How to get “cinematic real-time”? Progress in bisplacement shader development

Artem Andreev // rybets@mail.ru

Russia, St Petersburg GUT, E&CG dept.

Revision: April 2 2007

Bisplacent? Why not displacement?

- There are several solutions to achieve cinematic quality using **displacement** maps on planes or non-planar surfaces in real - time…
- But do gamers really needs all 0…255 height variations?
- May be 0 or 1 height levels is enough for photorealism?

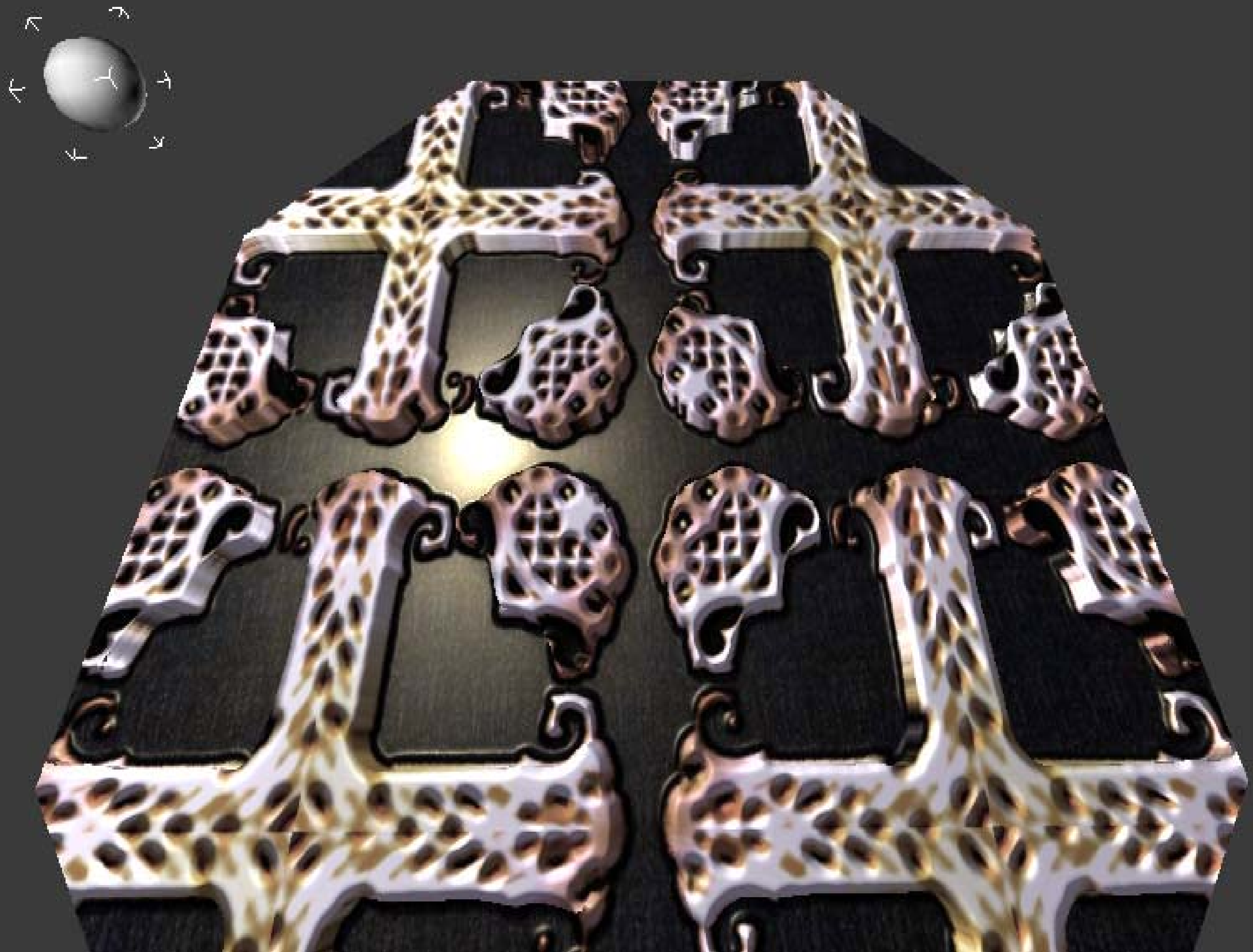
(“Bi…” – means 2. Besides, Google shows, that “bisplacement” appears in Chinese diesel ENGINES ☺).

- In 50% cases – ENOUGH, if it is used in a combination with a bump map and good shading formula.

Frame: 0
Perspective

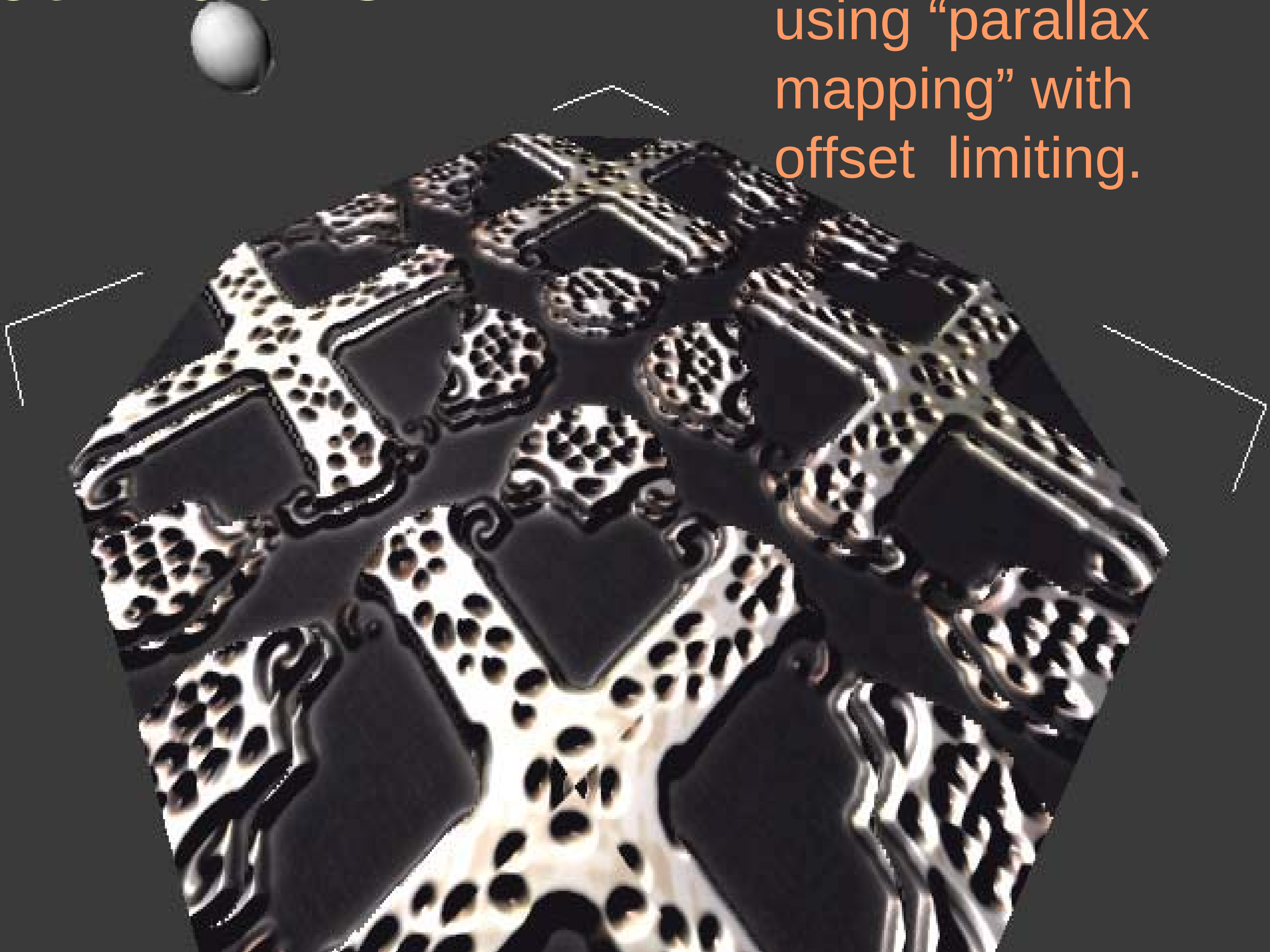
ALL Screenshot taken from NVIDIA FX Composer tm Scene Window

Something like this:



Not like this:

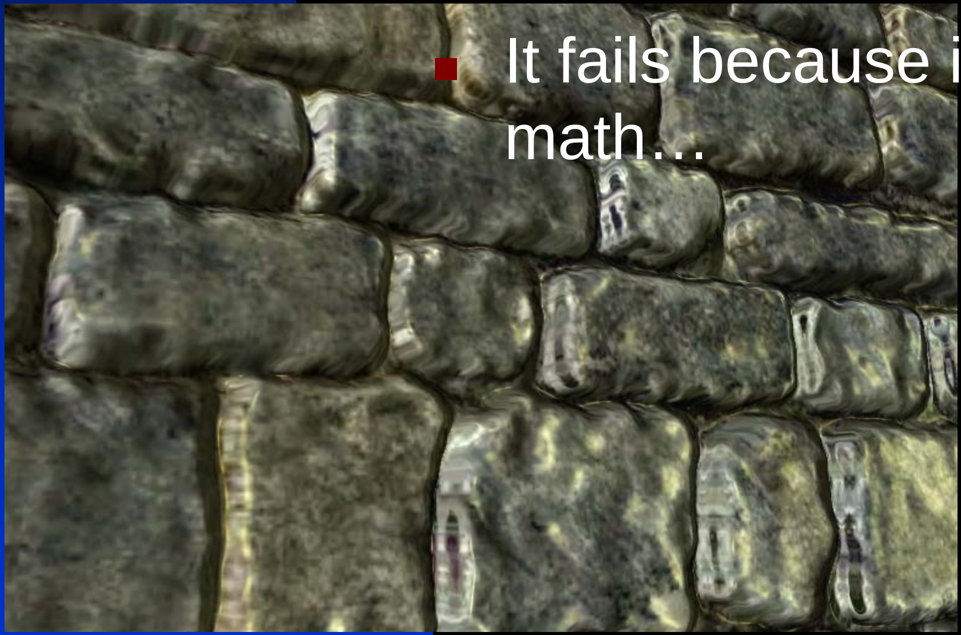
Image generated using “parallax mapping” with offset limiting.



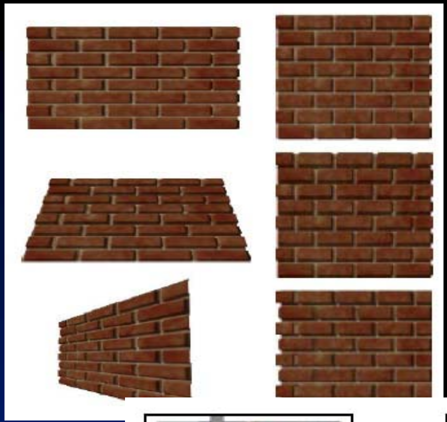
What wrong with parallax mapping?

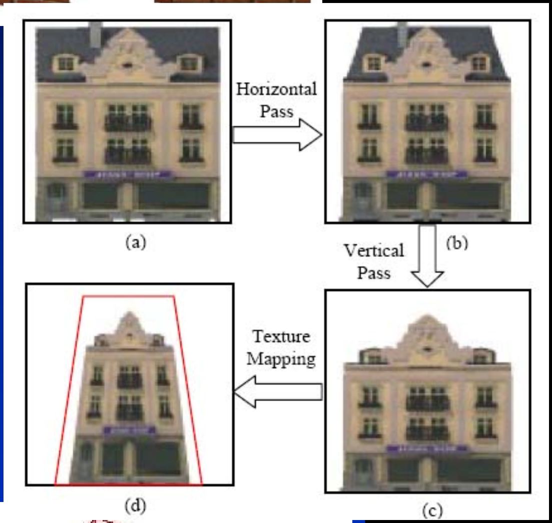
- Introduced by Kaneko [Kaneko 2001]
- Improved by Welsh [Welsh 2004]
- Broadly popularized by CG community.
- Use height map to determine texture coordinate offset for approximating parallax
- Uses view vector in tangent space to determine how to offset the texels
- When displacement seen from grazing angles, texture pattern appears twice ☹

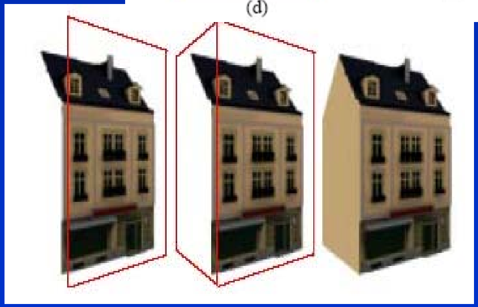
- May be this brick wall looks pretty, but parallax mapping shows that type of artifacts (texture doubling) on a grazing angles, anyway…
- It is very fast because of its simple math…
- It fails because it uses too simple math…



Seems, this problem was solved long ago…







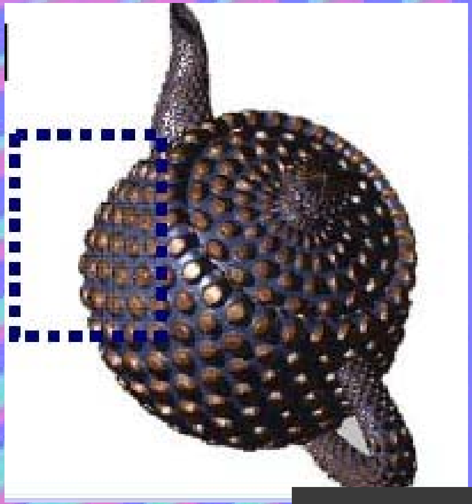
- Use simple textured hulls (parallelepiped is good for brick walls of buildings), update their appearance each time camera translates noticeably. This stage – “warping” does not use rotation, but only horizontal and vertical shifts, and could be done in software [Oliveira 2000].

