INTERAKTIVES RAY-CASTING VON VOLUMENDATEN

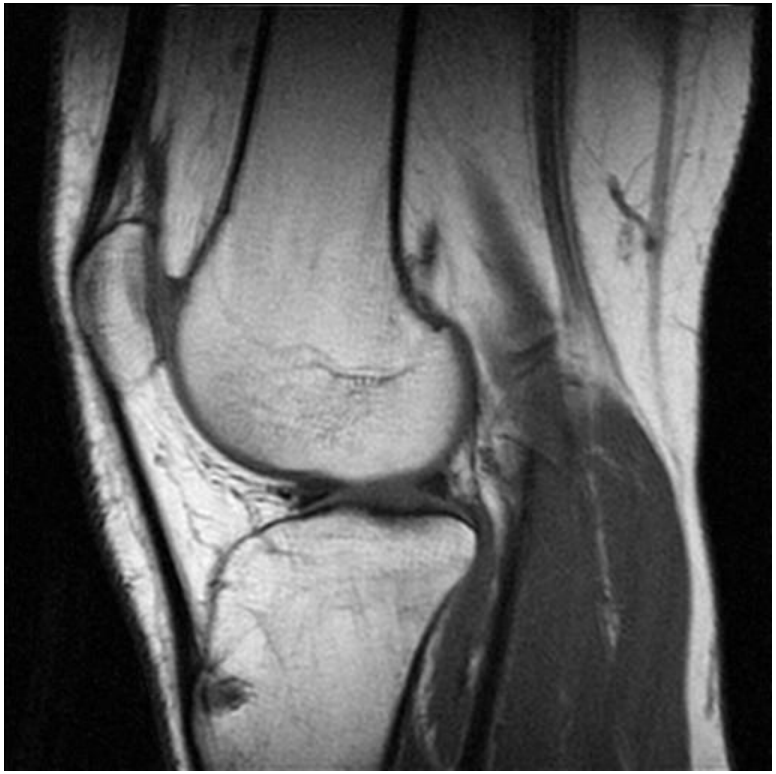
Kolloquium von Raphael Menges

Inhalt

- Problemstellung
- Komposition
- Ray-Casting in OpenGL / GLSL
- Transferfunktion
- Early Ray Termination
- Empty Space Skipping
- Stochastic Jittering
- Weiterführende Techniken

Problemstellung

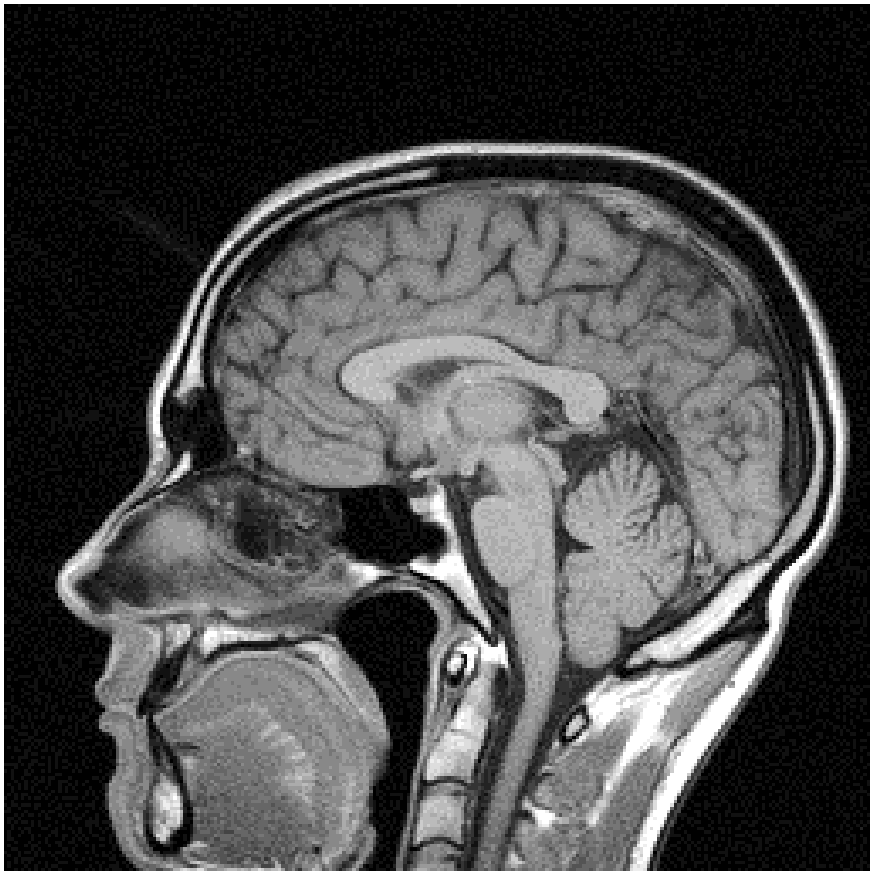
https://commons.wikimedia.org/wiki/File:Modern_3T_MRI.JPG



https://commons.wikimedia.org/wiki/File:Knie_mr.jpg

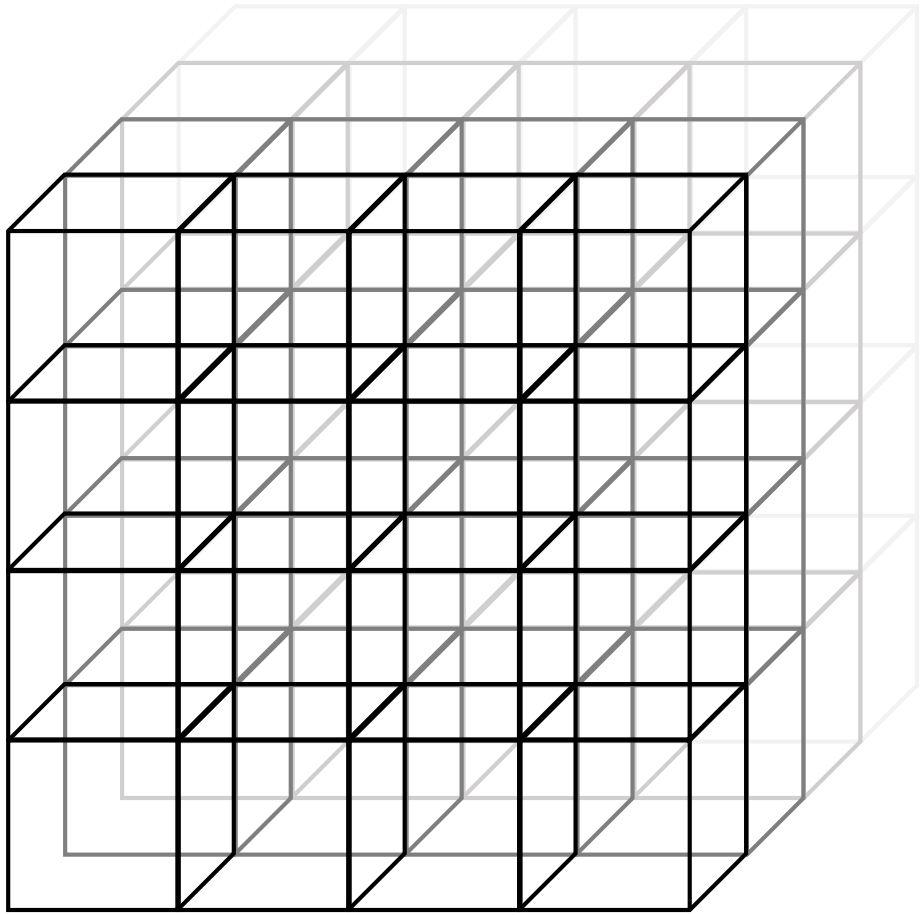
CT versus MRT

http://upload.wikimedia.org/wikipedia/en/1/16/MRI_of_human_head_%28sagittal_view%29.jpg

