INTERACTIVE VISUAL DECISION SUPPORT TECHNIQUES

Deliverable D5.1.1

Circulation:	PU: Public
Lead partner:	Fraunhofer
Contributing partners:	SINTEF
Authors:	Frank Michel, Tobias Franke, Thomas Gierlinger (Fraunhofer), André Brodtkorb (SINTEF)
Quality Controllers:	Ewald Quak, Tor Dokken (SINTEF), Michel Krämer (Fraunhofer)
Version:	1.0
Date:	31.10.2013

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	18.06.2013	Draft	Initial draft with deliverable structure
0.2	26.09.2013	Draft	First draft version of the deliverable
0.3	20.10.2013	Draft	Updated draft with partner / task leader input.
0.4	28.10.2013	Draft	Additional input from partners
0.6	30.10.2013	Draft	Proofreading by partners
0.7	31.10.2013	Final	Ready for QC
1.0	31.10.2013		Version to be submitted to the Project Officer

[1] Integers correspond to submitted versions

EXECUTIVE SUMMARY

This report presents the progress made in *WP5 Interactive Visual Decision Support Techniques* during the first year of IQmulus. The document summarizes the general vision and objectives of WP5, the position within the whole IQmulus project and describes the WP5 components and their interplay in more detail. The work conducted in WP5 during the first year is reported on the task level, as well as an outlook on the planned work in year two.

In general, the first year of activities in WP5 were devoted to understanding the user needs in order to transform them into technical specifications for guiding the development of the visualization technology. Moreover, the design of the interplay between the different visualization components and therefore the visualization architecture was conceptualized. This process was also supported by prototypical integration and testing of the expected visualization workflows. Based on the initial testing and evaluation a framework for the development of new interactive visual decision support techniques was set up based on the visualization architecture. First basic building blocks for the visualization prototype, which is due after year two, were implemented and tested with data from the processing services. The already implemented modules regard the Composite Multi-resolution Rendering, Deferred Mapping and Web-based Visualization. These modules will be extended and enhanced in the next project year.

TABLE OF CONTENTS

LIST OF FIGURES

1 VISION OF WP5

WP5 aims to develop visualization techniques for the *Interactive Visual Decision Support* in the context of geographical information systems. This WP is about visualization applications tailored for geo-experts and decision makers using fat and thin clients. On the one hand, fat clients will use the advantages of desktop applications with access to GPU resources and therefore to modern OpenGL technology. On the other hand, 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 (not for data administration or data processing experts).

1.1 OBJECTIVES AND EXPECTED RESULTS OF WP5

The core objective of WP5 is to develop a 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.

The technical evaluation of the feasibility of the various envisaged approaches to transfer GPU-based visualization techniques to the geospatial domain is part of the tasks of WP5.

1.2 POSITIONING OF WP5 TECHNOLOGY WITHIN IQMULUS

WP5 is one of the technological work packages in IQmulus. Figure 1 shows the synergies that WP5 has with other work packages in the project. In this context, the requirement analysis in WP1 will extract and consolidate the needs and the stories from the users in order to guide the technical development for the visualization technologies. The work in WP2 will design the baseline architecture as well as the different services and the general communication channels with the visualization components. WP3 will set up the technological infrastructure of the project, including the storage and the transmission of data resulting from WP4 and which will be used by WP5. Finally, WP6 will deal with the overall integration and testing of the developed components and services.

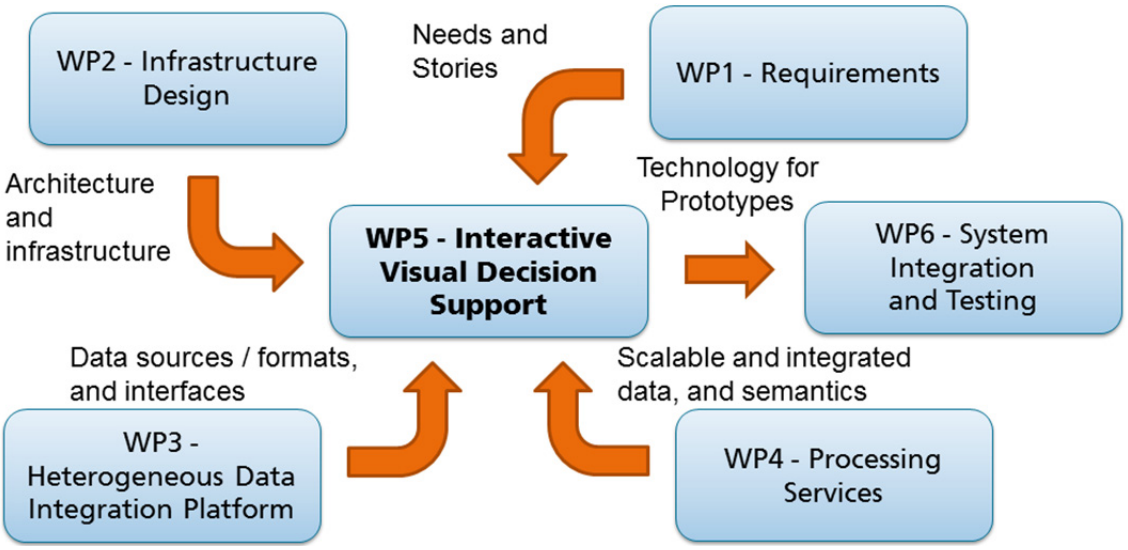


FIGURE 1: WP5 SYNERGIES WITHIN THE PROJECT STRUCTURE.

Figure 2 shows the position of the visualization technology within the current baseline architecture of IQmulus. The foreseen communication channel between the workflow editor and the visualization component will be used to provide information about the data to be visualized, for instance a URL of the location of the file(s). Moreover, the same channel will be used to transmit information from the visualization component to the workflow editor, e.g., about selected regions of interest by the user, which might be used for triggering a further step in the workflow. The visualization component has also a direct communication channel to the data access service, from where the data to be visualized will be retrieved.

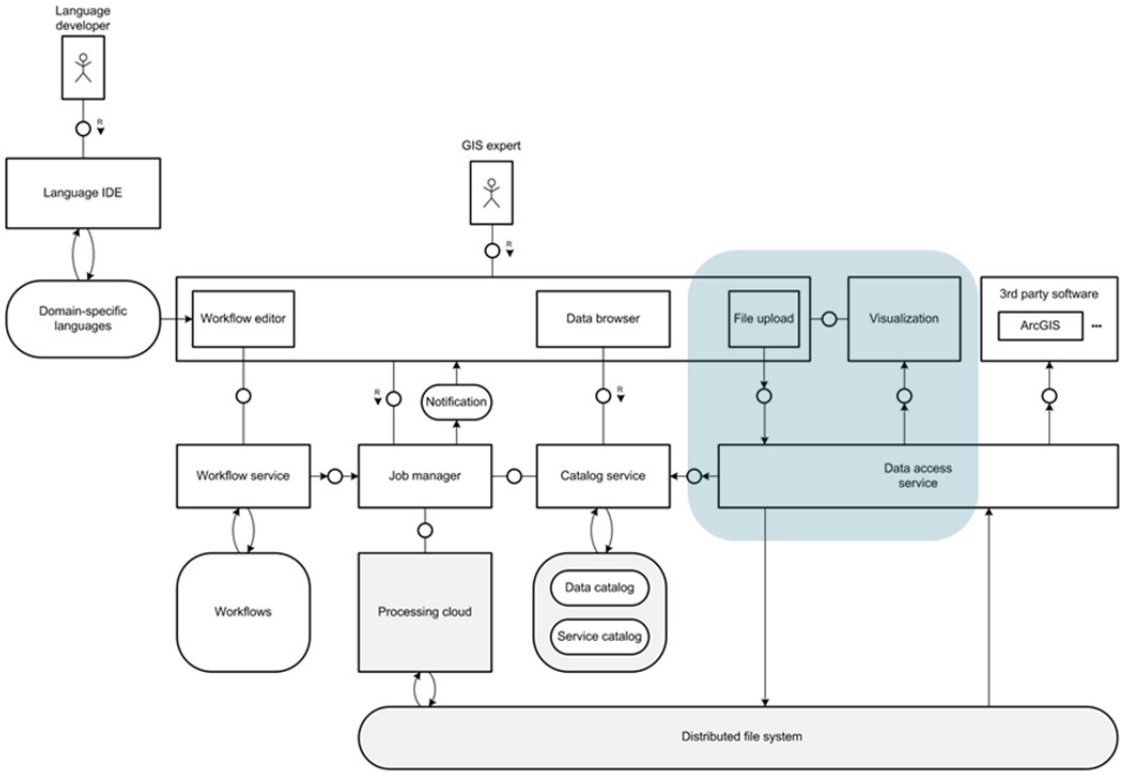


FIGURE 2: WP5 WITHIN THE BASELINE ARCHITECTURE OF IQMULUS.

1.3 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. All 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.

1.3.1 Composite multi-resolution visualization

Interactivity is a key concept that is required for a user to gain insight into the underlying data sets: a user must be able to navigate and interact with the data at a high frame rate, and with a minimal delay on interaction. To achieve this goal for large data sets, modern hardware features of graphics processing units will be utilized in combination with multi-resolution methods to dynamically reduce the amount of data being shown at a time. Multi-resolution in this sense includes building acceleration structures (such as k-d trees, quad trees, etc.), and representing the data in a compact fashion suitable for use in modern graphics processors.

The composite and multi-resolution visualization further aims to combine different types of data into a common view. Nothing can beat the eyes of a trained expert in certain situations, and human experience and intuition can give a skilled operator insight in new and unforeseen ways. The composition of multiple data sets can therefore increase the visual understanding, and draw forth new meaning from the composition. A naïve example of composition is the combination of a digital elevation model, point clouds, satellite imagery and flood simulation results. A flood simulation will not always be able to correctly capture "small" features (such as a levee, rail road, or a highway), as these can be smaller than the resolution of the simulation grid. These features, however, often have a strong impact on floods. Whilst an expert user will quickly discover if the flood simulation takes roads and train tracks into account, it can be difficult to do so automatically for a computer.

The rendering technique for different data sets might be “incompatible”, in the sense that their composition is highly non-trivial (especially true for transparent objects). We therefore focus our efforts on developing new scientific visualization methods that will extract the most information from combinations of data sets such as

- Vector-based cartographic information (e.g., hydrography, cadastre, roads, utility lines);
- Raster-based terrain data (e.g., elevation data, land use);
- Vector- and tensor-based simulation data (e.g. , wind, pollution, flooding).

1.3.2 Deferred visualization

The deferred visualization module aims at supporting visual decision making and interactive visual communication on large data sets by utilizing screen space information for image generation as well as user interaction. The core concept involves the adaptation, extension, and further development of a real-time rendering algorithm termed Deferred Shading to fulfil the requirements of scientific visualization. The basic idea of Deferred Shading is to decouple complex per-pixel shading calculations from geometric transformation and visibility detection when rendering a 3D model.

