08 Detail mapping

Steve Marschner

CS5625 Spring 2019

Hierarchy of scales

macroscopic

mesoscopic

microscopic

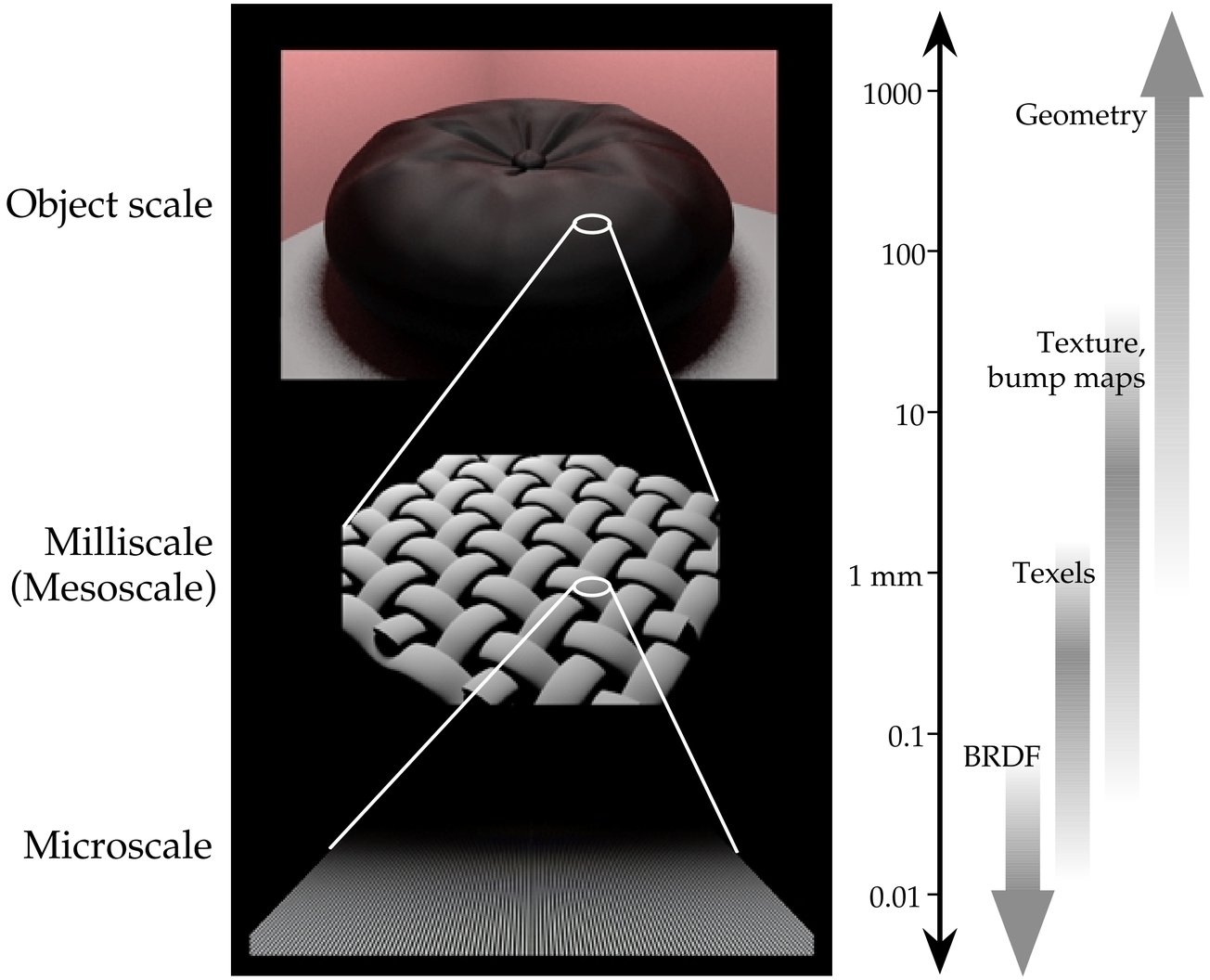
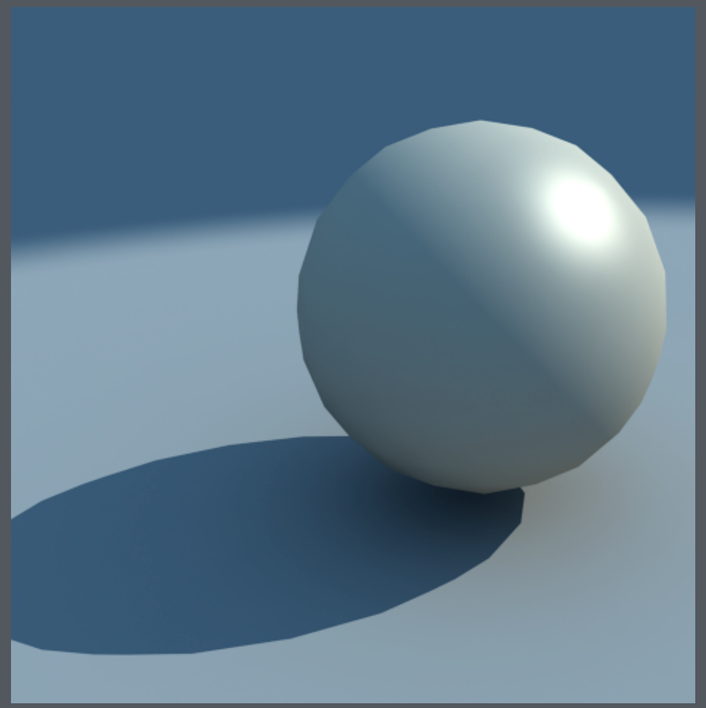


