INTERACTIVE VISUAL DECISION SUPPORT TECHNIQUES

Deliverable D5.1.2

Circulation:	PU: Public
Lead partner:	Fraunhofer
Contributing partners:	SINTEF
Authors:	Frank Michel, Tobias Franke, Thomas Gierlinger (Fraunhofer), André Brodtkorb (SINTEF)
Quality Controllers:	André Brodtkorb (SINTEF)
Version:	1.0
Date:	06.11.2014

Consisting of

SINTEF	STIFTELSEN SINTEF, Department of Applied Mathematics, Oslo, Norway
Fraunhofer	Fraunhofer Institute for Computer Graphics Research, Darmstadt, Germany
CNR-IMATI-GE	Institute for Applied Mathematics and Information Technologies of the National Research Council (CNR-IMATI), Genova, Italy
MOSS	M.O.S.S. Computer Grafik Systeme GmbH (MOSS), Munich, Germany
HRW	HR Wallingford Ltd (HRW), Wallingford, UK
FOMI	Hungarian National Mapping and Cadastral Agency (FOMI), Institute of Geodesy, Cartography and Remote Sensing, Budapest, Hungary
UCL	University College London (UCL), Research centre for Photogrammetry, 3D Imaging and Metrology, London, UK
TUDelft	Delft University of Technology (TUDelft), Department of Earth and Climate Sciences & Man-Machine Interaction Group, Delft, The Netherlands
IGN	Institut National de l'Information Géographique et Forestière (IGN), Paris, France
UBO	Université de Bretagne Occidentale (UBO), European Institute for Marine Studies, Brest, France
Ifremer	L'Institut Francais de Recherche pour l´Exploitation de la Mer (Ifremer), Brest, France
Liguria	Regione Liguria, Genova, Italy

DOCUMENT HISTORY

Version[1]	Issue Date	Stage	Content and Changes
0.1	21.10.2014	Draft	Initial draft with deliverable structure
0.2	28.10.2014	Draft	Work done initial content added
0.3	03.11.2014	Draft	Work done completed
0.4	05.11.2014	QC	Ready for QC
1.0	06.11.2014	Final	Final version ready for submission

[1] Integers correspond to submitted versions

EXECUTIVE SUMMARY

This report presents the progress made in *WP5 Interactive Visual Decision Support Techniques* during the second year of IQmulus. The work conducted in WP5 during the second year as well as an outlook on the planned work in year three is reported on task level.

In the second project year the activities of WP5 were focused on enhancing the visualization modules already developed in the first project year and extending the IQmulus visualization framework. The developments were driven by the IQmulus showcases, defined during the second project year.

TABLE OF CONTENTS

1 VISION OF WP5

WP5 aims to develop visualization techniques for the *Interactive Visual Decision Support* in the context of geographical information systems. The WP is about visualization applications tailored for geo-experts and decision makers, as opposed to data administration or data processing experts, and will utilize both *fat* and *thin* clients. Fat clients will use the advantages of desktop computers with access to powerful GPU resources through modern OpenGL technology, and thin clients will use web technology to allow the creation of mobile applications. In both application scenarios (desktop / fat and mobile / thin), the targeted visualization techniques will be able to handle large heterogeneous geo-spatial data sets in an interactive manner for geo-experts and decision makers.

The following gives a brief summary of the objectives and technological components of WP5 as presented in D 5.1.1. Please refer to this deliverable for more details.

1.1 OBJECTIVES AND EXPECTED RESULTS OF WP5

The core objective of WP5 is to develop better support for visual decision making and interactive visual communication on large heterogeneous geo-spatial and temporal data sets for expert users, consultants preparing decisions, and decision makers. This will include:

- Developing multi-resolution methods for visualization of previously uncorrelated data sets;
- Leveraging modern GPU-based features on local graphics workstations and web-based clients;
- Supporting new and existing interaction techniques for the decision making process by GPU-based methods;
- Developing a 3D-web-based visualization architecture for the deployment of dedicated visualization applications to (expert and lay) web users;
- Establishing new or extending existing visualization driven data formats.

Technical evaluation of feasibility for the various envisaged approaches is part of WP5, in order to transfer GPU-based visualization techniques to the geospatial domain.

1.2 TECHNOLOGICAL COMPONENTS OF WP5

WP5 technology aims to improve the support for visual decision making on large heterogeneous geo-spatial and temporal data sets. The focus is on utilizing modern GPU- and 3D-web-based technologies to achieve this goal. Therefore, the visualization components are not foreseen to act as general purpose visualisation clients. Instead the developments concentrate on algorithms and interaction metaphors that benefit from the utilization of modern GPU features. Other operations may be performed in 3rd party applications.

The visualization development can be decomposed based on functionality and technical approach, which results in the following list of main components:

- Composite multi-resolution visualization: Enables interactive visualization of previously uncorrelated data by developing multi-resolution methods for composite visualization.

- Deferred visualization: Increases efficiency of the visualization based on deferred mapping, i.e., utilizing image space information for both image generation and user interaction.
- 3D-web-based visualization: Utilizes modern web-based 3D technology for interactive visualization.
- HUB: A web service portal that decides on the best representation of data and visualization technique given the capabilities of an output device.

Figure 1 shows a sketch of the architecture, outlining interactions between internal components as well as other parts of IQmulus.

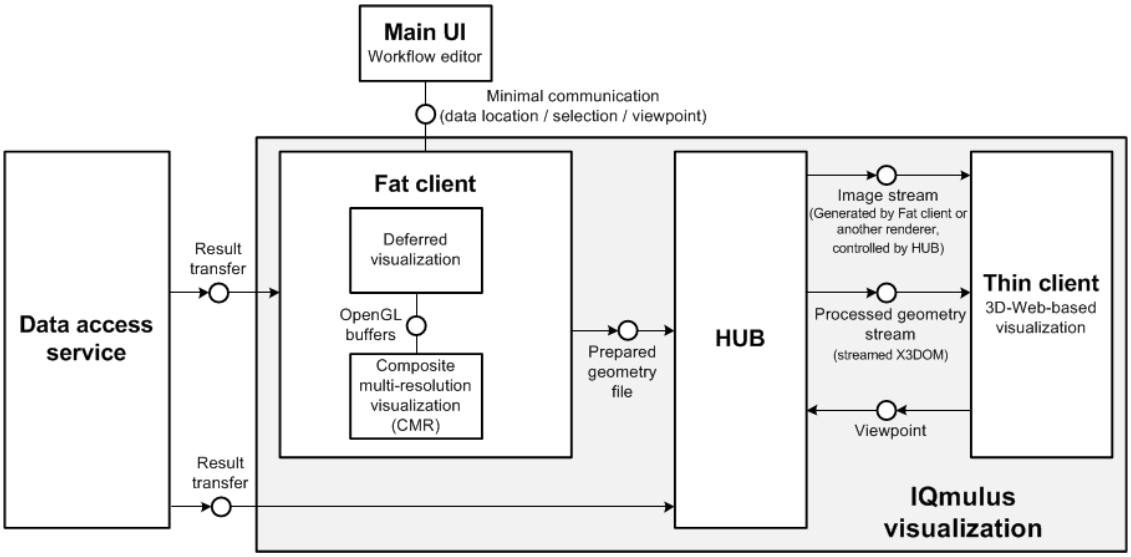

FIGURE 1: ARCHITECTURE OF THE IQMULUS VISUALIZATION COMPONENT.

