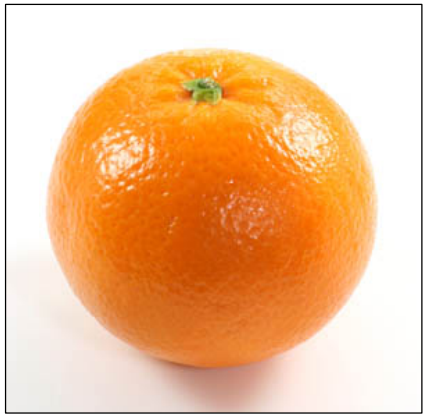
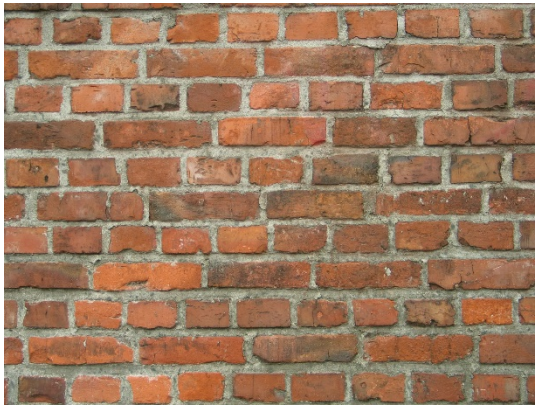
§5 Mapping-Techniques

5.1 Motivation





5.1 Motivation

- So far, all surfaces (polygonal objects, parametrized free form surfaces) are smooth – in contrast to real natural surfaces.

5.1 Motivation

- The explicit geometric representation of surface details is often too expensive (modeling and rendering).

➡ Geometric details are only simulated using different mapping-techniques.

- Initially there was only pure texture mapping (Catmull 1974): „(...) projection of (2d structures and) patterns onto surfaces of objects (...)“
- Based on this many different variants have been developed and many are supported in hardware!

➡ Today geometric/structural/color surface details are simulated using bitmaps, i.e. 2d-images.

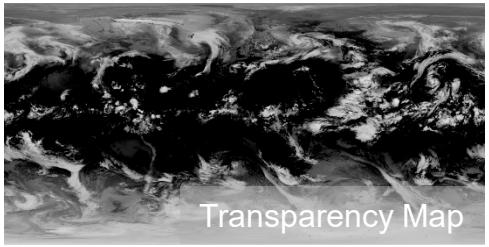
5.1 Motivation

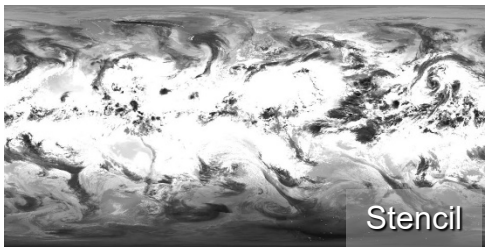
Variants

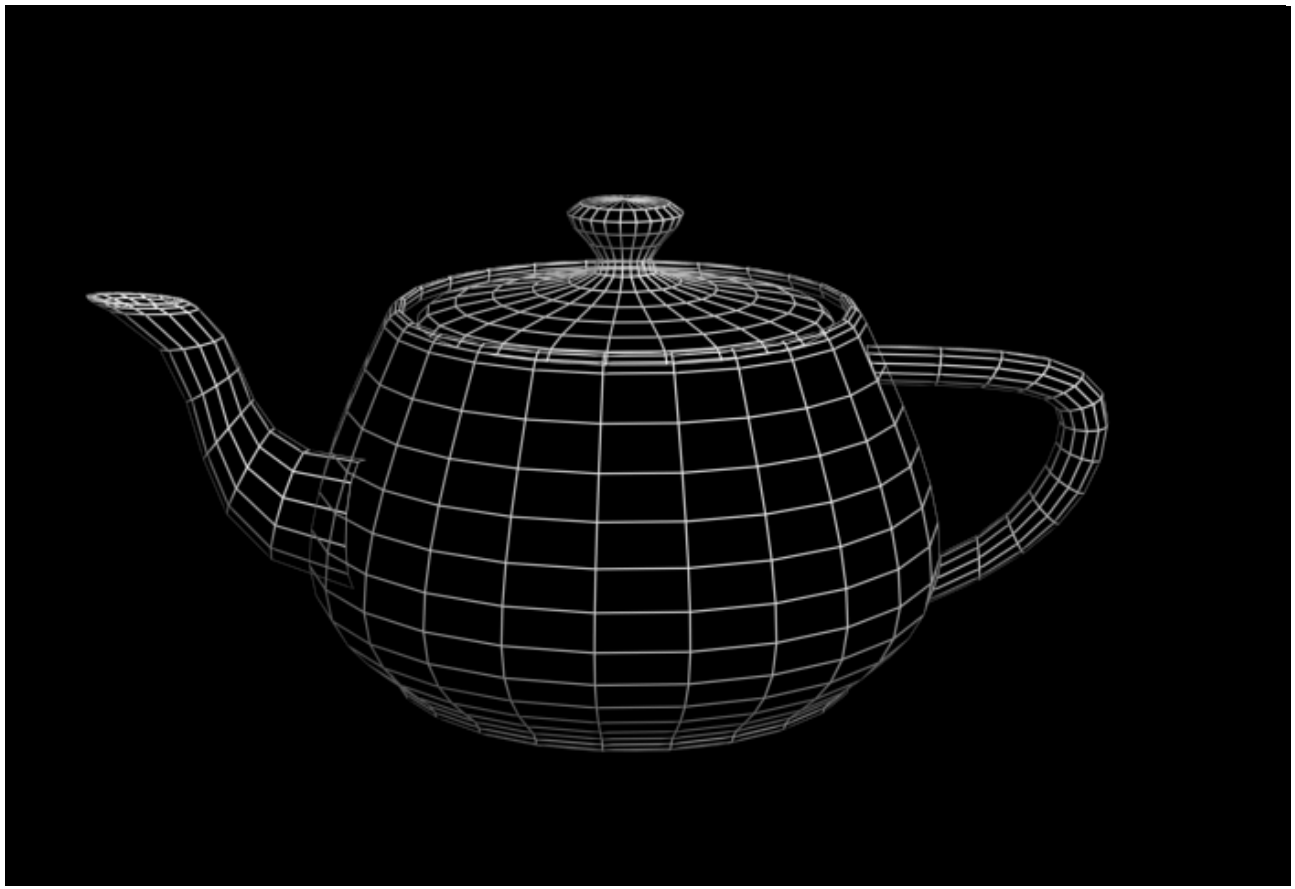
Year	Inventor	Method	Property	
1974	Catmull	**Texture Mapping**	Color	
1976	Blinn, Newell	**Reflection Mapping**	Reflections	
1978	Blinn	**Bump Mapping**	Normals	
1985	Gardener	**Transparency Mapping**	Transparency	
1986	Greene	**Environment Mapping**	Reflections	
1987	PIXAR	**Displacement Mapping**	Shape, geometry	