Figure 1: Applicability of Techniques

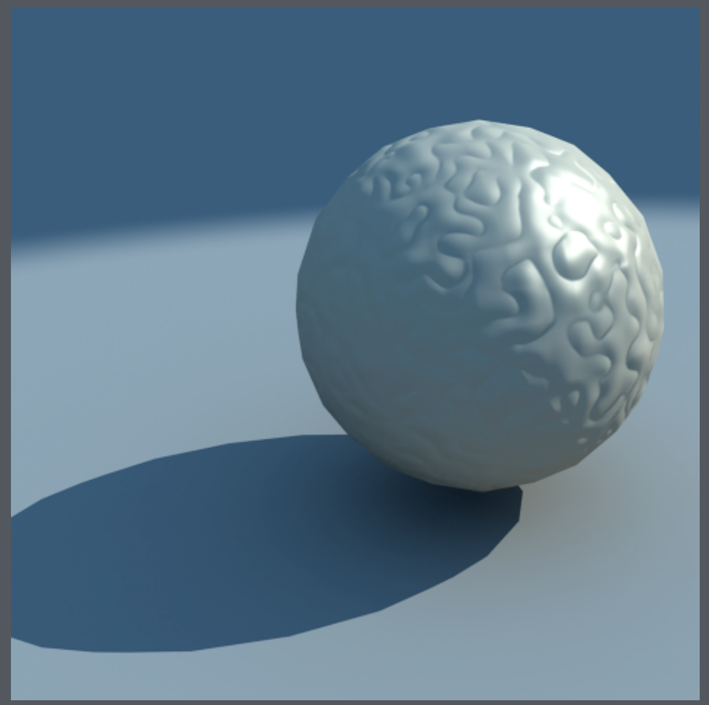
Displacement and Bump/Normal Mapping

Mimic effect of geometric detail/meso geometry

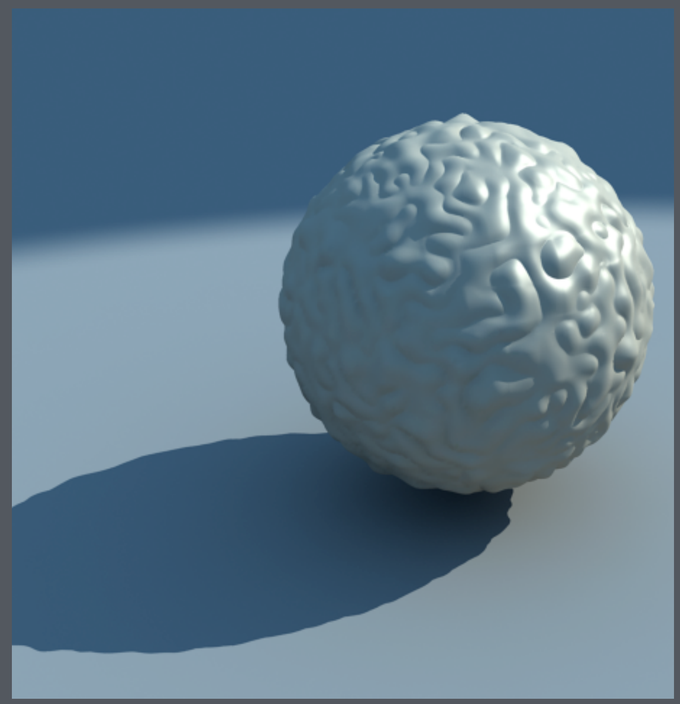
- Also detail mapping



Geometry



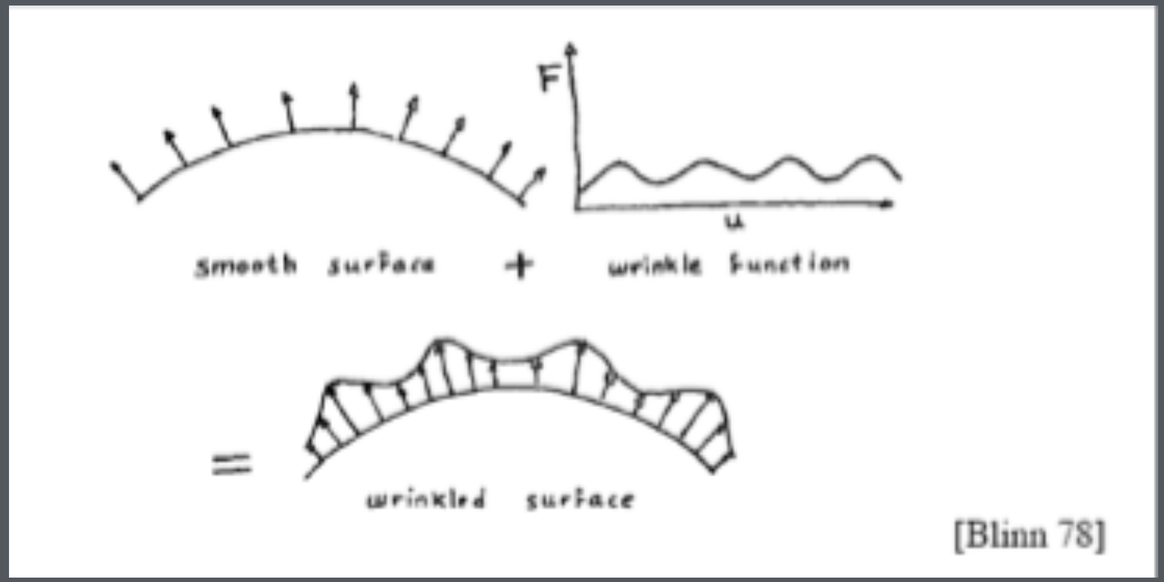
Bump mapping



Displacement mapping

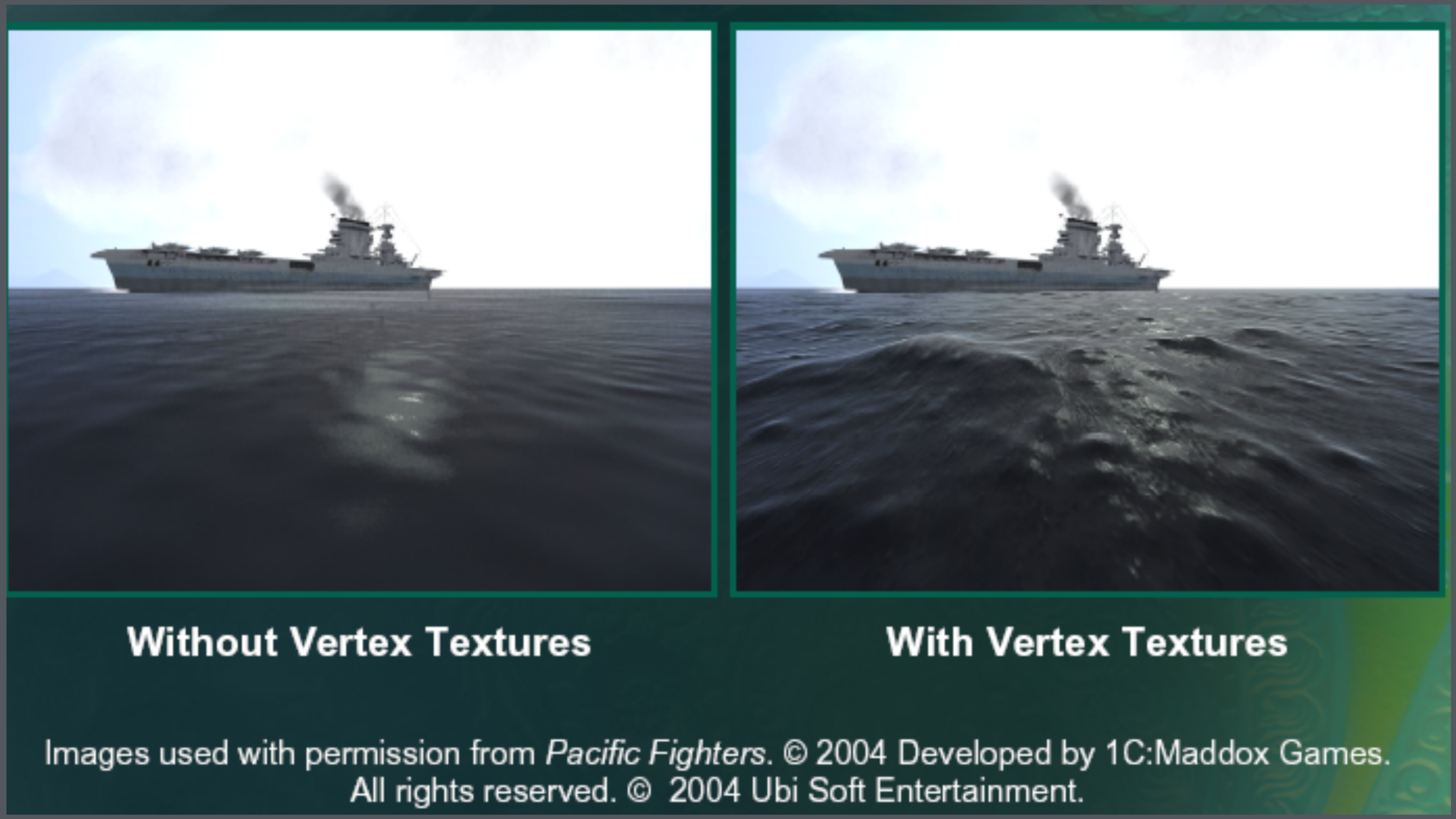
Displacement Mapping

- `Pnew = Pold + DM(u)* N`



$$\mathbf{p}'(u,v) = \mathbf{p}(u,v) + h(u,v)\mathbf{n}(u,v)$$

Displacement in vertex shader



Displacement Maps

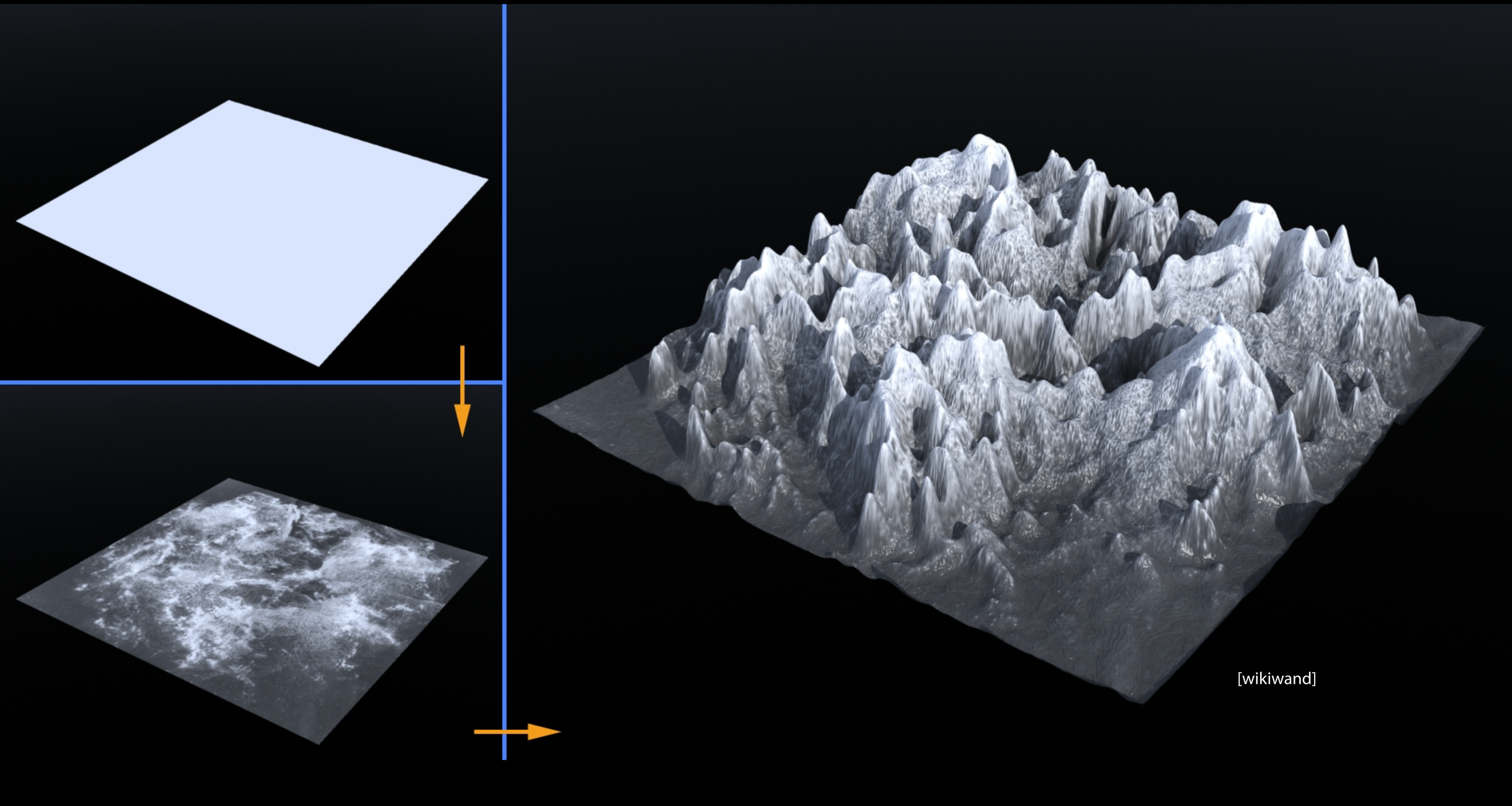
Pros

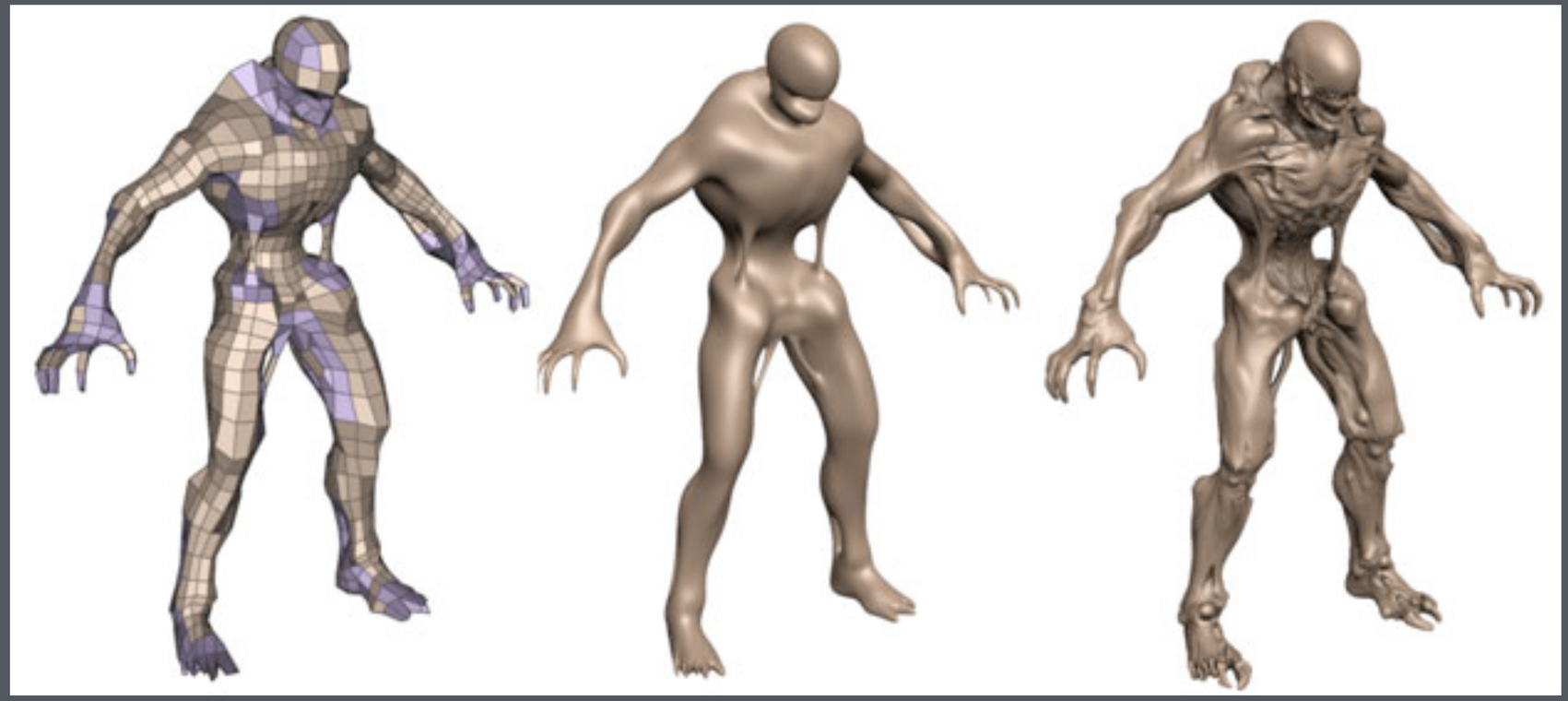
- Gives you very complex surfaces

Cons

- Gives you very complex surfaces
- Or boring with small numbers of vertices

Relationship with tesselation shaders


[wikiwand]