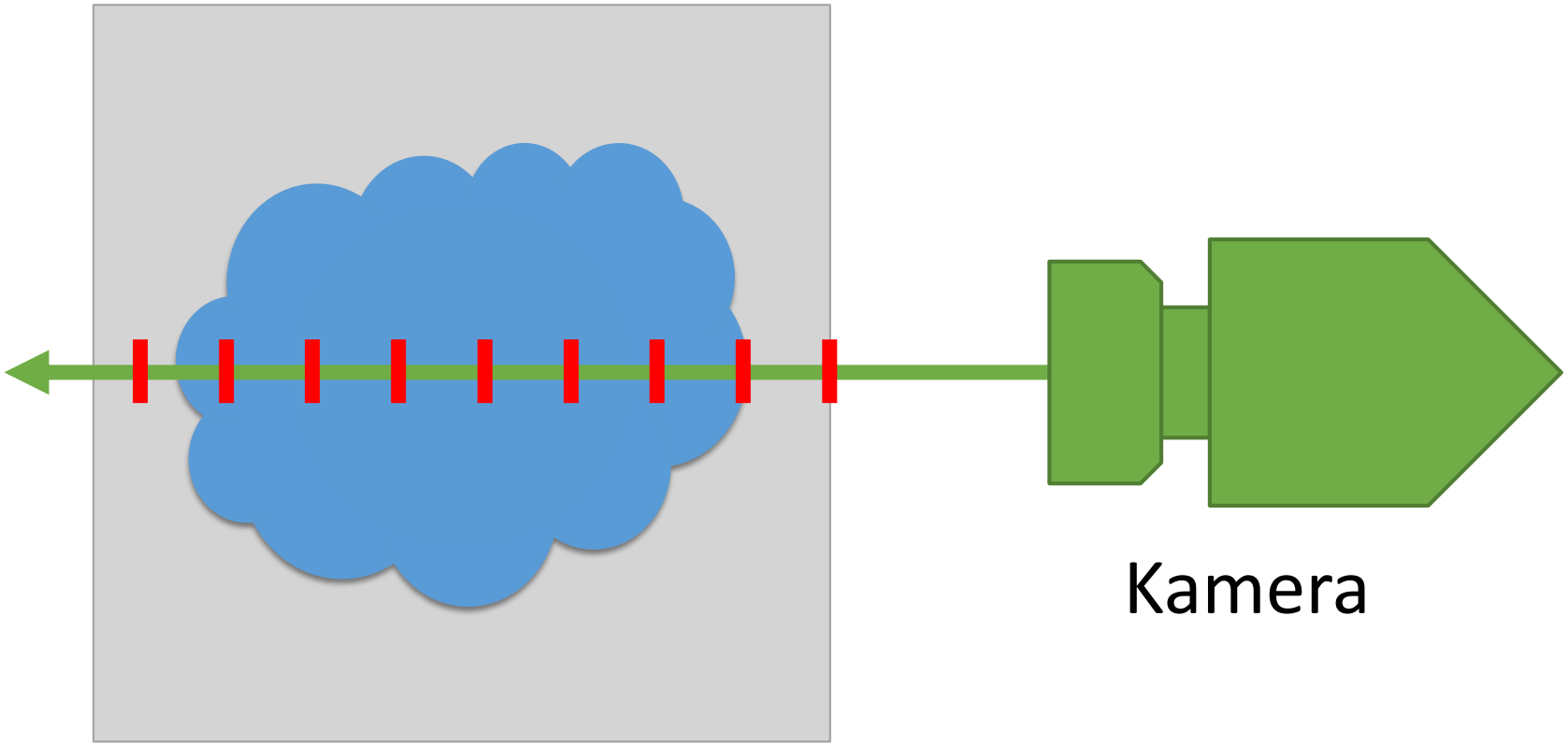


http://www.bostonaccidentinjurylawyer.com/CT_head_slice.jpg

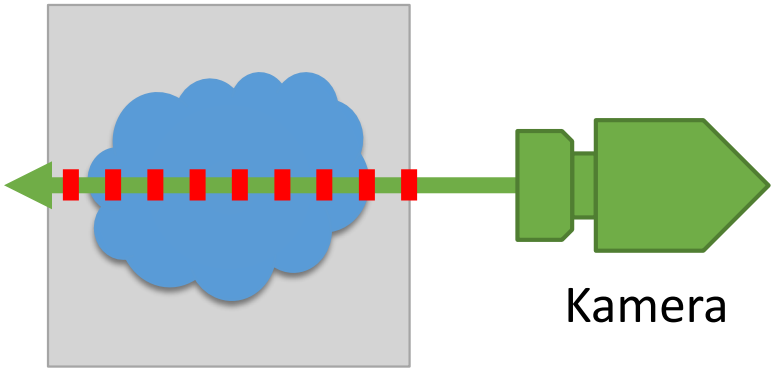
Uniformes Netz



Problemstellung



Komposition



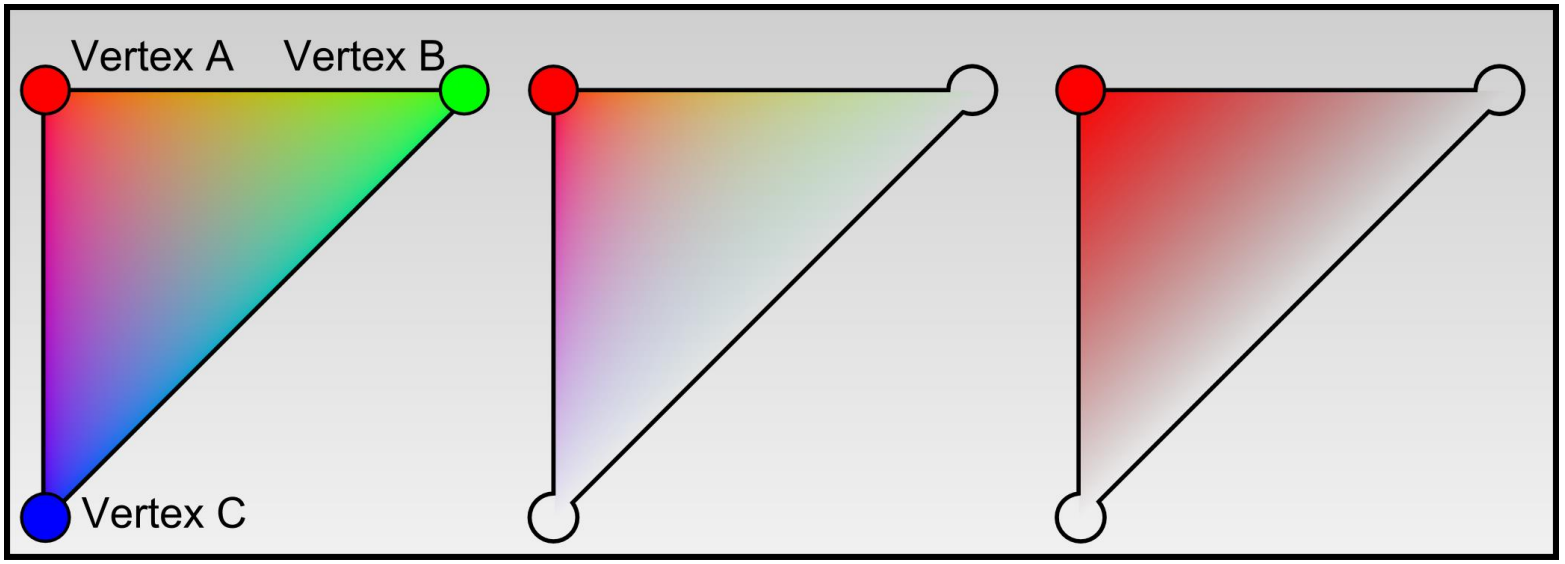
~~$C_{dst} \leftarrow C_{dst} + (1 - \alpha_{dst}) * C_{src}$~~

$$C_{dst} \leftarrow C_{dst} + (1 - \alpha_{dst}) * C_{src} * \alpha_{src}$$

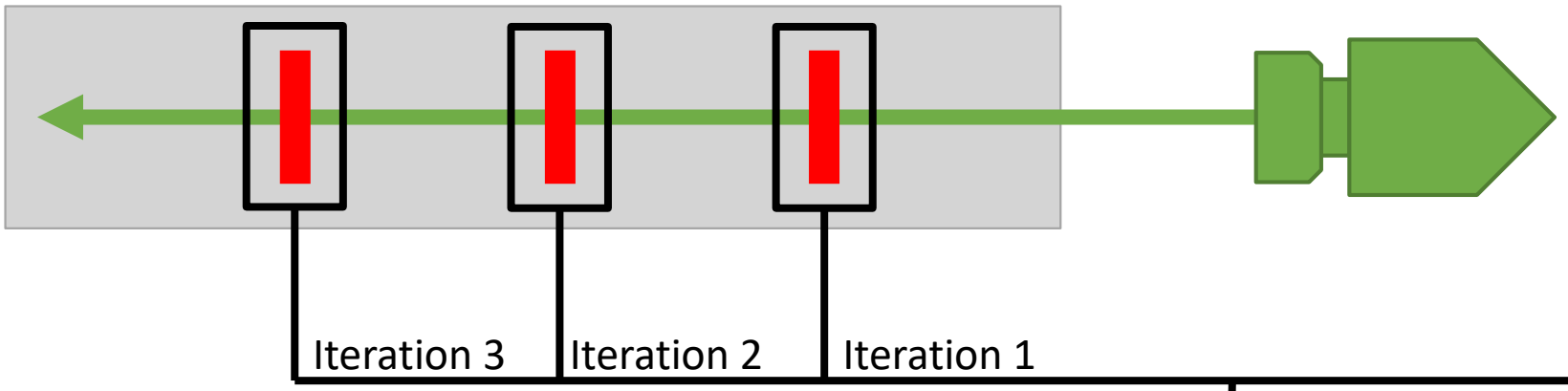
$$\alpha_{dst} \leftarrow \alpha_{dst} + (1 - \alpha_{dst}) * \alpha_{src}$$

Herleitung zum Beispiel in „Real-Time Volume Graphics“ zu finden (siehe Ausarbeitung)

Color Bleeding



Komposition



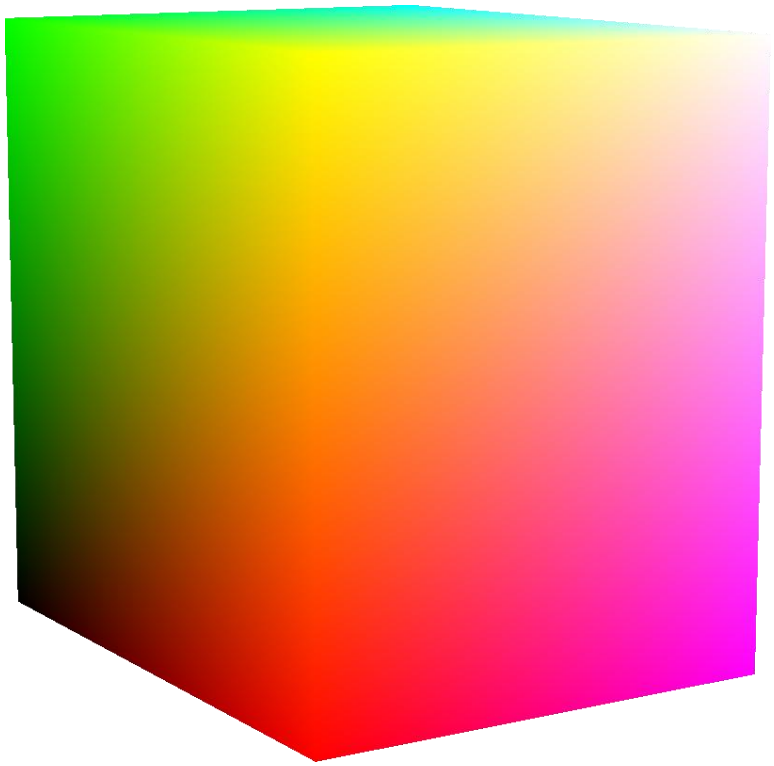
$$C_{dst} \leftarrow C_{dst} + (1 - \alpha_{dst}) * C_{src} * \alpha_{src}$$