- Further methods: Procedural Mapping, 3D Texture Mapping, …

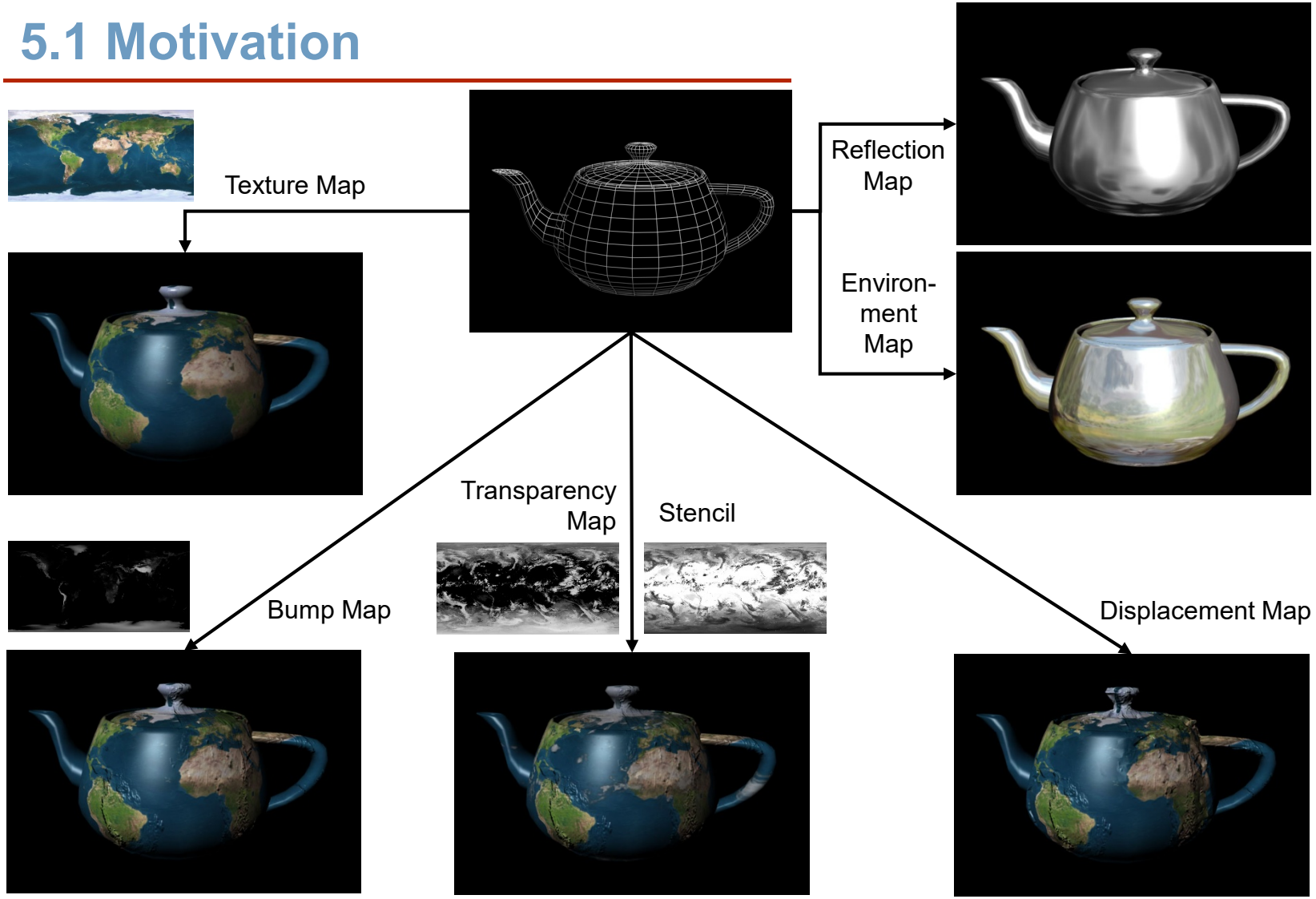
5.1 Motivation







5.1 Motivation



5.1 Motivation

Goals

- Rendering of surface details (materials)
- Without expensive geometry computations
- Without expensive representation
- Without expensive rendering

„All it takes is for the rendered image to look right.“

(Jim Blinn, SIGGRAPH‘84)

Principle

Mapping $T: \mathbb{R}^3 \rightarrow \mathbb{R}, \mathbb{R}^2, \mathbb{R}^3 \ , (x, y, z) \mapsto m$

Problems

- Realization of mapping? Step A
- Storage of mapping? Step B

} Usually done in two steps.

5.2 Texture Mapping

Step B

- Usually: Store the mapping values in bitmaps („texture map“)
- Example: $T: \mathbb{R}^2 \rightarrow \mathbb{R}^3$ with $T(u,v)$: $[0,1] \times [0,1] \rightarrow \mathbb{R}^3$
  - (u,v) are the so-called **texture-coordinates**.
- Alternative: Procedural generation

Step A

- Mapping of 3d-point-coordinates to 2d-texture-coordinates

Combined: $(x,y,z) \in \mathbb{R}^3 \xrightarrow{A} (u,v) \in \mathbb{R}^2 \xrightarrow{B} T(u,v) \in \mathbb{R}^3$

- For visible vertices only.
- For pixel within visible polygons: interpolation

5.2 Texture Mapping

Terms

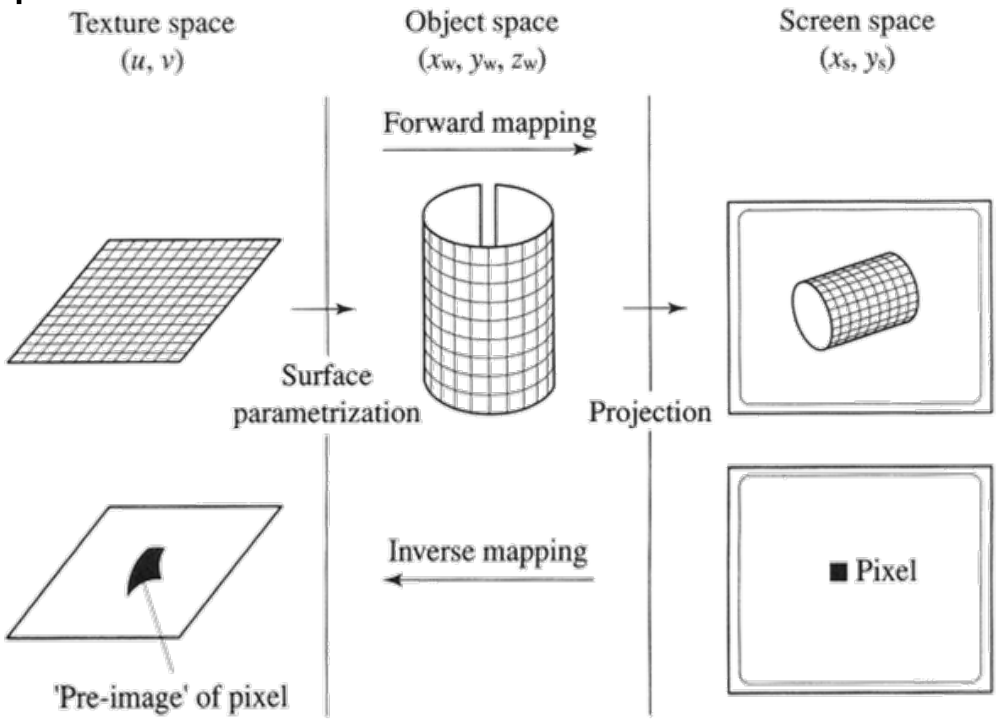
Texture map: The images that should be mapped onto the object (real photo or rendered image)

Texel: Pixel of the texture map.

Different interpretations

Forward mapping: The mapping from texture to screen space (via object space).

Inverse Mapping: The mapping from screen space to texture space.



5.2 Texture Mapping

- In practice it is often easier to split the mapping into two different sub-steps (in the view of the forward mapping):

1. First use a suitable (simple) mapping of the texture onto a simple intermediate, auxiliary surface, e.g. rectangle, cube, cylinder, sphere, etc.

