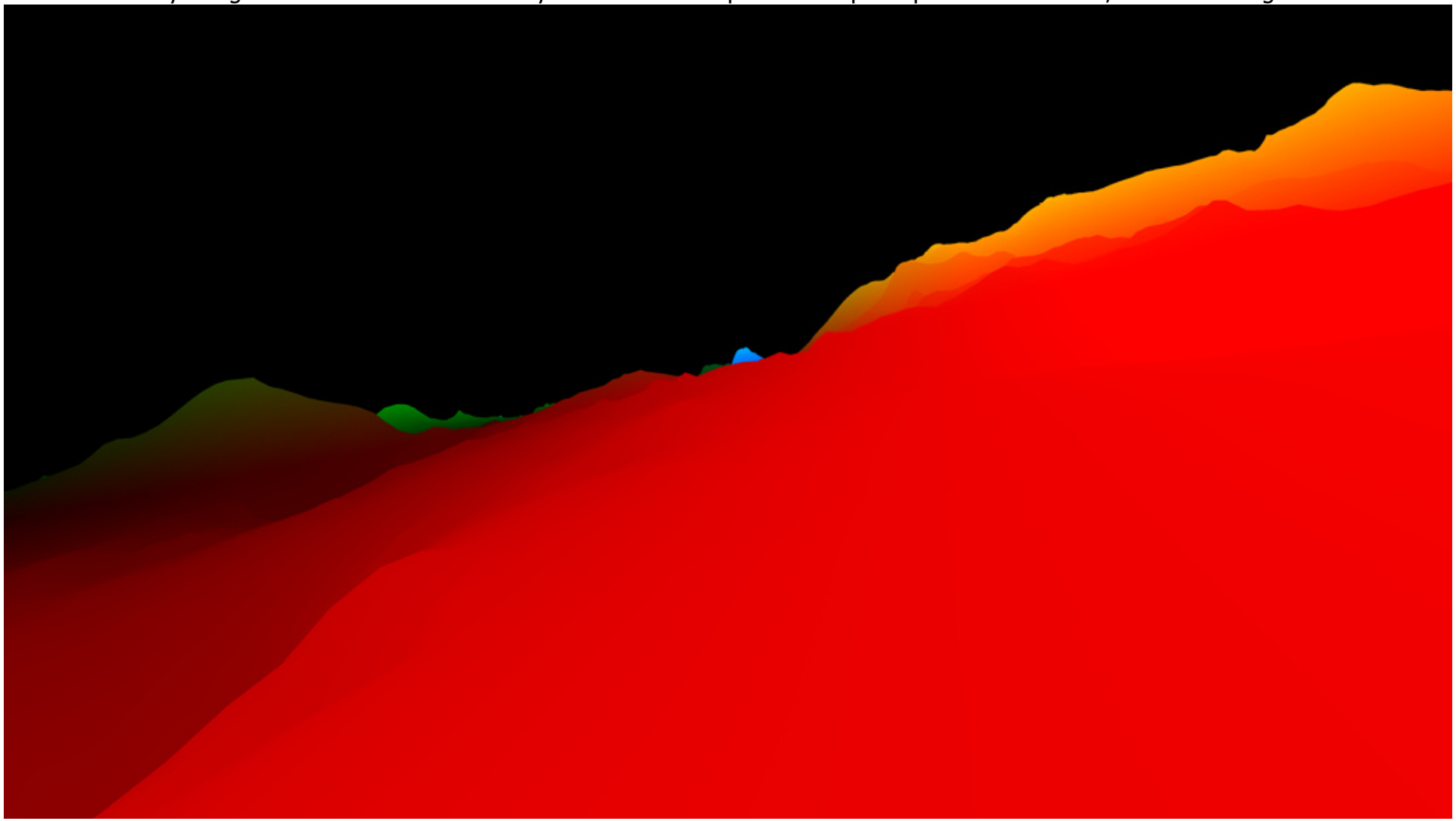
- XYZ is modified with an additive displacement to simulate hand-held camera.
    - to save space, this displacement is simply the terrain function feeded with time instead of world coodinates.
- Camera target follows exactly the same formula and parameters as position, but with a different random texel
- Other parameters: camera speed, fov
- Total, 8 bytes per camera path (2 for position texel seed, 2 for target texel seed, 1 for position y offset, 1 for target y offset, 1 for speed, 1 for fov)

the techniques :: shading

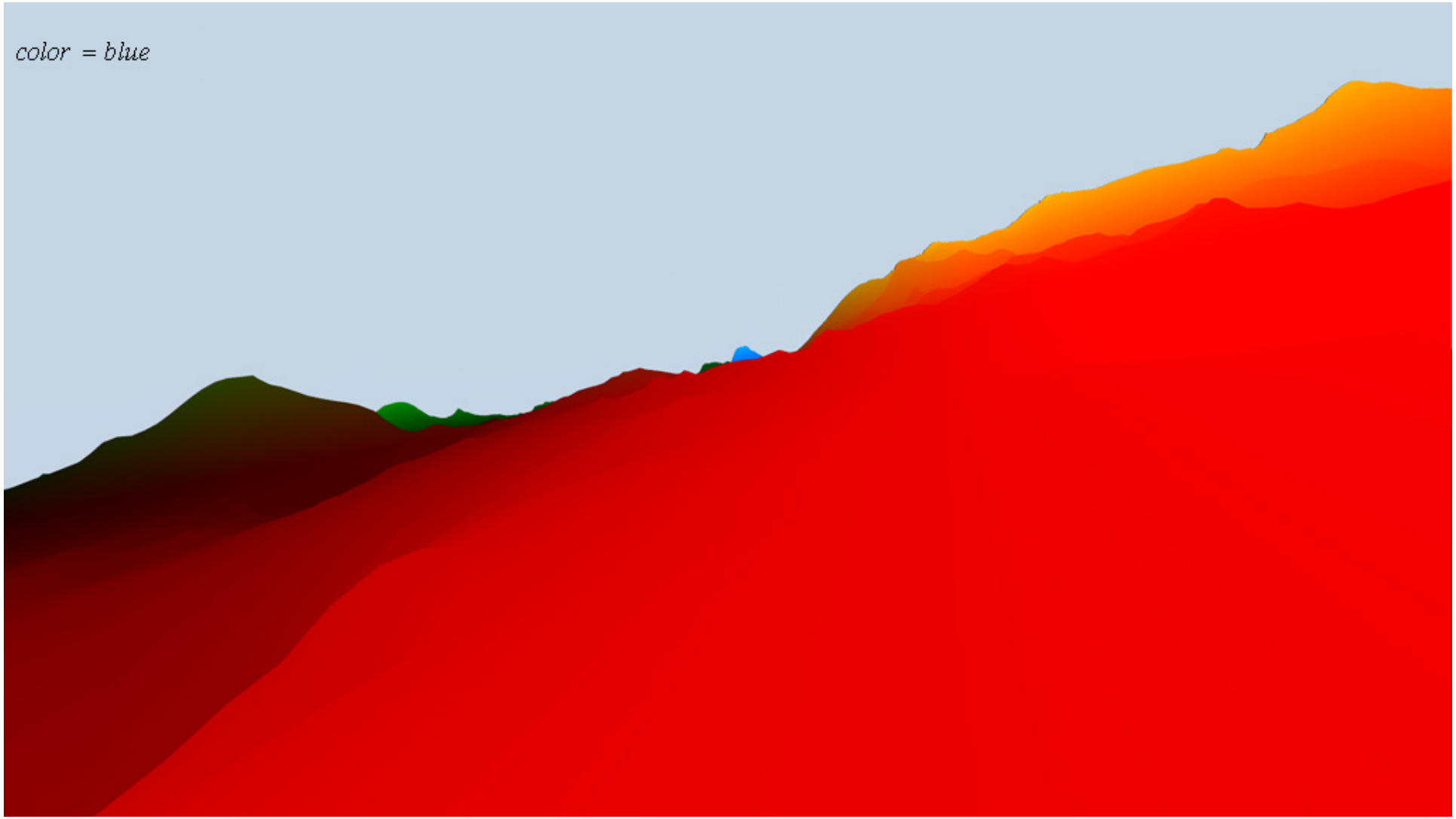


the techniques :: shading

This is the only thing drawn in the traditional way. The rest is computed in a quad+procedural shader, from this image.