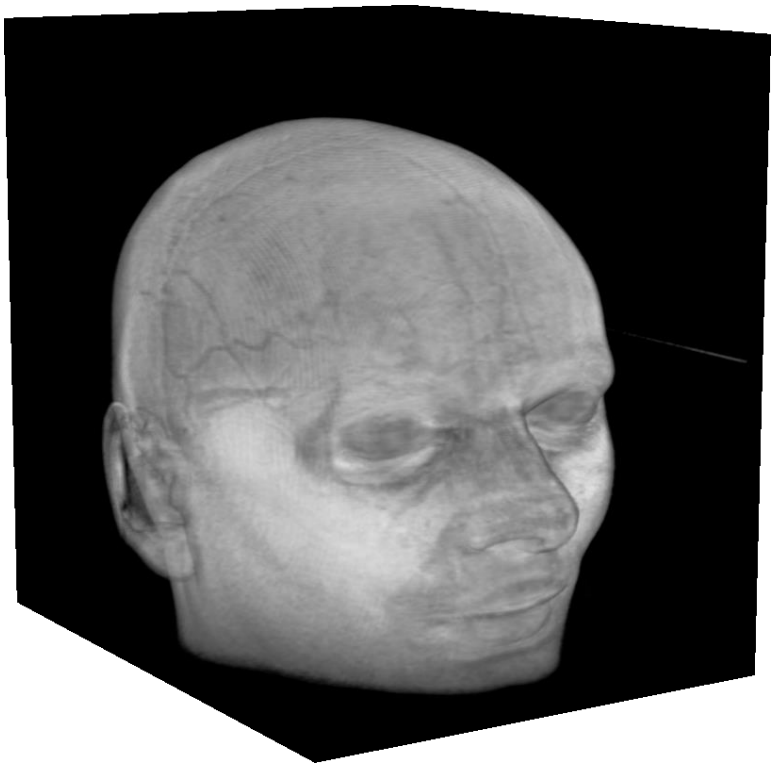
$$\alpha_{dst} \leftarrow \alpha_{dst} + (1 - \alpha_{dst}) * \alpha_{src}$$

Ray-Casting Algorithmus

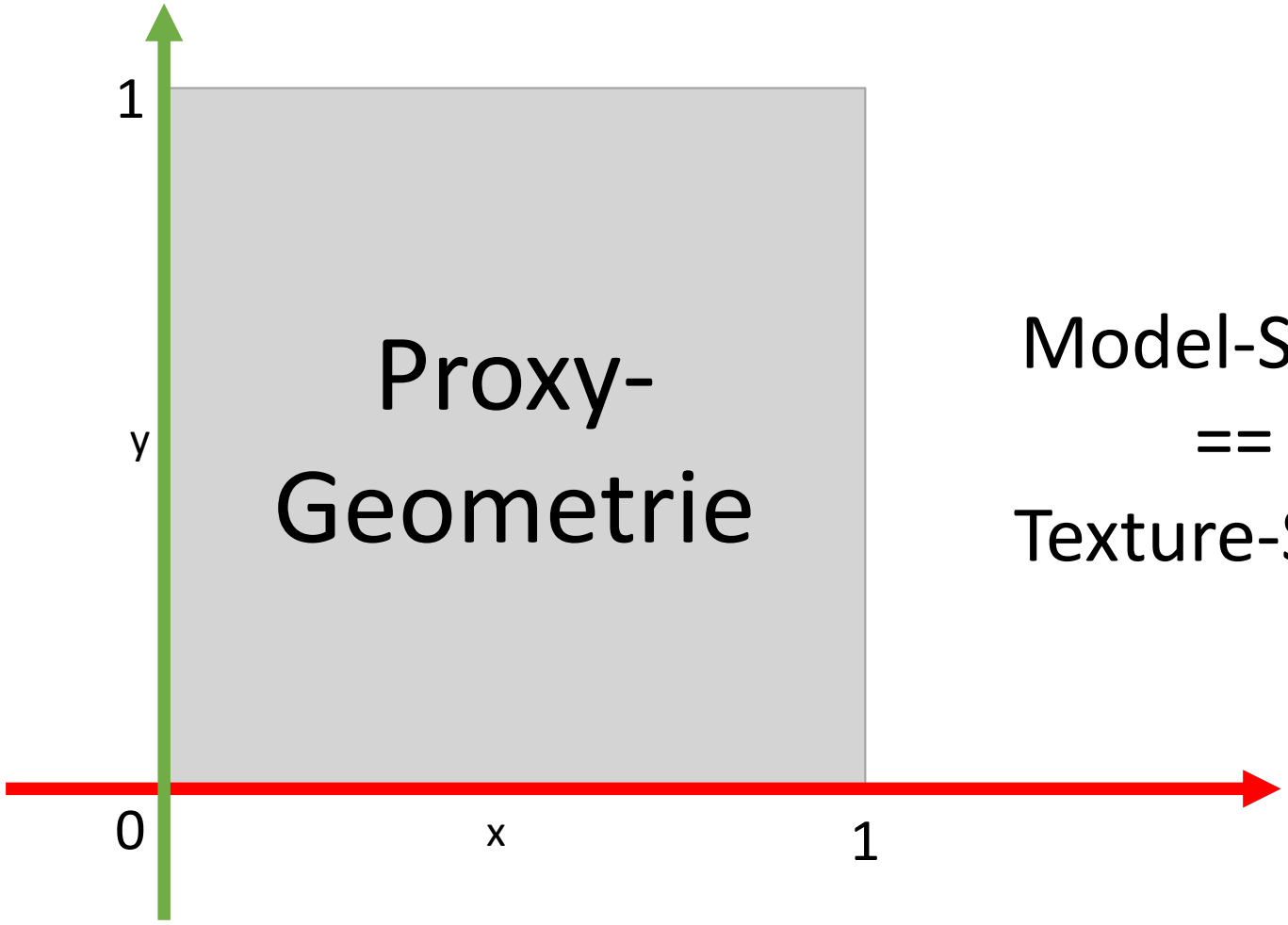
```
dst = 0;

for(Iterationen)
{
        src.rgba = Volumen(pos);
        dst.rgb = dst.rgb + (1-dst.a) * src.rgb * src.a;
        dst.a = dst.a + (1-dst.a) * src.a;
        pos =  pos + Abtastweite * Strahlrichtung;
}

return dst;
```

OpenGL / GLSL





Ray-Casting Proxy-Geometrie



Model-Space
==
Texture-Space

Ray-Casting Shader

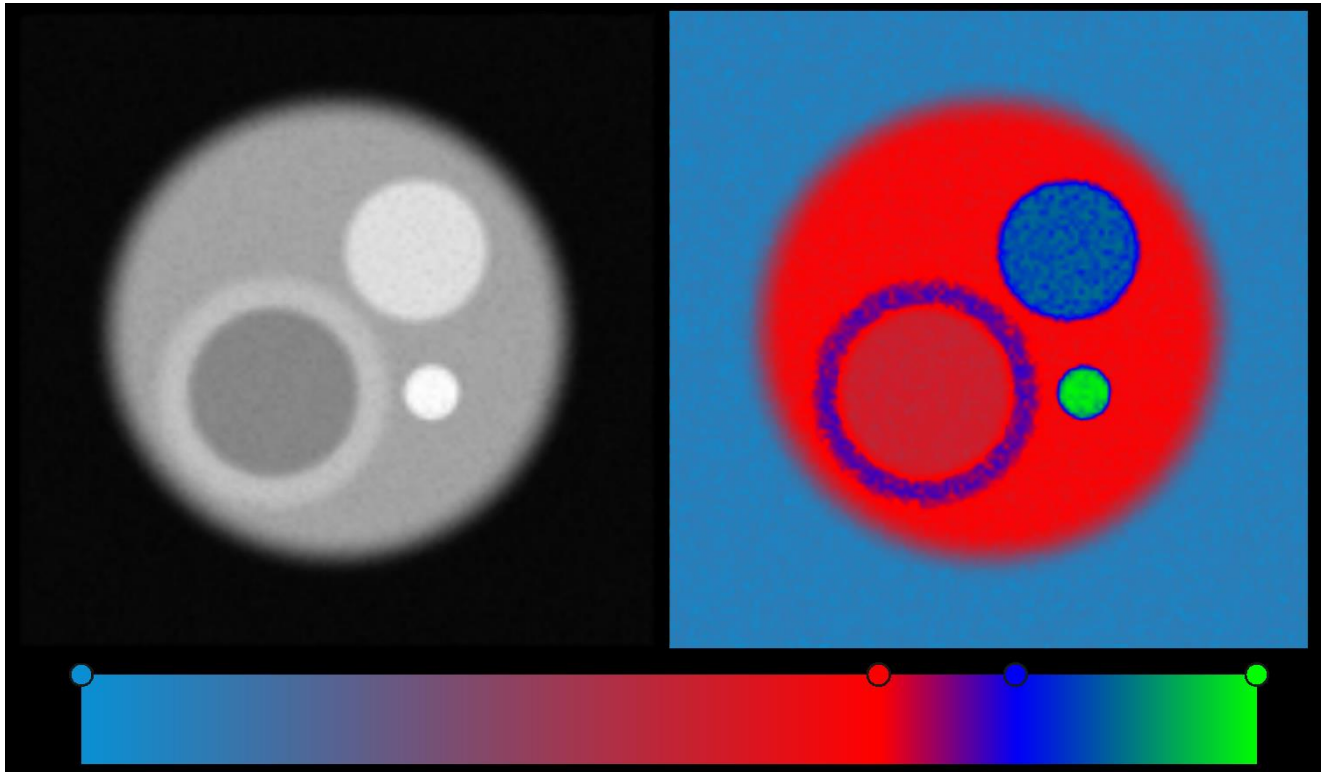
```
#version 330 core
in vec3 position;
in vec3 direction;

out vec4 fragmentColor;

uniform sampler3D uniformVolume;

const float STEP_SIZE = 0.008;
const float ITERATIONS = 1000;

void main()
{
  // Use input from vertex shader
  vec3 dir = normalize(direction);
  vec3 pos = position;

  // Variables for composition
  float src;
  float dst = 0;
```

...

