

# Vistle Part II

## Hands-On Workshop for 3D Visualization

**Speakers:** Martin Aumüller (martin.aumueller@hls.de)  
Susanne Malheiros (susanne.malheiros@hls.de)



Centre of Excellence in Exascale CFD

# Overview

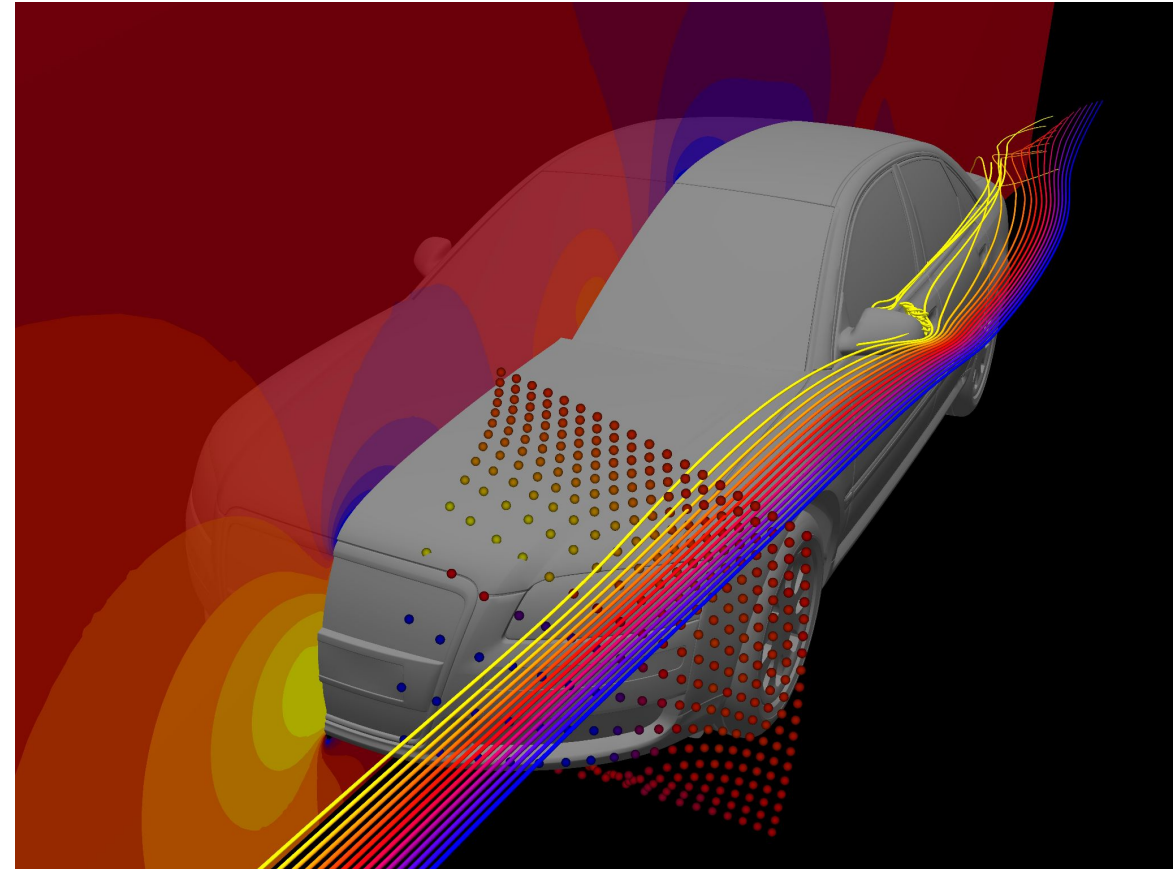
H L R **I** S

- Introducing Vistle
- **Live Demo:** Visualizing CFD Data with Vistle
- **Exercise:** Visualizing OpenFOAM Simulation Results

