Fundamentals of Computer Graphics and Image Processing

Textures, Mappings (06)

doc. RNDr. Martin Madaras, PhD.
martin.madaras@fmph.uniba.sk

Overview

- Texture mapping
  - 3D Models with texture coordinates
  - UV map parametrization
  - Perspective correction
- Aliasing
- Anti-aliasing
  - Supersampling
  - Mip Maps
- Advanced textures
  - Environment mapping
  - Bump mapping
  - Normal mapping, Displacement mapping etc.

How the lectures should look like #1

- Ask questions, please!!!
- Be communicative
- More active you are, the better for you!

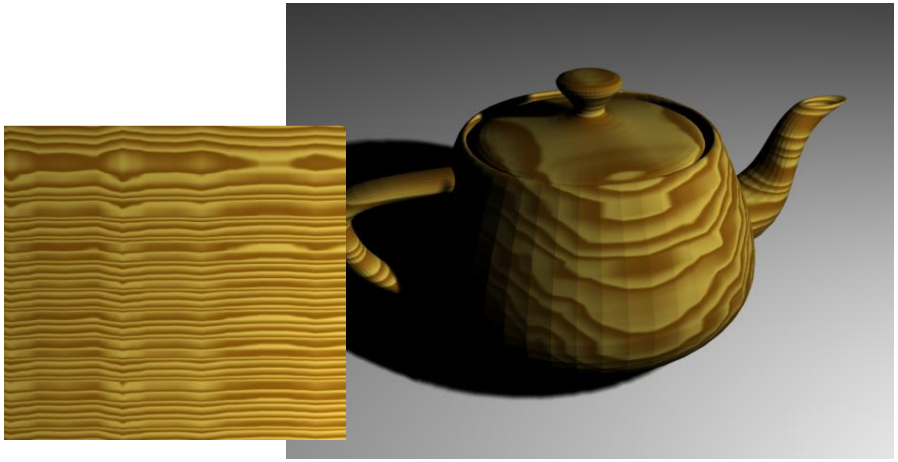
Material

- Visually distinguishes 2 objects with identical geometry

- For now, we focus on object’s own color

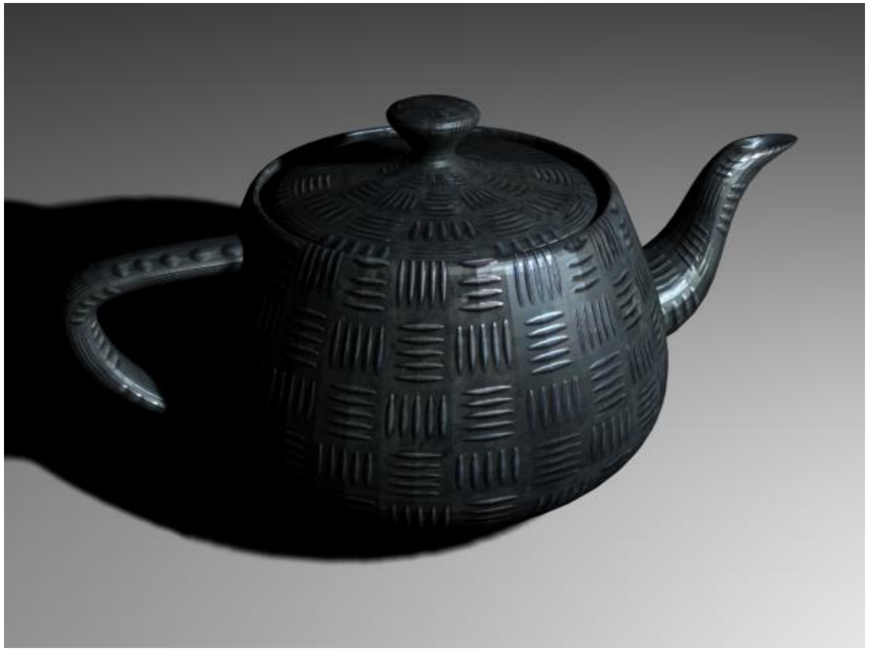
Texture

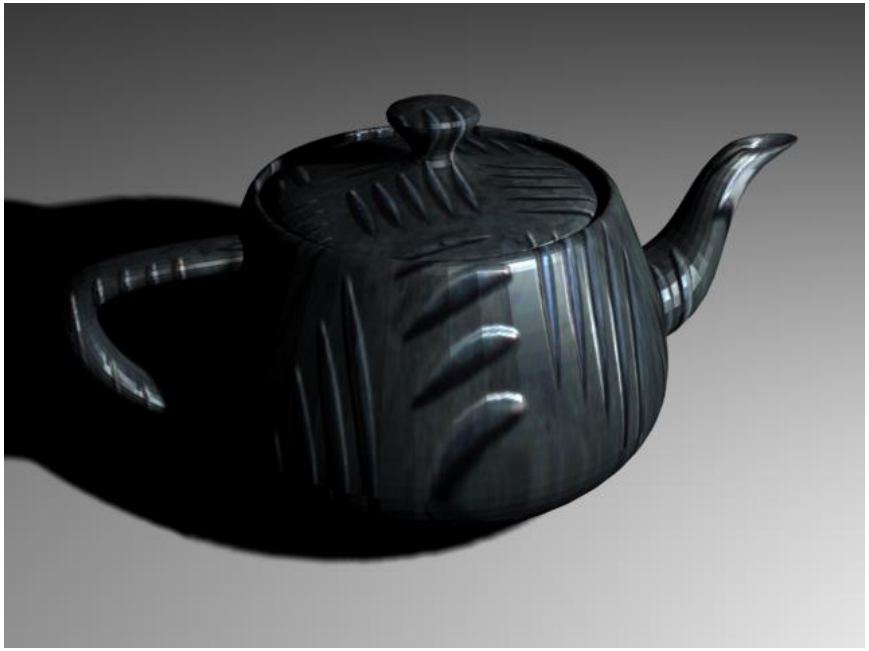
- Used to define object’s color appearance
  - 2D bitmap
  - Volumetric - texels
  - Procedural texture



Texture mapping

- Mapping between object space and 2D texture space





- New coordinate system: Texture coordinates

Texture mapping

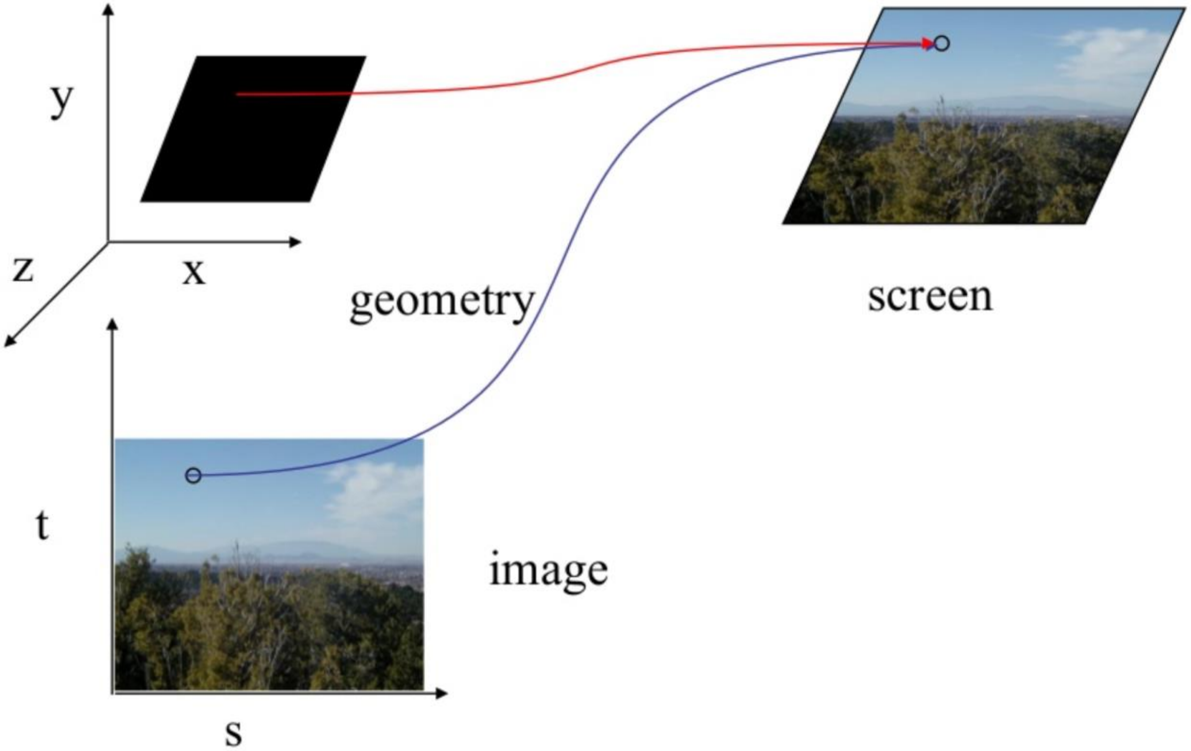
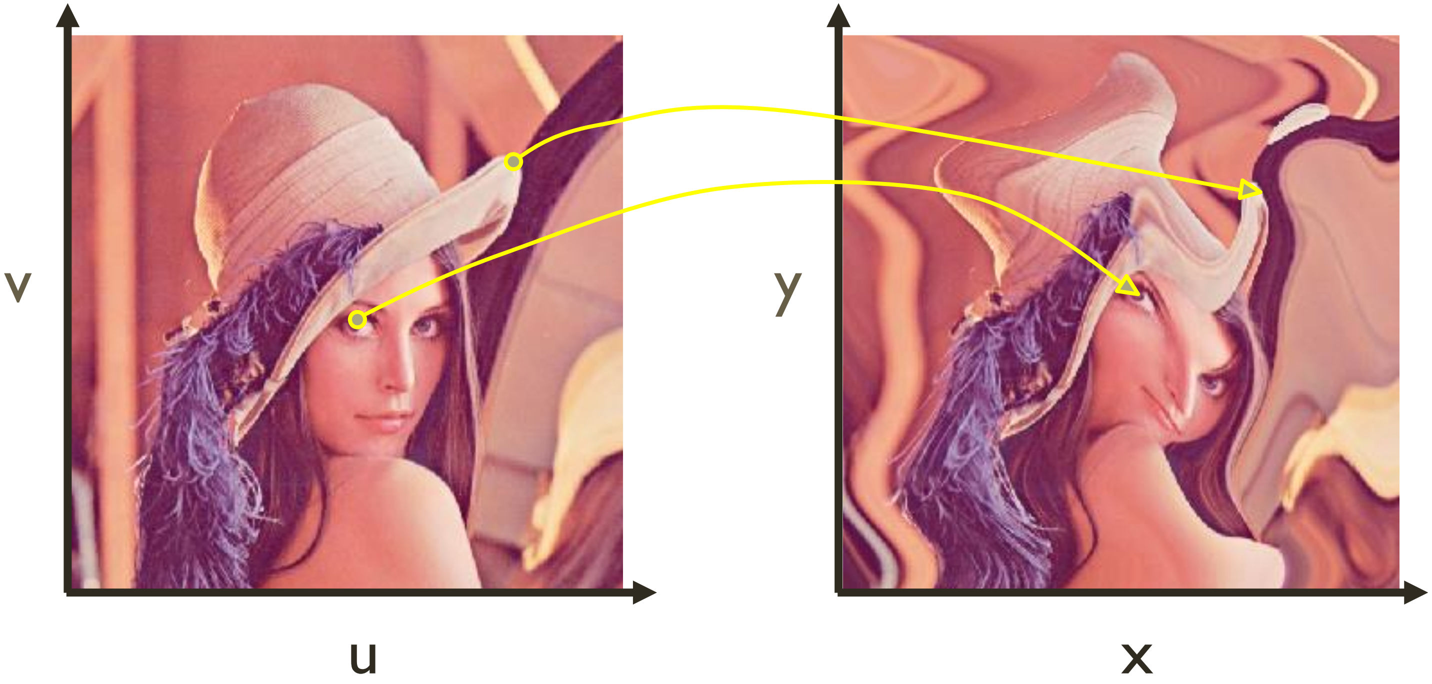


Image mapping

- Mapping
  - Forward
  - Inverse
- Resampling
- Filtering

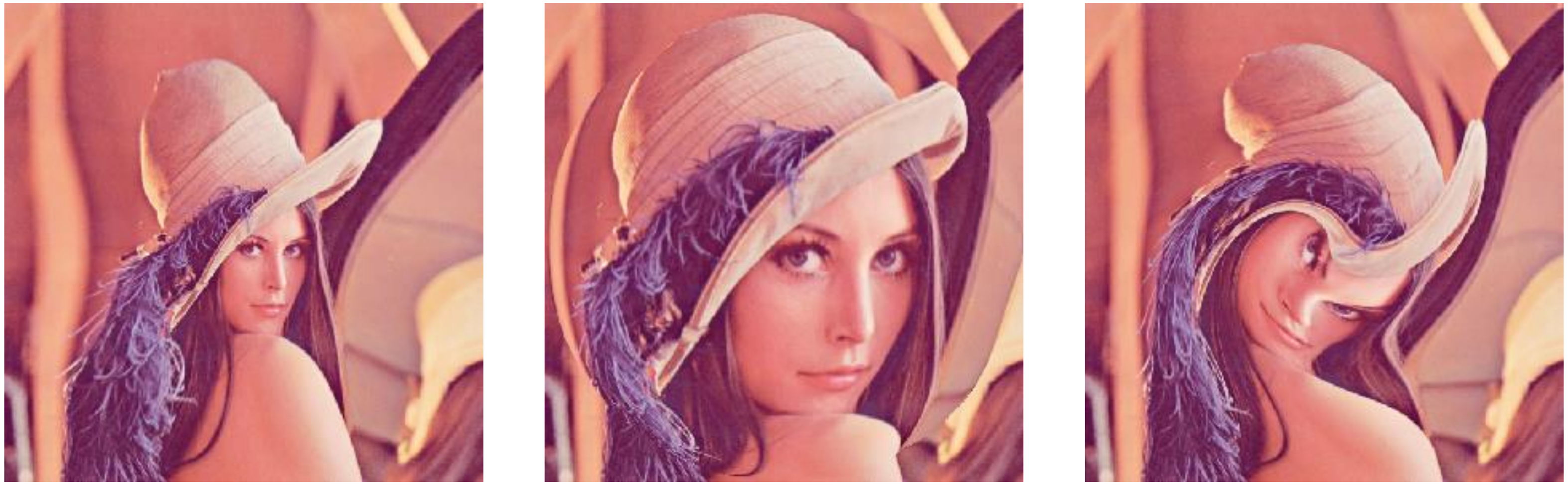
Mapping

- Define image transformation
  - Describe the destination (x, y) for every location (u, v) in the source (or vice-versa, if inversible)



Other mappings

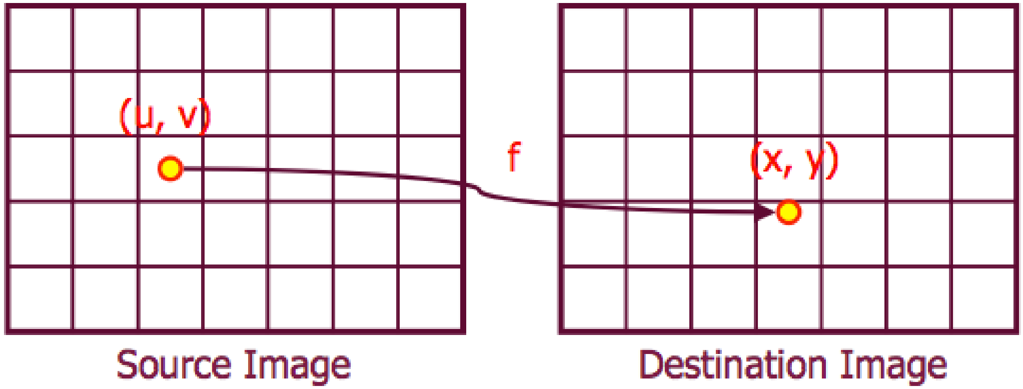
- Any function of u and v
  - x = f_x(u,v)
  - y = f_y(u,v)