Ray-Casting Shader

```
  // Do raycasting
  for(int i = 0; i < ITERATIONS; i++)
  {
    // Get value from volume
    src = texture(uniformVolume, pos).r;

    // Front-To-Back composition (only alpha part)
    dst += (1.0-dst) * src;

    // Prepare for next sample
    pos += dir*STEP_SIZE;

    // Check whether still in volume
    if(pos.x > 1 || pos.y > 1 || pos.z > 1 ||
      pos.x < 0 || pos.y < 0 || pos.z < 0)
    {
      break;
    }
  }

  // Output
  fragmentColor = vec4(dst.rrr, 1);
}
```

Transferfunktion



Transferfunktion(Volumenwert) = Materialeigenschaft

Transferfunktion Shader

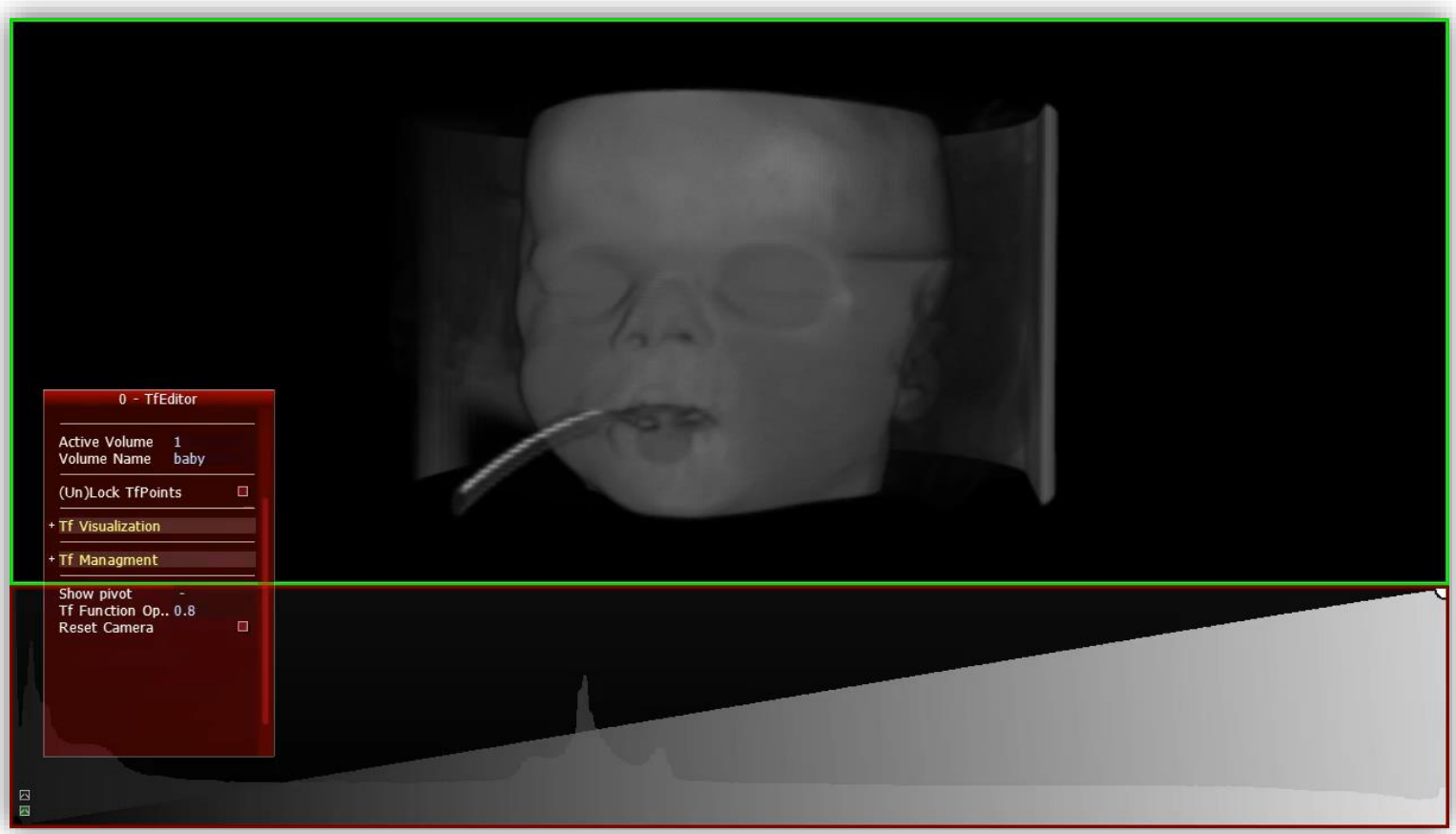
```
#version 330 core
in vec3 position;
in vec3 direction;

out vec4 fragmentColor;

uniform sampler3D uniformVolume;
uniform sampler1D uniformTransferfunction;

const float STEP_SIZE = 0.008;
const float ITERATIONS = 1000;

void main()
{
  // Use input from vertex shader
  vec3 dir = normalize(direction);
  vec3 pos = position;

  // Variables for composition
  vec4 src;
  vec4 dst = vec4(0,0,0,0);
  float value;
```

...

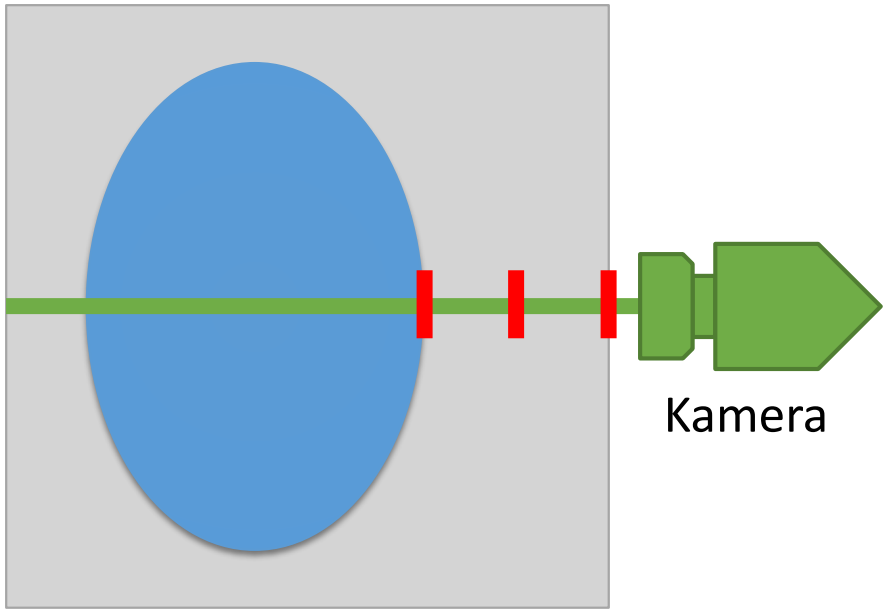
Transferfunktion Shader

```
  // Do raycasting
  for(int i = 0; i < ITERATIONS; i++)
  {
    // Get value from volume
    value = texture(uniformVolume, pos).r;

    // Use value as coodinate in transferfunction
    src = texture(uniformTransferfunction, value).rgba;

    // Front-To-Back composition
    dst.rgb += (1.0-dst.a) * src.rgb * src.a;
    dst.a += (1.0-dst.a) * src.a;

    // Prepare for next sample
    pos += dir*STEP_SIZE;

    // Check whether still in volume
    if(pos.x > 1 || pos.y > 1 || pos.z > 1 ||
      pos.x < 0 || pos.y < 0 || pos.z < 0)
    {
      break;
    }
  }

  // Output
  fragmentColor = vec4(dst.rgb, 1);
}
```

