Geometry and Textures II

Prof. Dr. Markus Gross

ETH zürich

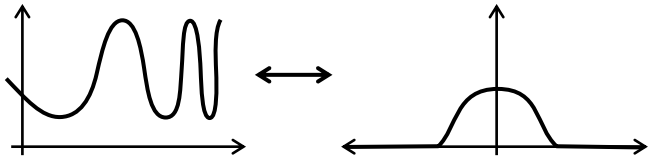
cgl

Texture Filtering

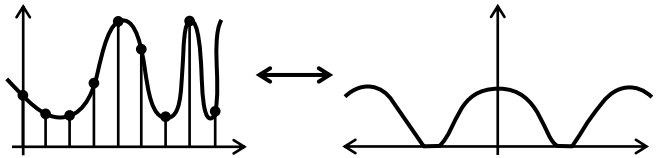
- Sampling and reconstruction

Sampling

Spatial Domain Frequency Domain



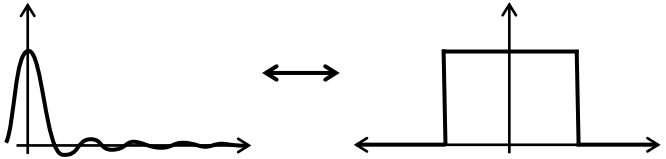
Original function



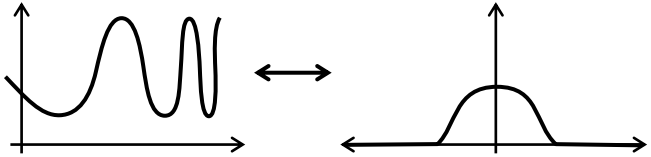
Sampled function

Reconstruction

Spatial Domain Frequency Domain



Filter function



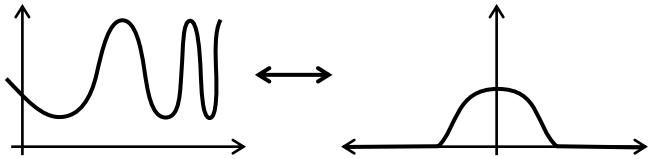
Reconstructed function

Texture Filtering

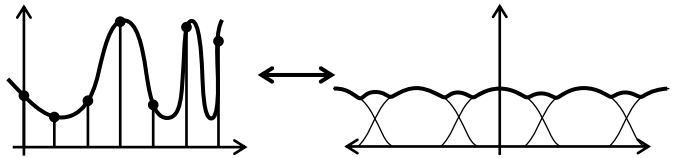
- Aliasing

Sampling

Spatial Domain Frequency Domain



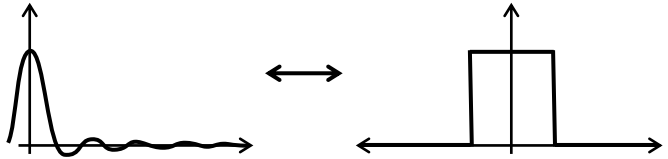
Original function



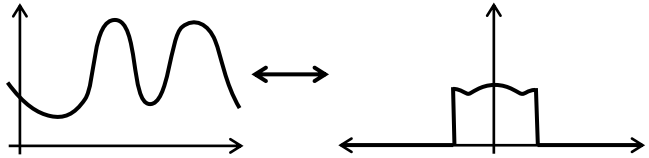
Sampled function

Reconstruction

Spatial Domain Frequency Domain