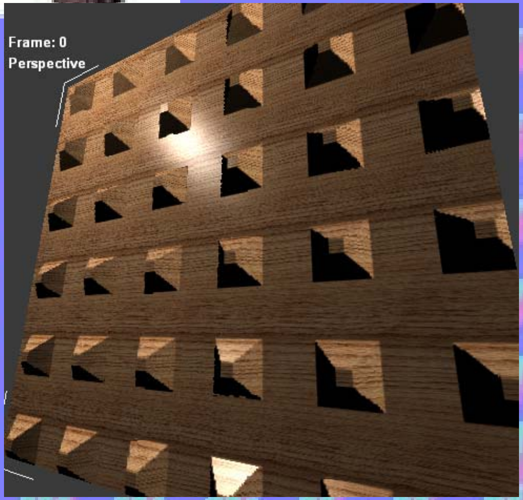
- “Relief texture mapping” seems very attractive but needs additional resources to keep data (view independent and dependent).

- May be not 60 times per second, but sometimes we need to update everything, even if object is not visible!

- Optimization trick, it was an incarnation of MS Talisman Architecture, [Kajiya 1996]: Instead of performing complex rendering for every frame, use view dependent textured layers… in 1996, when graphic memory and bandwidth was small, it was a temptation…

Hypercube?...





- **Solutions, were 4 or 5 dimensional textures are used [Wang 2003]. Approach could not compete in a 3d gaming industry: too much memory required for a small pattern.**

- Sources like [Lobel 2004] or [Policarpo 2004] represent evolution of subject. But in a “heavy weight” there are 2 competitors:

- Competitor #1: “Dynamic image-space per-texel displacement mapping with silhouette antialiasing via parallax occlusion mapping”? [Brawley - Tatarchuk 2005 - 2006]

- Competitor #2: “Per – pixel displacement mapping with distance functions” [Donnelly 2005] or “Steep Parallax Mapping” [McGuire 2005]?

As a research problem – No doubt. But!

- [Brawley - Tatarchuk 2005] First competitor requires 8 samples (texture access) in 64 instructions.
- It was 200, now it is 96 instructions, if it works on SM 3 model…
- At higher frequencies (detailed height field) needed 2-3 passes.
- That is all Technological Demos Stuff!!!

Technological demos shaders are not “in game” shaders!

- There is no explicit definition what is T.D., but practically it is created to demonstrate 100% of computational horsepower of last generation of 3d hardware.
- NOT a D3D or OpenGL standard (new features not standard yet), T.D. use to cut corners accessing hardware via special drivers tricks (good example – R2VB functions).
- Runs on ATI or NVIDIA, “reverse engineering” is prohibited.
- Camera is not free -> artifacts are hidden…
- GAME use a lot of visual phenomena's to render. Techno – demos are often “about something cool”. Cool stuff “eats” all horsepower.
- Games must SHARE performance between visual phenomena's!
- Hence – there are no “skin”, ”displacement”, modern diffuse lightning approximations, scattering… Game development focus on easy tricks, like PM, Blooms, HDR…

In a blue corner of a ring...

- Competitor #2 [Donnelly 2005]:
- Requires: “Spherical distance” data, and tools to compute them is not available yet (afaik). This volumetric texture 256*256*16 bytes or 512*512*32 bytes takes a lot: I could not imagine, that 128Mb video card could keep a game level with a massively usage of displacement, implemented that way. May be in 2008 it would be a mainstream…
- 16 iterations, old hardware require multiple passes.
- Fast, but how about game titles? I mean not a one pattern, but hundreds textures of a game level?

IMPATIENCE?

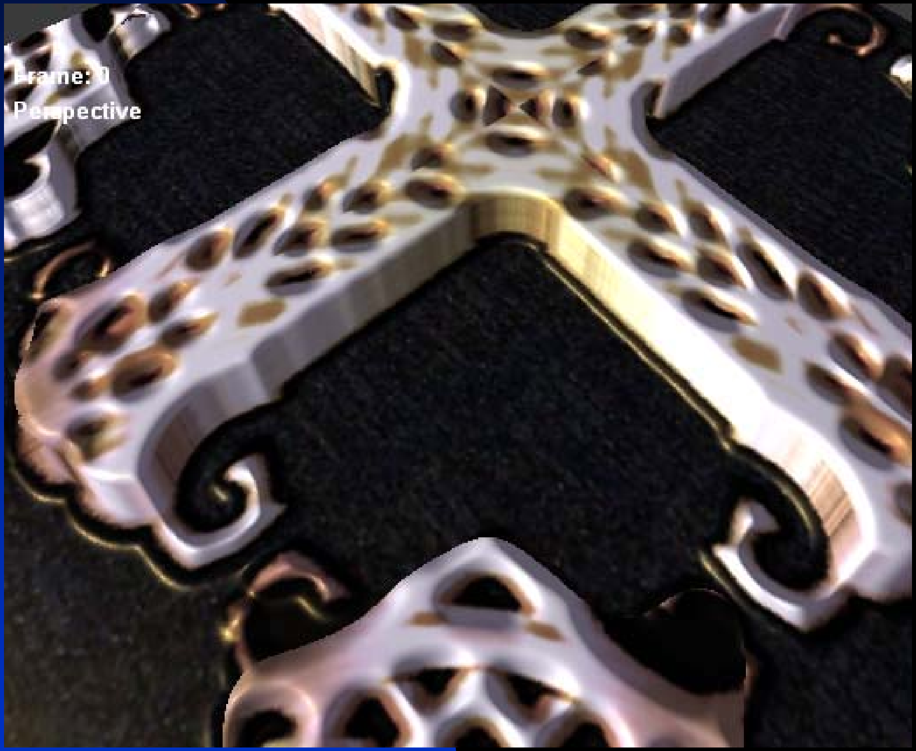
- Shader also make a SHADING, not just evaluation of texture coordinate and fetching…
- Fourth generation of graphic hardware – ATI 9x00, Nvidia 6x00 family must be supported (it is: via multi pass).

  Do you mean all that 4 pixel pipeline, 128 megabyte stuff?
- They are slow: do not support dynamic branching, simultaneous texture fetching and computation…
- Boss says: No objections! Game MUST FLY on them!
- This means – forget about displacement…

…and use Bisplacement!

- Easy to create texture, bump and height field.
- Single pass on ps_2_0, ps_2_b.
- 6 reads of texture in total (diffuse OR bump) if DFC is non-supported in GPU.
- If DFC supported it may be faster (but that is compilation / hardware dependent).
- Diffuse texture RGBA (A channel stores preprocessed binary height field)
- Bump map RGB, A channel is used for gloss map.
- 47 – 53 app. arithmetic slots used in a pixel shader.

Bisplacement:

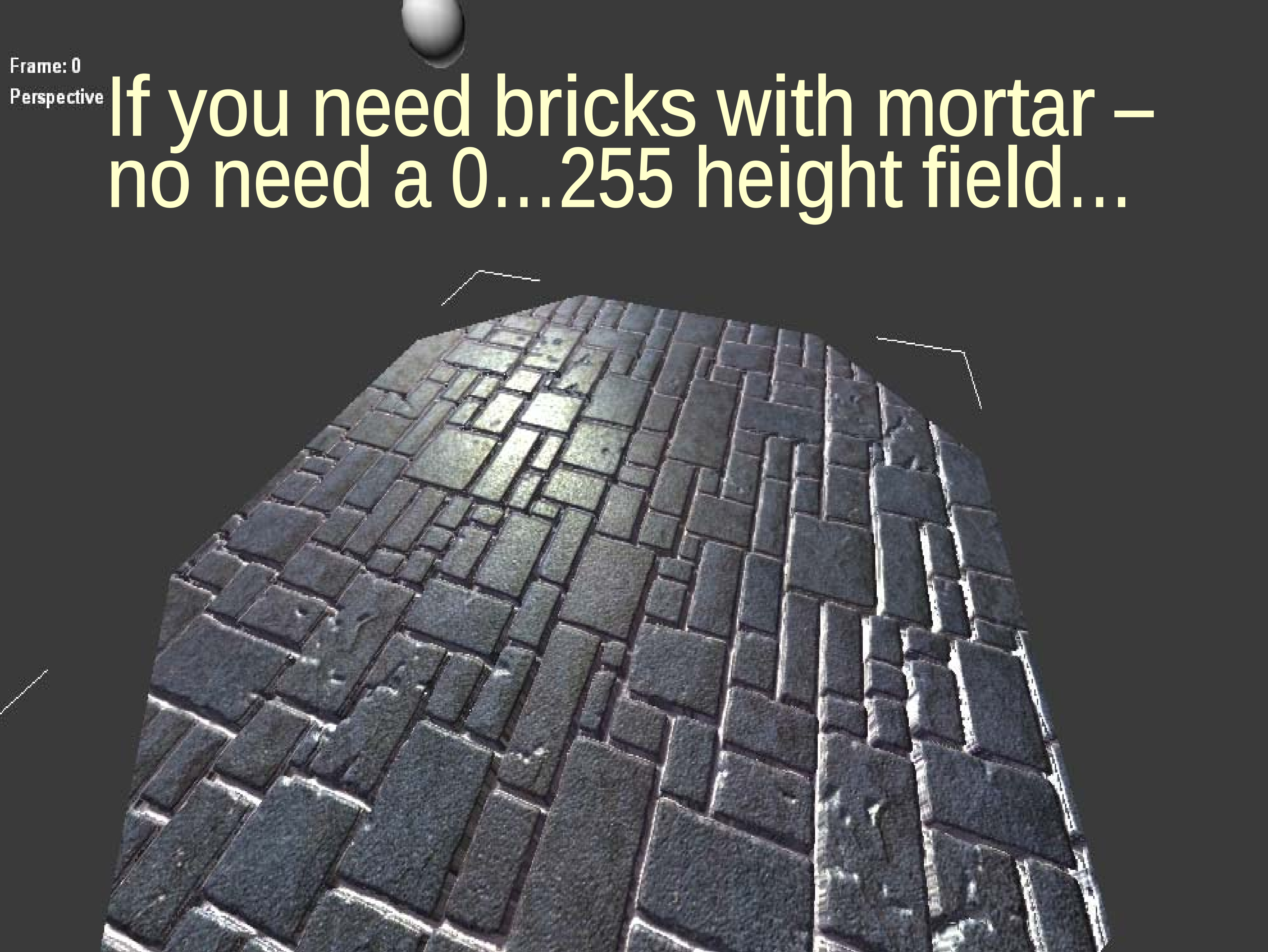


- No iterations
- It is not per-fragment “ray-tracing”
- Appearance is not a set of patterns, like classic “fur rendering” makes
- All pixels on a vertical “wall” appear smoothly in a normal direction
- Feature: sometimes small details are still plane (like fonts serif)…
- Not only 2 levels of height!

Combined with Advanced Shaders to achieve cinematic realism

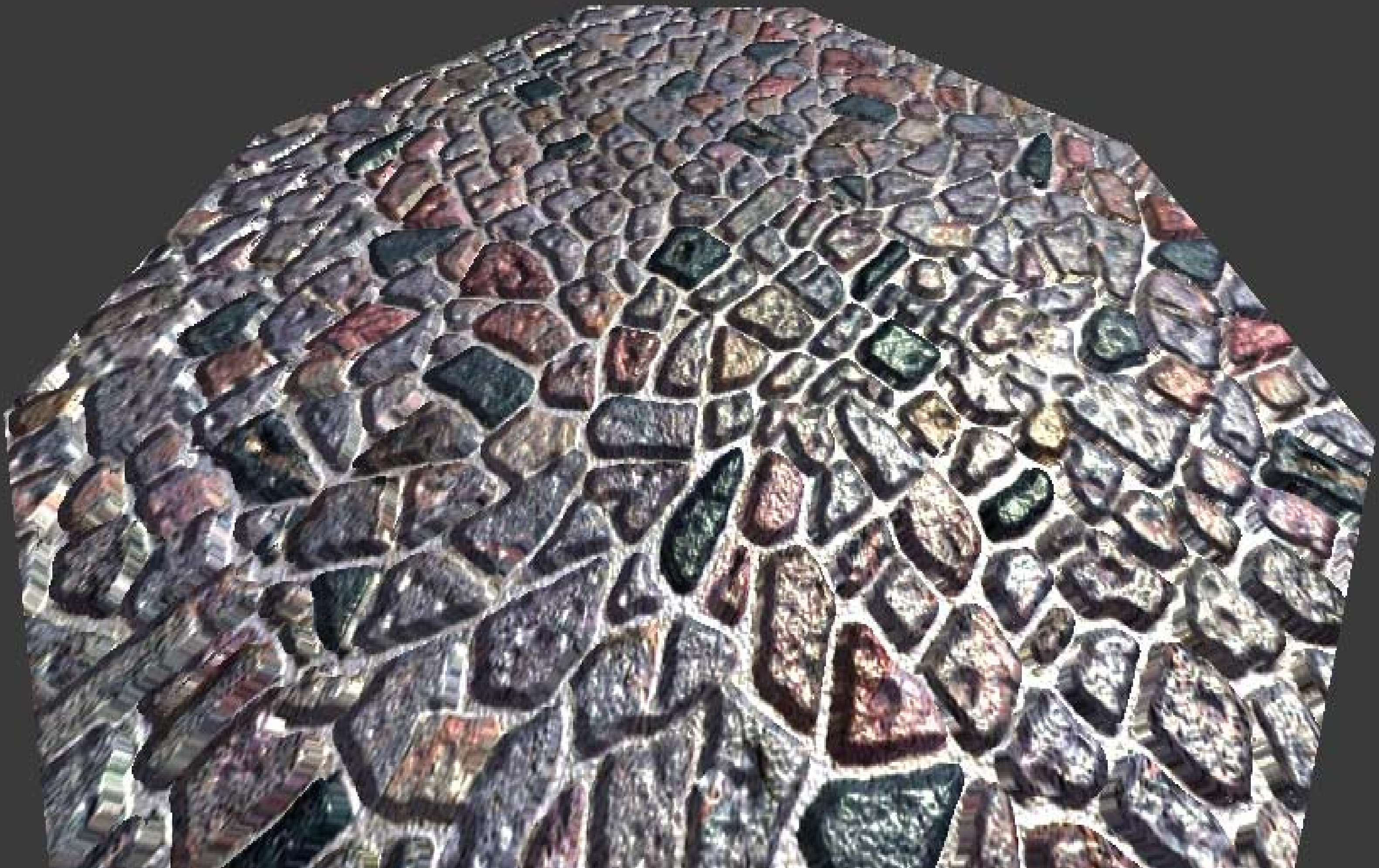
- Two levels of displacement is not enough to achieve cinematic realism… But displacement is not an only way that promise it.
- Bump-mapping is a relatively old method, [Blinn 1978], and since that time a lot of techniques were developed. Good overview of basic methods programming could be founded in [Fernando-Kilgard 2003], or in various articles from web.
- Combining Bump Mapping with specular and diffuse terms with bisplacement, a new level of realism could be achieved in a single pass shader…

