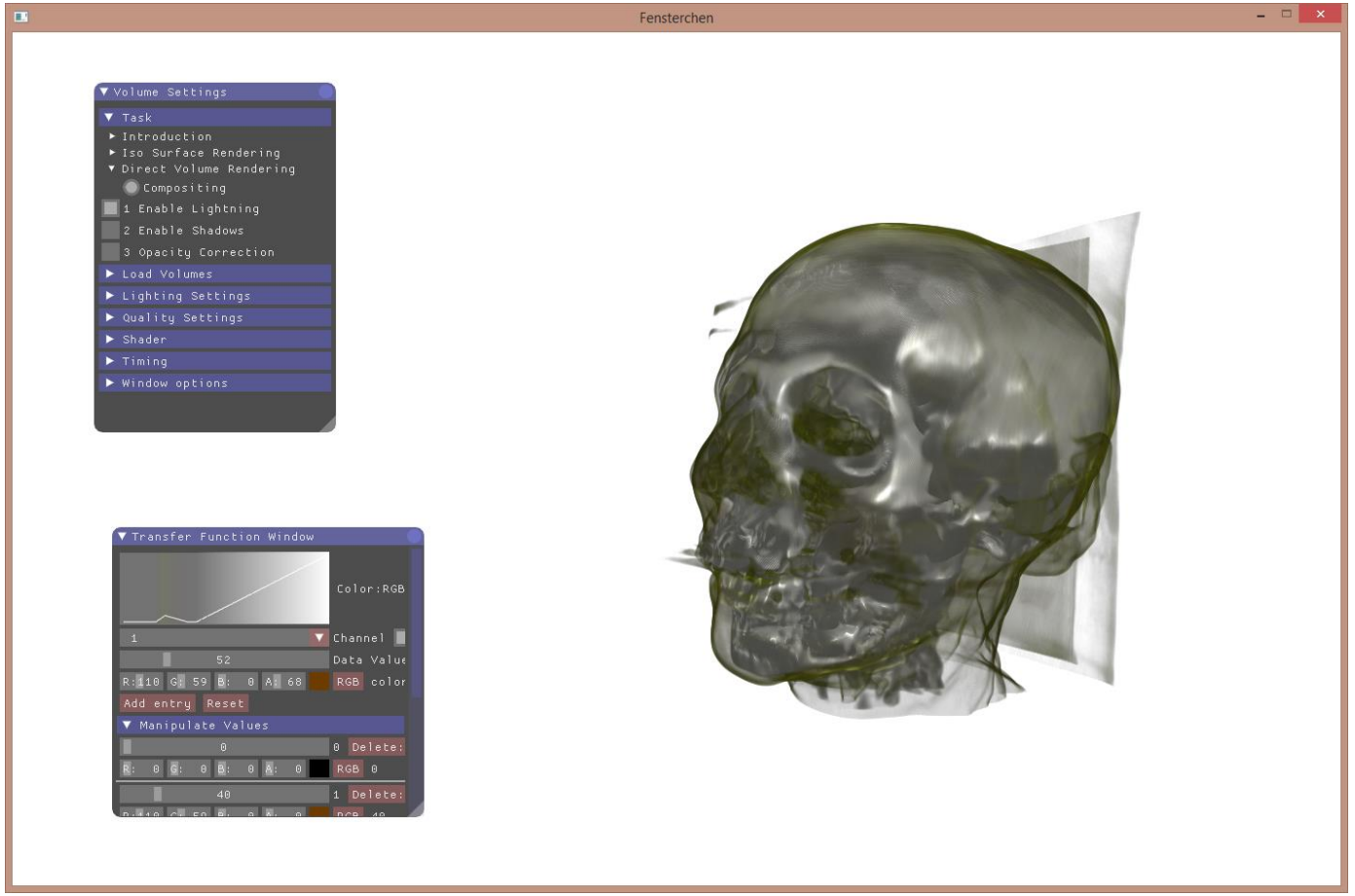
Lab Class Visualization

- Part 1: Assignments in InfoVis: *20 Pt.*
  (Patrick Riehmann, Henning Gründl)
- Part 2: Assignments in SciVis: *20 Pt.*
  (Sebastian Thiele, Carl-Feofan Matthes)
- Part 3: Final Project: *60 Pt.*
  (Patrick Riehmann, Henning Gründl, Sebastian Thiele)