   ➡ „*s-mapping*“

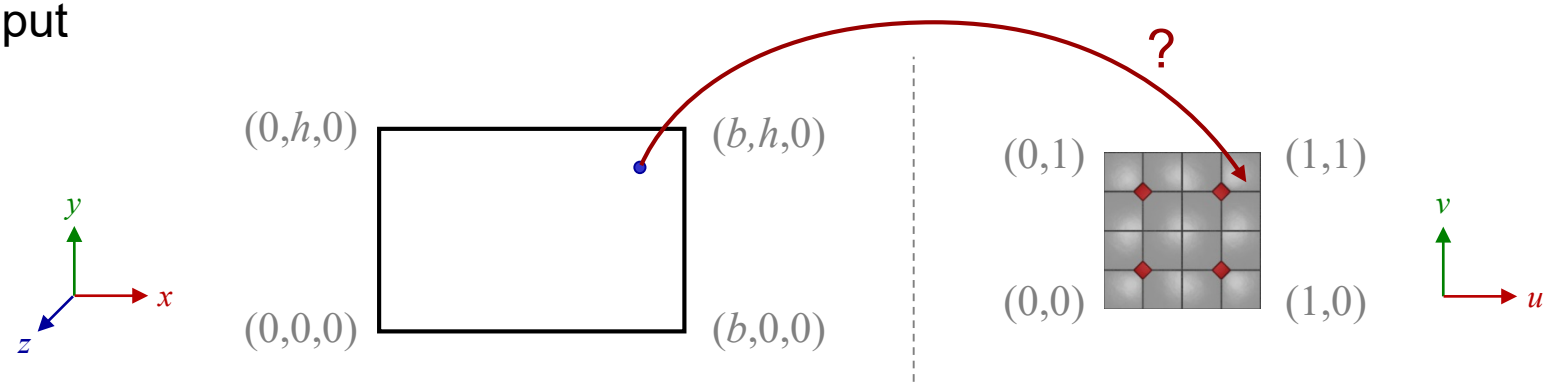
2. From the intermediate surface the texture is mapped onto the final surface of the object.

   ➡ „*o-mapping*“

5.2 Texture Mapping (s-mapping)

Example: Intermediate surface is planar rectangle

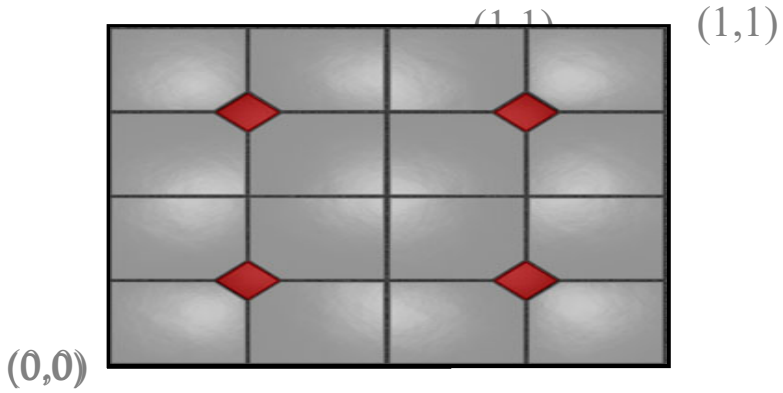
- Input



- Determine the texture-coordinates

$$u(x,y,z) = \frac{x}{b}, \quad v(x,y,z) = \frac{y}{h}$$

$$(x,y) = [0,b] \times [0,h]$$



5.2 Texture Mapping (s-mapping)

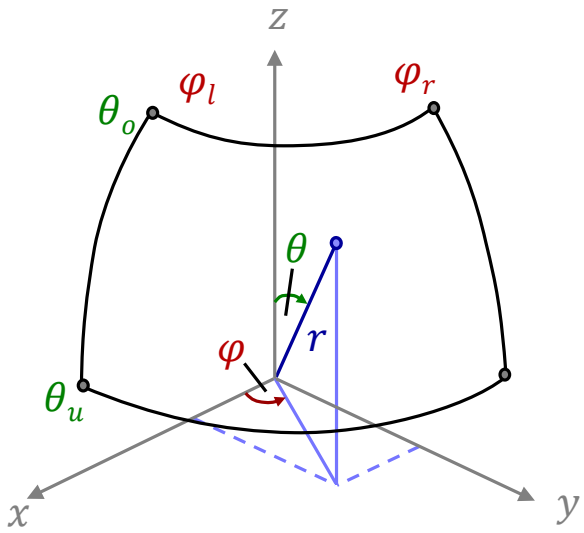
Example: Intermediate surface is a sphere

- Spherical coordinates: $(x, y, z) = (r\cos\varphi\sin\theta\,, r\sin\varphi\sin\theta\,, r\cos\theta)$
  with: $\varphi \in [0,2\pi], \theta \in [0,\pi]$.
- Mapping of planar rectangle to sphere ➡ distortions
- Limit mapping to a sub-sphere.
- Determine the texture-coordinates

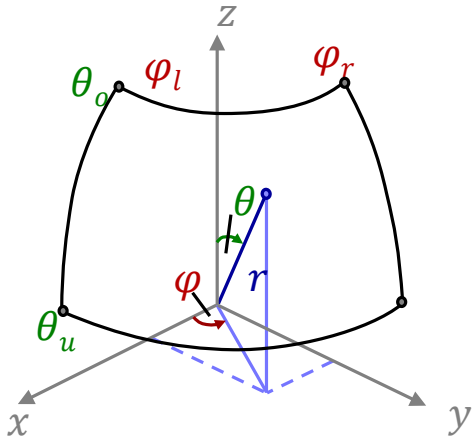
$$u(\varphi,\theta) = \frac{\varphi-\varphi_l}{\varphi_r-\varphi_l},$$

$$v(\varphi,\theta) = \frac{\theta_u-\theta}{\theta_u-\theta_o},$$

$$(\varphi,\theta) \in [\varphi_l,\varphi_r] \times [\theta_o,\theta_u].$$



5.2 Texture Mapping (s-mapping)

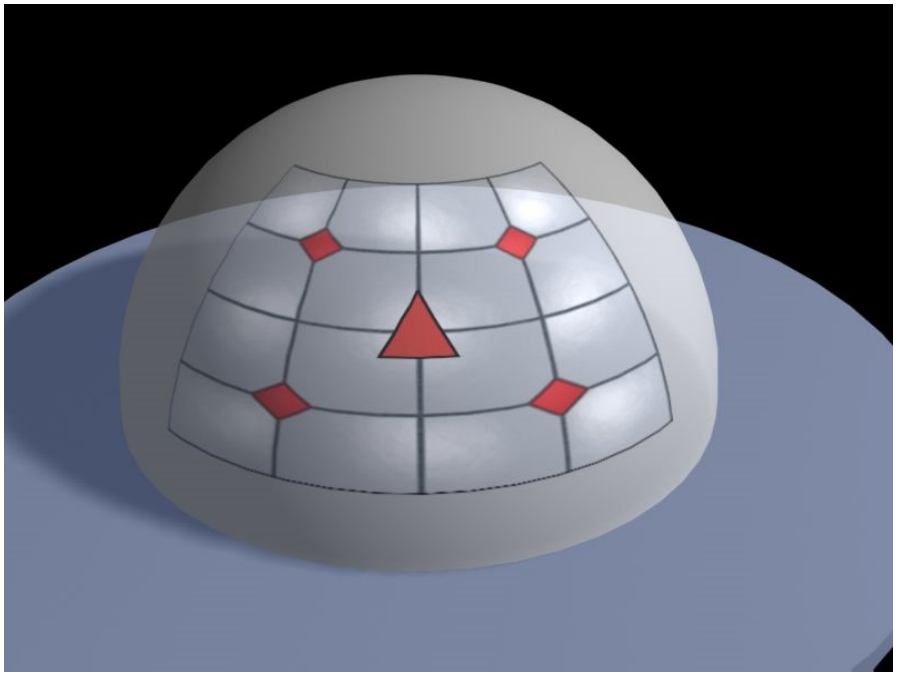


Example: Intermediate surface is a sphere

- Sub-sphere: $\varphi_l = 0, \varphi_r = \frac{\pi}{2}, \theta_o = \frac{\pi}{4}, \theta_u = \frac{\pi}{2}$.

$$(u, v) = \left(\frac{\varphi}{\frac{\pi}{2}}, \frac{\frac{\pi}{2} - \theta}{\frac{\pi}{4}} \right),$$

with $u, v \in [0,1]$.