If you need bricks with mortar – no need a 0…255 height field…



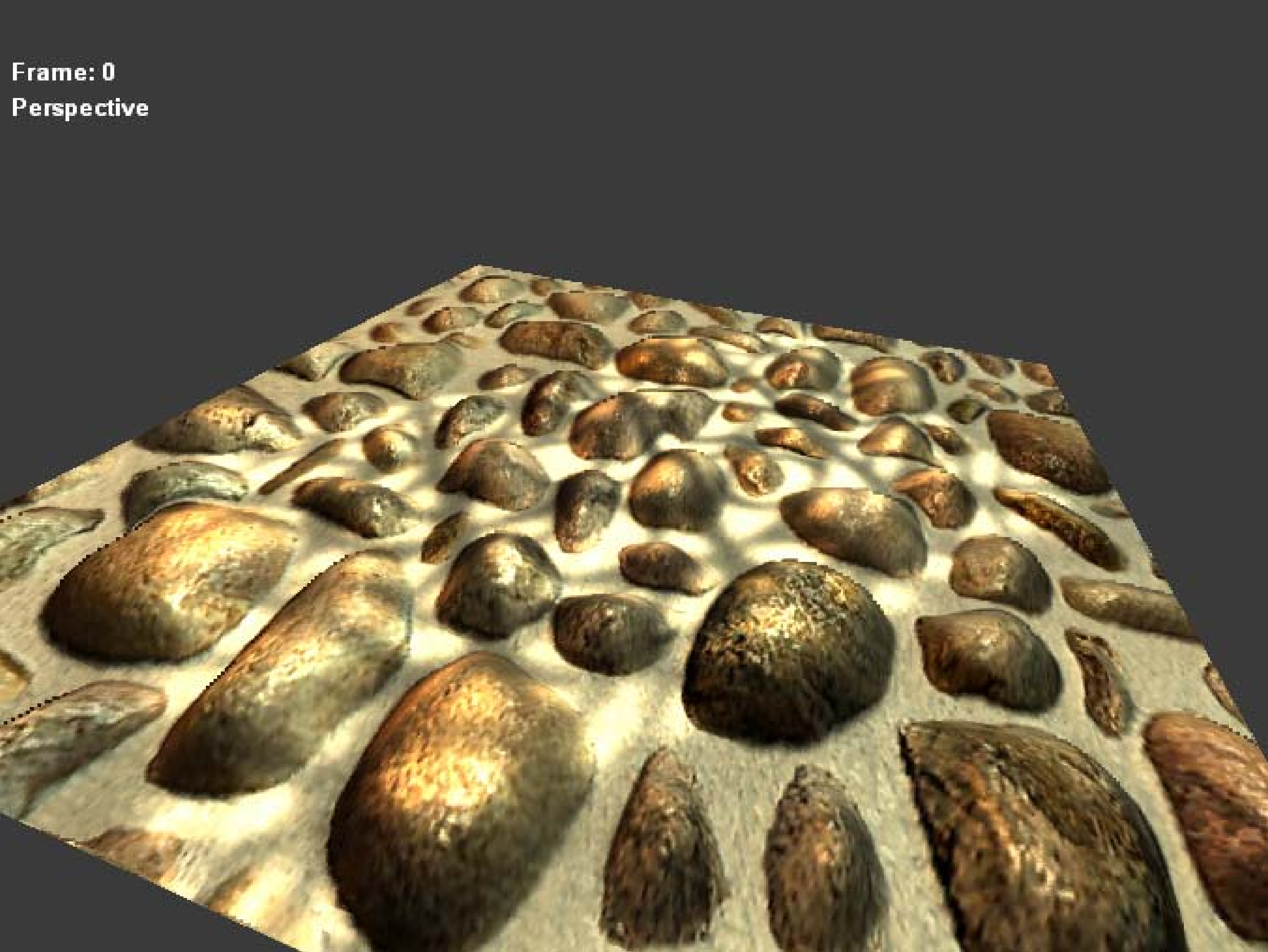
Frame: 0
Perspective

Take a look, and imagine, how many polygons was used!

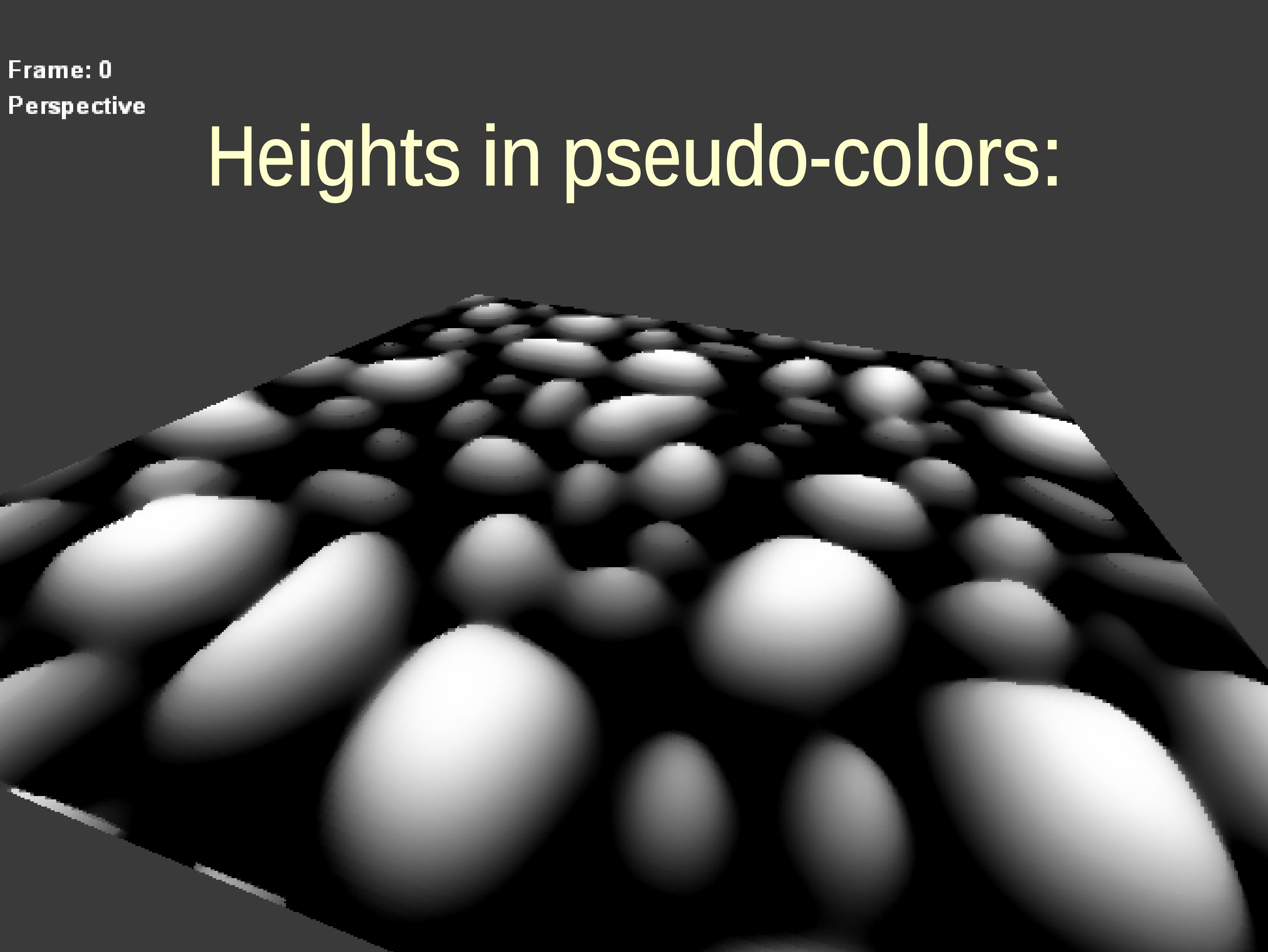


Do you still pray “LOD”?

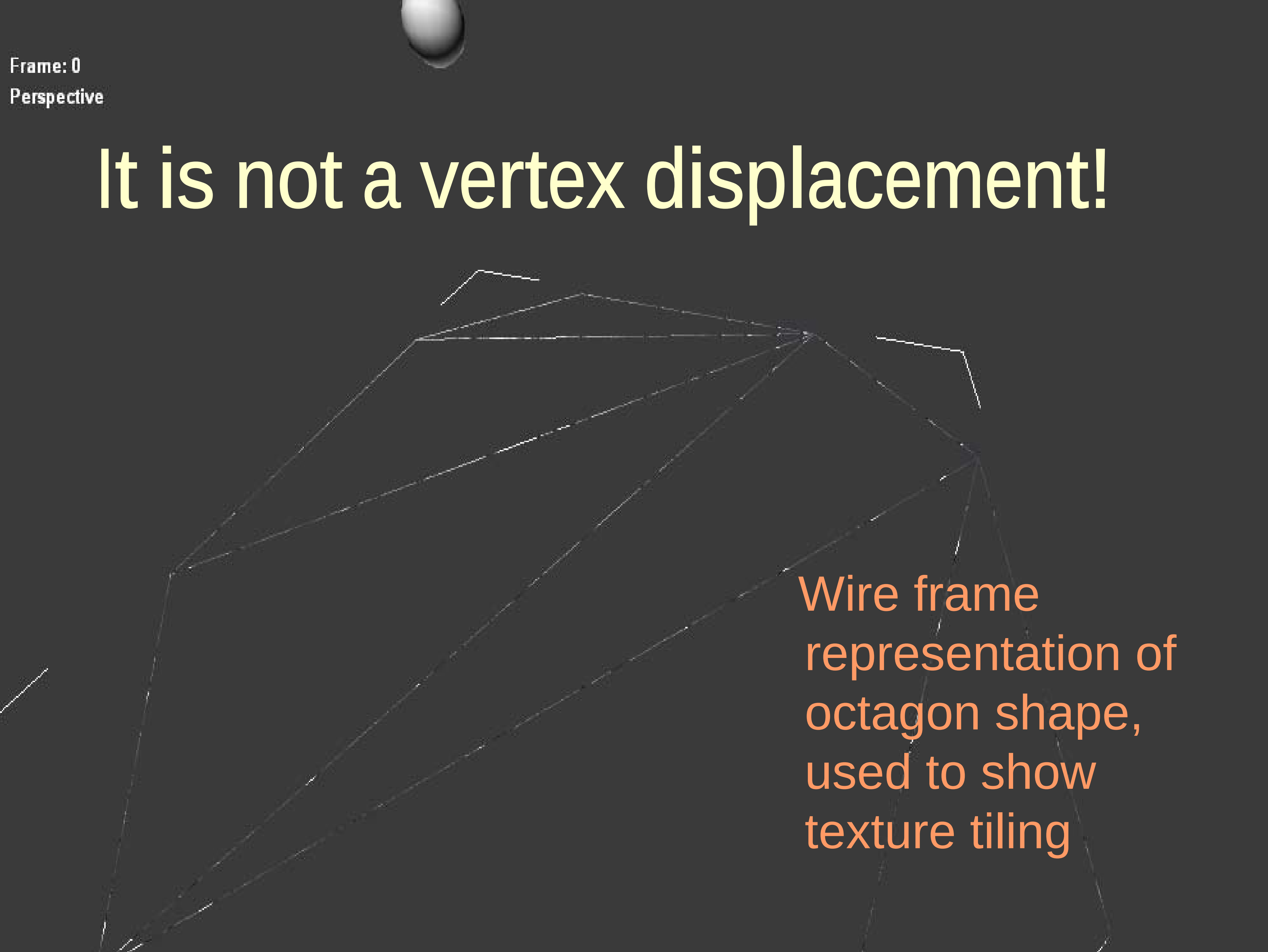
- It is not easy to convince CG people: everybody believe that vertex displacement is not for “current generation of games”.
- They say: “displacement is a good and old idea, but it was never used in games via software tessellation “on the fly” on curved surfaces…”
- They say: “In a real-time, we need a sophisticated geometry LOD processing”
- Empty pixels in quads must be pumped too…
- They say: “Rendering displacement map in hardware via vertex shaders was introduced in Matrox Parhelia as a part of DirectX9, and we still do not have any game title that supports it…”
- They say: “May be unified shader architecture could spend its power for intensive vertex T&L…”


Frame: 0
Perspective

Heights in pseudo-colors:

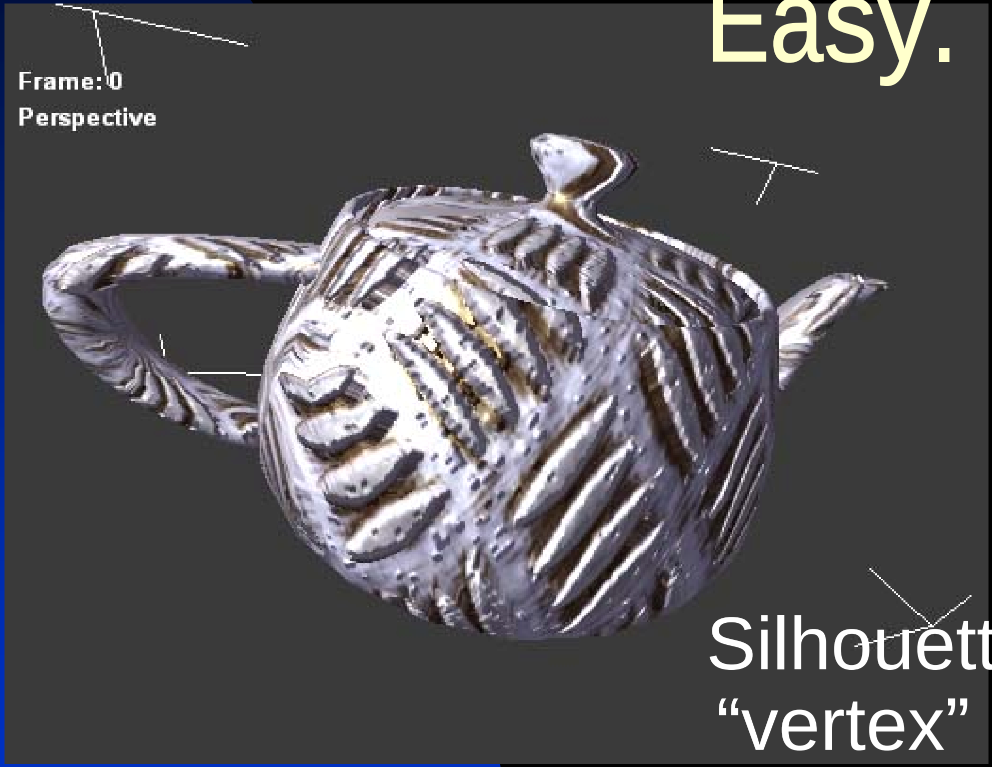


It is not a vertex displacement!