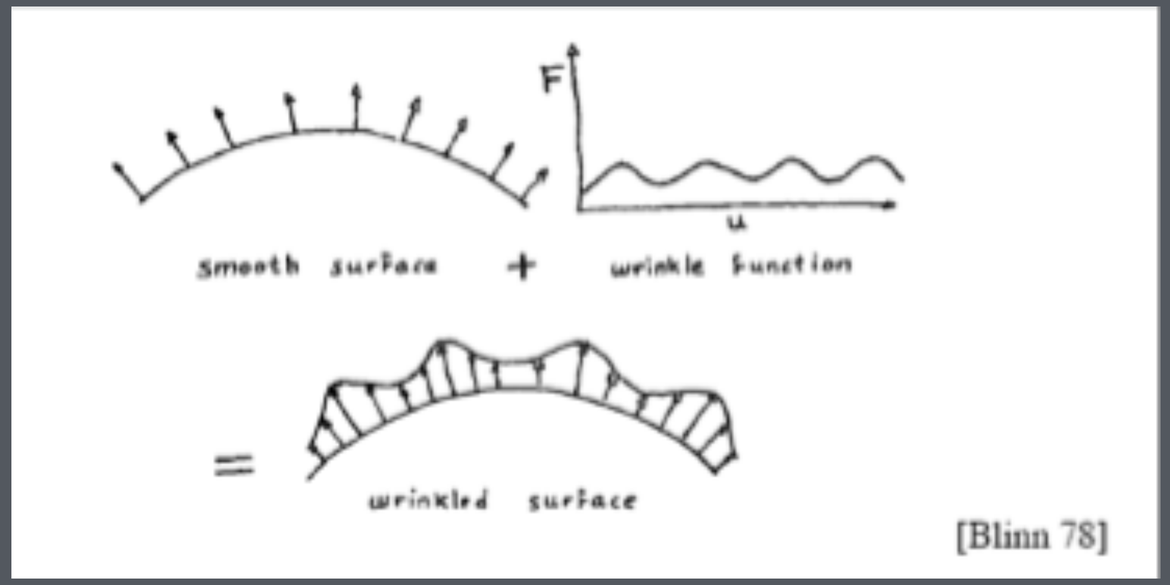
Original

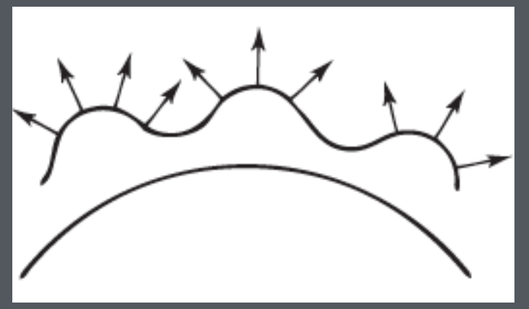
Tesselated

Displacement Mapped

Bump mapping

“Simulation of Wrinkled Surfaces” Blinn 78

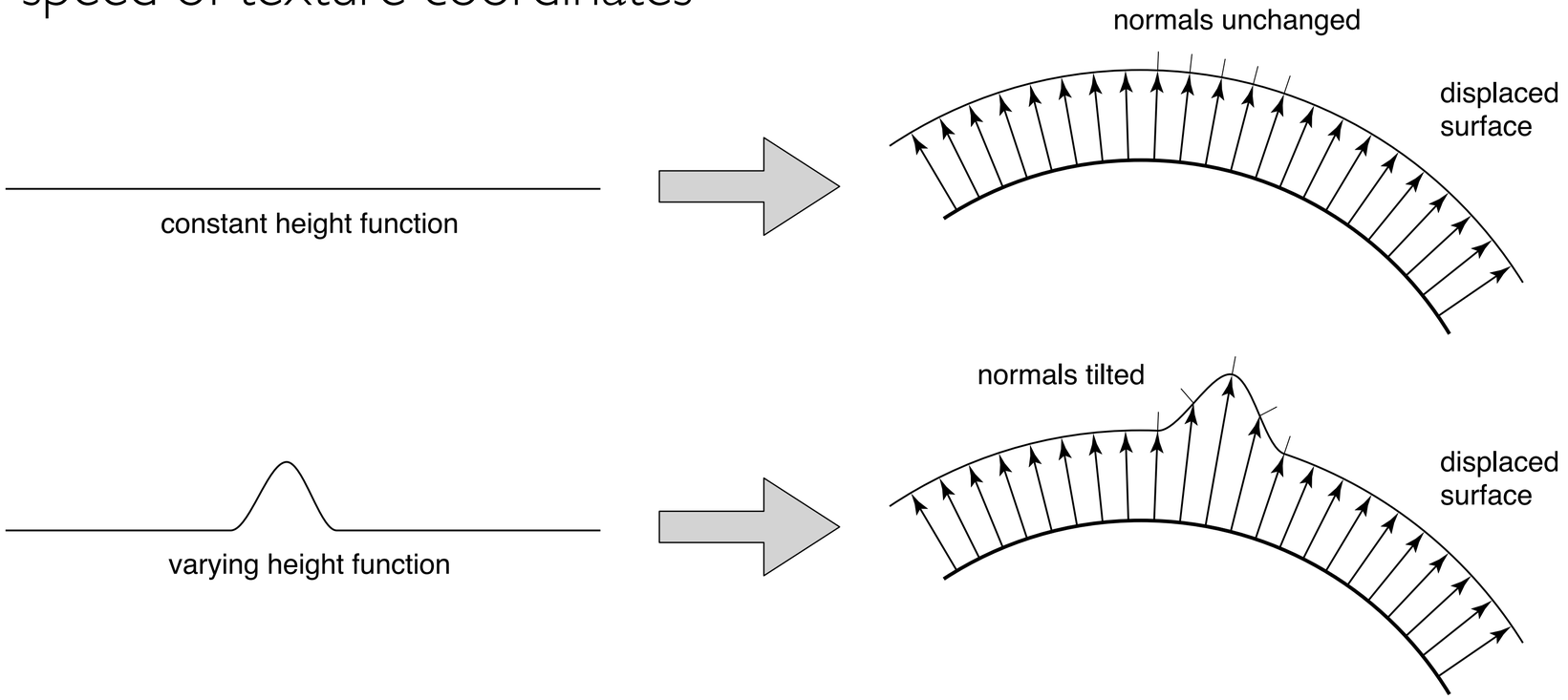




Blinn: keep surface, use new normals

Normals in displacement mapping

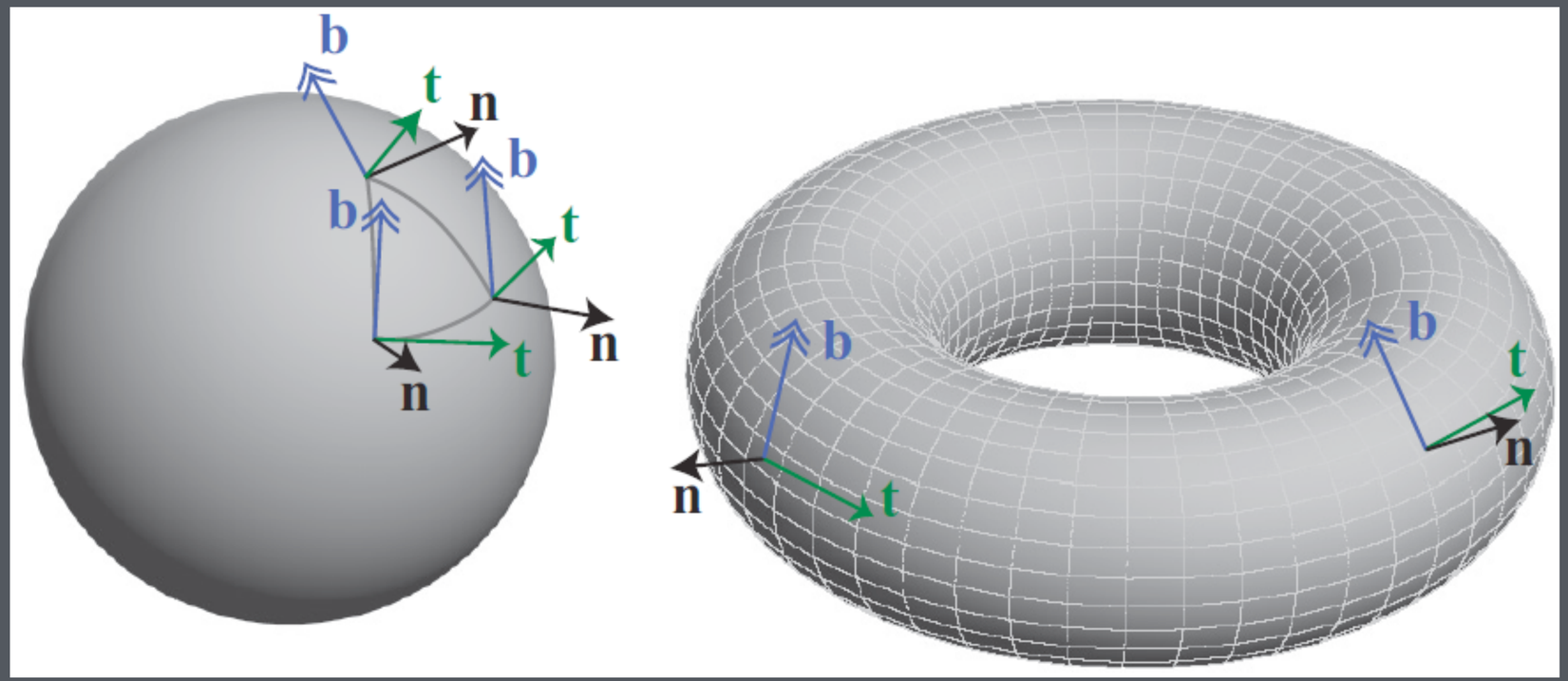
- **Displacement changes the surface normal, depending on:**
  - derivative of height function
  - orientation of texture coordinates
  - speed of texture coordinates



How to change the normal?