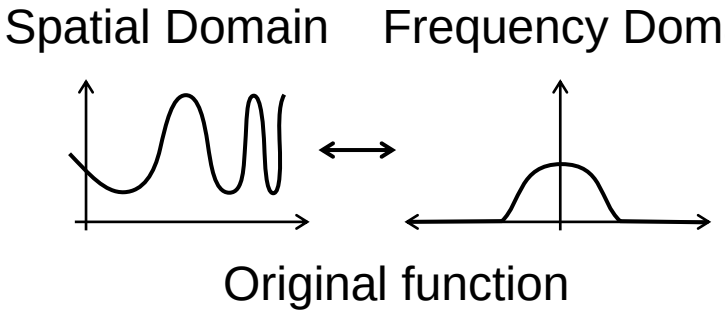
Filter function

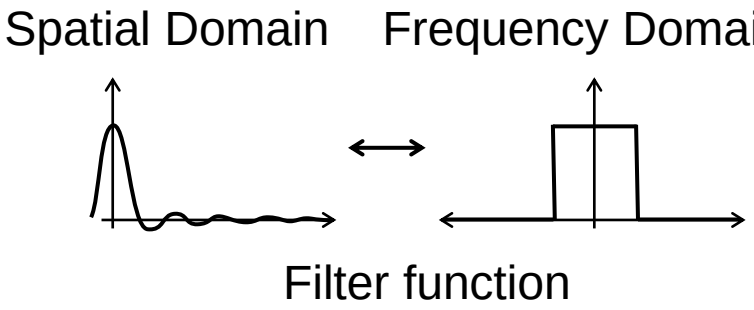


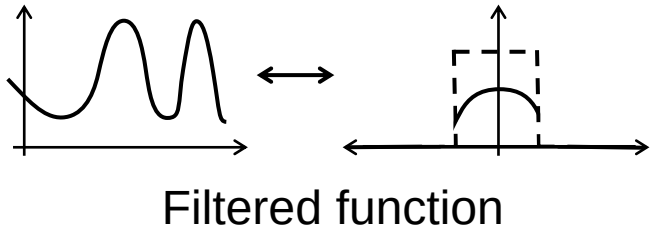
Reconstructed function

Texture Filtering

- Low-pass filtering to avoid aliasing

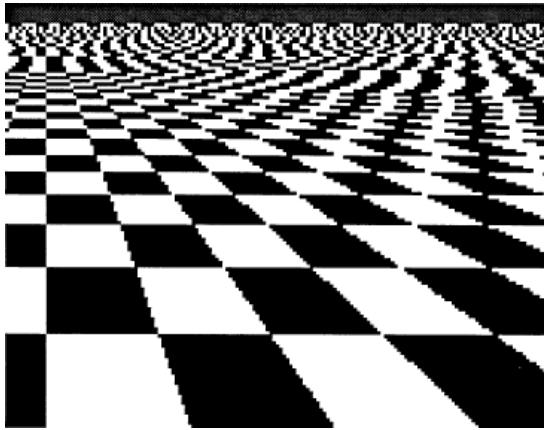






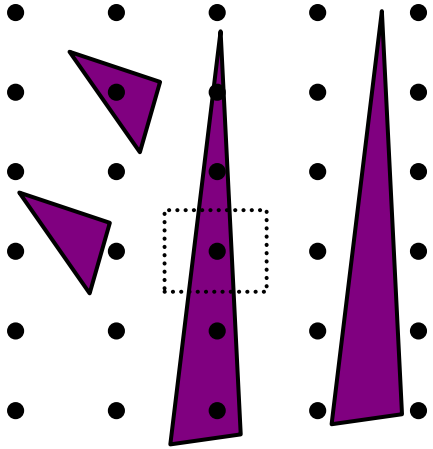
Texture Filtering

- Aliasing in computer graphics



Jaggy Edges

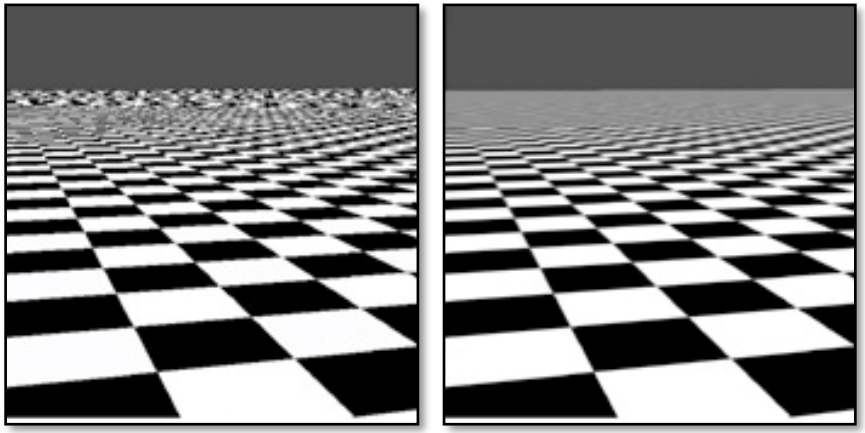
Moiré patterns



Loss of detail

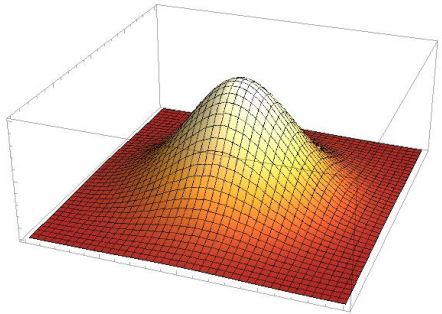
Texture Filtering

- Low-pass filtering avoids aliasing

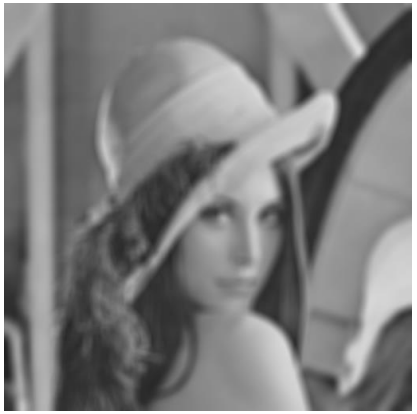


Texture Filtering

- Typical low-pass filters



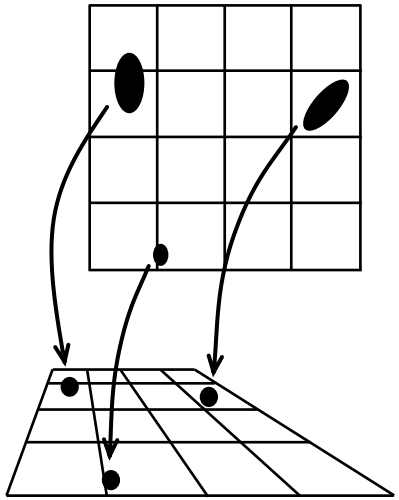
isotropic 2D Gaussian:



$$f(x,y) = \frac{1}{2\pi\sigma^2} e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)}$$

