Realtime Computer Graphics on GPUs

Textures

Jan Kolomazník

Department of Software and Computer Science Education
Faculty of Mathematics and Physics
Charles University in Prague

Computer Graphics Charles University

Introduction

TEXTURES

- Appearance enhancement:
  - color modulation with raster images (“bitmap”)
  - bump-mapping to fake geometric detail
  - possible modulation of transparency, reflectance, environment light
- Texture definition:
  - 1D or 2D data array (“bitmap texture”)
    - widely used, broad hardware support
  - 3D data array (“volume texture”)
  - procedural – callback algorithm in every fragment (programmable GPU)

Textures

- Appearance enhancement:
  - color modulation with raster images (“bitmap”)
  - bump-mapping to fake geometric detail
  - possible modulation of transparency, reflectance, environment light
- Texture definition:
  - 1D or 2D data array (“bitmap texture”)
    - widely used, broad hardware support
  - 3D data array (“volume texture”)
  - procedural – callback algorithm in every fragment (programmable GPU)

Texture Access

TEXTURING API

- Texture handle creation:

```
unsigned int texture;
glGenTextures(1, &texture);
```

- Texturing unit activation and texture binding:

```
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, texture);
```

- Data upload:

```
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE↩
    , data);
```

- Texturing parameters:

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
...
```

Texturing API

- Texture handle creation:

```
unsigned int texture;
glGenTextures(1, &texture);
```

- Texturing unit activation and texture binding:

```
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, texture);
```

- Data upload:

```
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE
    , data);
```

- Texturing parameters:

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
...
```

TEXTURING API

- Texture handle creation:

```
unsigned int texture;
glGenTextures(1, &texture);
```

- Texturing unit activation and texture binding:

```
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, texture);
```

- Data upload:

```
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE↩
      , data);
```

- Texturing parameters:

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
...
```

Texturing API

- Texture handle creation:

```
unsigned int texture;
glGenTextures(1, &texture);
```

- Texturing unit activation and texture binding:

```
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, texture);
```

- Data upload:

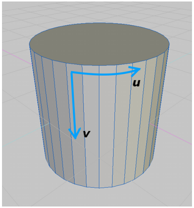
```
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE
     , data);
```

- Texturing parameters:

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
...
```

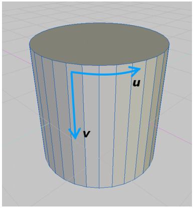
TEXTURE MAPPING

- 2D textures have to be mapped to an object surface
  - texture coordinates [u, v] ([s, t] in OpenGL) defined in every vertex
  - GPU interpolates them correctly into individ. fragments
  - bitmap data need to be interpolated (among adjacent texture pixels = ”texels”)



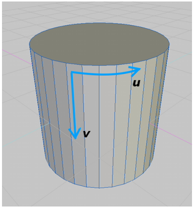
Texture Mapping

- 2D textures have to be mapped to an object surface
  - texture coordinates [u, v] ([s, t] in OpenGL) defined in every vertex
  - GPU interpolates them correctly into individ. fragments
  - bitmap data need to be interpolated (among adjacent texture pixels = ”texels”)



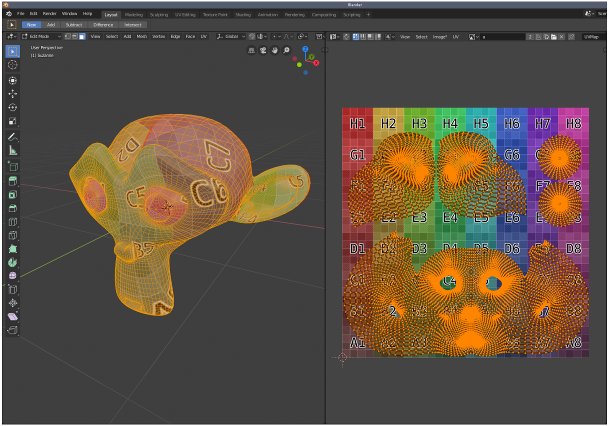
Texture Mapping

- 2D textures have to be mapped to an object surface
  - texture coordinates [u, v] ([s, t] in OpenGL) defined in every vertex
  - GPU interpolates them correctly into individ. fragments
  - bitmap data need to be interpolated (among adjacent texture pixels = ”texels”)