First, need some frame of reference

- Normal is modified with respect to that
- Have tangent space basis: t and b
- Normal, tangent and bitangent vectors



Geometry for displacement

- **geometric inputs**
  - *u* tangent (unnormalized) as vertex attribute
  - *v* tangent (unnormalized) as vertex attribute
  - height field as a texture
- **vertex stage**
  - compute displaced vertex position
    - look up displacement value from texture
  - compute normal to displaced surface
    - compute derivatives of height by finite differences
    - add offset to the base surface tangents
    - normalized cross product is the shading normal
- **fragment stage: just compute shading**

(or compute them ahead of time and store height and derivatives in a 3-channel texture)

Bump Mapping

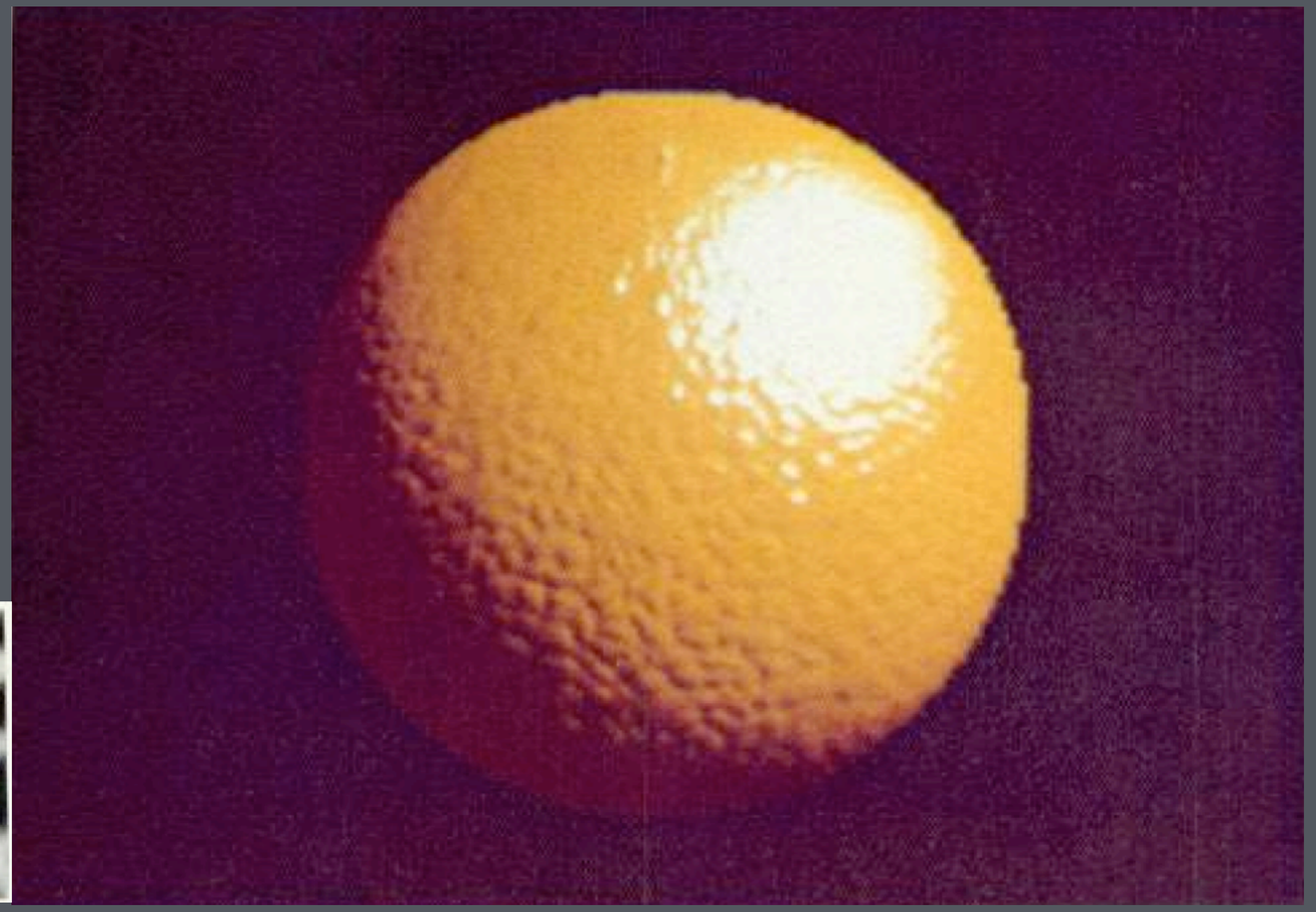
Alter normals of surface

- Only affects shading normals

Also, mimics effect of small scale geometry

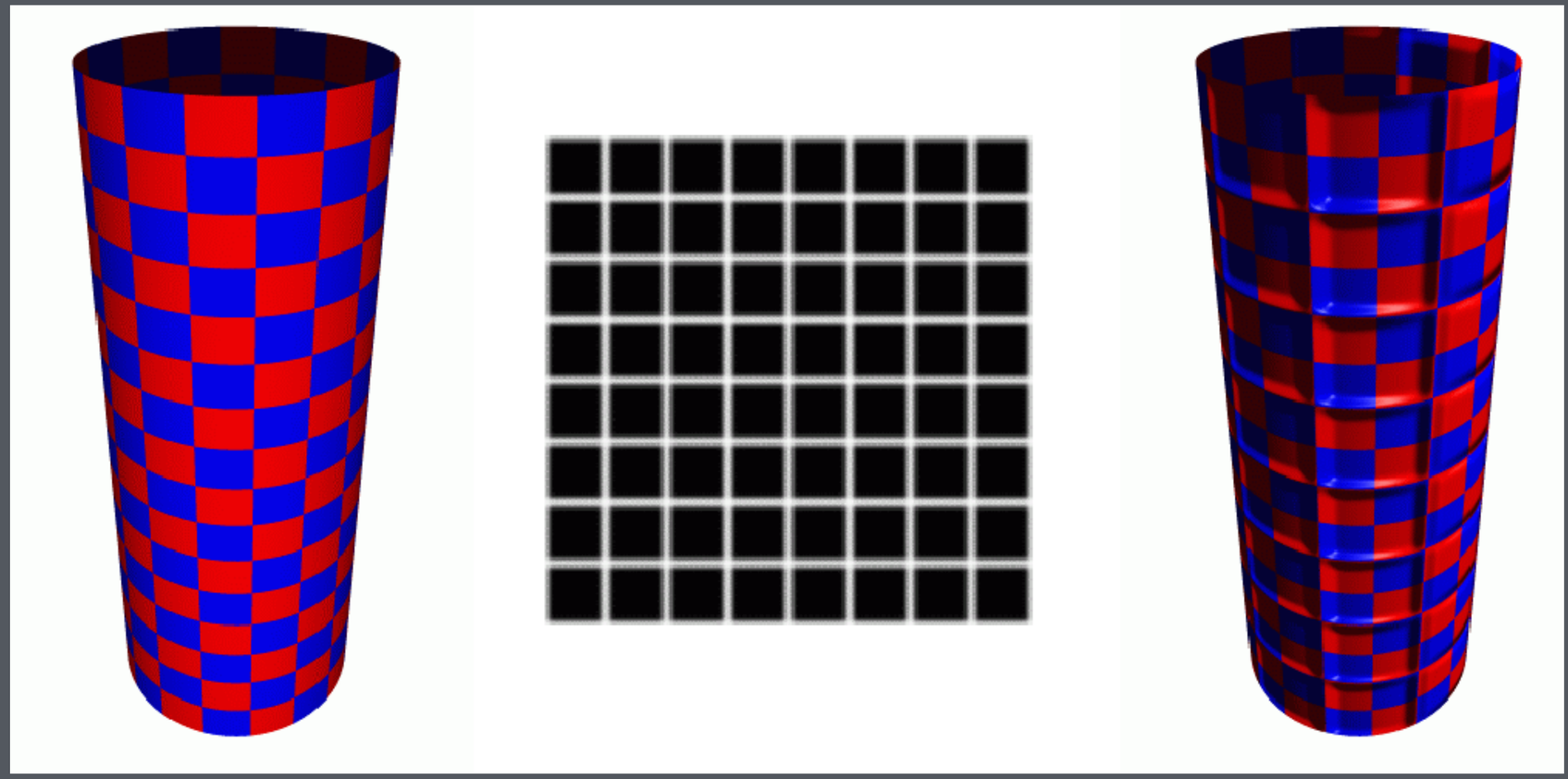
- Detail map
- Except at silhouette
- Adds perceived bumps, wrinkles

Bump mapping



[Blinn 1978]

Bump Mapping



Bump mapping

Displacement mapping is expensive

- requires densely tessellated geometry
- many triangles to rasterize

For small displacements, the most important effect is on the normal

- so just do that part; don’t displace the surface

Bump mapping is then a fragment operation

- doesn’t require dense tessellation
- doesn’t actually displace the surface
- gives shading that looks just like displaced surface

Bump Mapping

Bump mapping

Geometric inputs

- tangent vectors (unnormalized) as vertex attributes
- height field as a texture
- no dense triangulation needed