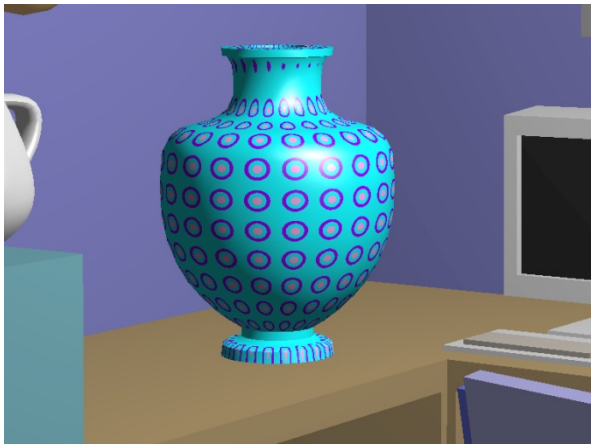


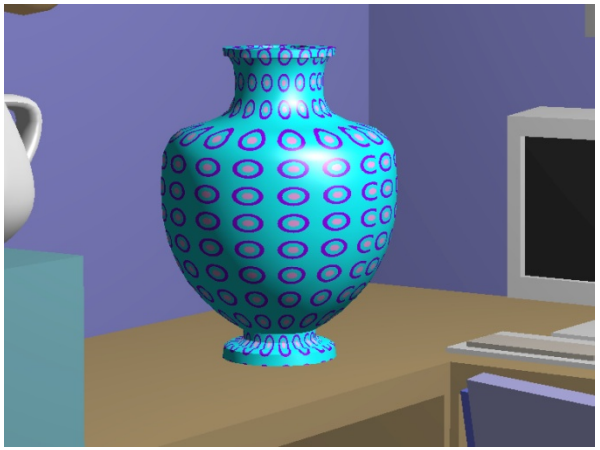
5.2 Texture Mapping (s-mapping)

Example: Different intermediate surfaces

Planar

Sphere





Cylinder

5.2 Texture Mapping (o-mapping)

Techniques for o-mappings

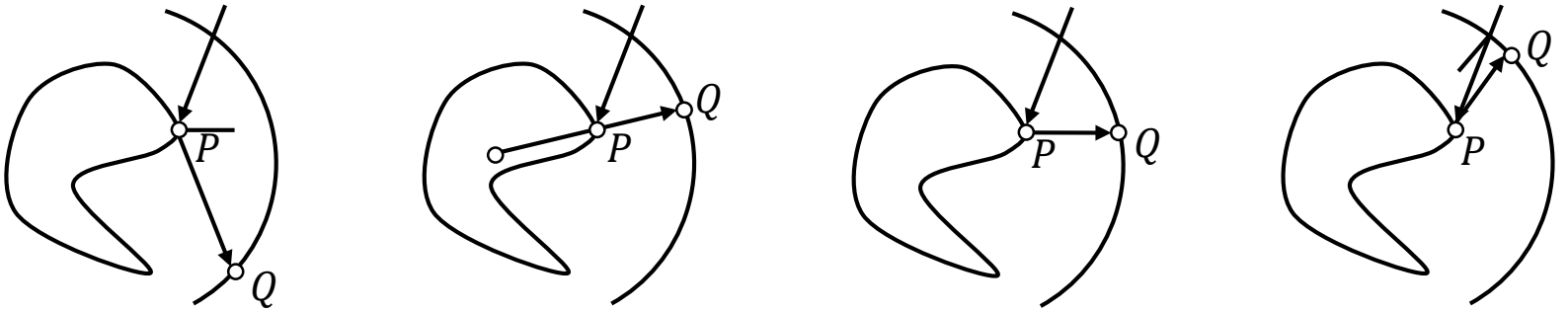
Determination of the point Q on the intermediate surface from the point P on the object:

a) Reflection ray: ideal reflection at the object.

b) Object center: ray from the object center through P.

c) Normal vector: normal of the object at P.

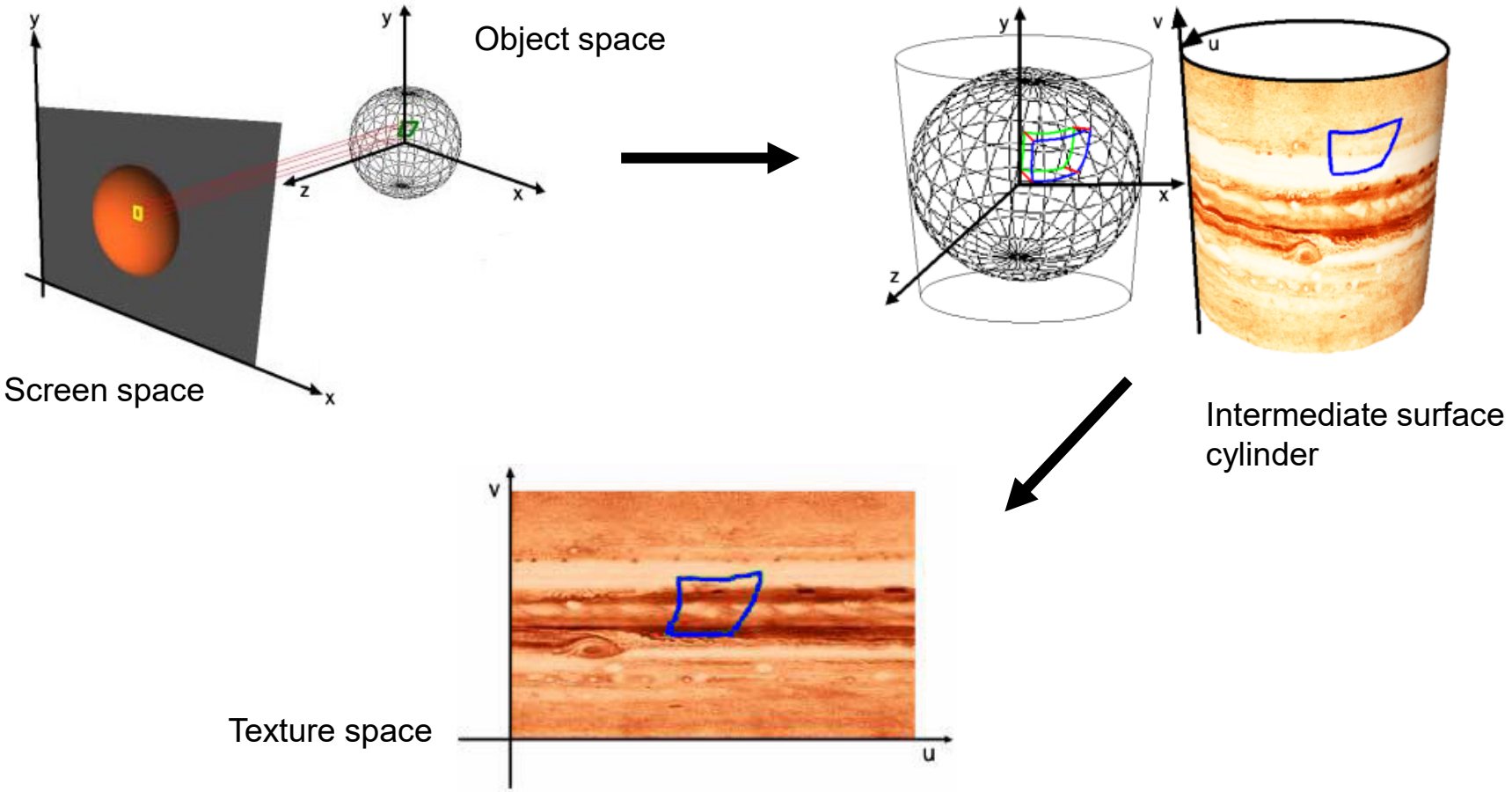
d) Normal of the intermediate surface at intersection.



a) b) c) d)

5.2 Texture Mapping (s- and o-mapping)

Inverse mapping with intermediate surface:



5.2 Texture Mapping

Without Texture

Example:

