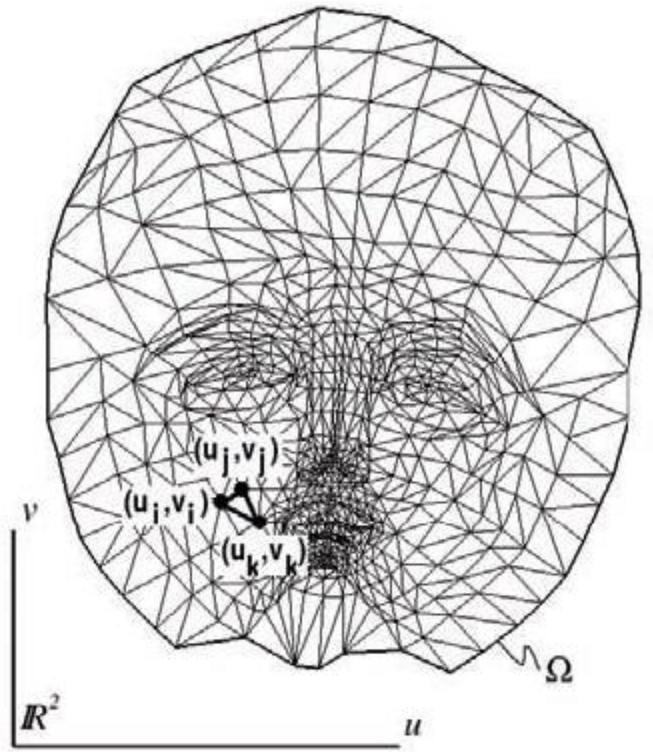
TEXTURING

SEMINAR 9

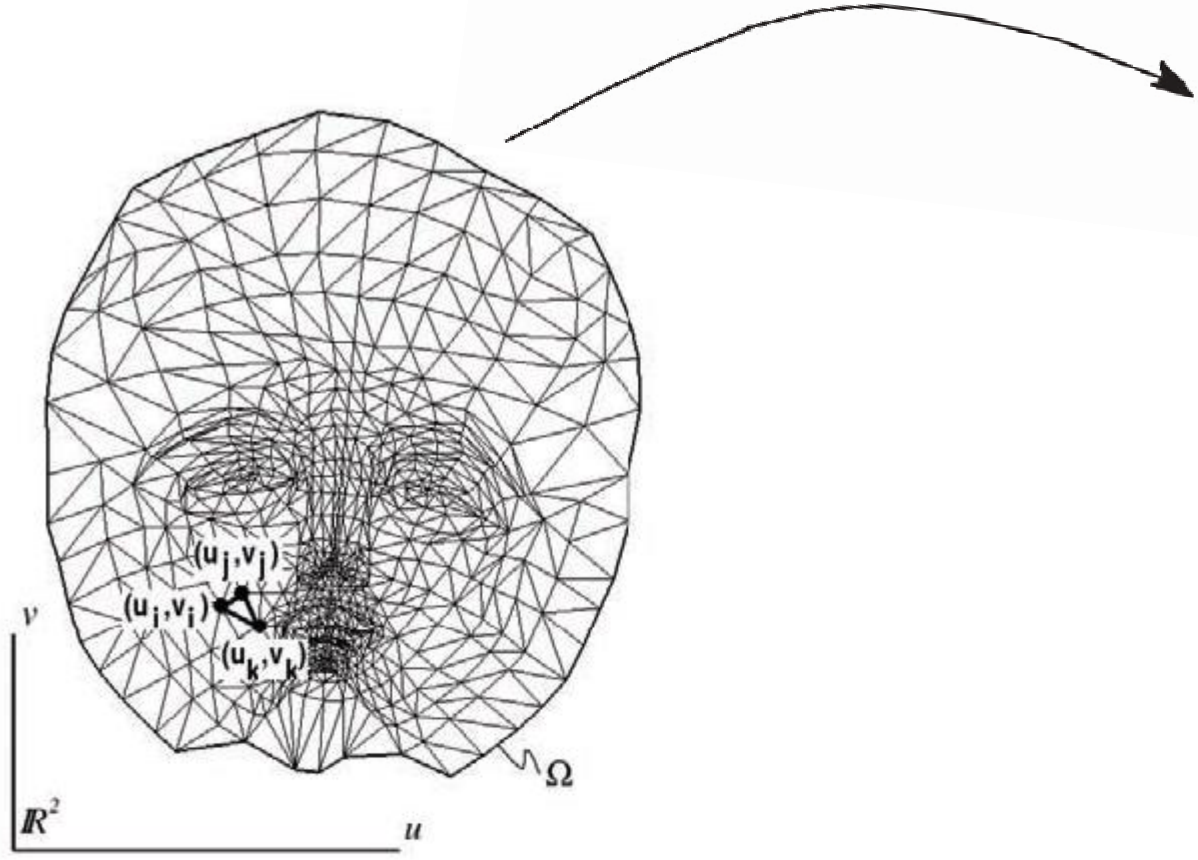
Computer Graphics 2

Parametric surface (1)

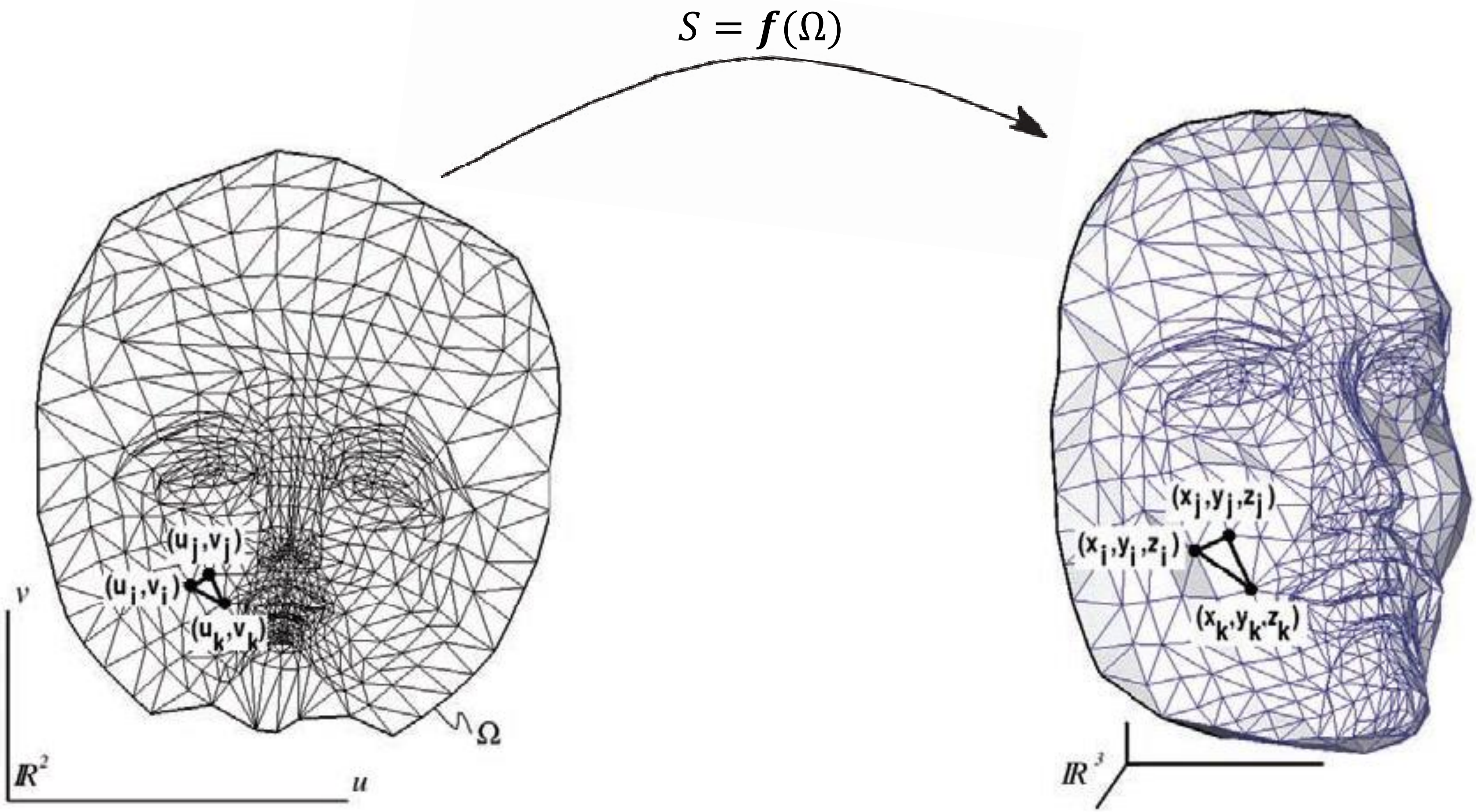


Parametric surface (2)