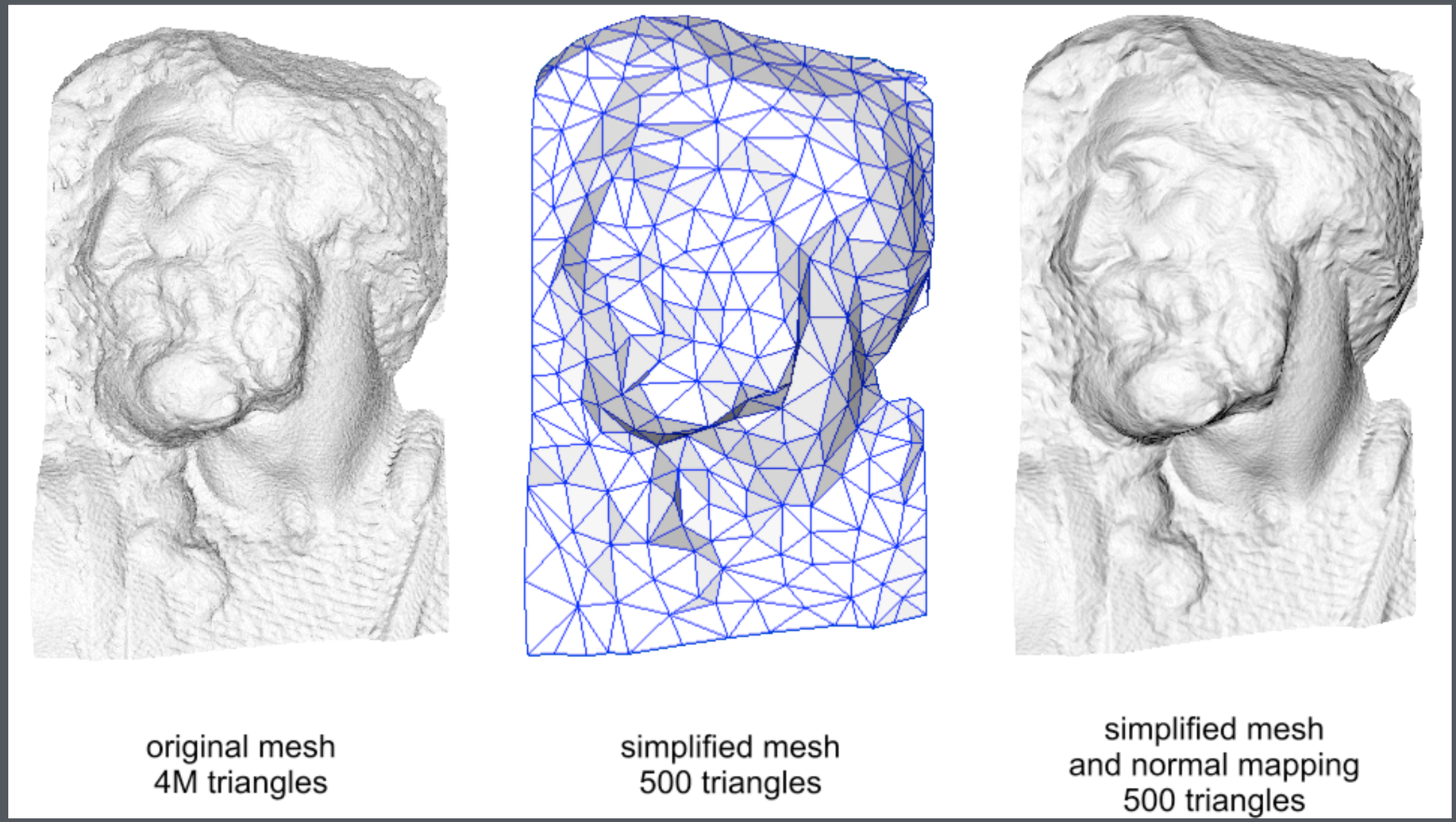
Vertex phase

- simply transform and pass through the position and tangents

Fragment phase

- compute normal to displaced surface
  - compute derivatives of height by finite differences
  - add offset to the base surface tangents; cross product is the shading normal
- compute shading using displaced normal

Normal mapping



[Paolo Cignoni]

Normal mapping

Geometric prerequisites

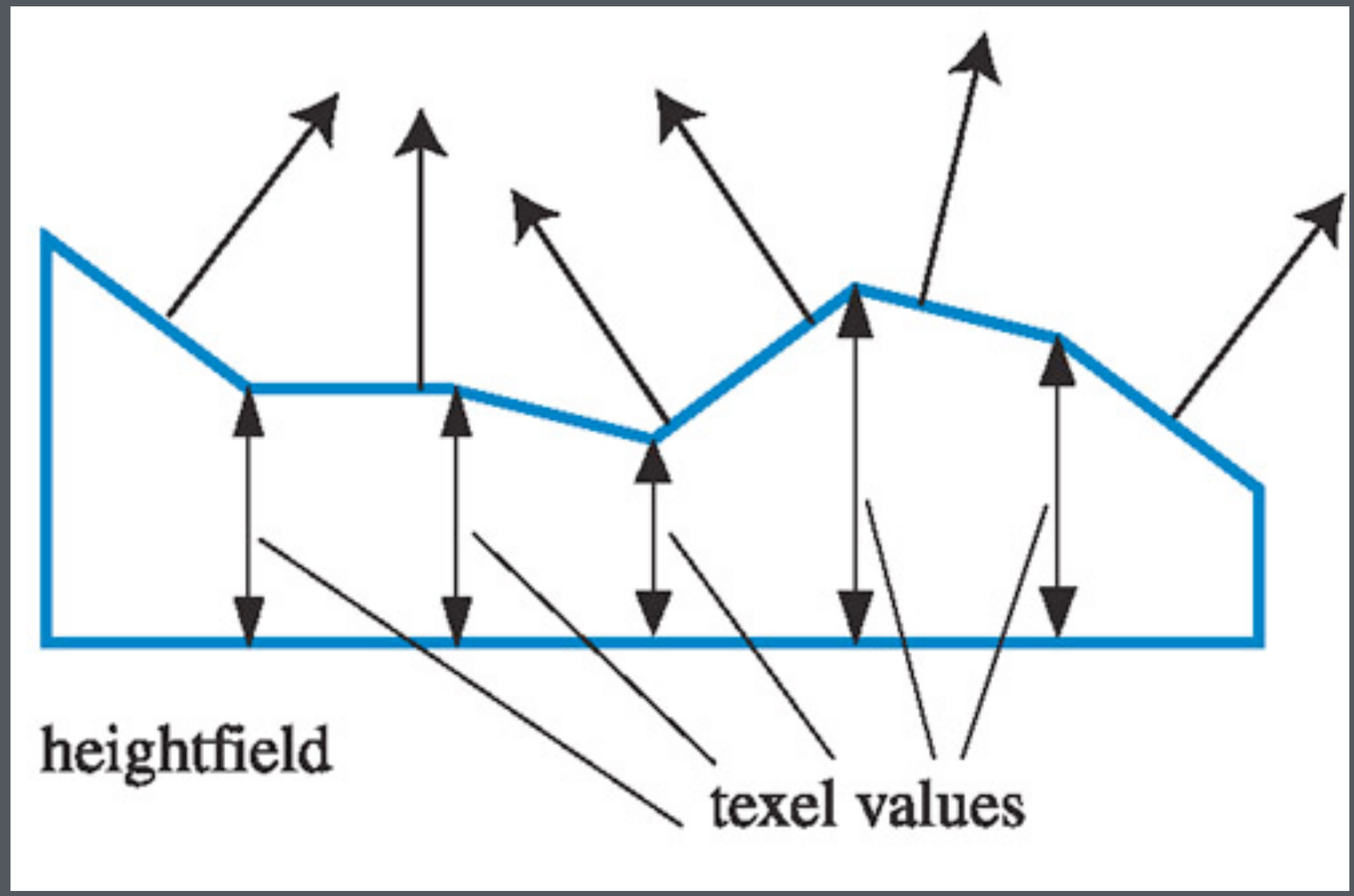
- Texture map (3 channels) representing normal field
  - single lookups into normal map required
- Smooth normals
- Unit tangent vectors
  - if you want to store normals in tangent space (and you do)
- No dense triangulation needed
- No finite differencing needed

Geometric logic

- look up normal from map
- transform into (tangent-u, tangent-v, normal) space

Heightfield: Blinn's original idea

Single scalar, more computation to infer N'



Perturbed normal given height map

Normal is determined by partial derivatives of height

- in the local frame of the displacement map:

$$\mathbf{n}_{\text{disp}} = (h_u, h_v, 1)$$

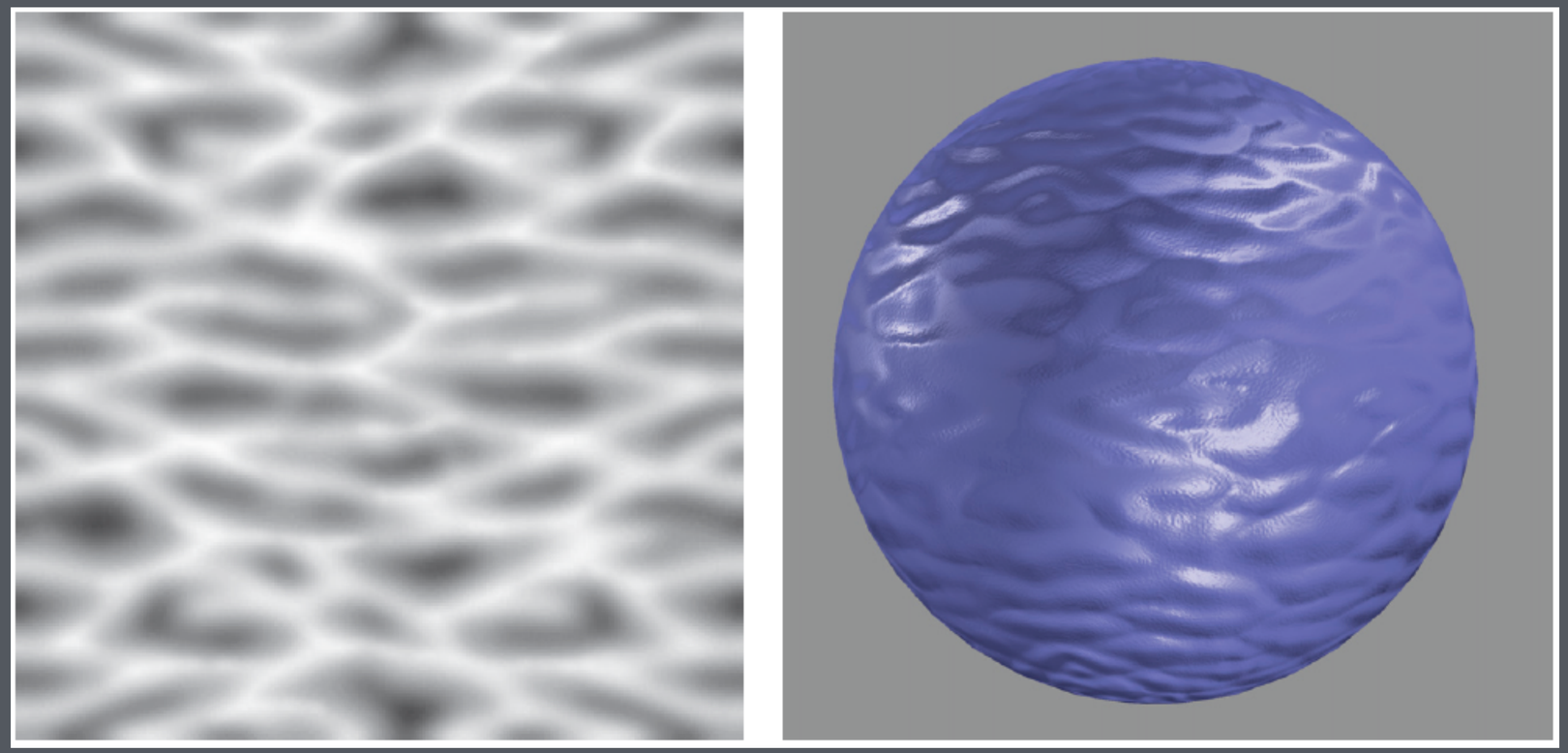
- approx: heights are small compared to radius of curvature (constant normal)
- then the displaced surface is locally a linear transformation of the height field
- normal transforms by the adjoint matrix (as normals always to)
- perform 4 lookups to get 4 neighboring height values
- subtract to obtain finite difference derivatives

