What I did at Fraunhofer IGD

Seyedmorteza Mostajabodaveh

March 27, 2016

Abstract

In November 2014, I moved to Darmstadt to work in Fraunhofer IGD and do my master thesis. In this report, I will present the tasks which I have done beside my master thesis at Fraunhofer IGD as Hiwi. Additionally, my supervisor Andreas Dietrich (andi.dietrich@googlemail.com) which was working at Fraunhofer IGD before was directing me toward these tasks. You can contact him for more information.

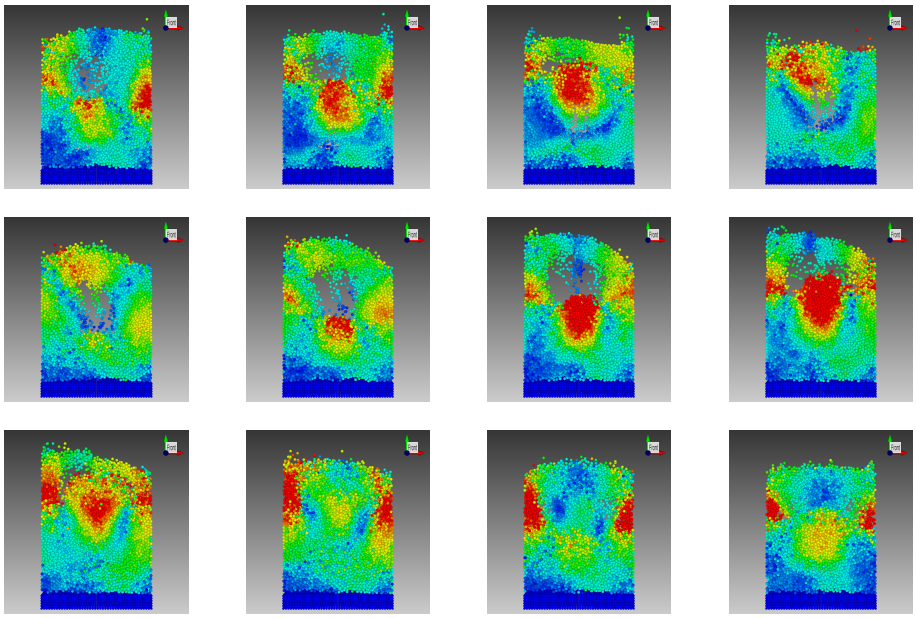


Figure 1: **Query particles from HBase database and their visualization** in VELaSSCo.

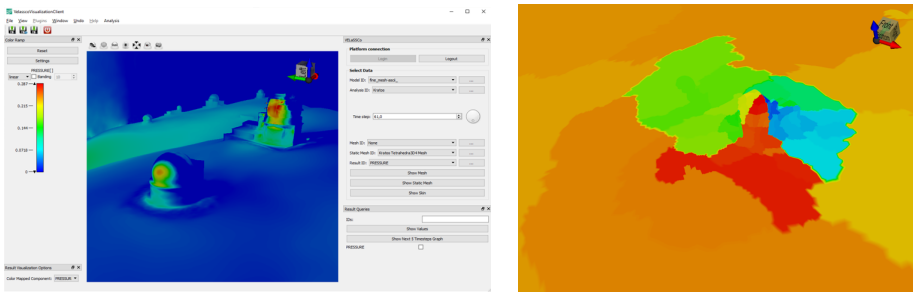


Figure 2: **Telescope boundary mesh** colormapped with pressure values in left figure and partition IDs in right figure. First the telescope boundary mesh is queried from the HBase database and then the results on the boundary is being queried from HBase database. The colormapping is done on the clinet side.