Final Project Vis

- Topic: Up to you (Either InfoVis or SciVis)
- Expenditure of time: Ca. 80h/Student
- Requirements:
  - Autonomous implementation / No groups allowed
  - Unique and fresh kind of visualization
  - At least two complex interaction techniques

SciVis - Assignments



Implementation of various methods and techniques in **volume visualization**

SciVis - Assignments

- 8 Assignments with overall 20 points
- Use lecture videos and slides of this and the last year, if you don’t know what to do
  - 2014: http://www.uni-weimar.de/de/medien/professuren/vr/teaching/ss-2014/course-visualization/
    - User: vr
    - Pwd: vr2014_ss
  - 2015: http://www.uni-weimar.de/de/medien/professuren/vr/teaching/ss-2015/course-visualization/
    - User: vr
    - Pwd: vr2015ss_buw

SciVis-Assignments

- Only who passed the InfoVis assignment will be allowed to attend the final submission of the SciVis part
- Final submission after last class
  - Location: Lint Pool
  - Enroll: List in front of office (Bauhausstr. 11, first floor)
  - Possible date: Tue, 07.07.2015

ExSciVis

- Requirements
  - OpenGL 3.2
    - Ivy Bridge or dedicated GPU
  - C++ 11

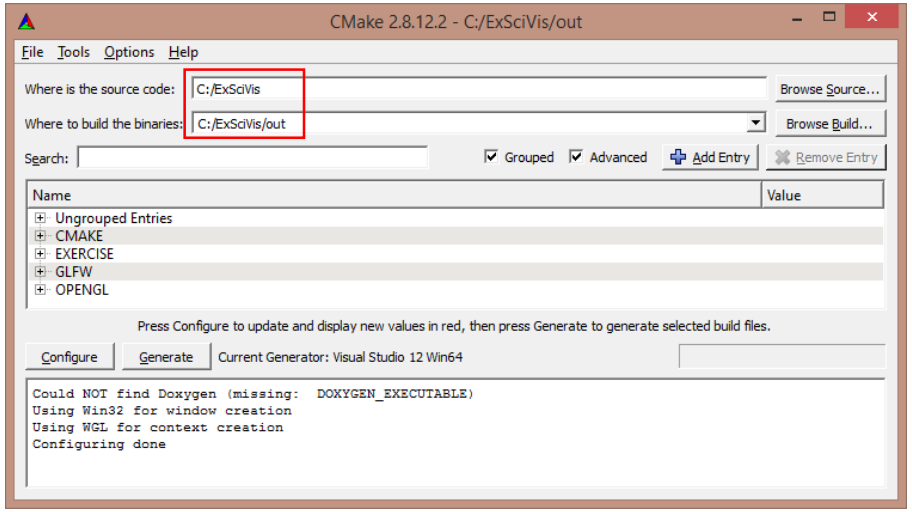
- OpenGL 3.2 Framework
- Raycaster implemented in **OpenGL Shading Language** GLSL
  - Many build in functions
- No need to touch any C++ files
  - But you can, if you want to experiment