2 PROGRESS REPORT ON WP5 TASKS

The following subsections describe the performed activities for the different tasks in WP5.

The focus of WP5 during the second year has been to further develop different visualization methods in the respective tasks. Besides the work on new visualizations, a WP5 internal code camp was held in Darmstadt, focusing on component integration and preparation of the year 1 review. All partners in WP5 also attended the code camp in Darmstadt in March 2014, in which further work on component integration was performed.

The developments are based on the requirements resulting from the development of the IQmulus showcases, which all WP5 partners contributed to. All WP5 partners also participated in the consortium meeting in Delft and further specified the visualization parts of the use case scenarios / showcases.

2.1 TASK 5.1: COMPOSITE VISUALIZATION METHODS

The goal of the composite visualization task is to develop new methods for interactive visualization of combinations of data sets like vector-based cartographic information, raster-based terrain data and vector- and tensor-based simulation data. Multi-resolution methods must be developed to reach the goal of interactivity for large data sets.

2.1.1 Work Performed

The focus of the second project year has been to further develop the capabilities of the interactive composite and multi-resolution visualization (called the CMR framework). The work has been driven by the needs of the defined showcases (super user stories), and we have developed three different visualization prototypes:

- LR splines together with original point cloud,
- Time-varying rain-gauge data, and
- Multi-resolution PLY rendering.

LR splines together with original point data

The marine scenario produces compact LR splines from huge point-clouds, which need to be visualized and interactively inspected by a user. LR splines are high-order spline representations of surfaces, which supports local refinement around sharp features such as the crests of the sand dunes in Figure 2. GPUs support only polygonal meshes, and we therefore have to convert the LR spline representation to a triangulation. The naive way of doing this is to create a polygonal geometry by regular sampling of the spline surface. Unfortunately, this will either give a coarse tessellation of the surface, or a too large a number of polygons for high levels of interactivity.

A more efficient way of visualizing an LR spline surface, is to perform the tessellation directly on the GPU itself. When opening an LR spline surface, the surface is first converted to an irregular set of Bézier patches on the CPU, a process which takes very little time. The Bézier patches are then transferred to the GPU, together with acceleration data structures. These patches are converted on-the-fly to triangles using the tessellation control and evaluation shaders, and shaded with per pixel shading in the fragment shader. This means that the triangulation is generated instantly for every single frame, enabling the use of interactive view-dependent level

of detail. The result is a per-pixel-accurate visualization of the LR spline surface at more than interactive framerates.

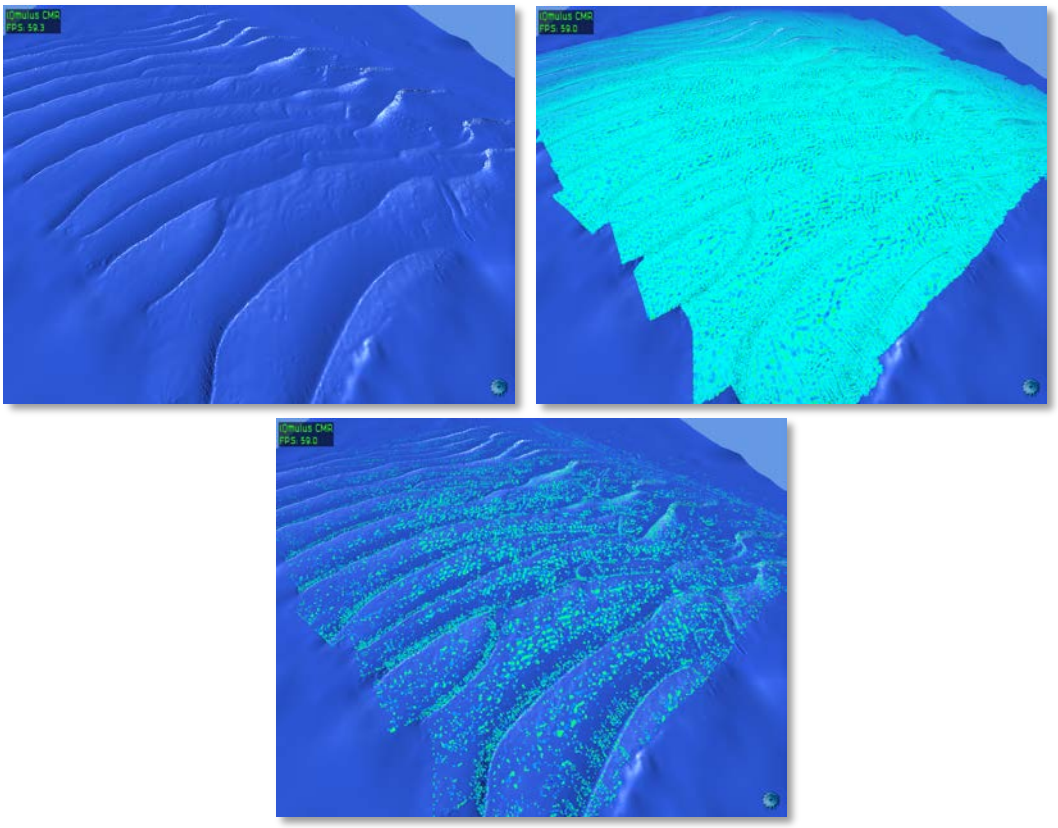

FIGURE 2: VISUALIZATION OF LR SPLINES TOGETHER WITH THE ORIGINAL DATA SET. THE FIRST IMAGE SHOWS THE LR SPLINE SURFACE, THE SECOND SHOWS THE INPUT DATA POINTS (WITH COLOR CODED DISTANCE FROM SURFACE), AND THE THIRD SHOWS ONLY POINTS WITH HIGH ERRORS WHEN COMPARED TO THE COMPUTED SURFACE.

The LR spline surface has been computed from a point cloud, and a user will often want to compare the results with the original data set to assure that the quality is sufficient. Figure 5 shows an example of this, in which the original point cloud is visualized simultaneously with the surface. Unfortunately, the point cloud contains such a high density of points that it becomes difficult to appreciate where the largest errors are. The user is therefore presented with an on-screen user interface to manipulate the point cloud, by changing transparency, colour mapping, and other parameters interactively. He or she can further select to show only the points with the largest distance from the surface, as shown in the third figure, and individual points can be inspected to see their specific distance.

Time-varying rain-gauge data

Time varying data can be challenging to understand and navigate, as the temporal relationship is often lost. It is therefore important to have a good user interface which enables not only spatial, but also temporal navigation. We have developed internal datastructures and a user interface for this. For datasets with multiple timesteps, the user is presented with an on-screen interface to navigate in time using a slider. Figure 3 shows an example of this for rainfall in the Ligura region. By changing the slider, the visualized dataset is interactively updated to represent the newly selected time, and navigation both forward and backward in time is supported. Individual rain gauges can be inspected by clicking on them to reveal their actual value.