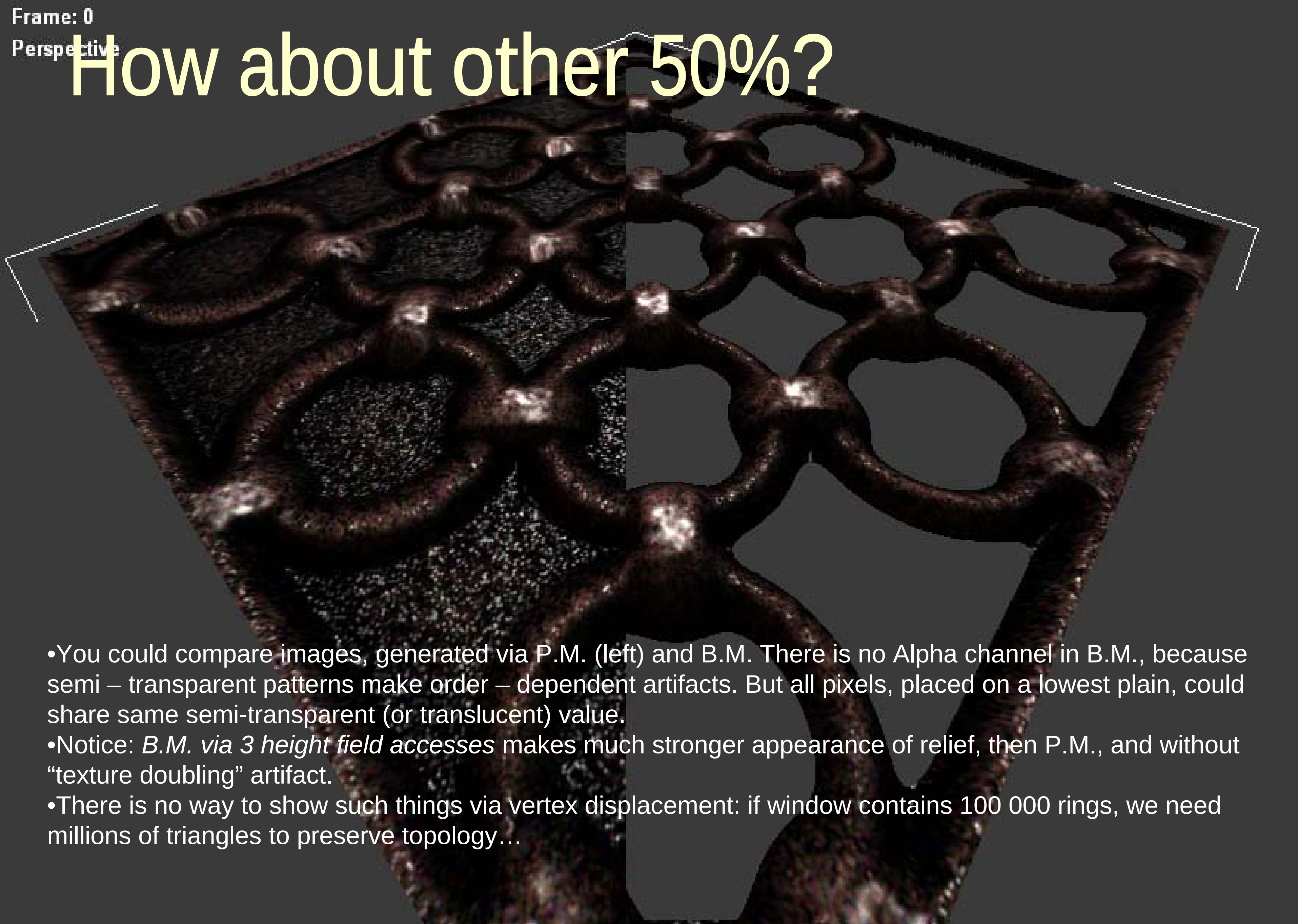
Wire frame representation of octagon shape, used to show texture tiling

Bisplacement on non-planar surface? Easy.



Silhouette is smooth, unlike "vertex" displacement techniques, but this is not a severe limitation of bisplacment

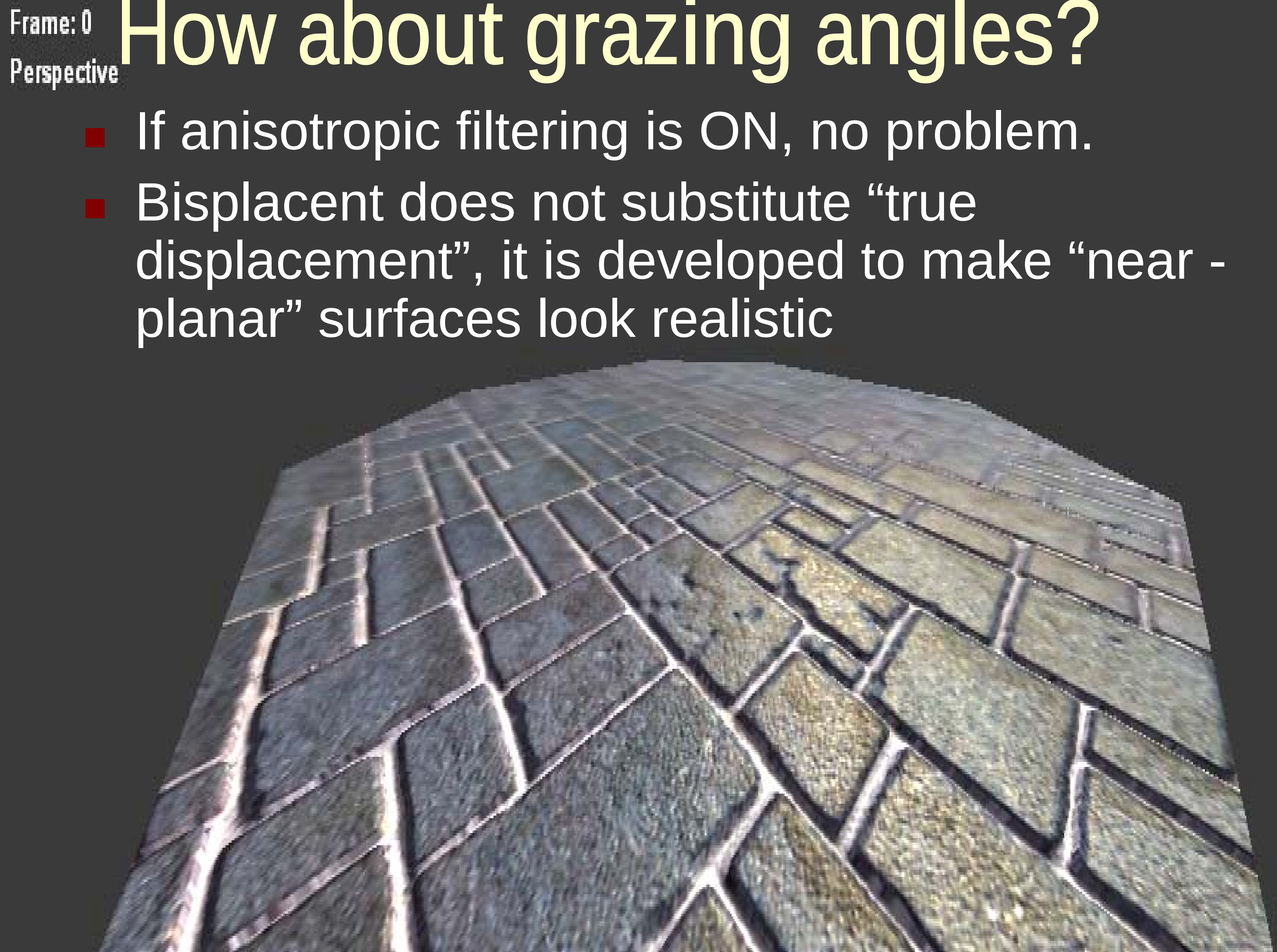
How about other 50%?



- You could compare images, generated via P.M. (left) and B.M. There is no Alpha channel in B.M., because semi – transparent patterns make order – dependent artifacts. But all pixels, placed on a lowest plain, could share same semi-transparent (or translucent) value.
- Notice: *B.M. via 3 height field accesses* makes much stronger appearance of relief, then P.M., and without “texture doubling” artifact.
- There is no way to show such things via vertex displacement: if window contains 100 000 rings, we need millions of triangles to preserve topology…

How about grazing angles?

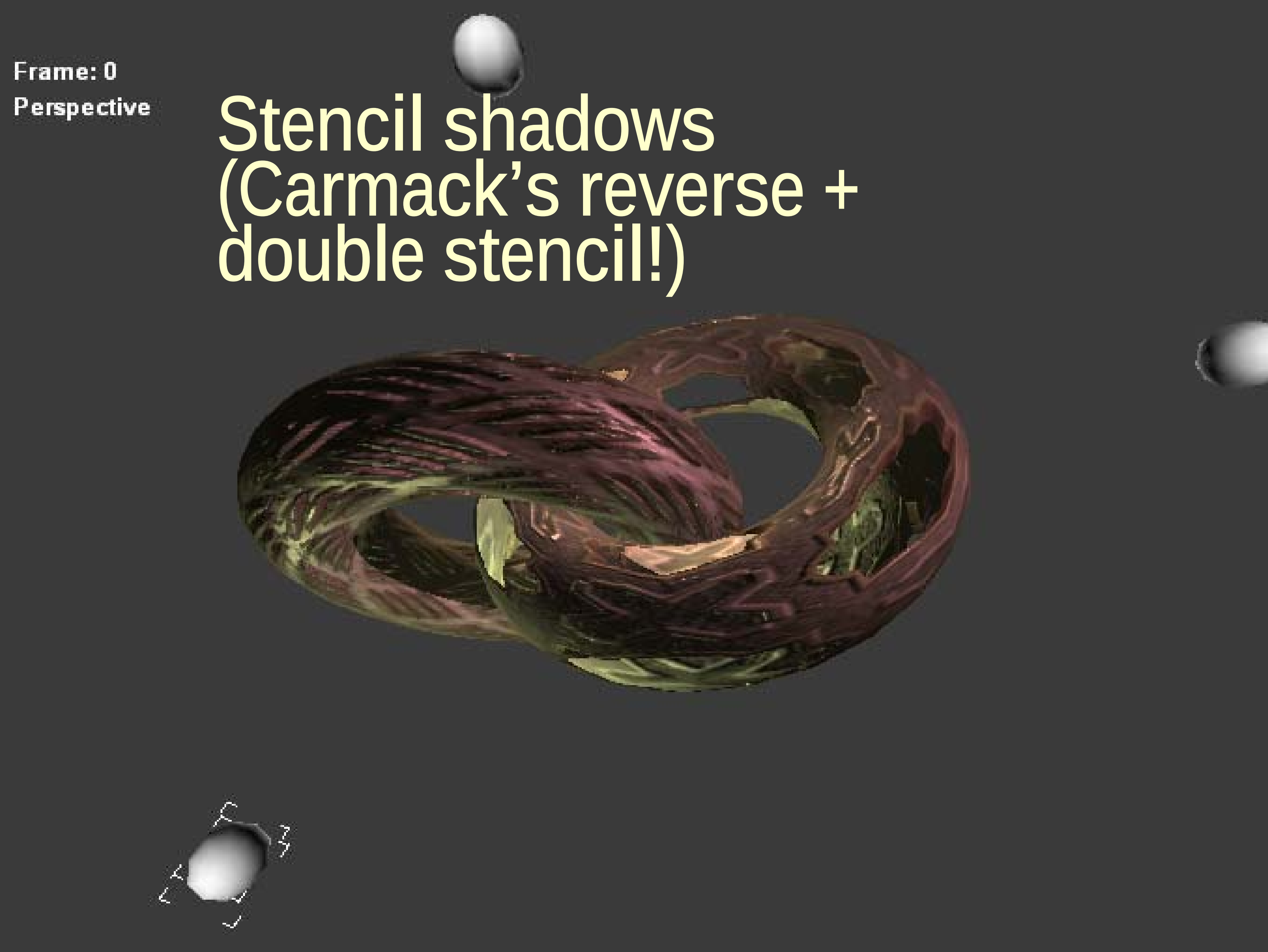
- If anisotropic filtering is ON, no problem.
- Bisplacent does not substitute “true displacement”, it is developed to make “near - planar” surfaces look realistic