VELaSSCo project. "Visualization for Extremely Large Scale Scientific Computing" known as VELaSSCo is an EC funded project which deals with end-user visualization of Big Data. The idea behind this project is to store the datasets on a distributed database like HBase (Hadoop) or EDM. The user can

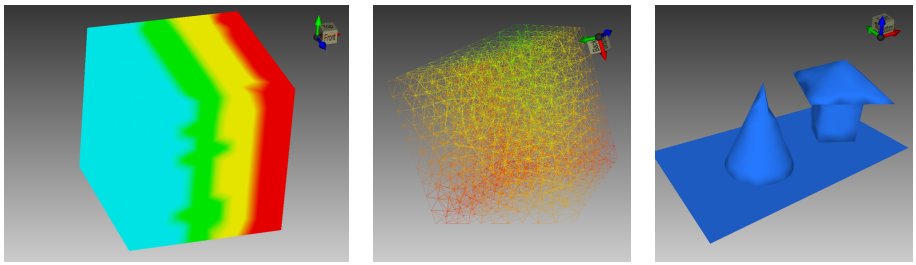


Figure 3: **Querying simulation mesh** from HBase database. The results on the surface and inside simulation mesh are queried separaretly and visualized using colormapping.

query a specific part of data in the database or a specific operation on the data in the database. The user's query is processed on the HPC and the result is sent in a GPU-friendly format for the user to visualize e.g. Figure 2 shows the boundary mesh of telescope dataset with colormapped pressure values. The user queries for the mesh boundary. The HPC computes the boundary of the mesh and send it for the user. The user sends another query to request for pressure values on the vertices of the boundary mesh. Then, the client visualizes the boundary with pressure values colormapped on its surface.

In this project, I mainly focused on implementing two direct result queries used for querying results on a specific vertex indices, and querying mesh draw data for models in the database. Mesh drawing data query result needed to be returned in a GPU friendly format, so the received data on the client side can be copied directly into an OpenGL buffer directly. I implemented these two queries using Thrift C++ API. On the client side, I implemented a plug-in to support VELaSSCo Plugin for indoor application called Rapid Prototyping Environment (RPE). This plugin is able to:

- query for available models, their analysis, results, meshes, and static meshes.
- Query for computing boundary mesh of a specific dataset (Figure 2).
- Query for Finite Element Meshes (FEM) / Discrete Element Meshes (DEM) and render them (Figure 3).
- Query for results on a specific node (or evaluation in all time steps).
- Query for different timesteps for DEM cases and create an animation.

Ray-tracing higher-order primitives. This task starts with a simple example which was rendering of a pipecube using pipes which are made of two spheres and one cylinder. The GPU version was implemented by Andreas Dietrich and I implemented the CPU version using Intel Embree. The CPU

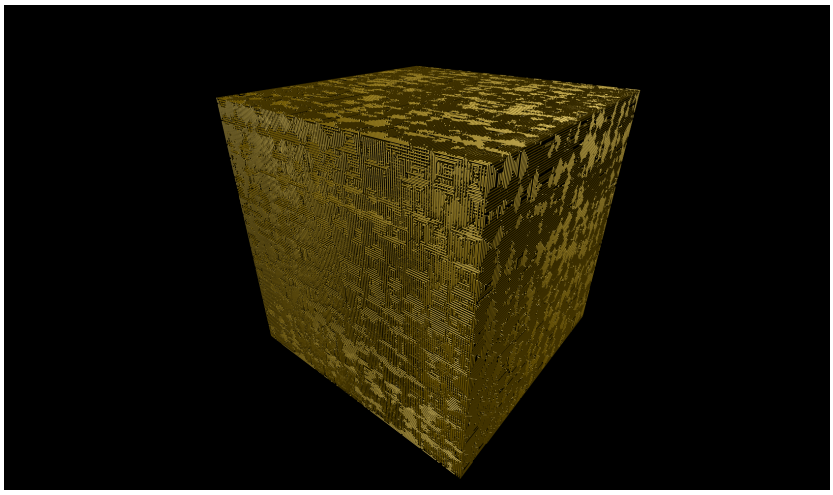


Figure 4: **Pipecube** made of more than 15.6 million pipes, ray-traced on Full HD on 17 fps using Intel Embree on a Intel Core i7-4960X.