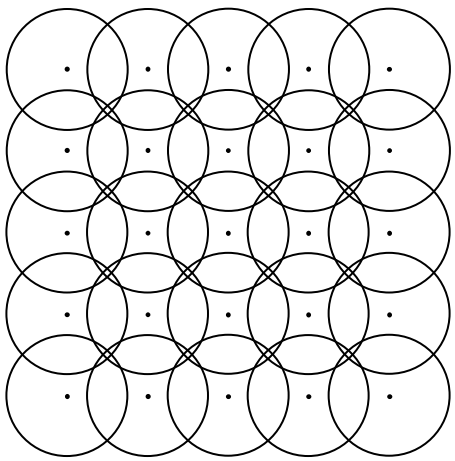
Texture Filtering

- Filtering in texture vs. image space

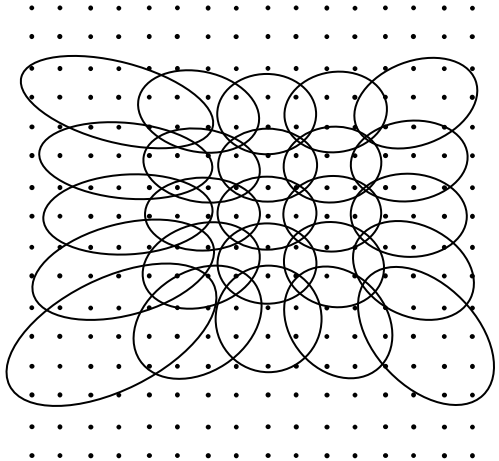


Texture Filtering

- Anisotropic spatially-varying texture filtering



Screen Space

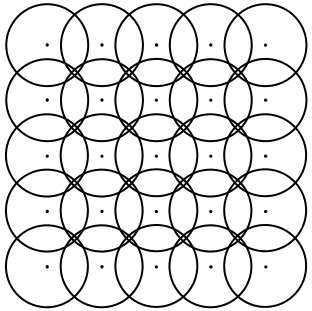


Texture Space

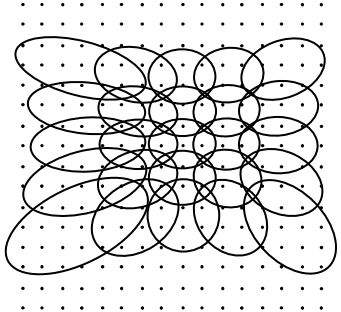
Texture Filtering

- Anisotropic spatially-varying texture filtering

Screen Space



Texture Space



Isotropic 2D Gaussian:

$$f(x,y) = \frac{1}{2\pi\sigma^2} e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)}$$

Anisotropic 2D Gaussian:

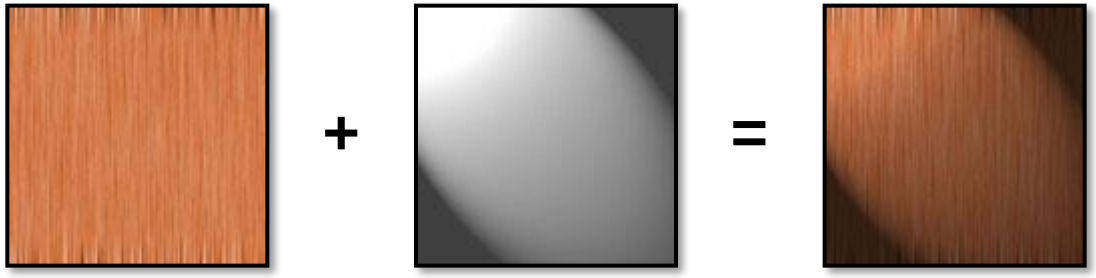
$$f(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\left(\frac{x^2}{2\sigma_x^2}+\frac{y^2}{2\sigma_y^2}\right)}$$