Implementation

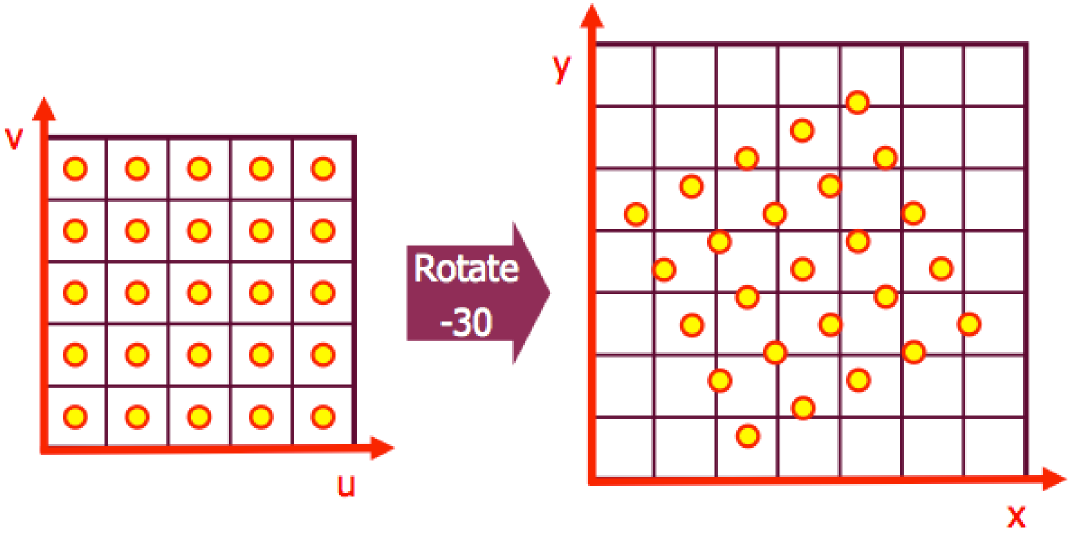
- Forward mapping

```
for(int u=0; u<umax; u++) {
  for(int v=0; v<vmax; v++) {
    float x = $f_x$ (u,v);
    float y = $f_y$(u,v);
    dst(x,y) = src(u,v);
  }
}
```



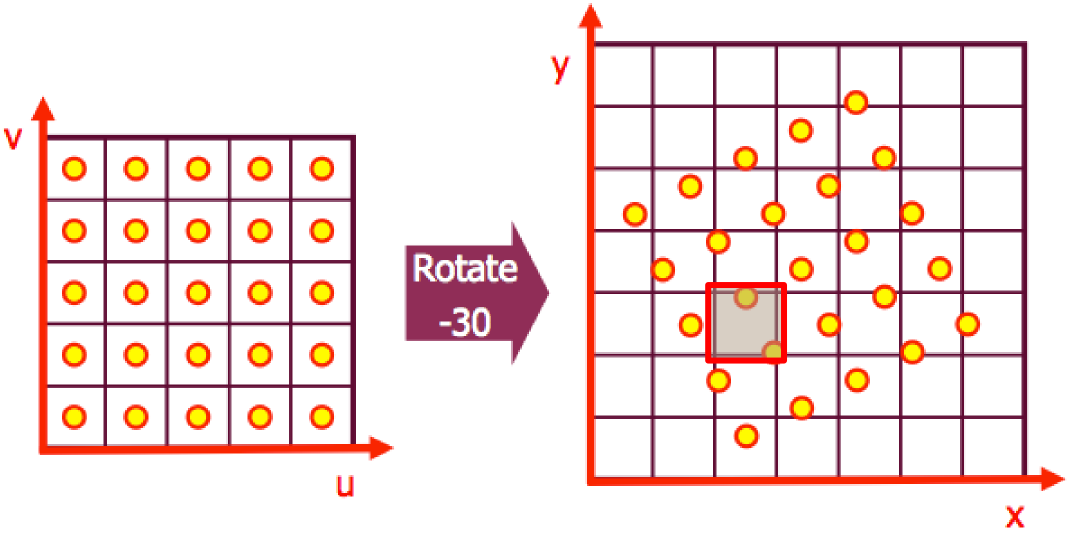
Forward mapping

- Iterate over source image
- But …



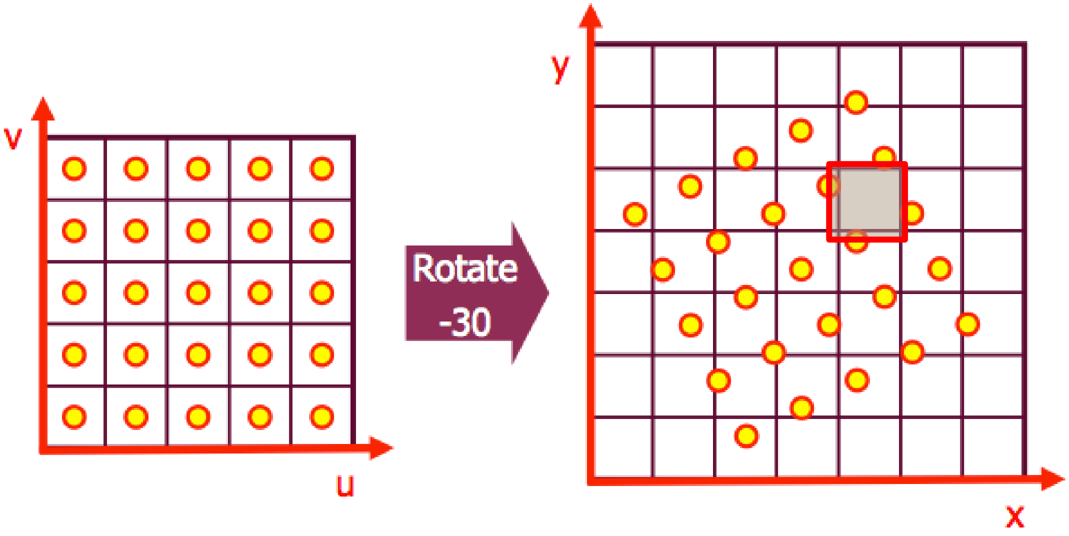
Forward mapping

- Iterate over source image
- Many pixels map on the same destination!



Forward mapping

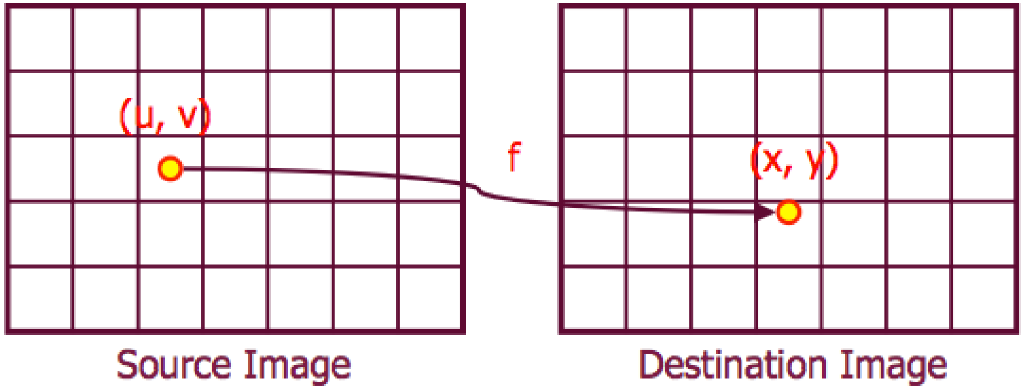
- Iterate over source image
- Some pixels will not be covered!



Implementation 2

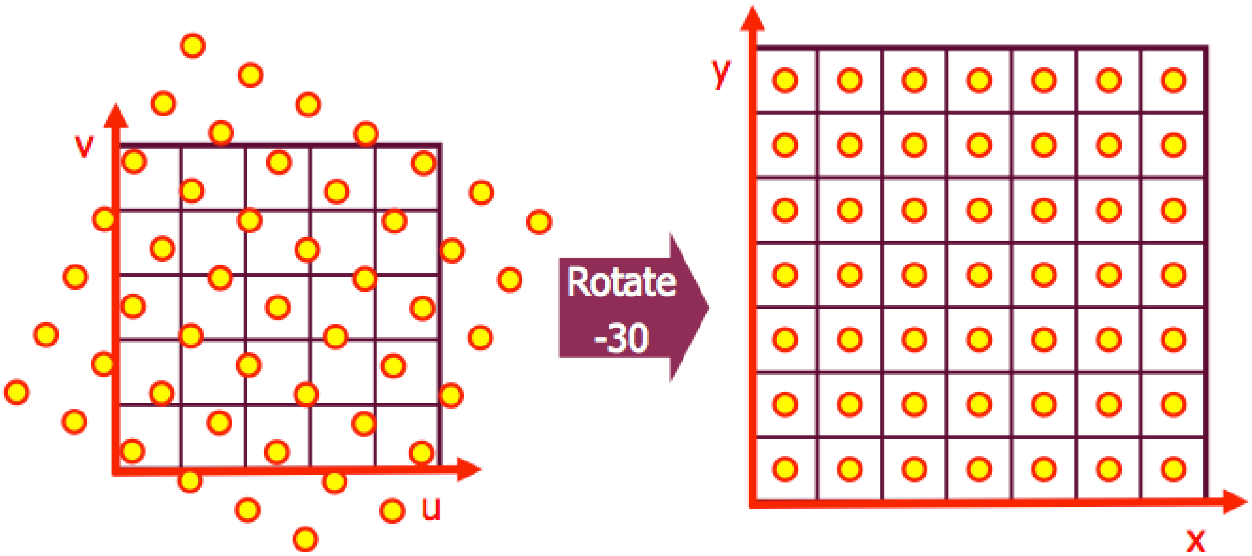
- Inverse mapping

```
for(int x=0; x<xmax; x++) {
  for(int y=0; y<ymax; y++) {
    float u = $f_x^{-1}$ (x,y);
    float v = $f_y^{-1}$ (x,y);
    dst(x,y) = *resample_src*(u,v);
  }
}
```



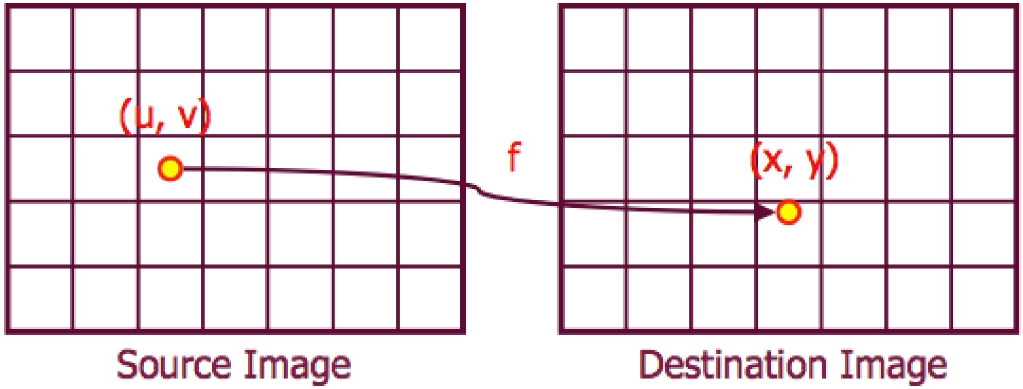
Inverse mapping

- Iterate over destination image
  - Must **resample** source
  - Much simpler but may oversample



Resampling

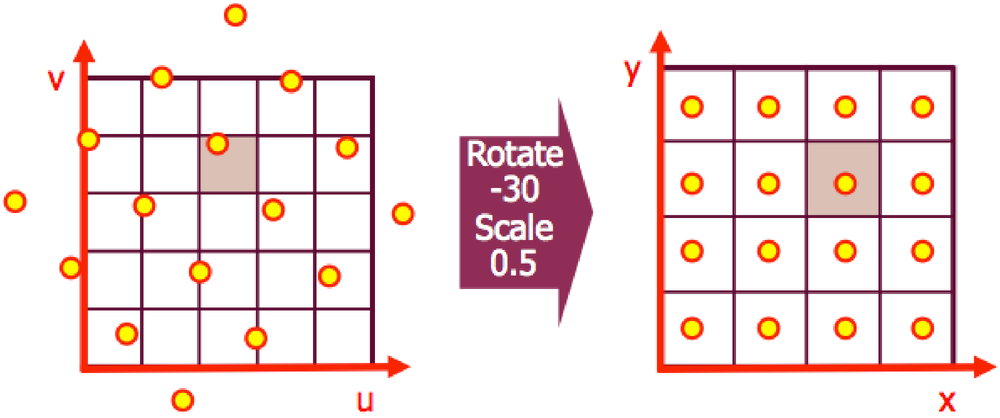
- Evaluate source image at arbitrary (u,v)
- (u,v) coordinates are generally not integer



Nearest neighbor

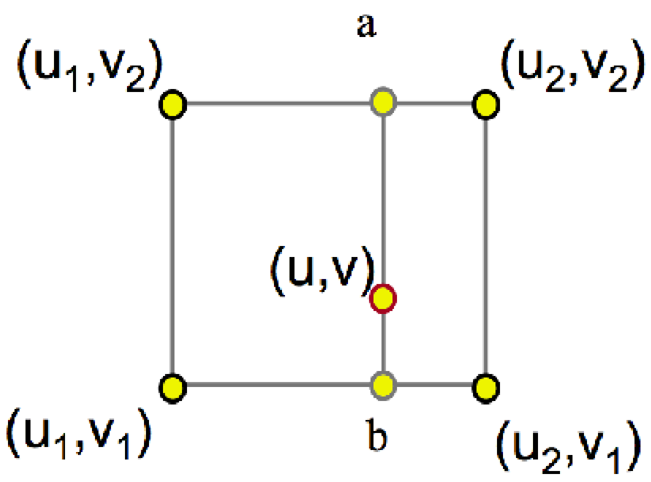
- Take value of closest pixel
- Fast! Low quality

```
int iu = trunc(u+0.5du);
int iv = trunc(v+0.5dv);
dst(x, y) = src(iu, iv);
```



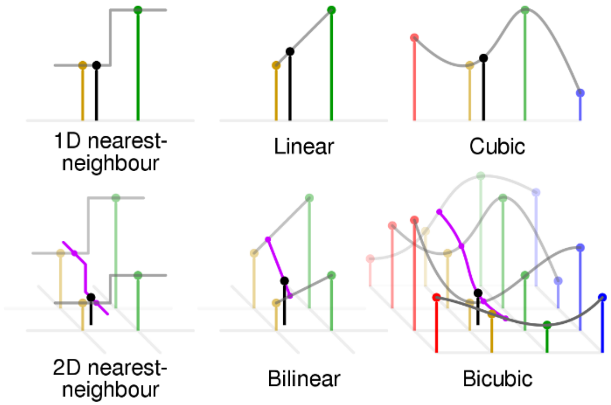
Bilinear filtering

- Bilinearly interpolate four closest pixels
  - a = linear interpolation of src(u1, v2) and src(u2, v2)
  - b = linear interpolation of src(u1, v1) and src(u2, v1)
  - dst(x, y) = linear interpolation of "a" and "b"
- Reasonably Fast. Good quality.