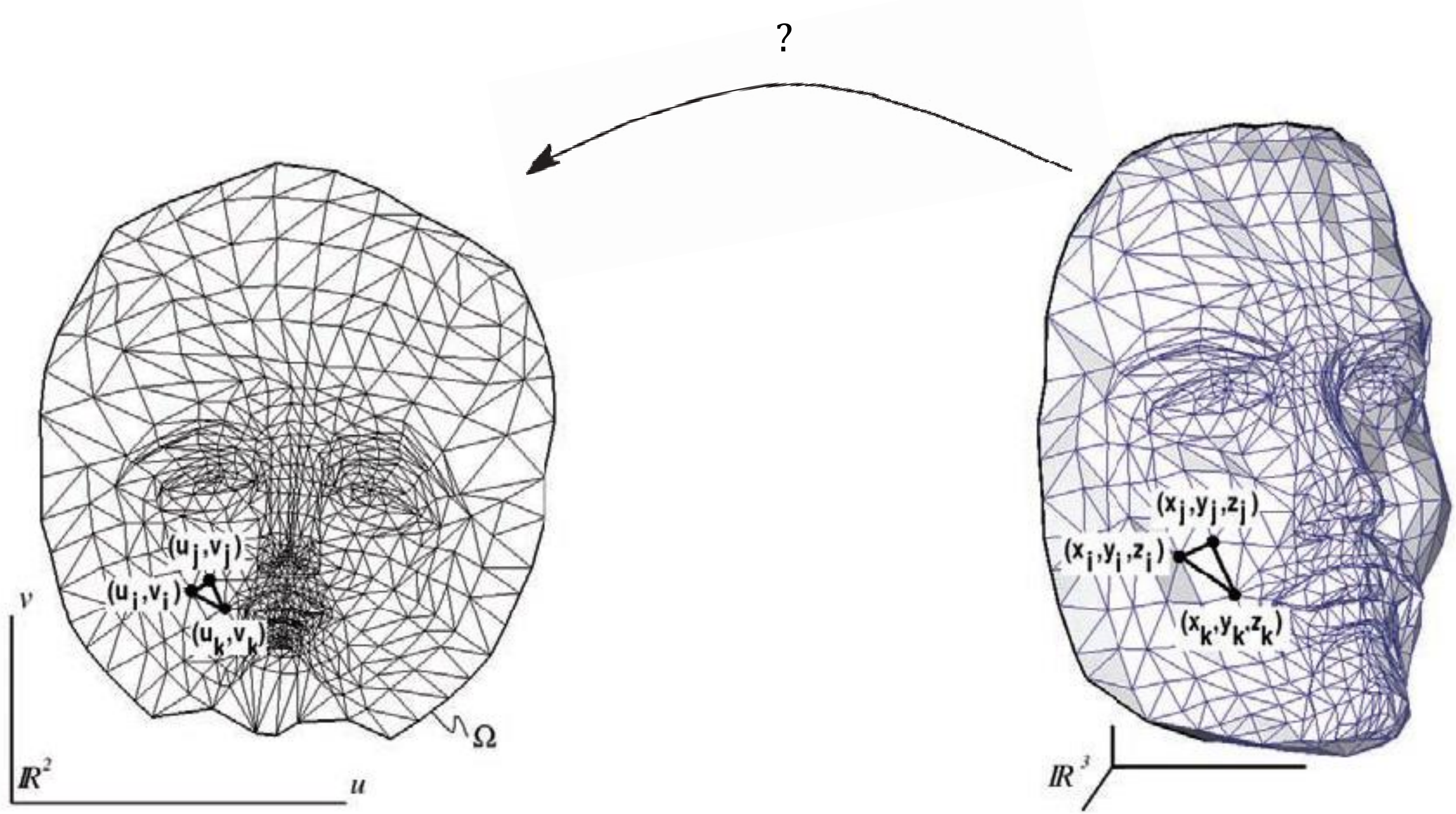
$S = \boldsymbol{f}(\Omega)$



Parametric surface (3)



Surface parameterization



Bump mapping (1)

- Simulates bumps and wrinkles
- Achieved by perturbing surface normal
  - Objects appear more complex

$$\mathbf{n} = \frac{\mathbf{P_u} \times \mathbf{P_v}}{|\mathbf{P_u} \times \mathbf{P_v}|}$$

$$\mathbf{P_u} = \left(\frac{\partial x}{\partial u}, \frac{\partial y}{\partial u}, \frac{\partial z}{\partial u}\right)$$

Bump mapping (2)

- Simulates bumps and wrinkles
- Achieved by perturbing surface normal
  - Objects appear more complex

$$\mathbf{n} = \frac{\mathbf{P_u} \times \mathbf{P_v}}{|\mathbf{P_u} \times \mathbf{P_v}|}$$

$$d(u,v) : \mathbf{P}' = \mathbf{P} + d(u,v)\mathbf{n}$$

$$\mathbf{P_u} = \left(\frac{\partial x}{\partial u}, \frac{\partial y}{\partial u}, \frac{\partial z}{\partial u}\right)$$

Bump mapping (3)

- Simulates bumps and wrinkles
- Achieved by perturbing surface normal
  - Objects appear more complex

$$\mathbf{n} = \frac{\mathbf{P_u} \times \mathbf{P_v}}{|\mathbf{P_u} \times \mathbf{P_v}|}$$

$$d(u,v) : \mathbf{P}' = \mathbf{P} + d(u,v)\mathbf{n}$$

$$\mathbf{p}'_\mathbf{u} = \mathbf{p_u} + \frac{\partial d}{\partial u}\mathbf{n} + d(u,v)\mathbf{n_u}$$

$$\mathbf{p}'_\mathbf{v} = \mathbf{p_v} + \frac{\partial d}{\partial v}\mathbf{n} + d(u,v)\mathbf{n_v}$$

$$\mathbf{P_u} = \left(\frac{\partial x}{\partial u}, \frac{\partial y}{\partial u}, \frac{\partial z}{\partial u}\right)$$

$$\mathbf{n} \times \mathbf{n} = 0$$

Bump mapping (4)

- Simulates bumps and wrinkles
- Achieved by perturbing surface normal
  - Objects appear more complex

$$\mathbf{n} = \frac{\mathbf{P_u} \times \mathbf{P_v}}{|\mathbf{P_u} \times \mathbf{P_v}|}$$

$$d(u,v) : \mathbf{P'} = \mathbf{P} + d(u,v)\mathbf{n}$$

$$\mathbf{p'_u} = \mathbf{p_u} + \frac{\partial d}{\partial u}\mathbf{n} + d(u,v)\mathbf{n_u}$$