Setup Project

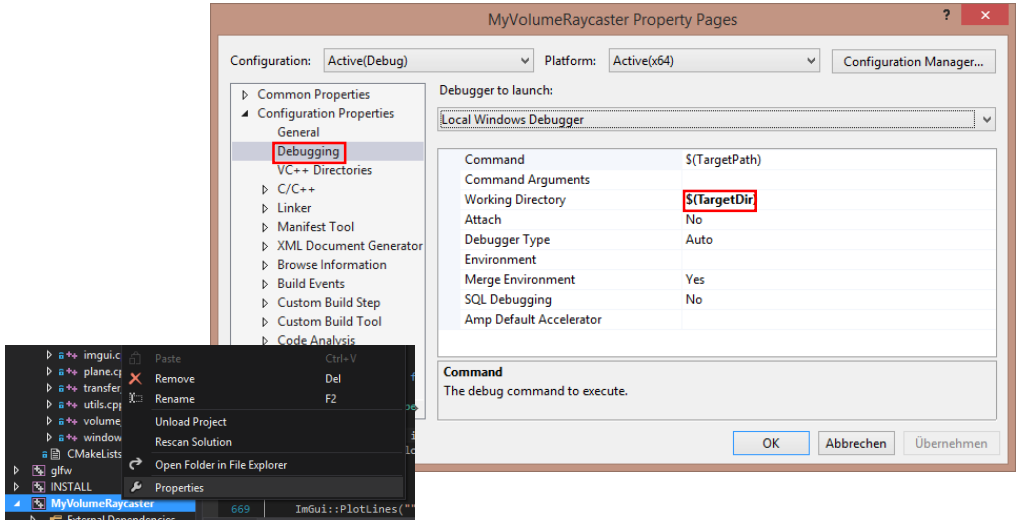
- Go to https://github.com/vrsys/ExSciVis/ (Optional)
  - Fork project (Optional)
- Download project
  - https://github.com/vrsys/ExSciVis/archive/master.zip
