



GLSL II

Oliver Kutter

19 June 2007

chair for computer aided medical procedures
department of computer science | technische universität münchen

Offscreen Rendering

OpenGL Framebuffer Objects

- Motivation: Offscreen Rendering für Shading, Effekte, GPGPU mit wählbarer Genauigkeit (Standard FB nur 8bit)
- Lightweight Solution für Offscreen Render To Texture (replaces OpenGL PBuffers)
- Attachment Points für
 - Color Render Targets (je nach GPU, SM3.0 \leq 4, SM4.0 \leq 8)
 - Depth Render Target (max. 1)
 - Stencil Render Target (max. 1)

Create FBO and bind color target

```
GLuint fbo, color;
// Create an FBO
glGenFramebuffersEXT(1, &fbo);
// Create color texture
glGenTextures(1, &color);
glBindTexture(GL_TEXTURE_2D, color);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA8, width, height, 0,
GL_RGBA, GL_UNSIGNED_BYTE, NULL);
// Bind the FBO and attach color texture to it
glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, fbo);
glFramebufferTexture2DEXT(GL_FRAMEBUFFER_EXT,
GL_COLOR_ATTACHMENT0_EXT, GL_TEXTURE_2D, color, 0);
```

Aktivieren/Deaktivieren

```
glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, fbo);
```

```
glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, 0);
```