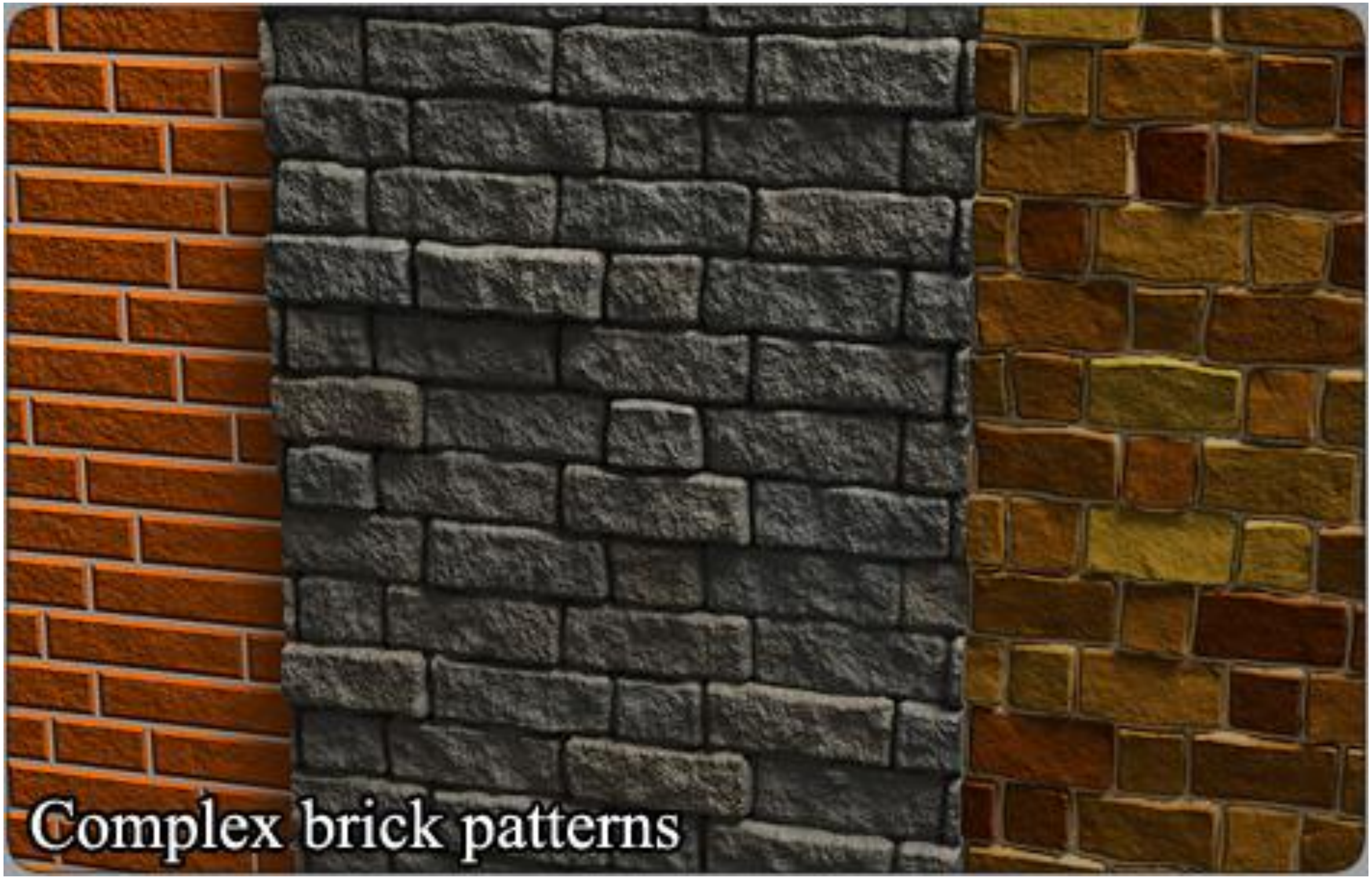
$$\mathbf{p'_v} = \mathbf{p_v} + \frac{\partial d}{\partial v}\mathbf{n} + d(u,v)\mathbf{n_v}$$

$$\mathbf{n'} = \mathbf{n} + \frac{\partial d}{\partial u}\mathbf{n} \times \mathbf{p_v} + \frac{\partial d}{\partial v}\mathbf{n} + d(u,v)\mathbf{n_v} \times \mathbf{p_u}$$

$$\mathbf{P_u} = \left(\frac{\partial x}{\partial u}, \frac{\partial y}{\partial u}, \frac{\partial z}{\partial u}\right)$$

$$\mathbf{n} \times \mathbf{n} = 0$$

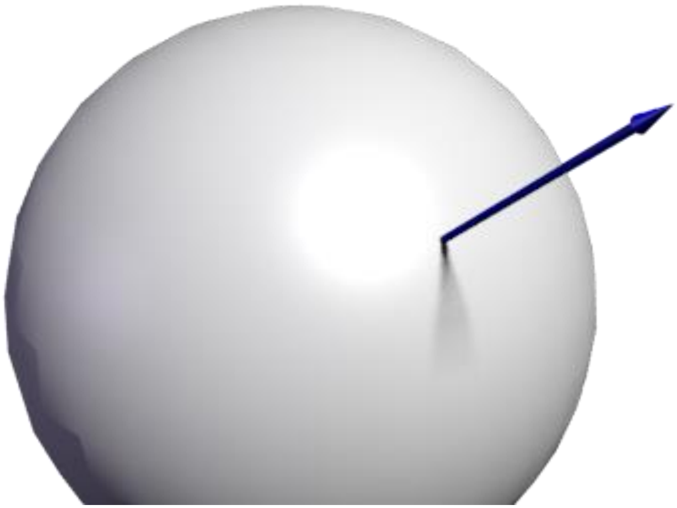
Bump mapping example



Normal mapping

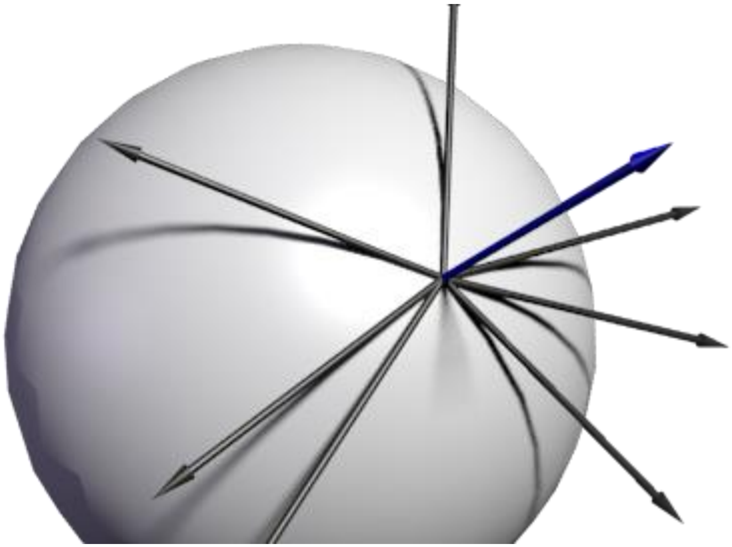
- Normal is directly stored in texture
  - Each component between $[0,1]$ should change to $[-1,1]$
  - To avoid problems with different models normal is stored in tangent space (TBN)
    - In practice light computation is converted to TBN
    - At exercise normal is converted to global coordinates ☺

TBN calculation



normal at point: $\boldsymbol{n} = known$

TBN calculation

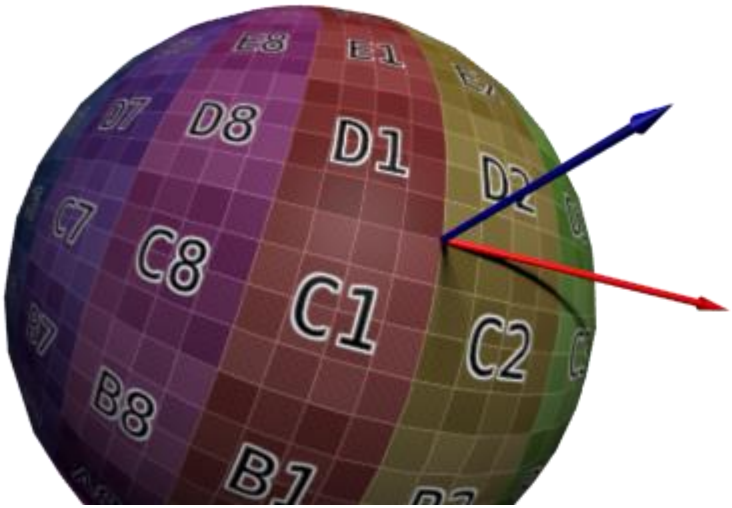


normal at point: $\boldsymbol{n} = known$

tangent at point: $\mathbf{t} = ?$

bitangent at point: $\mathbf{b} = ?$

TBN calculation

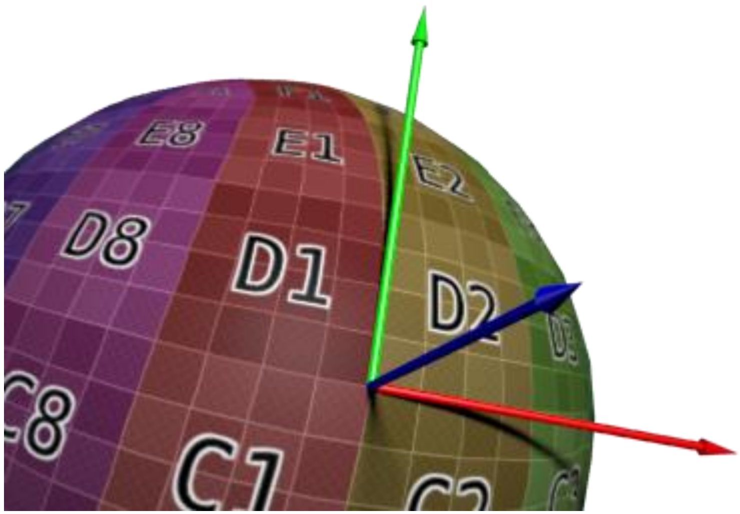


$\boldsymbol{n} = known \quad \boldsymbol{up} = (0,0,1)$

$$\mathbf{t} = \frac{\boldsymbol{n} \times \boldsymbol{up}}{|\boldsymbol{n} \times \boldsymbol{up}|}$$

$$\boldsymbol{b} = \frac{\boldsymbol{t} \times \boldsymbol{n}}{|\boldsymbol{t} \times \boldsymbol{n}|}$$