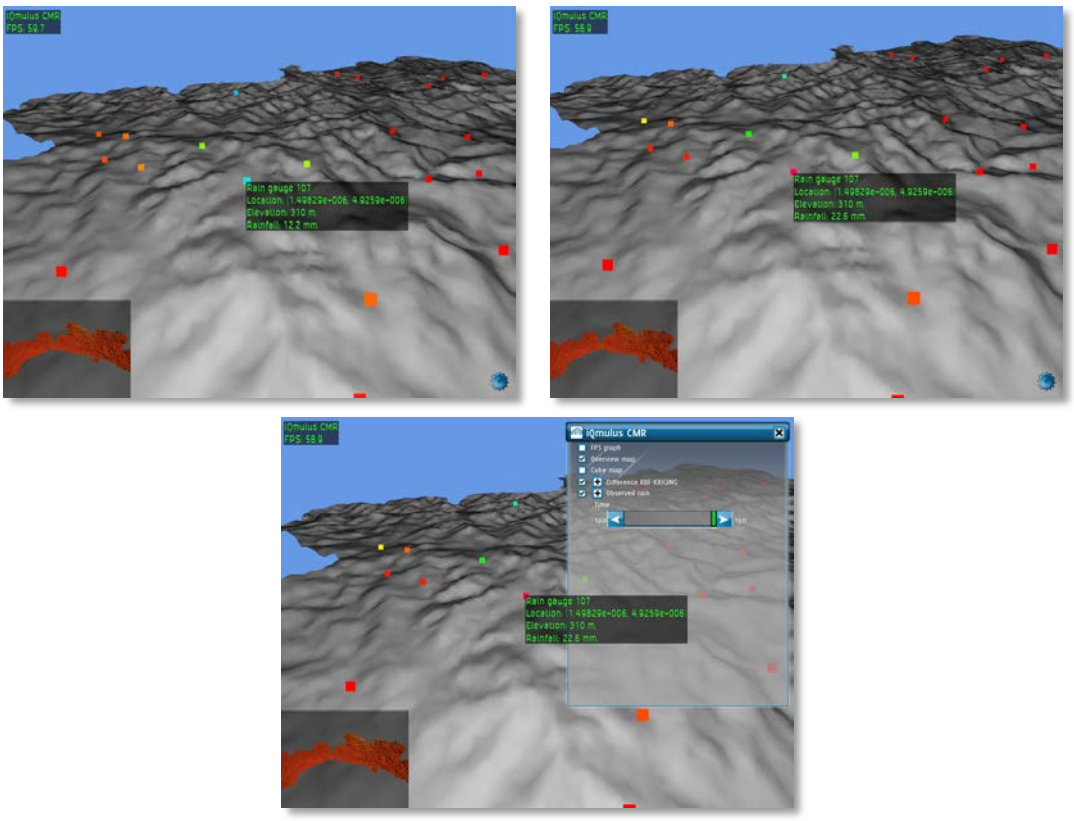

FIGURE 3: VISUALIZATION OF TIME-VARYING DATA-SETS. THE SLIDER IN THE THIRD IMAGE IS USED TO NAVIGATE TEMPORALLY THROUGH THE DATASET. NOTICE THAT THE SCREENSHOTS SHOW AN OVERLAY MAP OF THE WHOLE SCENE IN THE LOWER LEFT-HAND CORNER, AND THAT THE VALUE OF THE SELECTED RAIN GAUGE IS DISPLAYED WHEN CLICKING THE POINT.

Multi-resolution PLY rendering

Hierarchical multi-resolution triangulations are important for large datasets that become too large for interactive visualization, and we have developed a prototype for visualizing such datasets. The hierarchical triangulation can be represented as a tree structure, in which the root node contains the lowest resolution triangulation, covering the whole domain. Going down into the tree each new child node will cover a part of its parent triangulation, but with an increased fidelity.

The dataset is first read and bounding boxes computed for each node in the tree. When rendering, a fast half-space test with the six planes of the view frustum will discard nodes which are fully outside of view. A criteria based on distance from camera determines which of the remaining nodes to render, and the highest resolution nodes are rendered first. Since the parent node overlaps its children, the pixels rendered in this process are masked to prevent further updates. This process is continued until all nodes have been processed, and the hierarchical triangulation has been rendered. It should be noted that this process may give small artefacts of lower-resolution triangulations along the silhouette of the nodes, but this will in practice be small and on the boundary of a higher resolution node.

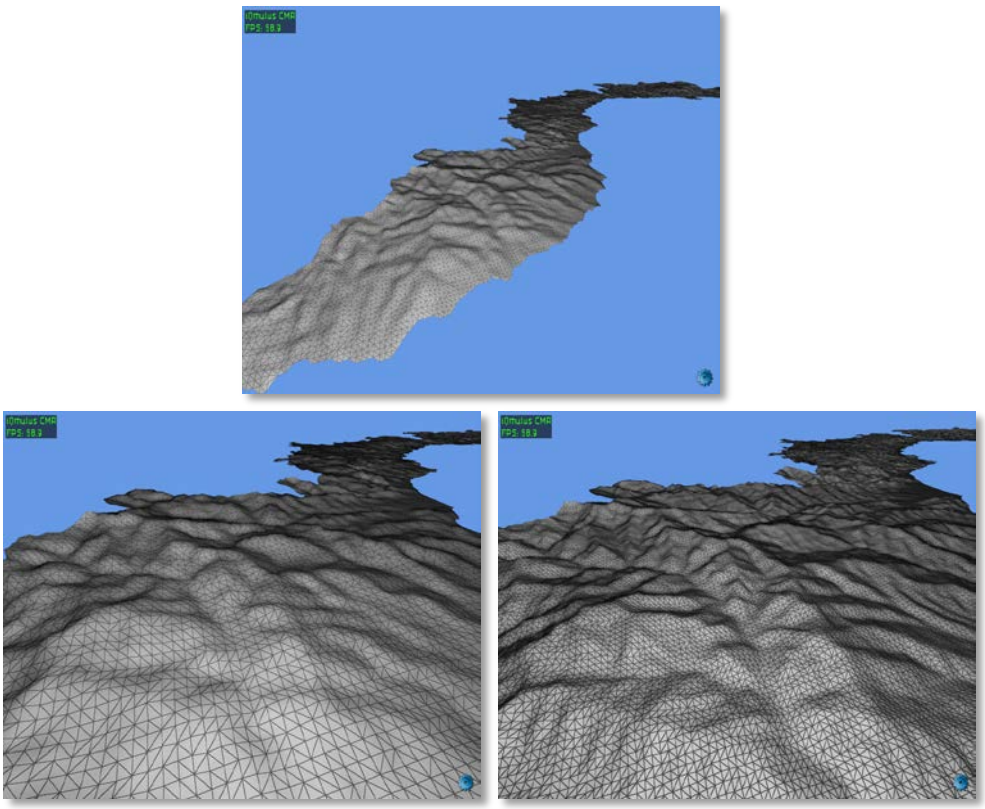

FIGURE 4: DYNAMIC LEVEL OF DETAIL WHEN VIEWING THE LIGURIA DATASET. THE FIRST IMAGE SHOWS ZOOMED OUT, THE SECOND SHOWS ZOOMED IN, AND THE THIRD IMAGE SHOWS ZOOMED IN WITH INCREASED LEVEL OF DETAIL. NOTICE THE DIFFERENCE IN QUALITY OF THE TWO LAST IMAGES.

