Real-Time Volume Graphics

[09] Improving Quality
This is what we want ...
But sometimes this is what we get ...
Volume Rendering Artifacts

- Volume Rendering Pipeline
  - Where are errors introduced?

Sampling
Volume Rendering Artifacts

- Volume Rendering Pipeline
  - Where are errors introduced?

Sampling → Filtering
Volume Rendering Artifacts

Volume Rendering Pipeline

Where are errors introduced?
Volume Rendering Artifacts

Volume Rendering Pipeline

- Where are errors introduced?
Volume Rendering Artifacts

Volume Rendering Pipeline

- Where are errors introduced?

1. Sampling
2. Filtering
3. Classification
4. Shading
5. Integration

[Visual diagrams and images related to volume rendering pipeline and artifacts.]
Volume Rendering Artifacts

Volume Rendering Pipeline

- Where are errors introduced?

1. Sampling
   - Low sampling rate

2. Filtering
   - Linear filtering

3. Classification
   - High frequencies

4. Shading
   - Quantized Gradients

5. Integration
   - 8 bit blending

Framebuffer
Sampling Artifacts
Sampling Artifacts
Sampling Artifacts

- **Reason:**
  - Low sampling rate

- **Solutions:**
  - Increase sampling rate to Nyquist frequency
    => at least 2 samples per voxel
  - Adaptive Sampling
    - Higher sampling rate where data contains high frequencies
Sampling Artifacts

Reason:
- Low sampling rate

Solutions:
- Increase sampling rate to Nyquist frequency
  \[ \Rightarrow \text{at least 2 samples per voxel} \]
- Adaptive Sampling
  - Higher sampling rate where data contains high frequencies
Sampling Artifacts

Reason:
- Low sampling rate

Solutions:
- Increase sampling rate to Nyquist frequency
  $\Rightarrow$ at least 2 samples per voxel
- Adaptive Sampling
  - Higher sampling rate where data contains high frequencies
Sampling Artifacts

• Remove woodgrain artifact by stochastic jittering of ray-start position
  • Look-up noise texture to offset ray-position in view direction
Filtering Artifacts

- $64^3$ volume, object-aligned slices

- bi-linear filter
- bi-cubic filter
Filtering Artifacts

Reason:
- Internal filtering precision dependent on input texture format
- Linear filters do not approximate the ideal reconstruction filter (sinc-Filter) very well

Solutions:
- Use internal format with higher precision
- Implement a HQ filter in a fragment program
Filtering Artifacts / HQ-Filters

- Discretized version of convolution integral

Discrete signal $f[x]$ is filtered by $h(x)$ to produce $g(x)$ via convolution:

$$g(x) = f[x] * h(x) = \sum_{i=[x]-m+1}^{[x]+m} f[i]h(x - i)$$

Filter width: $2m$
Filtering Artifacts / HQ-Filters

- Procedural evaluation of kernel and convolution
- Texture-based kernel and convolution

```
FetchInputSamples();
ComputeWeights_X();
for (i=0; i<3; i++)
    Convolution_X();
ComputeWeights_Y();
Convolution_Y();
```

see for example NVIDIA’s BicubicTexMagnification demo

pre-sampled B-spline kernel from [Hadwiger et al. 2001]
Filtering Artifacts / HQ-Filters

- Procedural evaluation of kernel and convolution
- Texture-based kernel and convolution

FetchInputSamples();
ComputeWeights_\textbf{X}();
for (i=0; i<3; i++)
  Convolution_\textbf{X}();
ComputeWeights_\textbf{Y}();
Convolution_\textbf{Y}();

see for example NVIDIA's BicubicTexMagnification demo
pre-sampled B-spline kernel from [Hadwiger et al. 2001]
Filtering Artifacts / HQ-Filters

- Transform original kernel into look-up texture

- Replicate kernel tile over output grid
Filtering Artifacts / HQ-Filters

- Easy when kernel separable
- Use same 1D kernel tile for all axes
- Sample two (or three) times and multiply weights
- Separable bi-cubic and tri-cubic filters need only one 1D RGBA kernel tile
Filtering Artifacts / HQ-Filters

- Distribution instead of gathering
- Works for all filter shapes and sizes
- Works for separable and non-separable kernels
- Multi-pass evaluation of filter convolution sum
Filtering Artifacts / HQ-Filters

- Tri-cubic gradients and second derivatives for isosurfaces possible in real-time

\[
\frac{\partial G(x_1, x_2)}{\partial x_1} \quad \text{and} \quad \frac{\partial G(x_1, x_2)}{\partial x_2}
\]
Filtering Artifacts / HQ-Filters

linear

cubic
Classification Artifacts

Pre-Classification

... → Classification → Filtering → ...

Post-Classification

... → Filtering → Classification → ...

Pre-Integrated-Classification

... → Filtering → Integral Lookup → ...