Height Field Bump Maps

Older technique, less memory

Texture map value is a height

Gray scale value: light is +, dark is -

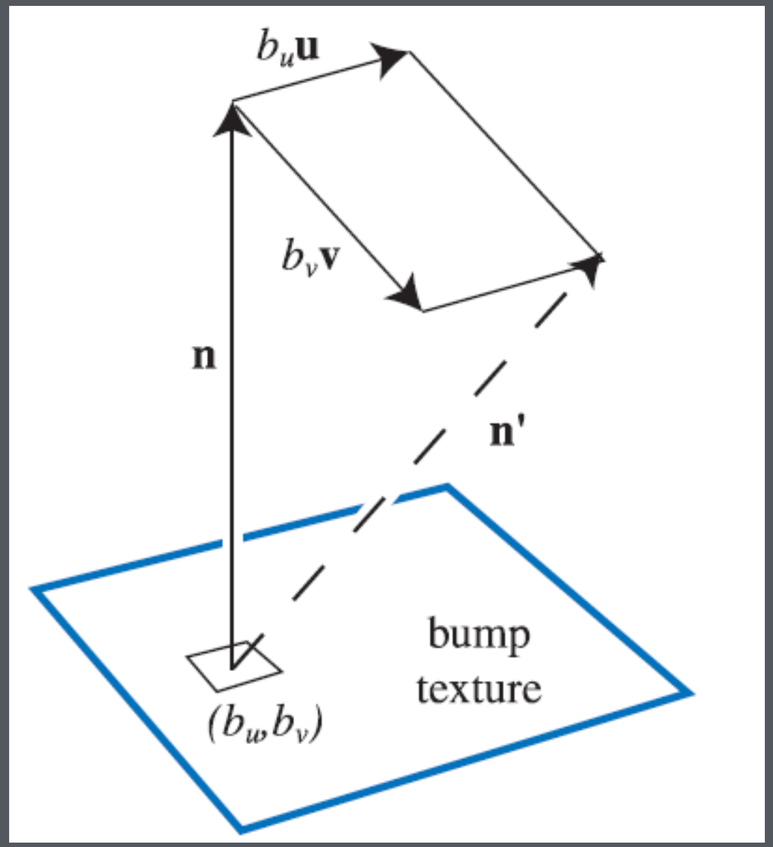


Bump Mapping

Look up bu and bv

N’ is not normalized

N’ = N + bu T + bv B



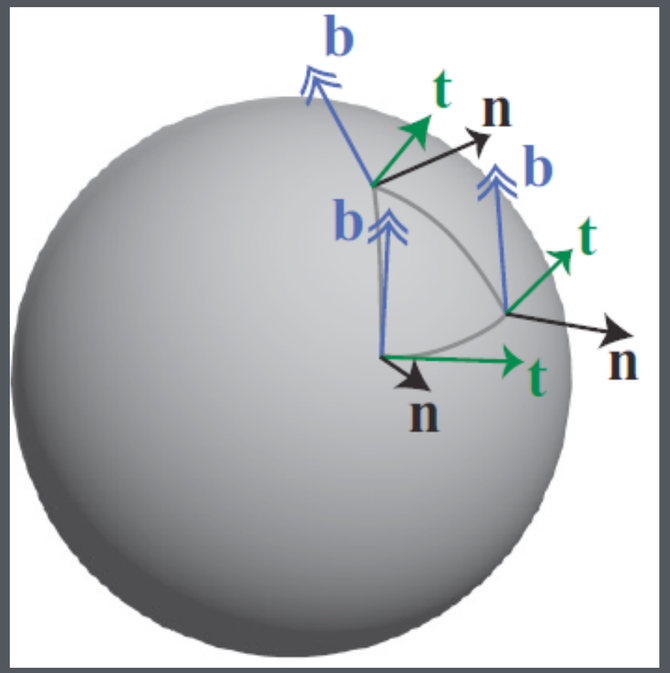
Rendering with Bump Maps

N’.L

Perturb N to get N’ using bump map

Transform L to tangent space of surface

- Have N, T (tangent), bitangent B = T x N



Normal Maps

Preferred technique for bump mapping for modern graphics cards

Store new normals in texture map

- Encodes (x, y, z) mapped to [-1, 1]