Transferfunktion Video



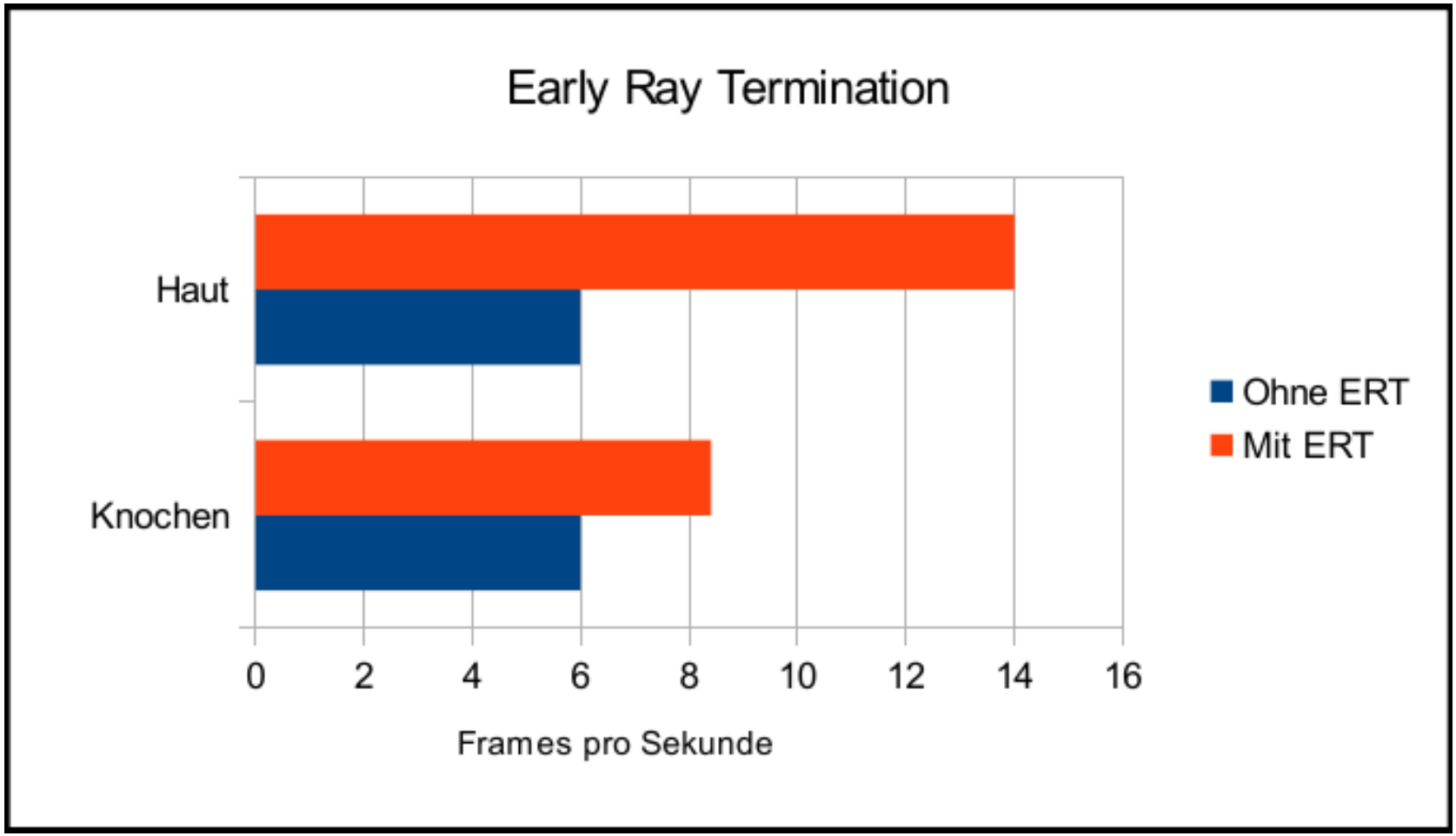
Early Ray Termination (ERT)



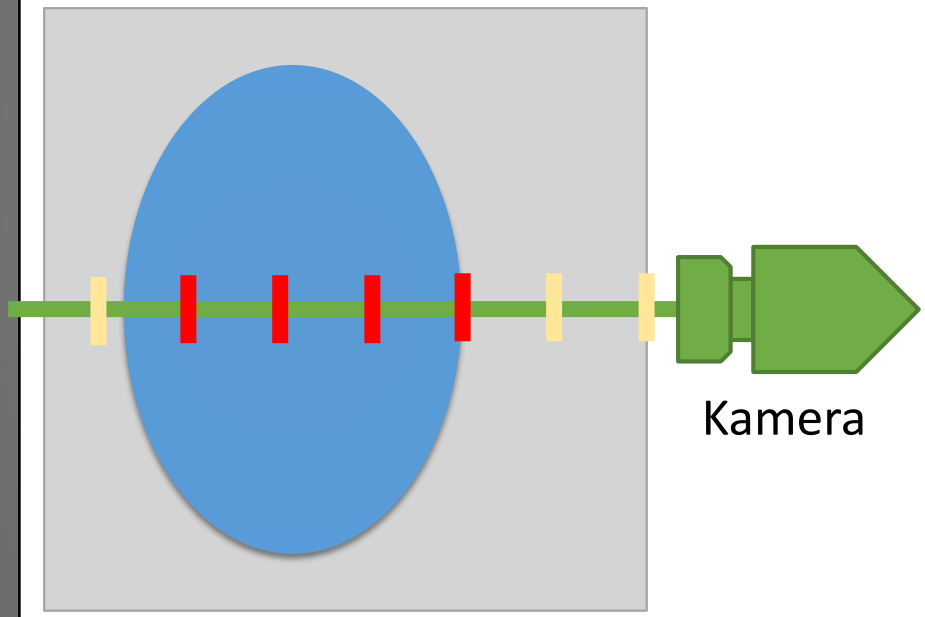

```
#version 330 core
...
const float ALPHA_THRESHOLD = 0.95;

void main()
{
  ...

  // Do raycasting
  for(int i = 0; i < ITERATIONS; i++)
  {
    ...

    // Front-To-Back composition
    dst.rgb += (1.0-dst.a) * src.rgb * src.a;
    dst.a += (1.0-dst.a) * src.a;
    ...

    // Early Ray Termination
    if(dst.a > ALPHA_THRESHOLD)
    {
      break;
    }
  }
  ...
}
```

Early Ray Termination (ERT)

Early Ray Termination (ERT)



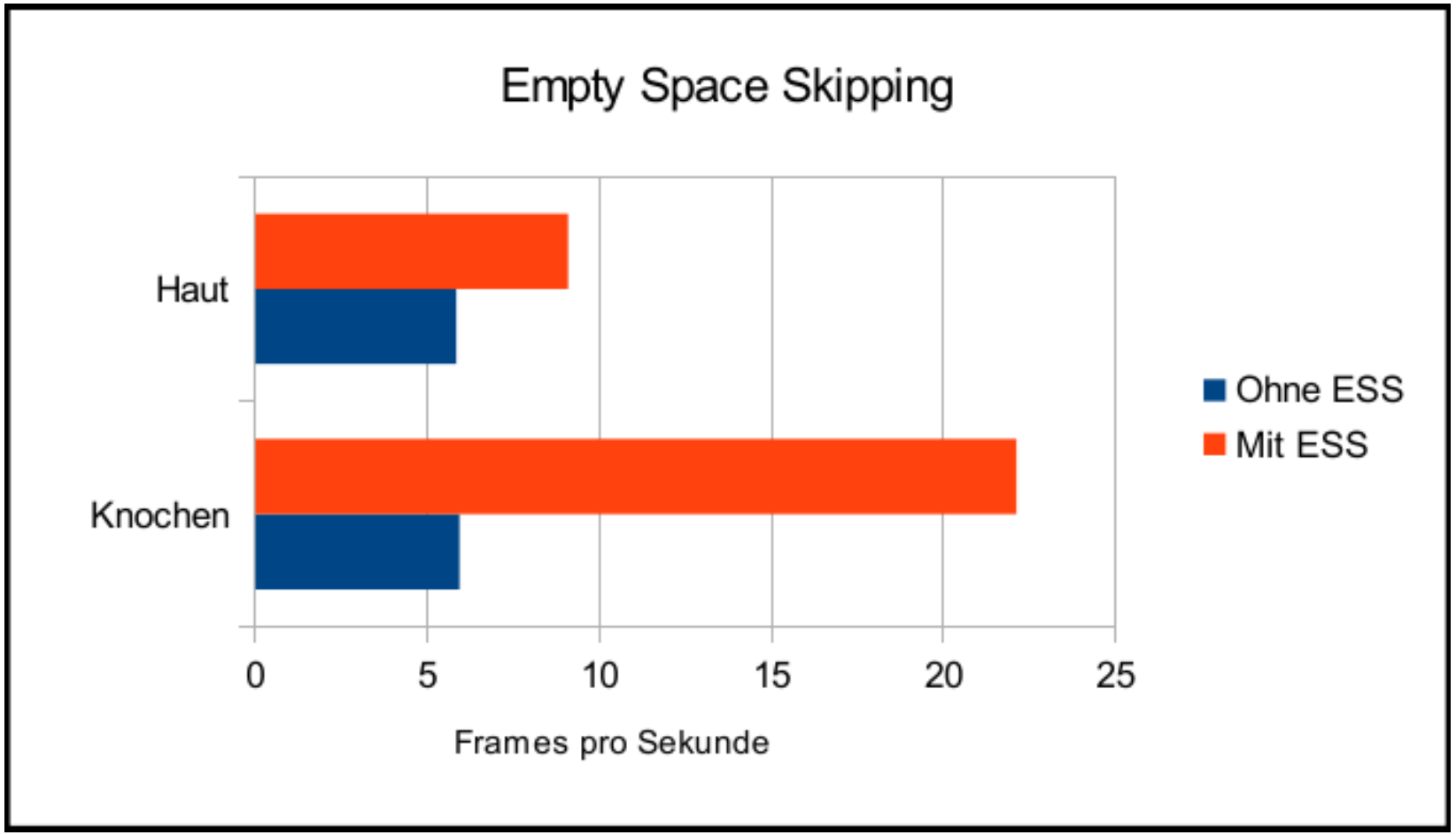
EMPTY SPACE SKIPPING (ESS)