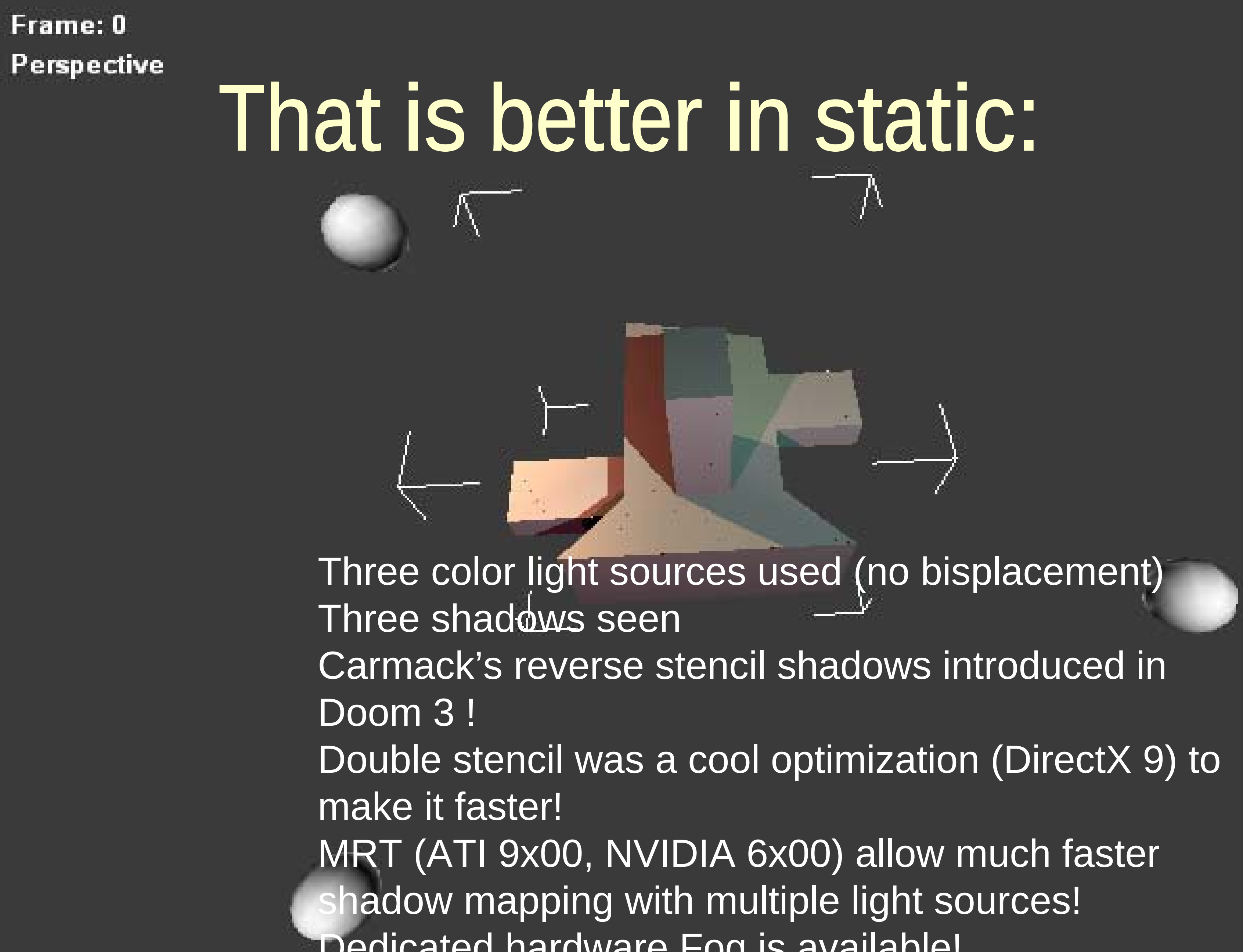
Frame: 0
Perspective

Three color light sources!
Diffuse and Specular with specular control map!
Fast Blinn shading (new implementation saves n instructions in case of n lights!)
Fog support!
Stencil shadows and fog Supported together!

“Rusted wood“



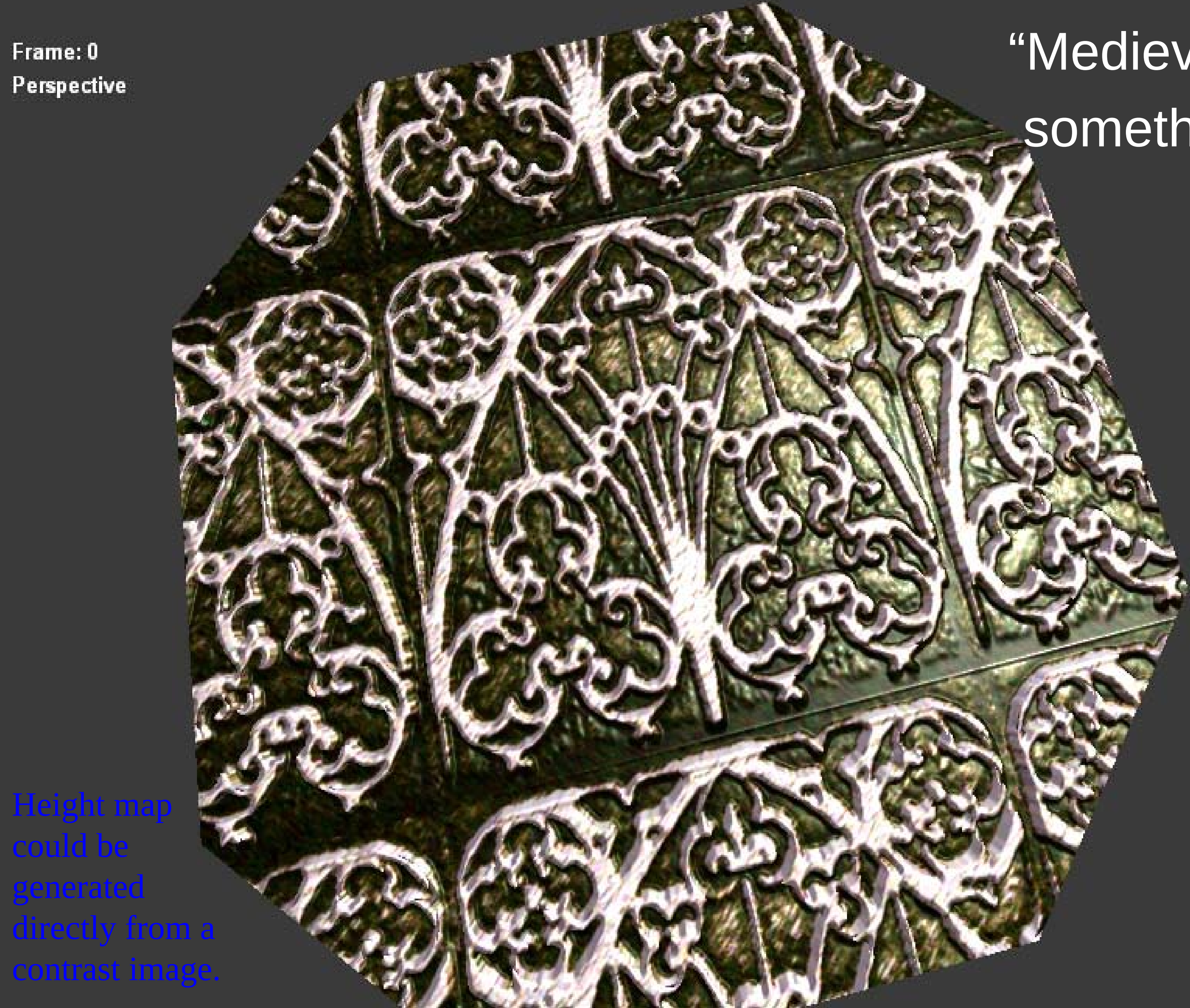
Stencil shadows (Carmack’s reverse + double stencil!)



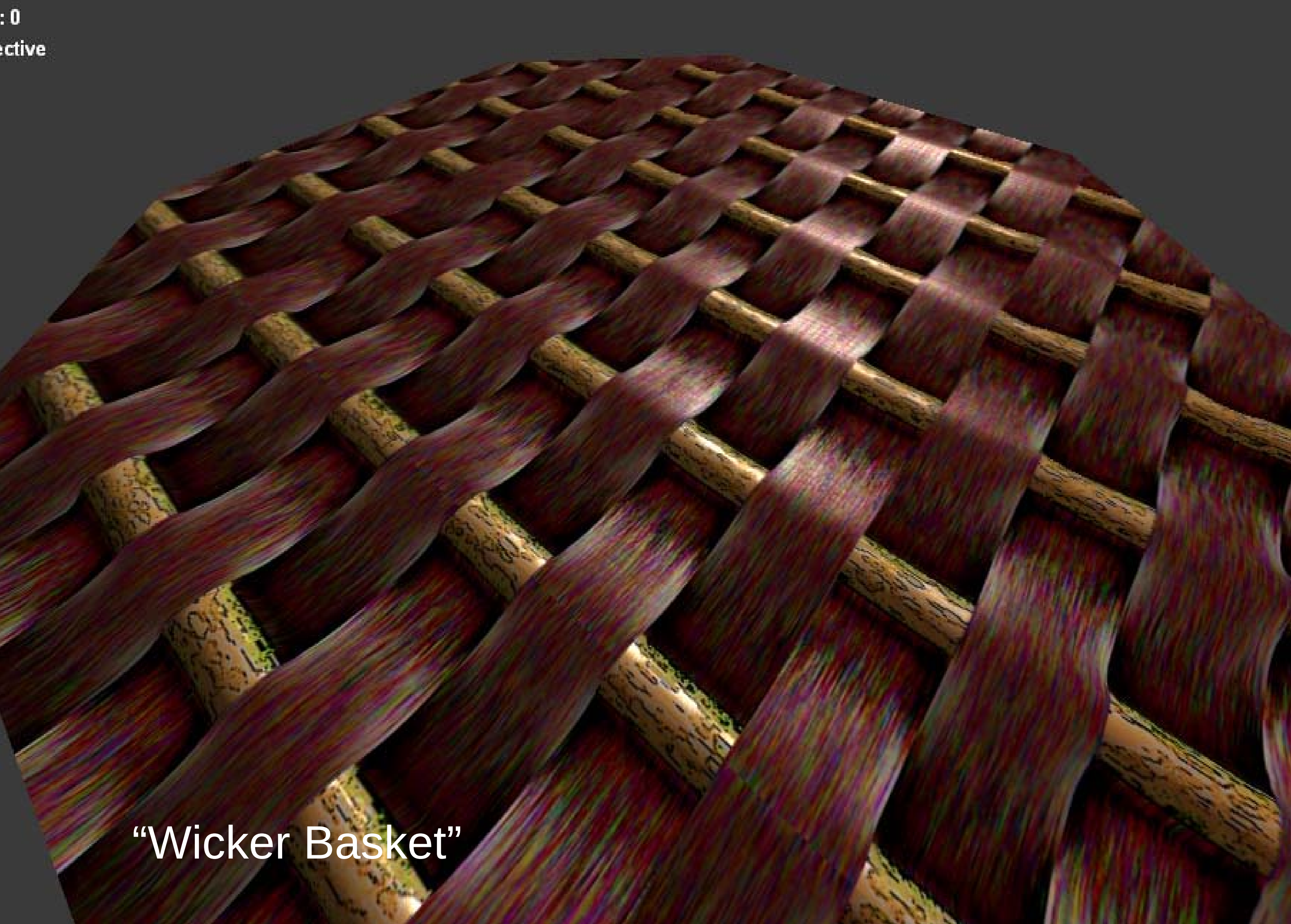
That is better in static:

Three color light sources used (no bisplacement)
Three shadows seen
Carmack’s reverse stencil shadows introduced in Doom 3 !
Double stencil was a cool optimization (DirectX 9) to make it faster!
MRT (ATI 9x00, NVIDIA 6x00) allow much faster shadow mapping with multiple light sources!
Dedicated hardware Fog is available!

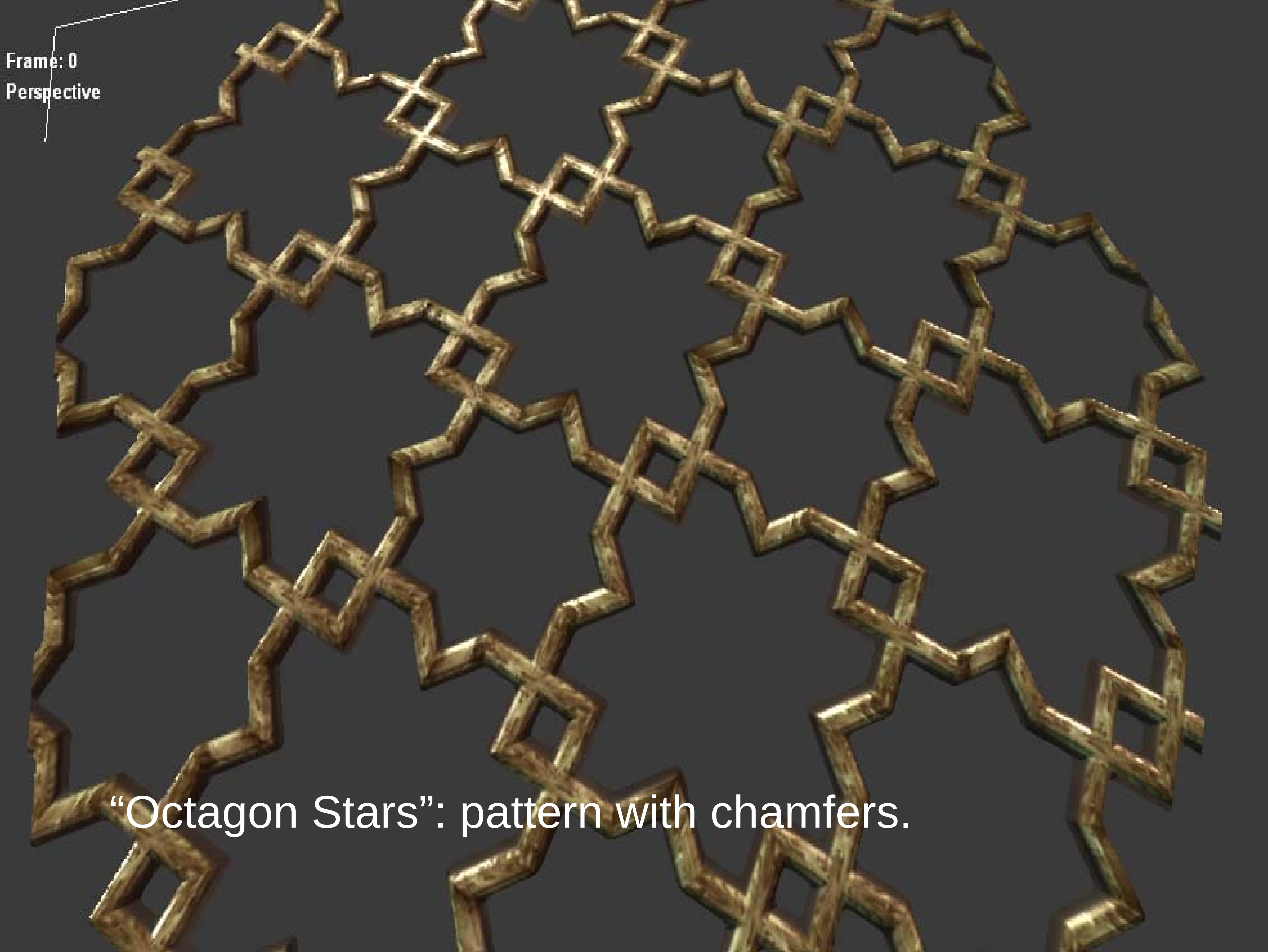
“Medieval something”



Height map could be generated directly from a contrast image.