More memory but lower computation

Normal Map

Height Map

- Store

```
colorComponent = 0.5 * normalComponent + 0.5
```

-

- Use

```
normalComponent = 2* colorComponent -1
```

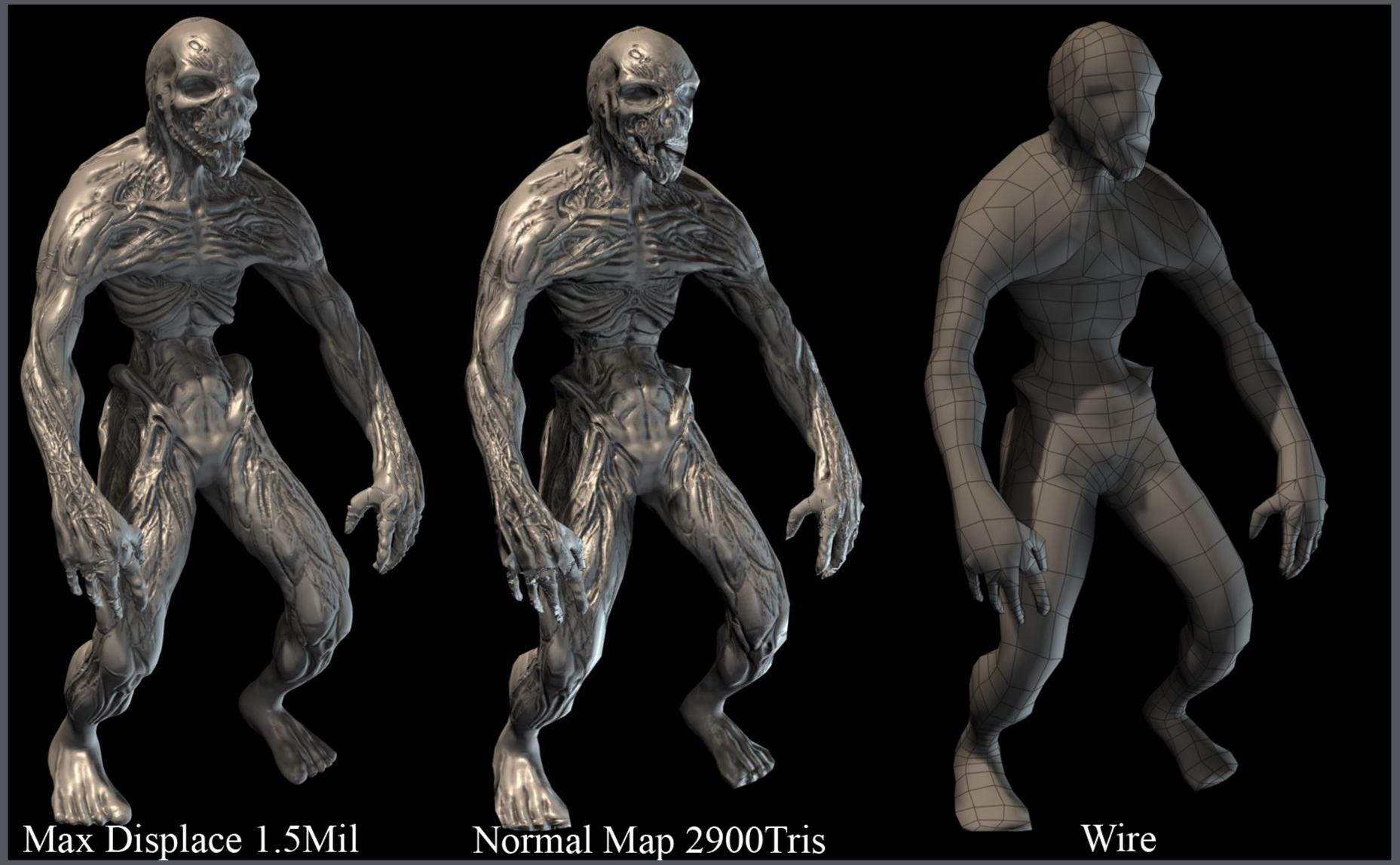
Normal Map

Creating Normal Maps

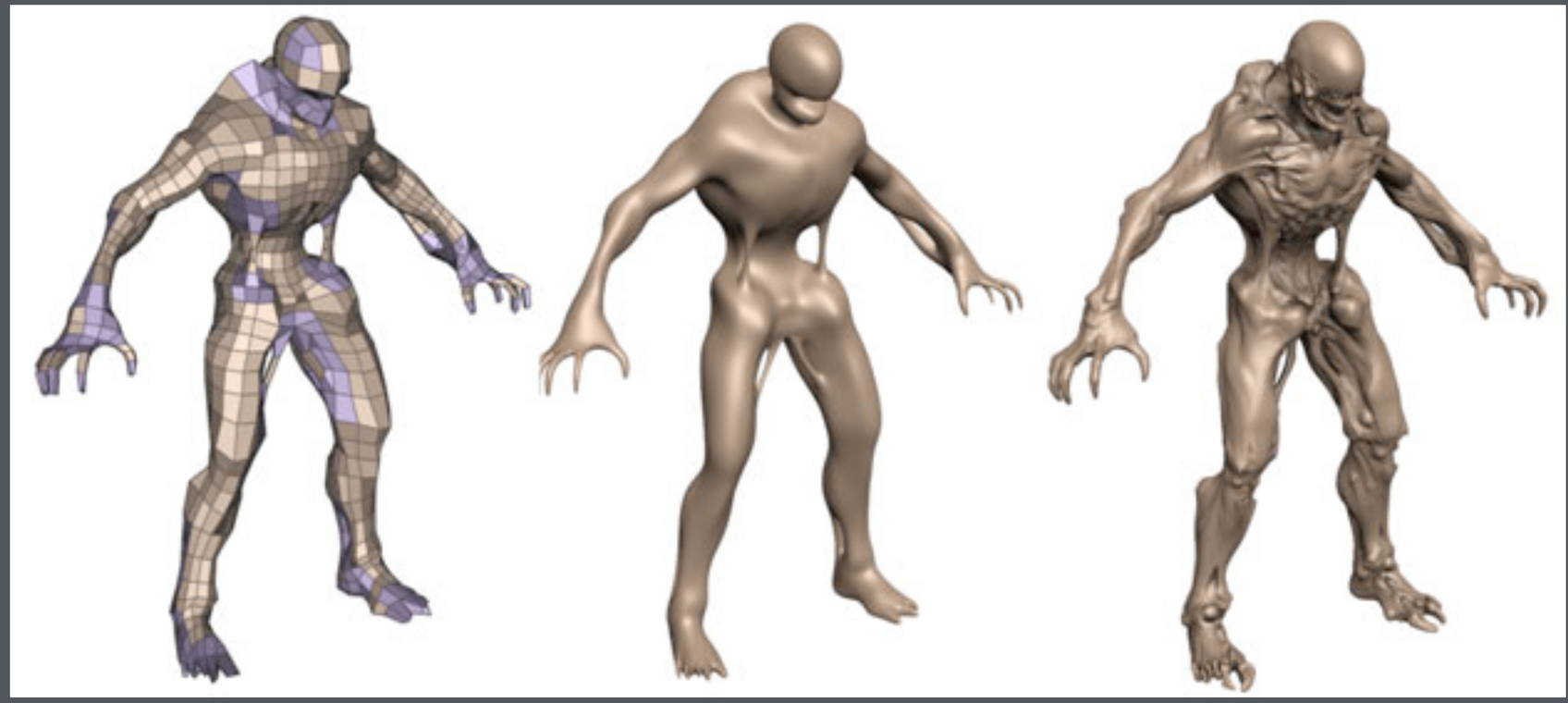
First create complex geometry

Simplify (in modeling time) to simple mesh with normal map

Displacement Maps vs. Normal Maps



Compare with the opposite view

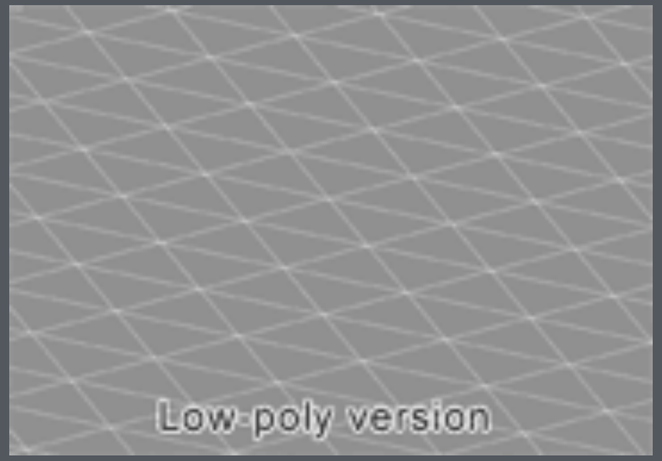


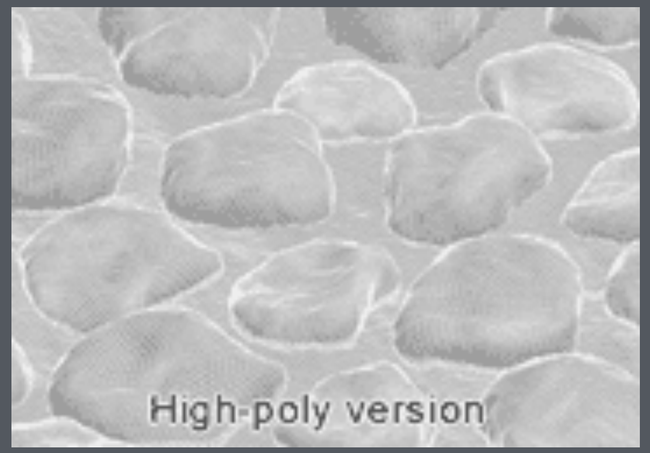
Original

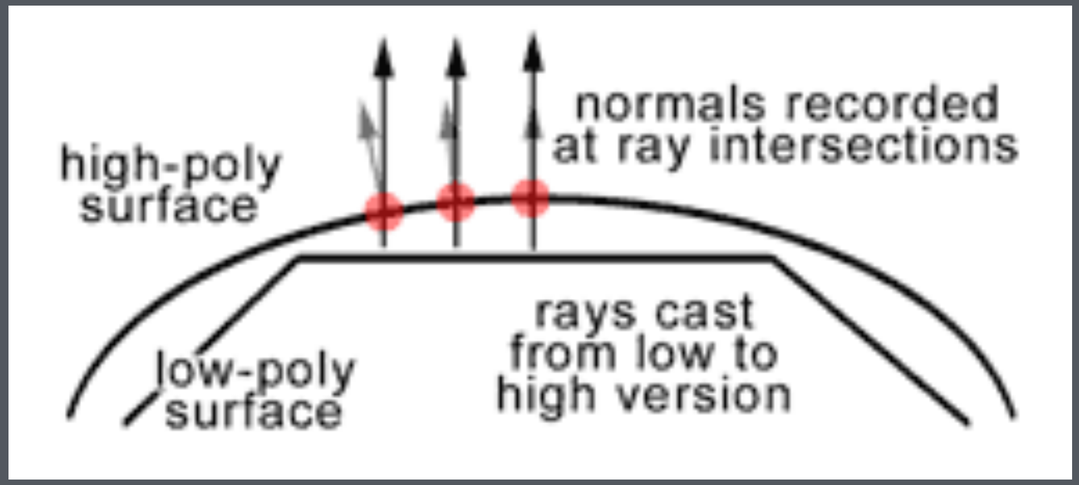
Tesselated

Displacement Mapped

Creating Normal Maps







Which space is normal map in?

World space

- Easy computation, but can’t use the same normal map for…
  - two walls
  - A rotating object

Object space

- Better, but cannot be reused for…
  - deforming objects
  - different objects with similar material

Tangent space

- Can reuse for deforming surfaces
- Transform lighting to this space and shade

Parallax Mapping

Problem with normal mapping

- No self-occlusion
- Supposed to be a height field but never see this occlusion across different viewing angles