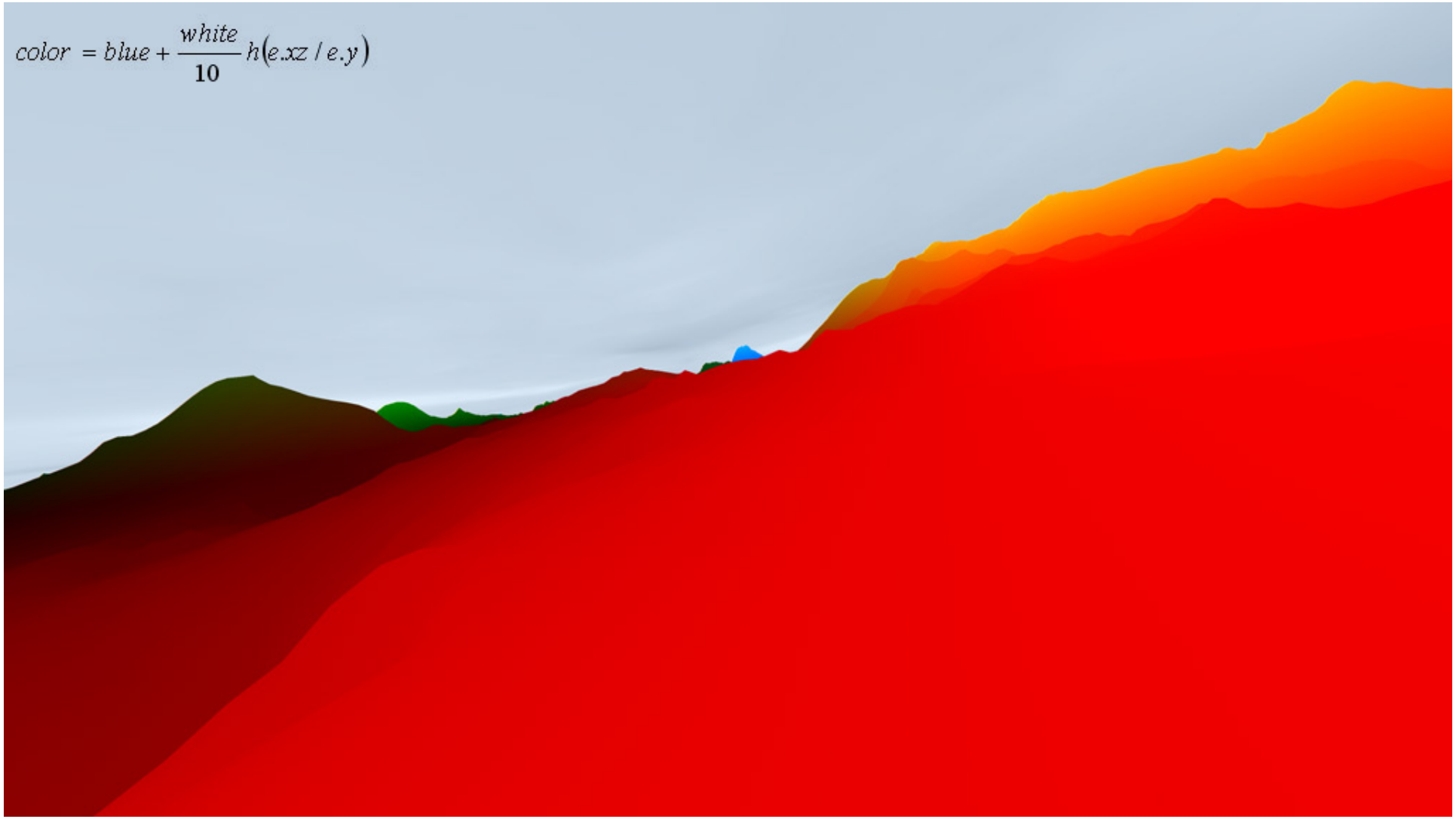


the techniques :: shading



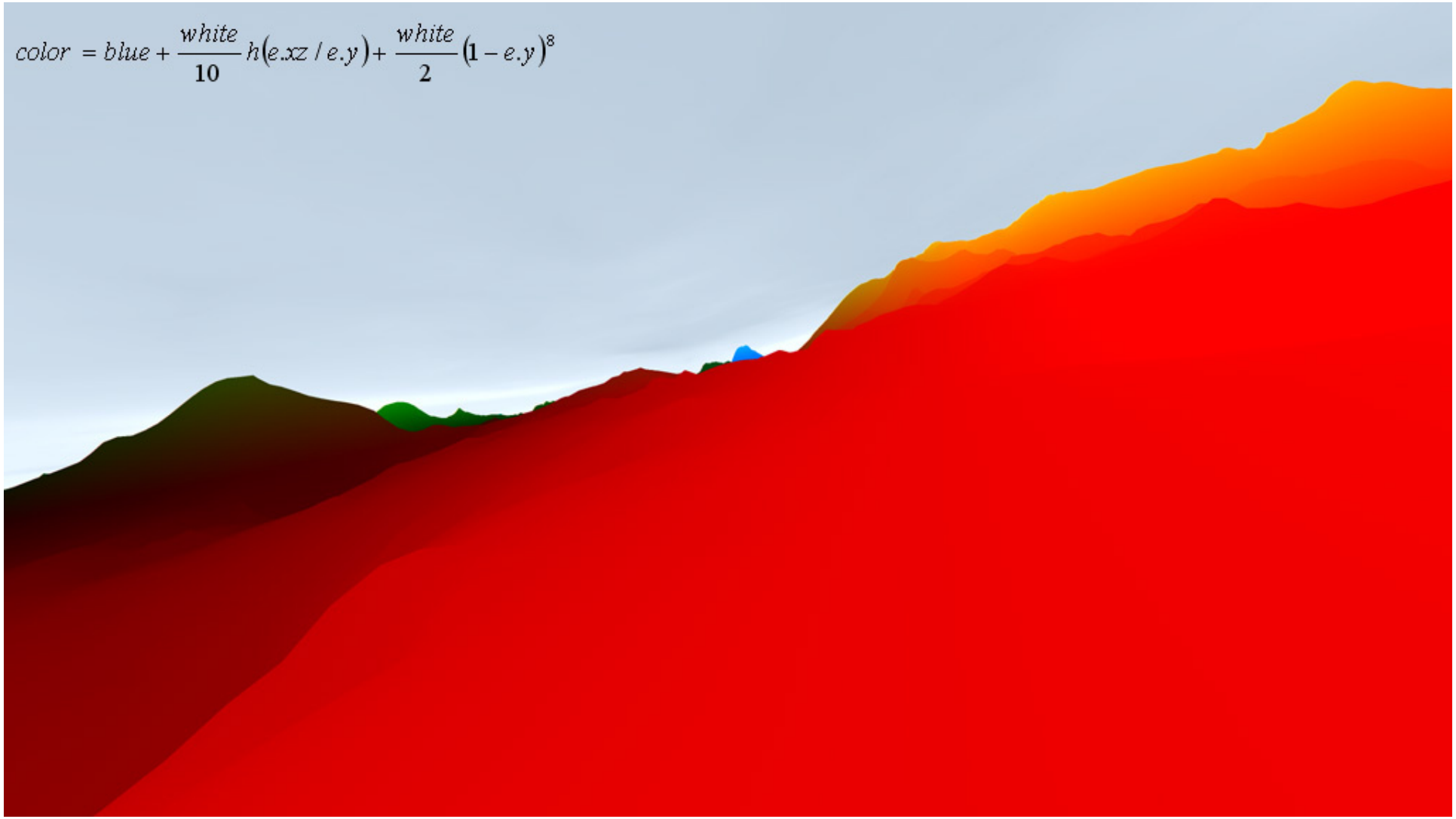
the techniques :: shading

$$color = blue + \frac{white}{10} h(e.xz / e.y)$$

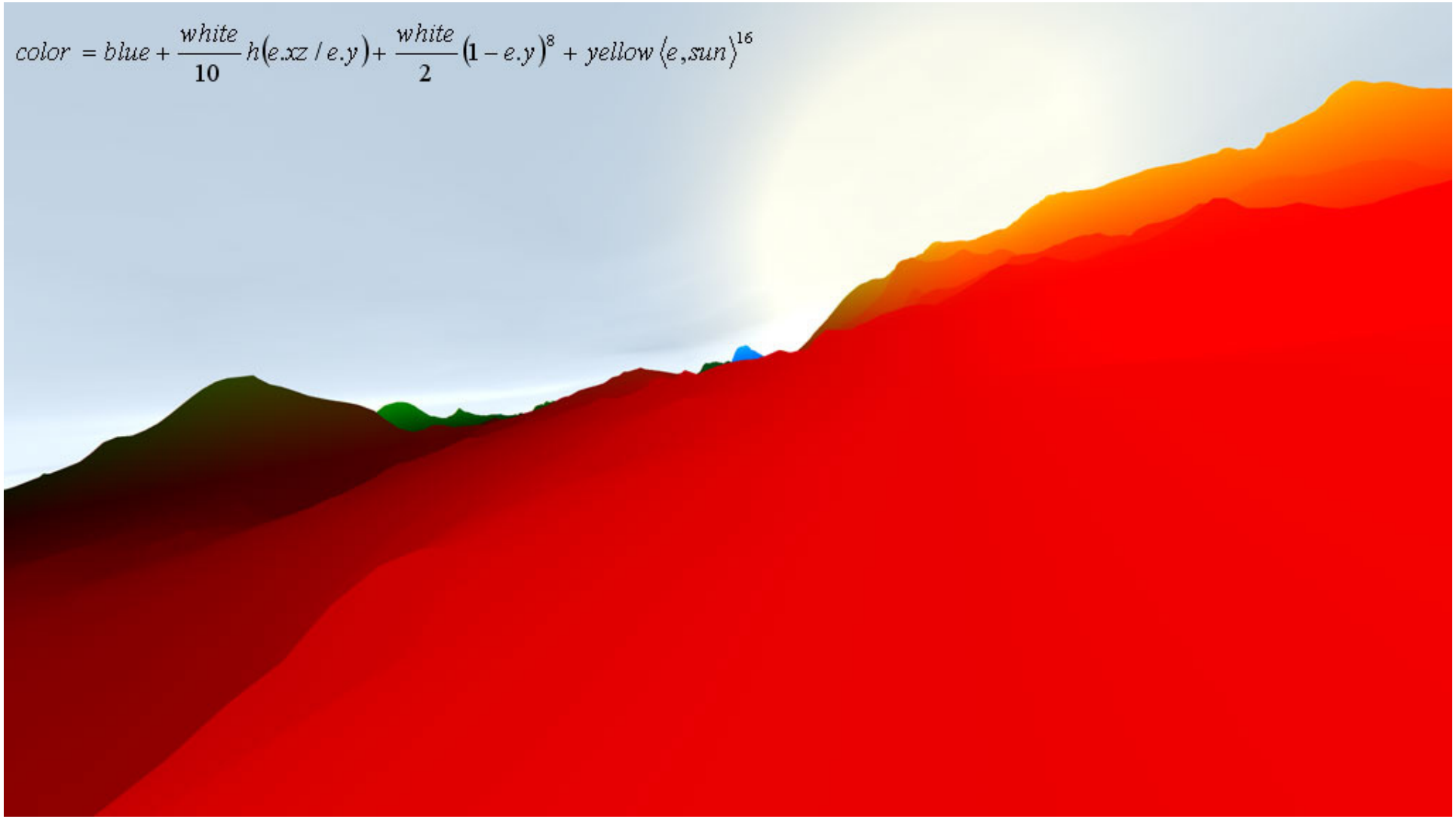


the techniques :: shading

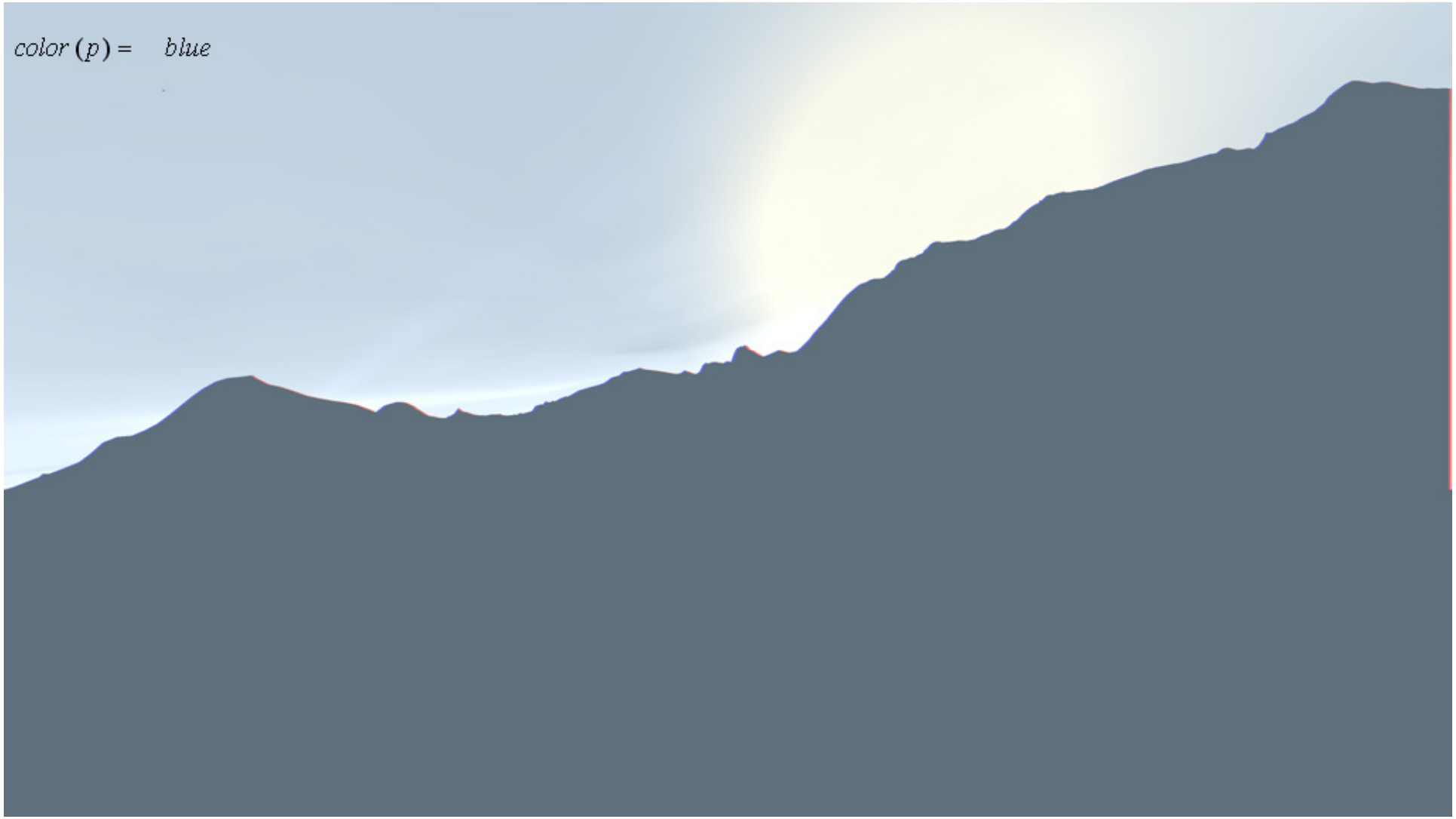
$$color = blue + \frac{white}{10} h(e.xz / e.y) + \frac{white}{2} (1 - e.y)^8$$