  - Or clone (Optional)
- Generate project file/makefile with cmake
- Build and start project

Setup Project (Windows only)

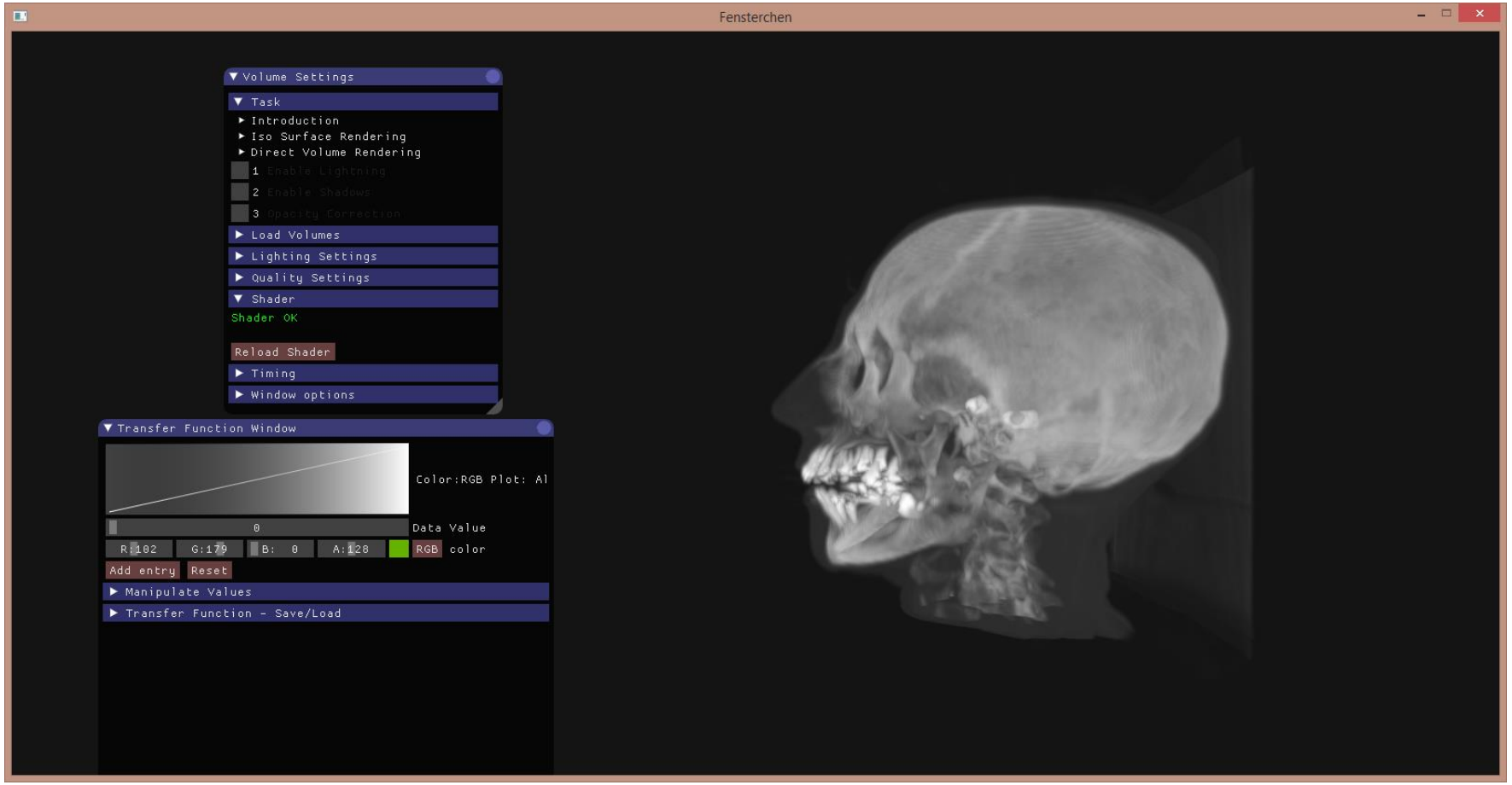
- Important
  - Build project files in folder inside source folder