TBN calculation



$\boldsymbol{n} = known \quad \boldsymbol{up} = (0,0,1)$

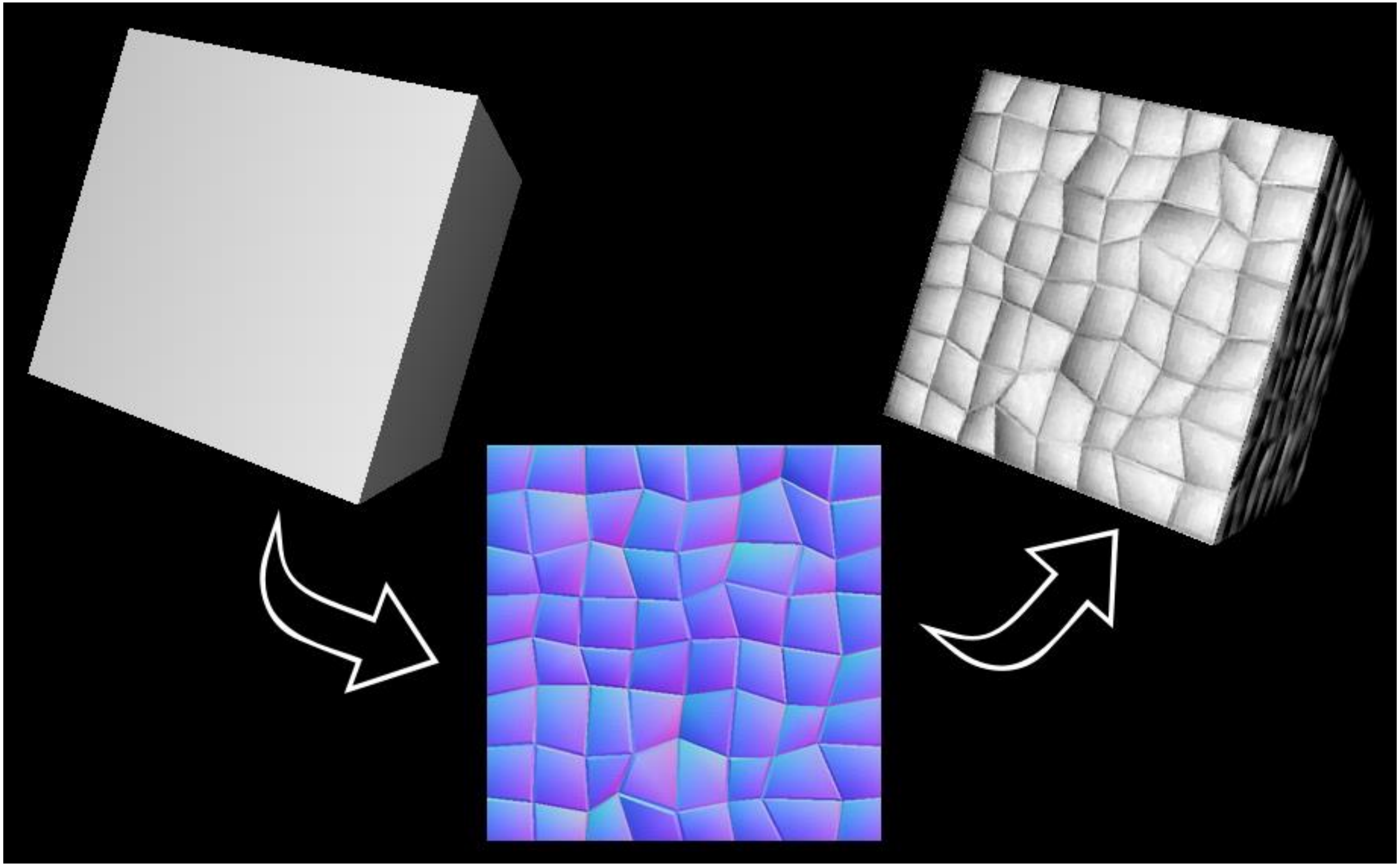
$\mathbf{t} = \frac{\boldsymbol{n} \times \boldsymbol{up}}{|\boldsymbol{n} \times \boldsymbol{up}|}$

$\boldsymbol{b} = \frac{\boldsymbol{t} \times \boldsymbol{n}}{|\boldsymbol{t} \times \boldsymbol{n}|}$

$\boldsymbol{TBN} = [\boldsymbol{t}, \boldsymbol{b}, \boldsymbol{n}]$

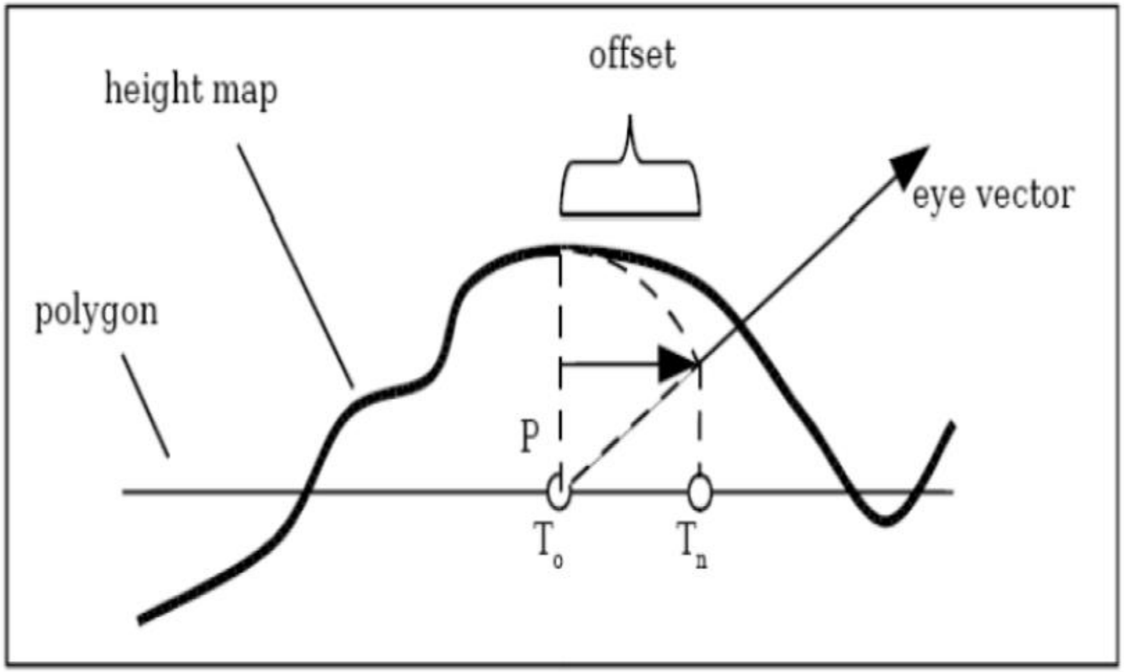
$\boldsymbol{n}' = \boldsymbol{TBN} * NormalMap(u, v)$

Normal mapping example



Parallax mapping

- Displaces texture coordinates at a point by a function of view angle (in tangent space) and a height map
- At steeper values texture is displaced more giving the illusion of depth