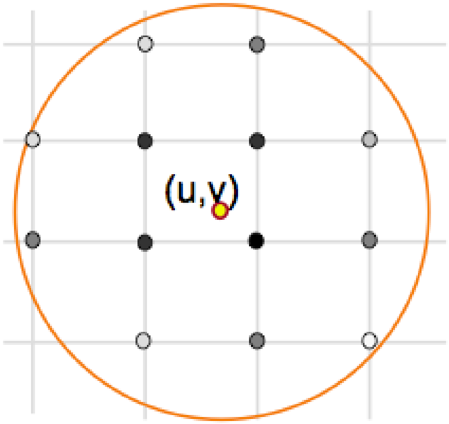


Other filters

- Bicubic Filtering
  - Considers 4x4 pixels (16 pixels)
  - Smoother, less artifacts
  - Computationally expensive

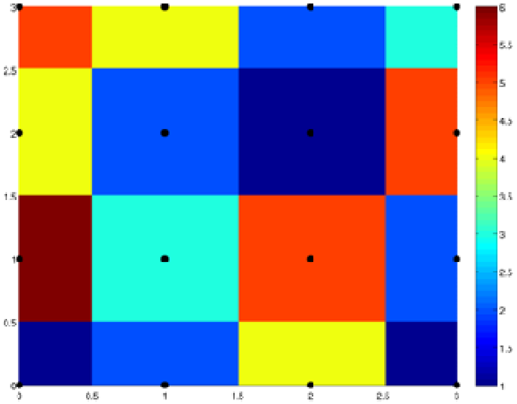


- Gaussian Filtering
  - Uses weighted sum of neighborhood
  - Weights are normalized using Gaussian function

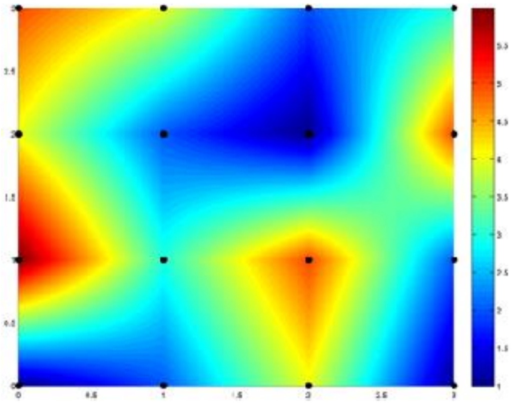


Filtering comparison

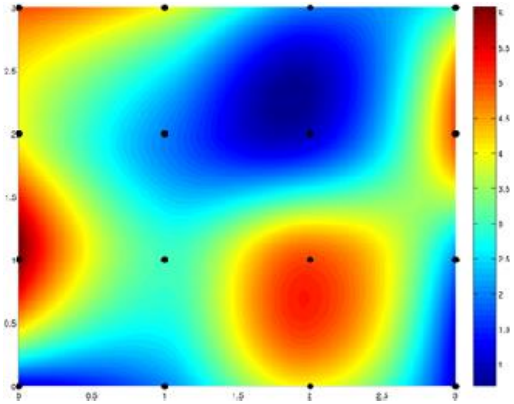
- Comparison of resampling quality



Nearest

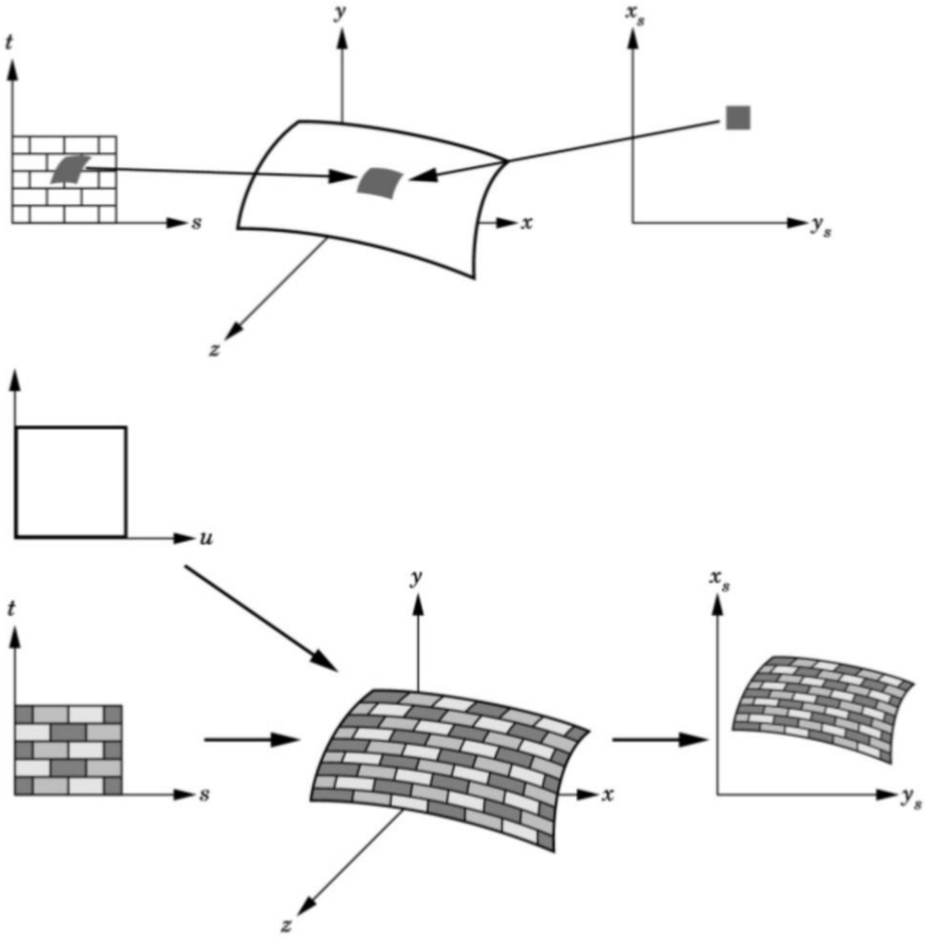


Bilinear



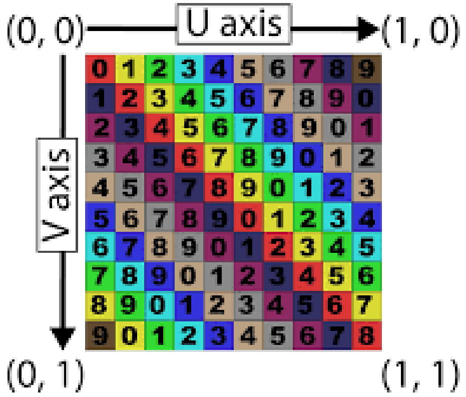
Bicubic

Texture mapping

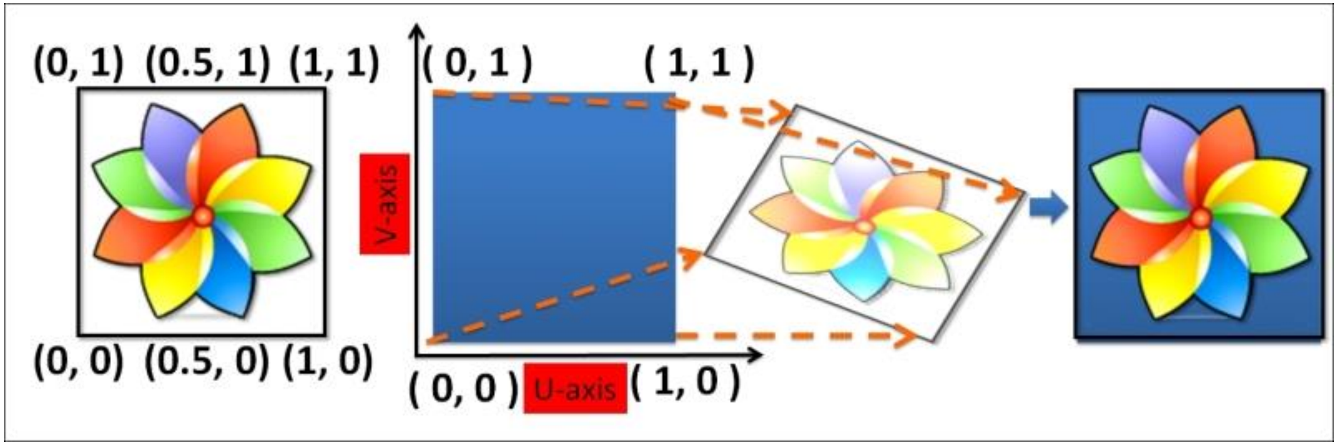


Texture mapping

Normalized Texture Coordinates



0	1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9	0
2	3	4	5	6	7	8	9	0	1
3	4	5	6	7	8	9	0	1	2
4	5	6	7	8	9	0	1	2	3
5	6	7	8	9	0	1	2	3	4
6	7	8	9	0	1	2	3	4	5
7	8	9	0	1	2	3	4	5	6
8	9	0	1	2	3	4	5	6	7
9	0	1	2	3	4	5	6	7	8



Texture mapping

2D texture space

↓ *parameterization*

3D object space

↓ *model transform*

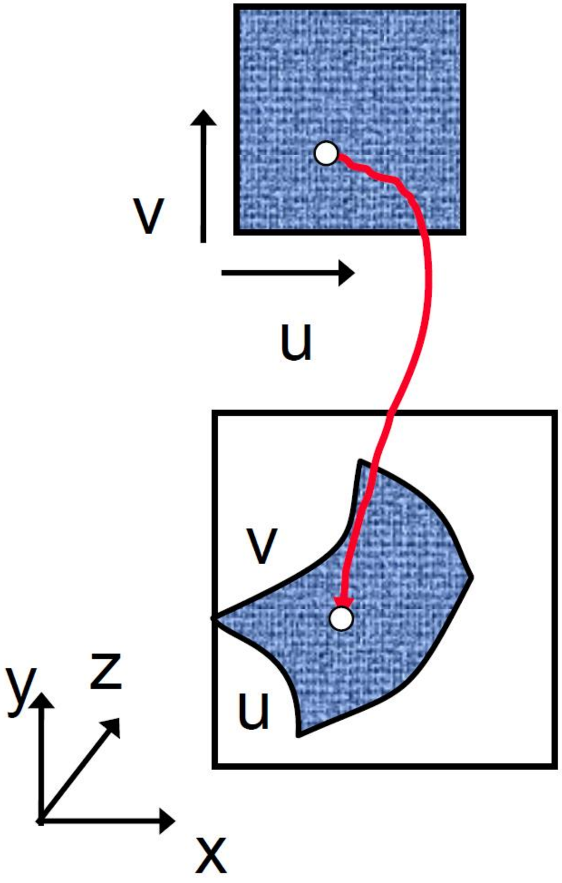
3D world space

↓ *viewing transform*

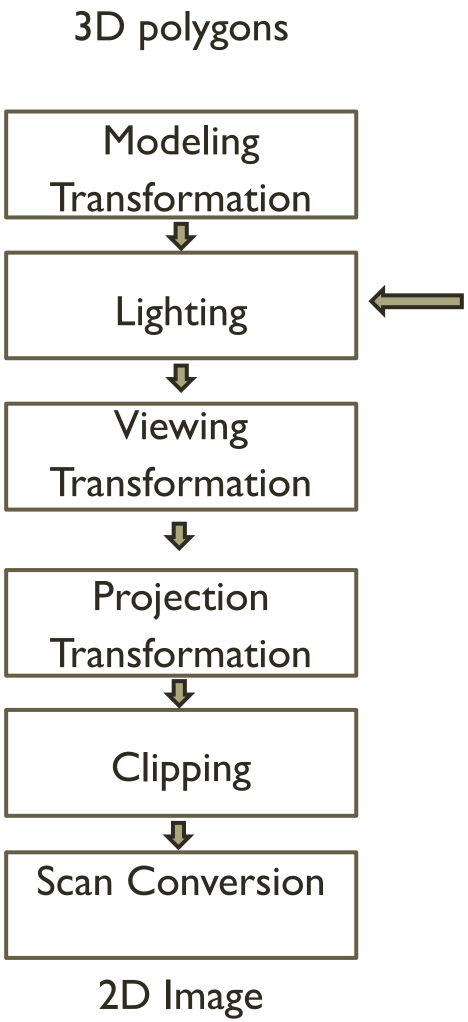
3D camera space

↓ *projection*

2D image space (screen)

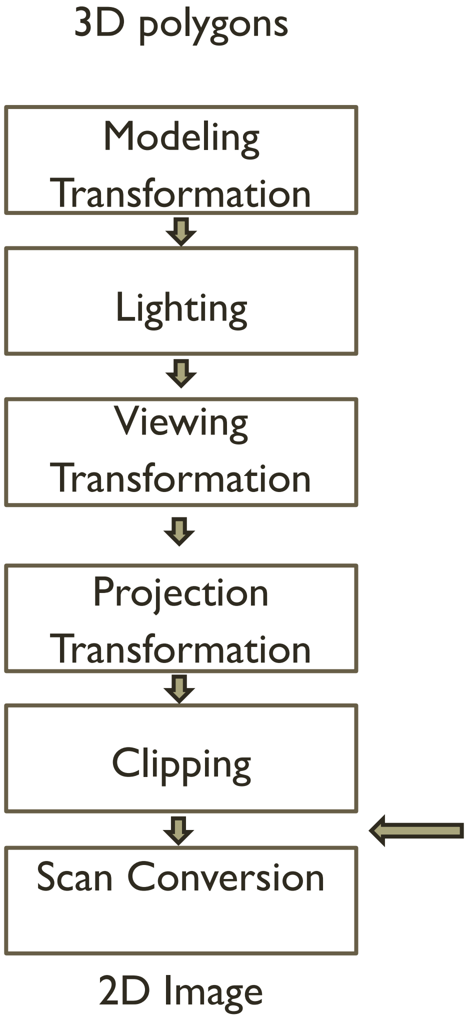


3D rendering pipeline



- Lighting + color from image
  - Could be implemented here

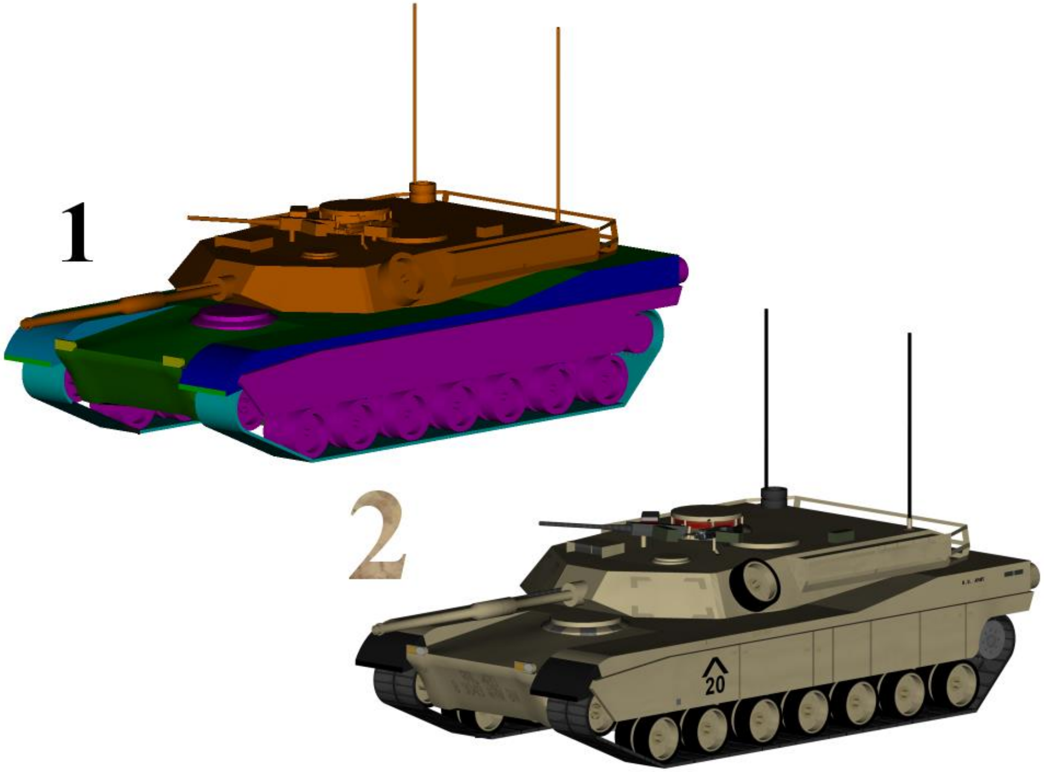
3D rendering pipeline



- GPU Texture mapping
  - Fragment shader implementation

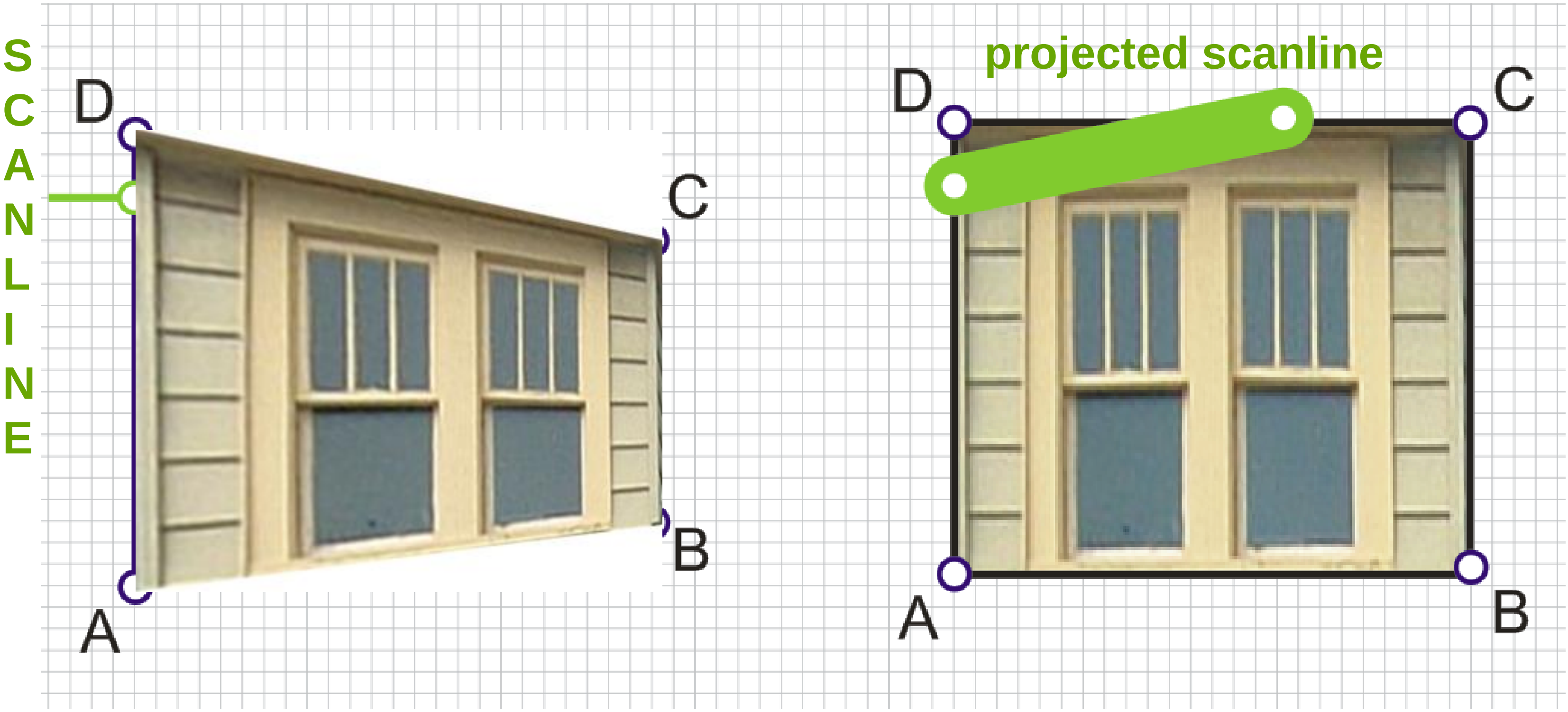
Texture mapping

- Add visual detail to surfaces of 3D objects
  - 1) Parameterized mesh
  - 2) Final textured model