In a conventional feed-forward rendering pipeline, a primitive (triangle or point) is transformed into the camera coordinate system, projected onto the image plane, and subsequently scaled / translated by the current viewport transformation to yield window coordinates (also called screen space or image space coordinates) of the primitive's vertices. The screen space coordinates of the primitive's vertices are 3-dimensional, consisting of the (x, y) position in a window and an associated depth (related to the distance from the image plane). Afterwards the primitive is rasterized - i.e., split up into pixel-sized fragments for which the colouring / shading calculations are applied. After these operations a visibility test is performed for each fragment by comparing the fragment's depth to the depth of the closest previously rasterized fragment at the same screen space (x, y) position. If the depth of the current fragment is greater than the one of the previously rasterized fragment, the current fragment is further away from the camera than the previously rasterized one and therefore discarded. Otherwise the calculated color of the current fragment is written to the frame buffer for display and the depth of the current fragment is stored in a memory region on the graphics board called depth buffer (or z-buffer) for subsequent visibility tests. This approach to visibility determination is hardware-accelerated and generally fast. However, since primitives are handled independently of each other, a pixel in the frame buffer may be overwritten multiple times during rasterization (whenever a fragment closer to the camera is determined). The coloring / shading computation for discarded fragments is wasted, which provides an opportunity for optimization that is exploited by Deferred Shading.

Deferred Shading, as the name suggests, defers the coloring / shading computations to a later stage of the rendering pipeline, namely after the visibility determination. This is accomplished by a multi-pass rendering approach. In the first pass, the geometry is rendered without applying any coloring / shading using the standard feed-forward pipeline. The required information is stored in dedicated memory areas on the graphics board (off-screen buffers) analogous to the depth buffer. For shading calculations these off-screen buffers may contain screen space 3D positions, normals, and material information. The off-screen buffers are collectively referred to as G-buffer (geometry buffer). The information in the G-buffer is utilized in the second rendering pass to perform per-pixel shading for only the visible fragments.

Developing visualization and interaction methods based on a deferred rendering approach promises increased interactivity in the IQmulus context for the following reasons:

- Complex per-pixel visualization calculations are performed only on the visible part of the data (which is especially relevant since we are dealing with large data sets)
- As long as the camera position and orientation do not change, the screen space information remains valid. Accordingly, any user interaction in screen space is foreseen to be significantly faster than interaction working directly on the data, since there is no need to re-render the data set.

Since scientific visualization is not about realistic shading but mapping data to meaningful representations, highlighting relevant areas in the data set, we call our approach deferred

mapping. We have developed an initial version of the deferred mapping module which is detailed in section 3.2.1.

1.3.3 3D-web-based visualization

For the 3D-web-based visualization we build on the open standard X3D, for which a browser-based implementation X3DOM exists. X3DOM in turn employs WebGL, an open standard derived from OpenGL supported by all major browser vendors, with which data sets are visualized. This part is meant to work on mobile devices and other thin clients powerful enough to render 3D data sets. The user's end-device retains full control of the data and its visualization, but may stream additional sets as needed when navigating through the scene. Since thin clients will rarely be used for scientific analysis rather than exploration, the visualization on these devices will be simple but broad enough for experts to take a quick look at the data.

Because of the potential size of the data sets which will be rendered, additional optimization structures need to be employed to efficiently display them. However, the data provided by WP4 might not be in such a format. Web-based applications furthermore need specific input formats native to Javascript only. It is therefore necessary to transform WP4 data sets to make them web-compatible. The HUB service is a major component which will ultimately handle WP4 data and turn it into a proper web-based visualization.

1.3.4 HUB

A major obstacle when delivering large data sets to thin clients such as tablets and smartphones is the wide variety of hardware capabilities. Mobile clients may or may not have, e.g.:

- dedicated GPU support
- enough RAM
- sufficient space on the screen
- bandwidth to download data
- security lockdowns to avoid for instance GPU hardware access
- etc.

Furthermore, web-based 3D applications require specific data formats in order to run. To overcome these issues, we build the HUB service, which when given input from the processing services and a request from a device that wants to view/visualize the data, decides on how to best deliver it. This may involve modifying the data, pre-processing it, or even require remote rendering because the end-device is not capable of doing so. This HUB service will contain two major components: a transcoder, which can reformat input from WP4 suitable for a device, and a remote renderer (preliminarily called "Pixi" based on the Instant Reality platform).

Web Service Architecture with transcoder: The HUB service extracts a template, which specifies the data and application characteristics and then it invokes the appropriate transcoder service for providing the concrete data converting it into a declarative deployment format for final presentation.

Client-side rendering: The visualization application can be realized via client-side rendering by utilizing X3DOM for real-time rendering. Important research issues here are scalable methods for binary compression and suitable caching strategies.

Server-side rendering: Alternatively server-side rendering is promoted, where the rendered images are transmitted/streamed to the client. Interactions are handled via WebSockets or XHR.

To cope with latency issues, suitable hybrid rendering and interaction methods have to be explored.

1.4 INTERPLAY BETWEEN THE TECHNOLOGICAL COMPONENTS OF WP5

This section describes only the architecture of the visualization part. For a complete overview of the system architecture and how the visualization component is integrated there please refer to deliverable D2.3.1 *IQmulus Architecture Design – baseline version*.

WP5 will provide two software components that are visible to end users:

1. A fat client, which is visualization software targeted at a standard desktop PC with a single, current generation graphics board.
2. A thin client, which utilizes 3D web technology and the automated visualization adaptation framework (the HUB), targeted at web browsers on desktop machines as well as on different kinds of mobile devices.

On the one hand, the fat client is envisioned to be operated by an expert user to explore results generated by the processing services and prepare the data for subsequent presentation. It features the GPU-based visualization and interaction methods (and only those) to accelerate this process. On the other hand, the thin client is focused on presentation scenarios and does not provide advanced interaction with the visualization data.

The internal interplay between the technological components of WP5 and the corresponding communication is shown in Figure 3.

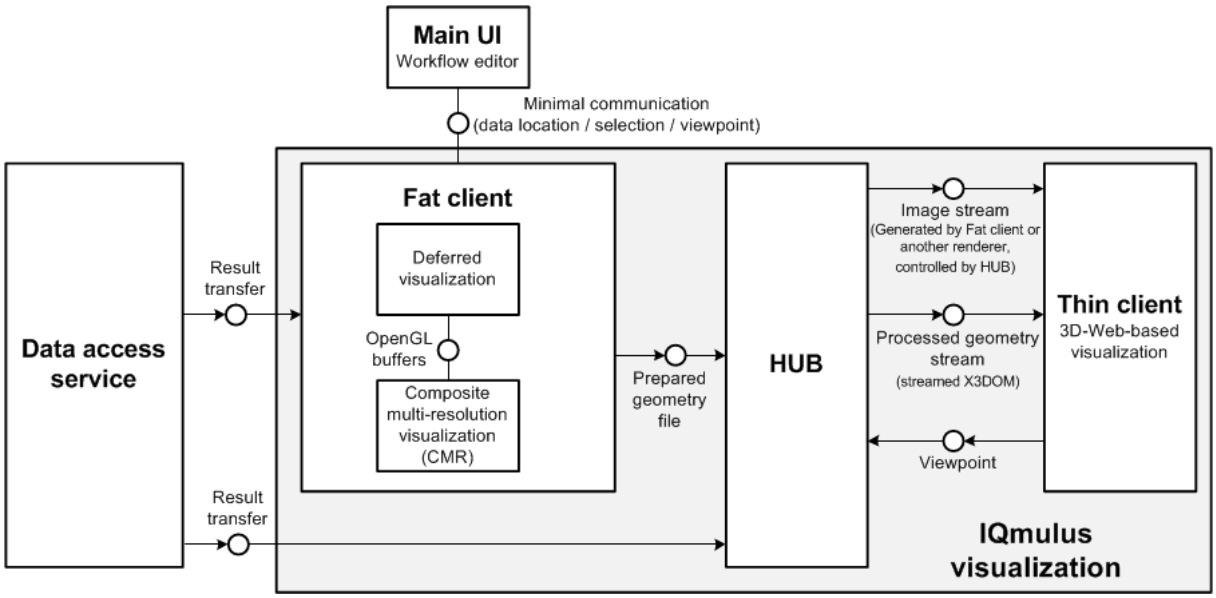


FIGURE 3: ARCHITECTURE OF THE IQMULUS VISUALIZATION.

The internal communication for the visualization architecture works as follows: *processed result data* (input data for visualization) is transferred from the processing services via the data access service to the fat client, which utilizes deferred and / or composite multi-resolution visualization to support a user in the analysis of such data. The *visualization result data* (input data manipulated by the visualization components) is a prepared or enhanced part of the original data set which highlights the relevant information in the data. This *visualization result data* (mainly geometry data) is communicated to the HUB, which chooses an adequate representation and visualization for the thin client, taking into account the client-side resources.

For high-end clients (e.g., web browsers running on desktop PCs) the prepared geometry after being transcoded to a 3D web compatible format (e.g., X3DOM) might directly be transferred and rendered on the client. Mid-range clients (web browsers on smaller laptops, netbooks or tablets) might require further reduction of the geometry, e.g., by external visibility determination. In this case, only the currently visible data is streamed to the client for performing the rendering. For low-end devices (e.g., web browsers on small smartphones) client-side rendering might be difficult due to resource limitations. In order to overcome this obstacle, a server-side rendered image stream is provided to the client. Server-side rendering could be provided by a dedicated instance of the fat client, which is accordingly controlled by the HUB. In this scenario the fat client has to provide a rendered image stream as well as provisions for external control of the camera / viewpoint.

To enable inspection of the original (processed) result data (data without being prepared for visual decision making) of the processing services using the thin client, a direct connection for data transfer from the data access service to the HUB is foreseen.

The processing services are configured and started by the workflow editor. The user interface to specify parameters for the different processing algorithms is part of the workflow editor. The workflow editor and the fat client are two different applications that can be operated in a stand-alone fashion. There will be minimal communication between the fat client and the workflow editor, in order to boost the advantages of GPU-based visualization and interaction. This communication will include only the specification of datasets / layers for visualization and data resulting from the accelerated GPU-based interaction, namely:

- Definition of the region of interest
- Visualization settings / parameters derived from user interaction with GPU-supported preview functionality

A high-level description of the interface between workflow editor and the fat client is given below:

Input to fat client (provided by workflow manager)		
Name	**Parameter**	**Description**
addLayer	URL to result file	Add a result layer to the visualization
deleteLayer	URL to result file	Remove a result layer from the visualization
setView	Camera description	Set the viewpoint / camera of the visualization to a given position / orientation
Output of fat client (sent to workflow manager)		
Name	**Parameter**	**Description**
sendRegionOfInterest	List of points	Vertices of a polygon defining a selection / region of interest

The communication between the fat client and the HUB is defined in a similar manner:

Input to HUB (provided by fat client)		
Name	**Parameter**	**Description**
exportDataSet	Geometry file	Transfer prepared geometry to HUB
exportViewport	Image or point cloud	Export current viewport to HUB for streaming to thin client

Output from HUB (sent to fat client if used for image stream generation)		
Name	**Parameter**	**Description**
setView	Camera description	Set the viewpoint / camera of the fat client to a given position / orientation

A workflow (chain of processing services) is always started from the workflow editor user interface, never by the fat client visualization. No communication with the processing services (except result data transfer via the data access service) is foreseen. The proposed user interface for the fat client is outlined in Figure 4.