Light maps

- Introduced in Quake

Light maps

- Simulates the effect of a local light source
- Can be pre-computed and dynamically adapted

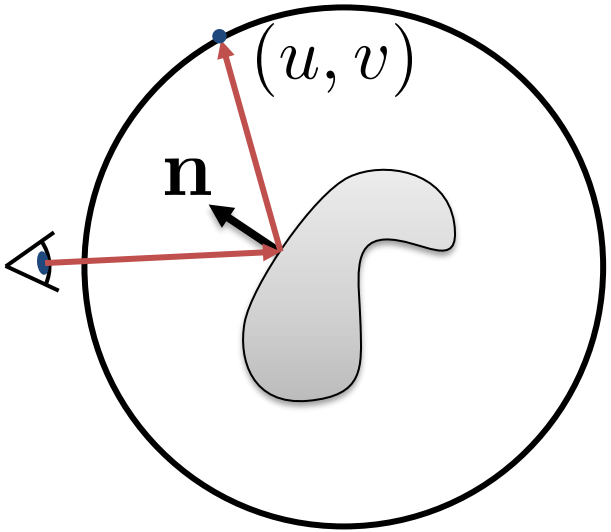


Environment Map

- Rendering reflective objects efficiently

Environment Map

- Texture coordinates are computed by intersecting the reflected ray with the surrounding sphere



Environment Map

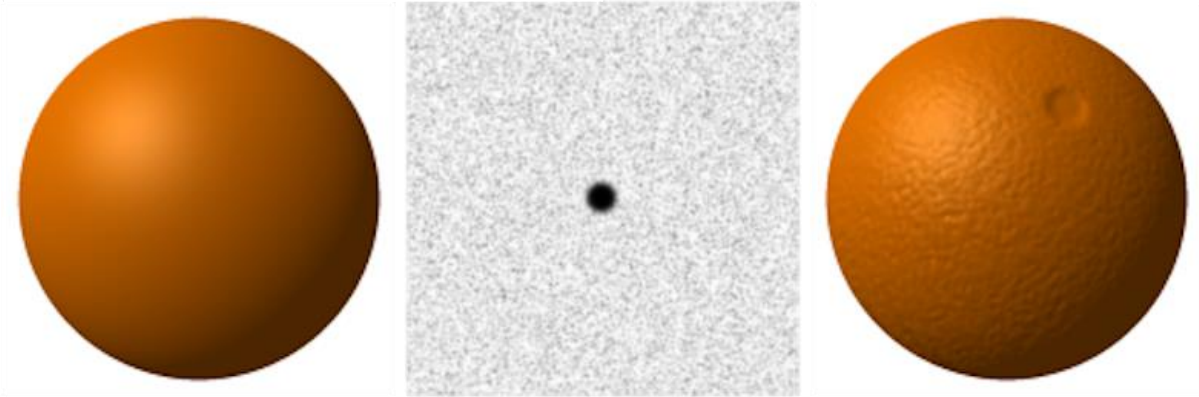
- Utilizing a cube allows simpler computations

Environment Map

- Photos can be used as environment maps

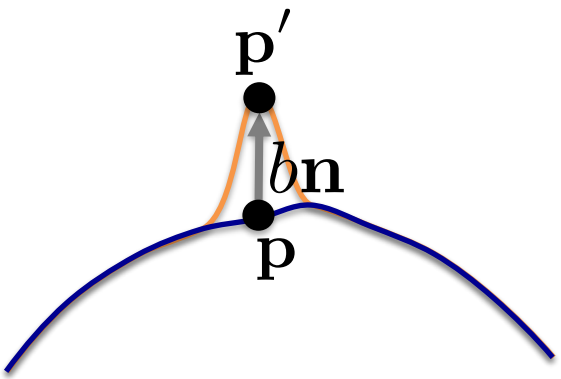
Bump Mapping

- Perturb surface normal according to textures
- Represents small-scale geometry



Bump Mapping

- How do bump maps relate to positional perturbations



$$\mathbf{n} = \frac{\partial \mathbf{p}}{\partial u} \times \frac{\partial \mathbf{p}}{\partial v} = \mathbf{p}_u \times \mathbf{p}_v$$