TEXTURE UNWRAP

- Cut along *seam edges*
- Flatten geometry to minimize distortion (prevent stretched faces)

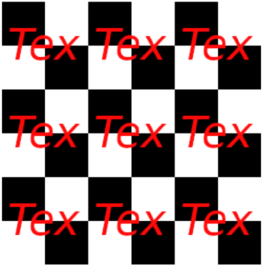


Texture Repetition

- Standard texture-coordinates domain: $[0, 1]^D$
  - Handle out-of-range values?
- Cyclic repetition (`GL_REPEAT`)
- Mirroring (`GL_MIRRORED_REPEAT`)
  - Every other tile is flipped $\to$ better continuity
- Clamping (`GL_CLAMP`, `GL_CLAMP_TO_EDGE`)
  - Optional explicit border color (`GL_CLAMP_TO_BORDER`)
  - Can be used for debugging (special color)

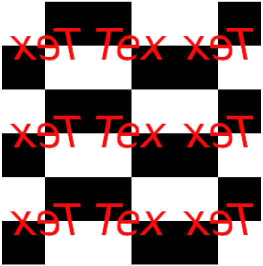
Texture Repetition

- Standard texture-coordinates domain: $[0, 1]^D$
  - Handle out-of-range values?
- Cyclic repetition (`GL_REPEAT`)
- Mirroring (`GL_MIRRORED_REPEAT`)
  - Every other tile is flipped → better continuity
- Clamping (`GL_CLAMP`, `GL_CLAMP_TO_EDGE`)
  - Optional explicit border color (`GL_CLAMP_TO_BORDER`)
  - Can be used for debugging (special color)



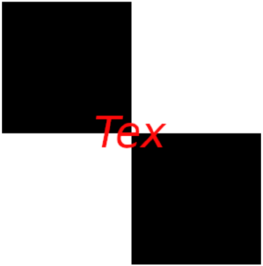
Texture Repetition

- Standard texture-coordinates domain: $[0, 1]^D$
  - Handle out-of-range values?
- Cyclic repetition (GL_REPEAT)
- Mirroring (GL_MIRRORED_REPEAT)
  - Every other tile is flipped → better continuity
- Clamping (GL_CLAMP, GL_CLAMP_TO_EDGE)
  - Optional explicit border color (GL_CLAMP_TO_BORDER)
  - Can be used for debugging (special color)



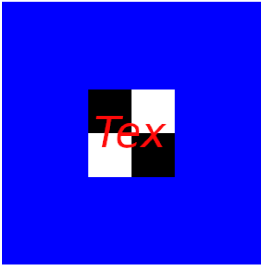
Texture Repetition

- Standard texture-coordinates domain: $[0, 1]^D$
  - Handle out-of-range values?
- Cyclic repetition (`GL_REPEAT`)
- Mirroring (`GL_MIRRORED_REPEAT`)
  - Every other tile is flipped $\rightarrow$ better continuity
- Clamping (`GL_CLAMP`, `GL_CLAMP_TO_EDGE`)
  - Optional explicit border color (`GL_CLAMP_TO_BORDER`)
  - Can be used for debugging (special color)



Texture Repetition

- Standard texture-coordinates domain: $[0, 1]^D$
  - Handle out-of-range values?
- Cyclic repetition (`GL_REPEAT`)
- Mirroring (`GL_MIRRORED_REPEAT`)
  - Every other tile is flipped $\rightarrow$ better continuity
- Clamping (`GL_CLAMP`, `GL_CLAMP_TO_EDGE`)
  - Optional explicit border color (`GL_CLAMP_TO_BORDER`)
  - Can be used for debugging (special color)