REAL-TIME VOLUME GRAPHICS
Klaus Engel
Siemens AG, Erlangen, Germany

Eurographics 2006
Classification Artifacts

- High frequencies in the transfer function $T$ increase required sampling rate
Classification Artifacts

multiple peaks
Classification Artifacts

multiple peaks
Classification Artifacts

multiple peaks

\[ T(s) \]
Classification Artifacts

multiple peaks

$T(s)$

$s$

50 x more slices
Classification Artifacts / Pre-integration

image-order

slice-by-slice

sample-by-sample

object-order

slab-by-slab

raySegment-by-raySegment
Classification Artifacts / Pre-integration

- Pre-integration table to large
  - 12bit x 12bit x RGBA8 = 4096x4096x32bit = 64MB

- Pre-integration table to slow to compute
  - even with integral functions

Solution:
- use Integral functions
- Store integral functions in 1D floating point texture
- two 1D table lookups
Classification Artifacts / Pre-integration

- Almost no performance penalty in sw raycasting
  - Cache last scalar value

- Currently slower on graphics hardware
  - Temporary fragment registers are zeroed
  - Two lookups into the volume per fragment
  - => multiple integration steps @ once (raycasting)

- Pre-integrated volume rendering still not “correct”
  - Assumes linear progression of scalar value along ray
  - => Gaussian Transfer Functions (Kniss Vis’03)
Classification Artifacts / Pre-integration

128 slices pre-classification

284 slices post-classification

128 slices post-classification

128 slices pre-integrated
Classification Artifacts / Pre-integration

single step
Classification Artifacts / Pre-integration

multiple peaks
Classification Artifacts / Pre-integration

many peaks
Classification Artifacts / Pre-integration

acoustic volume inside car cabine

REAL-TIME VOLUME GRAPHICS
Klaus Engel
Siemens AG, Erlangen, Germany

Eurographics 2006
Shading Artifacts
Shading Artifacts
Shading Artifacts

Reasons:
- Gradients are precomputed and quantized
- Interpolation of normals causes unnormalized normals

Solutions:
- Store gradients in high-precision texture, re-normalize in fragment program
  - 😞 too much memory
- Compute gradients on-the-fly
  - 😊 high-quality gradient
  - 😞 many texture fetches
Shading Artifacts

**Reasons:**
- Gradients are pre-computed and quantized
- Interpolation of normals causes unnormalized normals

**Solutions:**
- Store gradients in high-precision texture, re-normalize in fragment program
  - 😞 too much memory
- Compute gradients on-the-fly
  - 😊 high-quality gradient
  - 😕 many texture fetches
Shading Artifacts

Pass in shifted texture coordinates

```cg
// samples for forward differences
half3 normal;
half3 sample1;
sample1.x = (half)tex3D(Volume, IN.TexCoord2).x;
sample1.y = (half)tex3D(Volume, IN.TexCoord4).x;
sample1.z = (half)tex3D(Volume, IN.TexCoord6).x;

// additional samples for central differences
half3 sample2;
sample2.x = (half)tex3D(Volume, IN.TexCoord3).x;
sample2.y = (half)tex3D(Volume, IN.TexCoord5).x;
sample2.z = (half)tex3D(Volume, IN.TexCoord7).x;

// compute central differences gradient
normal = normalize(sample2.xyz - sample1.xyz);
```
Shading Artifacts

- Neighbor lookup for anisotropic voxel data
  - Constant offsets

- Neighboring voxels + gradient correction
Shading Artifacts

Constant offsets

Neighboring voxels + gradient correction
Shading Artifacts

Constant offsets

Neighboring voxels + gradient correction
Blending Artifacts

8 bit
fixed point
blending

16 bit
floating point
blending

32 bit
floating point
blending
Blending Artifacts

Reason:
- 8 bit blending accumulates error in the framebuffer
- Lower alpha values have higher error

Solutions:
- Blending into a floating point FBO
- Nv4x support 16 bit floating point blending
- Nv3x, R3xx and R4xx do not support floating point blending
  => implement blending in a fragment program
- Ping-pong blending to prevent read/write race conditions
Blending Artifacts

- The graphics hardware does not support floating point blending (NV3x, R3xx, R4xx) => implement blending yourself
- Clean: ping-pong
Blending Artifacts

- The graphics hardware does not support floating point blending (NV3x, R3xx, R4xx)
  => implement blending yourself
- Clean: ping-pong
Blending Artifacts

- The graphics hardware does not support floating point blending (NV3x, R3xx, R4xx) => implement blending yourself

- Clean: ping-pong
Blending Artifacts

- The graphics hardware does not support floating point blending (NV3x, R3xx, R4xx) => implement blending yourself
- Dirty: ping without the pong
Blending Artifacts

- The graphics hardware does not support floating point blending (NV3x, R3xx, R4xx) => implement blending yourself
- Dirty: ping without the pong
Blending Artifacts

- The graphics hardware does not support floating point blending (NV3x, R3xx, R4xx) => implement blending yourself
- Dirty: ping without the pong
Conclusions

- Artifacts are introduced in various stages of the volume rendering process.

- Current graphics hardware is flexible and precise enough to remove or suppress artifacts.

- Real-Time performance can still be achieved in most cases.