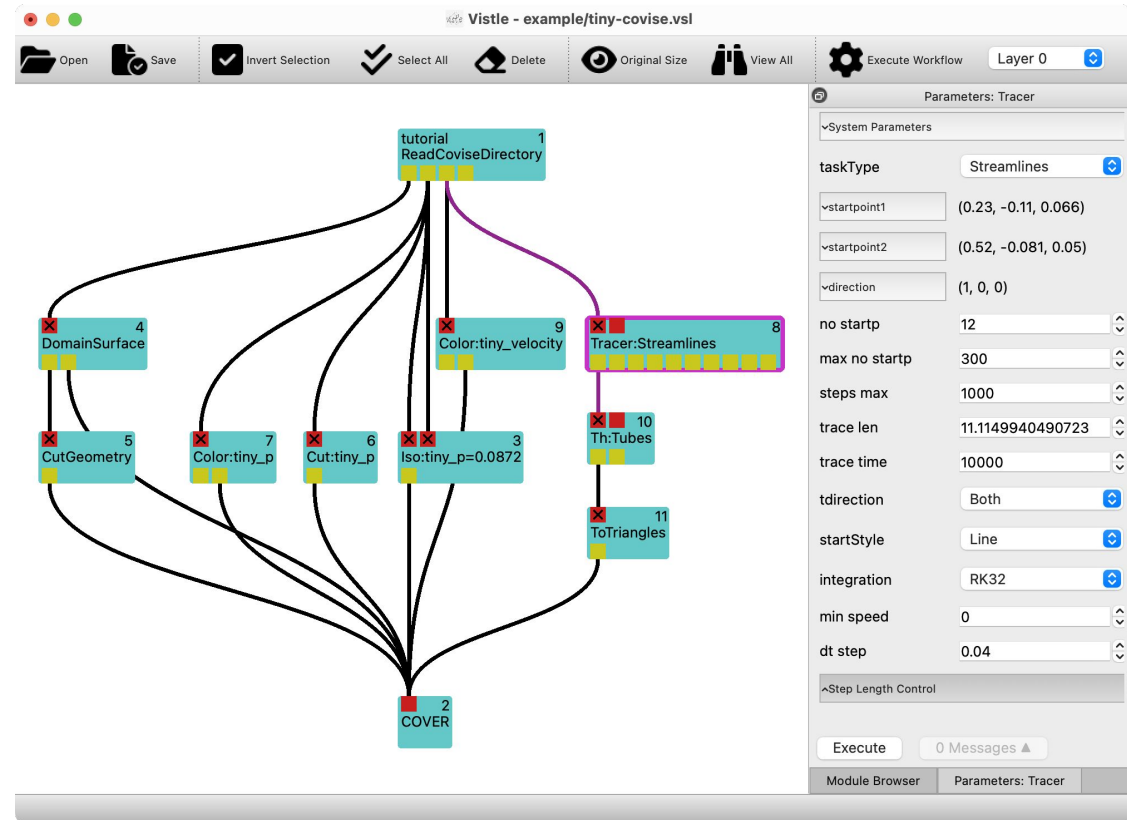
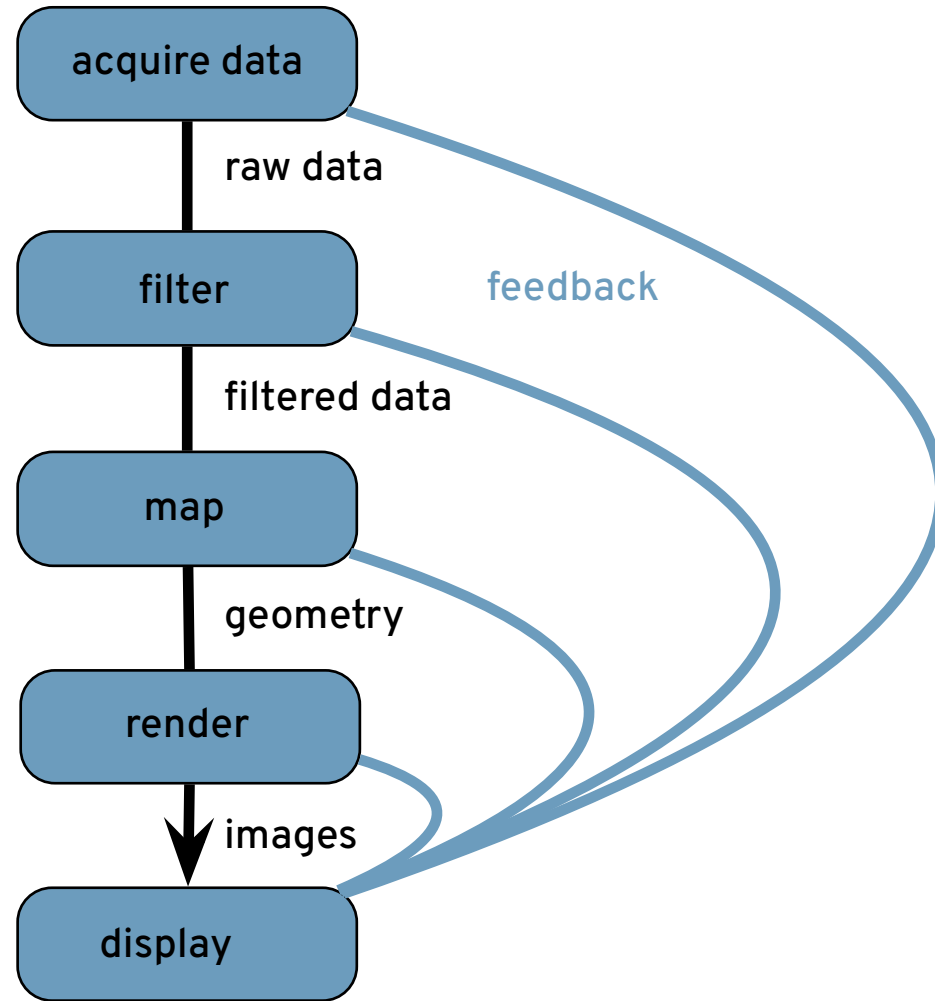
Introducing Vistle