Texture Combination

- Modern GPUs (since TNT) can combine more textures in one fragment ("multitexturing")
  - global (low-frequency) basis + detail texture
  - pre-computed lighting ("light-map")
  - "environment maps" – reflection of a surround scene
- Legacy combination operators:
  - REPLACE (source is ignored)
  - MODULATE (multiplication – values are abated)
  - DECAL (semi-transparent texture on an original surface)
  - INTERPOLATE (lerp, 2 sources)
  - DOT3_RGB[A] (inner product, for 3D)
  - ADD, ADD_SIGNED, SUBTRACT, ..
- programmable GPU (in "fragment shader"): arbitrary formula

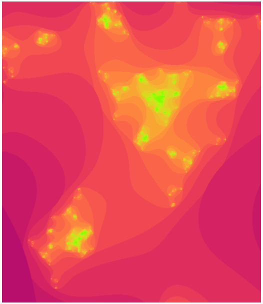
Texture Combination

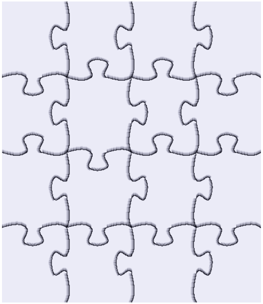
- Modern GPUs (since TNT) can combine more textures in one fragment (”multitexturing”)
  - global (low-frequency) basis + detail texture
  - pre-computed lighting (”light-map”)
  - ”environment maps” – reflection of a surround scene
- Legacy combination operators:
  - REPLACE (source is ignored)
  - MODULATE (multiplication – values are abated)
  - DECAL (semi-transparent texture on an original surface)
  - INTERPOLATE (lerp, 2 sources)
  - DOT3_RGB[A] (inner product, for 3D)
  - ADD, ADD_SIGNED, SUBTRACT, ..
- programmable GPU (in ”fragment shader”): arbitrary formula

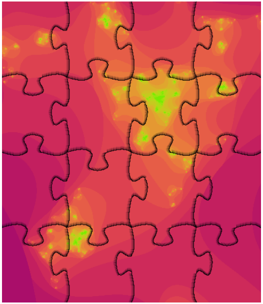
Texture Combination

- Modern GPUs (since TNT) can combine more textures in one fragment ("multitexturing")
  - global (low-frequency) basis + detail texture
  - pre-computed lighting ("light-map")
  - "environment maps" – reflection of a surround scene
- Legacy combination operators:
  - REPLACE (source is ignored)
  - MODULATE (multiplication – values are abated)
  - DECAL (semi-transparent texture on an original surface)
  - INTERPOLATE (lerp, 2 sources)
  - DOT3_RGB[A] (inner product, for 3D)
  - ADD, ADD_SIGNED, SUBTRACT, ..
- programmable GPU (in "fragment shader"): arbitrary formula

Legacy: Texture Combination II







Texture Mapping Units

- Hardware component for processing texels
- One texture mapping unit (TMU) handles one bitmap source
- Two jobs:

  Texture Addressing: texture coordinates → texels → fragments (pixels)

  Texture Filtering: interpolation, filtering
- Modern hardware – multiple texture units (one texture processed by multiple HW units)
- More TMUs → higher fill rate
- Spatial caching – neighboring fragments access texel from small neighborhood

Texture Mapping Units

- Hardware component for processing texels
- One texture mapping unit (TMU) handles one bitmap source
- Two jobs:

  Texture Addressing: texture coordinates $\rightarrow$ texels $\rightarrow$ fragments (pixels)

  Texture Filtering: interpolation, filtering
- Modern hardware – multiple texture units (one texture processed by multiple HW units)
- More TMUs $\rightarrow$ higher fill rate
- Spatial caching – neighboring fragments access texel from small neighborhood

Texture Mapping Units

- Hardware component for processing texels
- One texture mapping unit (TMU) handles one bitmap source
- Two jobs:

  Texture Addressing: texture coordinates $\rightarrow$ texels $\rightarrow$ fragments (pixels)