the techniques :: shading

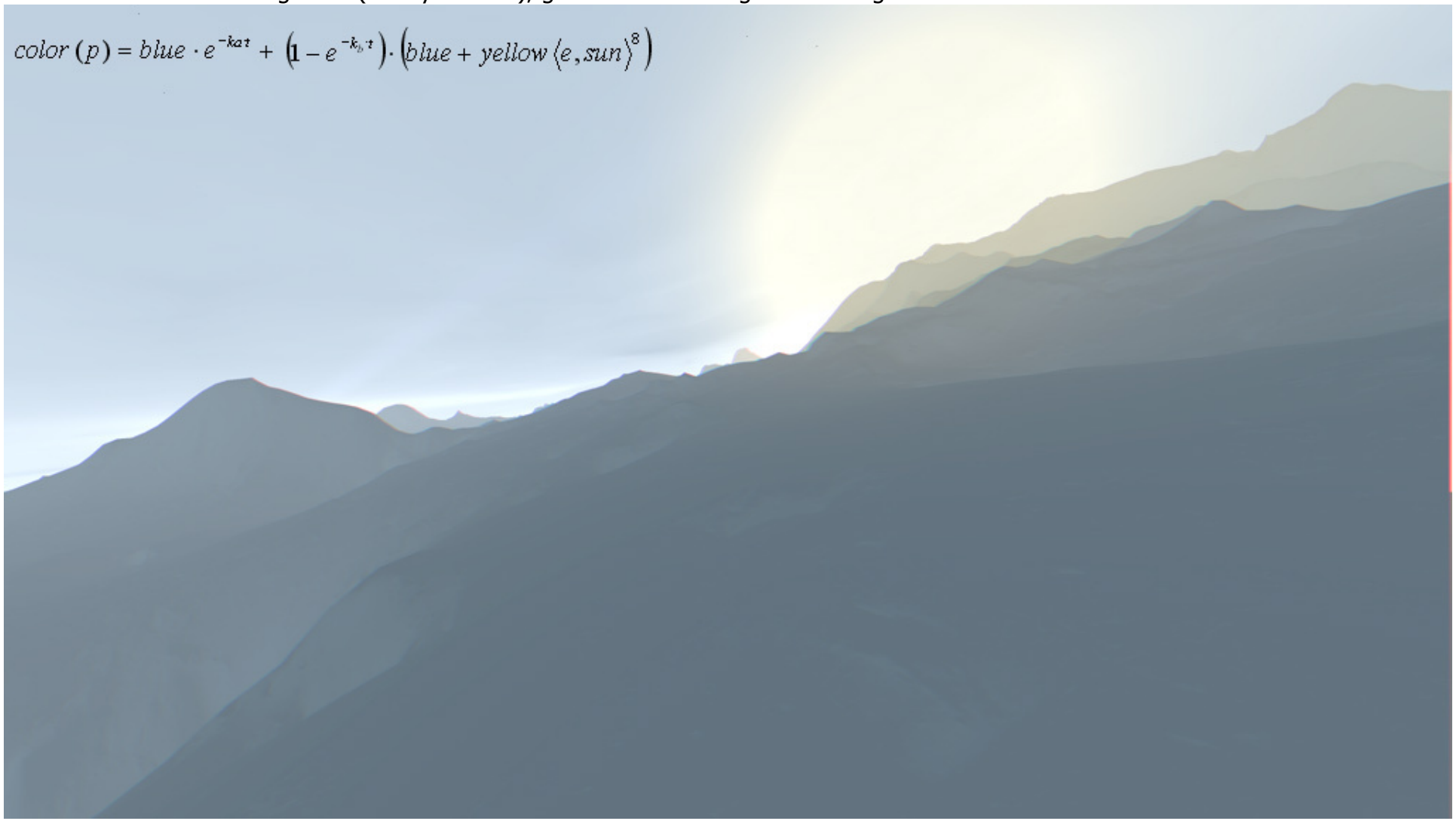


the techniques :: shading



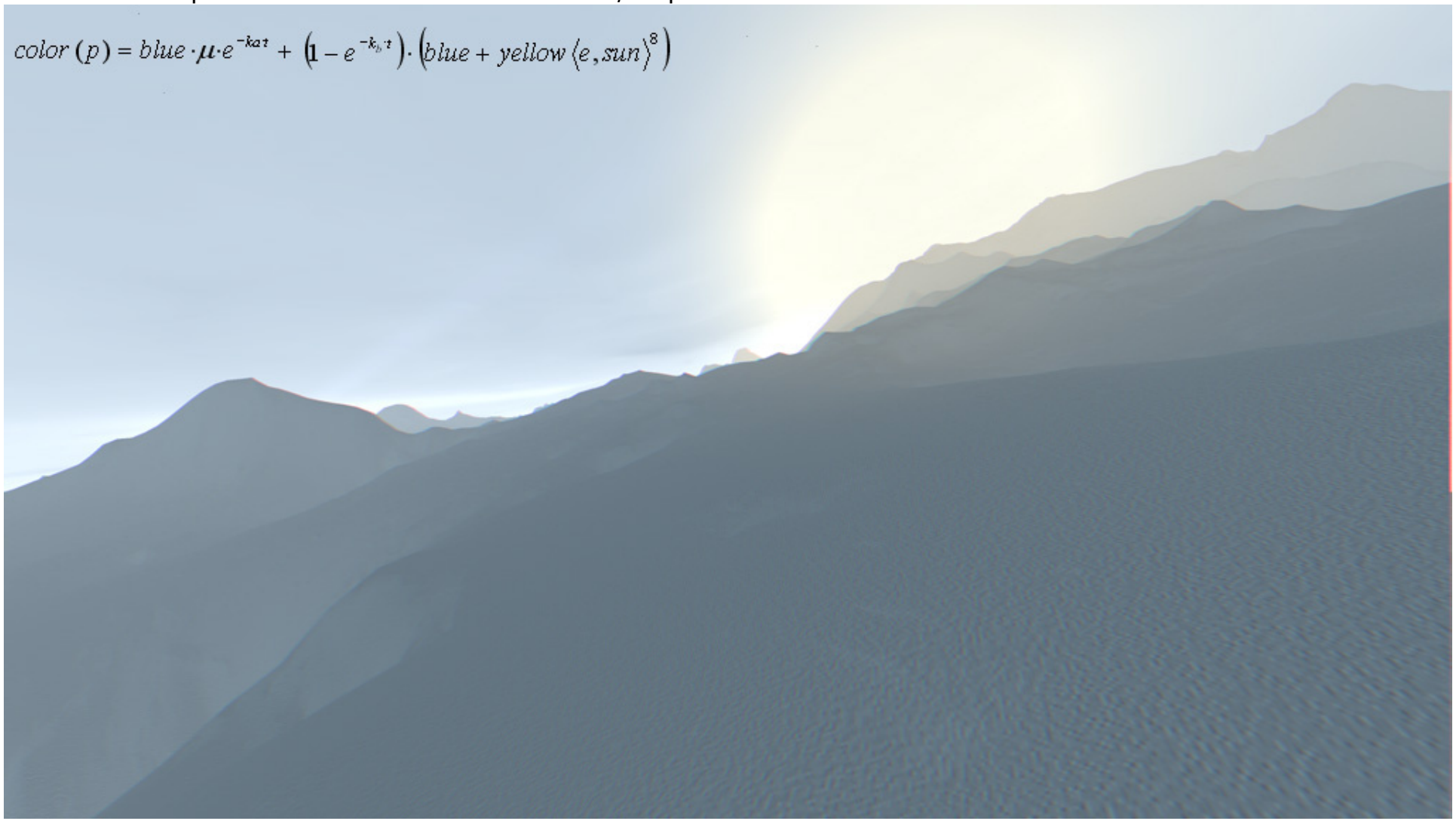
the techniques :: shading

Note the non uniform fog color (blue-yellowish), good to simulate light scattering.



the techniques :: shading

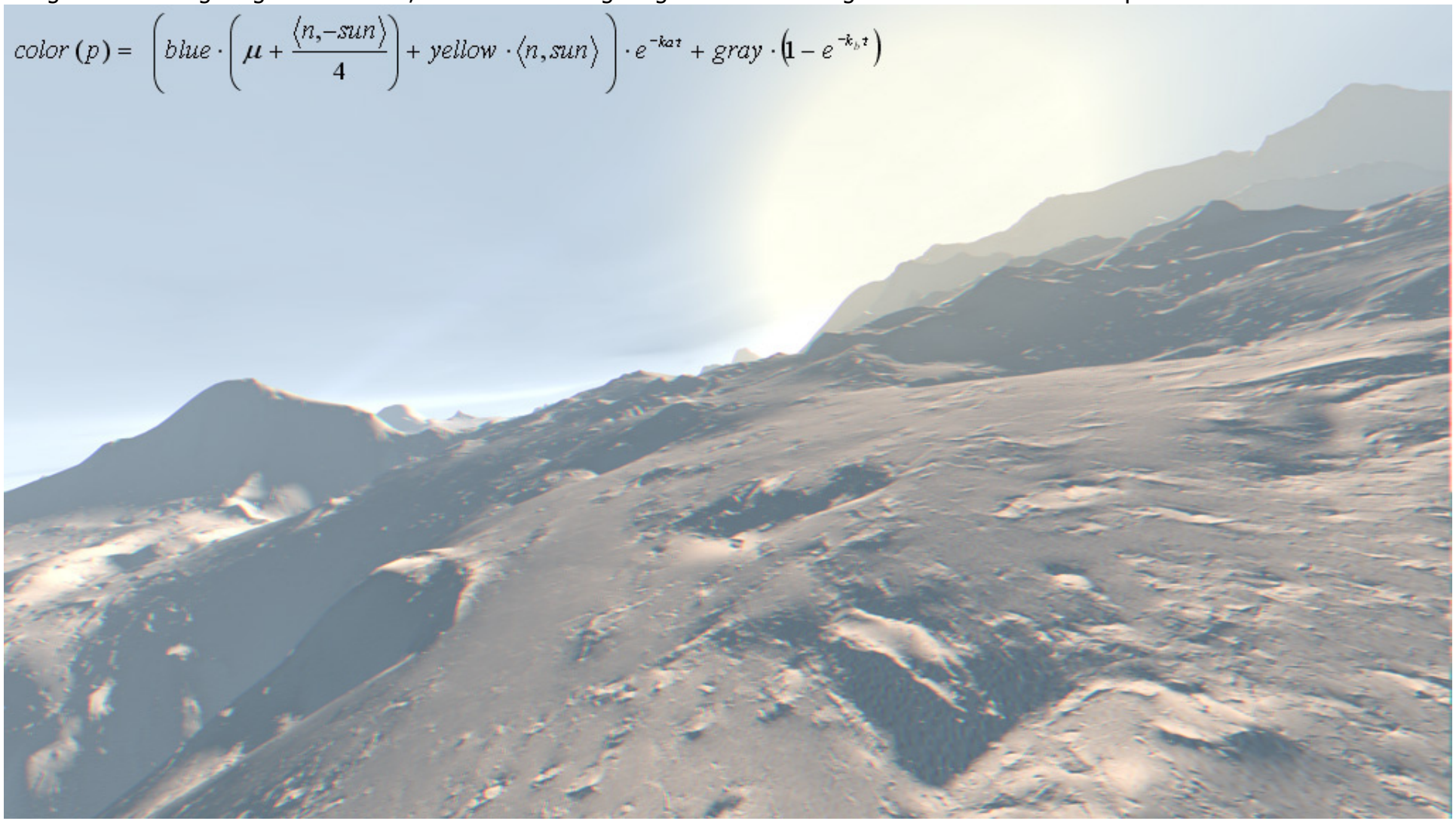
Some noise will provide more variation to the shadowed/flat parts



the techniques :: shading

Regular diffuse lighting. For ambient, with some backlighting is added to bring details to the shadowed parts.