2.1.2 Deviation of Work and Corrective Actions

There are no deviations from the work plan.

2.2 TASK 5.2: VISUALIZATION TECHNIQUES UTILIZING NEWEST GPU FEATURES

In this task we extend our existing visualization frameworks and address the following issues:

- How can modern GPU-techniques visualize geo-information data sets more efficiently?
- What is the possible impact of these techniques on the quality of geo-information visualization?
- How can image space based GPU-techniques (deferred mapping) be combined with user interaction techniques to support intuitive and interactive geo-information analysis?
- Can image space-based GPU-techniques be a basis for new deferred fetching algorithms to fetch secondary data in an optimal way?

2.2.1 Work Performed

In year 2 different base technologies were integrated into the IQmulus visualization framework which are the basis for current and upcoming developments in the second half of the project. Furthermore we developed new and enhanced existing modules for the visualization utilizing the deferred rendering pipeline implemented in year one of the project.

The framework now fully supports all current OpenGL shaders (vertex, fragment, geometry, tesselation and compute shaders) which are used in current and upcoming developments. In addition, the Fat Client was extended with different editors for shaders and uniforms to support the development of new shaders within the framework.

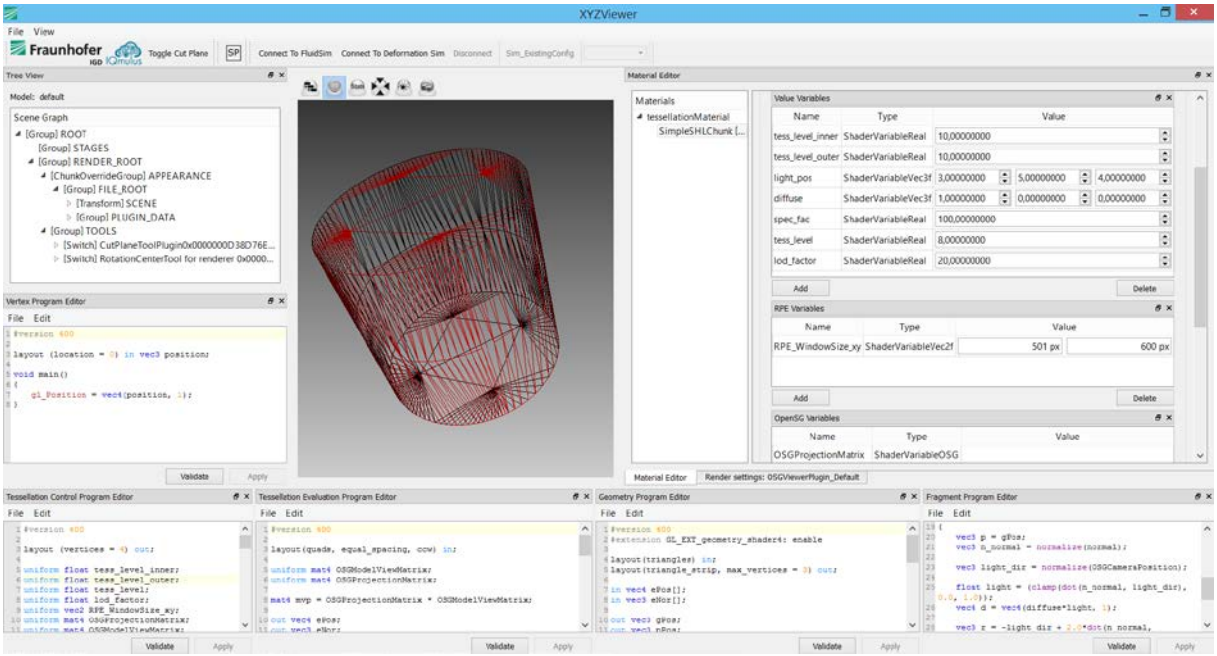

FIGURE 5: FAT CLIENT IN DEVELOPMENT MODE: DIFFERENT SHADER EDITORS AND THE RESULTING RENDERING IS SHOWN.

Figure 5. shows the Fat Client in development mode: The scenegraph is shown in the top-left sub-window. The top-right window shows uniform variables which are used by the shaders of the currently selected material. The bottom row of windows allows for editing the shader programs of different GPU pipeline stages. The center sub-window shows a rendering of the current scene. Two examples of GPU shaders are shown in Figure 6: The left image shows the

view-dependent tessellation of a terrain. The terrain is constructed from rectangular patches which are converted into triangles on the GPU using a tessellation shader. The quality of tessellation is adapted to the pixel-coverage in screen-space (patches of the terrain covering a larger area of the final image are subdivided into more triangles than those covering only a small area). The right image shows an example of geometry generation on the GPU. Shown is a wireframe rendering of a cube. The normals of the cube faces are visualized as small red or blue arrows. These arrows are generated directly on the GPU using a geometry shader.

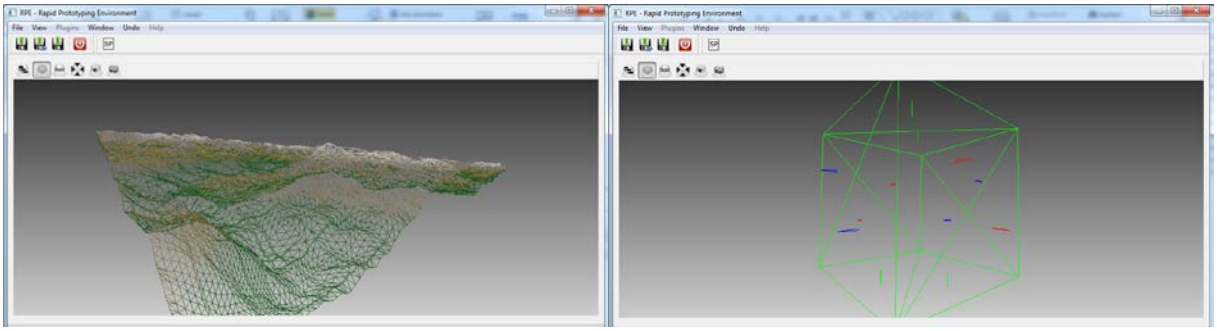

FIGURE 6: VIEW-DEPENDENT TESSELATION OF A TERRAIN USING TESSELLATION SHADERS (LEFT) AND AN EXAMPLE OF A GEOMETRY SHADER WHICH GENERATES THE GEOMETRY FOR VISUALIZING SURFACE NORMALS (RED AND BLUE ARROW GLYPHS, RIGHT) INTEGRATED INTO THE IQMULUS VISUALIZATION FRAMEWORK

To enhance the quality of the visualization of point clouds we implemented a splatting algorithm based on screen-space-triangulation[2] of *splats* using Geometry Shaders to efficiently reconstruct surfaces from the points. The visual results, which can be achieved by this technique, are of high quality. However, due to the required GPU resources performance is less than expected but still interactive (about 25 frames per second on a NVIDIA GTX 480 for the Stanford dragon point cloud shown in Figure 7).