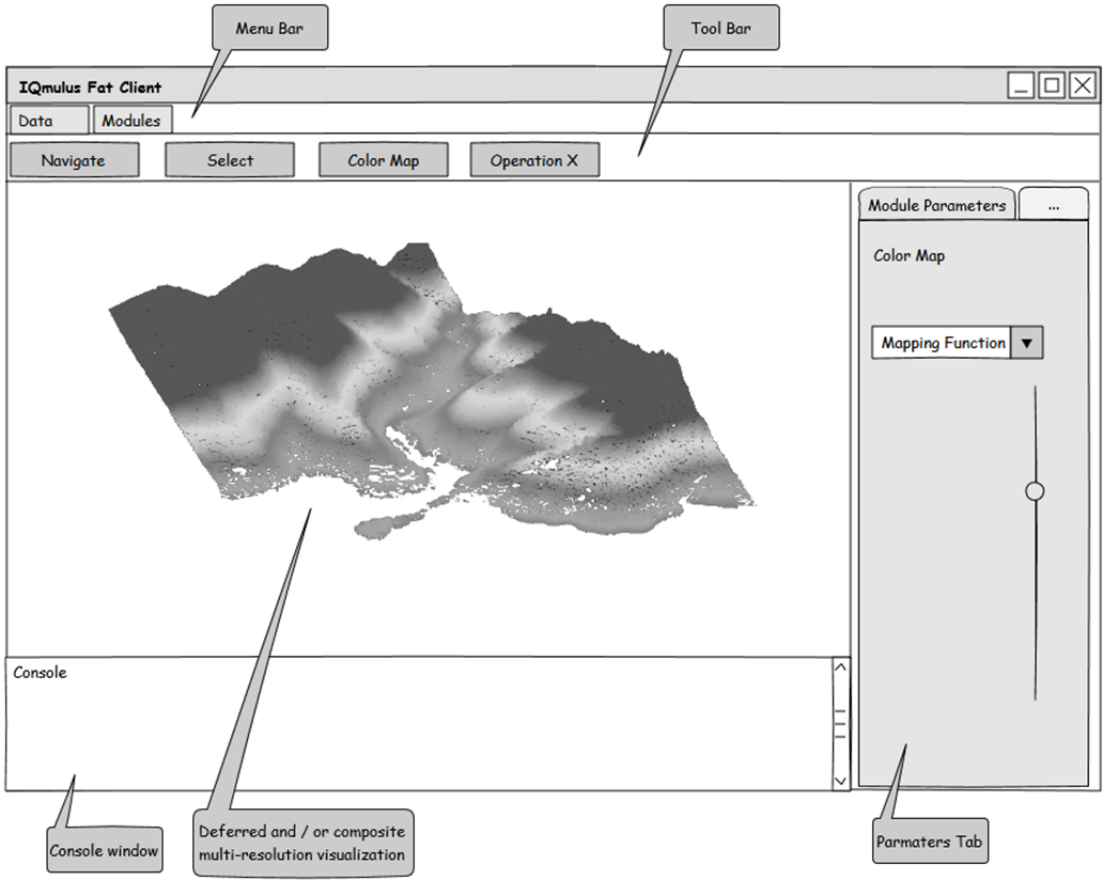


FIGURE 4: FAT CLIENT USER INTERFACE.

The functionality of the fat client will be grouped in modules. Each module might provide a number of operations or interaction modes. Operations of a module can be started via buttons in the tool bar. For each module, a dedicated tab widget for display and interactive adjustment of the parameters will be provided. The user can decide to show or hide the user interface associated with a module via a corresponding menu in the menu bar. The main area of the fat client window displays the current visualization (deferred and / or composite multi-resolution visualization). Moreover, a console window is foreseen for textual information display.

The thin client user interface is less complex, since its focus is on presentation scenarios. The thin client will display the current data in a (3D) web browser and offer user interface elements to adjust navigation modalities. Furthermore, the interfaces between thin client and HUB utilize open standards, namely a combination of REST and WebSockets for communication. These interfaces have been tested in initial implementations and proven suitable for parameter

transfer (REST) and streaming geometry / images (WebSockets). The interfaces between fat client and HUB as well as fat client and workflow editor will use the same technology.

The proposed approach for the integration of the deferred and composite multi-resolution visualization is to use OpenGL buffers (OpenGL's internal data structures which hold geometry and image/image space information). The creation of the buffers is performed by the fat client (which also maintains ownership of the buffers). The IDs, which allow the access to the buffers, will be communicated to the deferred visualization and composite multi-resolution visualization subsystems. This approach should allow for

1. Separated visualization: either deferred visualization or composite multi-resolution visualization fills the final image buffer.
2. Blended visualization: both visualization subsystems create a partial result and the final image buffer is created by overlaying/blending these partial results.
3. Re-use of image space buffers generated in the deferred visualization by the composite multi-resolution visualization or vice versa.

The OpenGL buffers are intended for shared resources only. Both visualization subsystems are free to generate or load additional data into private memory (CPU-side memory as well as GPU-side memory). The fat client will support data handling of the subsystems by providing data transfer from the data access service and local storage. Pointers to (CPU) RAM containing loaded data as well as ownership of the memory will be handed over to the requesting subsystem.

2 CONSOLIDATED REQUIREMENTS FOR WP5

The consolidated requirement phase (see Deliverable D1.2.2) resulted in the formulation of three super user stories within the context of land and marine scenarios:

1. Land showcase 1: “disaster management in case of flood”;
2. Land showcase 2: “assessment of flood damage”;
3. Marine showcase: “deriving accurate elevation models from heterogeneous data sets”.

Based on these main super user stories, the partners in WP5 identified the most relevant requirements which might have direct or indirect relations to the visualization technology. These requirements were classified into:

- User requirements;
- External (to WP5) technical requirements;
- Internal (to WP5) technical requirements.

This set of requirements defines the baseline of the technological development. Further requirements will be added to the baseline by means of further interaction with end users, either at supervised workshops or at their own premises after deploying the technology.

Requirement	Type
The visualization technology must support a range of different file formats including elevation models (e.g., DTM, DEM, DSM) and semantic information (e.g., shorelines, dams, channels, roads). The consortium has defined a reference file format per data type. Other formats may have to be converted to the reference formats before visualization. The currently agreed set of reference formats includes LAS, Shapefiles, GeoTIFF, PLY, OFF, XYZ, ASCII grid, NetCDF.	User/Internal
The main technological requirement of the fat client is data size. The fat client developments focus on exploiting a single GPU in a desktop PC. It is expected that the data coming from the data access service is in a format and size that fits into the main memory of the GPU (roughly 2GB). Hierarchical and / or streamable data formats and services might fulfil this requirement. Data files need to be provided as tiles, which are manageable by WP5 technology.	Internal/External
The fat client requires Microsoft Windows.	Internal
The fat client requires graphics hardware, which supports OpenGL 4.4+	Internal
Thin clients without dedicated 3D support require a browser with streaming capabilities (support for MJPEG or other means).	Internal
Thin clients with dedicated 3D support require a 3D-capable web browser (WebGL support), such as Google Chrome on Android or a custom app with WebGL enabled WebKit support on iOS.	Internal
The visualization must provide an overview map of the currently displayed results. As the visualization can only support a subset of the full data set due to memory constraints, an overview map of the full data set needs to be generated externally or on a reduced data resolution.	User/Internal
The visualization technology must be interactive (which means a frame rate of at least 15 frames per second) for representative/typical data sizes.	User
Uncertainties in simulation results and models must be visualized in an intuitive manner.	User

3 PROGRESS REPORT ON WP5 TASKS

In order to prepare the work to be conducted in WP5, the involved partners prepared questionnaires to collect information on the different software tools used by the members of the consortium. The collected feedback was evaluated from a visualization perspective. An additional questionnaire focused on collecting information about the use of physical simulation, and its corresponding support was designed and prepared. WP5 partners exchanged information via emails and Skype/telephone conferences in order to synchronize activities and to check the status of the results. Moreover, the involved partners have attended six physical meetings:

- The technical meeting in Darmstadt (5th November, 2012) was used to get a first impression of the integration issues between the different technical work packages. As part of this meeting, the various WP5 partners made presentations of their core technologies followed by discussions on the visualization requirements towards integration.
- At the kick-off meeting in Genova (21st - 23rd November, 2012) the core technology and expertise of the WP5 partners were presented to the consortium in order to motivate discussions and the development of scenarios as examples for the requirements analysis phase. Furthermore, the WP5 partners were involved in the architectural discussion of the IQmulus system.
- WP5 partners also participated in the plenary meeting in Budapest (18th – 20th March, 2013), and the requirements consolidation meeting in Darmstadt (22nd – 23rd April, 2013). During these two meetings, WP5 partners contributed to the analysis of user stories, and the extraction of use cases, as well as to the preliminary consolidation of the technical requirements, especially from the point of view of the visualization technology. As preparation for these meetings, the WP5 partners collected a set of synthetic user stories that envision possible visualization outcomes, in order to convey the potential of the technology to the end users.
- WP5 partners participated in the Code Camp Week (June 3rd – 7th, 2013) in Darmstadt. For the code camp, the WP5 partners defined two software projects to be developed during that week in order to explore the capabilities, interoperability, and integrability of the technological components provided by the individual partners. The two software projects were described as:
    1. Integration of deferred and composite multi-resolution visualization (fat client)
    2. Integration / data transfer between fat client, HUB, and thin client
- All WP5 partners participated in the IQmulus plenary meeting in Paris (October 14th – 16th, 2013). During the meeting a review of the first project year for WP5 as well as a planning session for the second project year from a technical viewpoint were conducted. Furthermore, a special joint WP4/WP5 session was held to synchronize between the processing services and the visualization component.

In general, the first year of activities was devoted to understanding the user needs in order to transform them into technical specifications for guiding the development of the visualization technology. Moreover, the design of the interplay between the different visualization components and therefore the visualization architecture was conceptualized. This process was also supported by a prototypical integration and testing of the expected visualization workflows.

The development of the two software projects of the Code Camp generated promising results in terms of compatibility and integrability of the technologies. Preliminary results demonstrated the potential offered by the visualization technology toward geo-spatial visual decision support. This was a first step toward the development and integration of the different technological

components, which assisted the partners in identifying the challenges to be handled during the upcoming period of the project. Details regarding the WP5 internal interfaces are described in Appendix 6.1.

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

3.1 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.

3.1.1 Work Performed

The focus in the first project year has been to develop a framework for interactive composite and multi-resolution visualization (called the CMR framework) and to integrate this framework into the fat client. The basic CMR framework has been developed, and the first prototype was integrated into the fat client during the Code Camp.

One of the main building blocks of the CMR framework is a library that can manage modern OpenGL details (including states, shader programs, buffers, and other data). The library includes support for basic and often used data types (such as triangle soups, indexed triangle soups, point clouds, etc.), and for basic and often used visualization techniques (Phong shading, textured rendering, etc.).

Built on top of the low-level OpenGL management library, we have constructed tools and utilities that aid in the development of more advanced algorithms. This includes texture-based text rendering and an accurate frame rate counter (visible in the lower left of Figure 5) to keep track of performance.