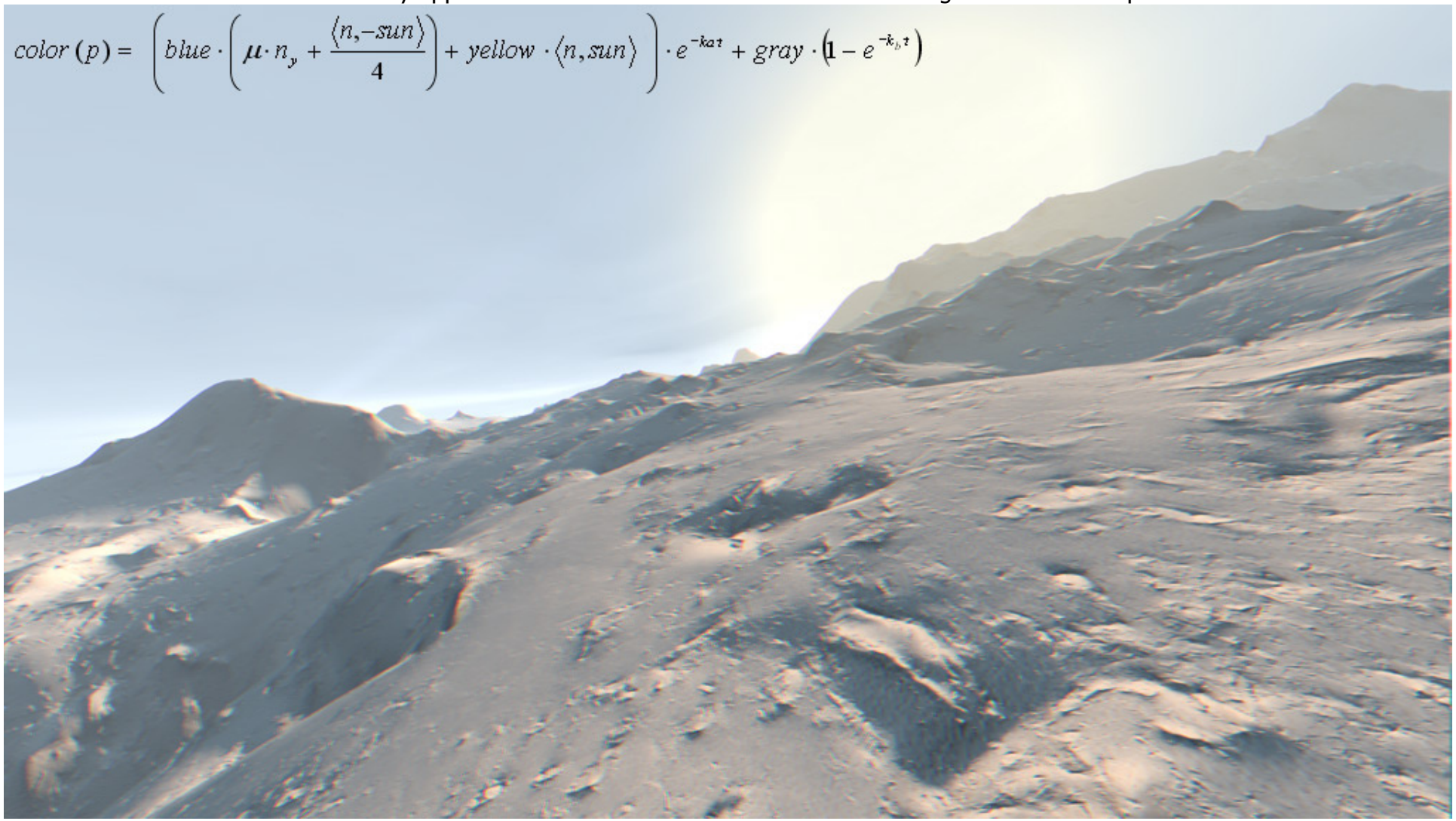
$$color(p) = \left(blue \cdot \left(\mu + \frac{\langle n, -sun \rangle}{4} \right) + yellow \cdot \langle n, sun \rangle \right) \cdot e^{-k_a t} + gray \cdot \left(1 - e^{-k_b t}\right)$$



the techniques :: shading

Modulation of ambient with normal.y approximates a bit of ambient occlusion and brings more detail to parts in shadow.

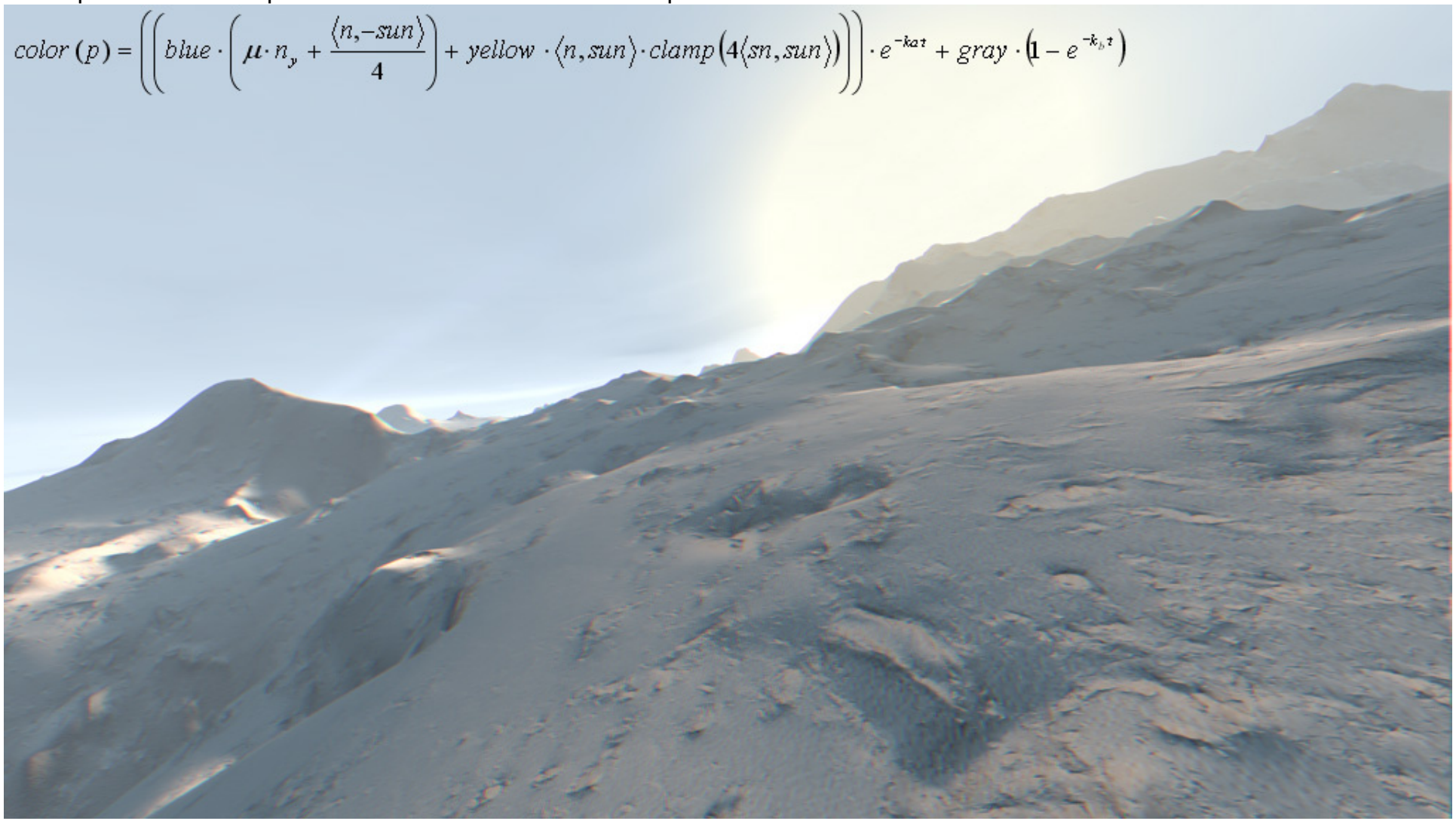
$$color(p) = \left(blue \cdot \left(\mu \cdot n_y + \frac{\langle n, -sun \rangle}{4} \right) + yellow \cdot \langle n, sun \rangle \right) \cdot e^{-k a t} + gray \cdot \left(1 - e^{-k_b t}\right)$$



the techniques :: shading

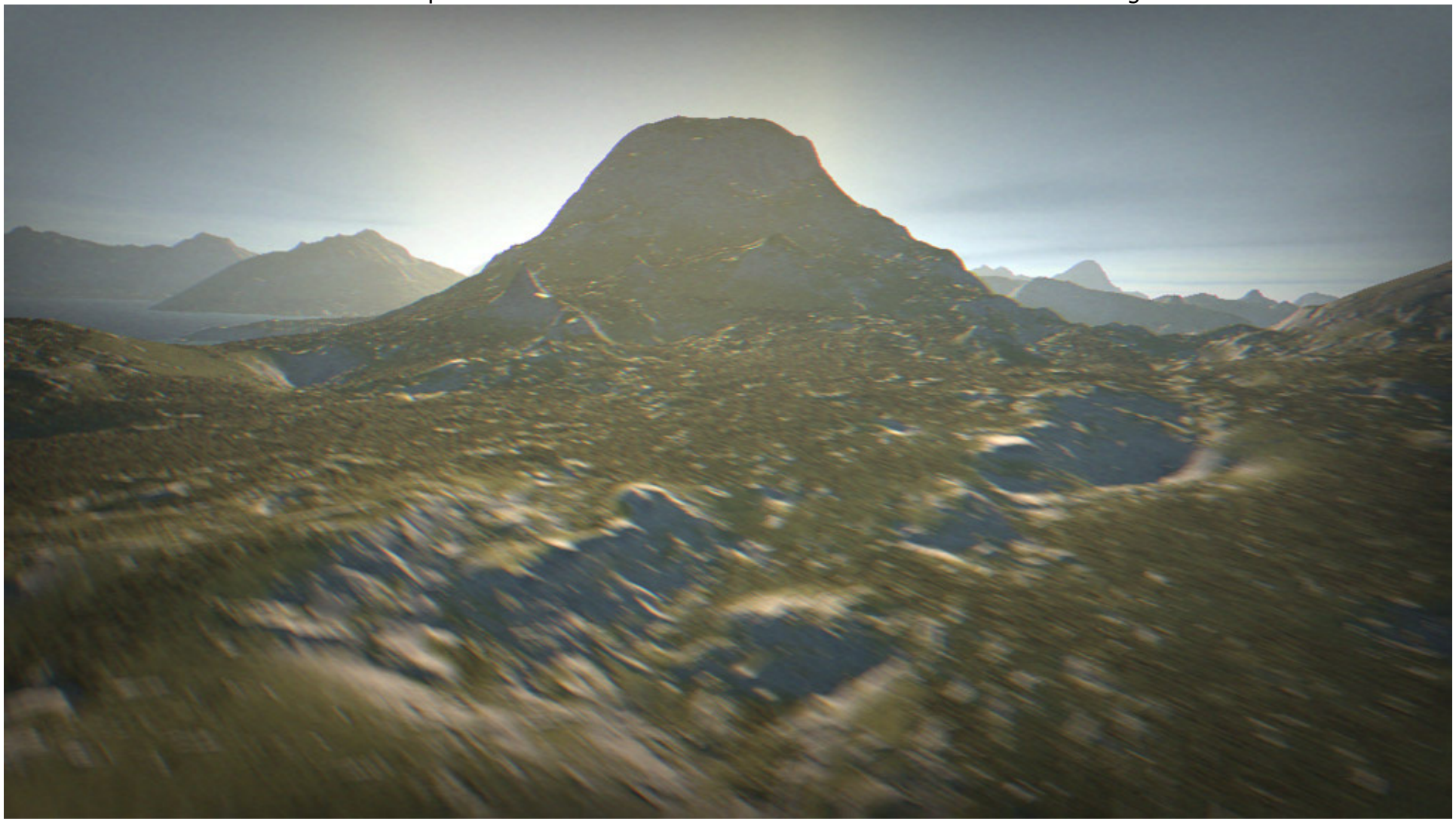
A simple saturated dot product with a smoothed normal can provide fake shadows…

$$color(p) = \left(\left(blue \cdot \left(\mu \cdot n_y + \frac{\langle n, -sun \rangle}{4} \right) + yellow \cdot \langle n, sun \rangle \cdot clamp\left(4\langle sn, sun \rangle\right) \right) \cdot e^{-ka\,t} + gray \cdot \left(1 - e^{-k_b\,t}\right) \right)$$