- Change Working directory to '$(TargetDir)'



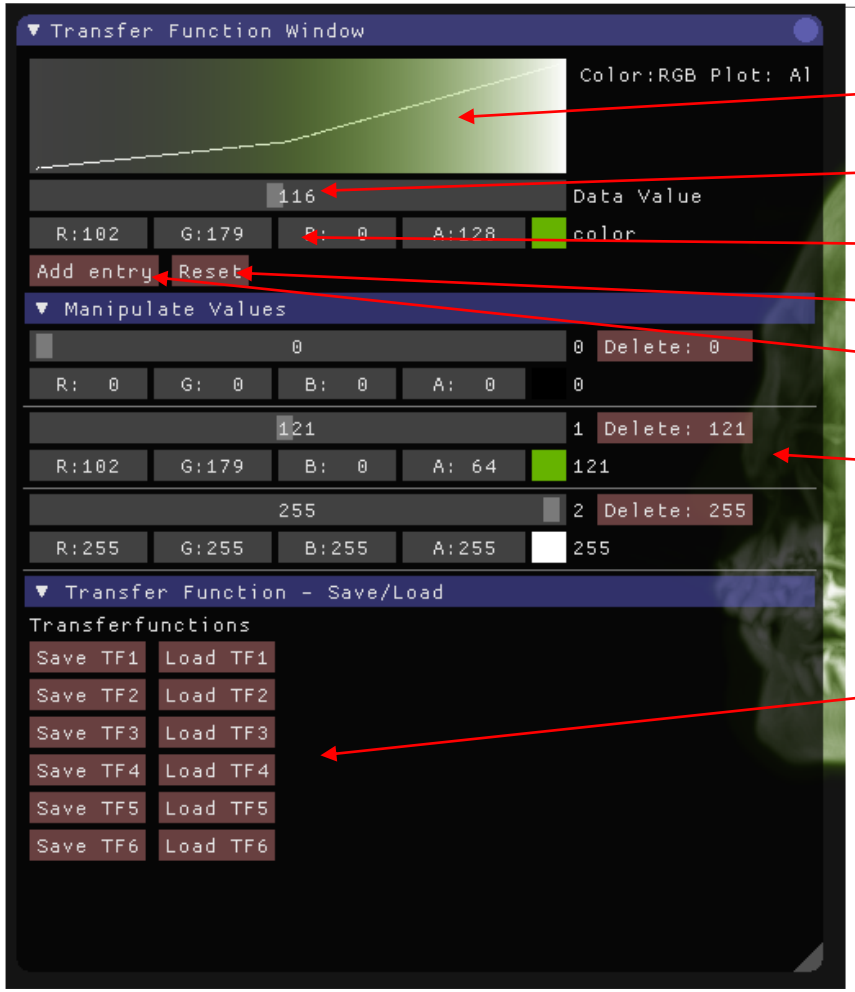
ExSciVis – First Look



ExSciVis – GUI Volume Settings

- Task -> Assignment Related
  - Introduction -> Assignment 1
  - Iso Surface Rendering -> Assignment 2 & 3 & 4
  - Direct Volume Rendering -> Assignment 5 & 6 & 7
- Load Volumes -> Different datasets to play with
- Lighting Settings -> Change colors, values and position of lights (assignment 4 & 5)
- Quality Settings -> Change sampling rate (hint: assignment 6), interpolation
- Shader -> Reload shader and debug your code (more info later)
- Timing -> Shows and visualizes framerates
- Window Option
  - Window size -> change the resolution of the window
  - Background color-> yes ;)

ExSciVis – GUI Transfer Function



Current transfer function

Set data value to add color/opacity

Set color/opacity

Delete all values

Add Entry for data value and edited color

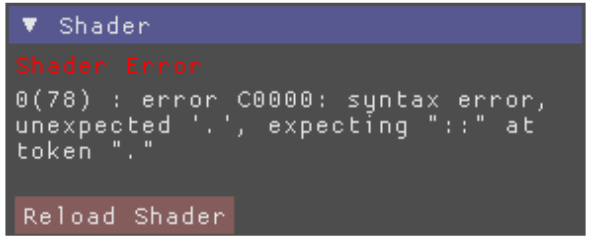
Manipulate existing entries here

Save/load transfer functions for later use

GLSL

- Framework uses the graphic card capabilities to increase rendering performance
- Everything happens inside .\ExSciVis\source\shader\volume.frag
- You don’t need to restart the program after making changes inside the file, simply press the reload shader button

- If there is an error in your code, it will be shown here
  - Program will keep running with last working shader



ExSciVis – Background

- For each fragment (vertex projection of proxy geometry) a ray is generated
- Code is executed on the gpu for each fragment
- Volume is a 3D Texture (1 Channel, 8Bit per channel)
  - 0.0 – empty/air
  - 1.0 – highest density

Code Peak

```
63  #if TASK == 21 // ASSIGNMENT 1
64      vec4 max_val = vec4(0.0, 0.0, 0.0, 0.0);
65
66      // the traversal loop,
67      // termination when the sampling position is outside volume boundarys
68      // another termination condition for early ray termination is added
69      while (inside_volume)
70      {
71          // get sample
72          float s = get_sample_data(sampling_pos);
73
74          // apply the transfer functions to retrieve color and opacity
75          vec4 color = texture(transfer_texture, vec2(s, s));
76
77          // this is the example for maximum intensity projection
78          max_val.r = max(color.r, max_val.r);
79          max_val.g = max(color.g, max_val.g);
80          max_val.b = max(color.b, max_val.b);
81          max_val.a = max(color.a, max_val.a);
82
83          // increment the ray sampling position
84          sampling_pos  += ray_increment;
85
86          // update the loop termination condition
87          inside_volume  = inside_volume_bounds(sampling_pos);
88      }
89
90      dst = max_val;
91  #endif
```

Code brackets for current assignment

Get data value at sample point

Get color value for data value

Do some magic ;-)