  Texture Filtering: interpolation, filtering
- Modern hardware – multiple texture units (one texture processed by multiple HW units)
- More TMUs $\rightarrow$ higher fill rate
- Spatial caching – neighboring fragments access texel from small neighborhood

Texture Mapping Units

- Hardware component for processing texels
- One texture mapping unit (TMU) handles one bitmap source
- Two jobs:

  Texture Addressing: texture coordinates $\rightarrow$ texels $\rightarrow$ fragments (pixels)

  Texture Filtering: interpolation, filtering
- Modern hardware – multiple texture units (one texture processed by multiple HW units)
- More TMUs $\rightarrow$ higher fill rate
- Spatial caching – neighboring fragments access texel from small neighborhood

SAMPLERS

- Sampling parameters for a texture access inside of a shader
- glBindSampler() + glBindTexture() – bind to a texture unit

```
#version 330

in vec2 v_tex;
out vec4 f_color;

uniform sampler2D u_texture;

void main() {
   f_color = texture(u_texture, v_tex);
}
```

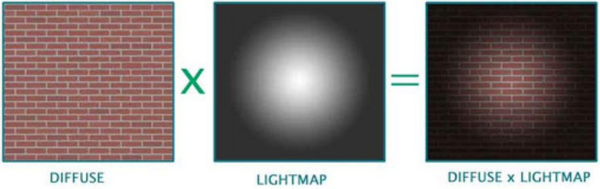
Examples

Advanced Texturing

- Most frequently used approaches:
  - gloss mapping (glossy reflection)
  - light mapping (alt: dark mapping) — lighting
  - shadow mapping — pre-computed shadow
  - ambient occlusion
  - bump mapping (normal-vector modulation)
  - parallax mapping (texture coordinates modulation)
  - environment mapping (environment reflection)

Light, Shadow Mapping, Ambient Occlusion

- Precompute lighting effects
- *Bake* into light/shadow map
- Static lighting – light source cannot be moved



Bump Mapping, Parallax Mapping

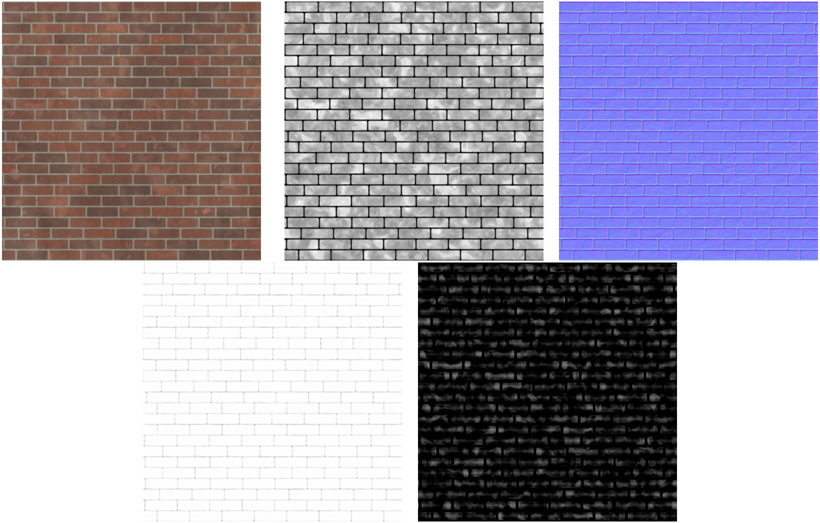
Bump Mapping:

- special texturing technique – impression of a bumpy surface
  - replaces complicated macro-geometry
  - modifies (modulates) normal vector in every pixel
  - Phong shading (normal interpolation) is recommended
  - human observer thinks that a surface is actually bumpy (much of the impression is inferred from specular reflections)

Parallax Mapping:

- simulate parallax
  - modulate texture coordinates based on displacement map
  - used together with bump mapping

TEXTURE BUNDLE

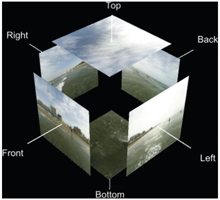


Environment Mapping

- reflection vector R converted to
  - spherical coordinates – more complicated
  - six cube faces – "cube mapping"