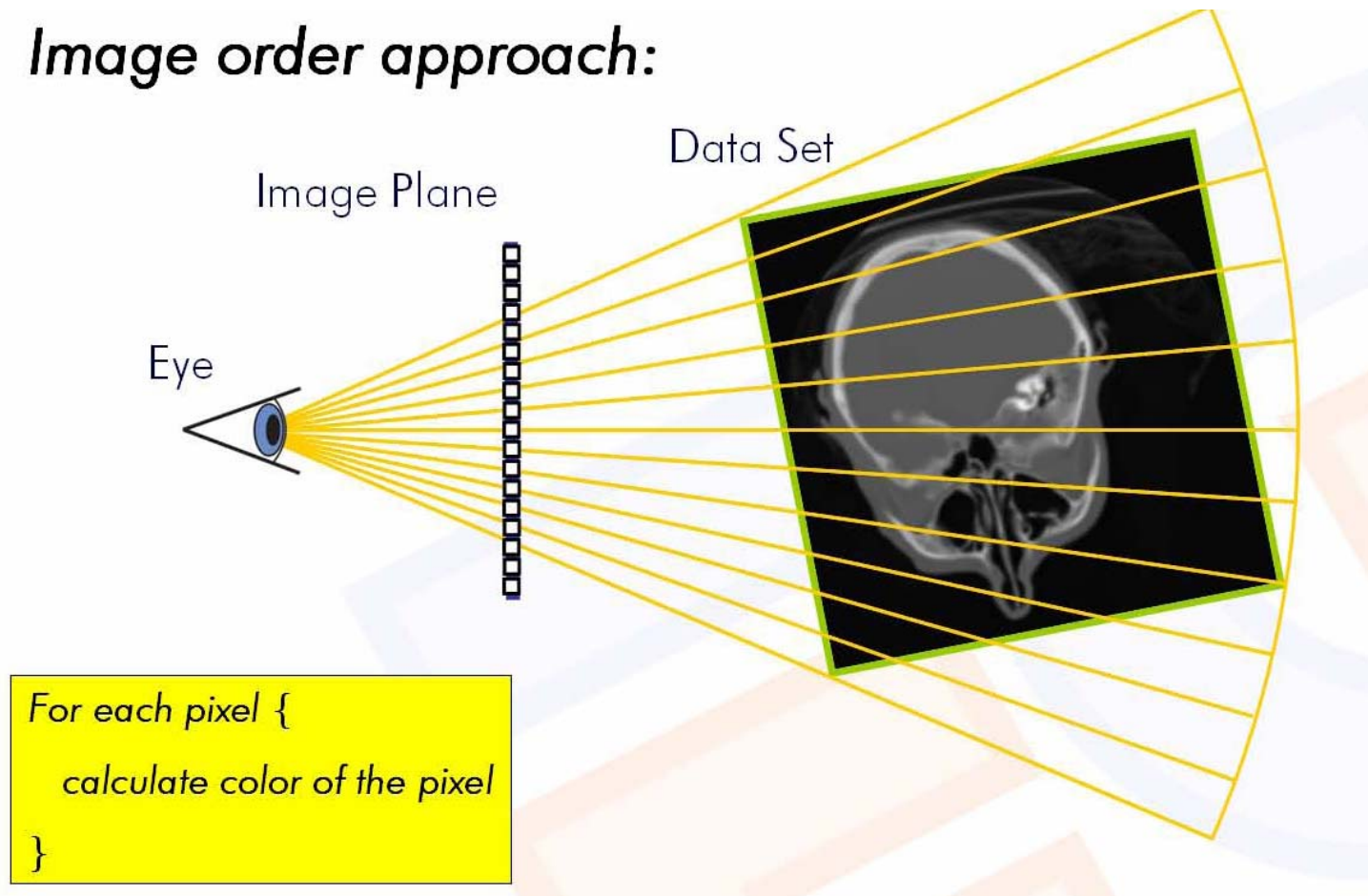
Image Post Processing

- Rendere im ersten Pass in Offscreen Render Target mittels FBO
- Für Post Pass setze Viewport Höhe und Breite auf selbe Höhe und Breite wie Texture
- Rendere ein FullScreen Quad texturiert mit dem Ergebniss aus 1. offscreen Pass. (orthographische Projektion!!)
- Mittels Shader lies Texels aus und wende Post Processing Filter an
- Rendere Ergebniss entweder in Framebuffer oder ein anderes Offscreen Render Target

Volume Rendering

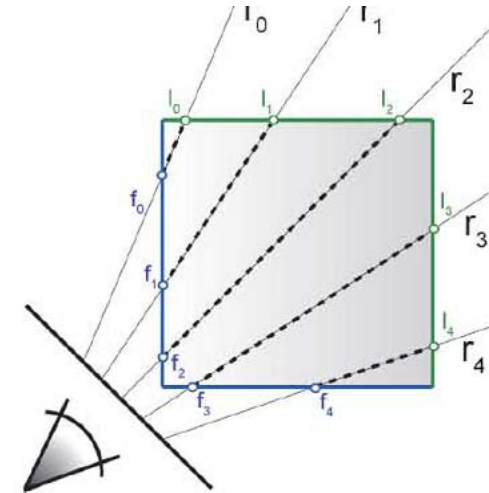
Volume Rendering by Raycasting

Image order approach:



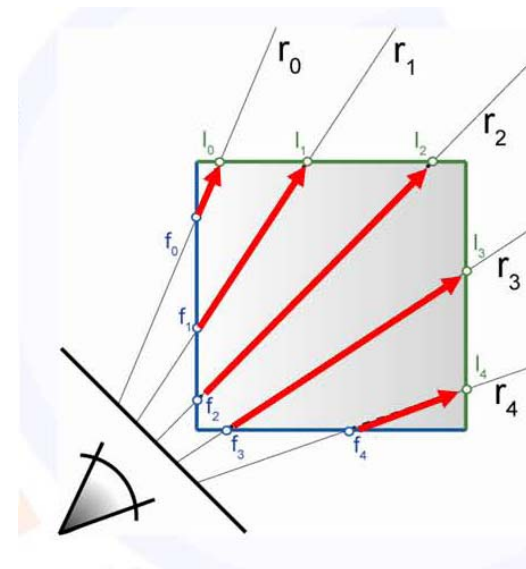
GPU Raycasting - Motivation

- Raycasting yields superior images over Proxy Geometry based DVR approaches
- Works correct for any perspective projection
- 32bit float precision rendering possible, when implemented in single loop in fragment shader
- Easy to implement, highly optimizable and adaptable to specific problems. Basically write shader for one ray and let GPU care for parallelization
- Real time capable (1024x1024 vp, 32bit fp, 256x256x512 volume @ 50 – 100Hz on Geforce 8800 GTX unoptimized!)

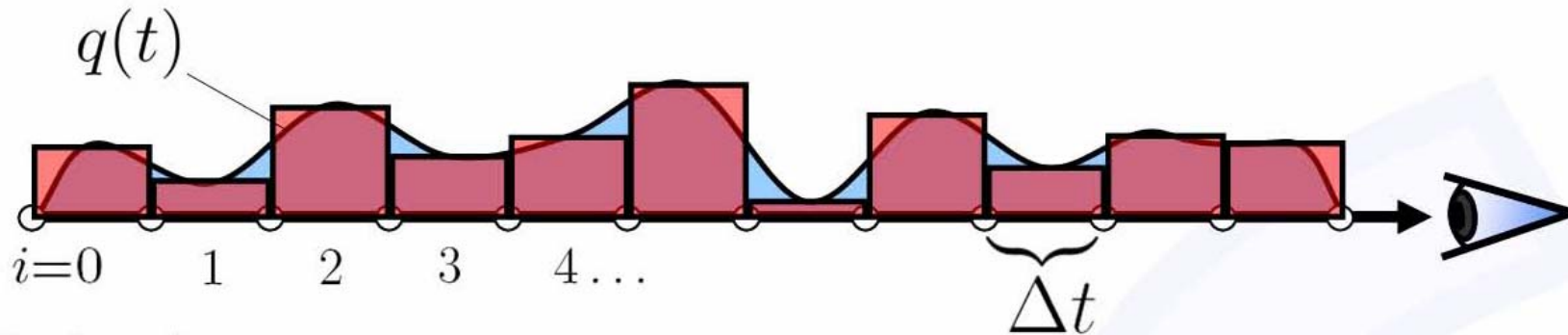


Raycasting Howto

- Render backfaces of color coded bounding geometry → ray exit positions
- Render frontfaces → ray entry positions
- Exit – Entry → Ray direction and length
- In Final Pass:
 Compute Volume
 Rendering integral
 for each ray in loop in
 shader on GPU