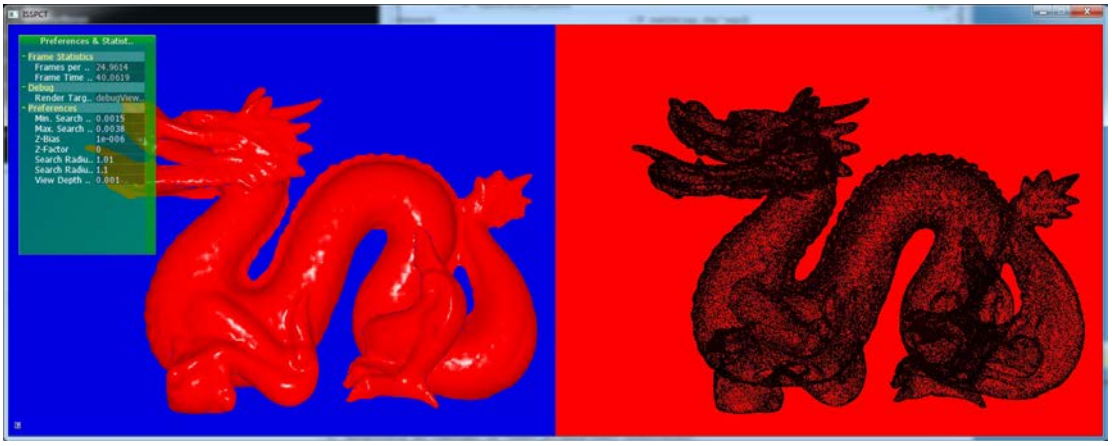

FIGURE 7: SCREEN SPACE TRIANGULATION USING GEOMETRY SHADER ON THE LOCAL GPU. THE RIGHT IMAGE SHOWS THE ORIGINAL POINT CLOUD, THE LEFT IMAGE SHOWS THE RECONSTRUCTED SURFACE USING SCREEN SPACE TRINAGULATED SPLATS.

Using the results of the screen-space-triangulation technique, we developed an advanced point cloud renderer based on the AutoSplats[3] approach. The point cloud rendering is performed locally on the GPU and requires only the point coordinates and optionally the colour. Splat and

[2] Preiner R., Wimmer M.,Interactive Screen-Space Triangulation for High-Quality Rendering of Point, April 2012

[3] Reinhold Preiner, Stefan Jeschke, Michael WimmerAuto Splats: Dynamic Point Cloud Visualization on the GPUIn Proceedings of Eurographics Symposium on Parallel Graphics and Visualization, May 2012

normal generation is performed on the GPU using a parallel nearest neighbour search, which utilizes shaders as well as the Transform Feedback feature of current graphics boards. The main advantage of this method is that it avoids any kind of pre-calculation / preparation of the input point cloud. Therefore, the approach can be used to visualize dynamic point cloud streams efficiently.

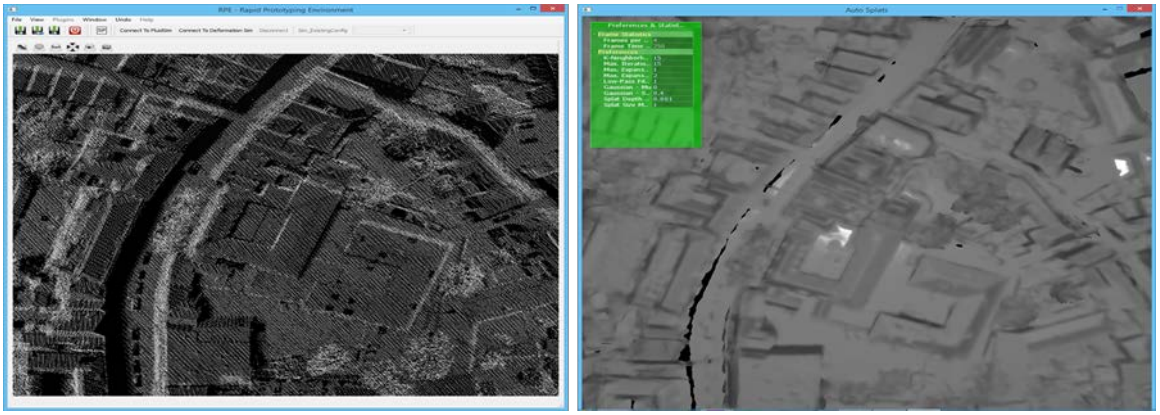

FIGURE 8: ADVANCED POINT CLOUD RENDERING USING AUTOSPLATS (RIGHT) APPLIED TO AN IQMULUS TEST DATASET (LEFT)

Figure 8 shows the effect of the advanced point-cloud-rendering on one of our test data sets. On the left side, the original data is shown, on the right the data is visualized using our point-renderer. As can be seen, the surface of the point cloud is reconstructed and reveals more structure and detail than the original data. Depending on the input point density, the algorithm may not be able to reconstruct a closed surface. This issue will be further investigated in the next period.

2.2.2 Deviation of Work and Corrective Actions

There are no deviations from the work plan.

2.3 TASK 5.3: GPU-SUPPORTED INTERACTION FOR DECISION MAKING

The challenge in Task 5.3 is to support interaction techniques (new and existing ones) with GPU-based methods to generate new levels of interactivity and improved quality for the decision making process. Based on the IQmulus showcases, we will investigate, design and implement interaction methods to enable the users to gain more insight into their data as it is possible today with their current tools.

2.3.1 Work Performed

We have implemented an on-screen user interface component that is directly coupled with the GPU visualization techniques for interactive modification of parameters. The interface is based on libRocket, and has been modified to use the newest GPU features for efficient rendering of the on-screen interface.

We have also implemented an on-screen map, which shows the whole dataset as well as the location of the viewer. The on-screen map is based on render-to-texture functionality, so that it consumes minimal GPU resources. For every new dataset that is added to the scene, the dataset is first rendered into a texture using an orthogonal view from above. When navigating the scene, these textures are then translated and rotated according to the current camera position, and rendered into the lower left hand corner of the screen. This consumes minimal resources, as a single texture is rendered in place of the full dataset.