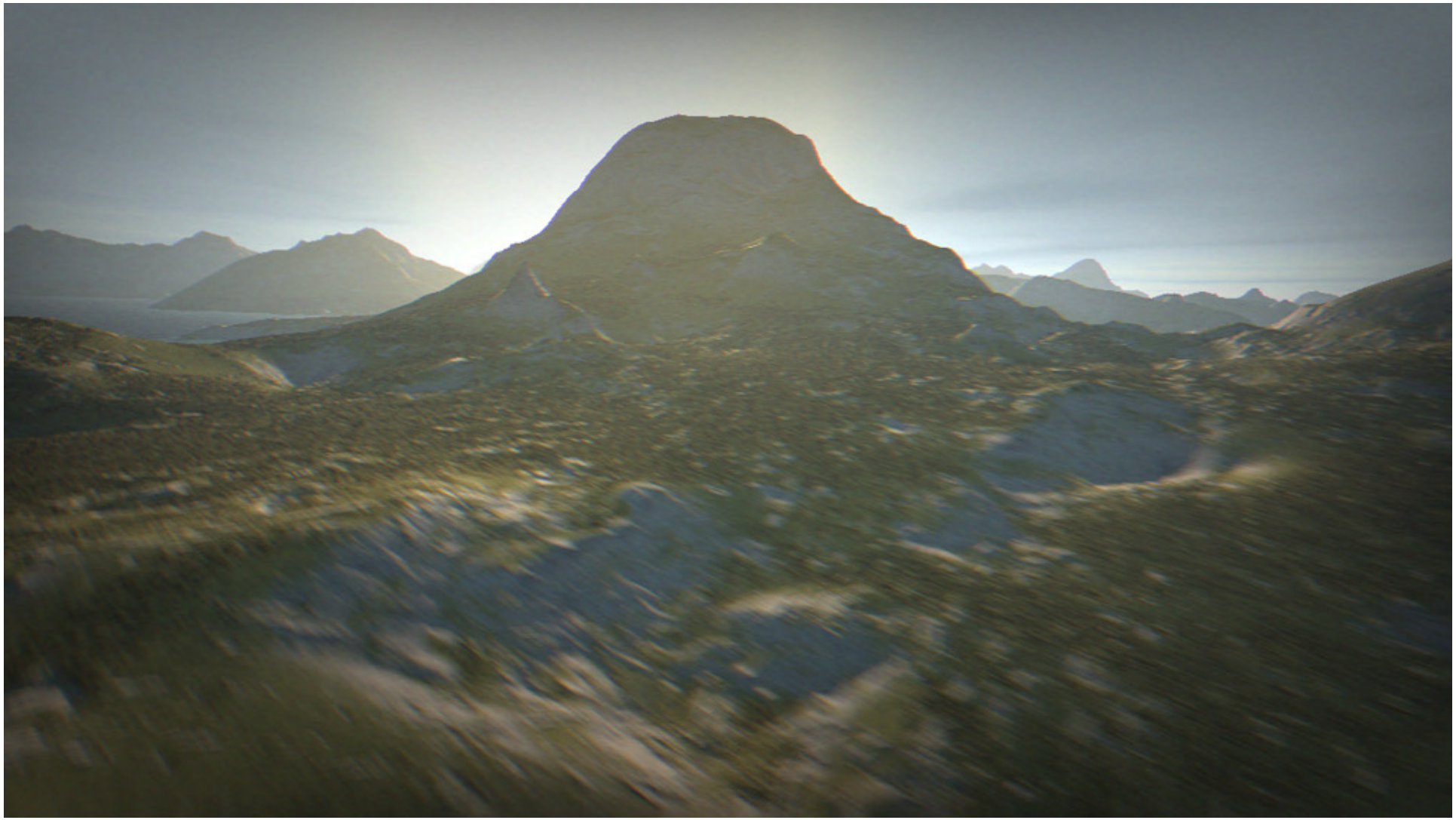


the techniques :: shading :: fake shadows

The idea is to ensure at least that the pixels which are behind or in the shadowed side of the mountain get dark.



the techniques :: shading :: fake shadows

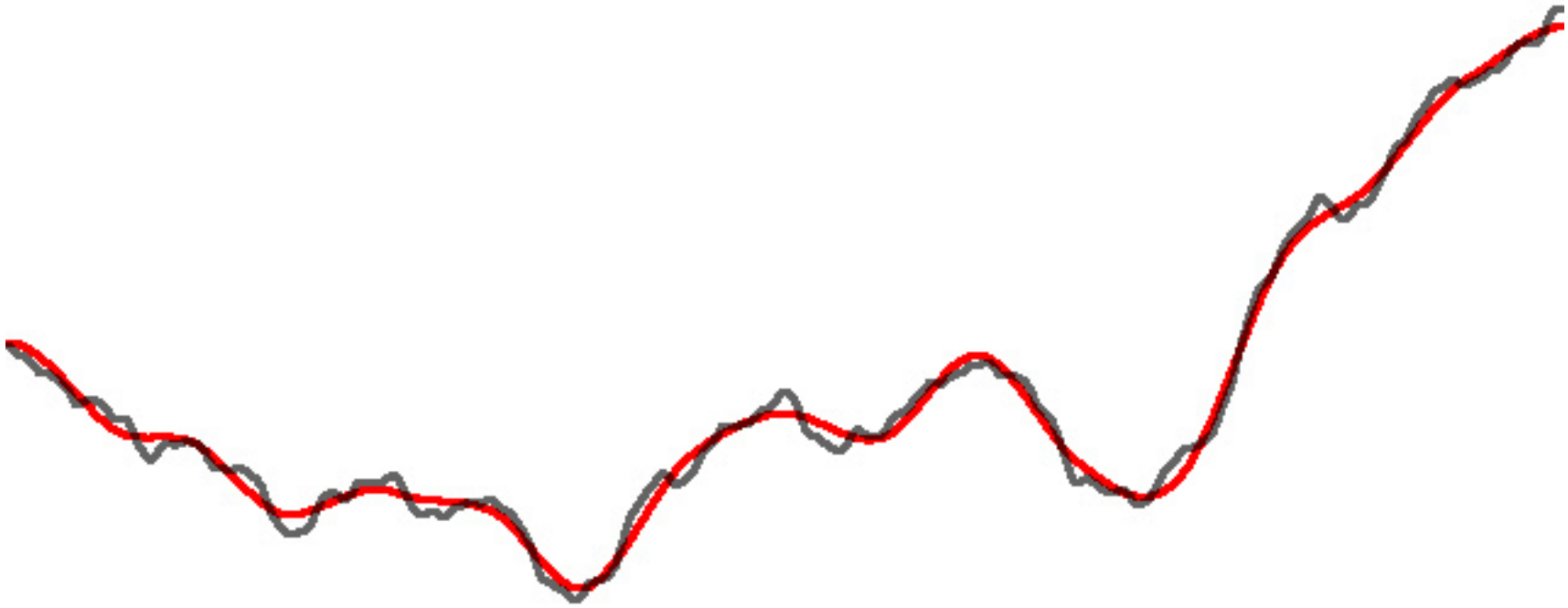


the techniques :: shading :: fake shadows

- Gray function is the terrain at full detail (say, 16 octaves)

the techniques :: shading :: fake shadows

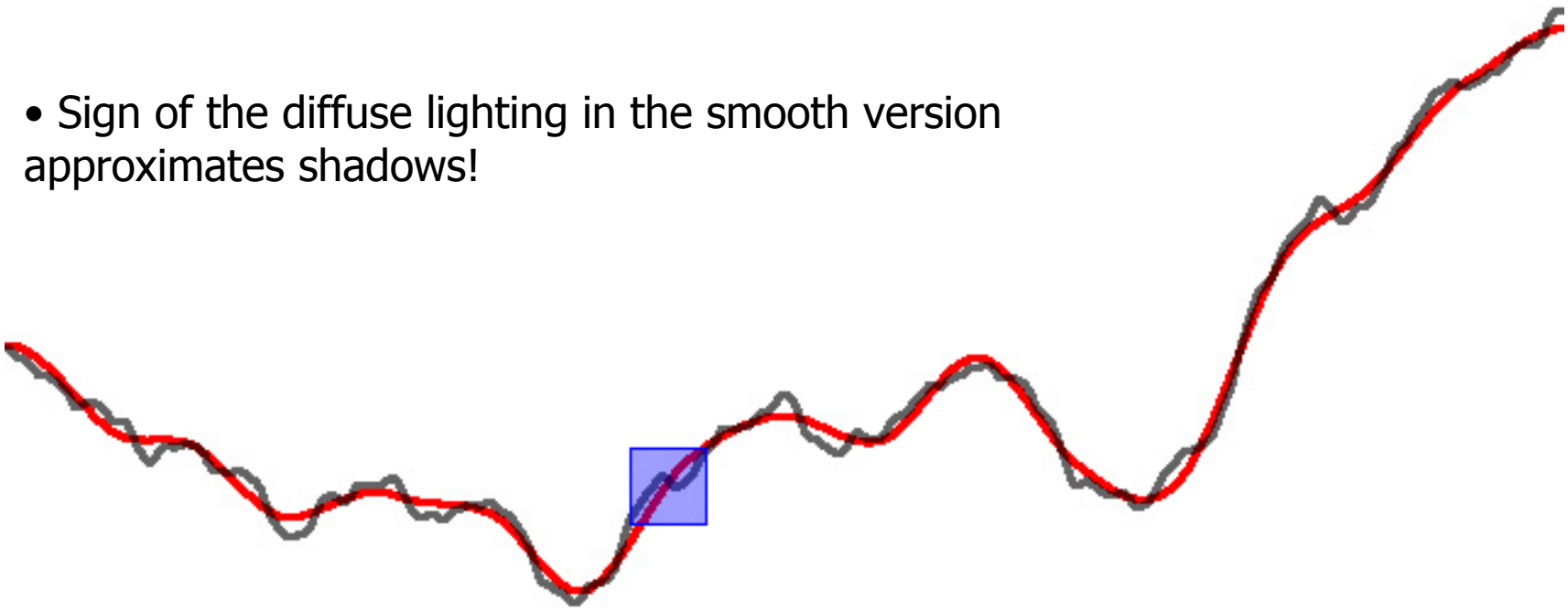
- Gray function is the terrain at full detail (say, 16 octaves)
- Red is same terrain at lower detail (say, 5 octaves)