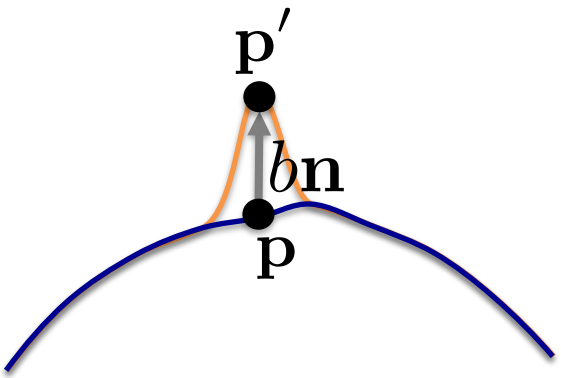
Parameter domain

$$\mathbf{p}' = \mathbf{p} + \frac{b\mathbf{n}}{|\mathbf{n}|}$$

$$\mathbf{n}' = \frac{\partial \mathbf{p}'}{\partial u} \times \frac{\partial \mathbf{p}'}{\partial v}$$

Bump Mapping

- How do bump maps relate to positional perturbations



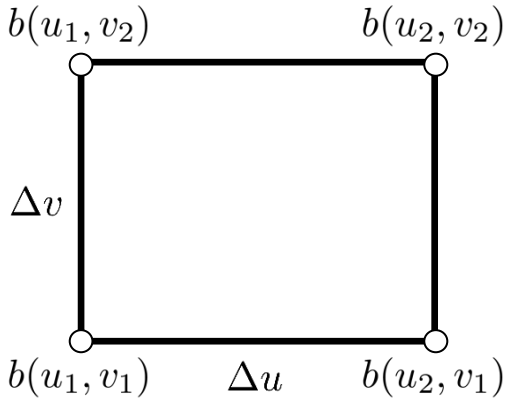
$$\mathbf{p}' = \mathbf{p} + \frac{b\mathbf{n}}{|\mathbf{n}|} \qquad \mathbf{n}' = \frac{\partial \mathbf{p}'}{\partial u} \times \frac{\partial \mathbf{p}'}{\partial v}$$

$$\frac{\partial \mathbf{p}'}{\partial u} = \frac{\partial \mathbf{p}}{\partial u} + \frac{\partial}{\partial u}\frac{(b\mathbf{n})}{|\mathbf{n}|}$$

$$\mathbf{n}' = \mathbf{n} + b_v(\mathbf{n} \times \mathbf{p}_u) + b_u(\mathbf{n} \times \mathbf{p}_v)$$

Bump Mapping

- How do bump maps relate to positional perturbations

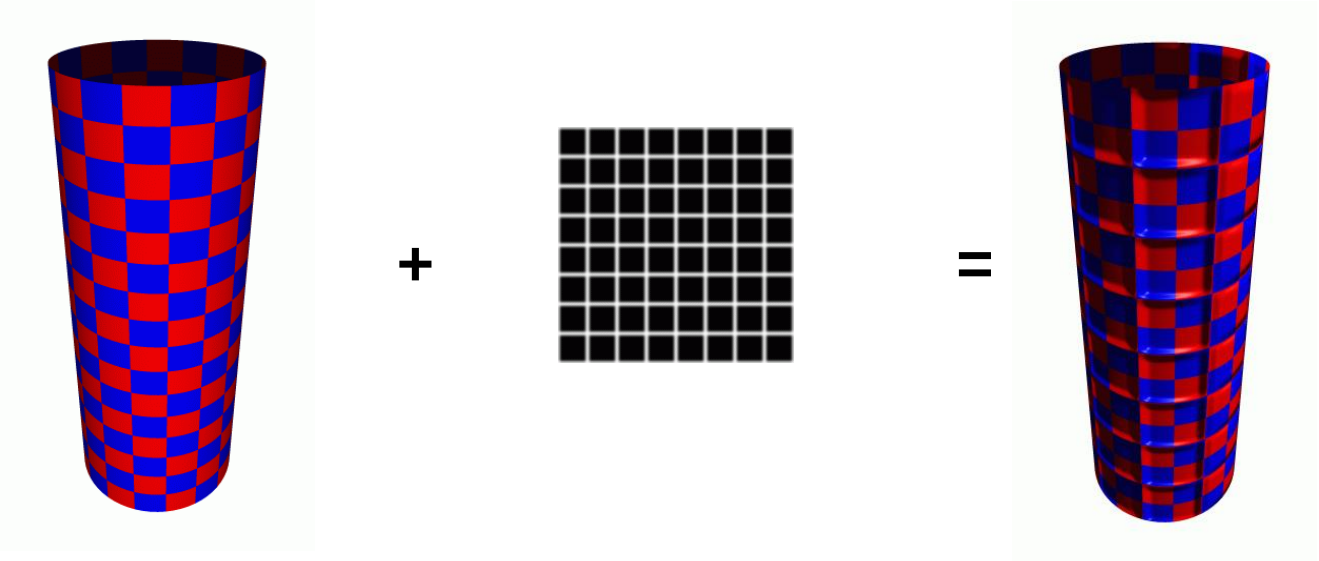


$$b_u = \frac{b(u_2, v_1) - b(u_1, v_1) + b(u_2, v_2) - b(u_1, v_2)}{2\Delta u}$$