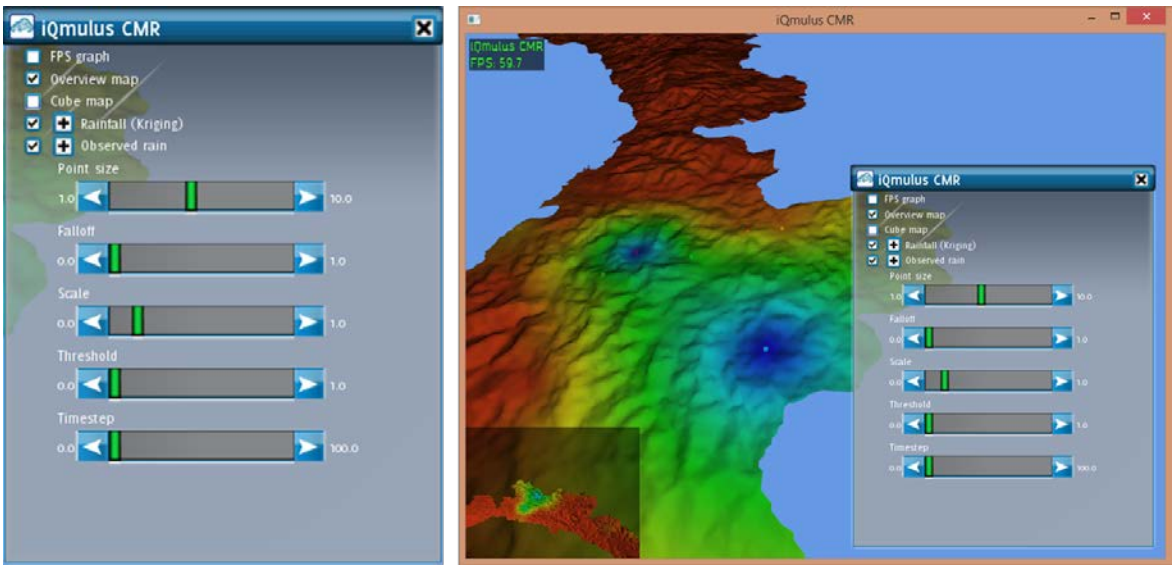

FIGURE 9: ON SCREEN UI AND MAP AS OVERLAY IND THE VISUALIZATION FRAMEWORK. A USER WILL INTERACT WITH VISUALIZATION PARAMETERS USING THE ON-SCREEN GUI, HERE SHOWN FOR THE LIGURIA DATASET.

During year 2 of the project, we also integrated the basic mechanisms for handling large raster images, e.g. multiband GeoTIFF, within the visualization framework. This enables us to process the data directly on the GPU. Based on these mechanisms we developed a prototype related to the IQmulus scenarios where this data format is used for storing results.

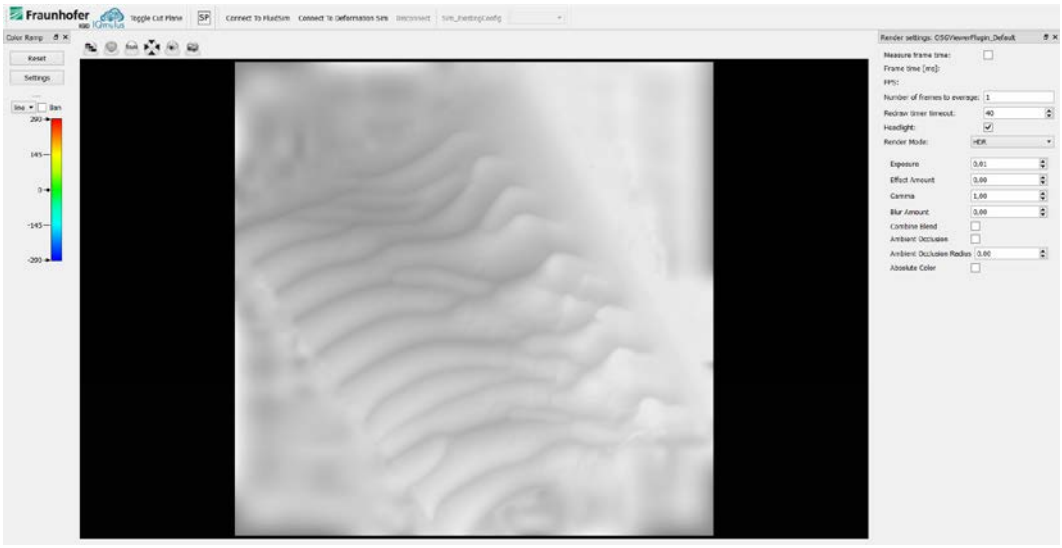

FIGURE 10: GEOTIFF OF SEA BOTTOM MEASUREMENTS LOADED AND TONEMAPPED IN THE IQMULUS VISUALIZATION FRAMEWORK

Figure 10 shows a GeoTIFF image of a sea bed scan in the IQmulus visualization framework. The image contains (negative) floating-point data, and the data is uploaded to the GPU using a floating-point texture. The values are mapped to the visible range using a fragment shader in the post-processing phase of a floating-point (High Dynamic Range) rendering pipeline. Please note that the per-pixel calculations performed on the texture may be replaced by arbitrary calculations, e.g. to perform a classification of a pixel.

The first prototype is related to the marine and land scenario concerning change detection in dune migration and landslides. In addition to the observed data, the changes, e.g. dune migration, between different acquisitions is calculated. To visualize this data interactively, we developed geometry and tesselation shaders, which generate 3D glyphs directly on the GPU, based on the data from the GeoTIFF. The shaders not only generate the glyph, but also take the size of the glyph in screen space into account to choose an appropriate level of tessellation (view dependent tessellation). The bigger the glyph would be on the screen the more it is tessellated. For small glyphs (in screen space) “low-tessellation” and for bigger glyphs “max-tesselation” will be applied (Figure 11).

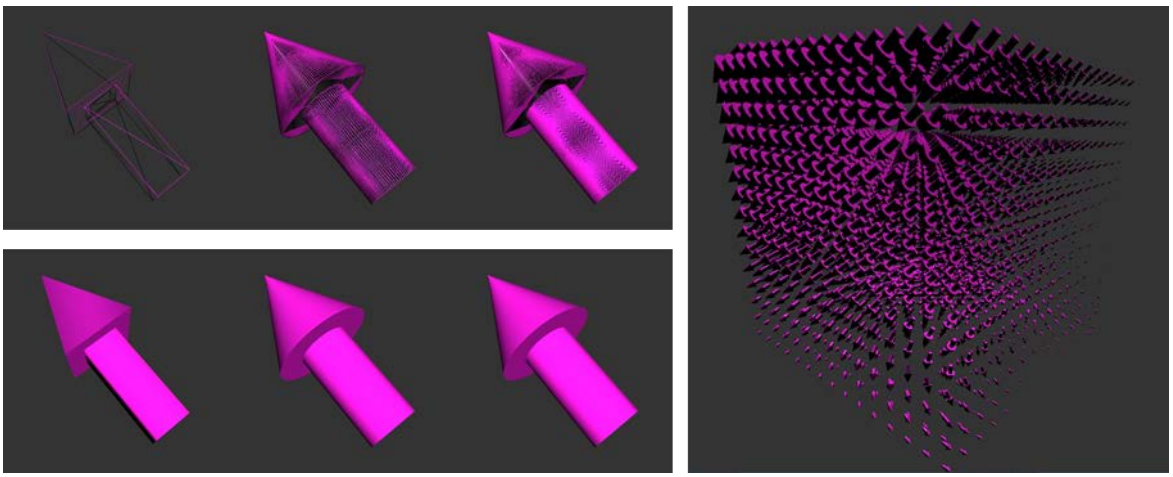

FIGURE 11: GPU GENERATED GLYPHS: LOW-, MID- AND MAX-TESSELLATED GLYPHS AS WIREFRAME AND SOLID(LEFT). A SYNTHETIC DATA SET IS VISUALIZED BY USING DIRECTION AND MAGNITUDE TO MANIPULATE THE GLYPHS (RIGHT).