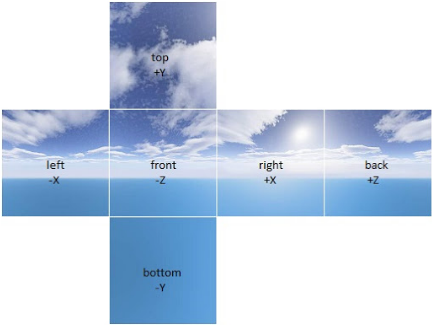
Environment Mapping

- reflection vector R converted to
  - spherical coordinates – more complicated
  - six cube faces – ”cube mapping”

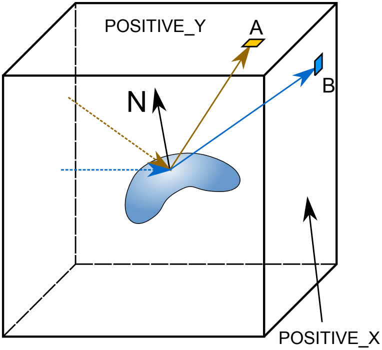


Cube Mapping

- cube-map texture consists of 6 square bitmaps
- POSITIVE_X, NEGATIVE_X, POSITIVE_Y, ...
- easy data acquisition – e.g. GPU rendering in real-time
- easy bitmal adressing, no vector normalization needed, only a division
  1. select max-value component – face
  2. compute 2D coordinates (two divisions)



Cube Mapping II



$$A : s = \left(\frac{x}{y}\right) \qquad t = \left(\frac{z}{y}\right)$$

$$B : s = \left(\frac{y}{x}\right) \qquad t = \left(\frac{z}{x}\right)$$

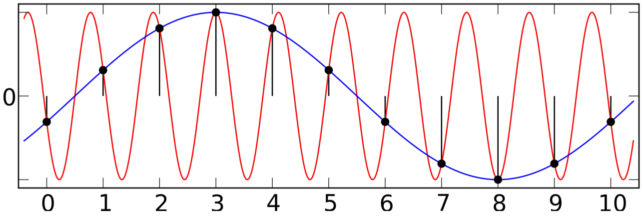
Filtering

ALIASING

- Reconstruction of original signal from discrete samples
- Problem when sampling frequency (f_{sampl}) below Nyquist limit:

$$f_{sampl} < 2f_{max}$$

- Shannon theorem
- Aliasing examples and preventions:
  - Moiré pattern (interference), rasterization
  - high speed rotation + camera, rolling shutter
  - fluorescent light + lathe
  - CD-quality audio sampling frequency

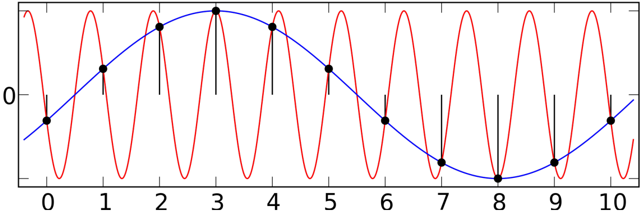


ALIASING

- Reconstruction of original signal from discrete samples
- Problem when sampling frequency (f_{sampl}) below Nyquist limit:

$$f_{sampl} < 2f_{max}$$

- Shannon theorem
- Aliasing examples and preventions:
  - Moiré pattern (interference), rasterization
  - high speed rotation + camera, rolling shutter
  - fluorescent light + lathe
  - CD-quality audio sampling frequency

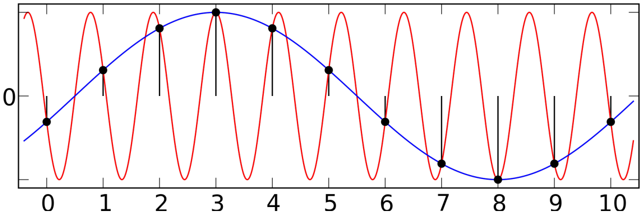


ALIASING

- Reconstruction of original signal from discrete samples
- Problem when sampling frequency (f_{sampl}) below Nyquist limit:

$$f_{sampl} < 2f_{max}$$

- Shannon theorem
- Aliasing examples and preventions:
  - Moiré pattern (interference), rasterization
  - high speed rotation + camera, rolling shutter
  - fluorescent light + lathe
  - CD-quality audio sampling frequency



