CAV Assignment 2

CAV Assignment 2

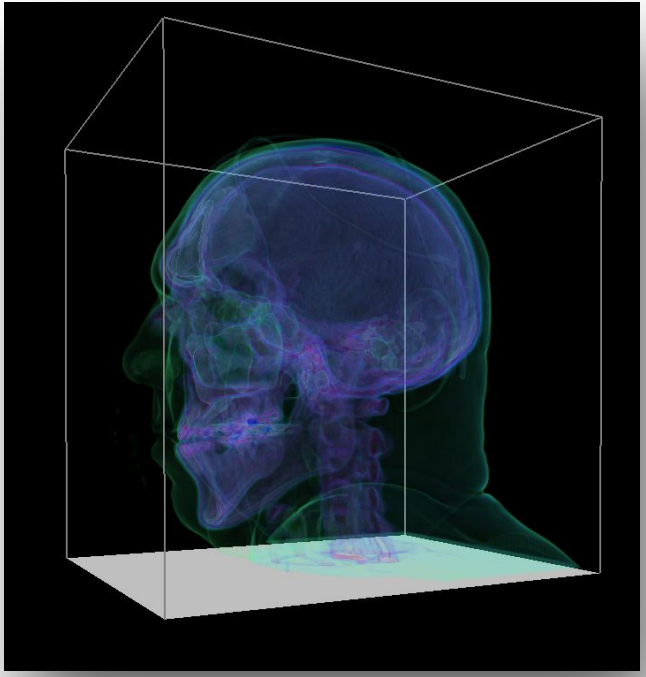
Volume Rendering

Floyd Chitalu

floyd.m.chitalu@ed.ac.uk

Aim

- Visualise volume data
- Convey “good” information
  - Skull, Skin, Form & Shape, etc
- Hide “bad” or “useless” information
  - Artefacts, Noise ...
- Interactive if possible



Requirements

- Raycasting-based volume renderer
- Step through volume by Composite Intensity Projection (CIP)
- Use Transfer Functions to map density values to optical properties
  - Color, opacity etc.
- Show the data from different viewpoints
- Marks
  - If skin/bone are clear and visualisation is good then should get mark of 70%
- Bonus Marks
  - Shear warping
  - Trilinear Interpolation
  - Local illumination and multidimensional transfer functions

Voxel Traversal

- Decide how to traverse into the volume
- Start by showing the volume from different axis
- Decide when to stop traversal
  - Threshold Value
  - Maximum Value
  - Average Value
  - Composite Value
- *How you accumulate values as you traverse is important!*

Transfer Functions

- Goal: Emphasize or classify features of interest in the data
  - … mapping values and other data measures to optical properties.
- Typically, implemented by lookup tables
  - … simple functions can also be computed.
- Important for showing skin/bone etc.
- Difficult and iterative design process
  - Requires significant insight into the underlying data set.
  - If unsure, design several which show different things well.
  - Plotting histograms to see distributions of density values can help.

3D Ray Tracing (*Algorithm*)

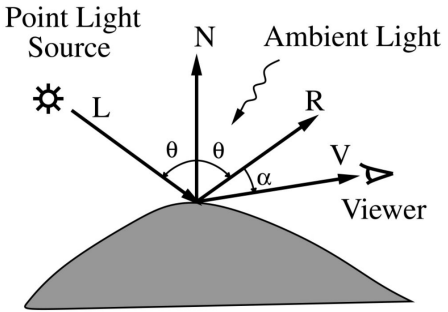
- For each pixel:
  - Find clip space coordinates (x, y). These are in the range (-1, 1)
- Augment with depth value to get *near-* and *far-clip* plane coordinates:
  - (x, y, -1) and (x, y, 1)
- Multiply these into world space
    - … using some *Inversed* Projection and View matrix
- *These are the endpoints of your viewport pixel ray in world space*

Volume Illumination

- Can help us better understand 3D structure of volume information
- Displays Visual Cues such as surfaces
- Highlights important gradients and makes them clear
  - The computed intensity is used to modulate the color components from the transfer function
- Allows for ISO surface display

Volume Illumination

- Voxels can scatter light, reflect light, or absorb light.
  - Local illumination (... ignores indirect light contributions, shadows etc.)
  - Global illumination
- In general, fully modelling scattering is very expensive
  - So, for now just model absorption and reflection.
- Use regular Phong Shading Model
  - Requires Normals (can be computed with gradient vectors)

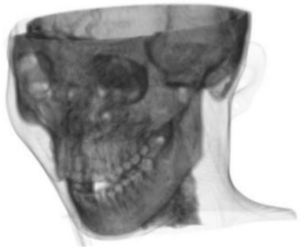


Estimating Normals

- 3D Volume has no “normal” only a gradient
- Calculate Gradient using Midpoint Method
  - Use it to find vector going in direction required (e.g. from most-dense to least-dense)

$$nx = V(x-1, y, z) - V(x+1, y, z)$$
$$ny = V(x, y-1, z) - V(x, y+1, z)$$
$$nz = V(x, y, z-1) - V(x, y, z+1)$$

$$N = (nx, ny, nz)$$



Result (Illuminated Volume)