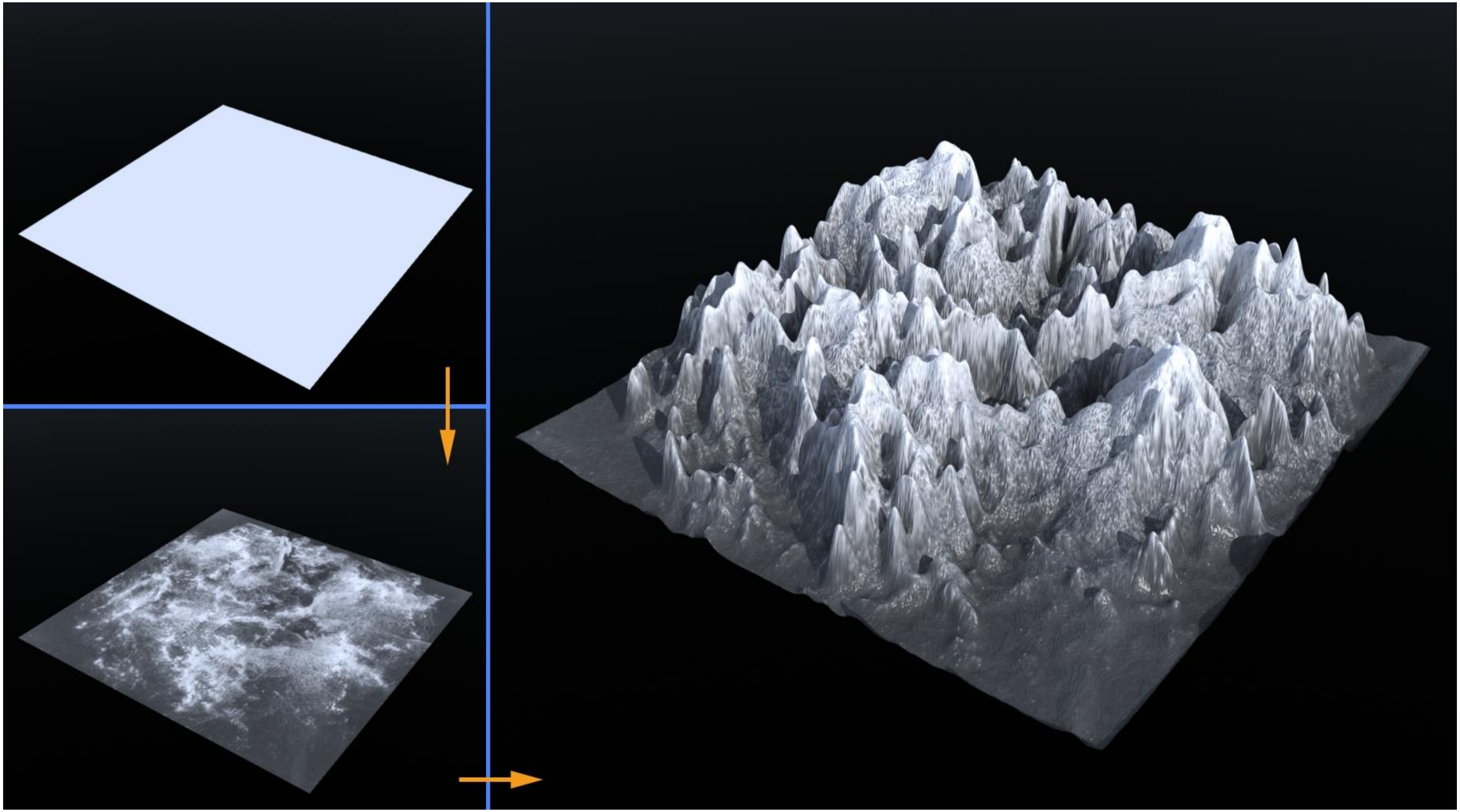


Parallax mapping example

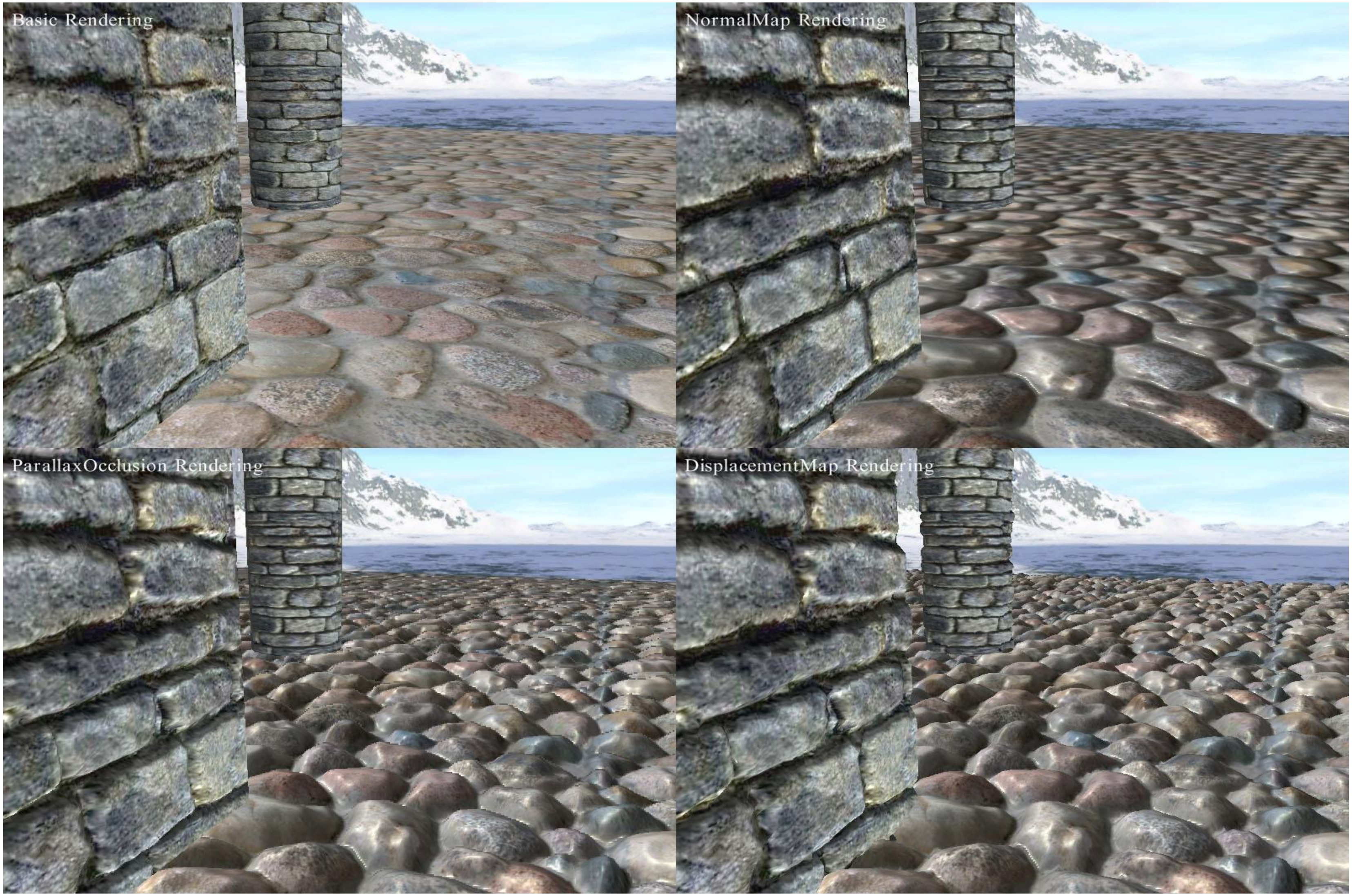
Displacement mapping

- Changes actual geometric position of vertices
  - $v' = v + DisplacementMap(u, v) * \boldsymbol{n}$
- Usually coupled with a subdivision step
  - Surface is tessellated on the GPU
  - New vertex positions are calculated with displacement
- From all presented techniques only displacement mapping changes positions of vertices
  - Therefore only displacement mapping alters object boundary