Aliasing Prevention

- Higher sampling frequency
- Preprocess signal – correctly remove high frequencies (low-pass filtering)
- Hide artefacts behind another (less disturbing phenomenon) – random noise

Texture Filtering

- texture ”seen from a distance” should be filtered (raster image sub-sampling)
  - otherwise ”alias” will appear (especially disturbing in motion)
- pre-processing techniques
  - *MIP-map* (”multum in parvo”), most popular (HW)
  - *RIP-map*, anisotropic miniatures
  - anisotropic filtering – dynamic method, MIP-map + number of linear samples
  - summary tables – pre-computed upper-left rectangle sums

Texture Filtering

- texture ”seen from a distance” should be filtered (raster image sub-sampling)
  - otherwise ”alias” will appear (especially disturbing in motion)
- pre-processing techniques
  - *MIP-map* (”multum in parvo”), most popular (HW)
  - *RIP-map*, anisotropic miniatures
  - anisotropic filtering – dynamic method, MIP-map + number of linear samples
  - summary tables – pre-computed upper-left rectangle sums

Texture Filtering

- texture ”seen from a distance” should be filtered (raster image sub-sampling)
  - otherwise ”alias” will appear (especially disturbing in motion)
- pre-processing techniques
  - *MIP-map* (”multum in parvo”), most popular (HW)
  - *RIP-map*, anisotropic miniatures
  - anisotropic filtering – dynamic method, MIP-map + number of linear samples
  - summary tables – pre-computed upper-left rectangle sums

Texture Filtering

- texture ”seen from a distance” should be filtered (raster image sub-sampling)
  - otherwise ”alias” will appear (especially disturbing in motion)
- pre-processing techniques
  - *MIP-map* (”multum in parvo”), most popular (HW)
  - *RIP-map*, anisotropic miniatures
  - anisotropic filtering – dynamic method, MIP-map + number of linear samples
  - summary tables – pre-computed upper-left rectangle sums

Texture Filtering

- texture ”seen from a distance” should be filtered (raster image sub-sampling)
  - otherwise ”alias” will appear (especially disturbing in motion)
- pre-processing techniques
  - *MIP-map* (”multum in parvo”), most popular (HW)
  - *RIP-map*, anisotropic miniatures
  - anisotropic filtering – dynamic method, MIP-map + number of linear samples
  - summary tables – pre-computed upper-left rectangle sums

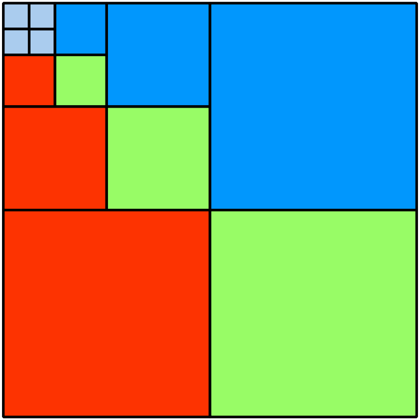
MIP-MAPPING

- texture subsampling in advance – binary fractional resolutions (1/4, 1/16, etc. – HW supported)
  - high quality sub-sampling with averaging
  - 3-component color (RGB) – convenient arrangement in memory
  - *glGenerateMipmap()*
- MIP-map utilization
  - compute level (according to required texture scaling)
  - either single texel fetch (speed)
  - or interpolation between two adjacent MIP-map levels or even bi-linear interpolation in the levels (at most 8 fetches = quality)