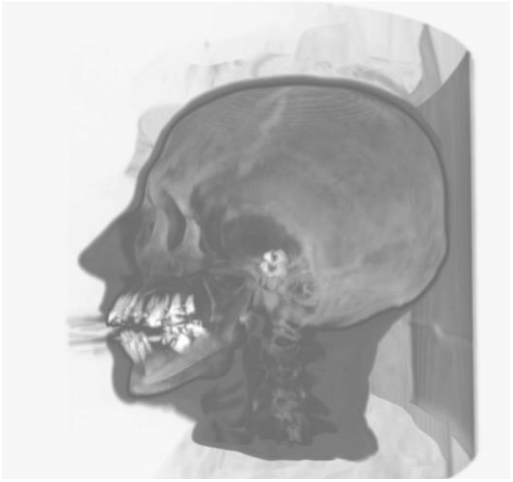
Increment ray

Check early ray termination

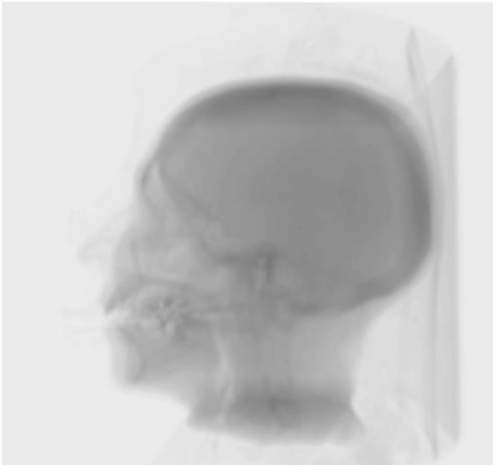
Code brackets for current assignment

Assignment 1 – Ray Traversal Schemes (2pts)

- Analyze and explain the implemented scheme!
  - Where happens the classification?



- Implement Average Intensity Projection



Assignment 2 – Iso Surface Rendering (3pts)

- Implement a first-hit ray traversal scheme for variable thresholds to visualize iso-surfaces

- Improve the intersection search using a binary search method
  - Hint: Do this after you finished assignment 4 to see the differences

Assignment 3 – Gradients (1pt)

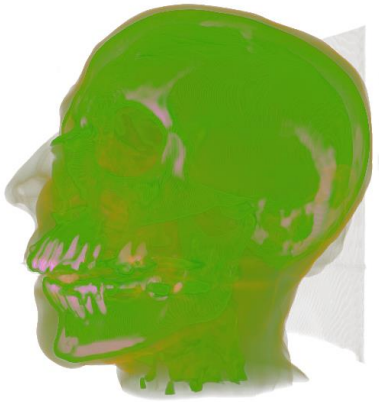
- Implement a function get_gradient() to calculate the gradient at a given volume sampling position
- For the actual gradient computation use the central differences method

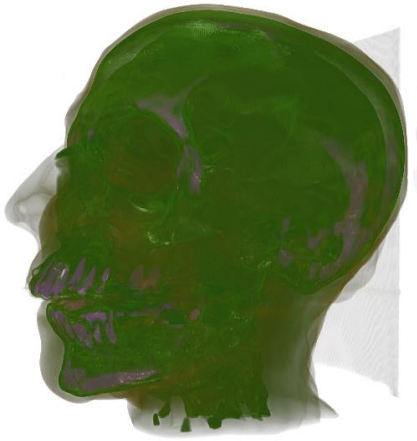
Assignment 4 – Shading (4pts)

- Determine the surface normal for the found intersection point and calculate a basic illumination for the iso-surface
  - Phong-model or the like
- Provide suggestions to improve the performance of the shading
- Extend the illumination calculation for the correct display of surface shadows

Assignment 5 – Compositing (4pts)

- Implement Compositing Traversal Scheme
  - Front-to-back
  - Back-to-front
- Explain the volume rendering integral in detail
- Use the generated volume gradients to calculate the local illumination for the volume samples during the compositing
  - Phong-model or the like
  - Pay attention to very small or zero-gradients





Assignment 6 – Opacity Correction (2pts)

- What is opacity correction?
- Where and why is it needed?

- Extend the existing compositing algorithm with opacity correction

Assignment 7 – Classification (2pts)

- What is Classification?
- What is the difference between pre- and post-classification?

- Define a transfer function with high frequencies
  - What are the problems?
  - How to deal with these problems?

Assignment 8 – Large Volume Rendering (2pts)

- How to deal with large volumes?
- What are the challenges?
- Which three steps are necessary to view a large volume dataset?