Intermediate pixels

- Remember polygon rasterization



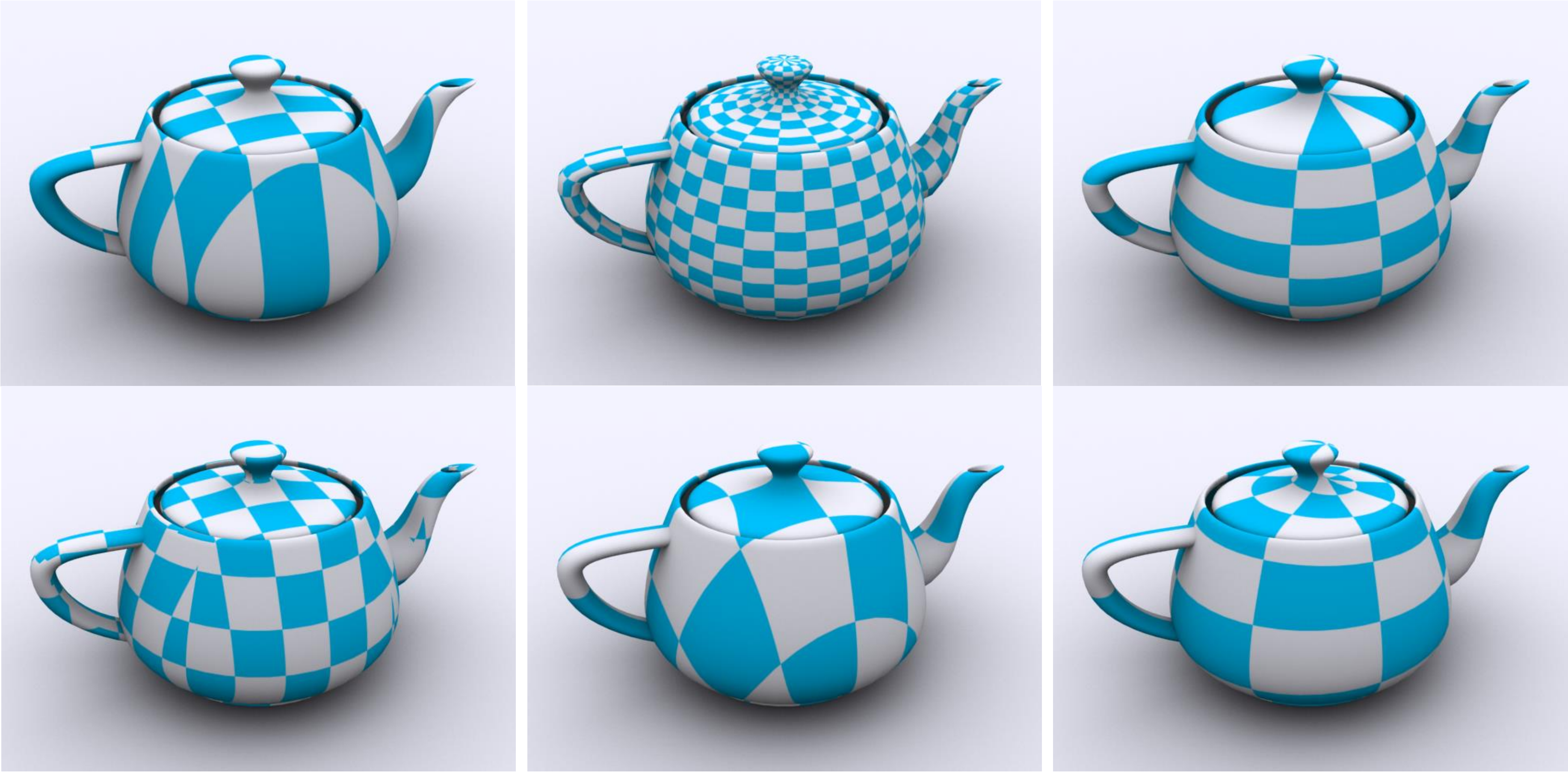
Screen space

Texture space

Texture usage

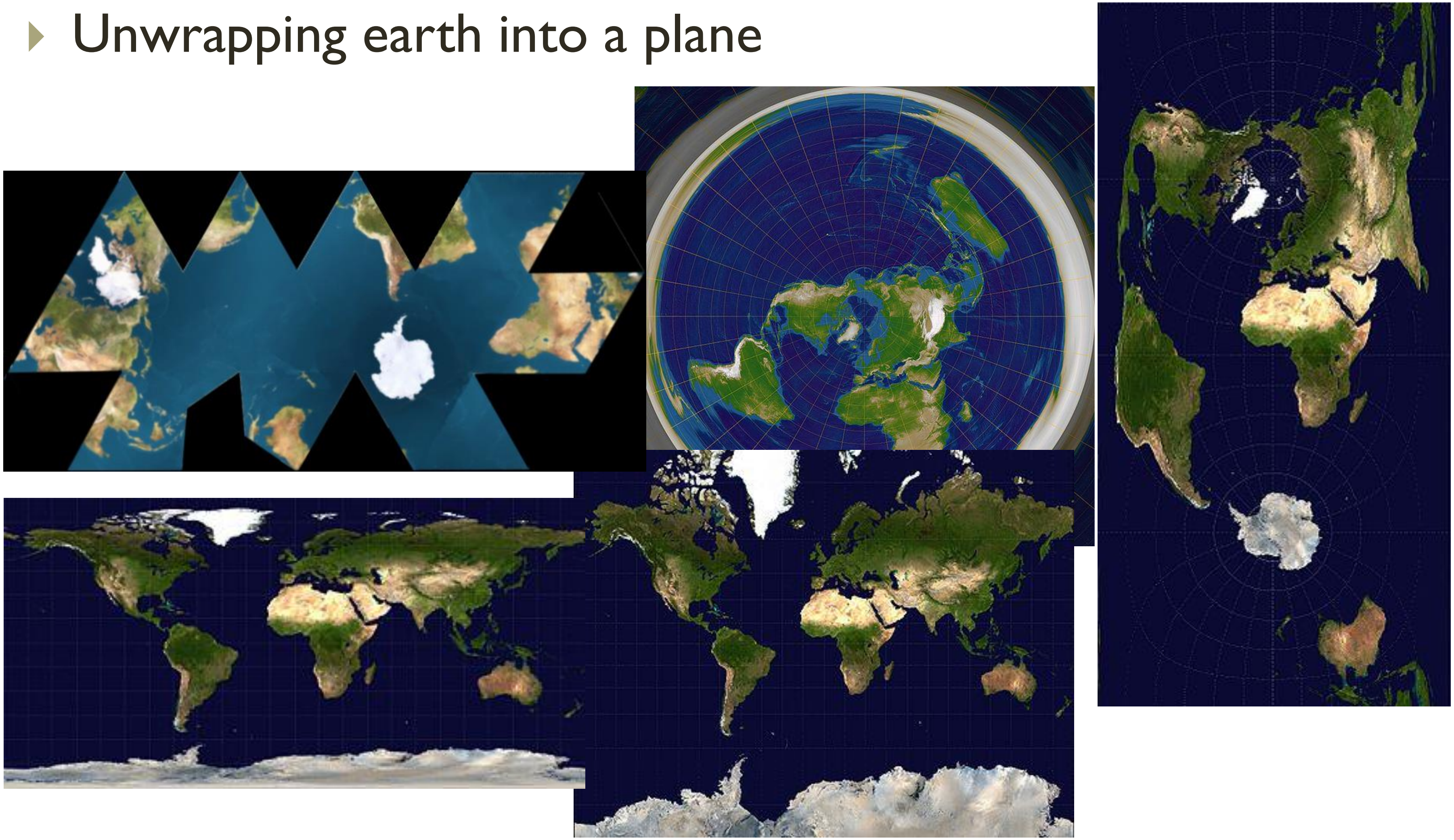
- object diffuse color
  - patterns, decals
- modulate surface properties
  - bumps, displacements
- modulate lighting properties
  - e.g. shininess
- simulate physical phenomena
  - reflection, refraction, global illumination

Texture mapping

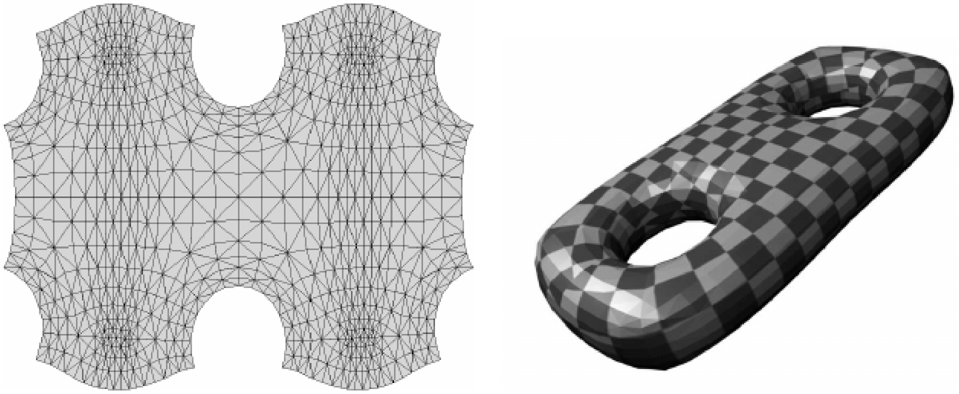


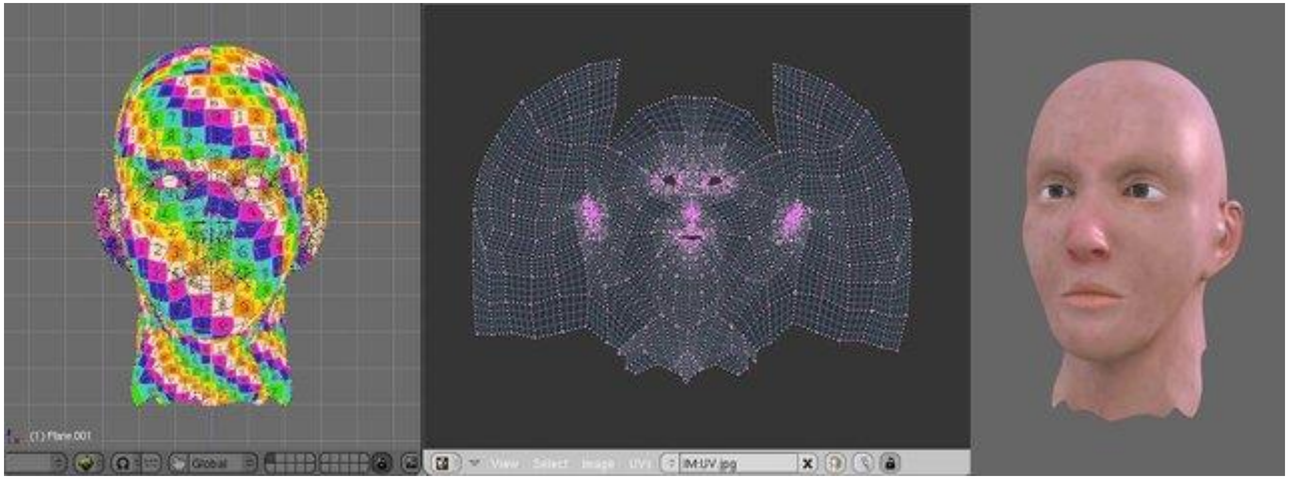
Example – cartography

- Unwrapping earth into a plane



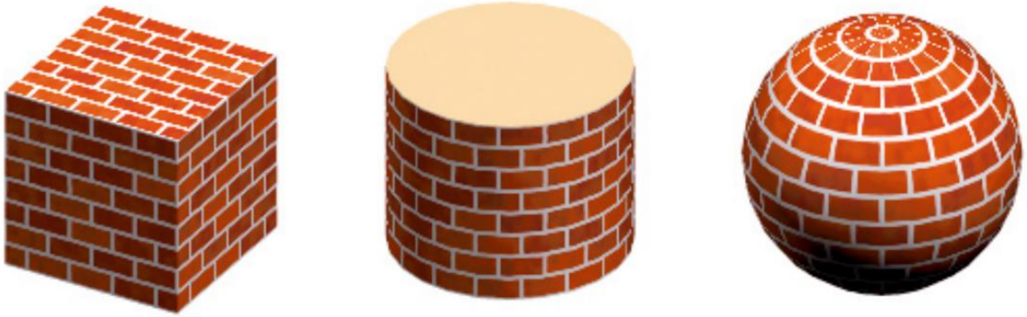
Parameterization





Parameterization

- Implicit parametrization by geometrical primitives

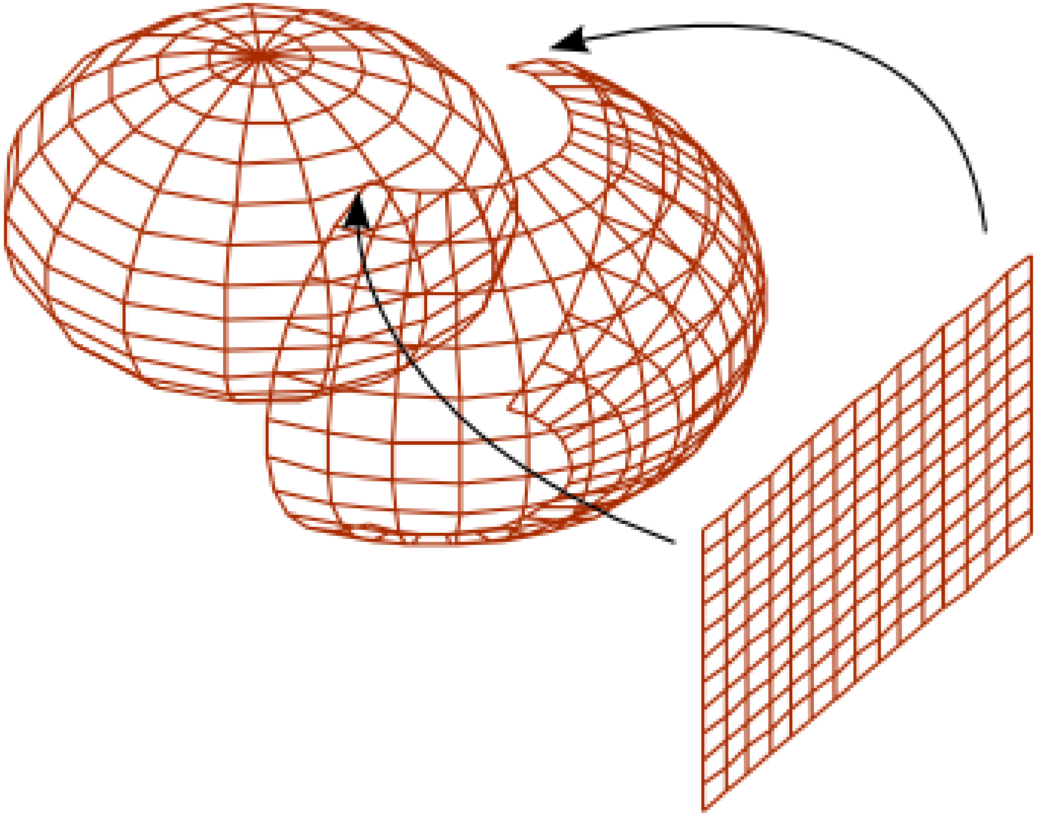


Parameterization

- XYZ to UV for sphere:

$$u = \frac{x}{\sqrt{x^2 + y^2 + z^2}}$$

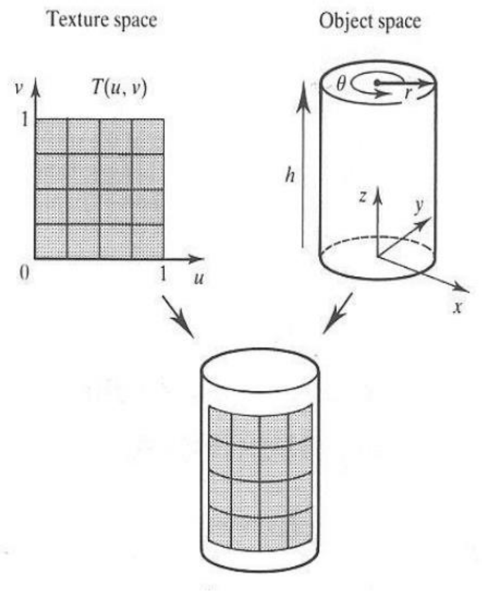
$$v = \frac{y}{\sqrt{x^2 + y^2 + z^2}}$$