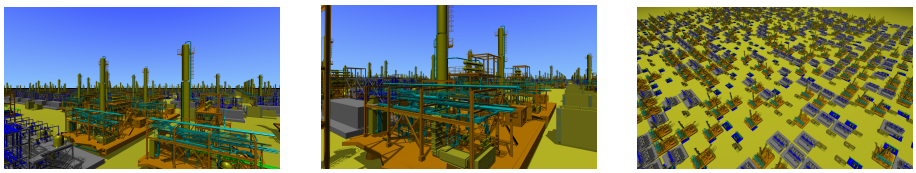


Figure 5: **Factory ray-tracing with Embree**.

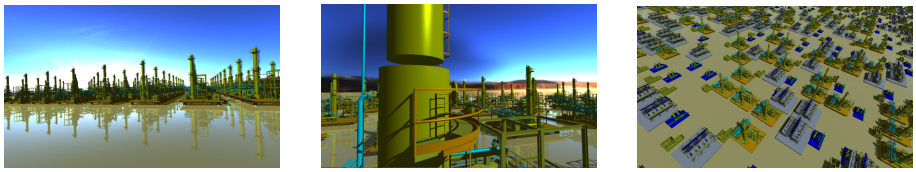


Figure 6: **Factory ray-tracing with Optix**.

version was able to handle very big pipecubes (Figure 7). This was an starting idea for implementing a version which uses the AVEIVA's RMV files to produce the factory models with higher order primtives (Sphere, Box, ...) and then ray-trace it on CPU or GPU. I implemented a factory viewer which ray-trace a group of factories on the CPU and GPU using Intel Embree and NVIDIA Optix (Figure XX, YY, ZZ).

PCB board software rasterizer. Implementing a software rasterizer for rendering triangulated PCB boards (Figure 8).

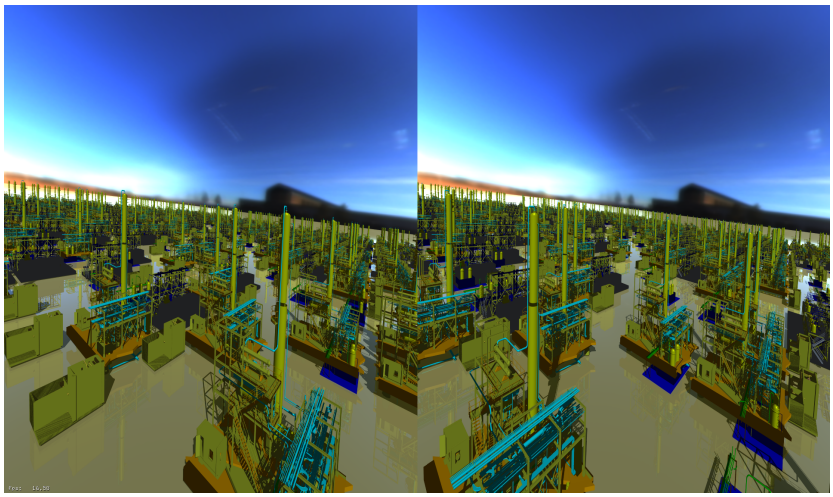


Figure 7: **Factory ray-tracing (Stereoscopic view)**.

Virtual Reality. Doing some bug fixing for Oculus rift plug-in an extending the plug-in for specific projects, to support a player camera inside the scene. Additionally, adding some features to interact with objects using Leap Motion. Figure 9 shows two screenshots of the projects which I did in Fraunhofer IGD.

Other Tasks.

- Implementing cross-section for unstructured simulation meshes (Figure 10).
- Adding very large screenshot capturing to RPE (Figure 11).
- Evaluating B-Spline on the GPU using tessellation shader (Figure 12).