In Figure 12 the glyphs are used to visualize the dune migration in the marine scenario. In contrast to a static image, the visualization is interactive and can be zoomed to different points of interest. The glyphs are automatically recalculated according to the users viewpoint, e.g. size, number, and tessellation level (Figure 13).

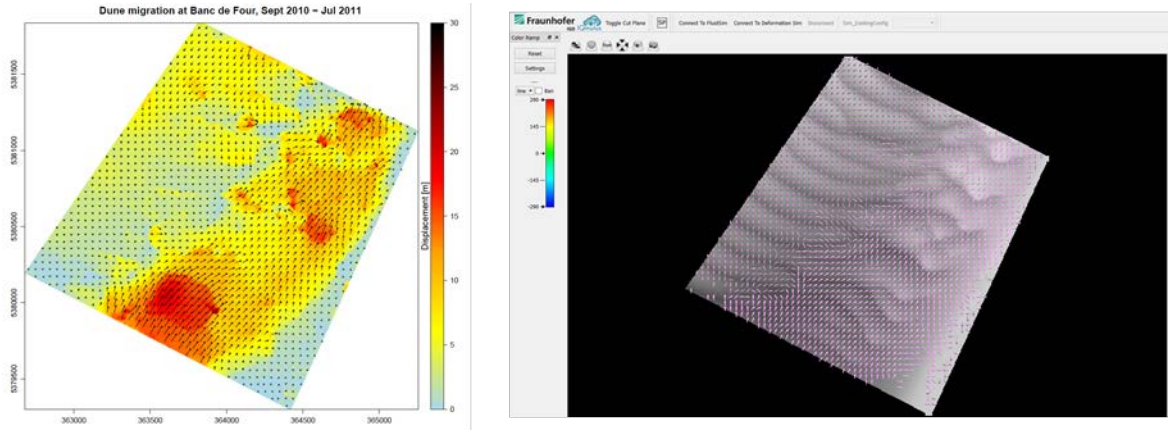

FIGURE 12: DUNE MIGRATION VISUALISED AS A STATIC IMAGE (LEFT) AND WITH INTERACTIVE GPU GENERATED GLYPHS

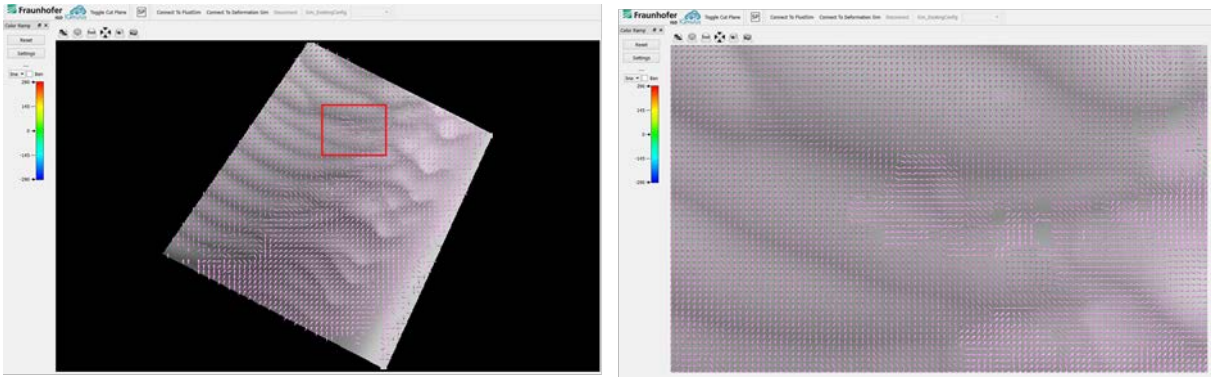

FIGURE 13: DIFFERENT ZOOM LEVELS OF THE DUNE MIGRATION DATA: FULL DATASET WITH MARKED AREA OF INTEREST (LEFT), ZOOMED IN VIEW OF THE AREA OF INTEREST WITH DYNAMICALLY RECALCULATED GLYPHS BASED ON THE VIEWPOINT (RIGHT)

2.3.2 Deviation of Work and Corrective Actions

There are no deviations from the work plan.

2.4 TASK 5.4: 3D-WEB-BASED VISUALIZATION

In this task the focus is on developing web applications by designing and implementing a fully automated Web Service Portal. The portal provides web services that automatically convert data into interactive 3D visualizations for the Web that can be delivered via a *hybrid* approach, providing both image streaming for low-end clients and direct web-based 3D rendering for more powerful thin clients.

2.4.1 Work Performed

We have implemented a library for rendereres called "Pixi", which supports a RESTful API for renderers. The initial specification of the RESTful API was published at Web3D 2013, and this library is responsible for managing communication and streaming between thin clients and the fat client. The library is easy to integrate into an existing architecture and the RESTful API is used to query scene objects such as a camera or a scene node. The API provides different access methods with content negotiation. Among other uses, this feature is important to query one specific object in various ways. Streaming of rendered images is handled with several methods, depending on what an end user device can support. The API offers single images as well as streams of images via WebSockets.

Pixi is not meant to be a remote rendering library alone. It can also provide partial computations performed remotely to enable demanding operations on a client. This may include computing data structures, calculating visibility, rendering complex materials and other similar operations. In the context of IQmulus, this means that problems which can only be partially solved on a client can be enabled by accessing a remote service (seeFigure 14).

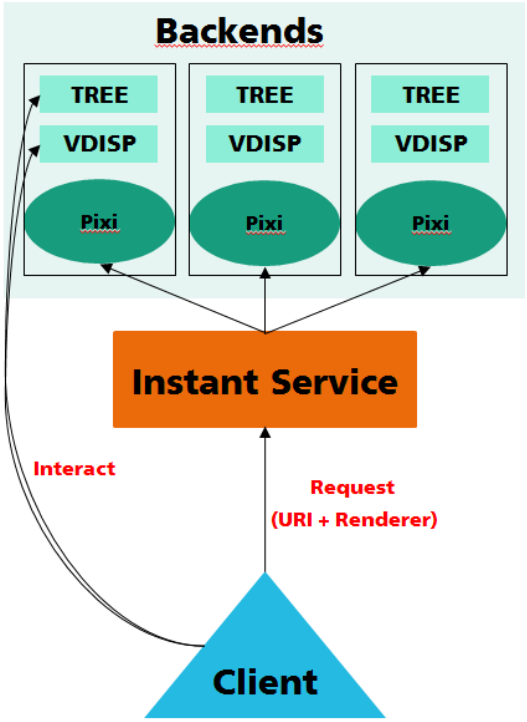

FIGURE 14: A CLIENT REQUESTS A URI TO BE RENDERED FROM AN INSTANT SERVICE, WHICH IN TURN INSTANTIATES A RENDERER THAT IMPLEMENTS THE PIXI INTERFACE. THE CLIENT AND PIXI RENDERER COMMUNICATE DIRECTLY AFTER THE COMMUNICATION HAS BEEN SET UP.