Displacement mapping example

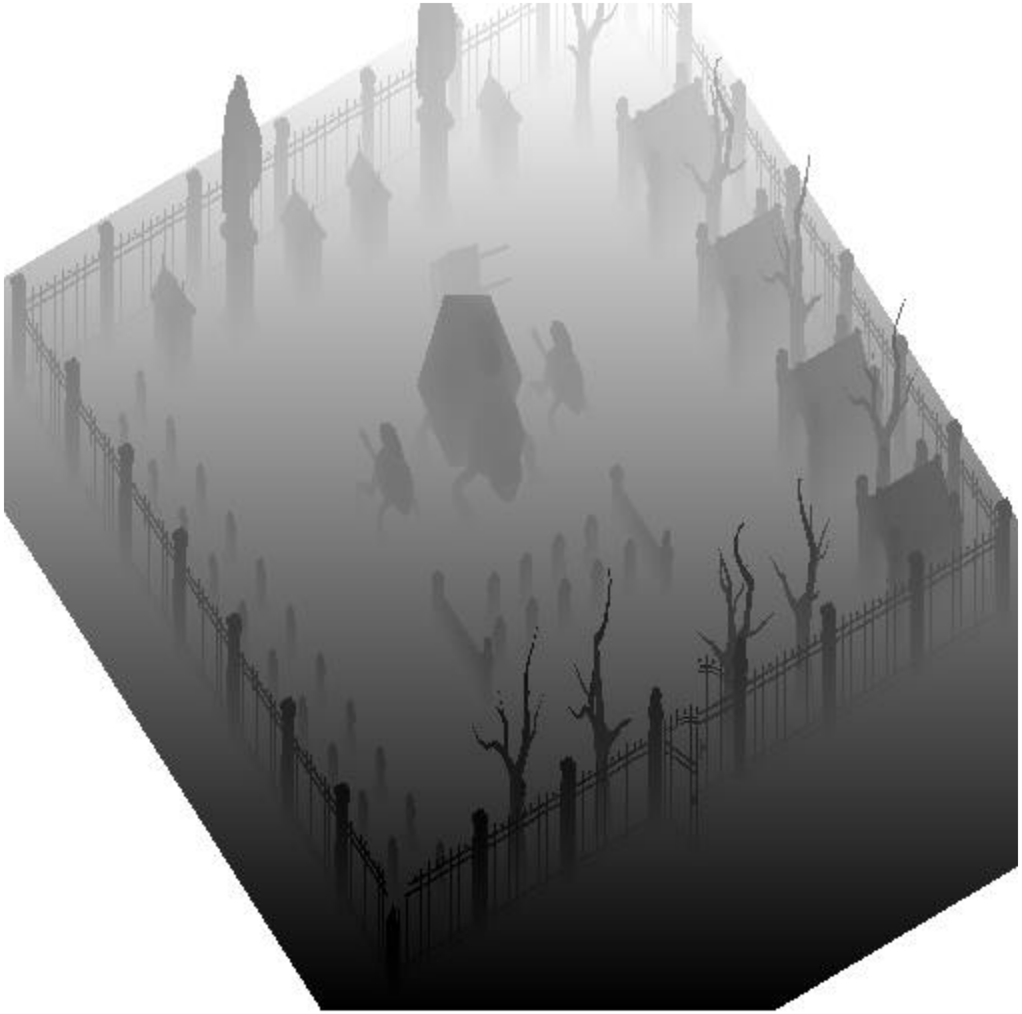


Normal vs. Parallax vs. Displacement

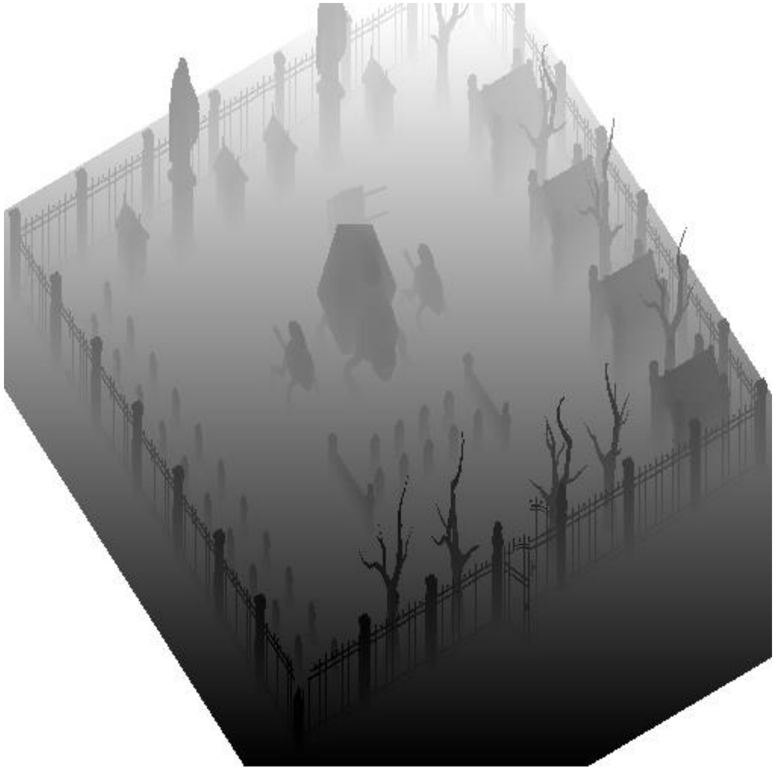


Shadow mapping

Shadow mapping (1)



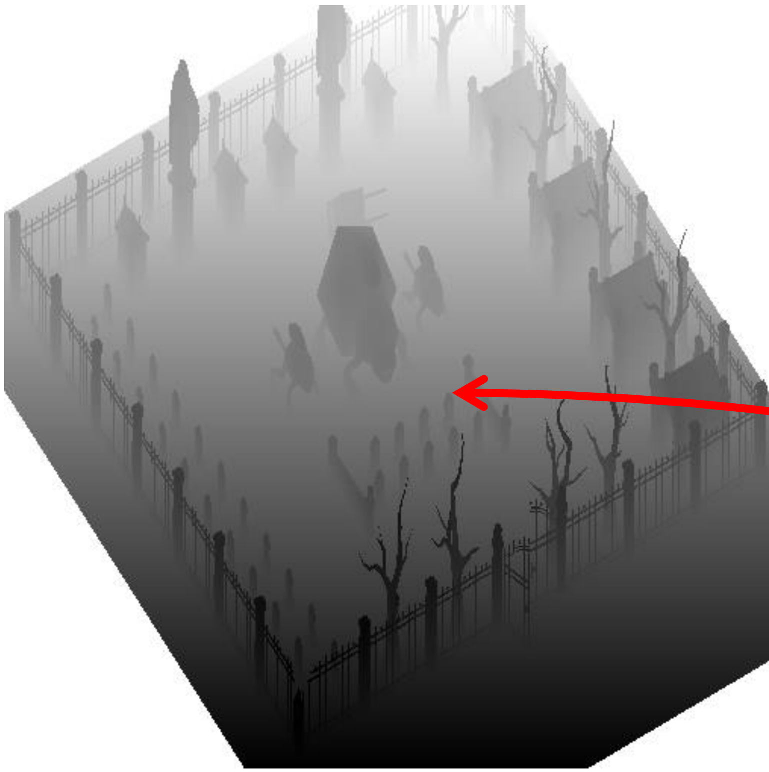
Shadow mapping (2)



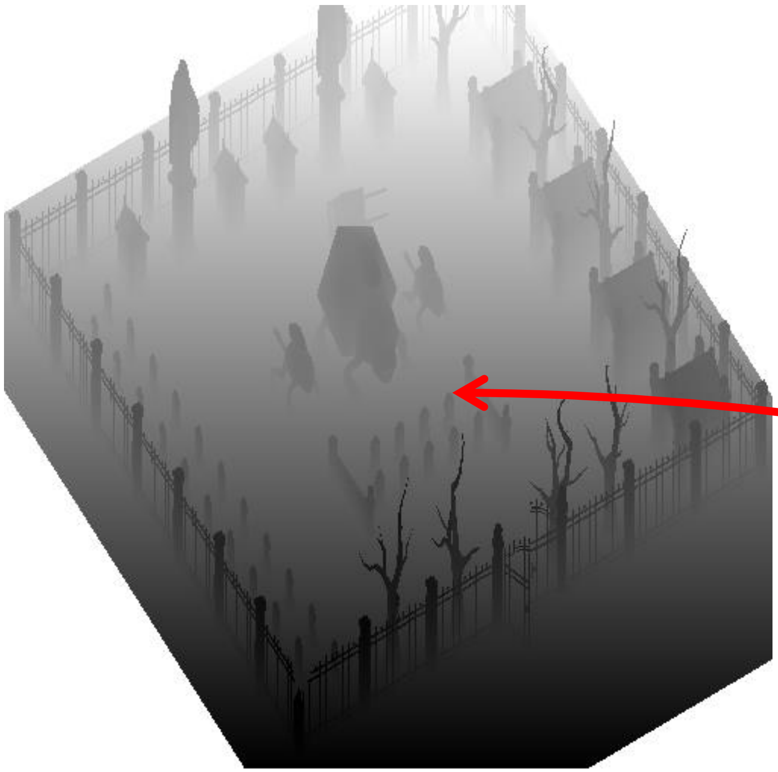
Shadow mapping (3)

$$light = LightMVP * vertex$$

OpenGL [-1,1] <> texture [0,1]