MIP-MAPPING

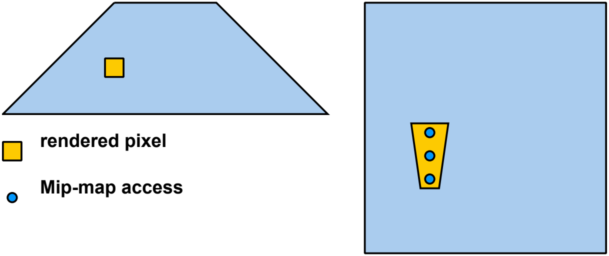
- texture subsampling in advance – binary fractional resolutions (1/4, 1/16, etc. – HW supported)
  - high quality sub-sampling with averaging
  - 3-component color (RGB) – convenient arrangement in memory
  - *glGenerateMipmap()*
- MIP-map utilization
  - compute level (according to required texture scaling)
  - either single texel fetch (speed)
  - or interpolation between two adjacent MIP-map levels or even bi-linear interpolation in the levels (at most 8 fetches = quality)

Mip-mapping II



Anisotropic Filtering

- back-projected screen pixel = deformed quadrangle
- MIP-map level according the higher sub-sampling (shorter size)
- multi-sampling (averaging) along the longer side



Custom Filtering

- Arbitrary filtering implemented in shader
- Integral images (summary tables)
- Multiple texture accesses
  - Incorporate perspective (anisotropy):
  - Derivatives between fragments: dFdx(), dFdy():

Example: flat normal

```
normalize( cross(dFdx(pos), dFdy(pos)) );
```

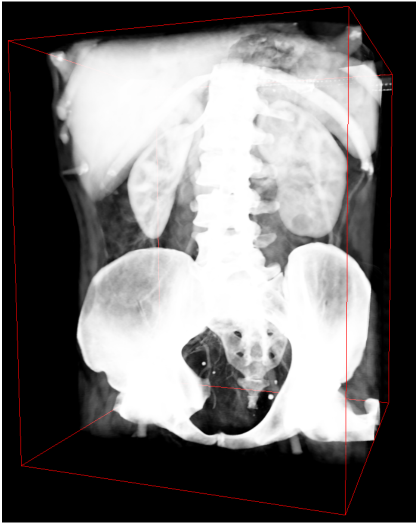
3D Textures

3D TEXTURE

- Trilinear interpolation
- Modeling material properties (marble, wood, clouds)
- Z-direction interpreted as time – animation
- Precomputed lighting effects: normal $\rightarrow$ texture coordinates
- Scientific applications
  - Tomography
  - Vector fields – fluid simulations, ...

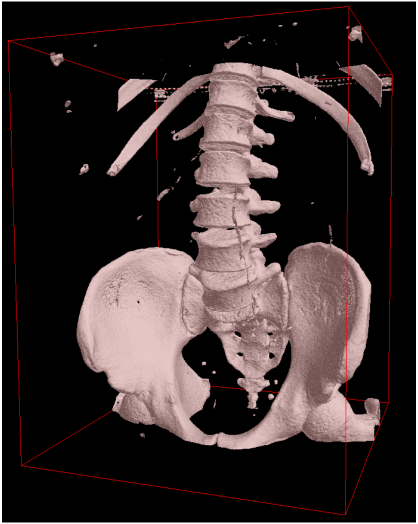
Application: Medical Data Visualization

(a) Maximum intensity projection

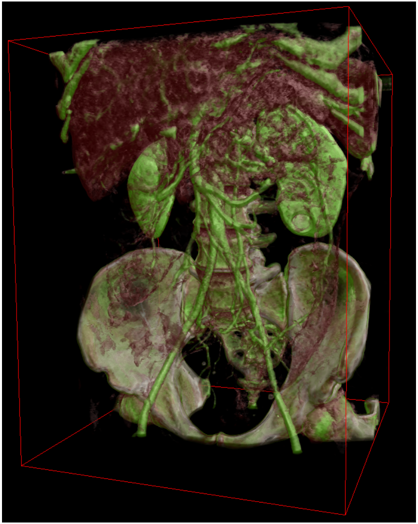


(b) Density integration

Application: Medical Data Visualization II



(a) Isosurfaces



(b) 1D transfer function