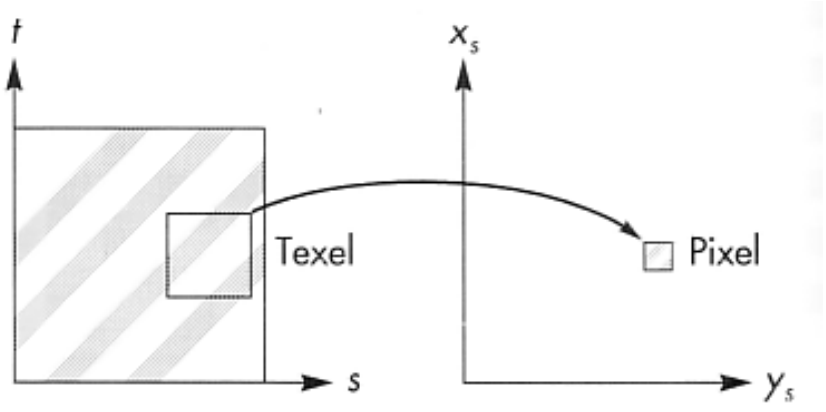
5.2 Texture Mapping

Texture mapping and aliasing

Texture mapping is prone to aliasing effects:

1. A pixel in screen space coordinates can overlap the area of multiple texels in texture space after back-projection.
   - ➡ Sampling?
   - ➡ Ideal solution: Integral
   - ➡ In practice: Sampling + filtering

Minification

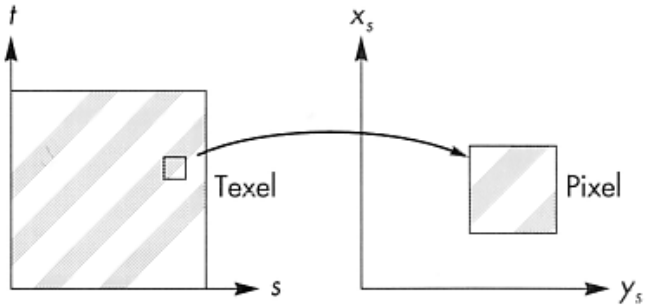


5.2 Texture Mapping

2. A texel in the texture can overlap multiple pixels in screen space.

   ➡ Sampling?

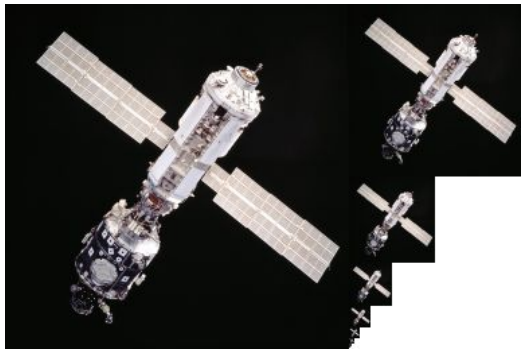
   Magnification



3. Texture maps are usually periodically extended, if larger surfaces are to be textured.

   ➡ Attention: Periodicity and sampling theorem!

➡ Oversampling, Filtering and Mip-Mapping



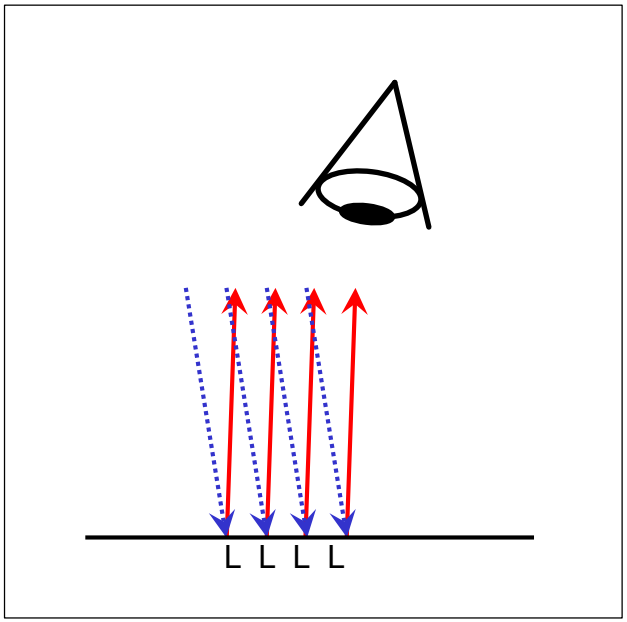
5.3 Bump Mapping

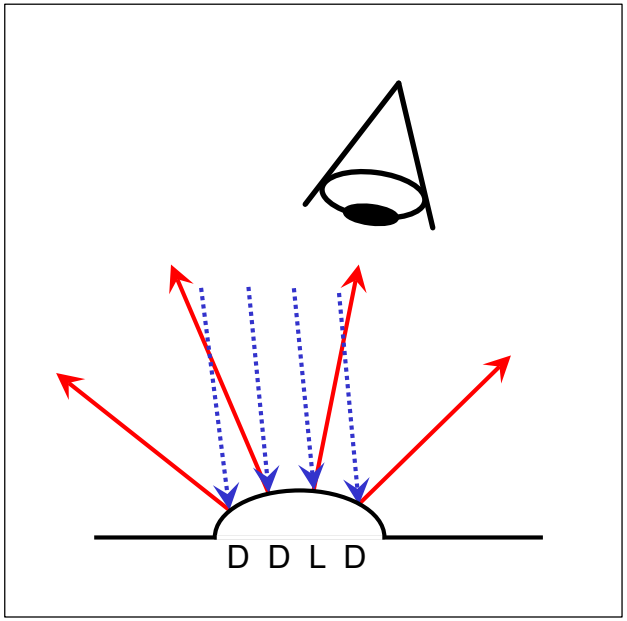
- Pure texture mapping generates the impression of textured but smooth/planar surfaces.
- In order to “roughen“ these surfaces and to give them a 3d-impression, use e.g. bump mapping
  - manipulates the normal for the evaluation of the illumination model,
  - it does **not** change the geometry of the surface itself.

Simulation of surface bumps on a smooth surface through the change of surface normal vectors.

5.3 Bump Mapping

Observation:





Legend:

incoming light — D = dark

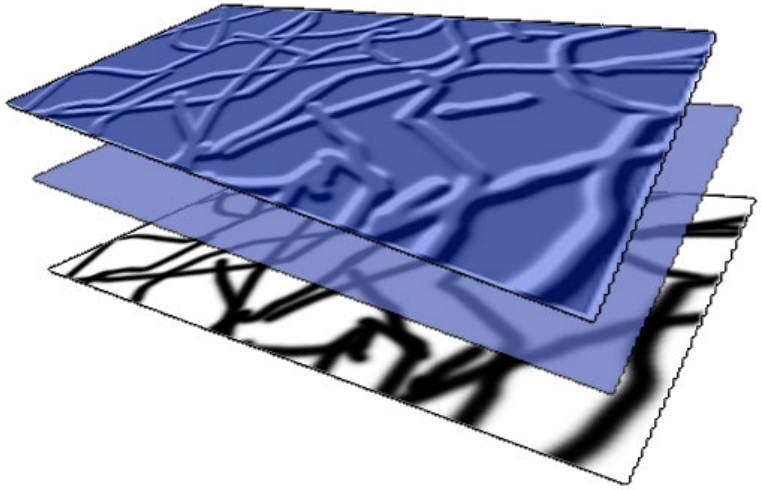
reflected light — L = light

5.3 Bump Mapping

- The change ΔN of the normal vectors N is procedural or uses a texture map.
- The normal change ΔN can be represented e.g. as a simple grey level texture.
  - The gradient in this texture (interpreted as vector field) yields length and direction of the change ΔN.
- Regular structures (e.g. golf ball) and irregular structures (e.g. tree bark) can be simulated.
- The silhouette of an object with bump map texture, is still smooth, i.e. differs visual perceivably from reality.

5.3 Bump Mapping

Example:



5.4 Displacement Mapping

- Overly the surface with a height field(e.g. as texture map), such that surface points are translated along the surface normal.
- This approach changes also the silhouette of objects realistically.