the techniques :: shading :: fake shadows

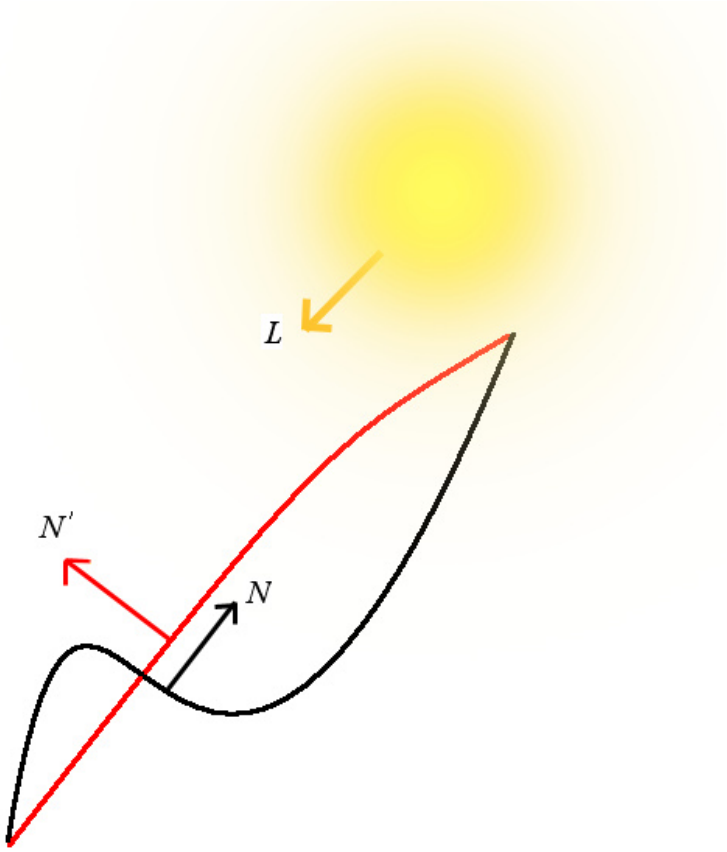
- Gray function is the terrain at full detail (say, 16 octaves)
- Red is same terrain at lower detail (say, 5 octaves)

- Sign of the diffuse lighting in the smooth version approximates shadows!



the techniques :: shading :: fake shadows

- Fake and fast soft shadows based on smoothed normal.



- N is the normal
- N' is the "smooth normal"
- Simple to combine with regular lighting:
  - Regular diffuse is $k_d = \langle N, L \rangle_+$
  - Modified is $k_d = \langle N, L \rangle_+ \cdot saturate\left(h \cdot \langle N', L \rangle\right)$
  - h controls the softness of the shadows

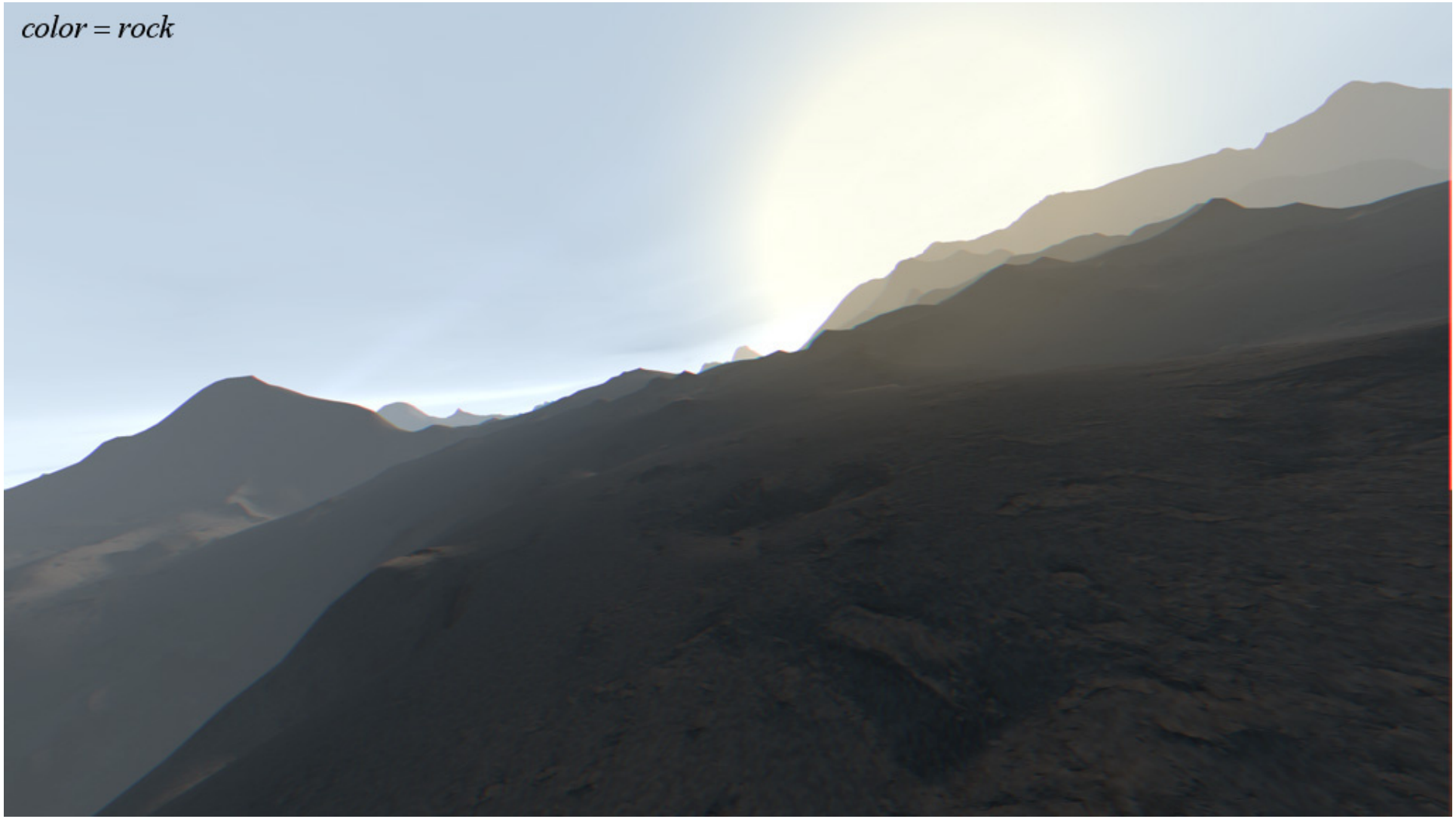
the techniques :: texturing

- How to combine/lerp layers of materials

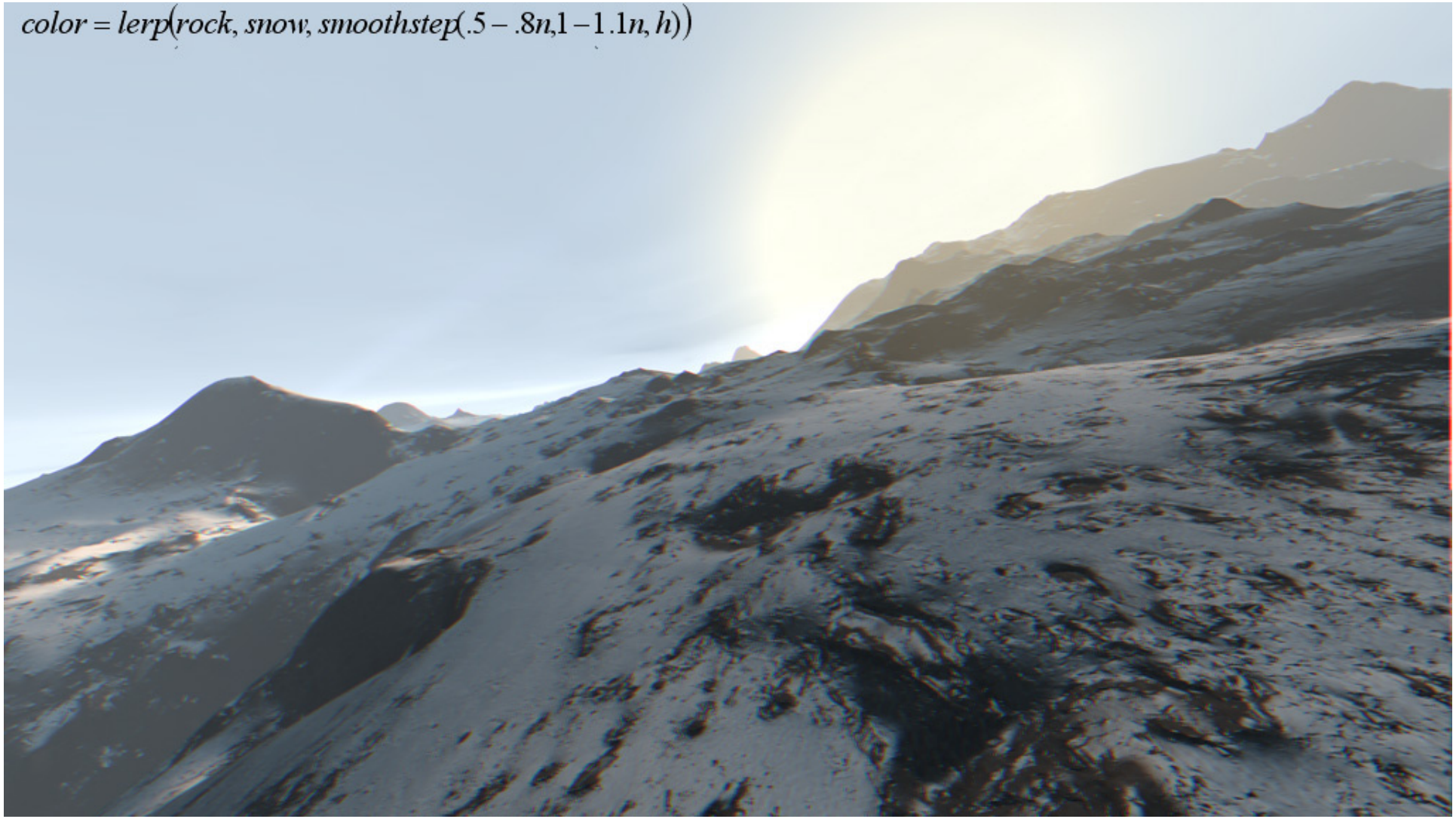
$$color = lerp(color1, color2, smoothstep(a, b, n_y))$$
$$color = lerp(color1, color2, smoothstep(a + h, b + h, c \cdot n_y))$$
$$color = lerp(color1, color2, smoothstep(a - c \cdot n_y, b - c \cdot n_y, h))$$
$$color = lerp(color1, color2, smoothstep(a - c \cdot n_y, b - d \cdot n_y, h))$$

- First one is the standard way
- In second one, *h* is a some noise-based function that breaks regularity and improves an natural look. In Elevated we used the terrain function again for *h*
- Third equation is mathematically equivalent to the second (up to the sign of *h*)
- Last equation adds even more control to the transition bands.