# Scientific Visualization: Motivation

- to understand and contextualize abstract data
- to analyze simulated and measured data
- communication tool for interdisciplinary teams
- to present results to the public



# Visualization Pipeline

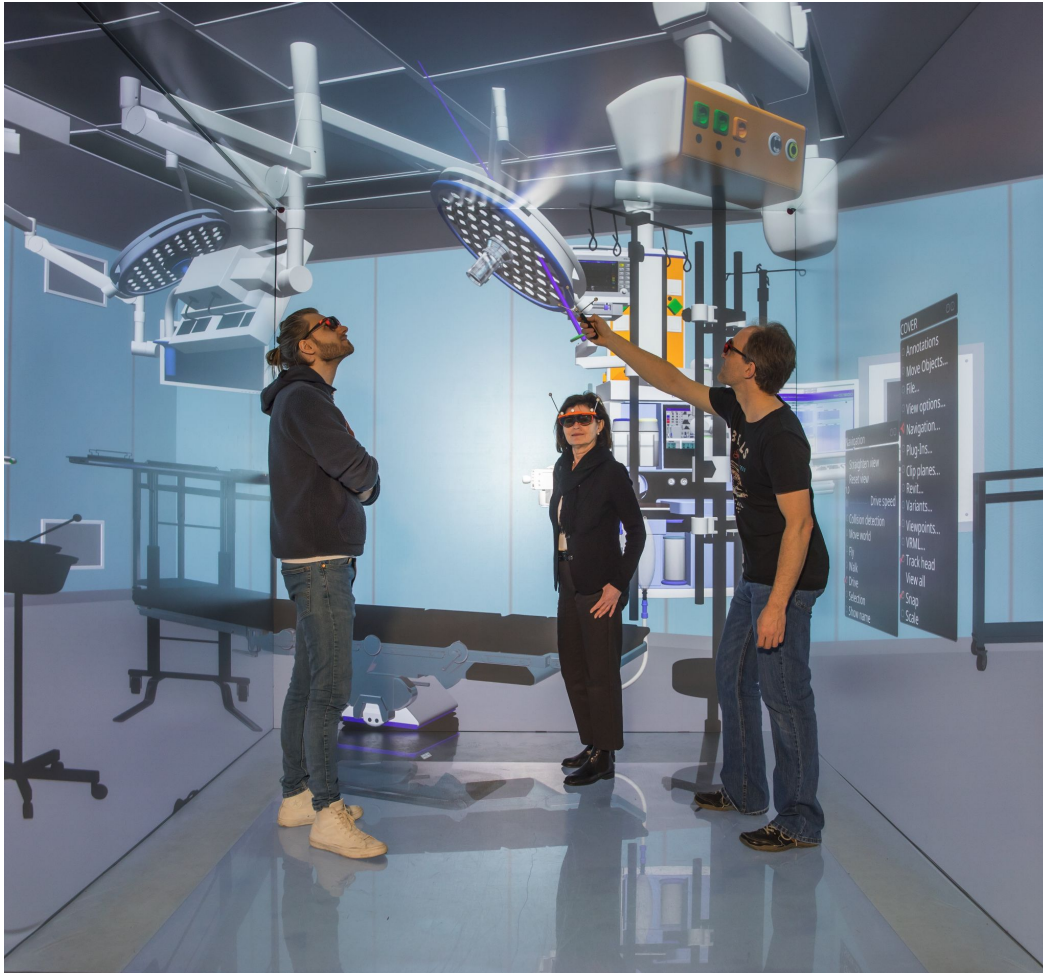


- nodes: modules
- arrows: from output to input ports



# COVER: VR and AR Visualizations

H L R I S

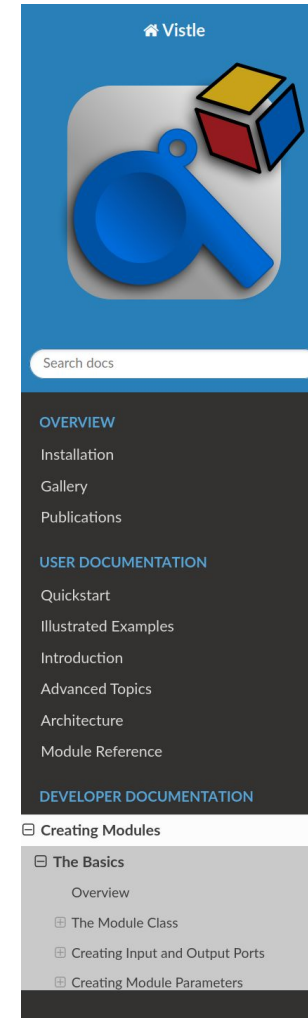


# Advanced Features

- writing custom modules
- running modules on the GPU
- In-situ visualization

## More Information:

- Documentation: <https://vistle.io/>
- Webinar recording: [“Vistle Part I: An Introduction to Immersive Visualizations of Large-Scale Scientific Data”](#)



🏠 / Module Programming / The Basics

## The Basics

Vistle is designed to be easily extensible through its modular structure. This tutorial will show you how to write your own Vistle module and how to incorporate it into the software, so you can use it right away.

### Overview

- [The Module Class](#)
- [Creating Input and Output Ports](#)
- [Creating Module Parameters](#)
- [Sample Module Code](#)
- [How to Add a Module to Vistle](#)

## The Module Class

The most common Vistle modules are the compute modules. They receive data from other modules through their input port(s), run computations on the data, and return the result(s) through their output port(s).

The following example shows the minimum code necessary to develop a Vistle module. It will create a module named **MyModule**.

```
#ifndef MYMODULE_H
#define MYMODULE_H

#include <vistle/module/module.h>

// this is a test module for the developer guide
class MyModule: public vistle::Module {
public:
    MyModule(const std::string &name, int moduleID, mpi::communicator comm);
    ~MyModule();

private:
    bool compute(const std::shared_ptr<vistle::BlockTask> &task) const override;
};

#endif // MYMODULE_H
```

**Live Demo: Visualizing CFD Data with Vistle**



## **Exercise: Visualizing OpenFOAM Simulation Results**

**Thank you for participating!**

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Sweden, Germany, Spain, Greece, and Denmark under grant agreement No 101093393.



Co-funded by  
the European Union



Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European High Performance Computing Joint Undertaking (JU) and Sweden, Germany, Spain, Greece, and Denmark. Neither the European Union nor the granting authority can be held responsible for them.



Centre of Excellence in Exascale CFD