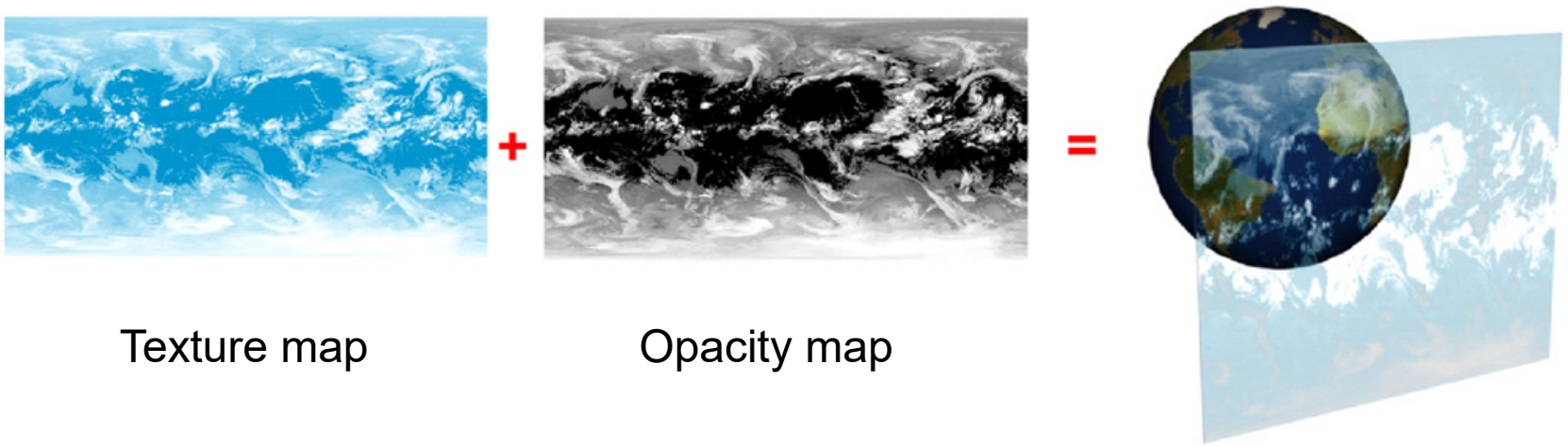
5.4 Displacement Mapping

Without displacement mapping

Example:

5.5 Opacity Mapping / Transparency Mapping

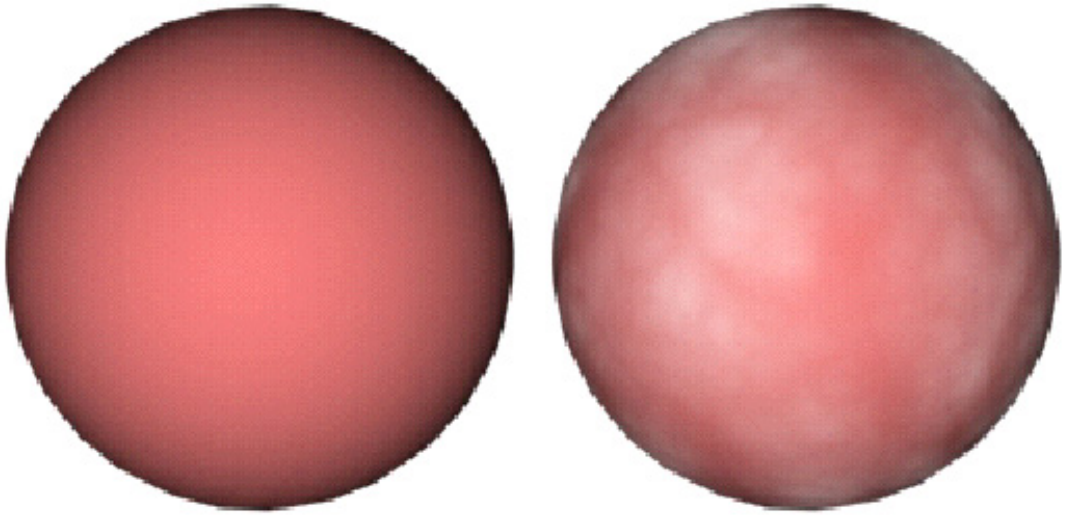
- Using opacity mapping the α-value of transparent surfaces is manipulated locally.
- The object, the opacity map is applied to, is partially and gradually transparent depending on the bitmap image.



Texture map Opacity map

5.6 Procedural Mapping

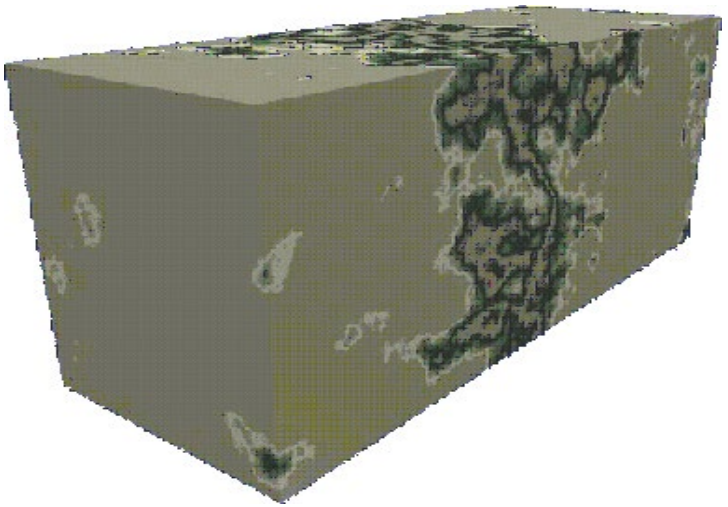
- For procedural mapping an algorithmic description for the simulation of irregularities is required, i.e. an algorithmic texture synthesis.
- ➡ Texture generation on the fly.
- This is used e.g. for the generation of 3d-textures.
- Example: Simulation of irregularities.



5.7 3D (Texture) Mapping

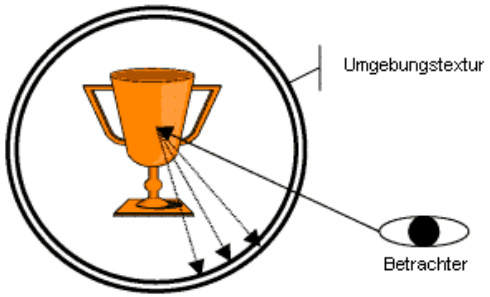
- Instead of a 2d image a three-dimensional texture (trivariate function) is used.
- Using procedural approaches realistic 3d patterns can be generated.

Wood grain



Marble

5.8 Environment Mapping



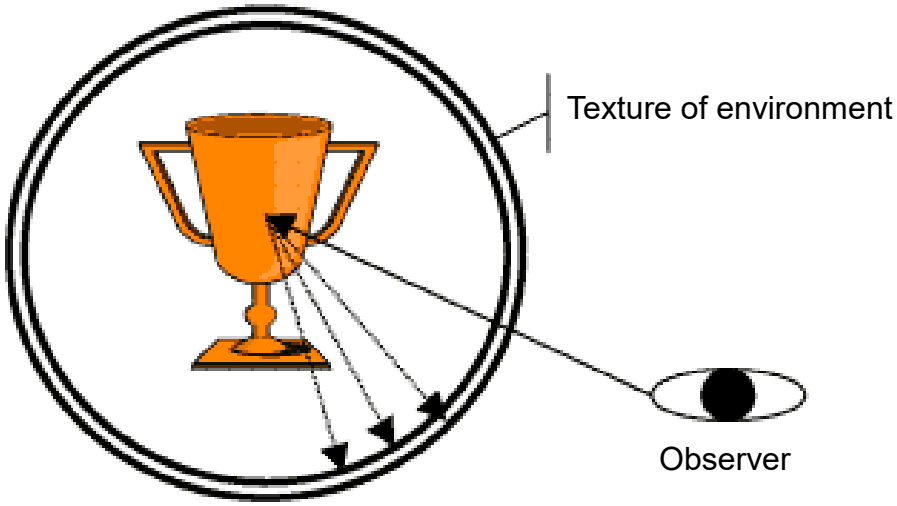
Motivation

- So far: Texture coordinates are fix, also if the object/camera moves.
- Problem: Not applicable for specular objects (e.g. shiny sphere)
- Usual solution: Ray-tracing
  - Correct simulation of the light path following the laws of geometric optics.
  - But: Software-rendering! Poor GPU-support!
  - ➡ No real time rendering!
- Goal of environment mapping: Approximation of reflections using the texture hardware !