$$b_v = \frac{b(u_1, v_2) - b(u_1, v_1) + b(u_2, v_2) - b(u_2, v_1)}{2\Delta v}$$

Bump Mapping

- Example

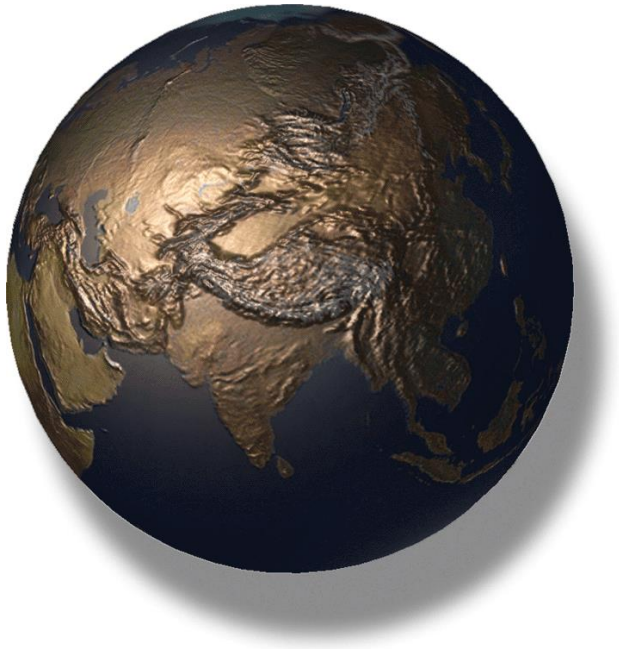


Bump Mapping

- Example

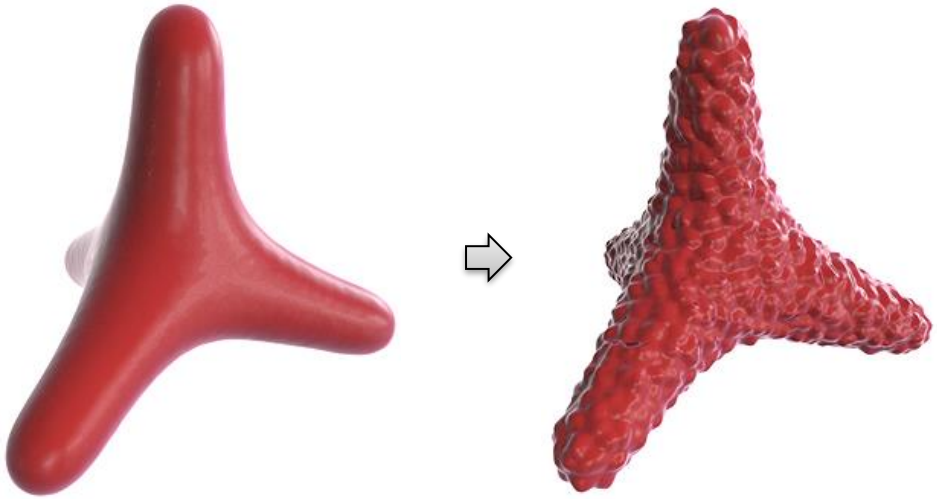
Bump Mapping

- Limitations
  - No bumps on the silhouette
  - No self-occlusions
  - No self-shadowing