“Wicker Basket”

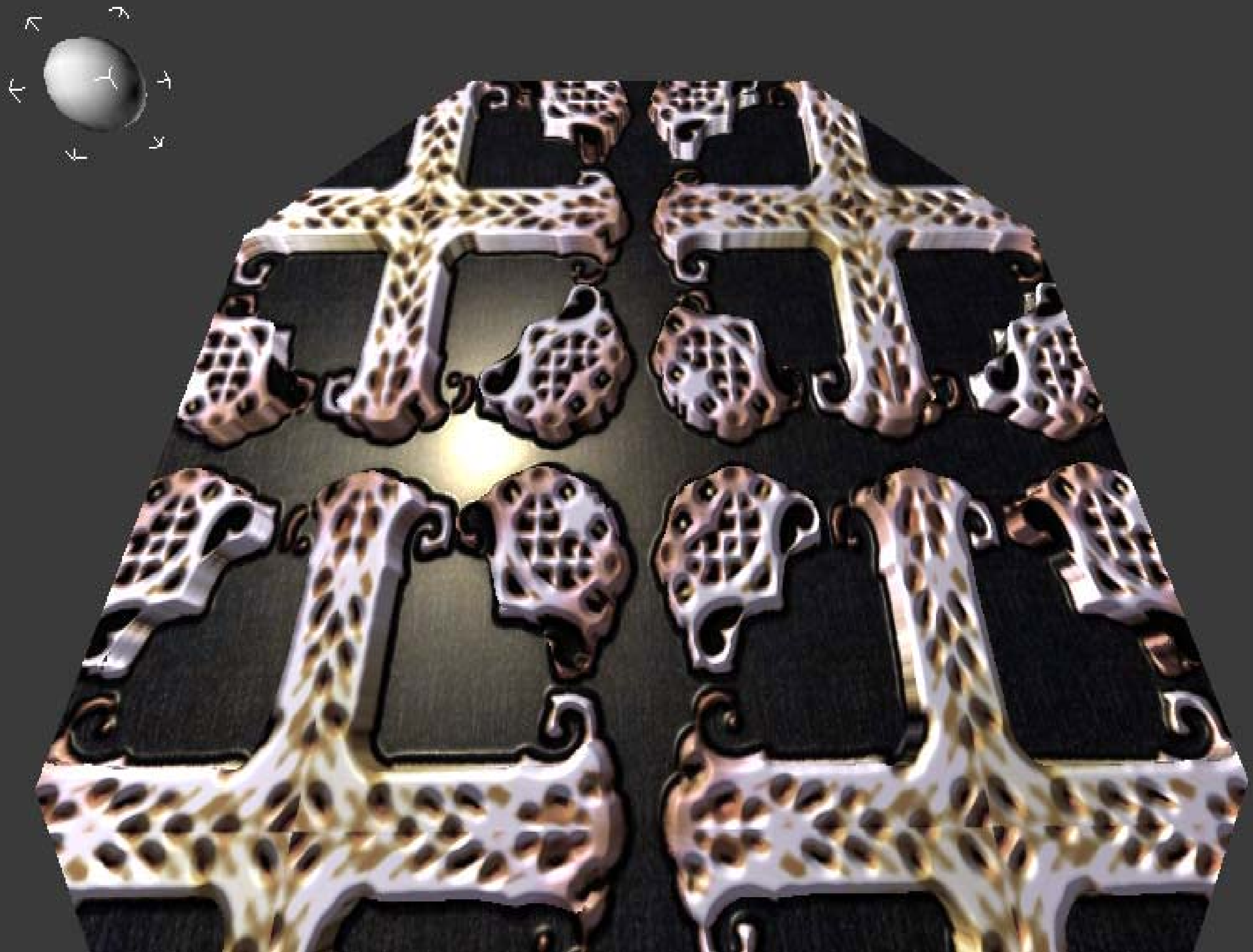


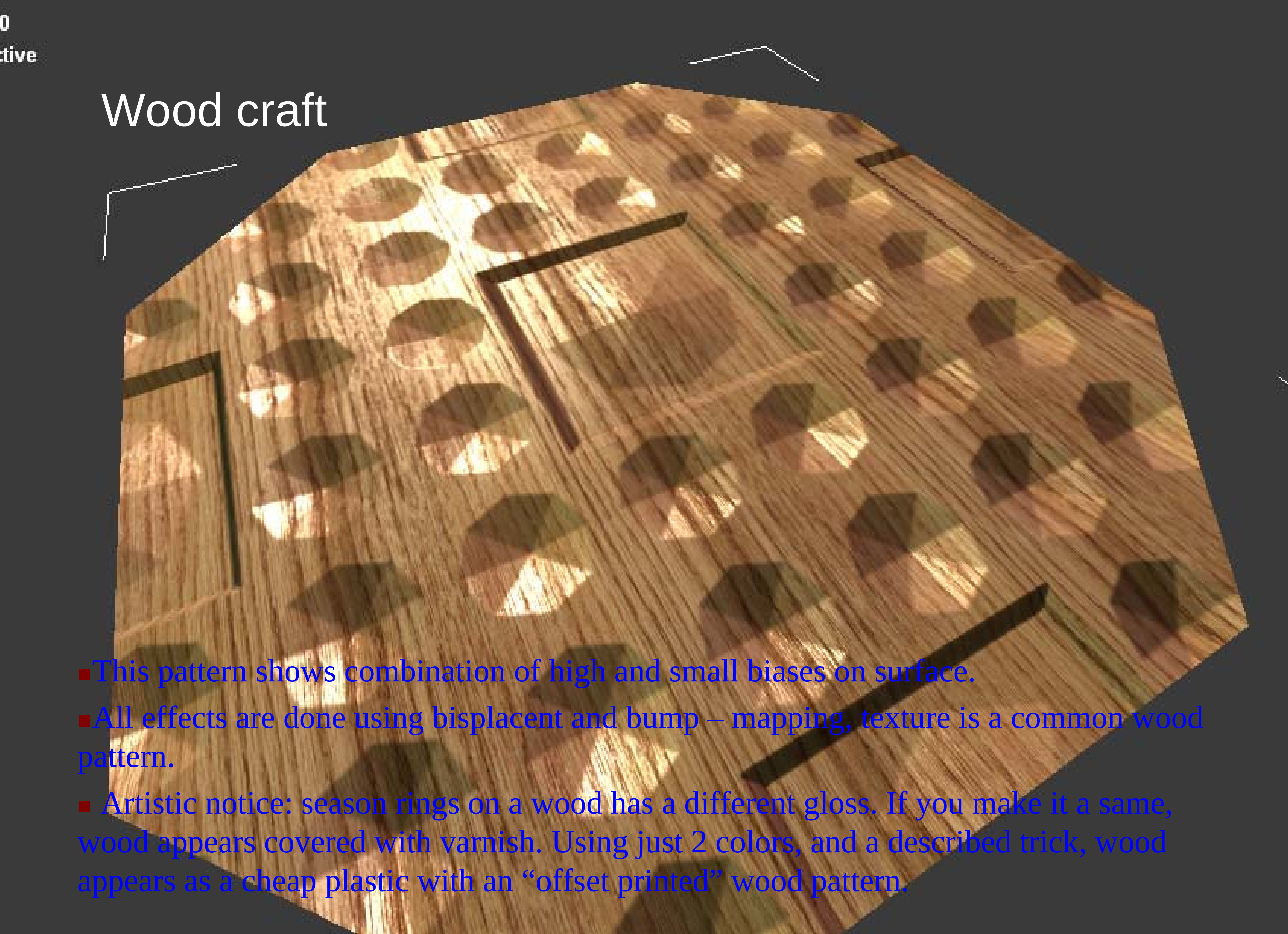
“Octagon Stars”: pattern with chamfers.

Frame: 0
Perspective

“Bronze Varnish”

- Shiny reflex seems to appear in a wrong place. But it is OK, Camera has a very small FOV.



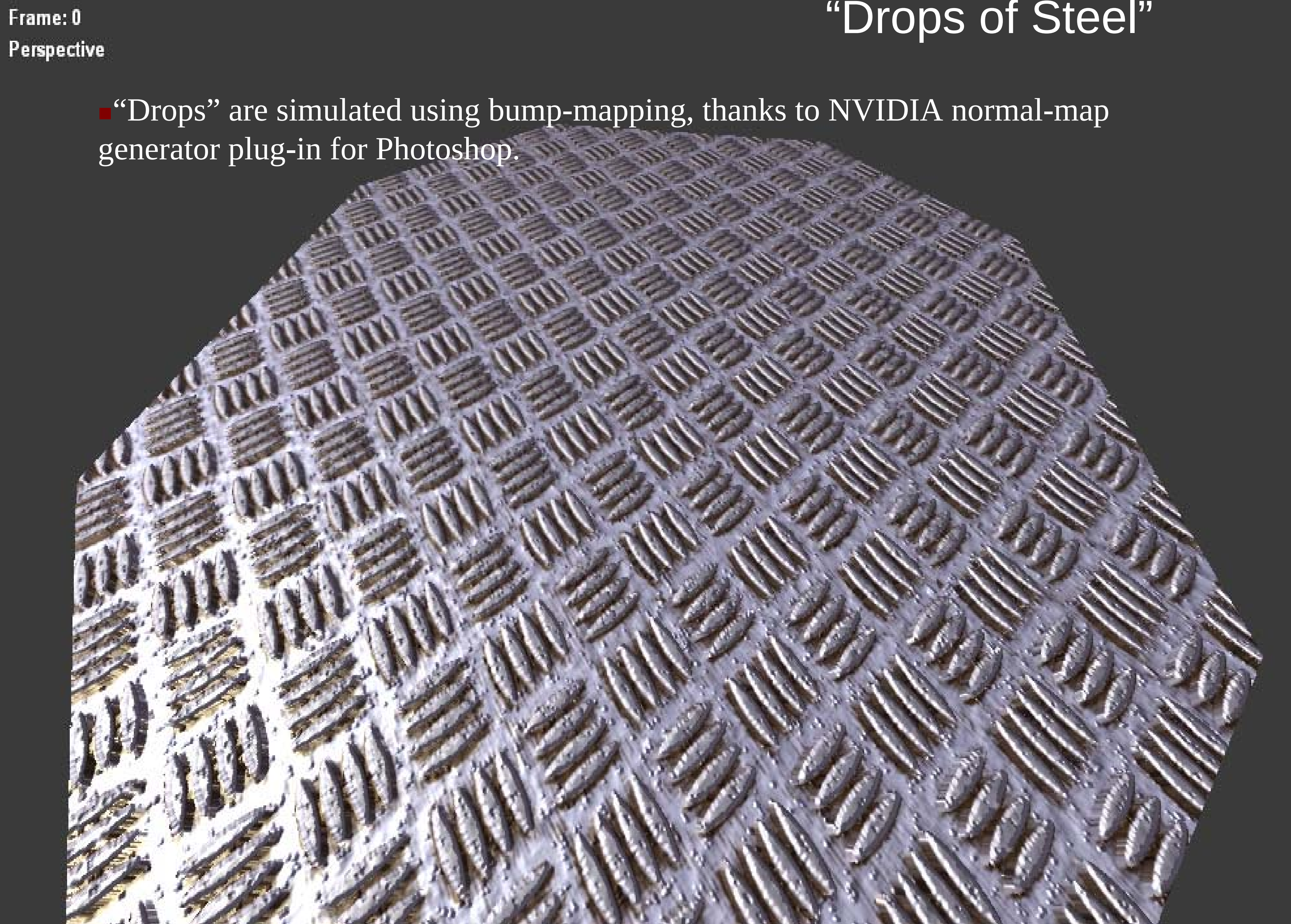


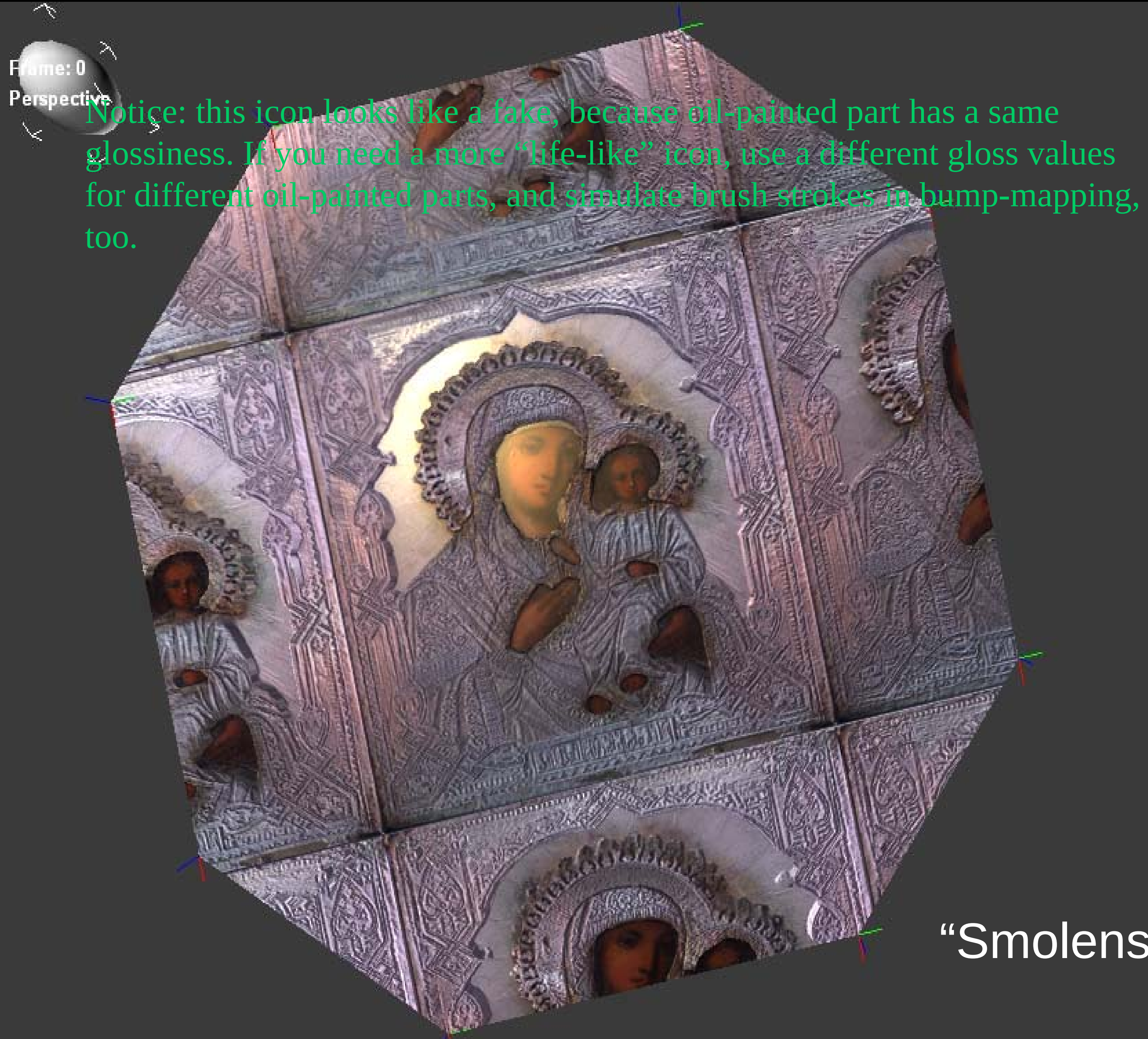
Wood craft

- This pattern shows combination of high and small biases on surface.
- All effects are done using bisplacent and bump – mapping, texture is a common wood pattern.
- Artistic notice: season rings on a wood has a different gloss. If you make it a same, wood appears covered with varnish. Using just 2 colors, and a described trick, wood appears as a cheap plastic with an “offset printed” wood pattern.