5.8 Environment Mapping

- Environment Mapping simulates realistic specular effects of a (virtual or real) environment of an object.
- Integration of a complex scene into a photo-realistic image, without explicit modelling of the environment.

1. Use an intermediate, auxiliary object (sphere, cube), and
2. project on its surface an image of the environment.

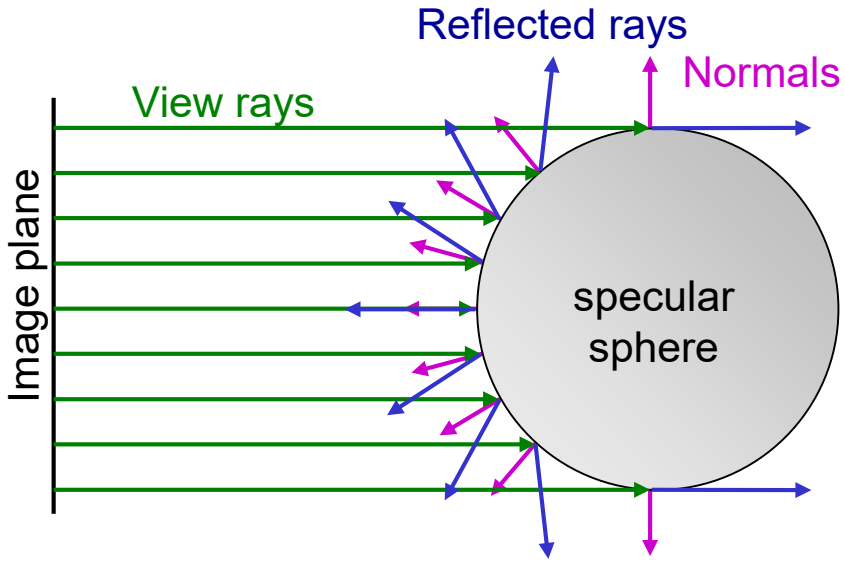


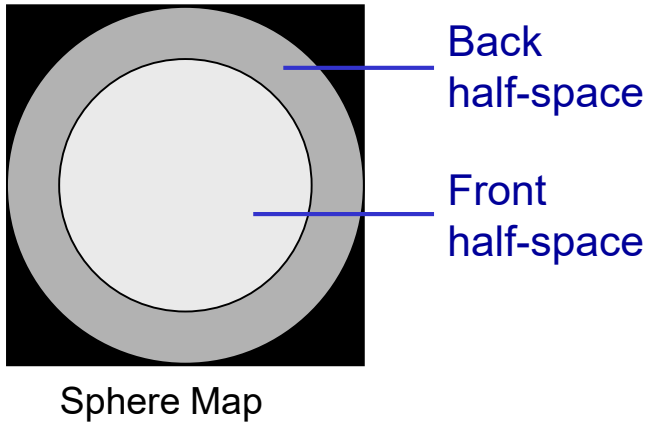
- Today hardware supported!

5.8 Environment Mapping

Historically oldest method: Sphere Mapping

- Geometry





- Idea: Observer is “infinitely” far away, sphere radius is very small.

5.8 Environment Mapping

Realization of step A

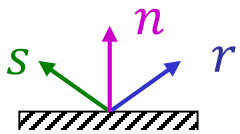
- Mapping of 3d-point-coordinates to 2d-texture-coordinates:

$(x, y, z) \in \mathbb{R}^3 \longrightarrow$ Reflection vector $r \in \mathbb{R}^3$ $\longrightarrow (u, v) \in \mathbb{R}^2$

Law of reflection

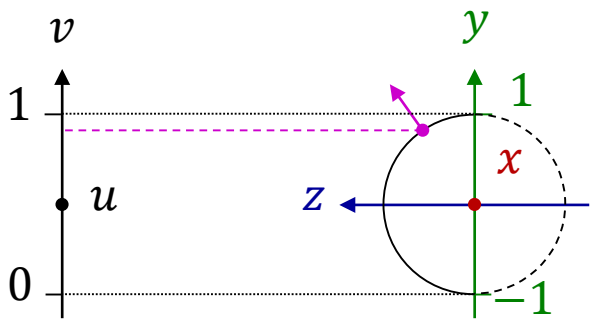
Reflection direction
↔ Texture-coordinates

- Law of reflection:



$r = 2(s \cdot n)\, n - s$ §4.6.2

- Geometry-setup: unit sphere in the origin

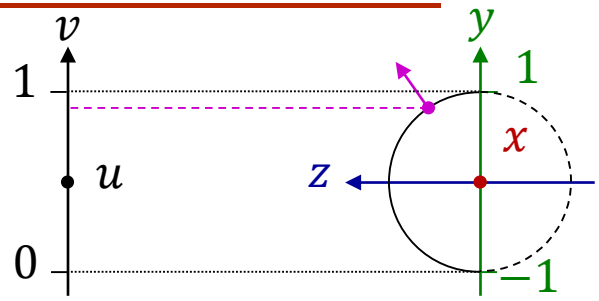


➡ Mapping: Point-coord. ↔ Texture-coord.

$$x = 2u - 1$$
$$y = 2v - 1$$
$$z = \sqrt{1 - x^2 - y^2}$$

Parallel projection of unit sphere

5.8 Environment Mapping



- Normal: $n = n_0 = (n_x, n_y, n_z)^t = (x, y, z)^t$
- View direction: $s = (0,0,1)^t$
- Law of reflection: $r = \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix} = 2n_z \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$

$x = 2u - 1$

- Solve for n and normalize: $n_0 = \frac{1}{\sqrt{r_x^2 + r_y^2 + (r_z+1)^2}} \begin{pmatrix} r_x \\ r_y \\ r_z + 1 \end{pmatrix} = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix}$
- Result: $u = \frac{r_x}{2\sqrt{r_x^2 + r_y^2 + (r_z+1)^2}} + \frac{1}{2}$, $v = \frac{r_y}{2\sqrt{r_x^2 + r_y^2 + (r_z+1)^2}} + \frac{1}{2}$