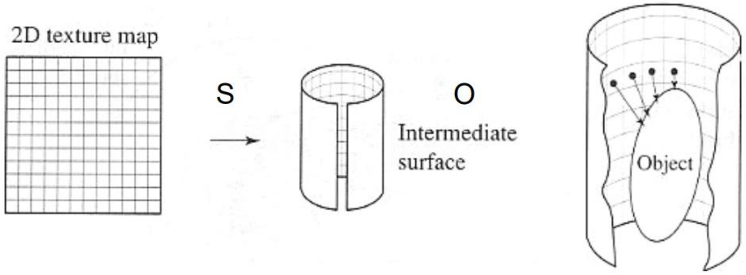
http://tobias.preclik.de/codeblog/?p=9

Parameterization

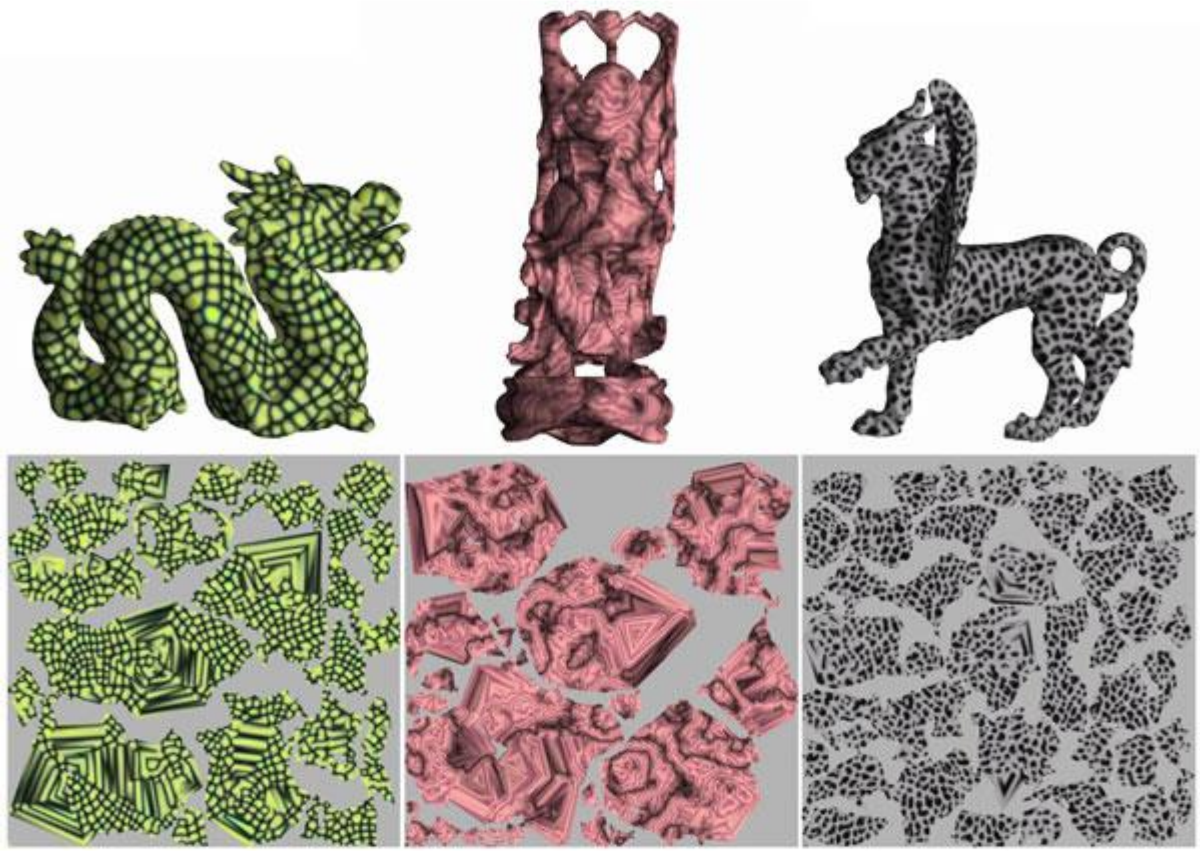
- Parameterization using an intermediate surface



$$x_o = r\cos\theta$$
$$y_o = r\sin\theta$$
$$z_o = h$$

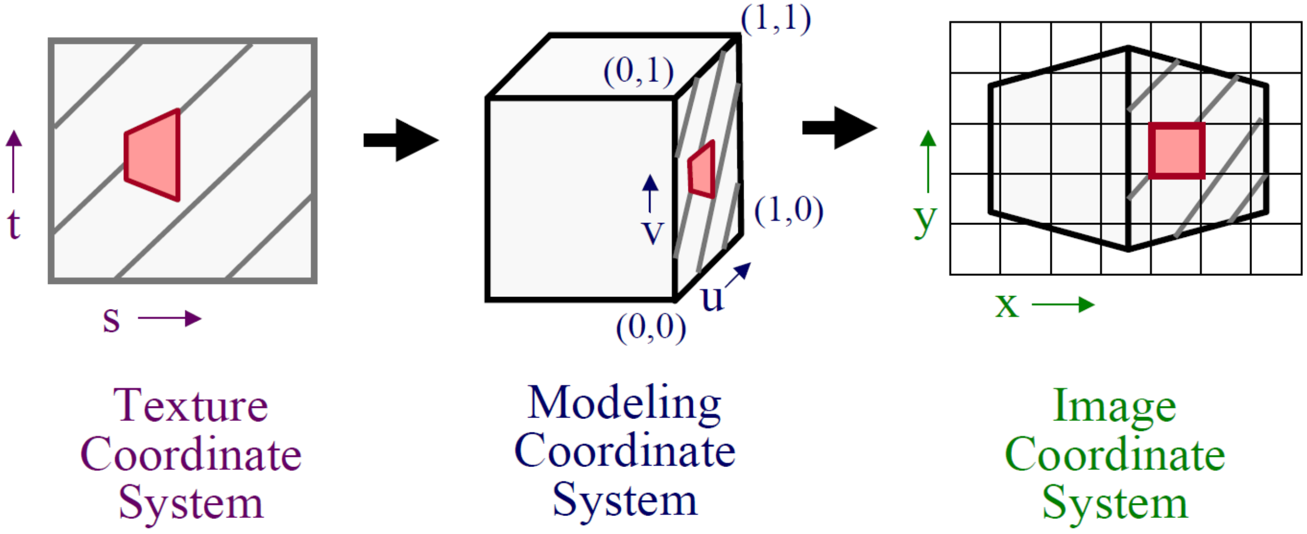


Parameterization



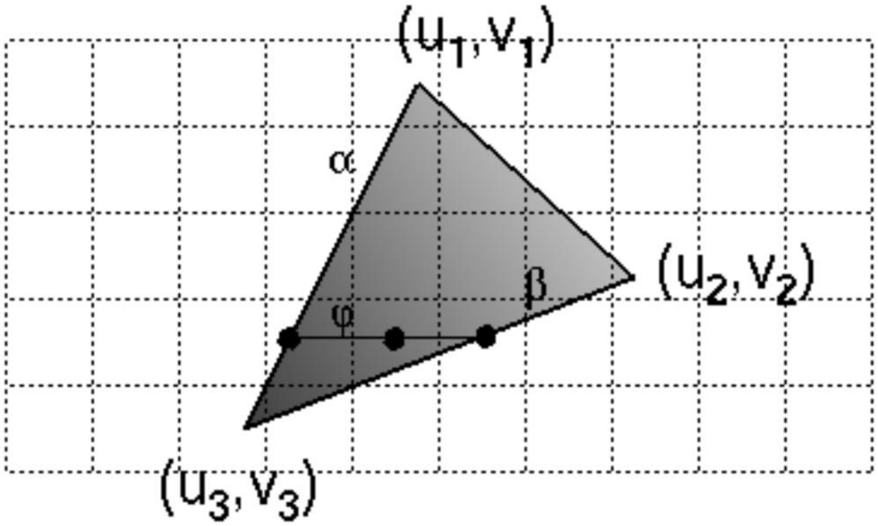
Texture mapping

- When drawing pixels, map from ...
  - image coordinate system (x,y) to
  - modeling coordinate system (u,v) to
  - texture coordinate system (t,s)



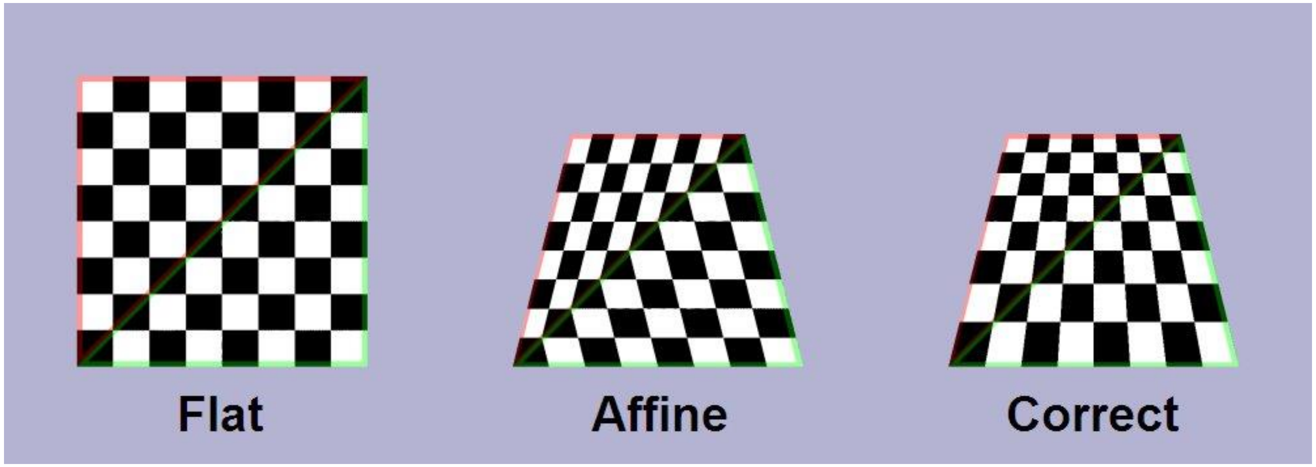
UV mapping

- Scan conversion
  - Interpolate texture coordinates down/across scan lines
  - Distort due to bilinear interpolation approximation
  - Cut polygons into smaller ones, or
  - Perspective divide at each pixel



Perspective correction

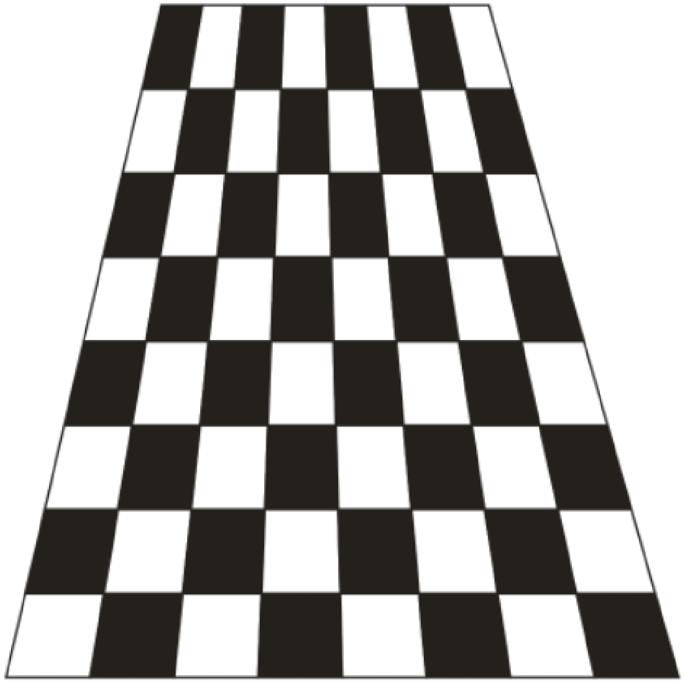
- Scan conversion
  - Interpolate texture coordinates down/across scan lines
  - Distort due to bilinear interpolation approximation
  - Cut polygons into smaller ones, or
  - Perspective divide at each pixel



Perspective correction

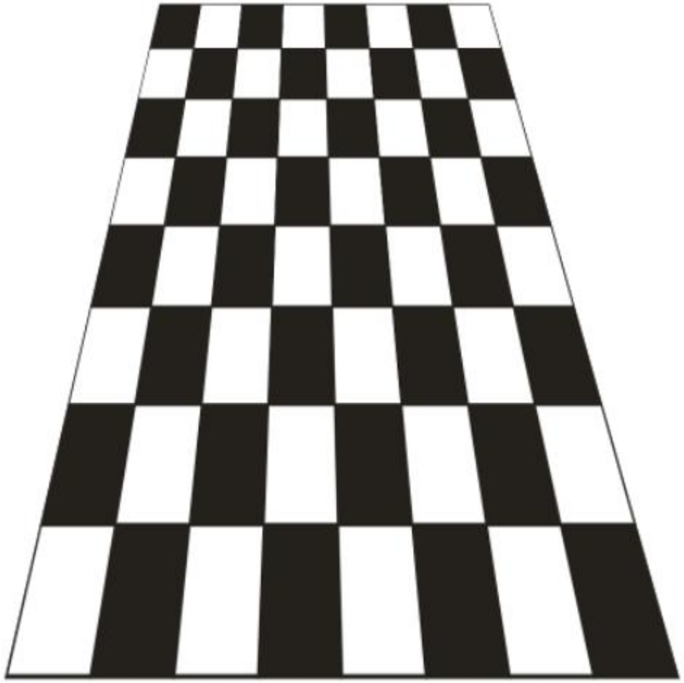
INCORRECT

$$T = (1-t)P + tQ$$



CORRECT

$$T = (1-t)\frac{P}{P_z} + t\frac{Q}{Q_z}$$

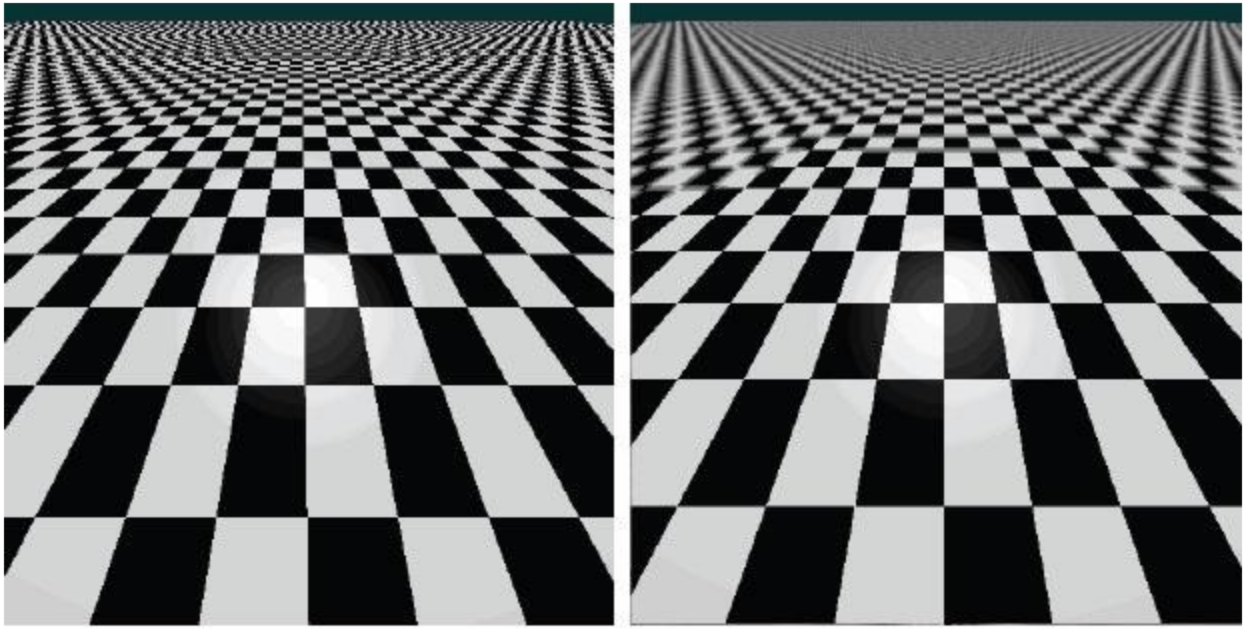


Overview

- Texture mapping
  - 3D Models with texture coordinates
  - UV map parametrization
  - Diffuse color textures
  - Other textures
    - Bump mapping
    - Environment mapping
- Aliasing
- Anti-aliasing
  - Supersampling
  - Mip Maps