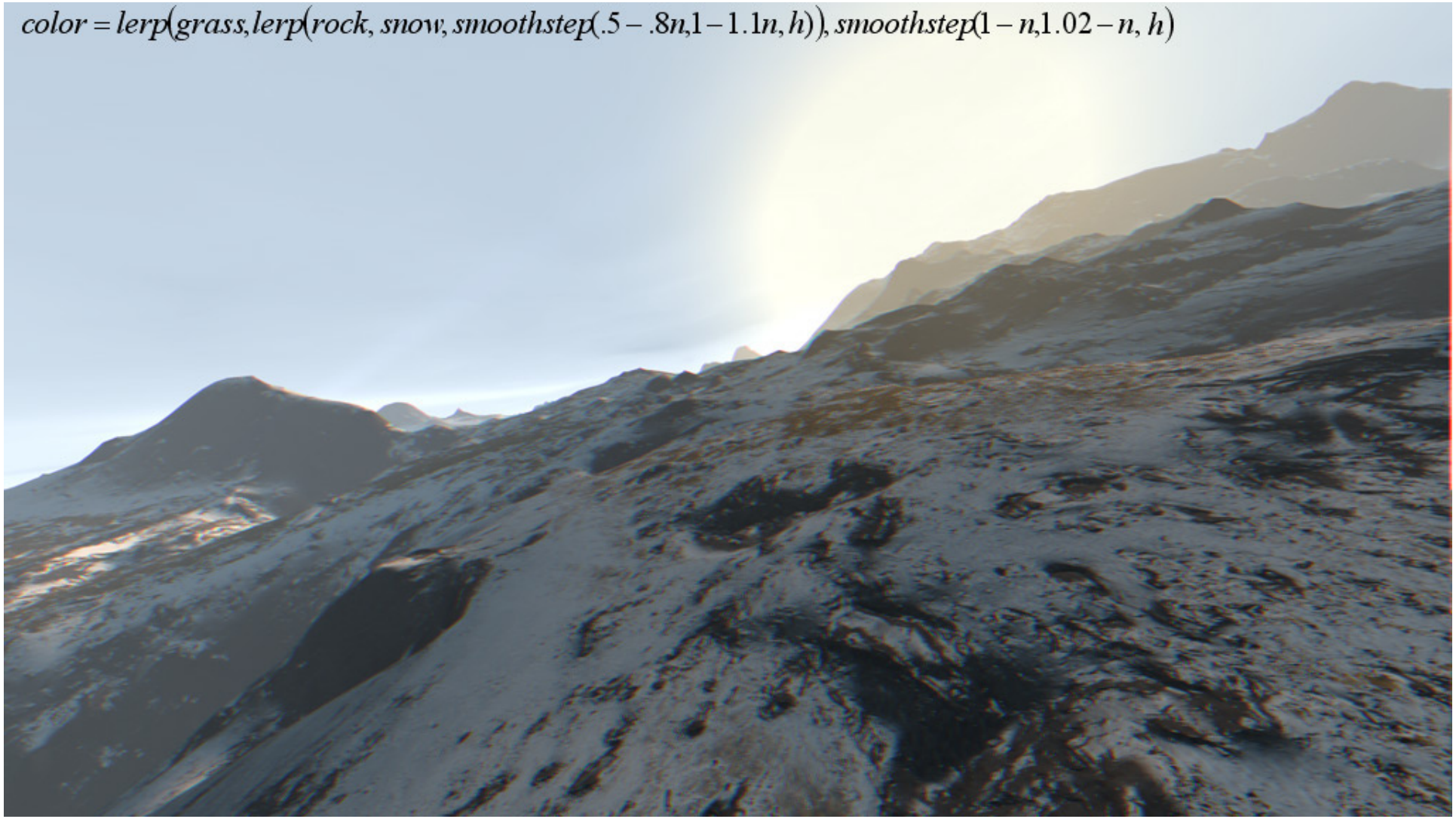
the techniques :: texturing



the techniques :: texturing

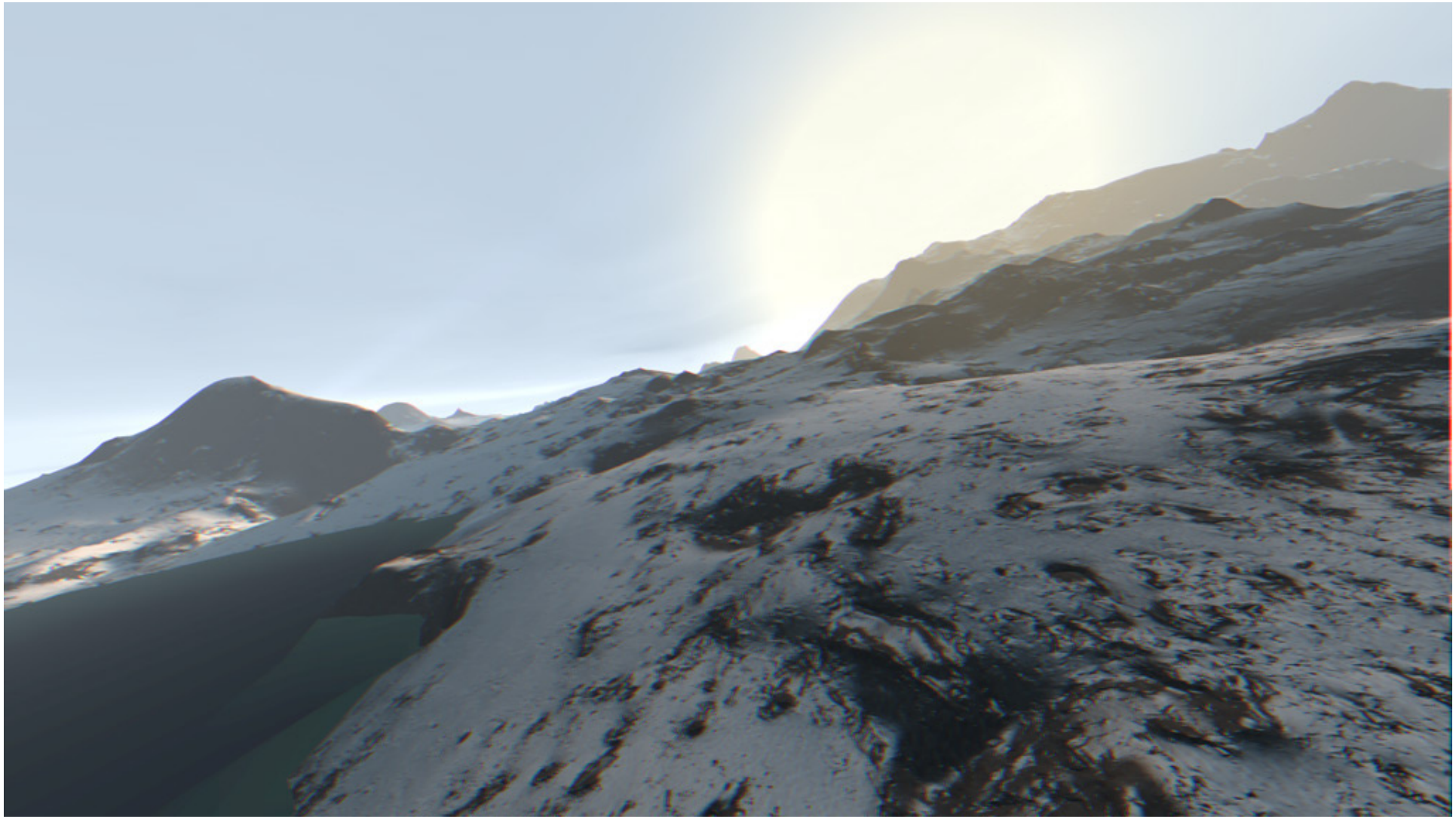


the techniques :: texturing

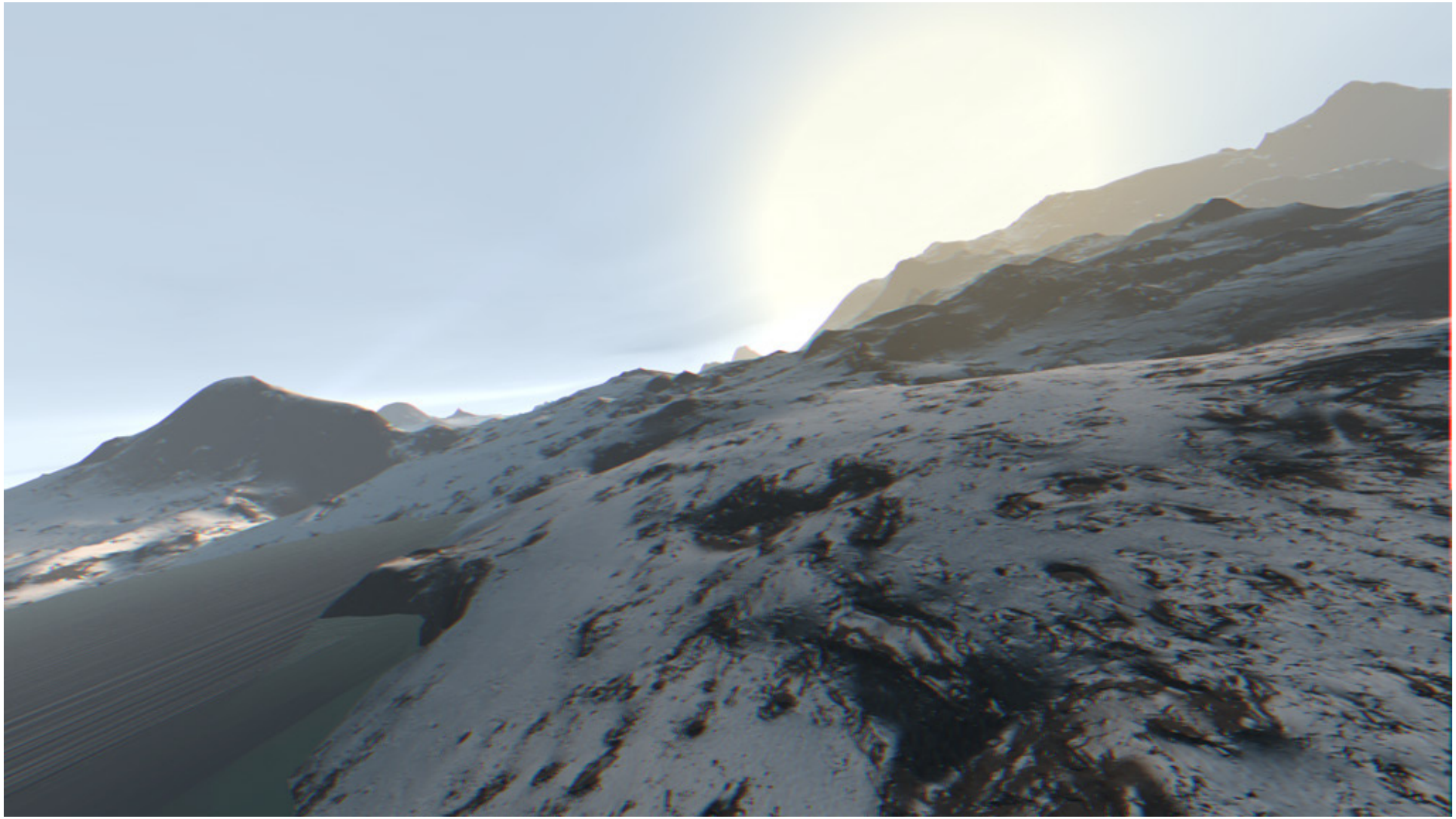


the techniques :: texturing

the techniques :: texturing

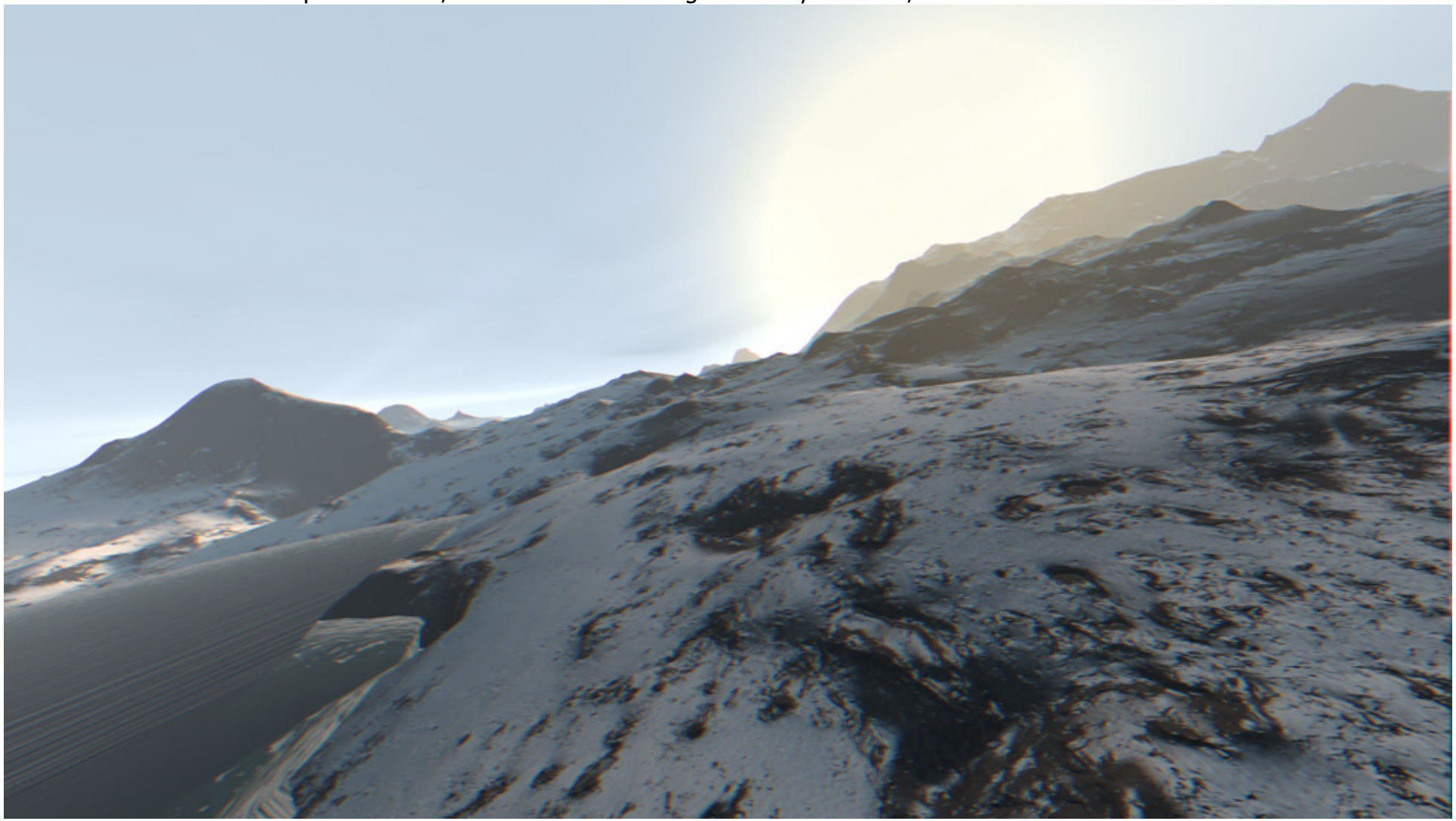


the techniques :: texturing

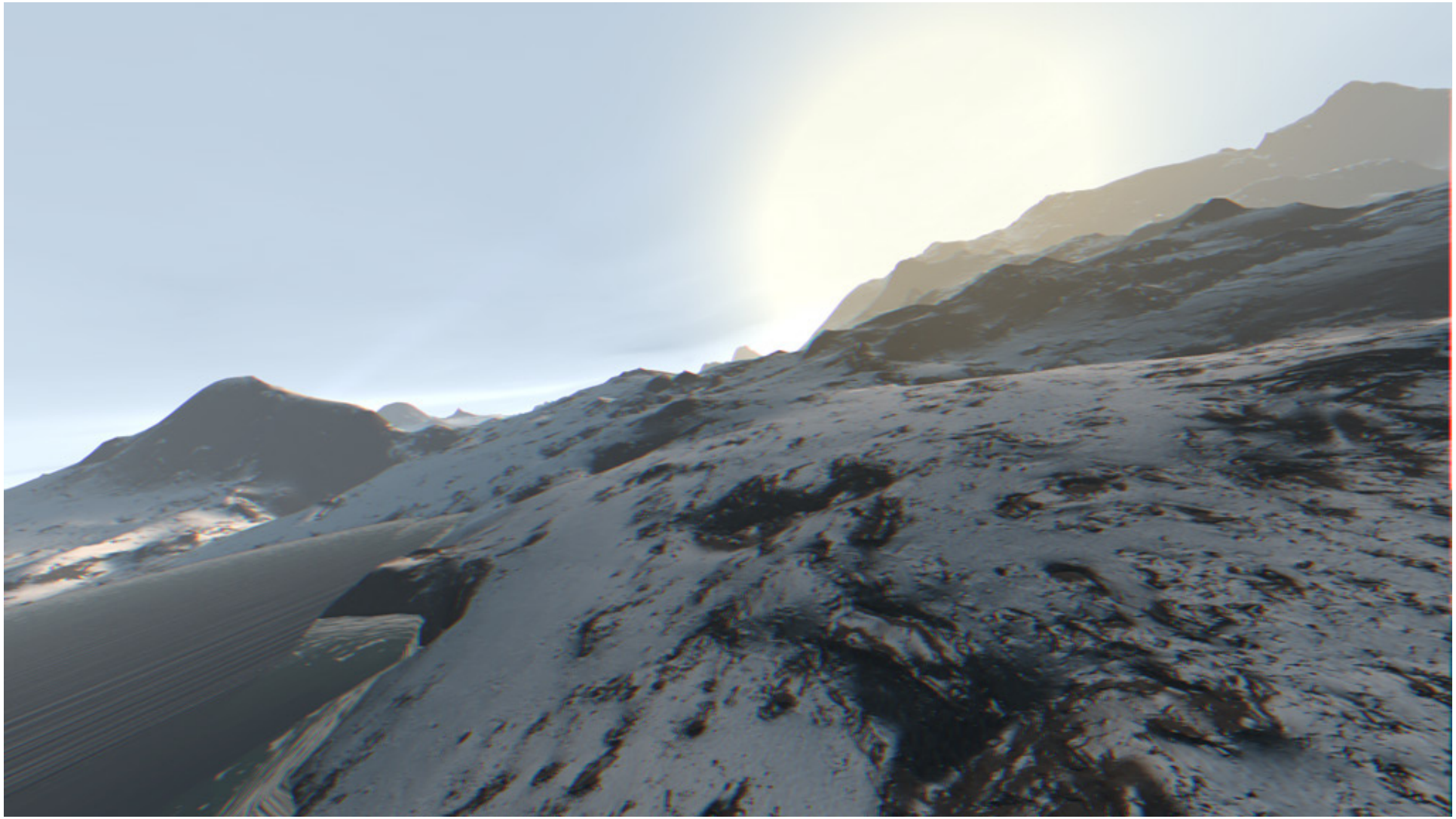


the techniques :: texturing