5.8 Environment Mapping

Example

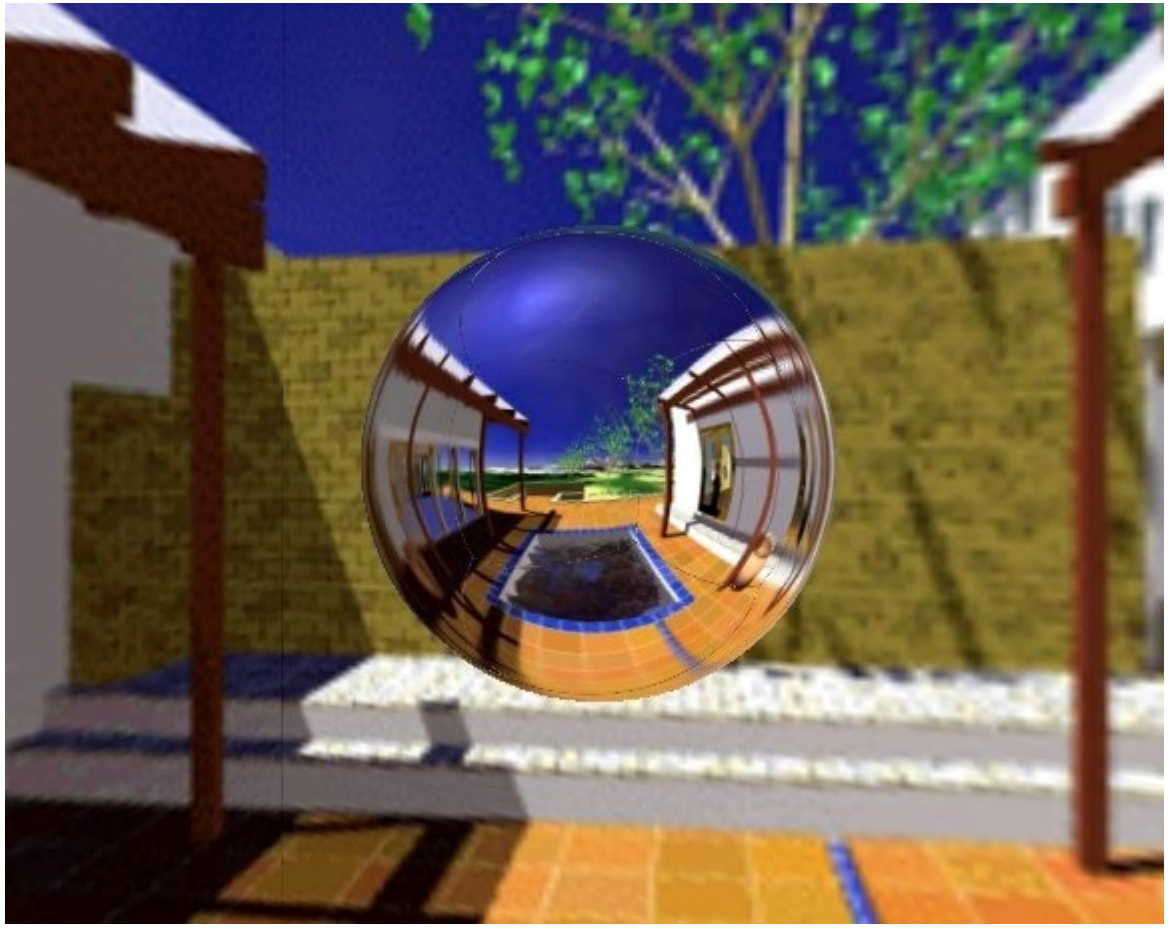
Important problems

- Sphere map only valid for one camera position!
- Dynamic re-computation of sphere map is expensive!

Improvements

- Dual-paraboloid-mapping
- Cube mapping (implemented in today's GPUs)

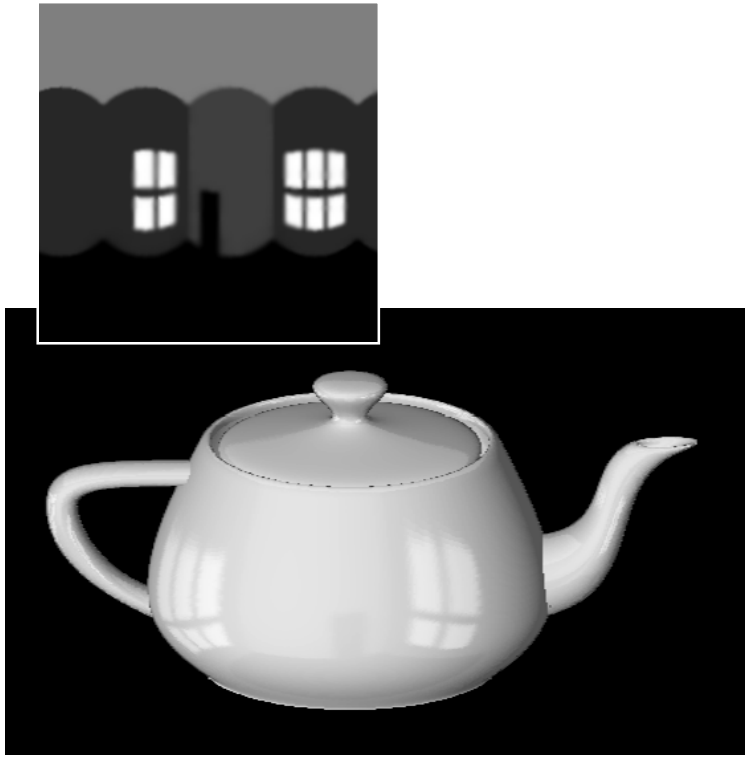
5.8 Environment Mapping



„Bubble“

5.8 Environment Mapping

Example:



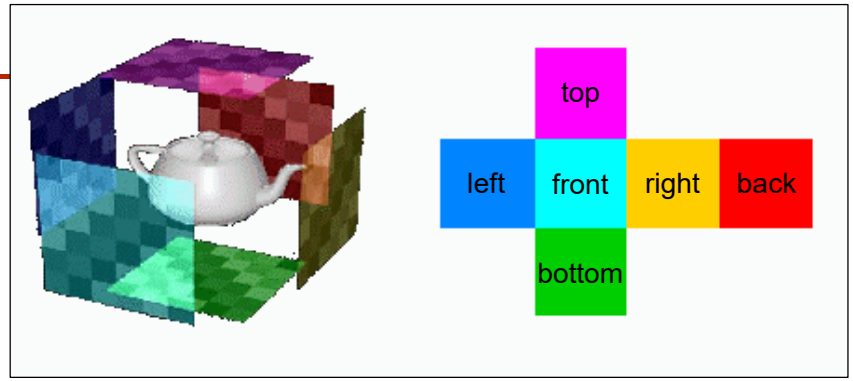
5.8 Environment Mapping

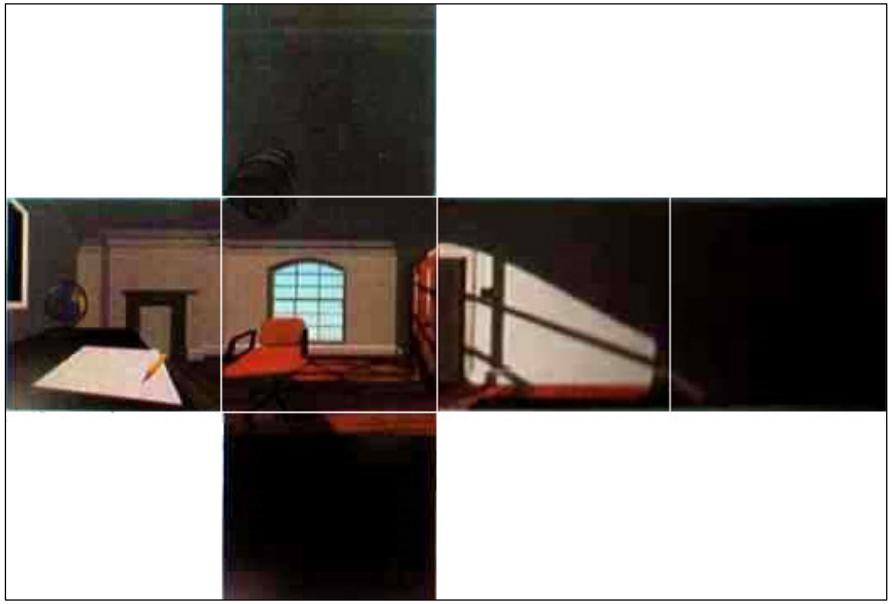
Example:

Without environment mapping

5.8 Environment Mapping

Example (Cube-Mapping):





5.10 Résumé

- All kinds of mapping-techniques are very prone to aliasing-effects!
- Different kinds of mapping-techniques can be combined and applied at the same time to the same object.
- Standard tools for rendering- and animation have these techniques build in.
- Mapping-techniques are the basic for almost all commercial computer graphics applications.

5.10 Résumé

Example: Chrome / Reflection Mapping + Ray Tracing

Goal

- What is texture mapping?
- What is o-mapping and what is s-mapping?
- Which types of aliasing-problem occur for texture mapping?
- What is bump mapping?
- What is displacement mapping?
- What is transparency mapping?
- What is environment mapping?