Aliasing

- "Moire pattern"
- Nyquist frequency
  - Sampling frequency >= 2x signal frequency

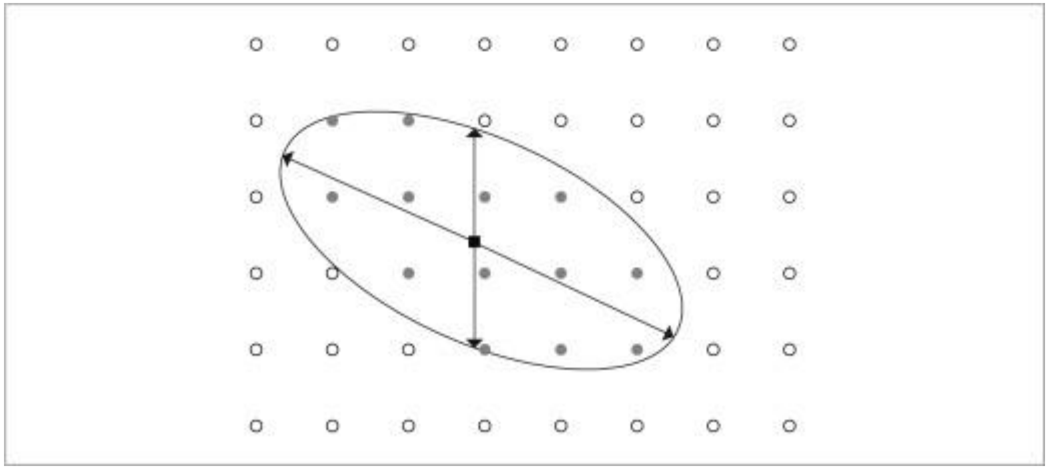


aliasing effects

anti-aliasing by over-sampling

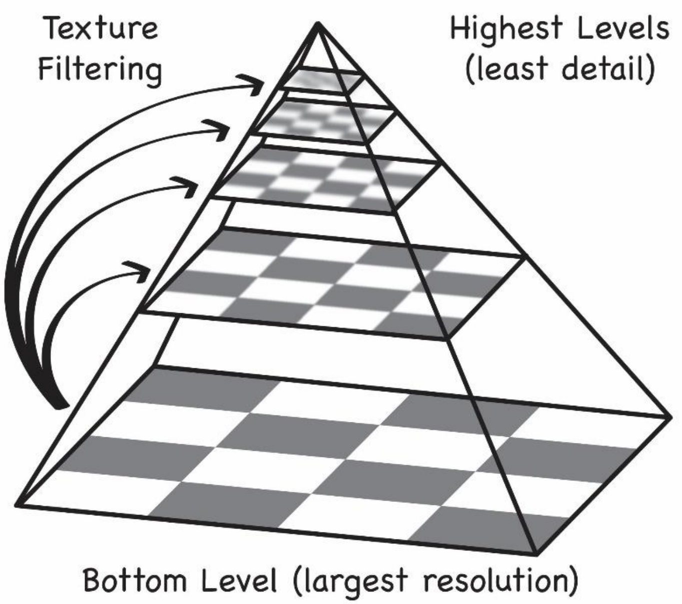
Texture filtering

- Ideally, use elliptically shaped convolution filter
- In practice we use rectangles



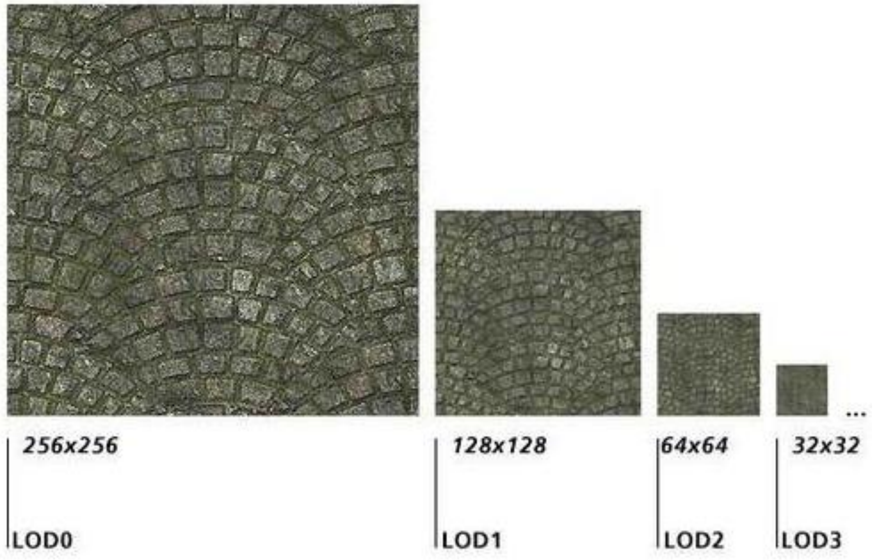
Texture filtering

- Size of filter depends on projective wrap
- Images can be pre-filtered
  - Mip Maps
  - Summed area tables

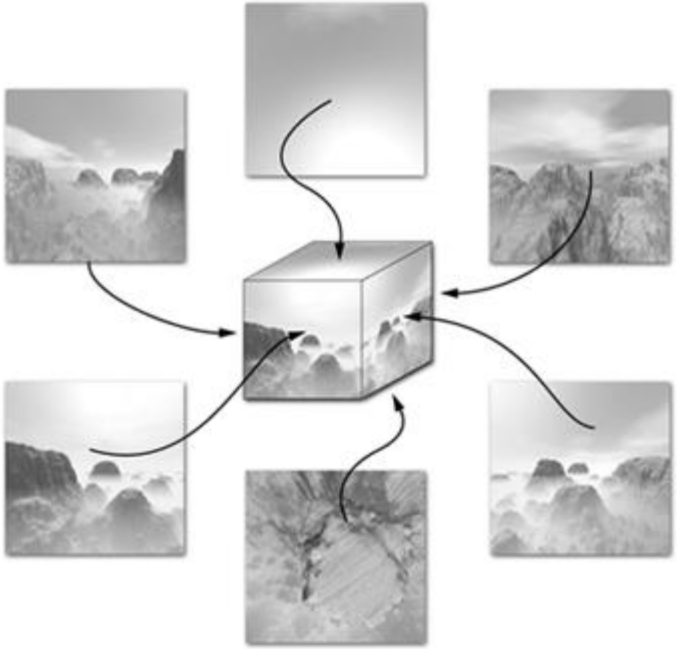

Texture Filtering
Highest Levels (least detail)
Bottom Level (largest resolution)

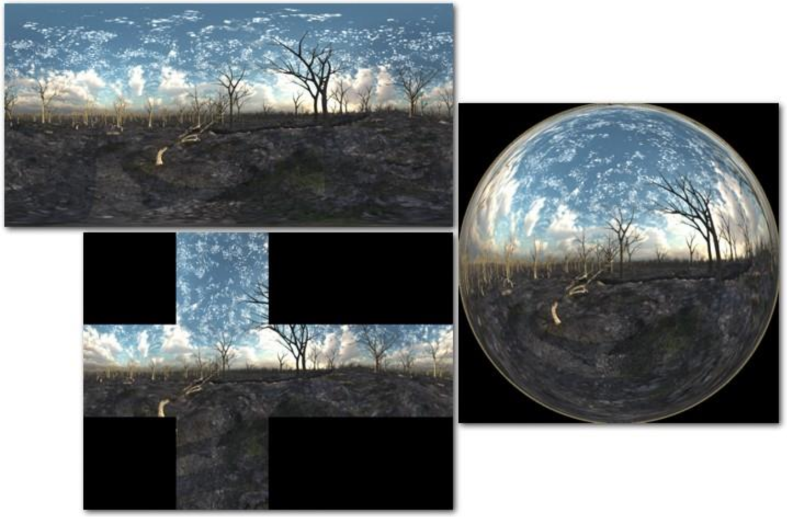
Mip maps

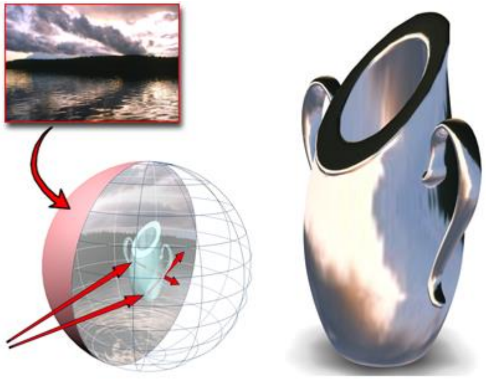
- Keep textures pre-filtered at multiple resolutions
  - For each pixel, linearly interpolate between two closest levels (e.g., trilinear filtering)
- Fast and easy for hardware



Environment mapping





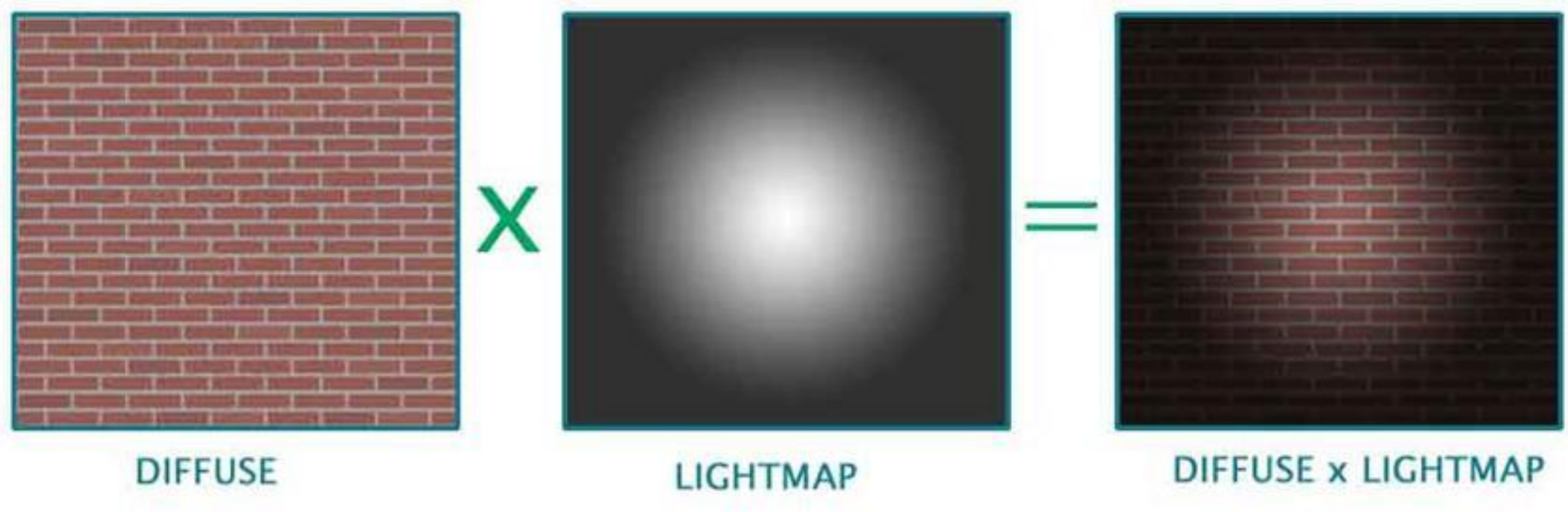


https://www.youtube.com/watch?v=LOeEfkzZ1ps

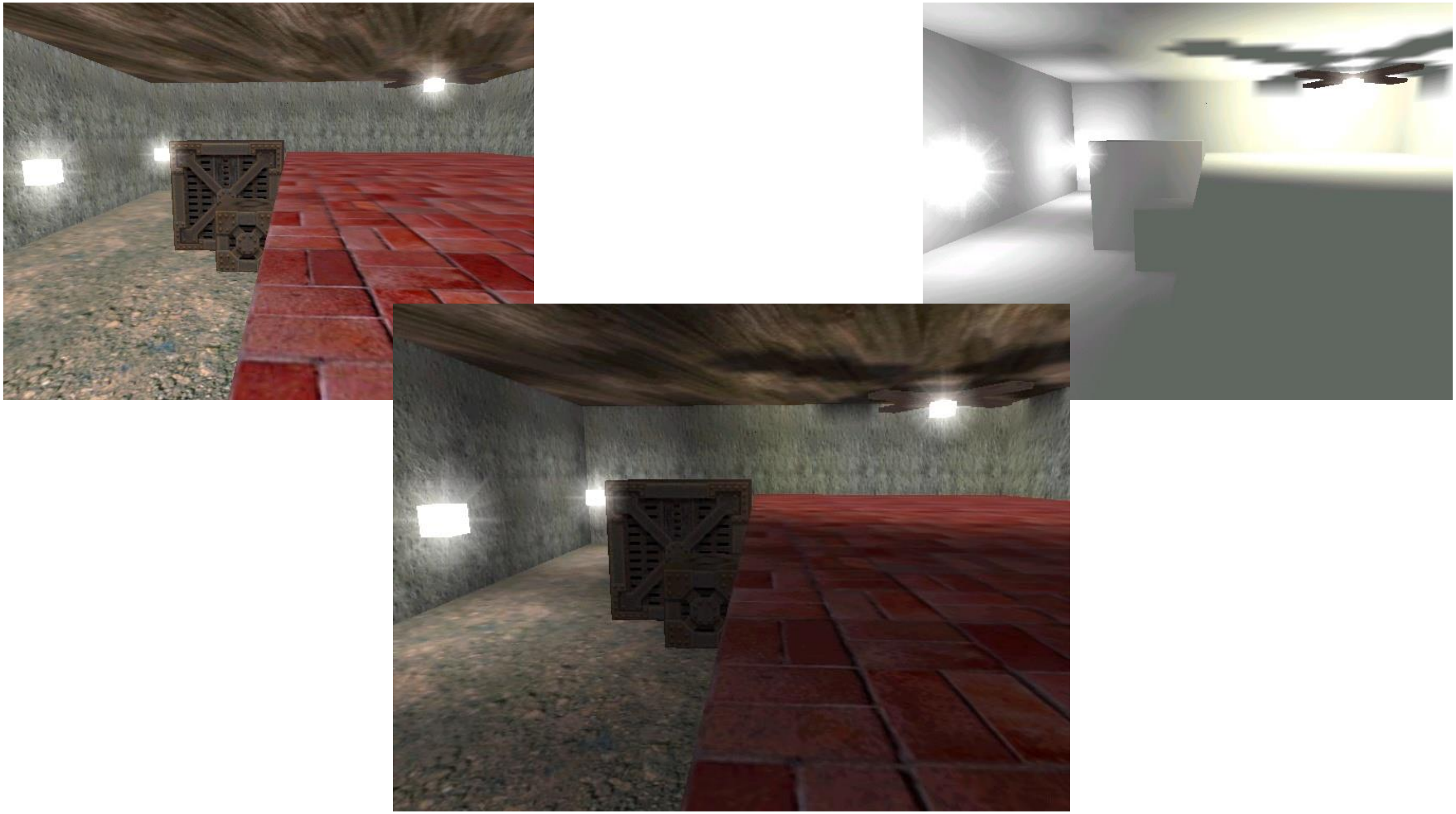
Light maps

Light maps

- Pre-computed high-quality lighting
- Stored into special texture (light map)
- Light map combined with the texture
- Texture baking (permanent)