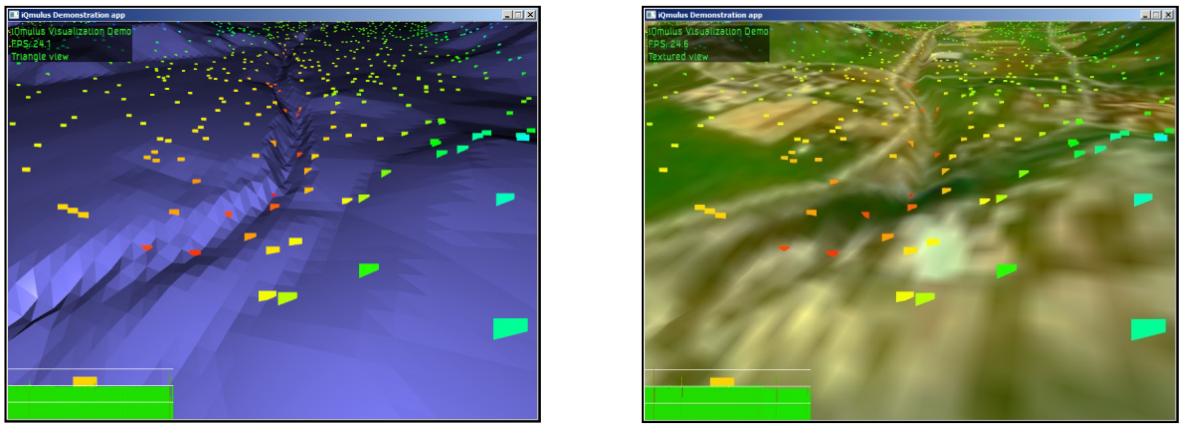


FIGURE 5: INSPECTION OF A DIGITAL ELEVATION MODEL AND THE ORIGINAL POINT CLOUD (VERNAZZA DATA SET PROVIDED BY LIGURIA). WHEN AN ORTHOPHOTO IS ADDED (RIGHT), WE SEE THAT THERE IS A MISMATCH BETWEEN THE TERRAIN/POINT CLOUD AND THE IMAGERY.

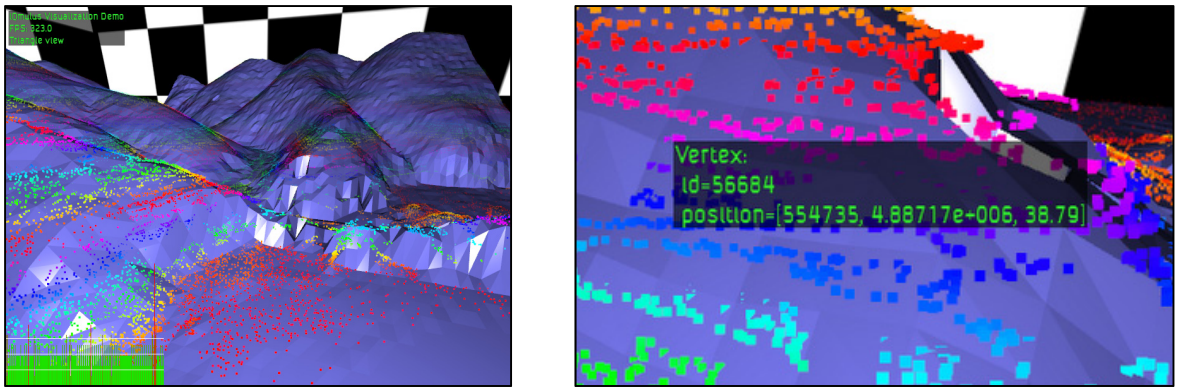


FIGURE 6: COMPOSITION OF A POINT CLOUD (VERNAZZA DATA SET PROVIDED BY LIGURIA) AND A DIGITAL ELEVATION MODEL. WHEN ZOOMING IN ON SPECIFIC PARTS OF THE MODEL, A USER CAN INSPECT THE ACTUAL DATA VALUES FOR EACH POINT.

Simultaneously to developing the framework, we have also developed the first prototype of a composite visualization. The visualization (see Figure 5 and Figure 6) displays a point cloud (Vernazza Data Set provided by Liguria) and terrain model in the same view. A user can enable or disable the overlay of an orthophoto, and also inspect individual data points to see their specific value. We lessen the effect of information overload by making the point cloud transparent when far away from the camera, and optionally only display a subset of the original points. At the same time, we offer a user full flexibility to inspect the underlying data set by "picking" a single point to display its values (see Figure 6, in which position (554735 m, 4887170 m) in the data set has the elevation 38.79 m.)

3.1.2 Deviation of Work and Corrective Actions

There are no deviations from the work plan.

3.1.3 Work Plan for the next Period

The focus of the first year has been to develop the foundations for the CMR framework, and a first prototype of composite visualization has been developed. Based on this framework, the focus in the second year will shift to developing new techniques and methods for composite visualization of different data sets (vector, raster, 2.5D, 3D, tensor, etc.). As new visualizations tailored towards the needs of the super user stories are being developed, it will also be necessary to develop multi-resolution techniques to achieve interactivity for even large data sizes. This can include building acceleration structures such as k-d trees, quad/oct trees, and to generate compact representations suitable for modern graphics hardware and the underlying data set.

In summary, the focus will be on developing new methods for the interactive visualization of different data sets to address the needs of the super user stories.

3.2 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 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?

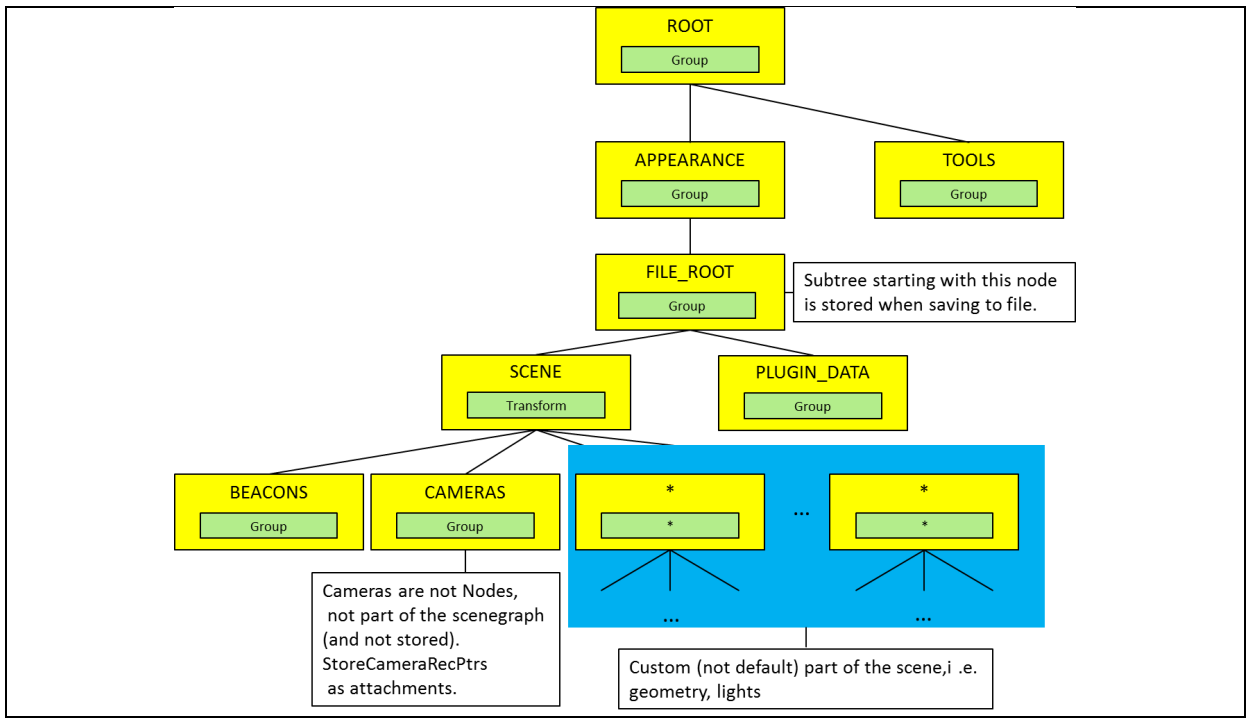
3.2.1 Work Performed

One of the main activities in this task during the first project year was the development of the fat client framework including proof of concept implementations of the deferred mapping and composite multi-resolution modules. The fat client developments are based on the Fraunhofer visualization framework RPE (Rapid Prototyping Environment). The RPE utilizes the OpenSG[2] scenegraph for OpenGL-based visualization, provides extensibility via dynamically loadable modules, and event-based support for user interaction. The user interface is realized using the Qt library.

Deferred mapping component

The development of a proof of concept deferred mapping module required the extension of the RPE towards a deferred rendering pipeline. This was accomplished by utilizing the concept of *stages* provided by the OpenSG scenegraph. A stage in OpenSG is a way to configure the scenegraph to render subgraphs into off-screen buffers. These off-screen buffers can be used in later stages to generate the final image for display. Stages can be placed in the scenegraph just as normal nodes (e.g., groups, geometries or transformations). However, unlike normal nodes, stages can control off-screen buffer resources and re-direct the scenegraph traversal to subgraphs in order to fill the off-screen buffers.

We have added stage-based rendering to the fat client framework by enhancing the underlying scenegraph layout as depicted below.



[2] http://www.opensg.org

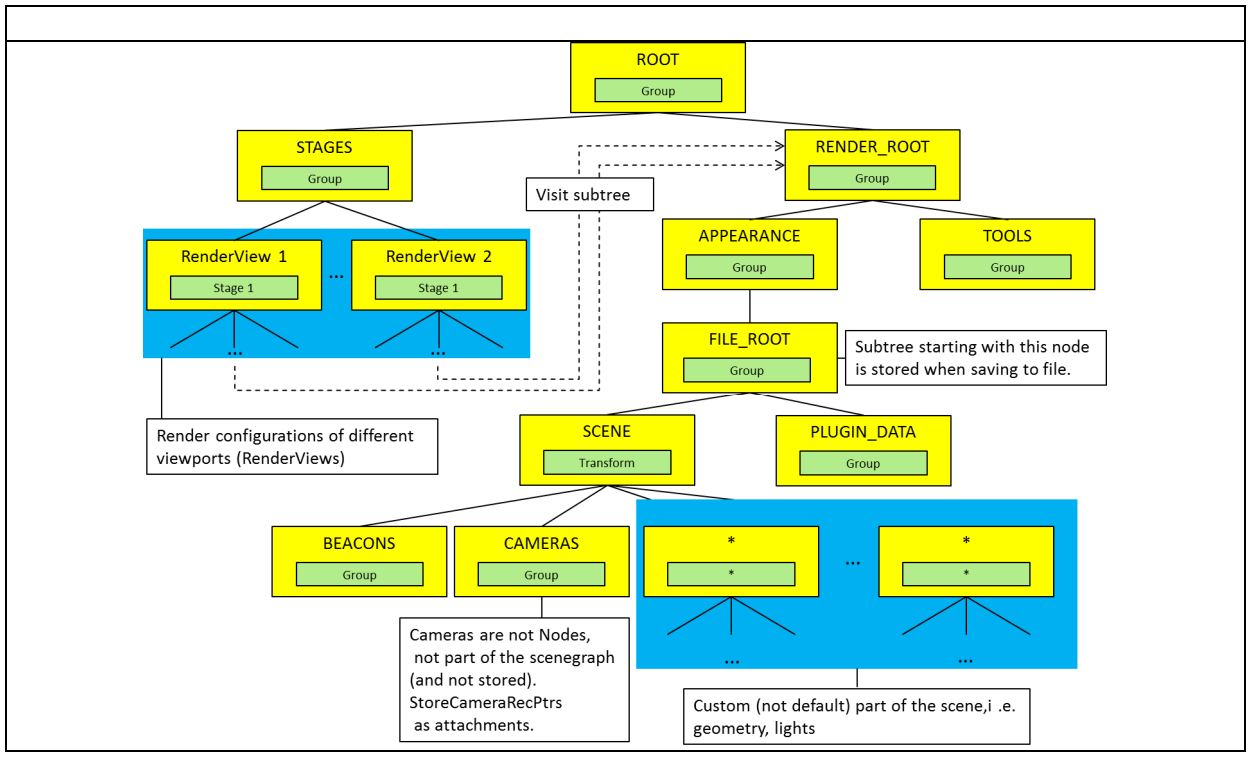


FIGURE 7: FAT CLIENT SCENEGRAPH LAYOUT FOR FEED-FORWARD RENDERING (TOP) AND NEW STAGE-BASED RENDERING (BOTTOM)