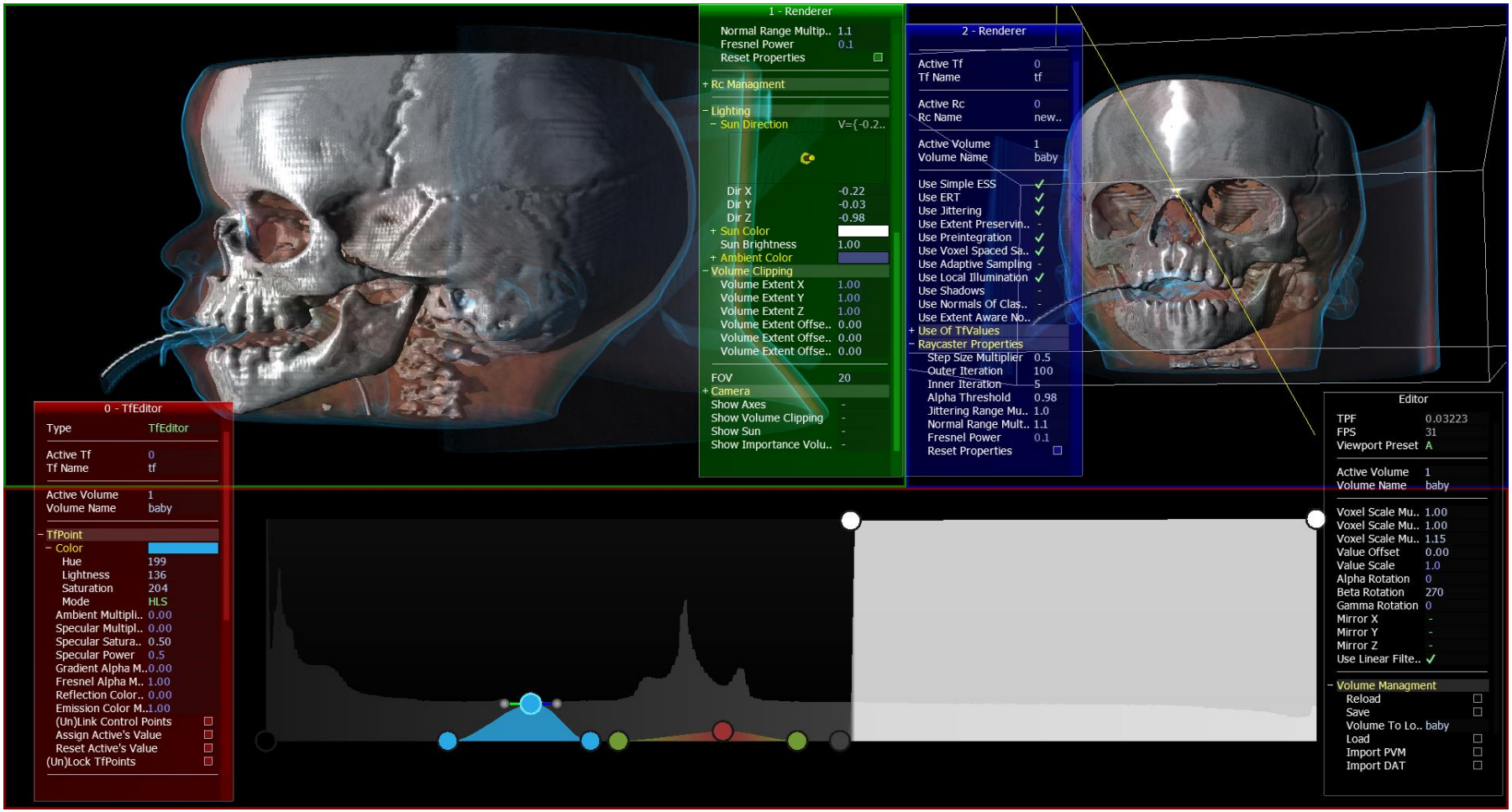

```
#version 330 core
...
const float EMPTY_SPACE_SKIPPING_THRESHOLD = 0.001;

void main()
{
  ...

  // Do raycasting
  for(int i = 0; i < ITERATIONS; i++)
  {
    // Get value from volume
    value = texture(uniformVolume, pos).r;

    // Use value as coodinate in transferfunction
    src = texture(uniformTransferfunction, value).rgba;

    // Empty Space Skipping
    if(src.a < EMPTY_SPACE_SKIPPING_THRESHOLD)
    {
      // Prepare for next sample and continue
      pos += dir*STEP_SIZE;
      continue;
    }
    ...

    // Operations like shading etc.
    ...

    // Front-To-Back composition
    dst.rgb += (1.0-dst.a) * src.rgb * src.a;
    dst.a += (1.0-dst.a) * src.a;
    ...
  }
  ...
}
```

EMPTY SPACE SKIPPING (ESS)

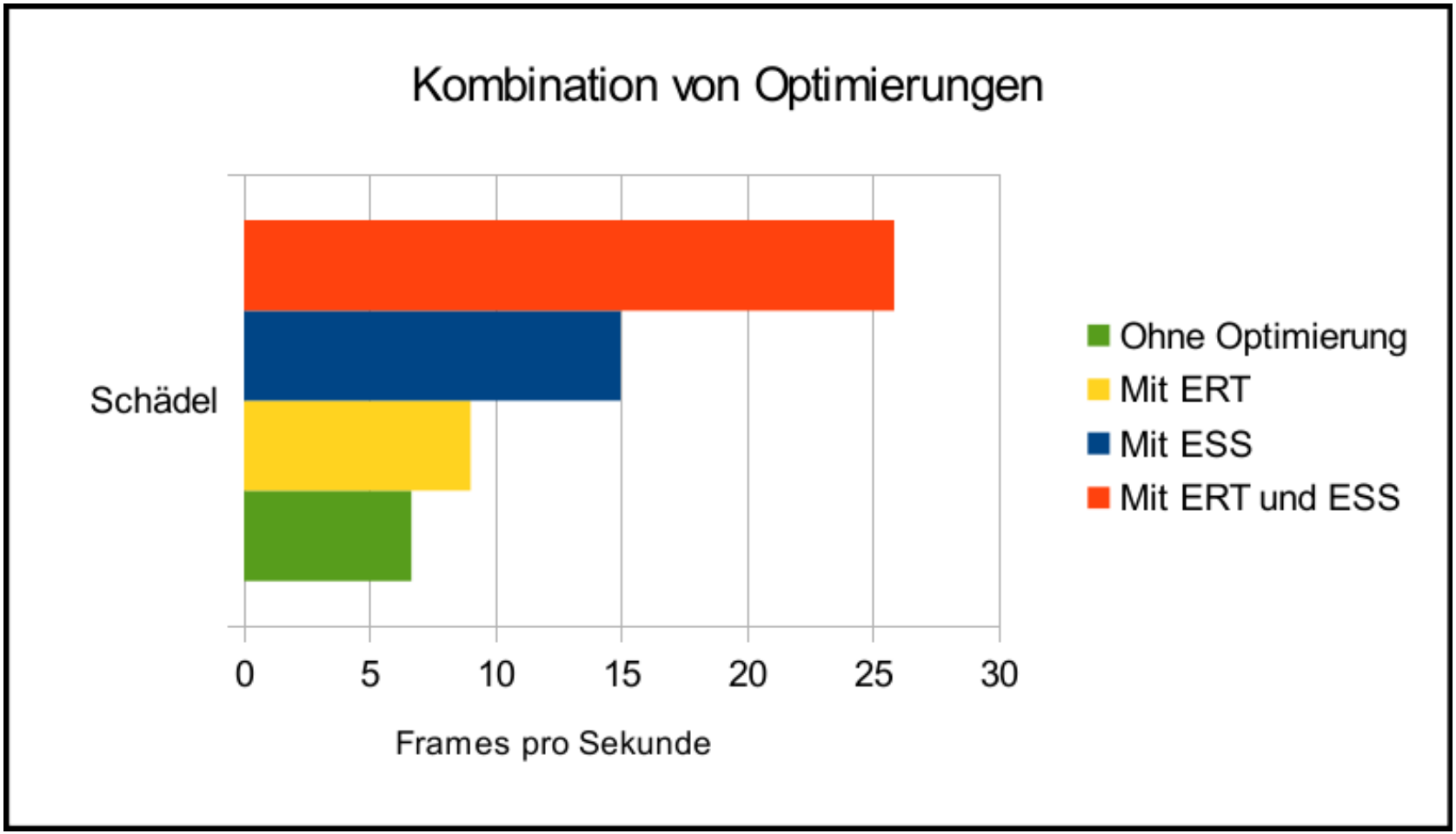
EMPTY SPACE SKIPPING (ESS)