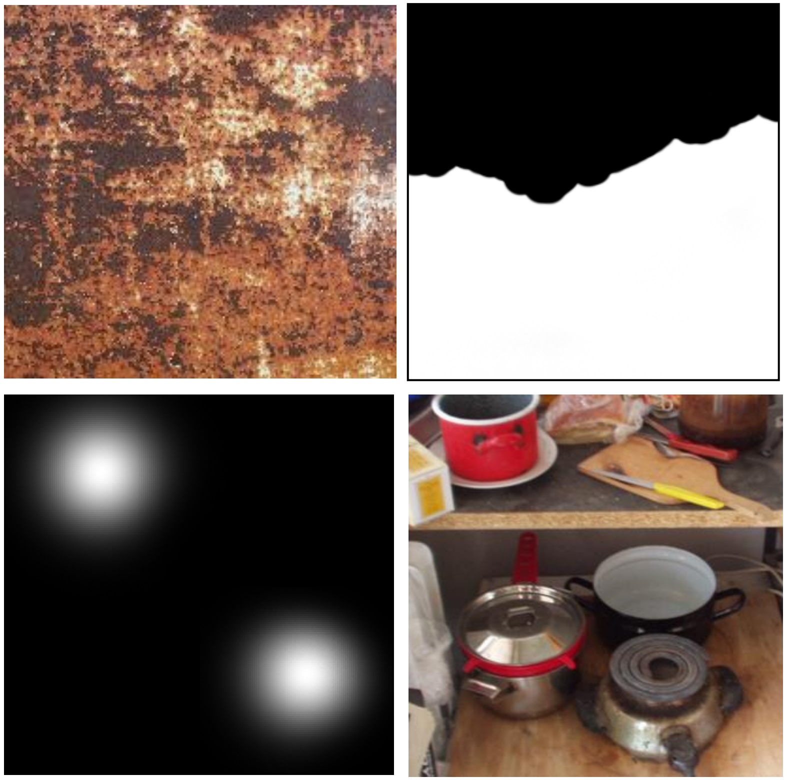
Light maps

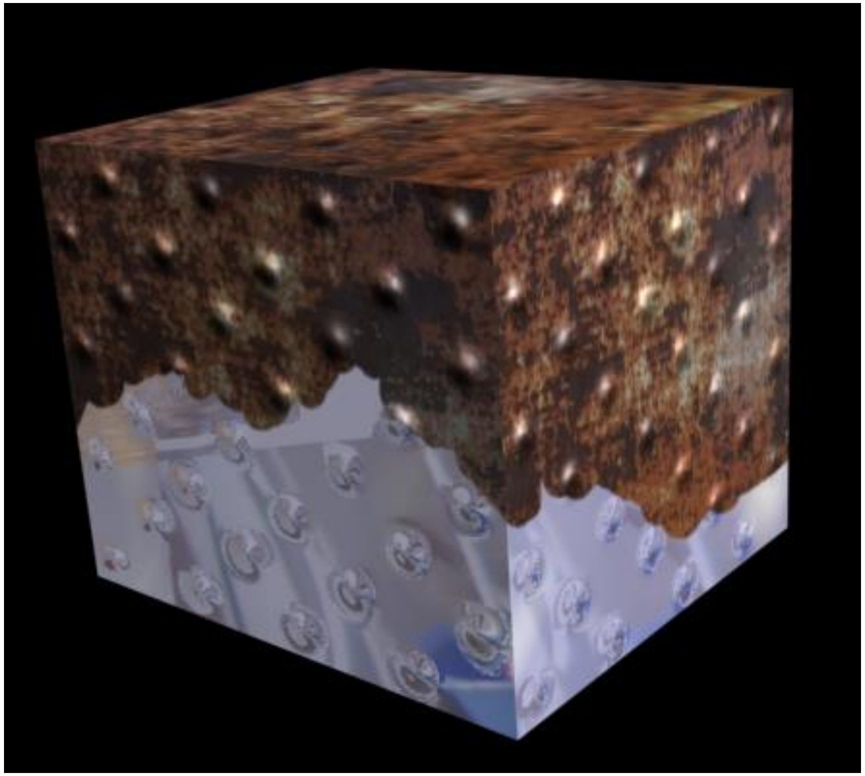


http://www.cs.bath.ac.uk/~pjw/NOTES/pics/lightmap.html

Multitexturing

- Combine multiple textures



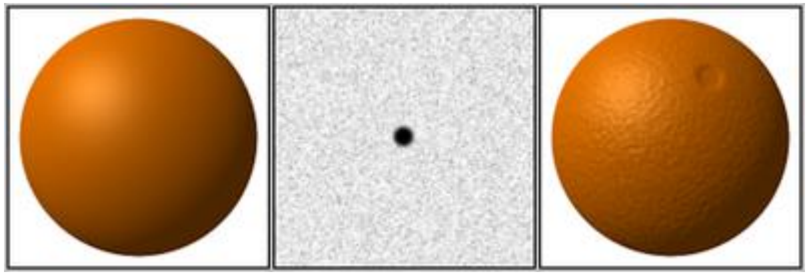


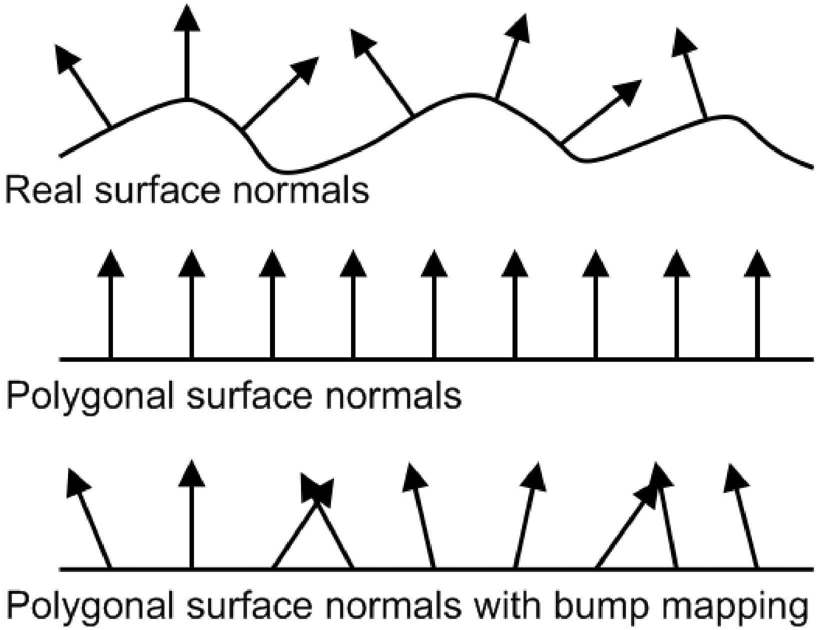
Overview

- Advanced Shading and Mapping
  - Bump Mapping
  - Normal Mapping
  - Displacement Mapping
  - Vector Displacement Mapping

Bump mapping

- A modified surface normal is calculated from the height map
- Modified normal is used during shading
- Geometry is not altered

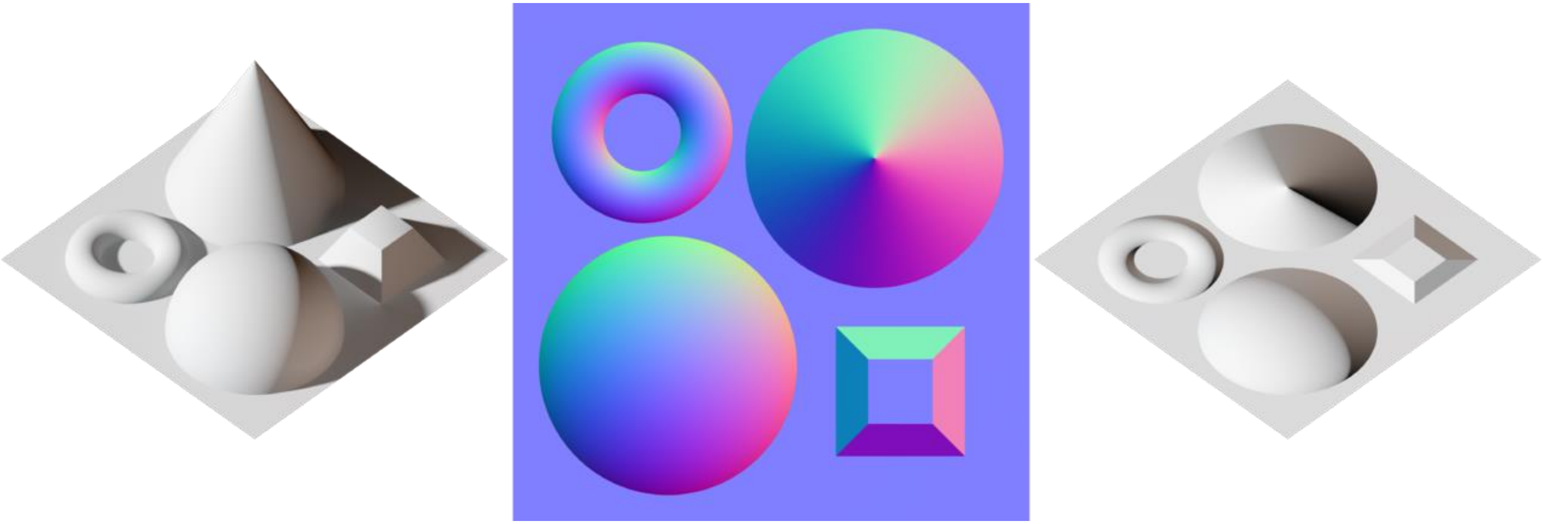



Real surface normals
Polygonal surface normals
Polygonal surface normals with bump mapping

Normal Mapping

- Fake lighting of bumps and dents
- “Dot3 bump mapping”
- Add lighting details without additional geometry
- Store normals from high-polygon object in texture
- Encode X,Y,Z as R,G,B color information

Normal Mapping



Normal Mapping

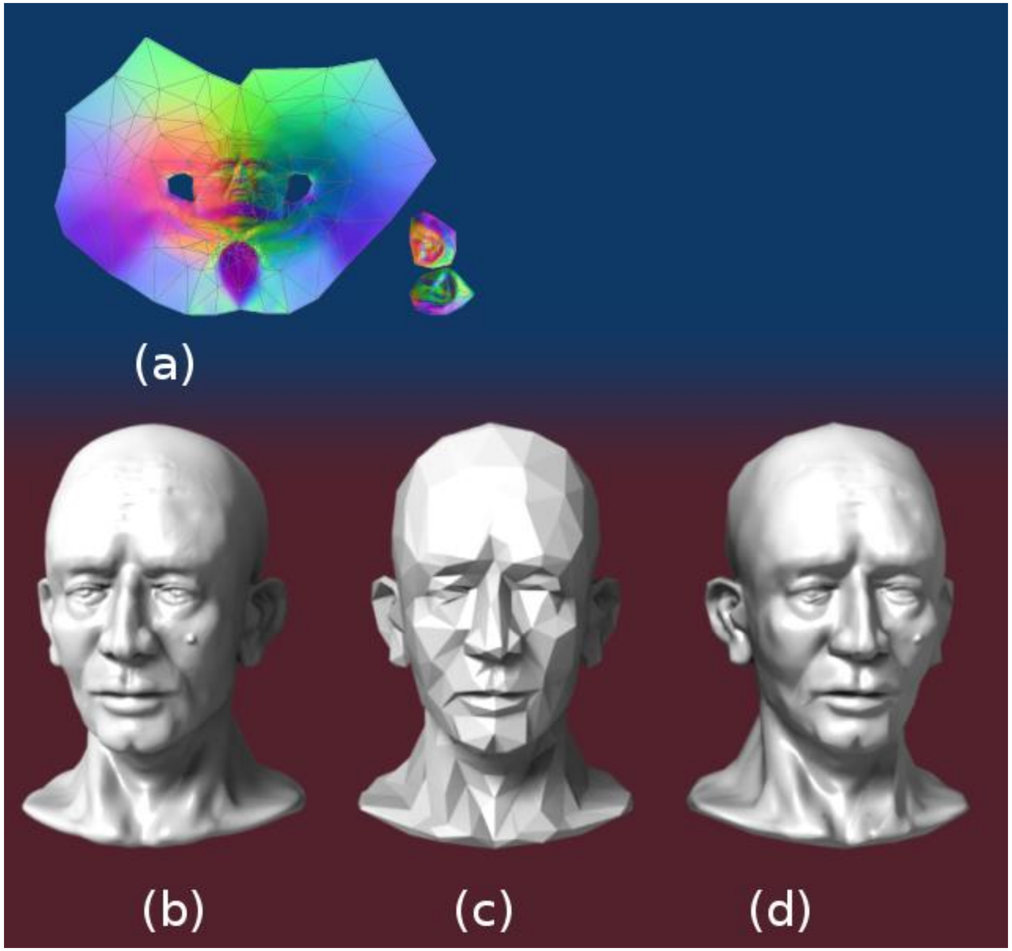
a) Normal map

(encoded in object space)

b) Original high-res model

c) Rendered low-res model

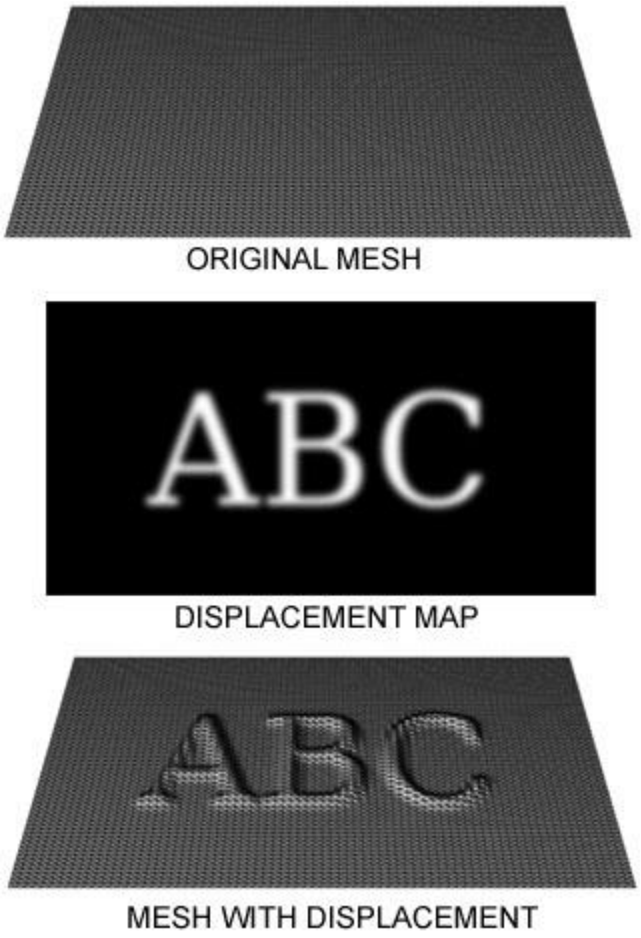
d) Applied normal map



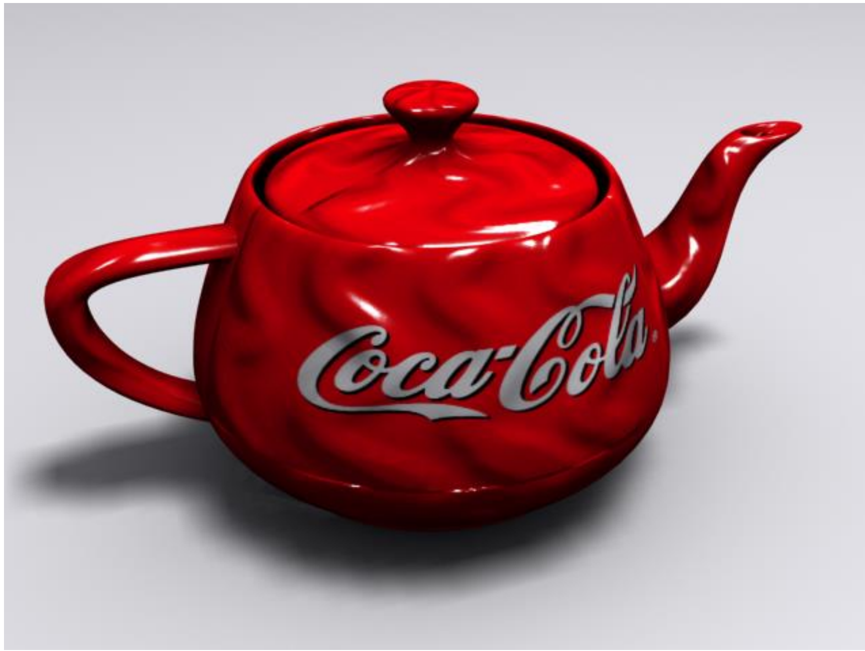
Displacement Mapping

- Move geometry as specified in texture
- Displacement in direction of surface normal
- Can add additional detail to a subdivided model
- Relies on dense geometry
- Usually used with adaptive tessellation techniques