The top image in Figure 7 shows the scenegraph layout for feed-forward rendering, which was used before the work in IQmulus started. The scenegraph is traversed directly from the root node and geometry encountered during traversal is directly rendered into the frame buffer. The user-supplied scene geometry is highlighted in blue in the bottom-right part of the graph. The extension to stage-based rendering is shown at the bottom of the figure. Traversal does not start at the root node, but at the stages node. For each viewport (Render View in the fat client jargon) a stage defining the off-screen rendering algorithm is inserted as a child of the stages node. Traversal is re-directed to the subgraph containing the scene geometry. The stage's off-screen buffers are filled during traversal and the final image is generated and written to the frame buffer as a subsequent render pass defined inside the stage.

While this extension is conceptually simple, the hard task was to develop stages that properly control the off-screen buffers and orchestrate the multi-pass rendering. In order to develop a first version of a deferred mapping stage, we have performed several experiments with stage-based rendering starting from relatively simple to more complex off-screen rendering algorithms. Example renderings using the developed stages are shown below:

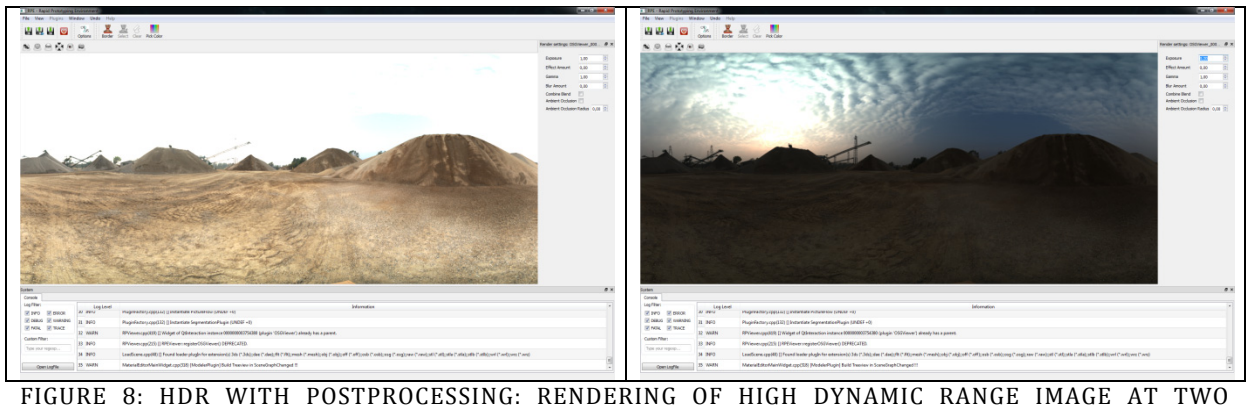


FIGURE 8: HDR WITH POSTPROCESSING: RENDERING OF HIGH DYNAMIC RANGE IMAGE AT TWO DIFFERENT EXPOSURES.

Figure 8 shows a rendering of a high dynamic range (HDR) image (i.e., pixel values are 32-bit floating point values instead of limited 8-bit values) applied to a quad as texture. While HDR rendering is not directly related to IQmulus, this rendering method is a simple example of off-screen rendering and was used to gain first experiences in developing OpenSG stages. The geometry is rendered to a floating point off-screen buffer and tone mapped in a second pass to 8-bit values that are drawn into the frame buffer for display.

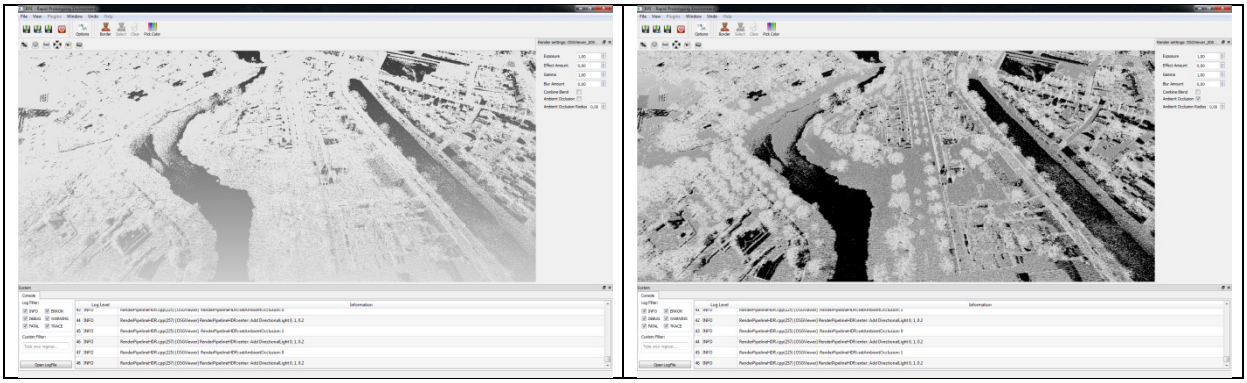


FIGURE 9: DEFERRED RENDERING WITHOUT (LEFT) AND WITH SCREEN SPACE AMBIENT OCCLUSION (RIGHT) OF IQMULUS MIDDELBURG DATA SET PROVIDED BY TUDELFT

Figure 9 shows a first example of deferred shading where the geometry of the Middelburg data set (see Appendix 6.2.2 for characteristics of the data set) is rendered to a G-buffer holding per-pixel positions. The G-buffer is used in a second rendering pass to apply screen space ambient occlusion, a deferred shading algorithm which darkens pixels in screen space if the environment is partially occluded by neighbouring pixels. Please note how the immanent structure of the data set is more visible when applying ambient occlusion (right image) as compared to the uniform lit reference (left image).

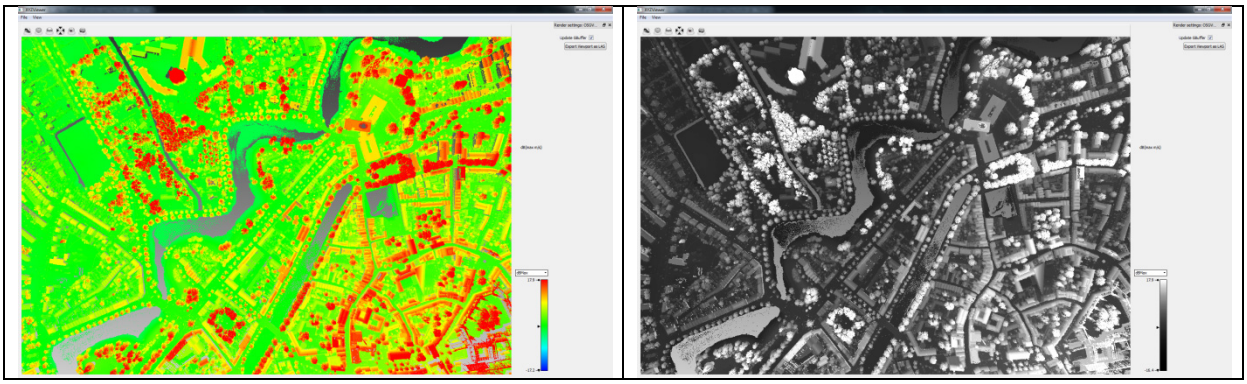


FIGURE 10: DEFERRED COLOR MAPPING OF IQMULUS MIDDELBURG DATA SET PROVIDED BY TU DELFT USING TWO DIFFERENT COLOR MAPS

Figure 10 shows a first result of a deferred mapping algorithm. In this example the Middelburg data set is again rendered to a G-buffer holding per-pixel positions. In a second pass color mapping is applied in screen space. The color mapping stage is further described in section 3.3.

We can state as a preliminary conclusion that the stage-based rendering approach supported by OpenSG is suitable to develop deferred mapping algorithms in the IQmulus context. Please note that the presented stage-based rendering approach is only a preliminary solution, since the stages developed so far are monolithic in the sense that they encapsulate a complete rendering algorithm. We envision a more flexible design where stages are used to generate only partial results of a deferred mapping algorithm where these partial results may be re-used by multiple

subsequent stages. This approach should allow for development of new deferred mapping algorithms by re-using existing atomic stages. However, the work on this design has only started and is part of the work of the next project period.

Data transfer accuracy

A study of the accuracy of the transmission of integer and floating data between the CPU and the GPU was conducted. The reasoning behind this study was the observation that floating point handling was restricted in early programmable GPU implementations. We wanted to verify that transferring data to and from current GPUs does not introduce any data quality degradation. Transferring 32-bit integer and 32-bit floating point data has been tested on the GPUs listed in Table 1.

TABLE 1: GPUS USED FOR DATA TRANSFER ACCURACY VALIDATIOIN

Model		GPU	Release Date
ATI	Radeon HD 4850	RV770	June 2008
ATI	Radeon HD 6670	Turks	April 2011
ATI	Radeon HD 7950	Tahiti	January 2012
NVIDIA	GeForce GTX 260	GT200	June 2008
NVIDIA	GeForce GTX 280	GT200	June 2008
NVIDIA	GeForce GTX 480	GF100	March 2010
NVIDIA	GeForce GTS 450	GF106	September 2010
NVIDIA	GeForce GTX 560 Ti	GF114	January 2011
NVIDIA	GeForce GTX 560	GF114	May 2011
NVIDIA	GeForce GTX 680	GK104	March 2012

The results show that 32-bit integer transfer is working without any problems. Regarding 32-bit floating point values one restriction has been detected regarding denormalized floating point numbers, which are generally flushed to zero.

3.2.2 Deviation of Work and Corrective Actions

There are no deviations from the work plan.