Basic Demo Program

- “include” - Place your header (.h) files here
- “src” - Place your source (.cpp) files here
- “obj” - Intermediate build folder
- “main.cpp” - Program entry point
- “volumeData” - Volume data
- “Makefile” - Configuration file to build the project
- “README” - Instructions for use

Demo Classes

- “mat”
  - Contains Matrix classes (for 2x2, 3x3 and 4x4)
- “vec”
  - Contains Vector classes (for 2, 3, 4)
- “vol”
  - Used to load and access Volume Data
- *Feel free to use and extend where needed!*

Demo Classes

- Draw()
  - Executed every frame.
  - It is where your logic should go
    - … both for calculations and rendering.
  - Currently:
    - It traces in 2D and terminates once it reaches a density over a given value.
    - It then outputs the colour of that density
  - This is the main function you should edit

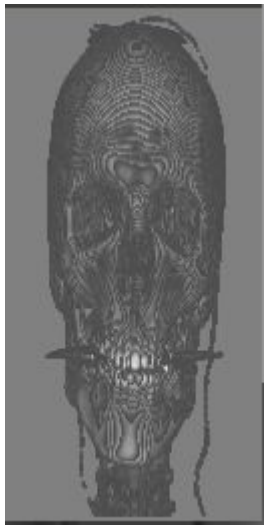
Demo Classes

- KeyEvent()
  - Here you can add interactive controls for keyboard presses.
  - See also “glutSpecialFunc”, “glutMouseFunc”, “glutMotionFunc” to add other kinds of interaction
- main()
  - You can add initialisation code here

Compiling & Running

- Compiling
  - Providing everything in correct place…
  - Just run “make”
  - Run “make clean” to remove any intermediate build files.
- Running the program
  - Just run “./cav”
- Any other problems contact me

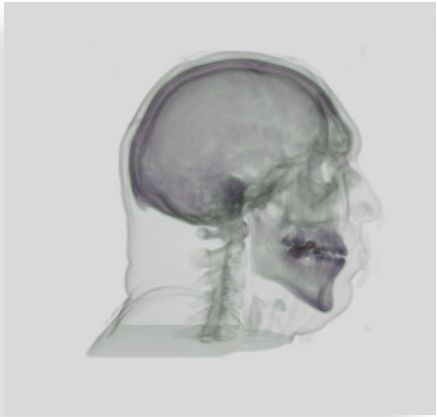
Sample Output

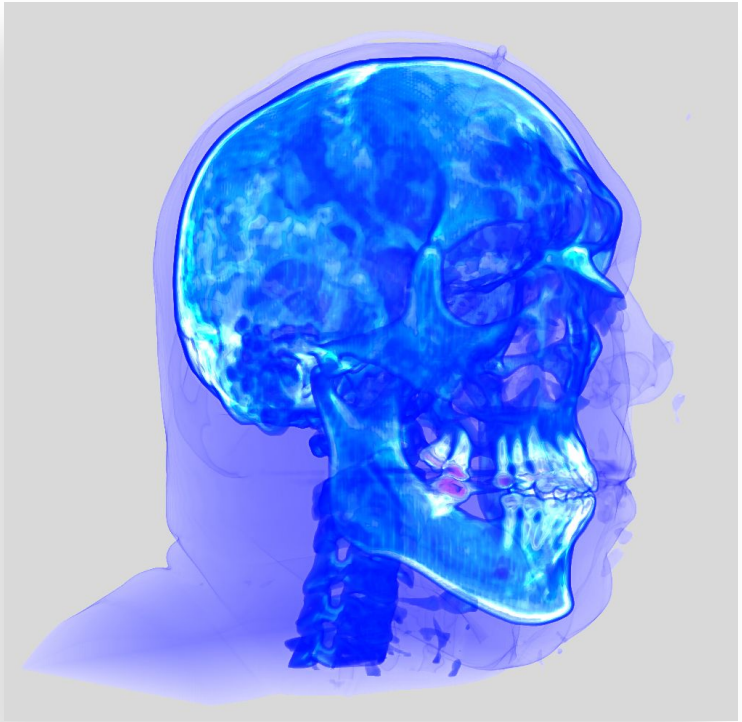


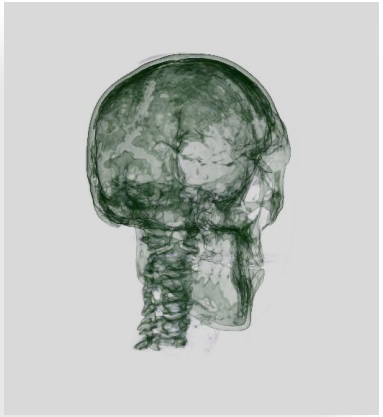
Getting Started…

- Program some form of Transfer Function
  - Gradient mapping range [0 - 1] to colours
- First trace into the volume in 2D
- Use density data as opacity
- Afterwards attempt to trace in 3D
- *Don't attempt to speed up code before it works*

More examples…







Lab session Task

- Designing a really basic transfer function...

FAQ

- Is the head volume meant to be a cube?
  - Yes. If you can account for one axis being smaller, then great! Otherwise you won’t be marked down.
- Can I use OpenGL to do the 3D viewport?
  - No and it wont work properly anyway.
  - Always use glVertex just to draw pixels
  - Please don't use GL_BLEND, do the blending yourself

Questions…?