Volume Rendering Integral



Back-to-front compositing

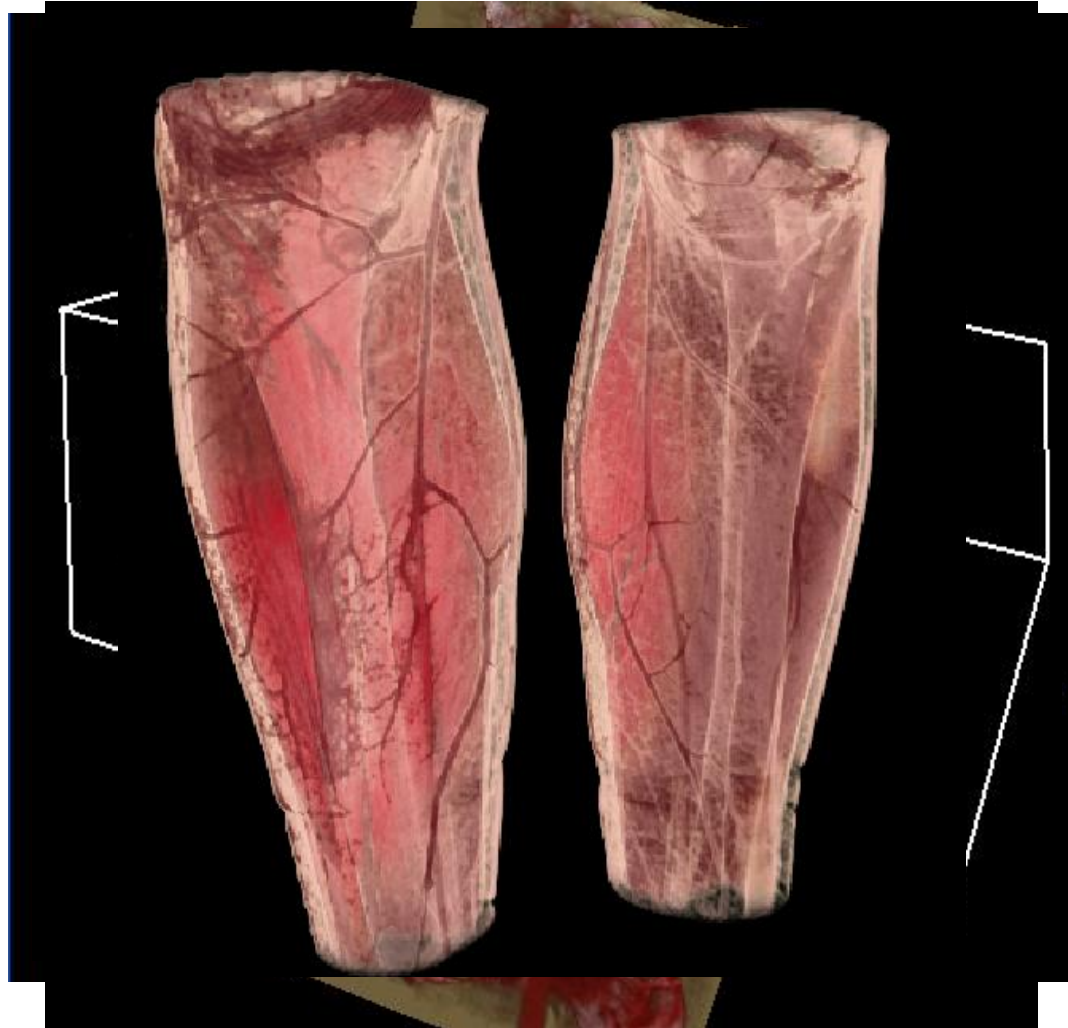
$$C'_i = C_i + (1 - A_i)C'_{i-1}$$

Front-to-back compositing

$$C'_i = C'_{i+1} + (1 - A'_{i+1})C_i$$

$$A'_i = A'_{i+1} + (1 - A'_{i+1})A_i$$

Assignment



Links, Tutorials and References

- GPGPU General Purpose GPU <http://gpgpu.org>
- Real-Time Volume Graphics (Siggraph 2004) Course Notes <http://www.vrvis.at/via/resources/course-volgraphics-2004/>
- Book Real-Time Volume Graphics <http://www.real-time-volume-graphics.org/>
- Vendor SDKs (ATI, NVIDIA) . Beispiele + Doku zu FBO und Single Pass Raycasting (nur NVIDIA SDK)

Fragen ?

Hiwi Wanted!

BA/MA/DA/SEP/IDP available!

Contact:

kutter@in.tum.de

kettner@in.tum.de

georgel@in.tum.de