3.2.3 Work Plan for the next Period

The work planned for the next project period focuses on four areas:

1. Refining the integration of internal WP5 components
2. Initial implementation of WP5 external interfaces
3. Refining the Deferred Mapping design and implementation
4. Supporting raster data visualization with modern OpenGL techniques

The WP5 internal interfaces are currently implemented in a prototypical fashion (REST interface, CMR integration). This work will be completed and stabilized early in the next project period. Once the internal REST / WebSocket communication is fully functional, we will tackle the integration with the external IQmulus system (interfaces to data access service and workflow manager). Initial developments of these interfaces are required to enable a first integrated prototype by the end of the next project period.

The deferred mapping stage-based rendering will be refined to provide more flexible re-use of generated off-screen buffers. An example utilizing the next iteration of deferred mapping will be

the efficient integration of color mapping and ambient occlusion by using shared G-buffers. Finally we will investigate further scenarios for providing direct benefit to the end users by utilizing modern OpenGL techniques. One scenario is efficient handling of large raster data by developing corresponding visualization methods around recent OpenGL extensions supporting tiled resources (ARB_SPARSE_TEXTURE, ARB_BINDLESS_TEXTURE). The work in the next project period will focus on 2D raster images, shapes, point clouds and triangular meshes. In the following periods we intend to shift focus towards volumetric simulation data, the corresponding use cases will be defined / refined during the next project period.

3.3 TASK 5.3: GPU-SUPPORTED INTERACTION FOR DECISION MAKING

The challenge here is to support interaction techniques (new and existing ones) by GPU-based methods to generate new levels of interactivity and improved quality for the decision making process. Based on the IQmulus scenarios and user requirements from WP1, in this task we will investigate, design and implement interaction methods to enable the users to:

- Select data to be visualized;
- Look for more details;
- Display changes over time;
- Compare data sets;
- Probe for result values;
- Inspect simulation results.

3.3.1 Work Performed

During the first project period we developed an interactive color mapping method based on the deferred mapping component described in the previous section (3.2.1). The implementation is realized in form of an OpenSG stage that decouples rendering of the geometry of the data set from the color map calculations. In the first pass, the geometry of the data set is rendered into a G-buffer, containing per-pixel positions. In the second pass the color mapping is applied in screen space using the G-buffer positions. The advantage of this approach to color mapping is a significantly increased interactivity in scenarios where the viewpoint (camera position and orientation) does not change. The color mapping can be utilized to visualize scalar values (height, error measures, uncertainty measures) that are available on a per-vertex-basis in the data set. A rendering of a tile of the Fele data set (see Appendix 6.2.1) with interactive color mapped height is shown in Figure 11.

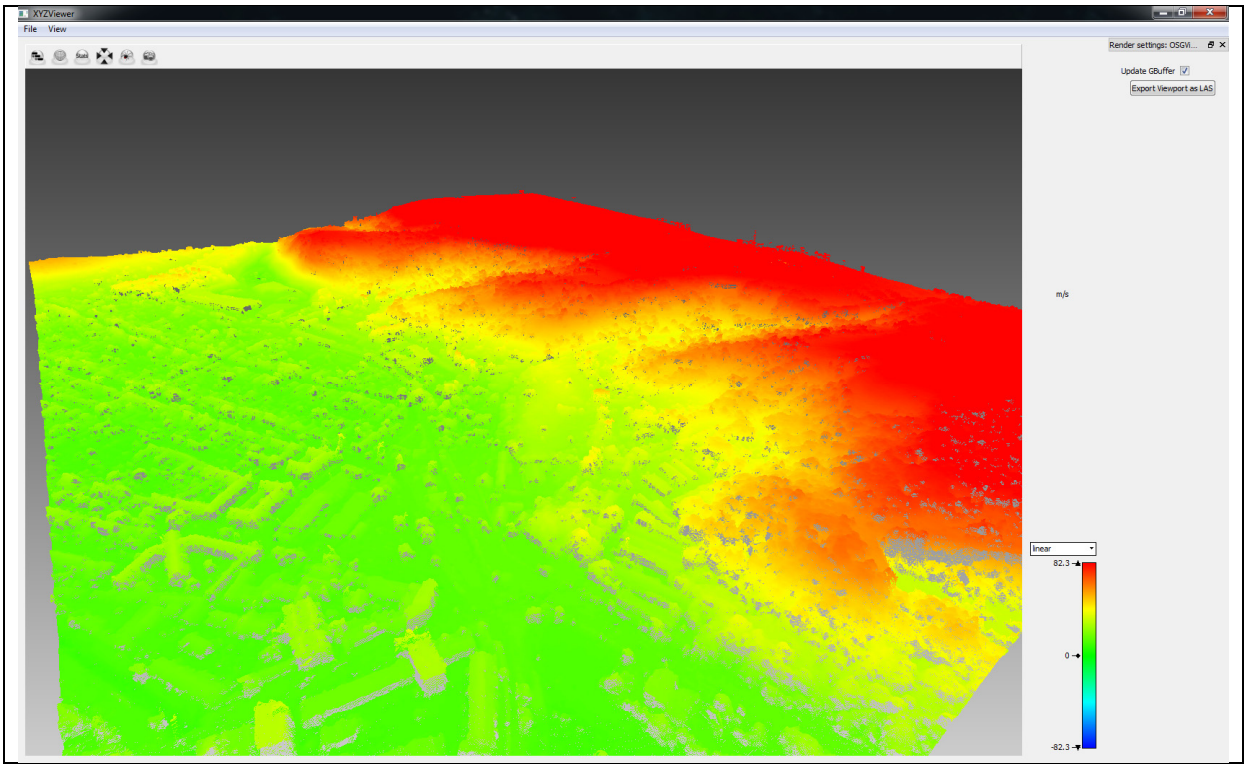


FIGURE 11: TILE OF THE FELE DATA SET PROVIDED BY FOMI WITH HEIGHT COLOR MAPPED USING INTERACTIVE DEFERRED MAPPING

In this example, the G-buffer update is triggered manually (checkbox in the top right corner of the window). If G-buffer update is disabled, color mapping can be changed interactively using the color map widget at the bottom right of the window. The color mapping range can be modified by selecting and dragging the corresponding numbers. Double-clicking into the color bar switches between different color maps.

The deferred color mapping performance was benchmarked on three computers:

TABLE 2: COMPUTERS USED FOR DEFERRED COLOR MAPPING BENCHMARK

	PC1	PC2	PC3
CPU	Corei7 870 2.93 GHz	Core2Quad 6600 2.4 GHz	Core2Quad Q9300 2.5 GHz
RAM	16GB	4GB	8GB
OS	Windows 7 64-bit	Windows 7 64-bit	Windows 7 64-bit
GPU	GeForce GTX 590	GeForce GTX 680	GeForce GTX 260
GPU RAM	1.5GB	2GB	896MB
GPU Driver	311.06	311.06	314.07

In order to validate the performance of the deferred color mapping approach, test data sets were loaded into the fat client and color mapping values were adjusted, first using standard feed-forward rendering and second using deferred color mapping. The test data sets were point cloud files of increasing size, generated by manually combining between one and four tiles of the Fele data set (see Figure 12 and Appendix 6.2.1 for characteristics of the data set). Each tile consists of approximately 14 million points. The rendering speed in frames per second is given in Figure 13.

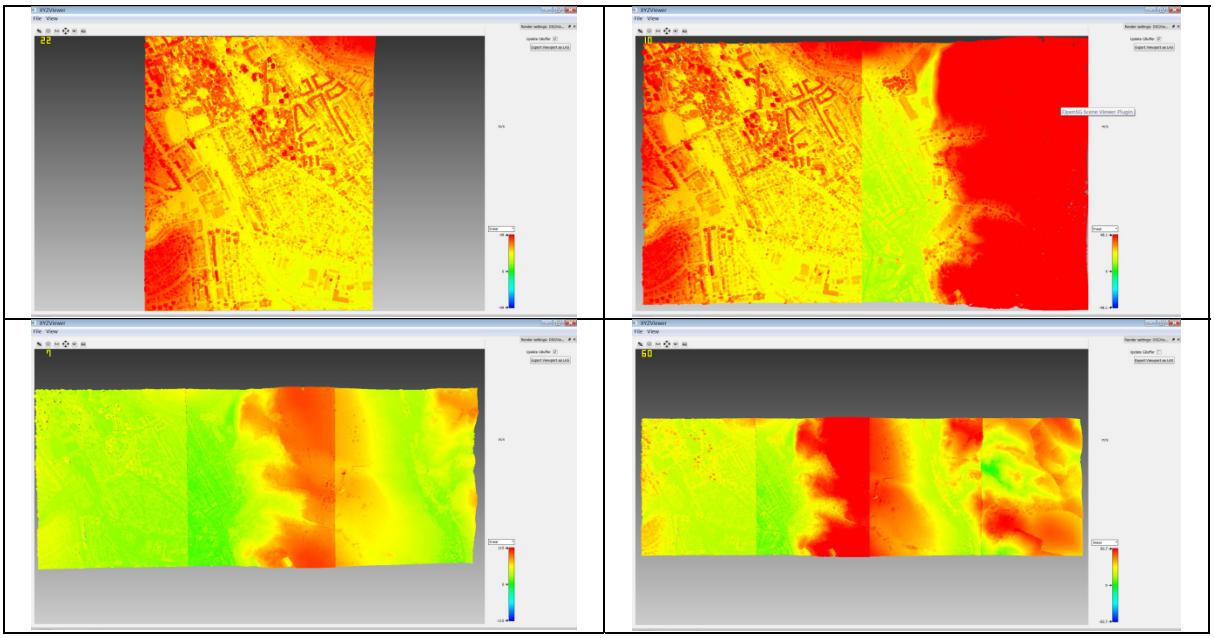


FIGURE 12: TEST DATA SETS: 1-4 TILES OF THE FELE DATA SET

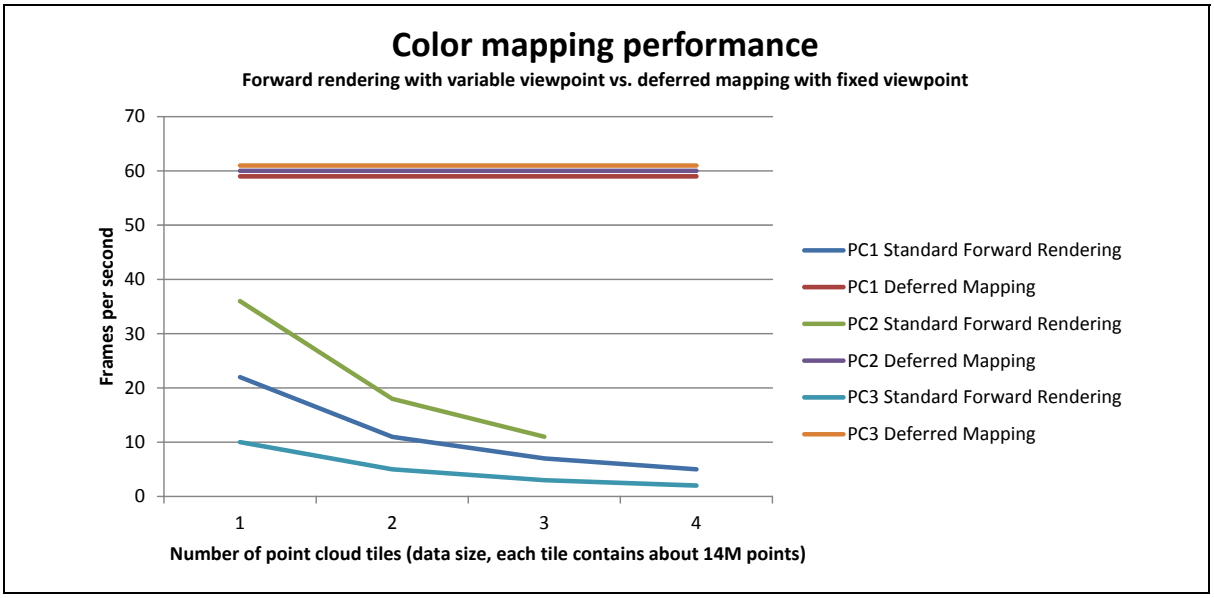


FIGURE 13:DEFERRED COLOR MAPPING PERFORMANCE (60 FPS IN ALL TEST CASES)

Both color mapping methods were tested using a static view (the camera was not moved). In this scenario the deferred color mapping yields always 60fps (monitor refresh rate) independent of the data set size, while the color mapping performance using continuous forward rendering is significantly slower and decreasing with data set size (the 4-tile data set did not load on PC2 due to memory constraints). However, we have to stress that the advantage of our screen space color mapping approach holds only in the scenario where the screen space buffers have been filled once and the camera is *not* moved. Rendering speed during navigation is equivalent to the forward rendering approach when using deferred mapping.