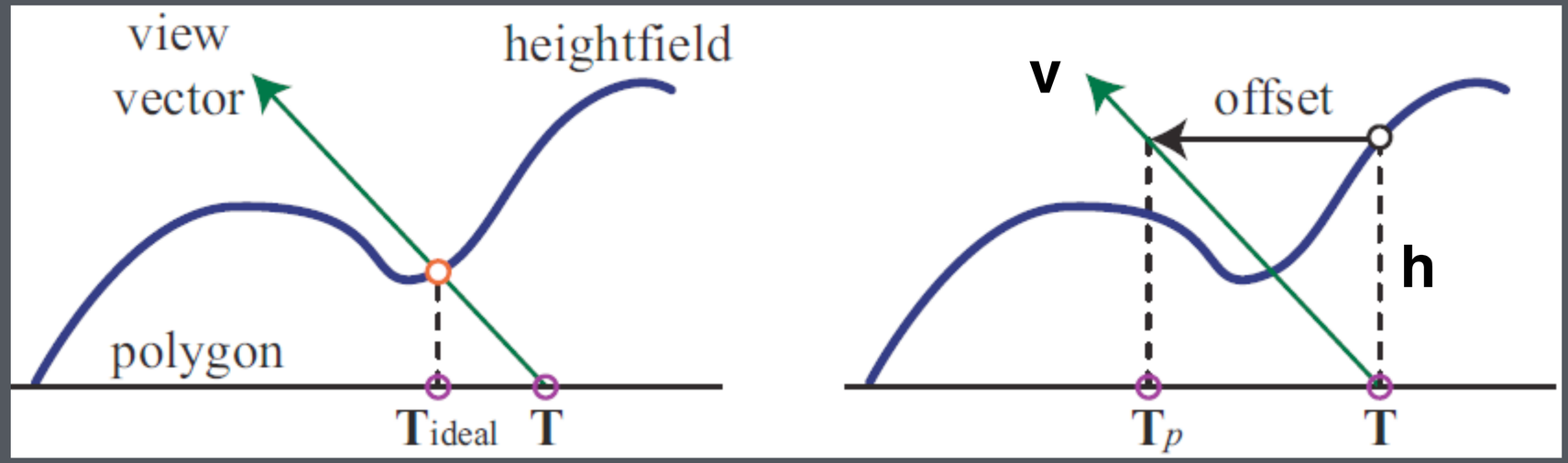
Parallax mapping

- Positions of objects move relative to one other as viewpoint changes

Parallax Mapping

Want T_{ideal}

Use T_p to approximate it



$$\mathbf{T}_p = \mathbf{T} + h\frac{\mathbf{v}_{xy}}{\mathbf{v}_z}$$

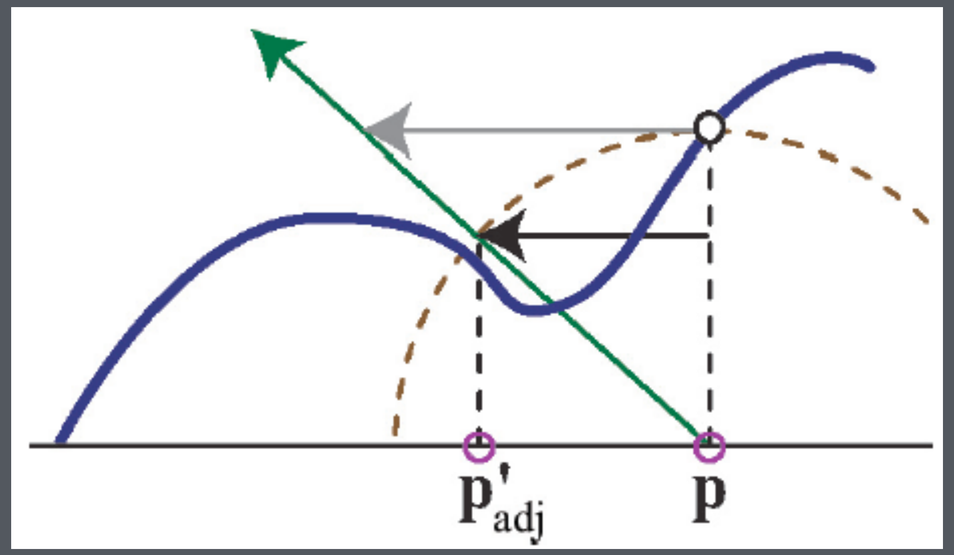
Parallax Offset Limiting

Problem: at steep viewing, can offset too much

Limit offset

- results in a simpler formula

$$\mathbf{p}_{\text{adj}} = \mathbf{p} + h\mathbf{v}_{xy}$$



Parallax Offset Limiting

Widely used in games

- the standard in simple bump mapping

Normal Mapping

Parallax Mapping Offset Limiting

1,100 polygon object w/
parallax occlusion mapping

1.5 million polygon

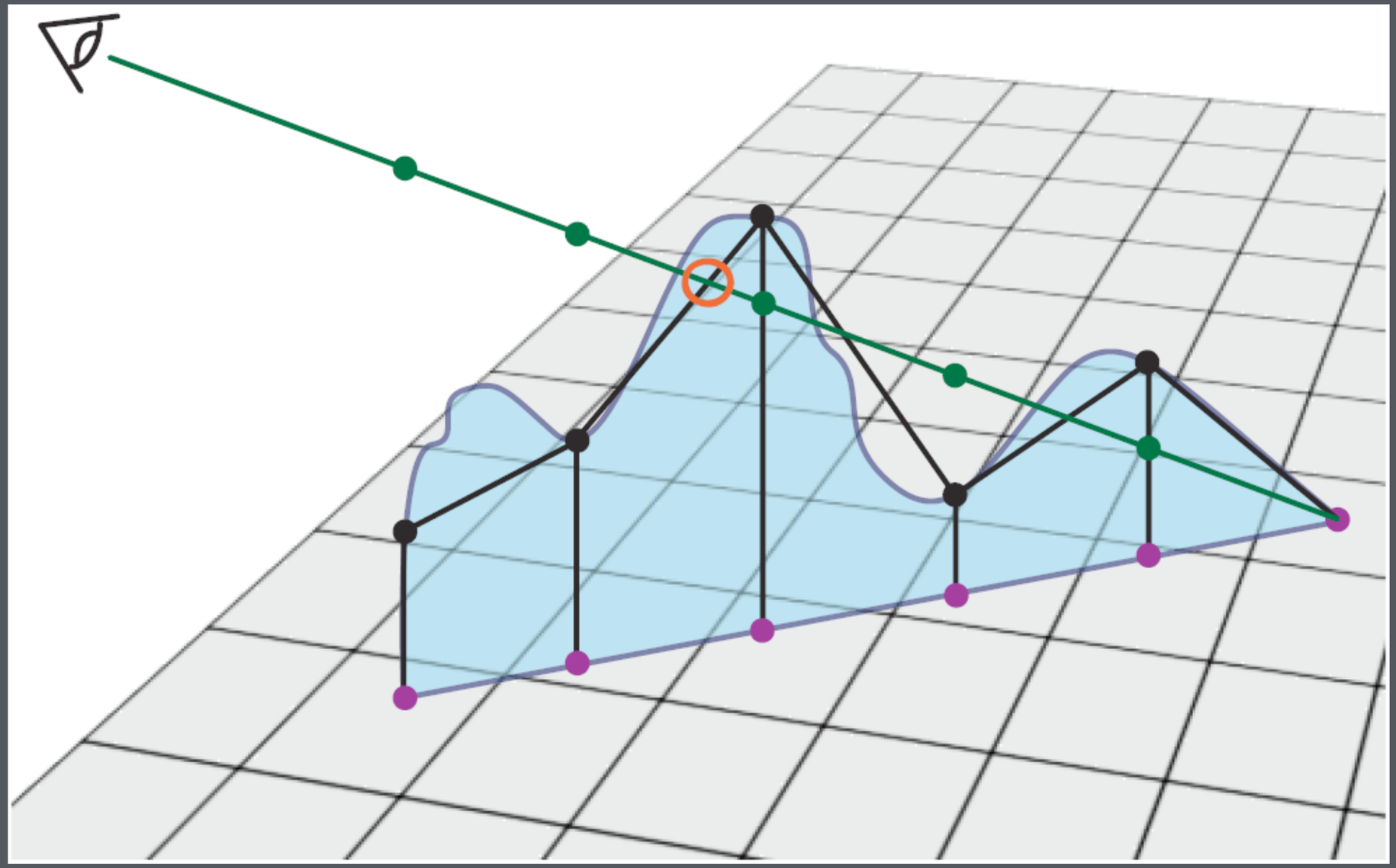
Relief Mapping

Aka Parallax occlusion mapping, relief mapping, steep parallax mapping

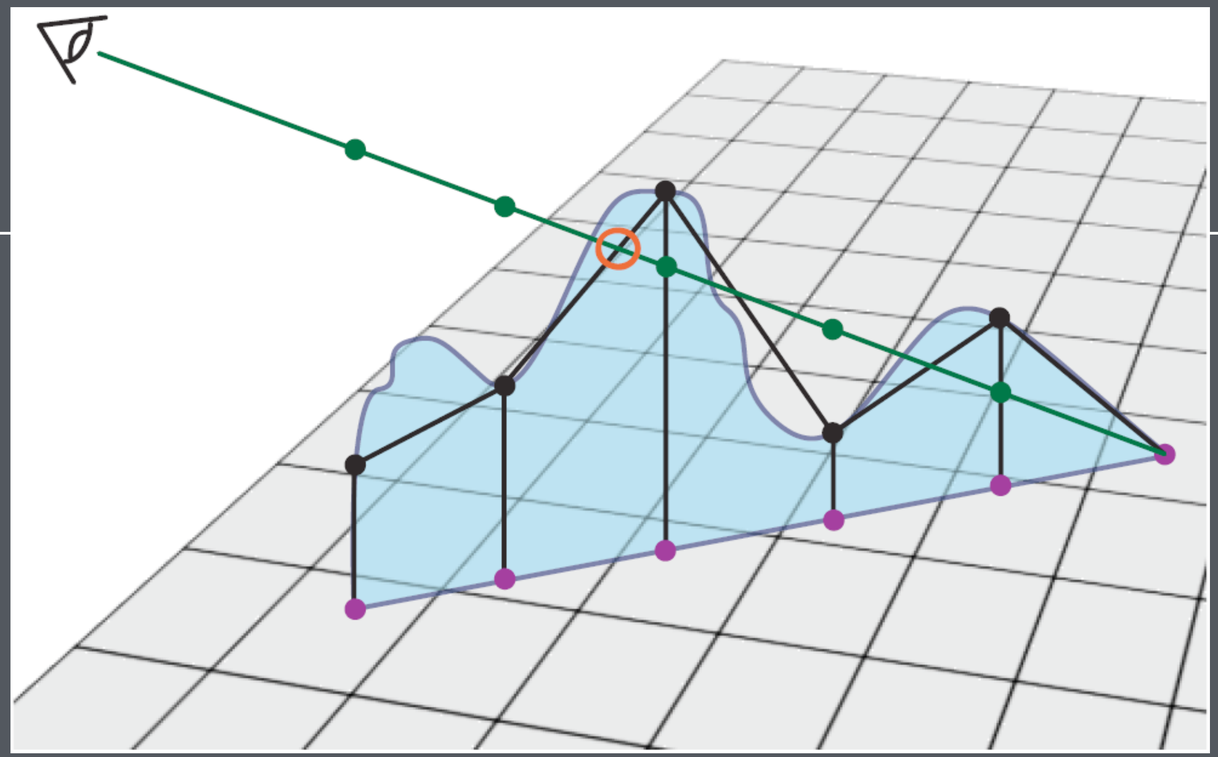
Tries to find where the view ray intersects the height field

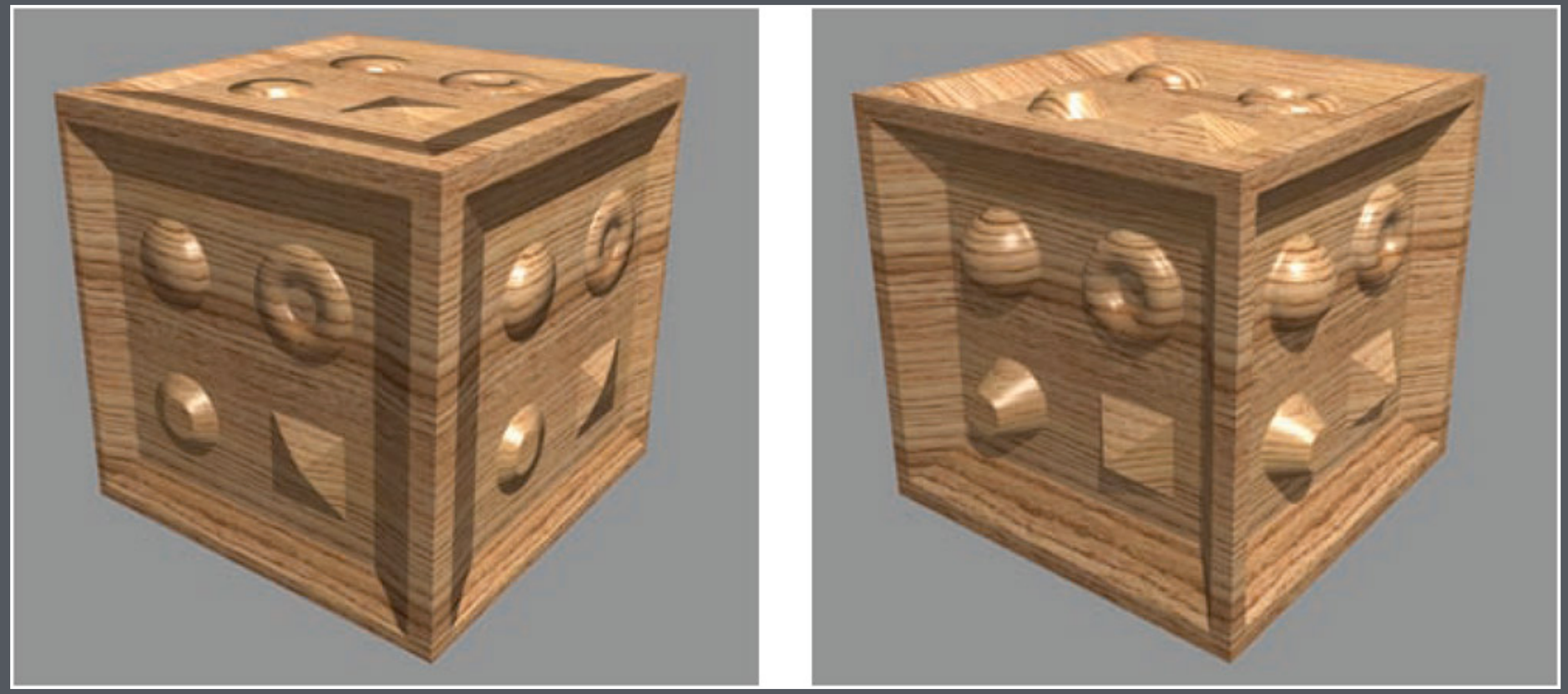
- Kinda

Relief Mapping



```
Sample along ray (green points)
Lookup violet points (texture values)
   /* Inferthe black line shape */
Compare green points with black points
Find intersect between two conditions
   prev: green above black
   next: green below black
```





Parallax Mapping

Relief Mapping

http://www.youtube.com/watch?v=5gorm90TXJM

Crysis, Crytek