“Drops of Steel”

- “Drops” are simulated using bump-mapping, thanks to NVIDIA normal-map generator plug-in for Photoshop.

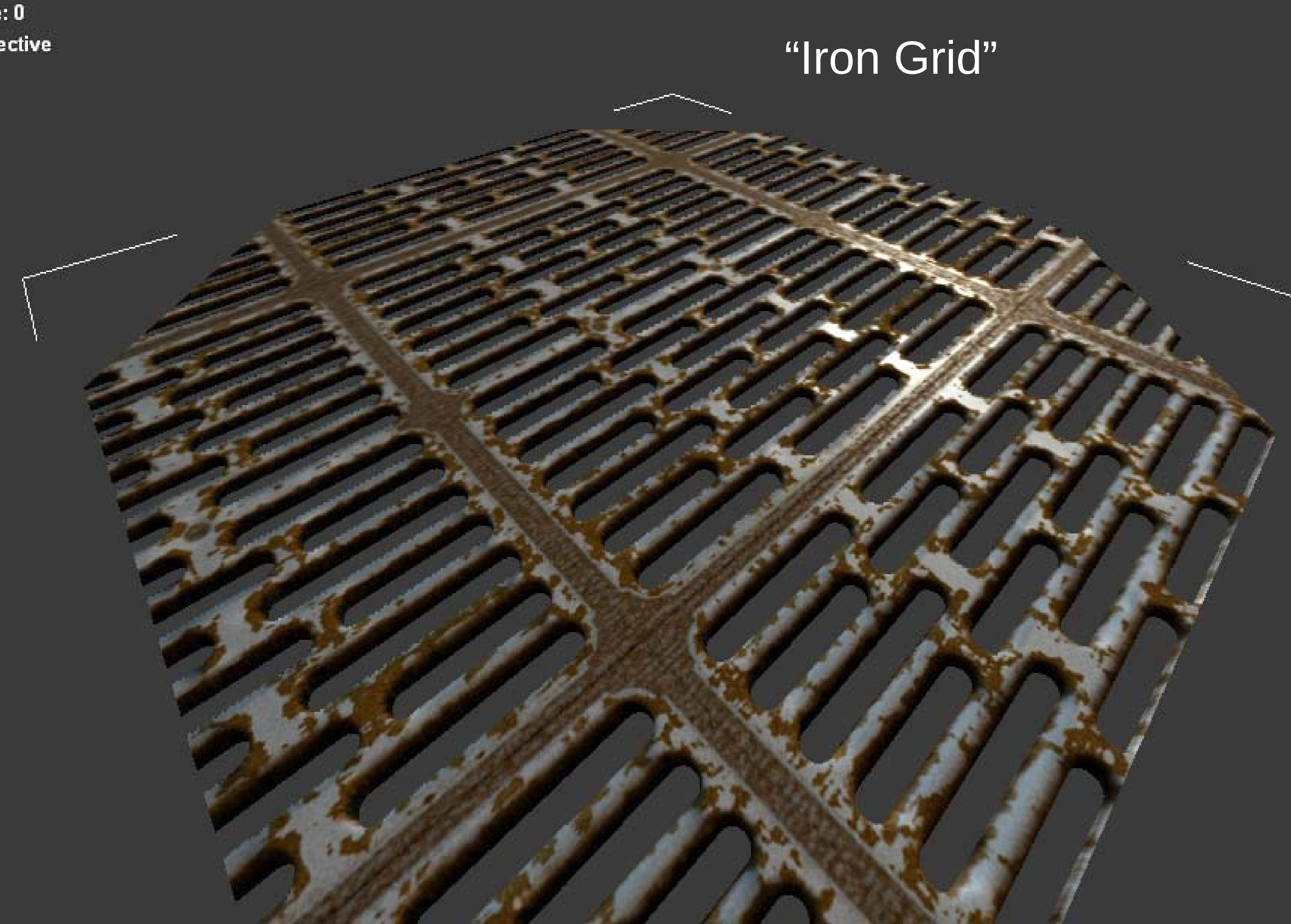


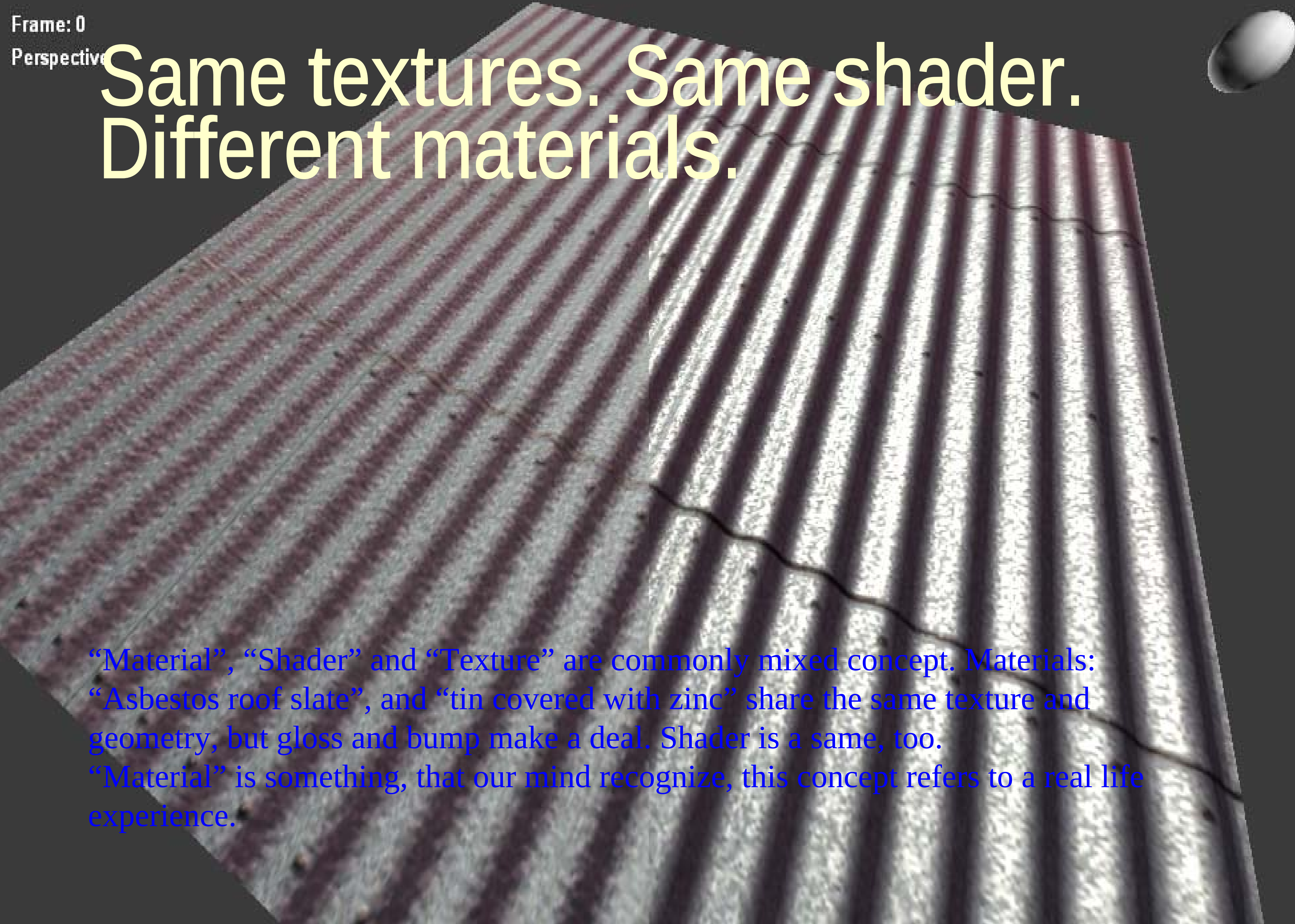


Notice: this icon looks like a fake, because oil-painted part has a same glossiness. If you need a more "life-like" icon, use a different gloss values for different oil-painted parts, and simulate brush strokes in bump-mapping, too.

"Smolensk Virgin"

“Iron Grid”