The idea is NOT to render perfect snow, but to draw something that *evoques* snow, and let the viewer's brain to trick the viewer.

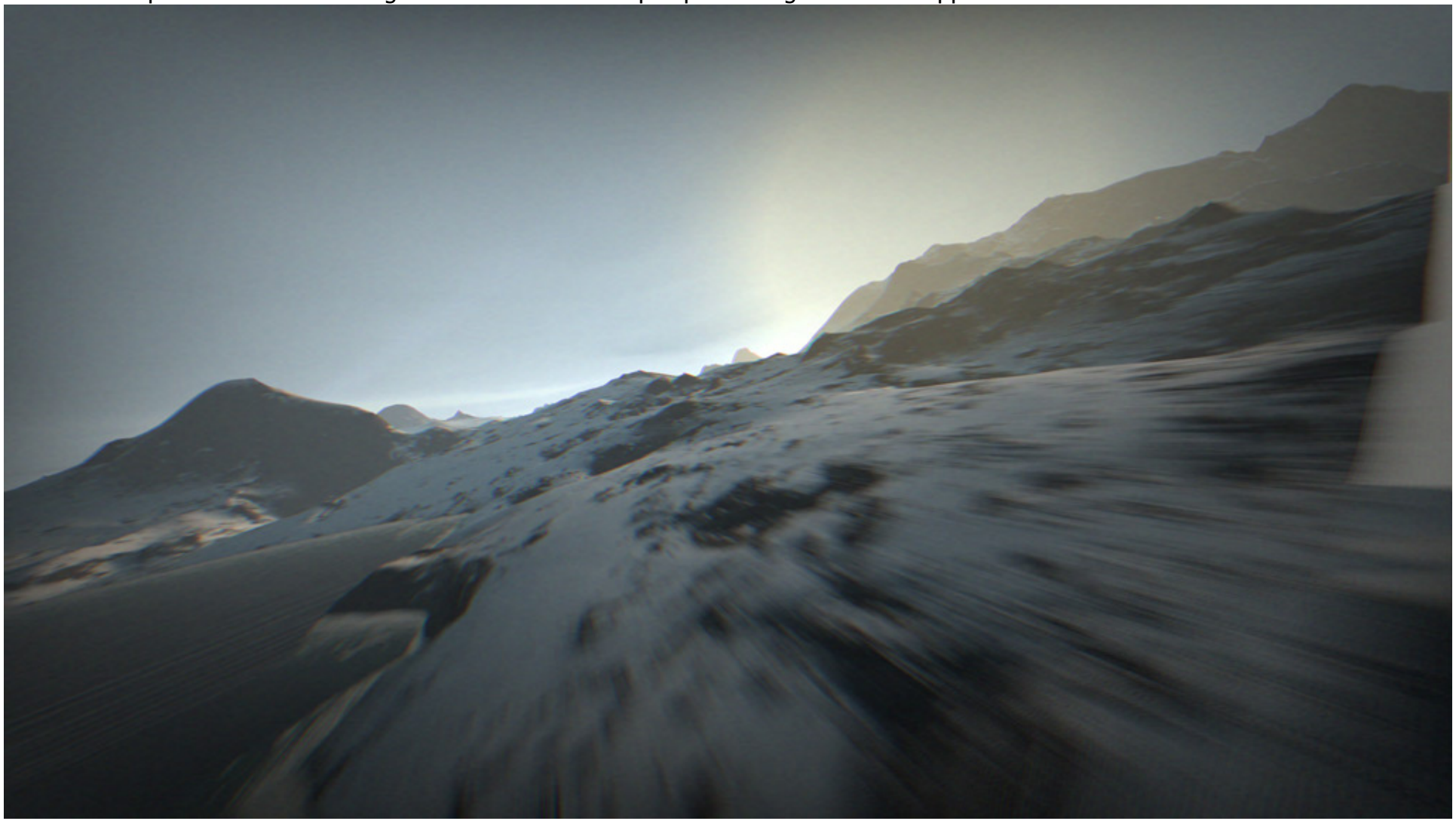


the techniques :: texturing



the techniques :: postprocessing

In the third pass the last two triangles are drawn and the postprocessing effects are applied.



.the.end.

- More info in http://iquilezles.org/www
- Thx to Gargaj for inviting me to give the seminar in this party
    - And all the orgas for making it possible
        - BTW, COME TO FUNCTION 2010