Further ongoing work in this task is an investigation into exporting the fat client's frame buffer as point cloud geometry in order to enable efficient visualization on the thin client.

3.3.2 Deviation of Work and Corrective Actions

There are no deviations from the work plan.

3.3.3 Work Plan for the next Period

The work during the next period will focus on

- Deferred selection;
- Deferred probing;
- Deferred interaction in the flood prediction scenario.

The land showcase defines a flood prediction and monitoring use case (User Story 1.2.2_SC2_1), which includes an interaction centric sub use case (1.2.1_31) related to interactively modifying the predicted height of the water level to see affected land parcels on the cadastre. This sub use case provides a good scenario for exploiting deferred mapping and interaction techniques. We will investigate how to support this scenario by deferred mapping and interaction methods, which may include interactive flooding preview calculated in screen space.

3.4 TASK 5.4: 3D-WEB-BASED VISUALIZATION

In this task the focus is on automatable processes for web application development by designing and implementing a fully automated Web Service Portal, which provides web services that automatically convert data into interactive 3D visualizations for the Web which can be delivered via an hybrid approach, providing both streaming for low-end clients and direct web-based 3D rendering for high-end machines.

3.4.1 Work Performed

An initial specification of a web-based infrastructure to enable high-performance querying and transfer of application specific data and rendering results was generated.

Acceleration structures were introduced to X3DOM (X3DOM is the technology behind the thin client IQmulus visualization framework) in order to render large terrain slices. This feature is needed for client-side web-based visualization of large terrain data sets. Figure 14 shows the triangular mesh of a terrain loaded and displayed with X3DOM. Tiles of the set nearer to the camera are displayed in higher resolution.

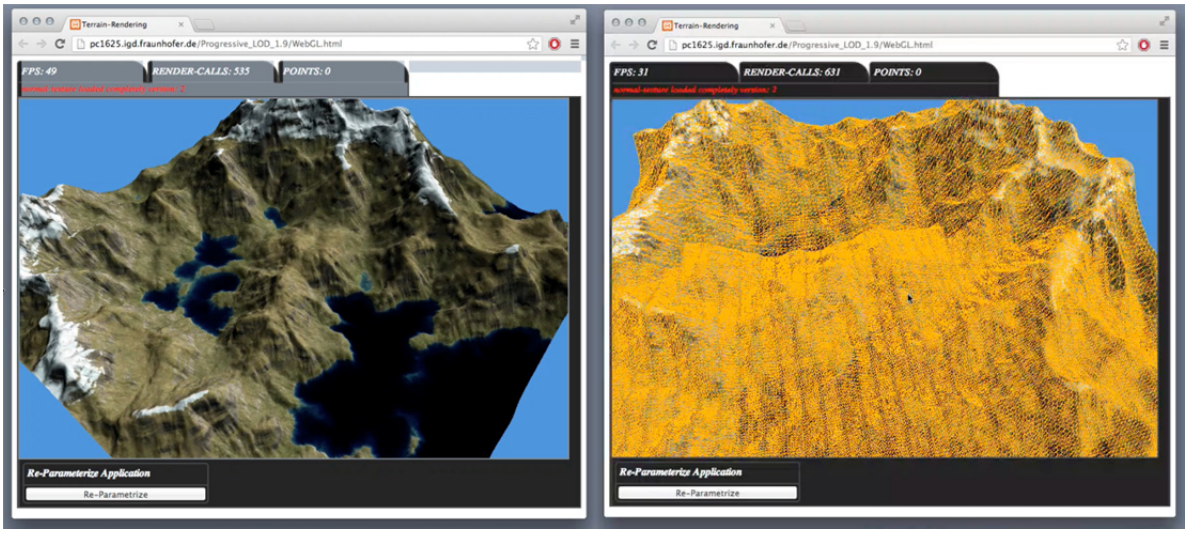


FIGURE 14: A TERRAIN SET RENDERED WITH AN OPTIMIZED BOUNDING VOLUME HIERARCHY (LEFT). THE LEVELS OF DETAIL VARY WITH THE USER'S VIEW, REDUCING THE GRANULARITY OF THE DISPLAYED DATA SET (RIGHT).

A RESTful interface for data communication between applications and services will be used to enable communication between thin clients with limited capabilities, such as mobile devices, and high-end rendering services which can deliver large-scale visualization results. A prototype of the RESTful interface with a hardware library InstantIO was implemented. Results are promising, since several web-applications can now interact with remote hardware usually not available to them. The implementation is now available to the public[3] and will hopefully generate more user suggestions. The RESTful interface specification was published at Web3D 2013 [1].

"Pixi" (a working title for the remote rendering component) needs the ability to stream an image and allow users control of scene elements from outside; therefore, we started a conceptual integration of a RESTful interface into it. InstantReality (the base platform for "Pixi") can now render into separate buffers, enabling the RESTful interface to access and stream them to a remote client, supporting a buffer render pipeline (see Figure 15).

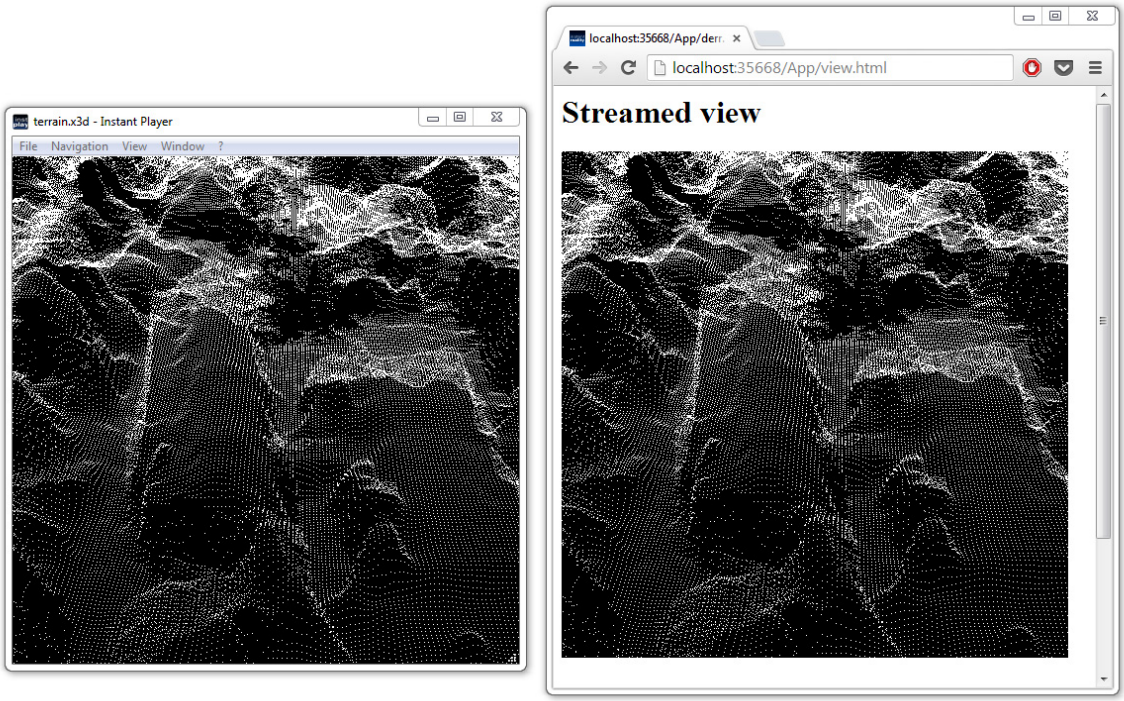


FIGURE 15: INSTANTREALITY (LEFT) STREAMING AN IMAGE TO A WEBSITE DISPLAYED WITH GOOGLE CHROME (RIGHT). THE RENDERER CAN BE LOCATED ON A REMOTE MACHINE AND STREAM TO ANOTHER DEVICE WITHOUT 3D CAPABILITIES. THE LAS FELE DATA SET WAS PROVIDED BY FOMI.

The first transcoder service for internal testing was set up. This part of the HUB architecture allows wildly differing platforms to access the same data. However, hardware-/software-restrictions on each platform may prohibit visualization completely. To avoid this problem, the transcoder service will check and convert WP4 generated data to the best representation for each platform before actual delivery and visualization take place. This might include turning the data into a streamable solution as seen in Figure 15, an X3DOM application as shown in Figure 14 or a hybrid of both, where both instances share part of the workload.

WP4 data can now be handled/loaded by the transcoder service (a testing environment was set up internally at Fraunhofer), which will translate geo data sets into X3D or web-compatible data.

[3] http://www.instantreality.org

Further intelligence has to be implemented to automatically generate data structures when converting huge data sets. The transcoder service was released with a public version of InstantReality, which can now handle limited synchronous jobs in the unlicensed free version. It is yet unclear when to trigger scenegraph and mesh optimizations depending on the input to the transcoder. This is something that will be investigated in the next period.

3.4.2 Deviation of Work and Corrective Actions

There are no deviations from the work plan.

3.4.3 Work Plan for the next Period

In order to run the HUB service and fully transform visualizations of large data sets automatically for user end devices, we need to fully integrate the RESTful architecture as proposed in our publication [1] into InstantReality, the server-side renderer. We can then extract another template for a streamed web-visualisation application, which will deliver the same result as the regular web-visualisation with X3DOM. Having done so, the next step will be to develop a service which accepts large data input from the processing services and decides on the basis of the user's end-device capabilities to start up a remote renderer on a cluster or to transcode and deliver the results as X3DOM data.

Web-visualisation with X3DOM needs more research into data structures which efficiently handle large volumes of data within JavaScript based browser applications. While we have already implemented new modes to deliver 2.5-D landscape data to a client, point-based rendering needs further analysis. Compressing data for web-based visualization is dependent on capabilities in W3C standards and therefore restricted. We plan on integrating results into the transcoder. Depending on the structure of data delivered by WP4, we may furthermore investigate scenegraph and mesh optimizations already built into the transcoder service.