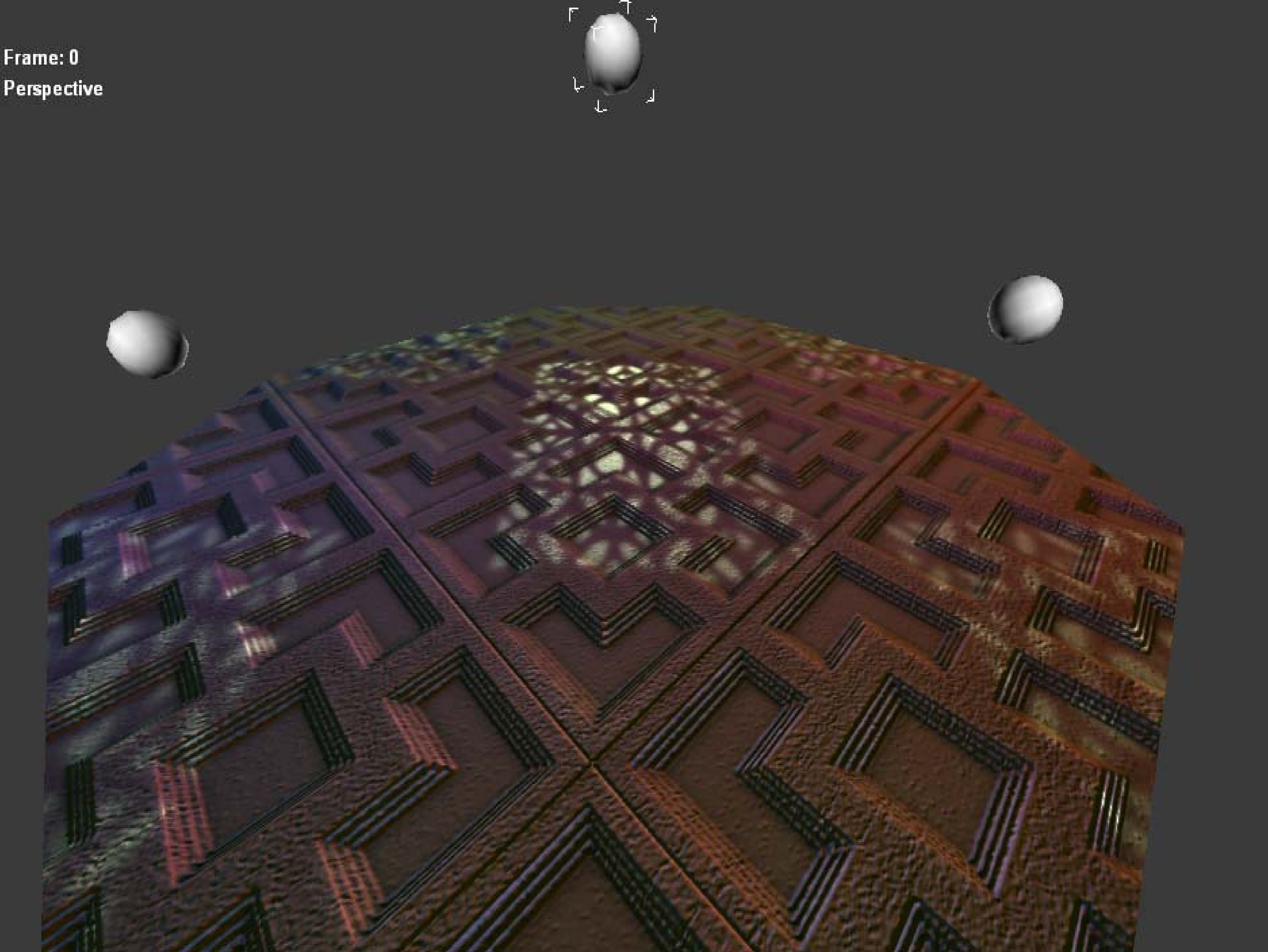


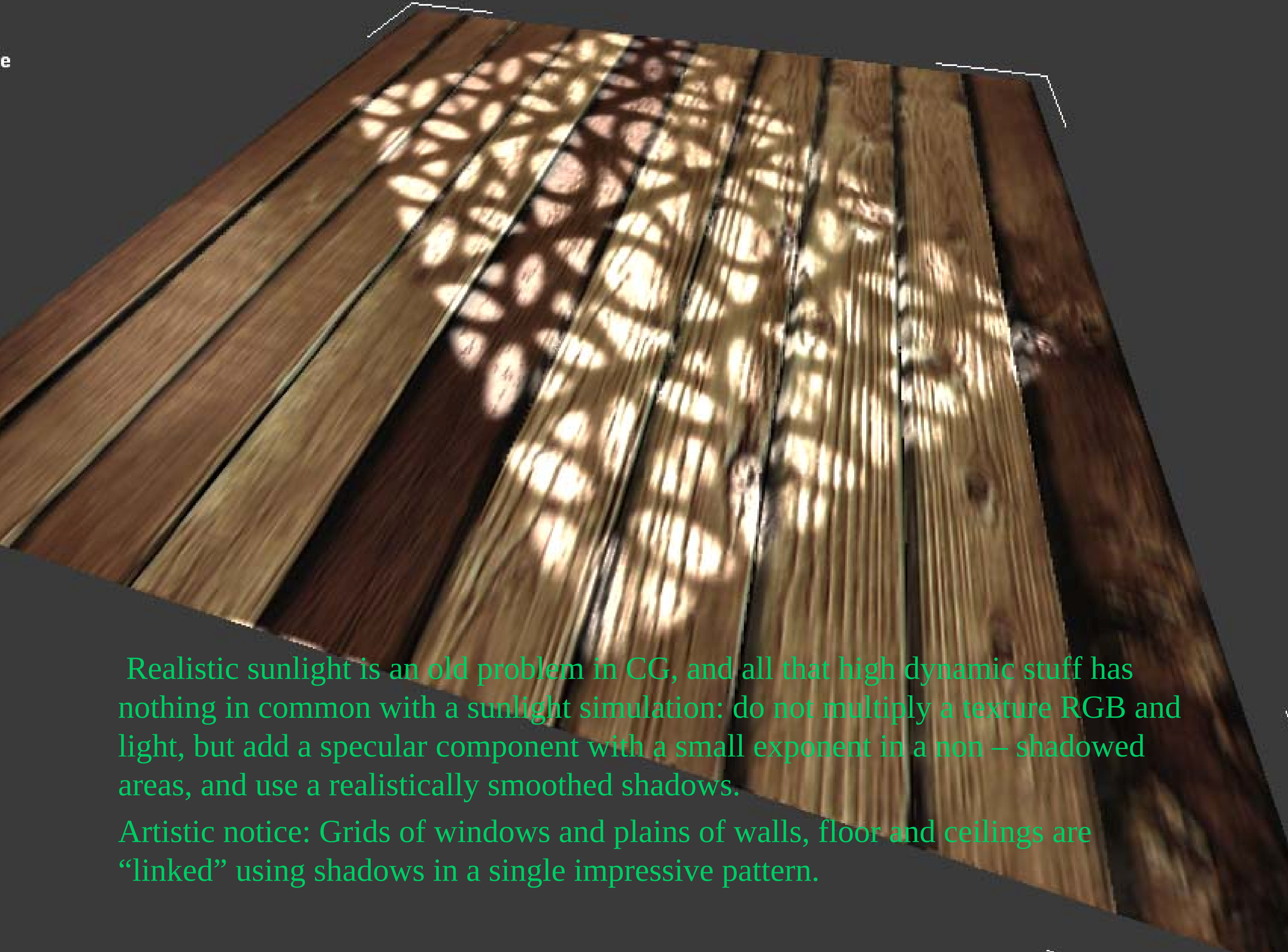
Same textures. Same shader. Different materials.

“Material”, “Shader” and “Texture” are commonly mixed concept. Materials: “Asbestos roof slate”, and “tin covered with zinc” share the same texture and geometry, but gloss and bump make a deal. Shader is a same, too.
“Material” is something, that our mind recognize, this concept refers to a real life experience.

Lightning model:

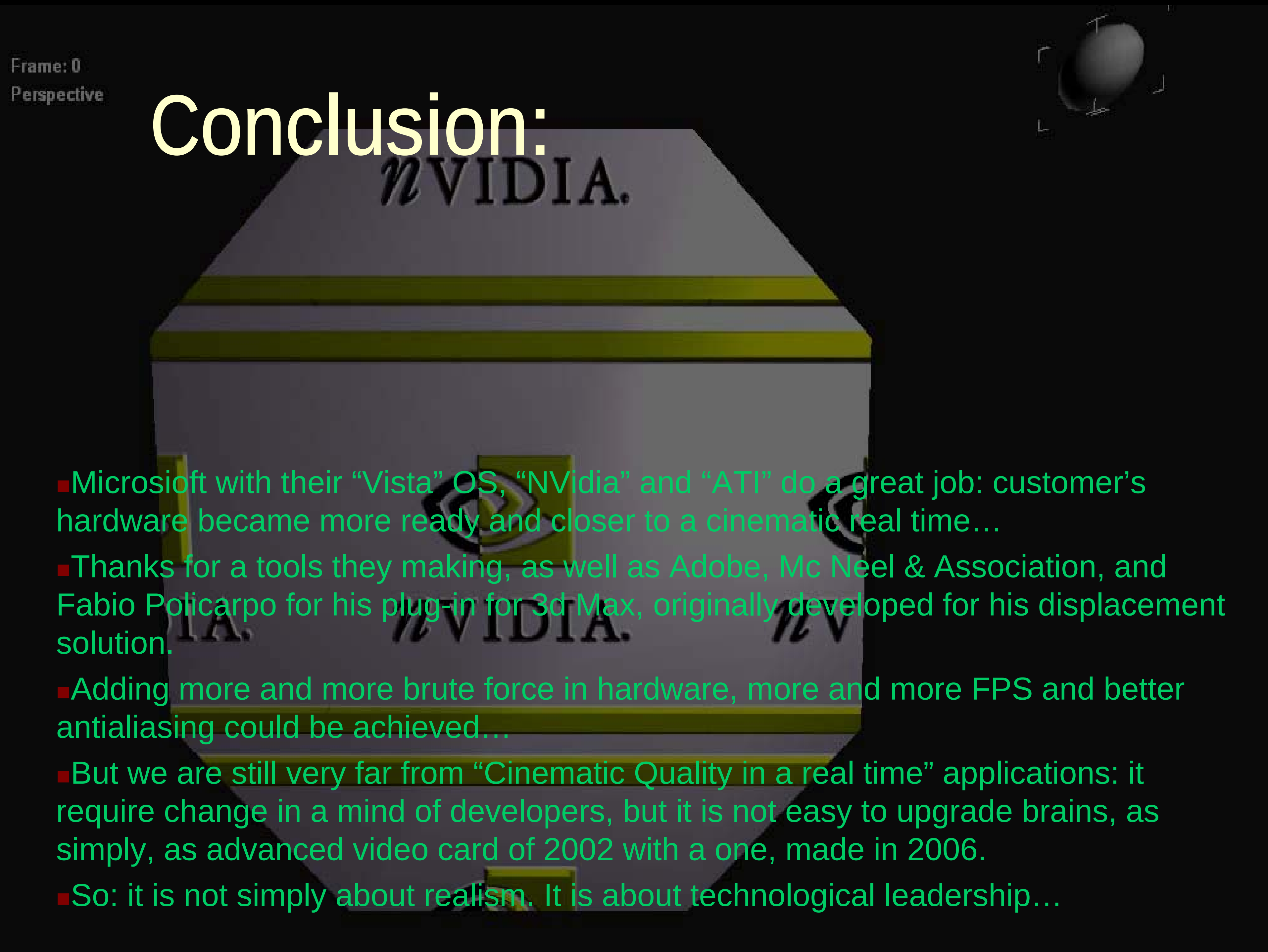
- Good lightning emphasizes a beauty of bisplacement mapping.
- There are 3 lights currently supported: 1 is static, and imitates sun, moon or main light source in a room. Lightmap is calculated using that lightsource and (if a *lightmap generation software* could), a reflections/irradiance transfer in a scene. That makes a soft lightning, and take in account light attenuation. Reflections from this light source is multiplied with a specula component of lightmap. Surely, a bump-map direction is used, when both diffuse and specular component computed.
- Another 2 lights are added: on a next sample they are made blue and red. Only diffuse component taken in account, they used to make shape much more relief and realistic. No attenuation computed.
- There is an ambient component, too, but it makes image less sharp, hence they are always made close to 0.
- In a future work, a stencil shadows would be used for each of this 3 light sources, if all 3 lightsources are shielded, ambient light would make image nicer. (Nextgen shader would make a light attenuation, too).
- Surely, everything was generated in a single pass in a PS_2_0.


Frame: 0
Perspective



Realistic sunlight is an old problem in CG, and all that high dynamic stuff has nothing in common with a sunlight simulation: do not multiply a texture RGB and light, but add a specular component with a small exponent in a non – shadowed areas, and use a realistically smoothed shadows.

Artistic notice: Grids of windows and plains of walls, floor and ceilings are “linked” using shadows in a single impressive pattern.



Conclusion:

- Microsioft with their “Vista” OS, “NVidia” and “ATI” do a great job: customer’s hardware became more ready and closer to a cinematic real time…
- Thanks for a tools they making, as well as Adobe, Mc Neel & Association, and Fabio Policarpo for his plug-in for 3d Max, originally developed for his displacement solution.
- Adding more and more brute force in hardware, more and more FPS and better antialiasing could be achieved…
- But we are still very far from “Cinematic Quality in a real time” applications: it require change in a mind of developers, but it is not easy to upgrade brains, as simply, as advanced video card of 2002 with a one, made in 2006.
- So: it is not simply about realism. It is about technological leadership…

Future work

- Implement self shadowing term from a “hemisphere light source”
- Achieve greater depths and higher frequencies of height field in PS_3_0
- More material samples (ceramics, wood, plastic, anisotropic etc.) and more 3d objects with bisplacement…
- Apply Stencil Shadows in a multi-pass rendering.

Some comments:

- *.fx, *.fxproj, *.x and all textures are available upon request, after negotiations ☺
- Shaders were tested on ATI9800 and NVIDIA 6600GT, no visible difference.

Appendix: Vertex Shader

```
vertexOutput VS_HeuristicalDisplacement(vertexInput IN)
{
    vertexOutput OUT;
    OUT.hPosition = mul( float4(IN.position.xyz , 1.0) , worldViewProj);
    OUT.texCoordDiffuse = IN.texCoordDiffuse;
    OUT.texCoordBump    = IN.texCoordDiffuse;   ;

   // Calculate Vertex world space position.
   float4 pos = mul(worldViewProj, float4(IN.position.xyz, 1.f));

   // Build TBN matrix.
   float3x3 tbnMatrix;

   tbnMatrix[0] = mul( IN.tangent,  world);
   tbnMatrix[1] = mul( IN.binormal, world);
   tbnMatrix[2] = mul( IN.normal,   world);

   float3 worldEyePos = viewInverse[3].xyz;

   // Set the view direction; convert to texture space.
   OUT.viewDirect  = worldEyePos - IN.position.xyz;
   OUT.viewDirect  = normalize (mul((tbnMatrix), OUT.viewDirect));

   // Set the light direction; convert to texture space.

   OUT.lightDirection = lightPos - IN.position.xyz;
   OUT.lightDirection = normalize (mul((tbnMatrix), OUT.lightDirection));
   return OUT;
}
```

Appendix: Fragment Shader ps_2_0 or PS_2_b info.

- ***************************************
- Target: GeForce 6800 Ultra (NV40) :: Unified Compiler: v77.72
- Cycles: 34.50 :: R Regs Used: 6 :: R Regs Max Index (0 based): 5
- Pixel throughput (assuming 1 cycle texture lookup) 188.24 MP/s
- ==
- Shader performance using all FP16
- Cycles: 28.50 :: R Regs Used: 3 :: R Regs Max Index (0 based): 2
- Pixel throughput (assuming 1 cycle texture lookup) 228.57 MP/s
- ==
- Shader performance using all FP32
- Cycles: 34.50 :: R Regs Used: 6 :: R Regs Max Index (0 based): 5
- Pixel throughput (assuming 1 cycle texture lookup) 188.24 MP/s
- ***************************************
- PS Instructions: 60
- ps_2_0

PS_3_0 info

- **
- Target: GeForce 6800 Ultra (NV40) :: Unified Compiler: v77.72
- Cycles: 51.50 :: R Regs Used: 5 :: R Regs Max Index (0 based): 4
- Pixel throughput (assuming 1 cycle texture lookup) 125.49 MP/s
- ==
- Shader performance using all FP16
- Cycles: 46.50 :: R Regs Used: 4 :: R Regs Max Index (0 based): 3
- Pixel throughput (assuming 1 cycle texture lookup) 139.13 MP/s
- ==
- Shader performance using all FP32
- Cycles: 51.50 :: R Regs Used: 5 :: R Regs Max Index (0 based): 4
- Pixel throughput (assuming 1 cycle texture lookup) 125.49 MP/s
- **
- PS Instructions: 67
- ps_3_0

References

- [Blinn 1978], Blinn James F. "Simulation of Wrinkled Surfaces" Computer graphics vol.12 pp286 – 292. Proc. Siggraph 78. http://research.microsoft.com/users/blinn/
- [Fernando-Kilgard 2003] Kilgard Mark J. "The Cg Tutorial" Addison – Wesley 2003. "Chapter 8. Bump Mapping" pp 199 – 233.
- [Kaneko 2001] Kaneko T. et Al "Detailed shape representation with parallax mapping". In Proceedings of the ICAT 2001 (The 11th international conference on artificial reality and Teleexistence), Tokyo, Dec. 2001 pp.205-208
- [Welsh 2004] Welsh T. "Parallax Mapping with offset limiting: a Per Pixel approximation of uneven surfaces", 2004 http://www.infiscape.com/doc/parallax_mapping.pdf
- [Wang 2003] Wang L et al. "View dependent displacement mapping", Siggraph 2003
- [Brawley – Tatarchuk 2004] Brawley Z, Tatarchuk N. "Parallax occlusion mapping: Self Shadowing, Perspective correct bump-mapping using Reverse Height Map Tracing", ShaderX3, 2004
- [Policarpo 2004] Policarpo Fabio "Relief Mapping in a Pixel Shader using Binary Search." http://www.paralelo.com.br/arquivos/ReliefMapping.pdf
- [Oliveira 2000] Oliveira Manuel. "Relief Texture Mapping", Siggraph 2000 http://www.cs.unc.edu/~ibr/pubs/oliveira-sg2000/RTM.pdf
- [Kajiya 1996]]J. Torborg and J.T. Kajiya, "Talisman: Commodity Realtime 3D Graphics for the PC," Proc. ACM Conf. on Computer Graphics Conference(SIGGRAPH '96), ACM, New York, NY, 1996, pp. 353-363.
- [Donnelly 2005] "Per Pixel Displacement Mapping with Distance Functions" GPU Gems2 2005
- [McGuire 2005] McGuire Morgan, McGuire Max. "Steep Parallax Mapping" I3D 2005.
- [Lobel 2004] Lobel Robin. www.divideconcept.net