DataVis Assignment 4

Quick Facts

- **Deadline: Tuesday, Jan 14, 11:59:59 a.m.**
- 3 tasks: get hands on ParaView, write a simple C++ app with VTK, lecture recap on data preparation
- You should solve the tasks in groups of three
- Be prepared to present your answers orally during the course
- Fill out **all** of the following pages
- **Upload all submission files to OPAL!**

Submission files for OPAL:

☐ Filled out submission form as PDF

☐ 2 Screenshots of gradient visualization for 4.1.3 as JPG, PNG or PDF

☐ ParaView state file (pvsm) file for 4.1.3

☐ Cpp file for 4.2

Team Members

Team Member 1: Last Name	Team Member 1: First Name	Team Member 1: Mat. No.

Team Member 2: Last Name	Team Member 2: First Name	Team Member 2: Mat. No.

Team Member 3: Last Name	Team Member 3: First Name	Team Member 3: Mat. No.

Introduction

In this assignment you will learn the basics of ParaView, a free end-user visualization tool, and VTK (**V**isualization **T**ool**K**it), an open source library. We provide a pre-compiled VTK 8.1 for Visual Studio 2013 and 2017 (VS12 and VS15). If you want to develop on Linux, please download the packages and build them on your own via the package manager and CMake. The computers in the PC labs have a pre-installed Visual Studio 2013. If you want to work there, please use the VS12 build. For those who want to build VTK on their own, this tutorial may help: http://programming.realworldrobotics.com/platforms-frameworks/vtk/compiling-vtk-using-cmake-and-visual-studio

The make folder contains solution files for Visual Studio 2013/2017 as well as CMake files intended to be used on GNU/Linux systems. Please note that we cannot provide support for Linux users. However, please let us know if you find mistakes or have improvement proposals.

Important note: Please upload **only** the source code files. No build files, no libs, no additional files. Please do not use any system-specific routines. Everything should compile under Visual Studio out of the box.

Task 4.1: Scientific Visualization with Paraview

Description: Get in touch with scientific visualization and use height field data.

Paraview: This VTK-based tool is available as a portable version, so you can use it in the PC lab without any installation. Please visit https://www.paraview.org/ for further details. Import the dataset Mount St. Helens (from /data/SainteHelens.dem).

Answer the following questions utilizing the **histogram** and **quartile** filter:

4.1.1 Describe how the elevation levels are distributed! Are their special values/outliers?

4.1.2 What is the median of the elevation values?

4.1.3 Visualize the gradients as arrow glyphs colored by their magnitude in 2D and in 3D. Use a meaningful color mapping. Upload two screenshots to OPAL, one for each visualization type, as well as the ParaView state file (pvsm). **hint**: make use of the filters “warp by scalar”, “tessellate“, “gradient [of unstructured data set]” and “glyph”

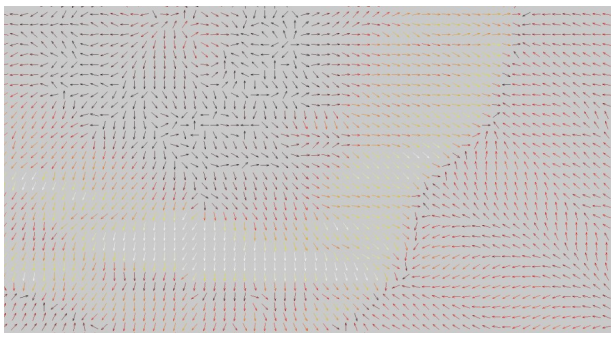


Figure 1-1. Example: Gradients of the elevation in 2D.

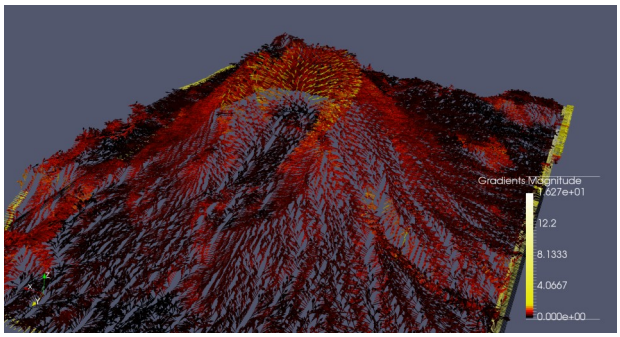


Figure 1-2. Example: Gradients of the elevation in 2D.