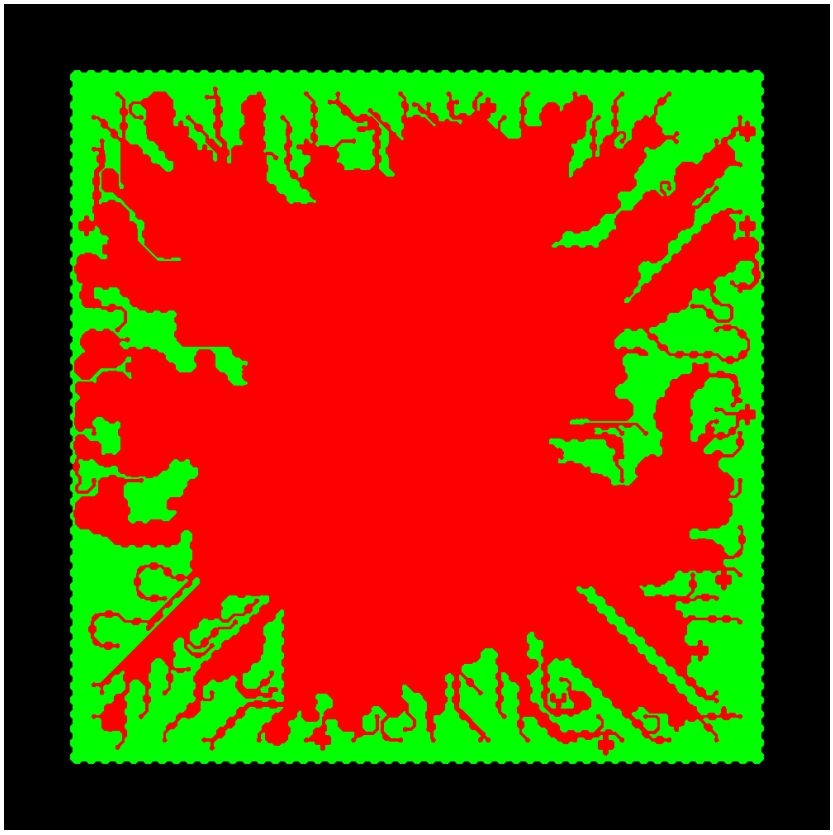
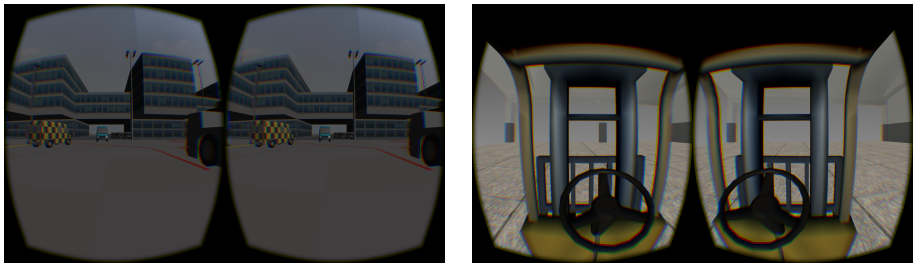


Figure 8: **Software PCB rasterization** done 107 fps on Core i7-4960X.



(a) Airport Scene (b) Forklift Driving

Figure 9: **Virtual Reality** Projects.