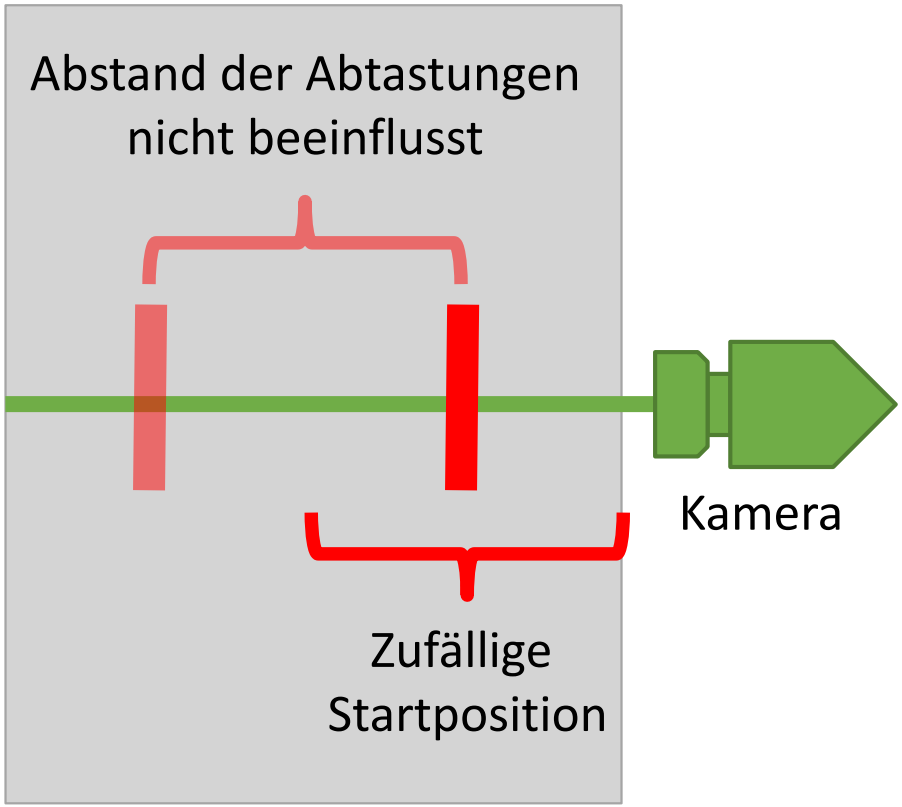
ERT + ESS


1 - Renderer
Normal Range Multip.. 1.1
Fresnel Power 0.1
Reset Properties
+ Rc Managment
– Lighting
– Sun Direction V={-0.2..
Dir X -0.22
Dir Y -0.03
Dir Z -0.98
+ Sun Color
Sun Brightness 1.00
+ Ambient Color
– Volume Clipping
Volume Extent X 1.00
Volume Extent Y 1.00
Volume Extent Z 1.00
Volume Extent Offse.. 0.00
Volume Extent Offse.. 0.00
Volume Extent Offse.. 0.00
FOV 20
+ Camera
Show Axes -
Show Volume Clipping -
Show Sun -
Show Importance Volu.. -
2 - Renderer
Active Tf 0
Tf Name tf
Active Rc 0
Rc Name new..
Active Volume 1
Volume Name baby
Use Simple ESS ✓
Use ERT ✓
Use Jittering ✓
Use Extent Preservin.. -
Use Preintegration ✓
Use Voxel Spaced Sa.. ✓
Use Adaptive Sampling -
Use Local Illumination ✓
Use Shadows -
Use Normals Of Clas.. -
Use Extent Aware No.. -
+ Use Of TfValues
– Raycaster Properties
Step Size Multiplier 0.5
Outer Iteration 100
Inner Iteration 5
Alpha Threshold 0.98
Jittering Range Mu.. 1.0
Normal Range Mult.. 1.1
Fresnel Power 0.1
Reset Properties
0 - TfEditor
Type TfEditor
Active Tf 0
Tf Name tf
Active Volume 1
Volume Name baby
– TfPoint
– Color
Hue 199
Lightness 136
Saturation 204
Mode HLS
Ambient Multipli.. 0.00
Specular Multipl.. 0.00
Specular Satura.. 0.50
Specular Power 0.5
Gradient Alpha M.. 0.00
Fresnel Alpha M.. 1.00
Reflection Color.. 0.00
Emission Color M.. 1.00
(Un)Link Control Points
Assign Active's Value
Reset Active's Value
(Un)Lock TfPoints
Editor
TPF 0.03223
FPS 31
Viewport Preset A
Active Volume 1
Volume Name baby
Voxel Scale Mu.. 1.00
Voxel Scale Mu.. 1.00
Voxel Scale Mu.. 1.15
Value Offset 0.00
Value Scale 1.0
Alpha Rotation 0
Beta Rotation 270
Gamma Rotation 0
Mirror X -
Mirror Y -
Mirror Z -
Use Linear Filte.. ✓
– Volume Managment
Reload
Save
Volume To Lo.. baby
Load
Import PVM
Import DAT

ERT + ESS

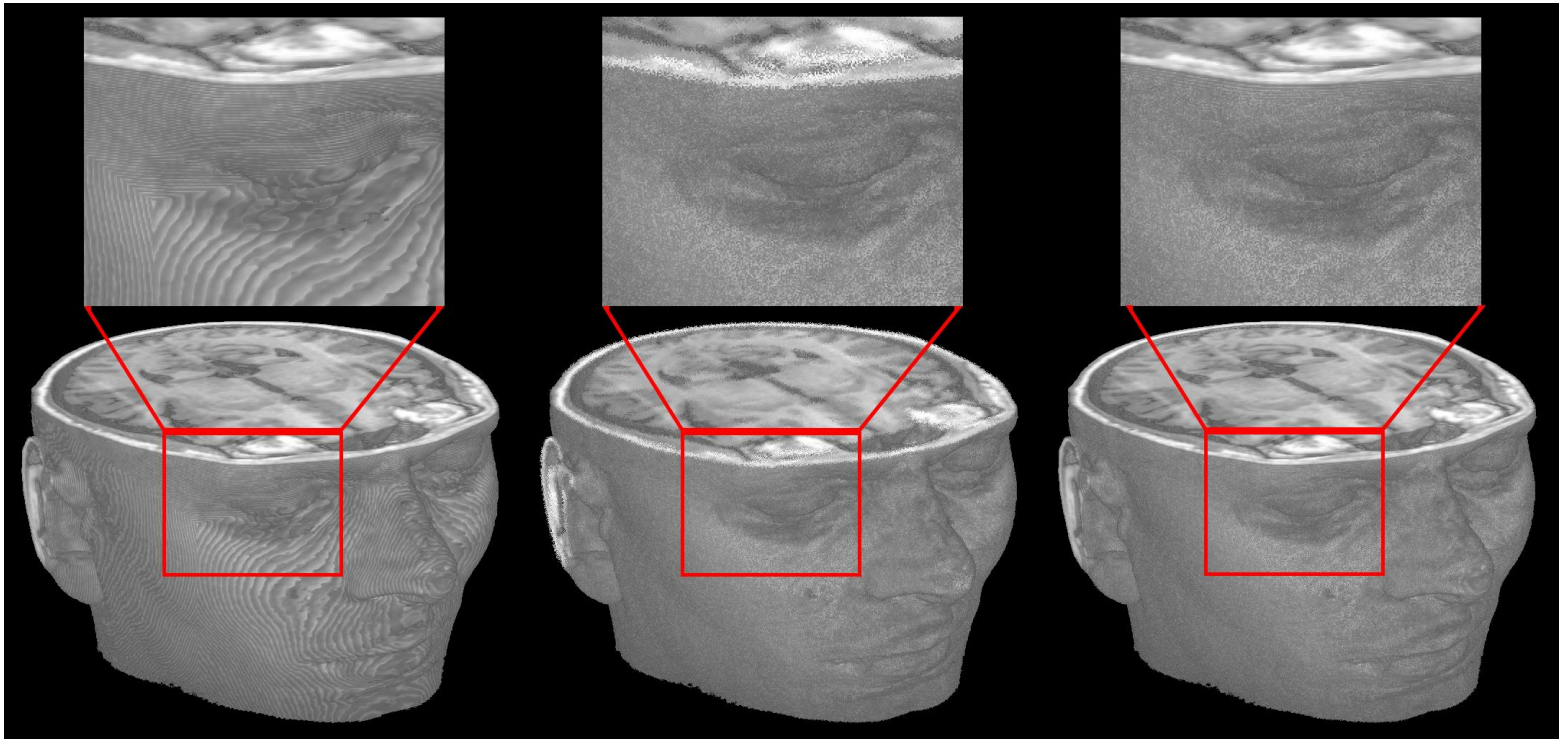

Kombination von Optimierungen
Schädel
0
5
10
15
20
25
30
Frames pro Sekunde
Ohne Optimierung
Mit ERT
Mit ESS
Mit ERT und ESS

Stochastic Jittering

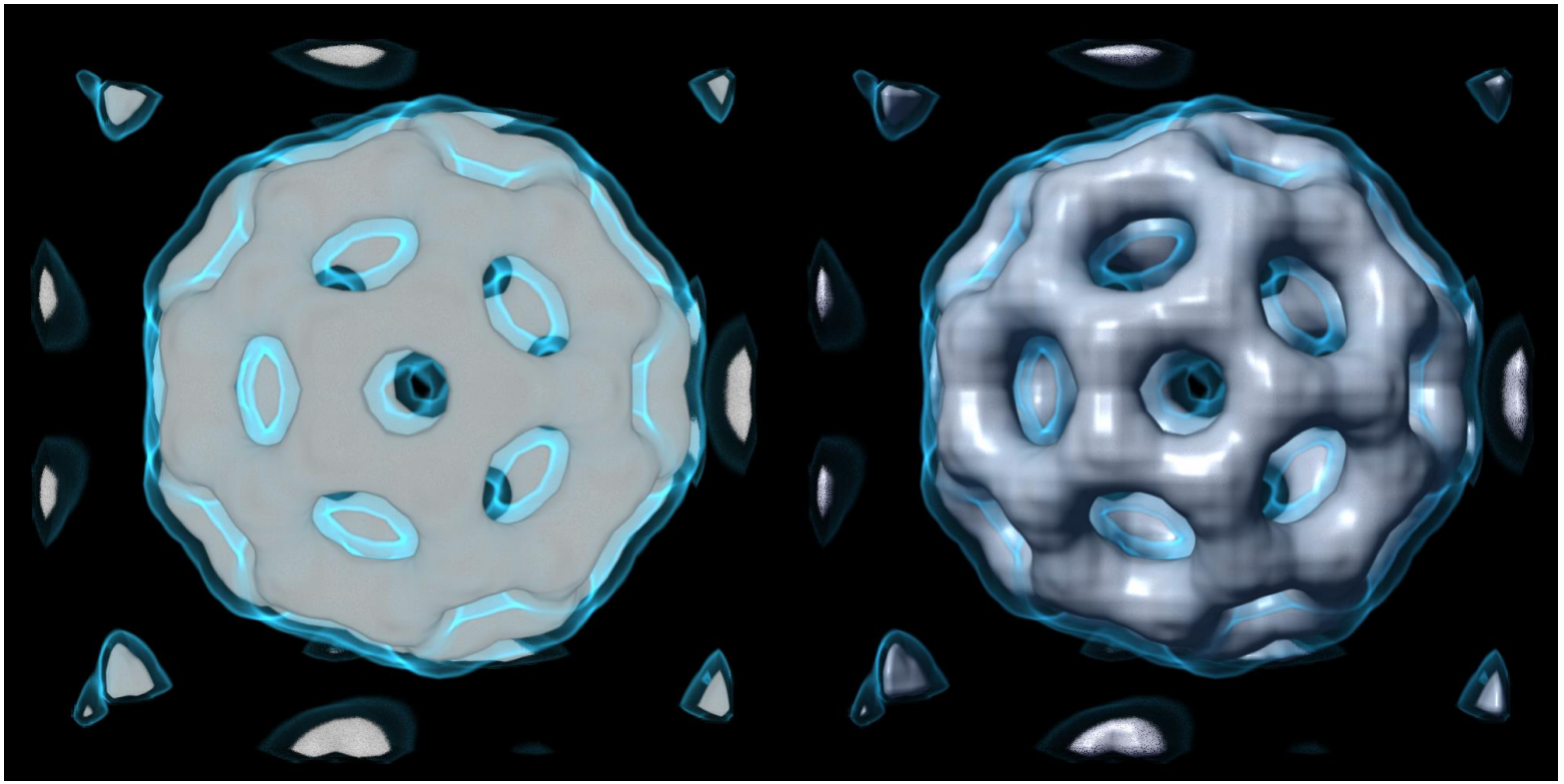



```
#version 330 core
...
uniform sampler2D uniformNoise;

// Resolution of noise texture
const float NOISE_RES = 64;

void main()
{
  ...

  // Read random value from noise texture
  float jitter = texture(uniformNoise,
            gl_FragCoord.xy/NOISE_RES).r;

  // Expand ray before start of iterations
  pos += dir * STEP_SIZE * jitter;

  // Do raycasting
  for(int i = 0; i < ITERATIONS; i++)
  {
    ...
  }
  ...
}
```