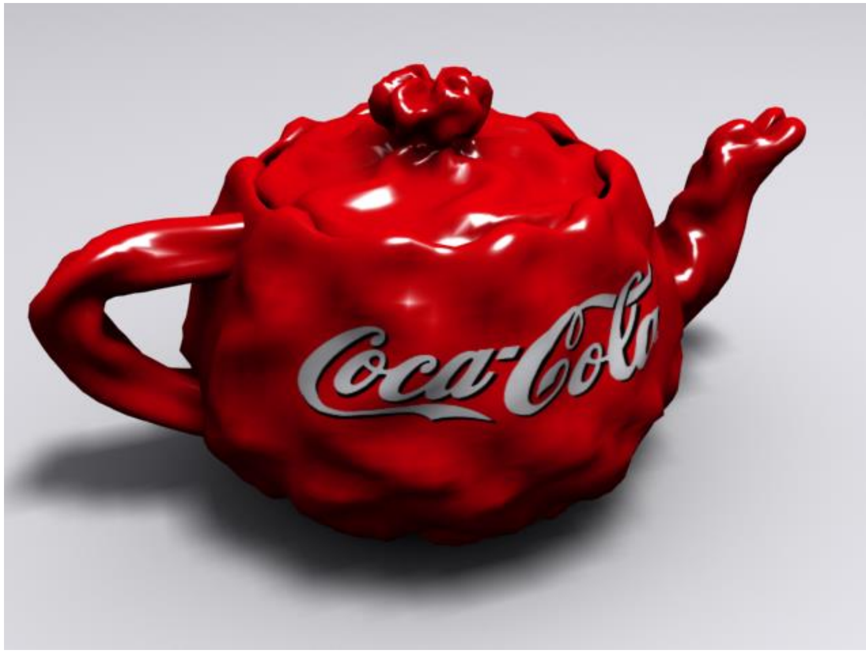
Displacement Mapping



Displacement Mapping



bump mapping

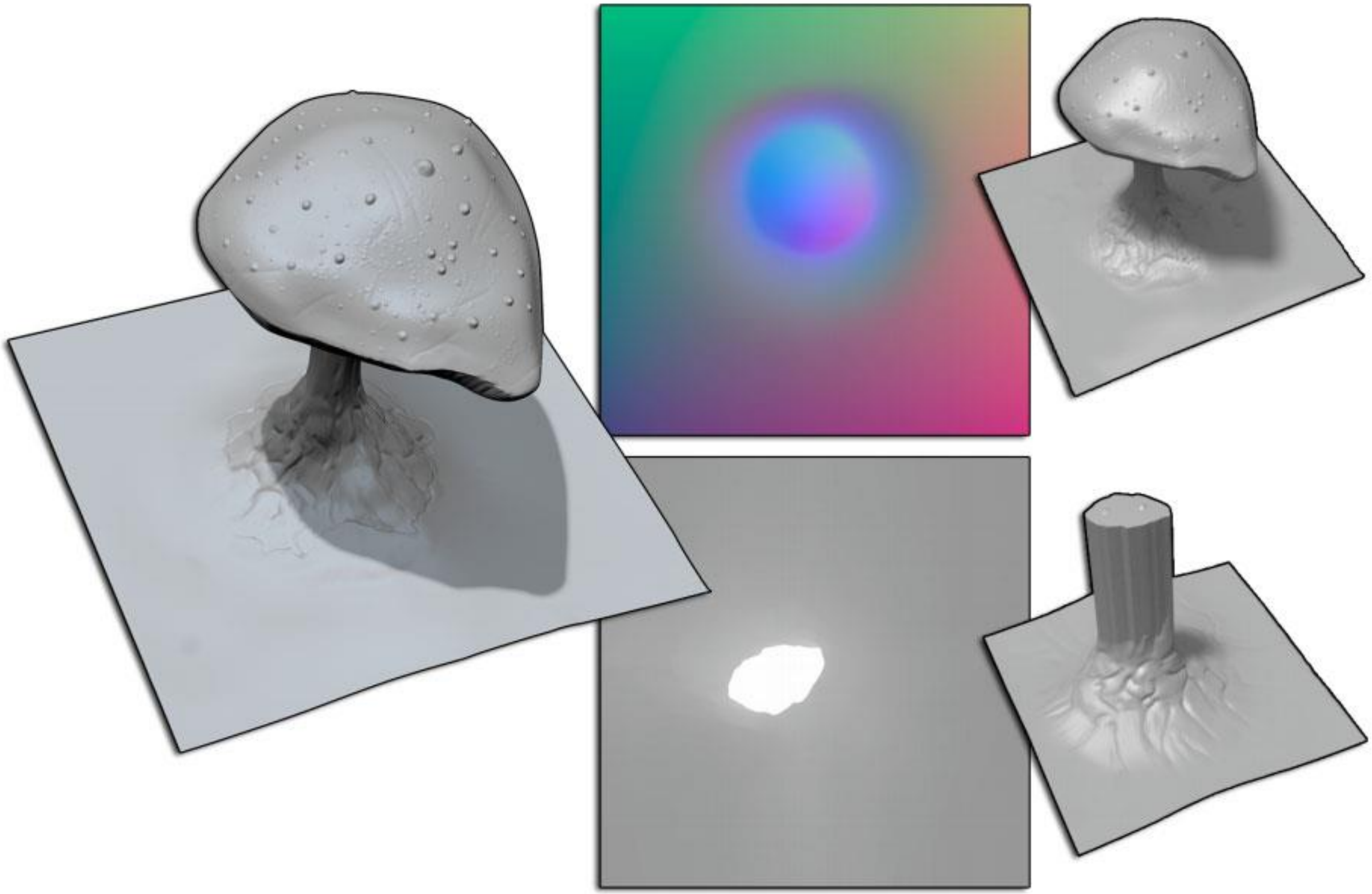


displacement mapping

Vector Displacement Mapping

- Displace geometry in any direction
- Generalization of displacement mapping
- Possible to store detailed geometry in textures
- Excellent for sculpting purposes (Z-Brush)

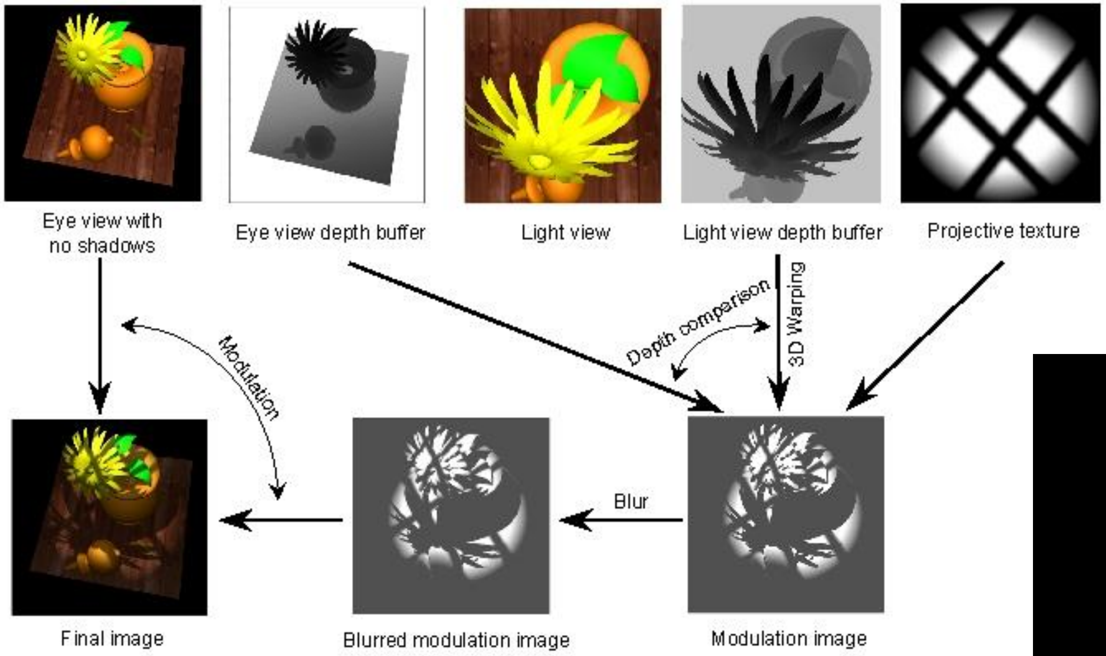
Vector Displacement Mapping



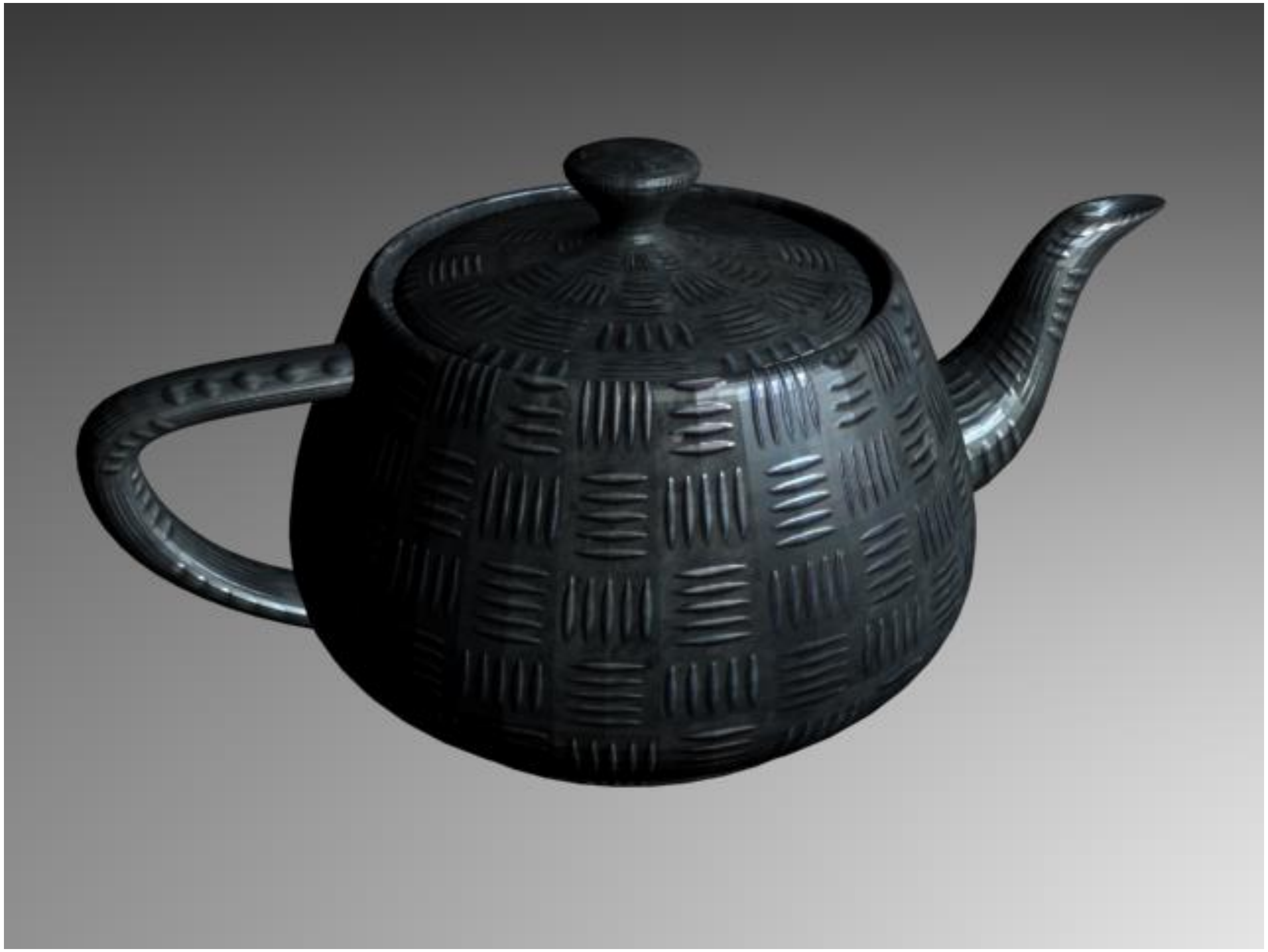
Shadow Mapping

- Two pass technique
- Obtain Light view depth buffer
- Compare each pixel rendered to with light depth
- Pixels further away are in shadow
- Needs margin of error for lit pixels
- Implementation usually has artifacts

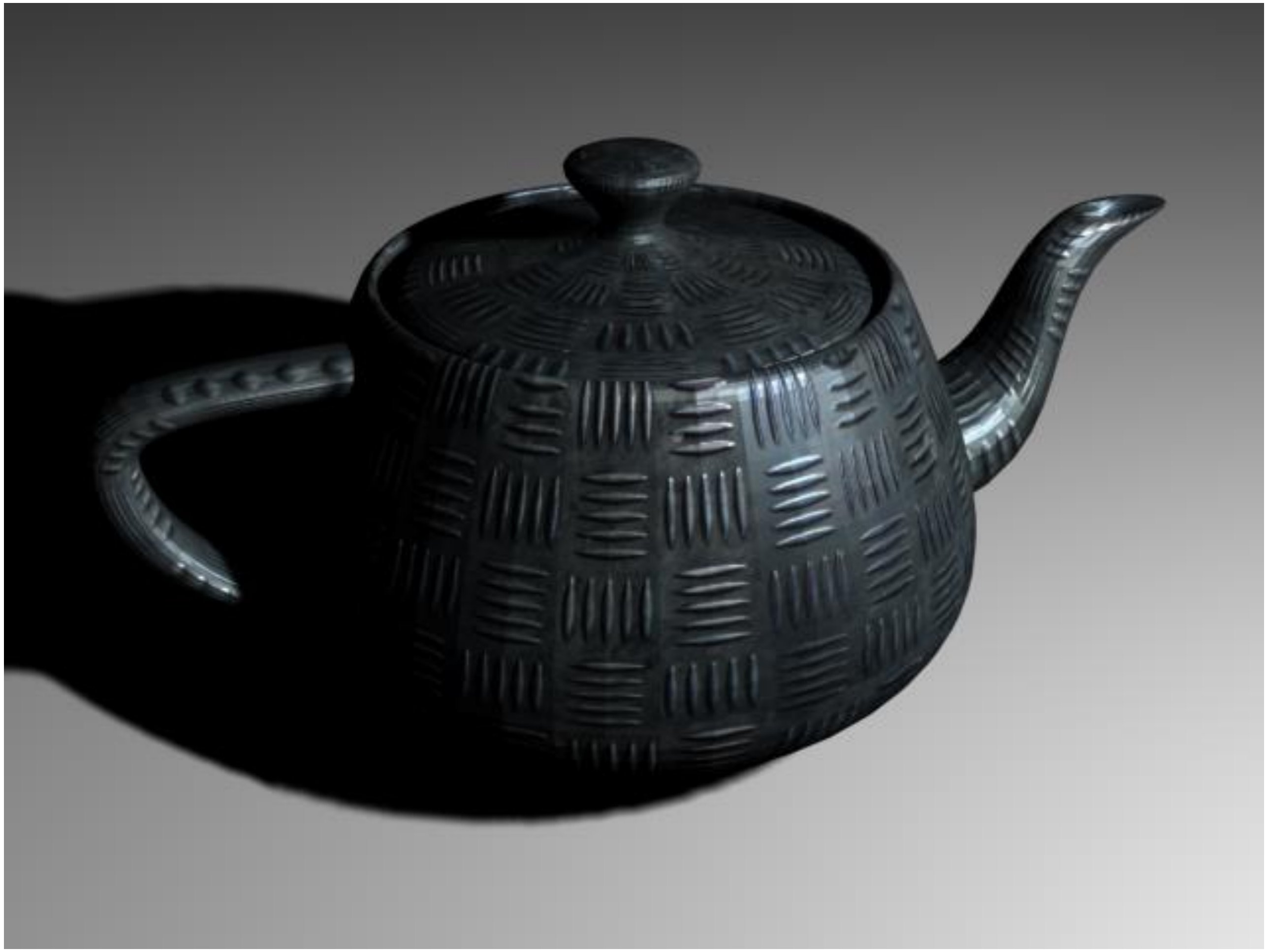
Shadow Mapping



Lights, visibility, texture…



What’s missing is shadow



Next Lecture

Shadows

Acknowledgements

- Thanks to all the people, whose work is shown here and whose slides were used as a material for creation of these slides:

Matej Novotný, GSVM lectures at FMFI UK

Peter Drahoš, PPGSO lectures at FIIT STU

Output of all the publications and great team work

Very best data from 3D cameras

Questions ?!

Skeletex
RESEARCH

www.skeletex.xyz

madaras@skeletex.xyz

martin.madaras@fmph.uniba.sk