Since Pixi is a library, it needs to be integrated into a renderer. With Pixi the renderer is capable of streaming its render context to a client in arbitrary resolutions, and the client can control the camera view and manipulate a basic tree structure of the rendered content. This includes

- Switching geometry on/off
- Highlighting geometry
- Adding new geometry nodes to existing content
- Transforming geometry
- Removing geometry

The render backend is realized with OpenGL, but can be switched to an internal path-tracer if necessary. A GPU cloud environment has been set up to handle heavy rendering tasks, such as GPU path tracing. On the client side, a Javascript library has been developed to rapidly build web applications, which create a simple streaming context and some basic controls (see Figure 15). The Javascript library is currently being integrated into a full fledged client-side rendering environment, running in a web browser. Users can interact, examine and modify the rendered scene without noticing whether the client, a remote server or both are creating the image on screen.

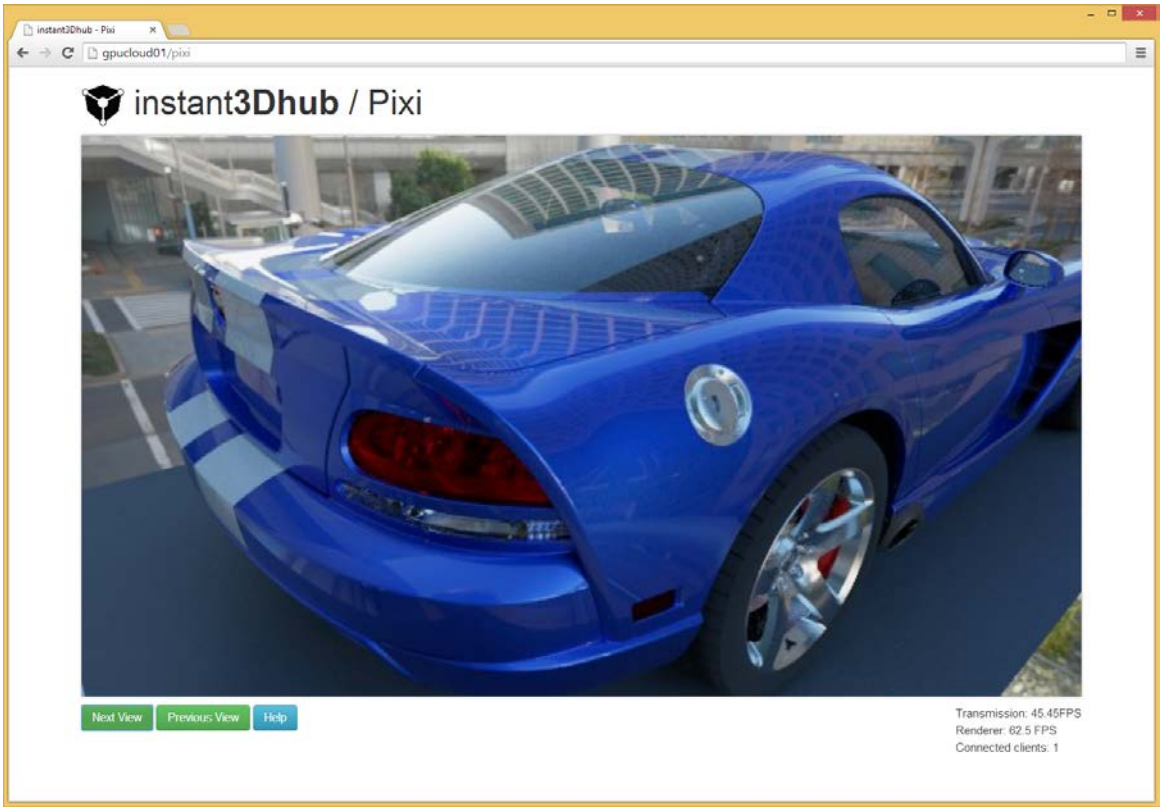

FIGURE 15: INTERACTIVE REMOTE GPU PATH TRACING VISUALIZED ON A WEB CLIENT RUNNING IN CHROME AT 45 FPS (ON A LOCAL NETWORK CONNECTION)

Whenever a client requests operations on any type of geometry or point cloud collection, a service daemon needs to be instantiated by another service, which accepts these kinds of requests. Formerly the transcoding service - which accepts arbitrary input mesh or point cloud data and transforms it into containers suited for web-based rendering - has provided this model. We have used this framework as a basis to build an abstract version called Instant Service, which can run any type of local operation on request.

2.4.2 Deviation of Work and Corrective Actions

There are no deviations from the work plan.

2.5 TASK 5.5: VISUALIZATION DRIVEN DATA FORMATS

This is mainly a research, analysis and design task within which we will collect the requirements on data formats derived from our algorithms developed in the tasks 5.1 to 5.4. These requirements will be mapped and compared to existing data formats. The comparison could be used in the future for developing visualization-aware data formats or extending existing ones so that they are optimally tailored to new graphics hardware architectures and GPU-based algorithms. An immediate application of such new or extended data formats in IQmulus is not foreseen, since most layers would be affected by data format changes, requiring additional adaptation and development efforts by the affected project partners.

This task starts in M31.

3 CONCLUSIONS AND CONSOLIDATED PLAN

In the second project year the activities of WP5 were focused on enhancing the visualization modules already developed in the first project year and extending the IQmulus visualization framework. The developments were driven by the IQmulus showcases, defined during the second project year.

The most significant results within WP5 during the current reporting period are:

- Modification and integration of LR splines visualization
- Full OpenGL Shader support in IQmulus visualization framework
- Screen Space Triangulation for point clouds (from deferred rendering engine)
- On-screen UI which is directly coupled with the GPU visualization
- Integration of GeoTIFF handling into the framework
- GPU generated glyph visualization
- Finalized RESTfull interface for server rendering engine
- Finalization of the Pixi library

For the next project period the main focus of WP5 will lie in refining the existing prototype based on the user test and not yet covered parts of the showcases as well as in further developing visualization and interaction capabilities related to the geospatial domain.

The main topics will regard:

- supporting large datasets without loss of interactivity,
- supporting new file formats (such as raster and vector datasets)
- adding support for new visualization types for the showcases.
- continue the work on streamed / dynamic point cloud visualization
- investigation of local data handling on the client side, including caching mechanisms for received data as well as GPU-driven (deferred) data fetching.
- integration of data, which is currently not used, into single visualizations to provide new means of data analysis to the user
- The further development of the GPU-based glyph generation including different glyph types, property mappings, and interactive updates for time-varying data
- investigate interaction techniques such as interactive glyph density for user-specified areas of interest
- full integration of Pixi into the IQmulus visualization framework to provide full streaming support for all renderers
- investigation of optimizations to the data structure and when to trigger scenegraph and mesh optimizations depending on the input to the transcoder
- creating a library for the proposed common data structure called HSI (Hybrid Spatial Index), which is a JSON container only describing the structure
- extending the Pixi library to support 1D and 3D buffers. While this works theoretically most web clients only understand 2D buffers.
- create a web based renderer which provides a 3D context that, from a user perspective, transparently handles all rendering calls