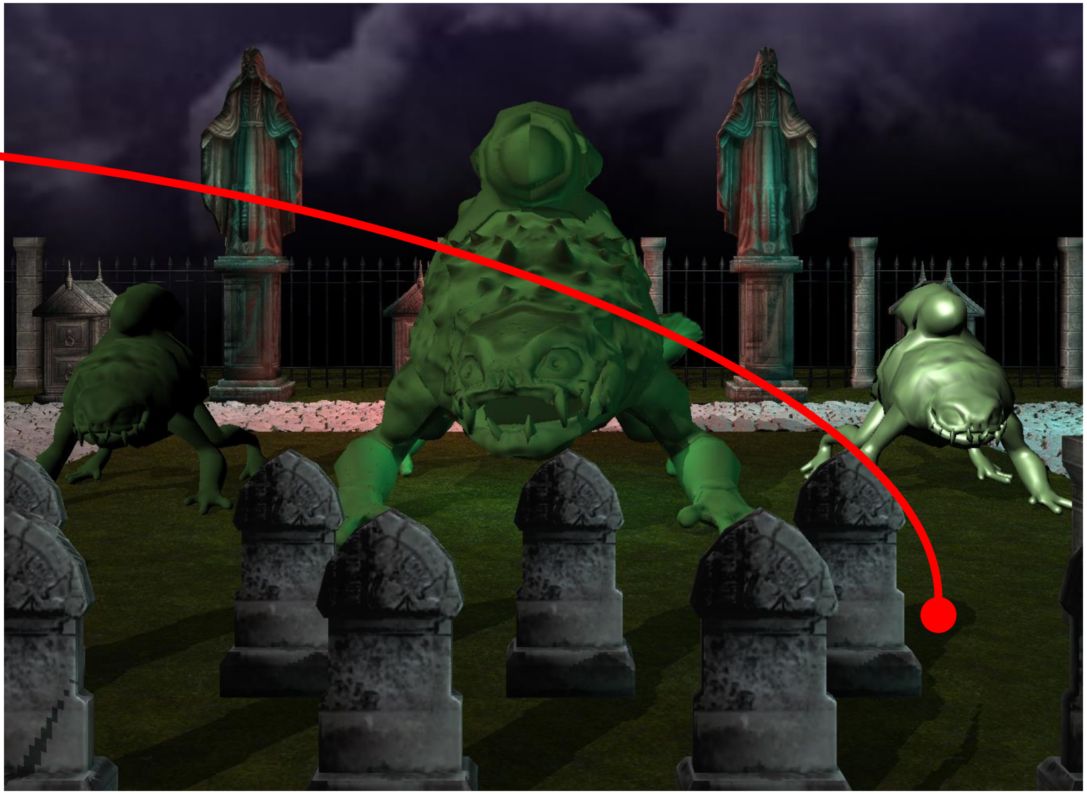


Shadow mapping (4)



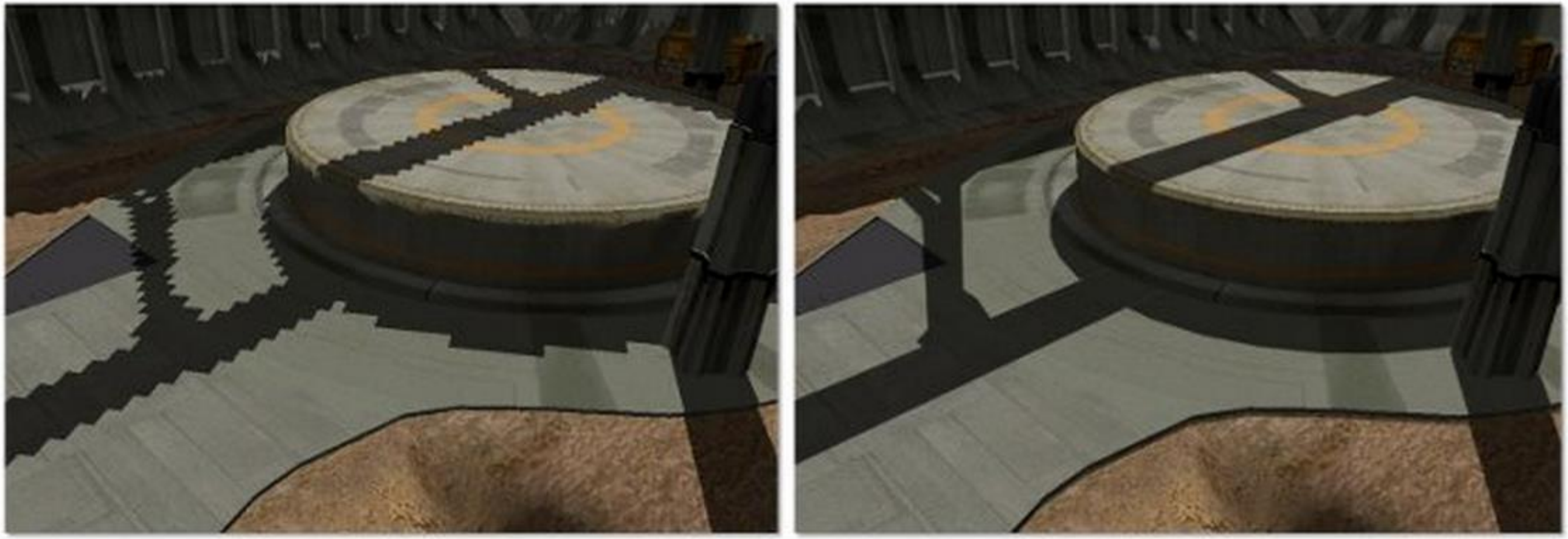
$$light = LightMVP * vertex$$ OpenGL [-1,1] <> texture [0,1]

$$shadowmap(light_x, light_y) < light_z / light_w$$



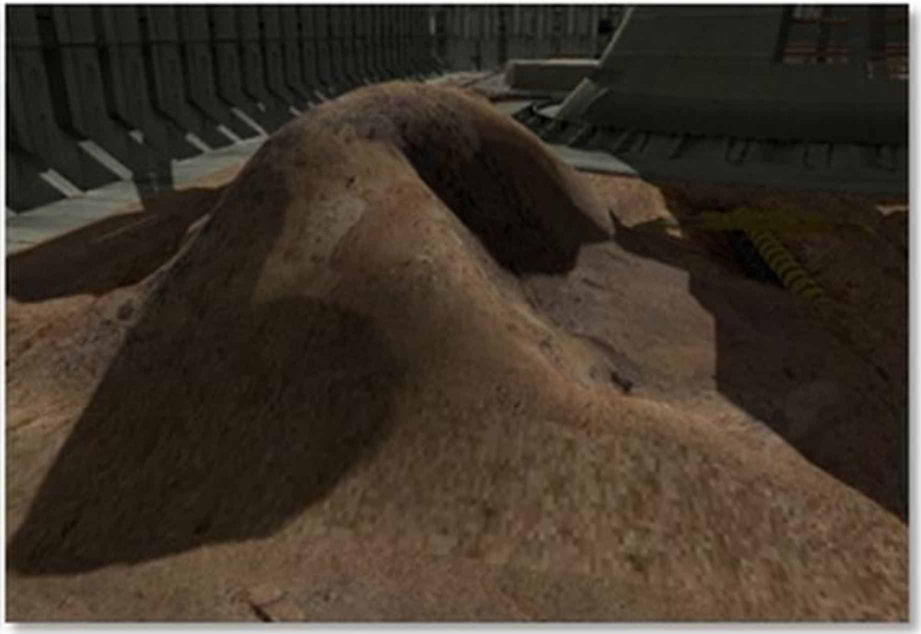
Perspective aliasing

- Pixels in view space are not in 1:1 ratio with texels in the shadow map
- Pixels in near plane are closer and require higher resolution
- With too high resolution shadows of small object disappear