Task 4.2: Scientific Visualization with C++ and VTK

Get The visualization pipeline of VTK is more detailed than the one in ParaView. For every part of the pipeline, you will have a responsible class instance. First of all, there is a **data source** that can be processed by **filters**. Then, depending on the data type, you will need to choose an appropriate **mapper**. To finally show the visualization on the screen, you will need an **actor**, a **renderer**, and a **window**.

As a starting point, check the application "assignment4". There is only one cpp file that provides all the functionality. Get a big picture of how the components are connected together and how everything works. **Rewrite** the file to solve the following tasks:

- show the data in 3D
  - use the warp filter (set a scale factor of 2 to see the elevation better)
- generate contour lines
  - use the warped model as input
  - decide many contour lines shall be drawn (15 is a good starting point)
  - generate the contour values equidistantly in the range of the data
  - color the lines in the same solid color (do not use a look up table), there is a hint in the code
- define your own color transfer function for the warped model
  - you can utilize `vtkLookupTable` or `vtkColorTransferFunction` (both derive from `vtkScalarsToColors`)
  - connect the color transfer function to the mapper via `setLookupTable`()
- put it all together in one visualization
  - generate two mappers: one for the warped model alone and one for the contour model
  - finalize the given function `createRenderWindowFromMultipleMappers`
  - there is a hook for your code ("student begin"), instantiate an actor for every mapper and add it to the renderer
  - do the rendering and interaction as before
- use `vtkScalarBarActor` to show a legend
  - you can add the actor to the renderer from the outside of the method via
    ```
    someWindow->GetRenderers()->GetFirstRenderer()->
    AddActor2D( scalarBarActor );
    ```
- the final visualization should look like in Fig. 2-1
- upload the cpp only!
- additional hints:
  - Use comments and document your code!
  - Code quality matters!
  - It's okay to build upon existing examples, but: name your sources!

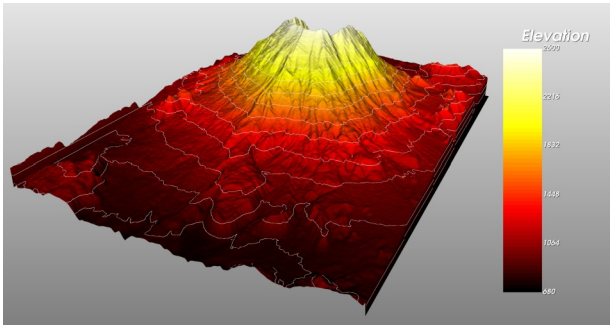
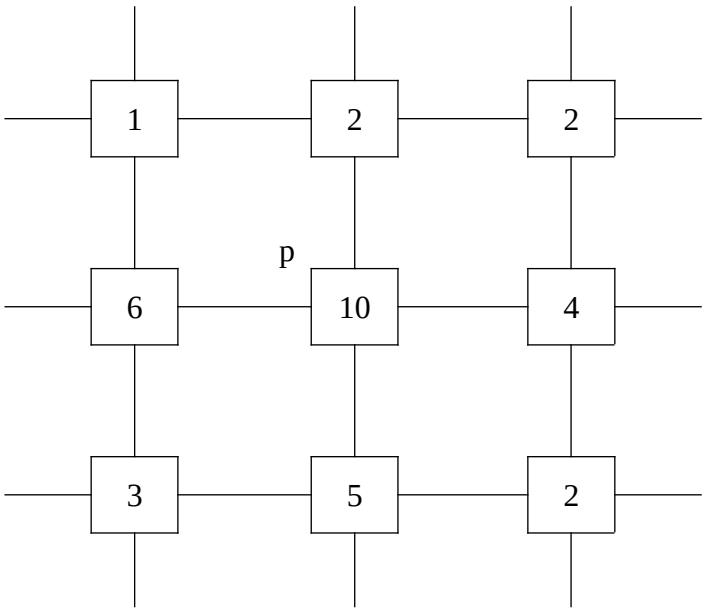


Figure 2-1. Elevation data as extruded scalar field with contour lines.

Task 4.3: Gradient computation

4.3.1 Given the following regular 2D grid with grid spacing *d=2* both in *x*- and *y*-direction:



Compute the (unnormalized) gradient at point p...

... (a) utilizing central differences

... (b) utilizing the Sobel operator

4.3.2 Describe the differences between both methods