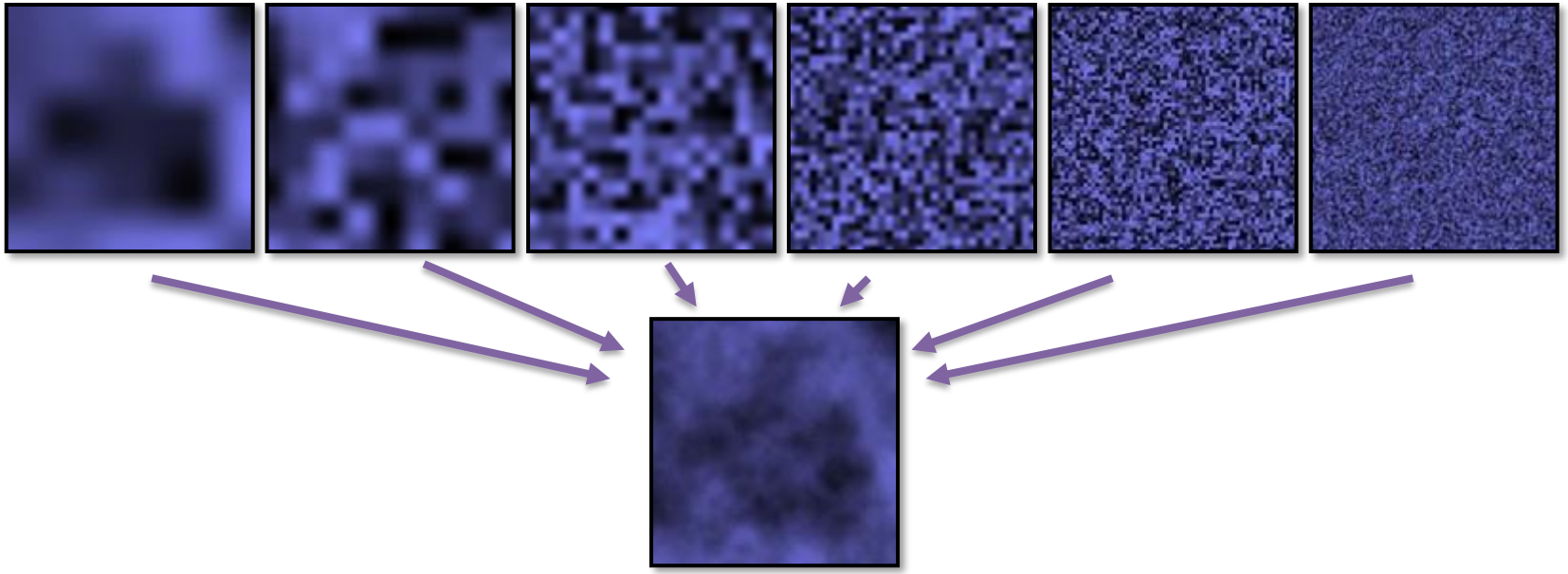
Displacement Mapping

- Perturb geometry based on textures



Procedural Textures

- Perlin Noise

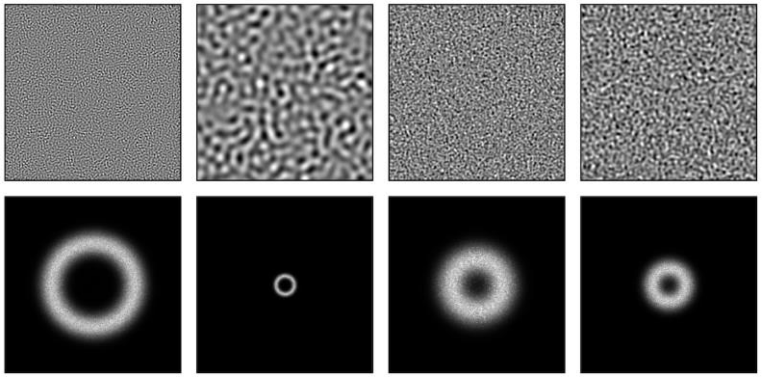


Procedural Textures

- Perlin Noise

Procedural Textures

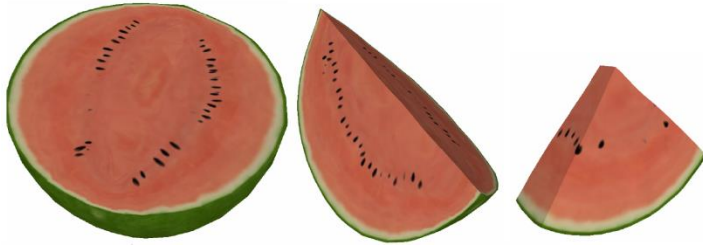
- Gabor Noise



More control over the spectral properties

Solid Textures

- 3D textures

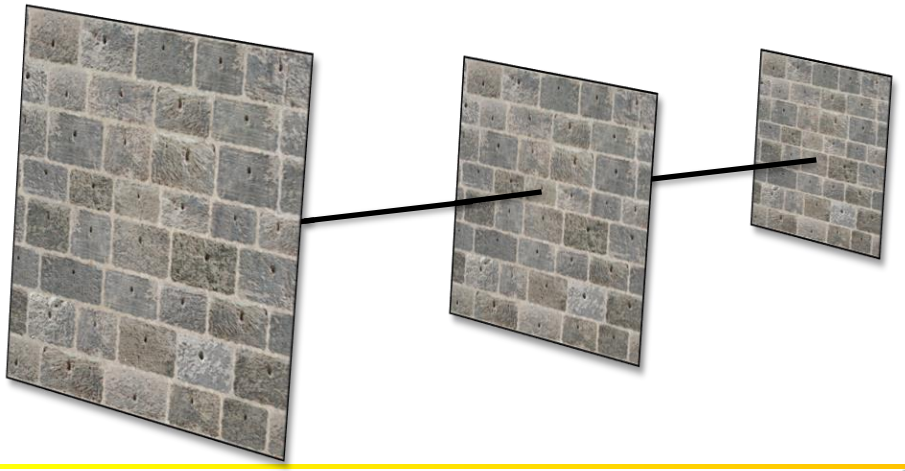


Mip-Mapping

- Store down-sampled versions of a texture

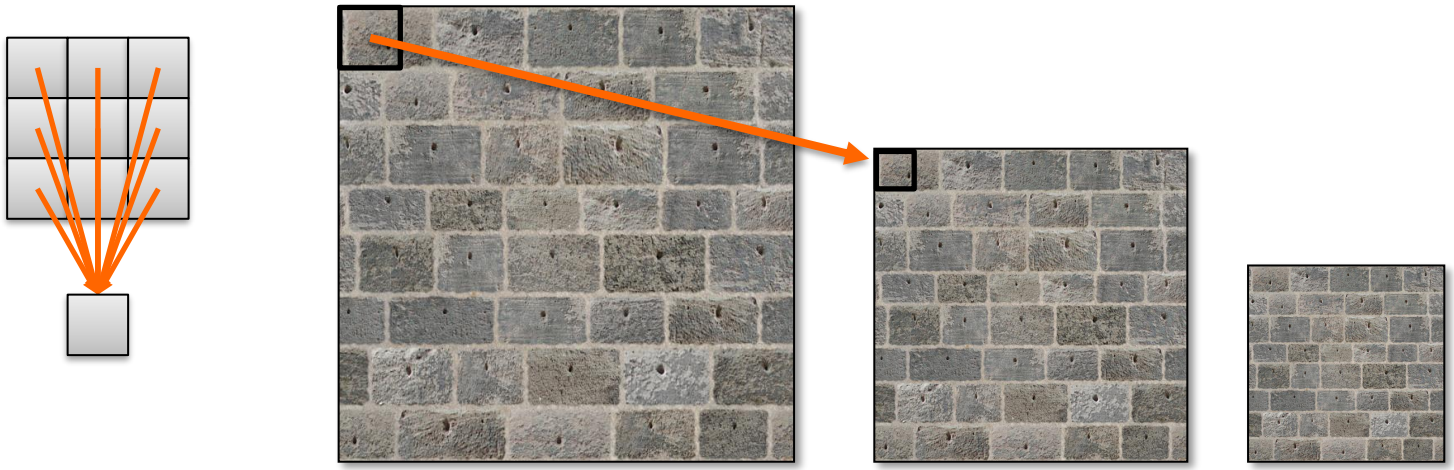
Mip-Mapping