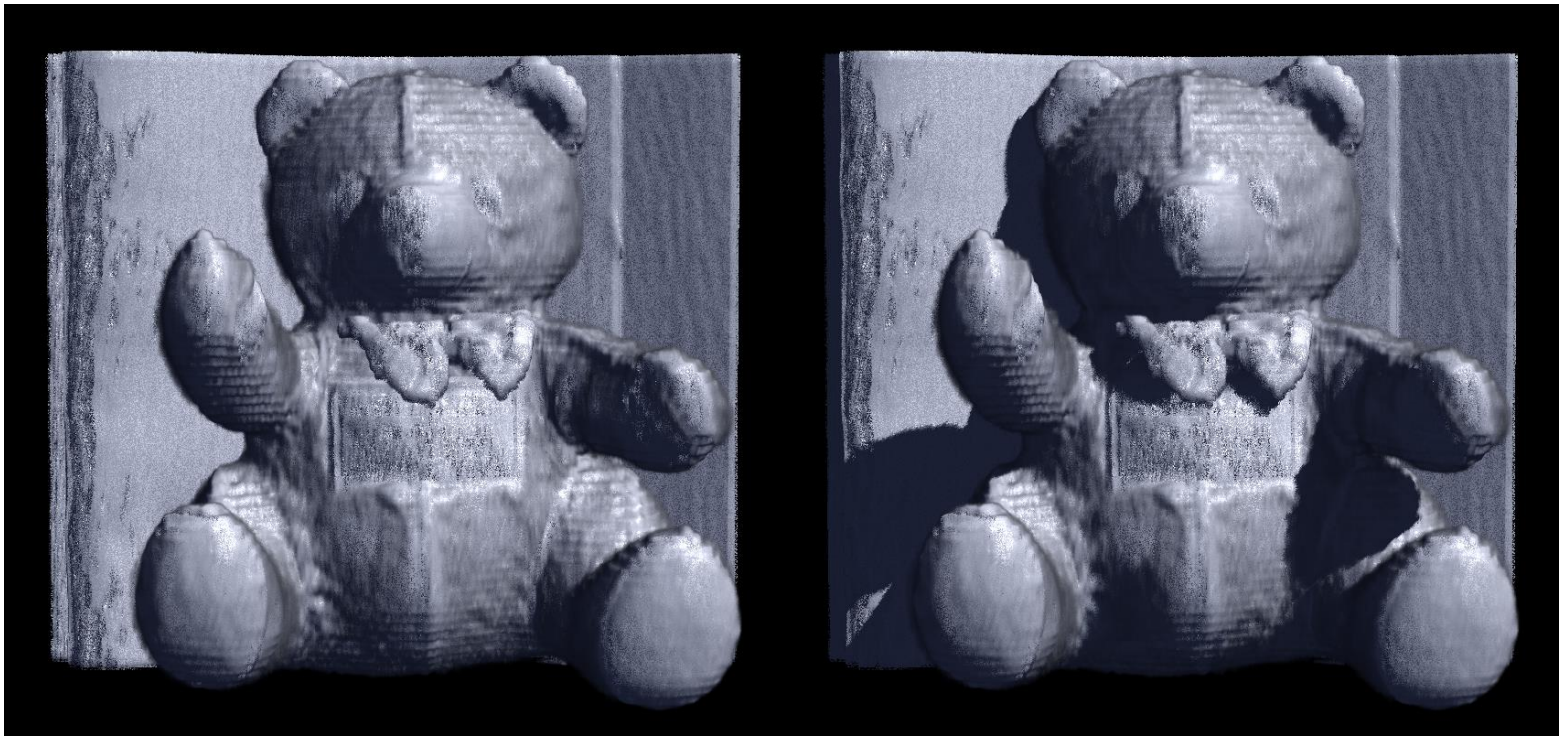
Stochastic Jittering



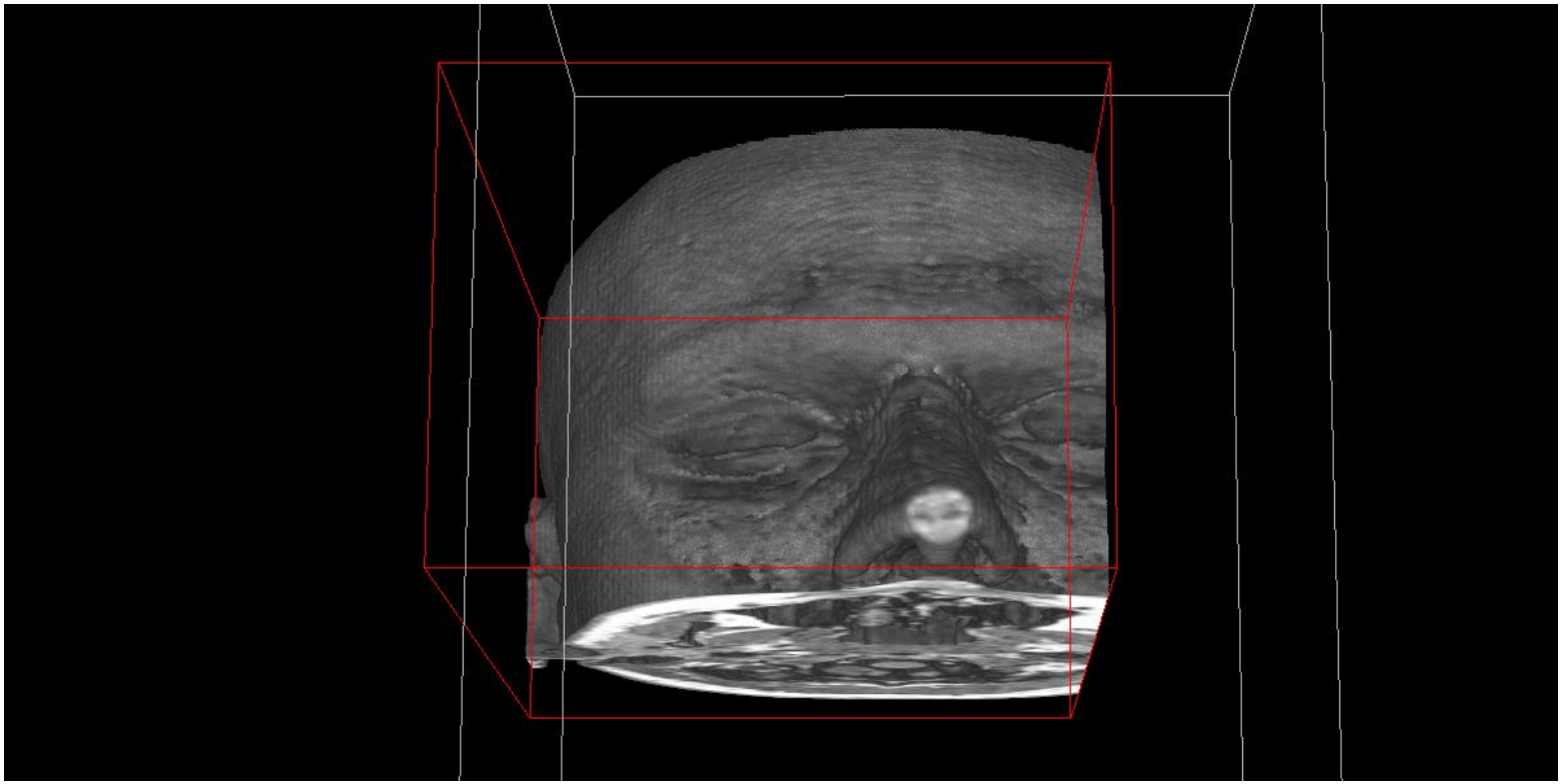
Local Illumination



Direkte Schatten



Volume Clipping



Quellen

- Alle Volumen auf den Abbildungen aus: http://www9.informatik.uni-erlangen.de/External/vollib/
  - MRI Head
  - Test Spheres
  - Baby Head
  - Foot
  - Bucky Ball
  - Teddy Bear