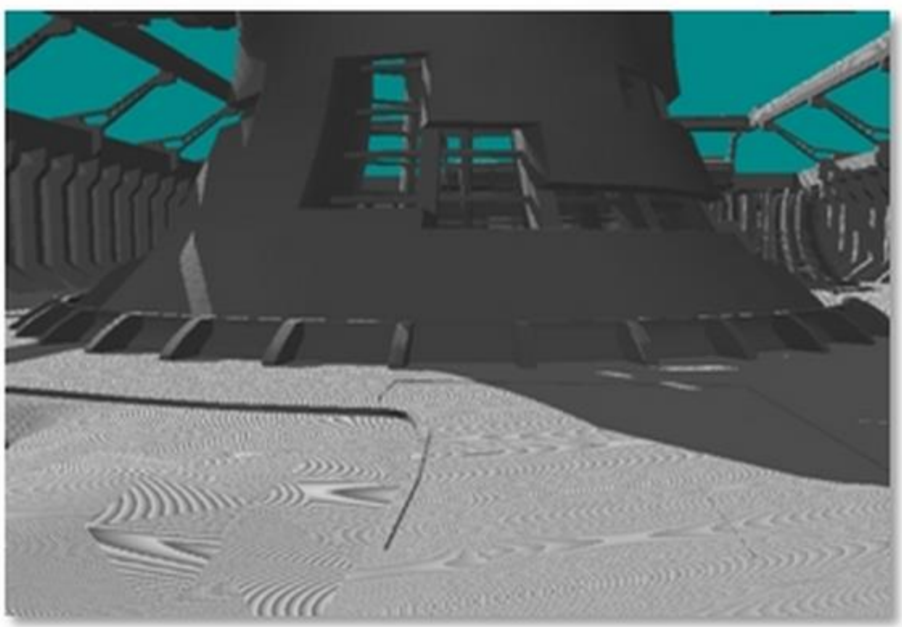
Projective aliasing

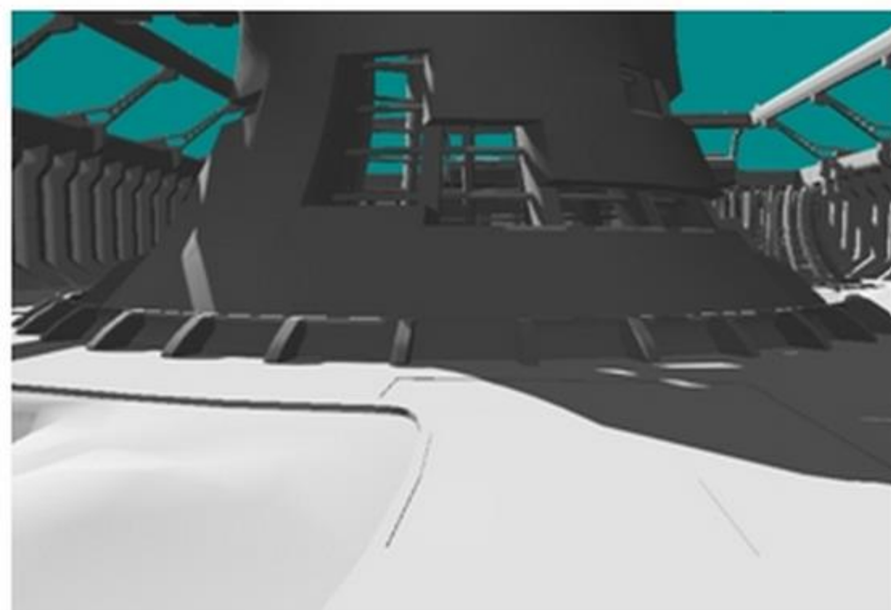
- Texels in camera space to texels in eye space are not in 1:1 ratio
- Occurs when surface normal is orthogonal to the light
- Caused by orientation of geometry with respect to the light



Shadow acne

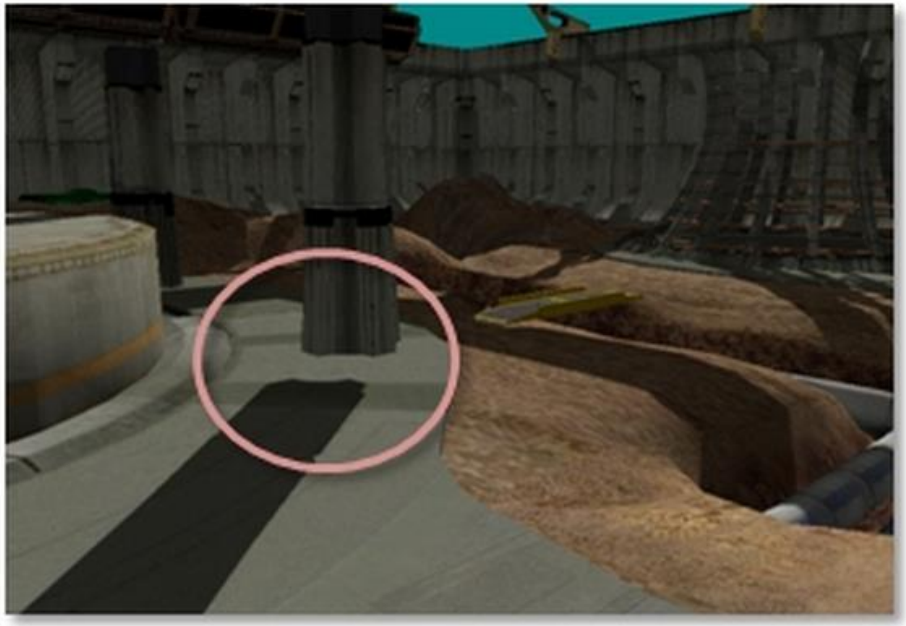
- Shadow map quantizes depth over an entire texel
- When shader compares values self-shadowing occurs
- Can be also caused by precision errors

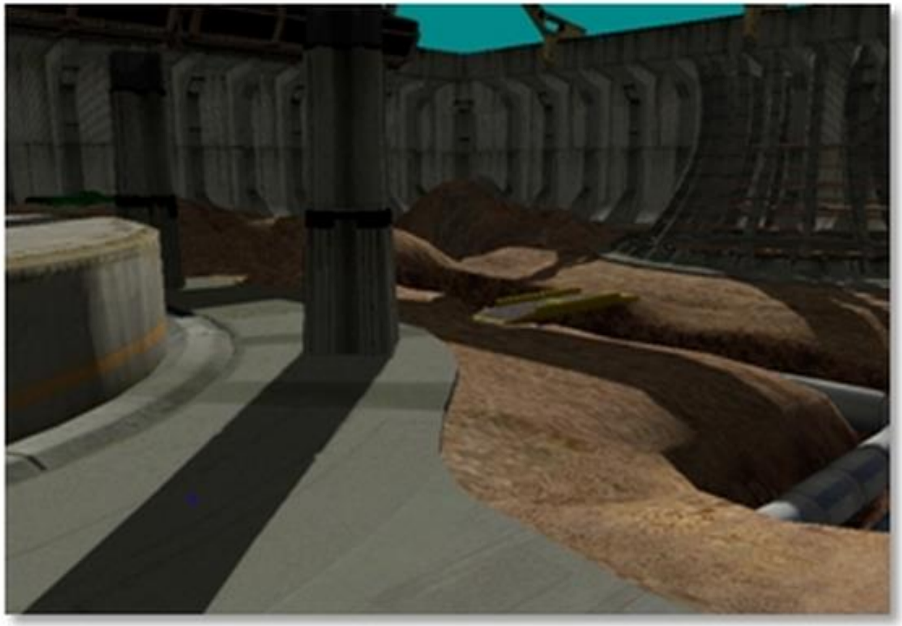




Peter panning

- Peter Pan got detached from his shadow and could fly
- Makes objects appear to float above the surface





Questions?