- Use low-res versions for far away objects
- Interpolate for in-between depths
- Avoids aliasing
- Improves efficiency



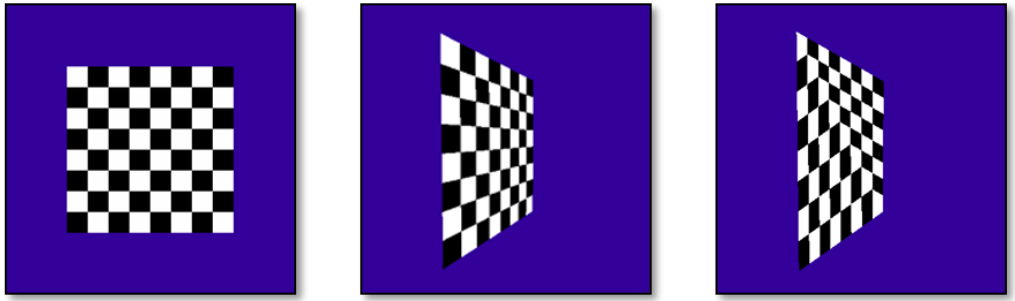
Mip-Mapping

- Compute lower resolution versions by weighted averages



Perspective Interpolation

- From texture coordinates of vertices to texture coordinates of pixels
- Linear interpolation in screen-space



Texture | We want | We get

Perspective Interpolation

- Linear variation in world coordinates yields non-linear variation in screen coordinates