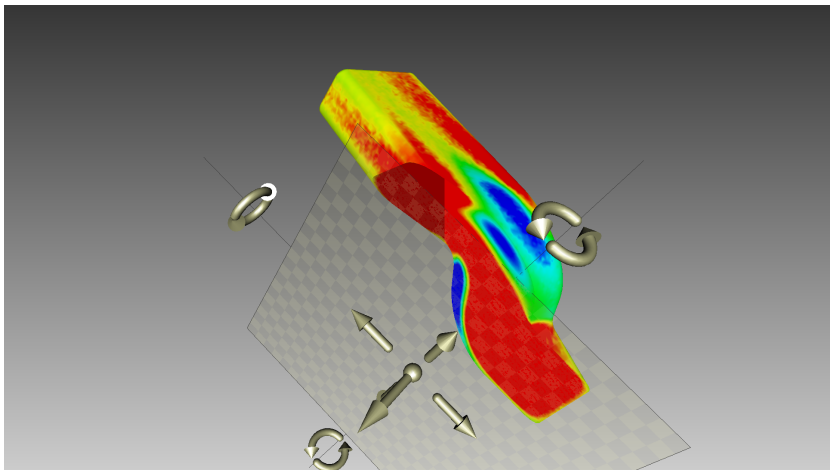


Figure 10: **Cross section** of wind tunnel dataset.

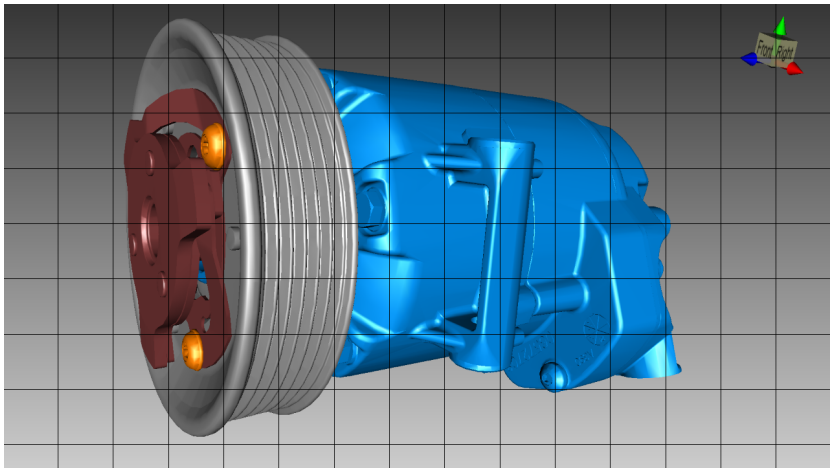


Figure 11: **Extremely high resolution screenshot capturing**.

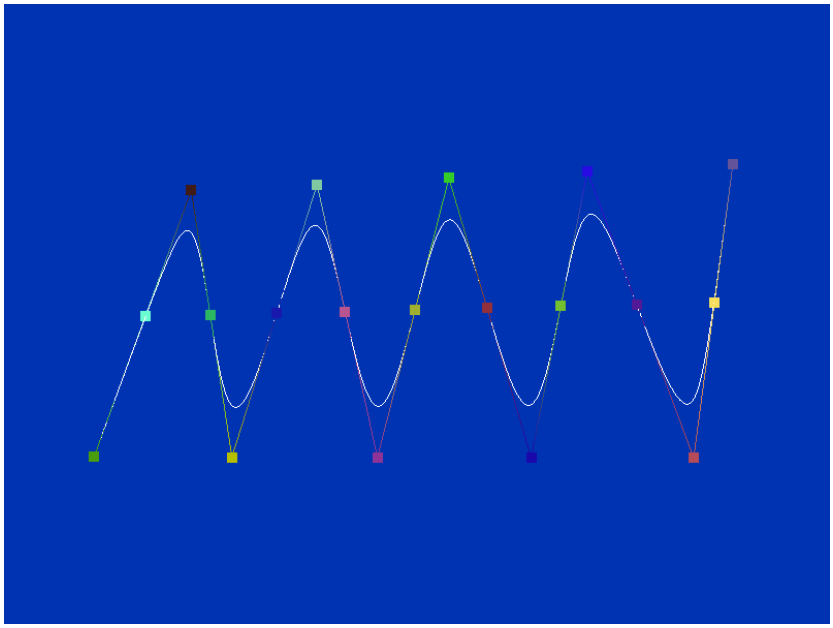


Figure 12: **B-Spline GPU Tessellation**.