3.5 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.

4 CONCLUSIONS AND CONSOLIDATED PLAN

The first year of activities was devoted to understanding the user needs in order to transform them into technical specifications for guiding the development of the visualization technology. Moreover, the design of the interplay between the different visualization components and therefore the visualization architecture was conceptualized. This process was also supported by a prototypical integration and testing of the expected visualization workflows.

The development of the two software projects of the Code Camp generated promising results in terms of compatibility and integrability of the technologies. Preliminary results demonstrated the potential offered by the visualization technology toward geo-spatial visual decision support. This was a first step toward the development and integration of the different technological components, which assisted the partners in identifying the challenges to be handled during the upcoming period of the project.

Besides supporting the analysis of user requirements from a visualization point of view and designing a prototypical implementation of the visualization architecture, the four tasks T5.1 – T5.4 started to develop first building blocks for the visualization component. For the next project period the main focus of WP5 will lie in the development of new methods based on the framework developed and the experiences gained in year one of the project:

Task 5.1: The focus in the second year will shift to developing new techniques and methods for composite visualization of different data sets (vector, raster, 2.5D, 3D, tensor, etc.). As new visualizations tailored towards the needs of the super user stories are being developed, it will also be necessary to develop multi-resolution techniques to achieve interactivity for even large data sizes. This can include building acceleration structures such as k-d trees, quad/oct trees, and to generate compact representations suitable for modern graphics hardware and the underlying data set.

Task 5.2: The work planned for the next project period focuses on four areas

1. Refining the integration of internal WP5 components
2. Initial implementation of WP5 external interfaces
3. Refining the Deferred Mapping design and implementation
4. Supporting raster data visualization with modern OpenGL techniques

The WP5 internal interfaces are currently implemented in a prototypical fashion (REST interface, CMR integration). This work will be completed and stabilized early in the next project period. Once the internal REST / WebSocket communication is fully functional, we will tackle the integration with the external IQmulus system (interfaces to data access service and workflow manager). Initial developments of these interfaces are required to enable a first integrated prototype by the end of the next project period.

The Deferred Mapping stage-based rendering will be refined to provide more flexible re-use of generated off-screen buffers. An example utilizing the next iteration of Deferred Mapping will be the efficient integration of color mapping and ambient occlusion by using shared G-buffers. Finally we will investigate further scenarios for providing direct benefit to the end users by utilizing modern OpenGL techniques. One scenario is efficient handling of large raster data by developing corresponding visualization methods around recent OpenGL Extensions supporting tiled resources (ARB_SPARSE_TEXTURE, ARB_BINDLESS_TEXTURE). The work in the next project period will focus on 2D raster images, shapes, point clouds and triangular meshes.

Task 5.3: The work during the next period will focus on deferred selection, deferred probing and deferred interaction.

The land showcase defines a flood prediction and monitoring use case (User Story 1.2.2_SC2_1), which includes an interaction centric sub use case (1.2.1_31) related to interactively modifying the predicted height of the water level to see affected land parcels on the cadastre. This sub use case provides a good scenario for exploiting deferred mapping and interaction techniques. We will investigate how to support this scenario by deferred mapping and interaction methods, which may include interactive flooding preview calculated in screen space.

Task 5.4: In order to run the HUB service and fully transform visualizations of large data sets automatically for user end devices, we need to fully integrate the RESTful architecture as proposed in our publication [1] into InstantReality, the server-side renderer. The next step will be to develop a service which accepts large data input from the processing services and decides on the basis of the user's end-device capabilities to start up a remote renderer on a cluster or to transcode and deliver the results as X3DOM data.

Further research into data structures which efficiently handle large volumes of data within JavaScript based browser applications will be conducted. While we have already implemented new modes to deliver 2.5-D landscape data to a client, point-based rendering needs further analysis. Compressing data for web-based visualisation is dependent on capabilities in W3C standards and therefore restricted. We plan on integrating results into the Transcoder. Depending on the structure of data delivered by WP4, we may furthermore investigate scenegraph and mesh optimizations already built into the Transcoder service.

5 PUBLICATIONS

[1] Tobias Franke, Volker Settgast, Johannes Behr and Bruno Raffin. *VCoRE: a web resource oriented architecture for efficient data exchange*. Web3D '13 Proceedings of the 18th International Conference on 3D Web Technology. New York : ACM Press, 2013, pp. 71-78.

6 APPENDIX

6.1 WP5 INTERNAL INTERFACES

During the IQmulus Code Camp, two projects were defined to evaluate the WP5 internal interfaces:

1. Integration of deferred and composite multi-resolution visualization
2. Integration / data transfer between fat client, HUB, and thin client

THE FIRST PROJECT AIMED TO EVALUATE THE POSSIBILITY TO TRANSFER INFORMATION IN OPENGL BUFFERS BETWEEN DEFERRED AND COMPOSITE MULTI-RESOLUTION VISUALIZATION (SEE FIGURE 16 FIGURE 16: CODE CAMP PROJECT 1: INTEGRATION OF DEFERRED AND COMPOSITE MULTI-RESOLUTION VISUALIZATION

). The goal was to:

- generate an OpenGL buffer inside the RPE,
- provide the ID for access to CMR,
- change the buffer content inside the CMR, and
- display the result in RPE.

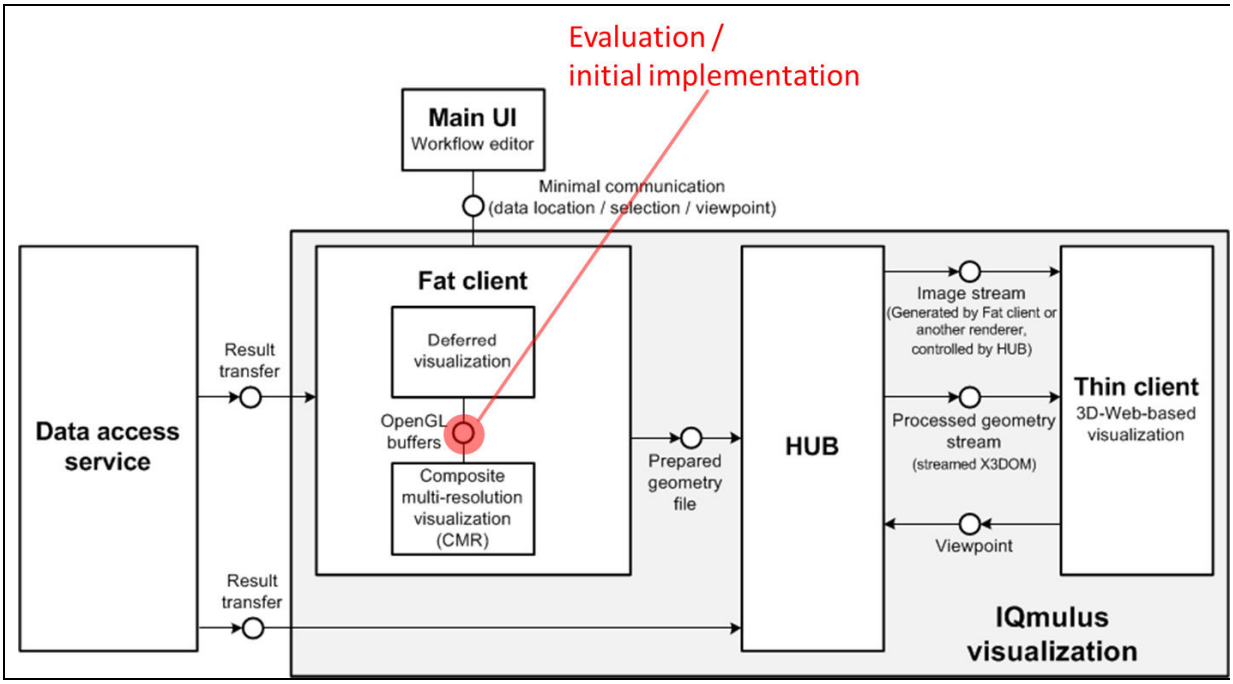


FIGURE 16: CODE CAMP PROJECT 1: INTEGRATION OF DEFERRED AND COMPOSITE MULTI-RESOLUTION VISUALIZATION

The following work has been performed in order to integrate RPE and the CMR visualization library:

- The CMR DLL has successfully been integrated into RPE CMake project, CMR functionality can be called from inside RPE.
- SVN Repository has been created to provide CMR lib updates.
- Basic integration: Render with CMR, display in RPE, basic control of camera. The result is shown in Figure 17.

- More complex interaction between components failed
  - Access to buffer IDs is not possible with current approach
  - Need different approach to integration (via OSGSimpleStage callback)
  - Will be tested / finished as next step

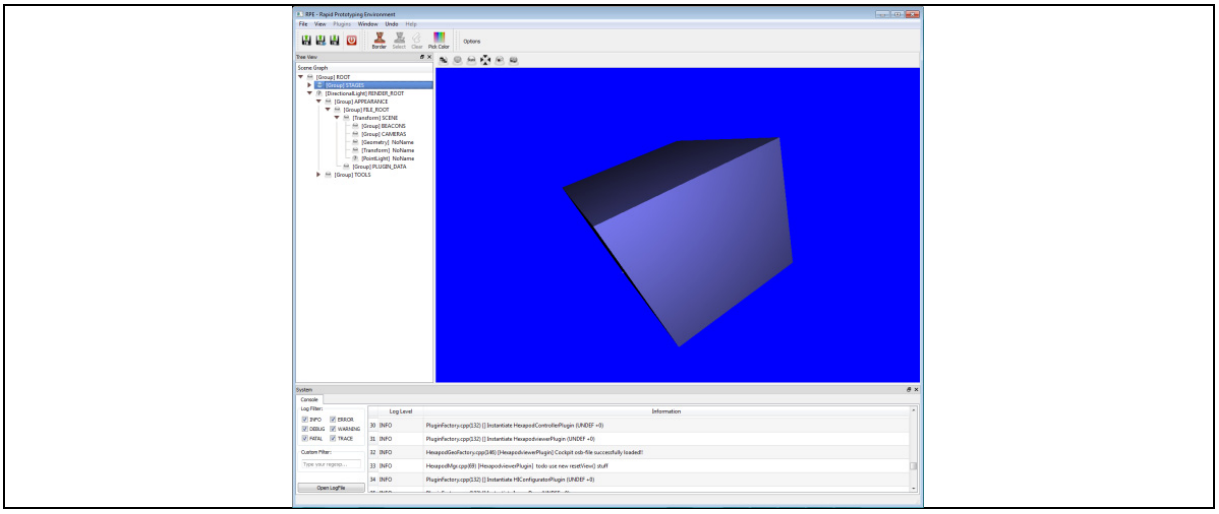


FIGURE 17: RESULT OF CODE CAMP PROJECT 1: AN IMAGE GENERATED BY CMR DISPLAYED INSIDE RPE

The second project was targeted at evaluating the data flow between fat client, HUB, and thin client. For this purpose a file has to be loaded into the fat client, transferred to the HUB, prepared by the HUB for client-side display, and finally transferred to the client, which performs the rendering (see Figure 18).

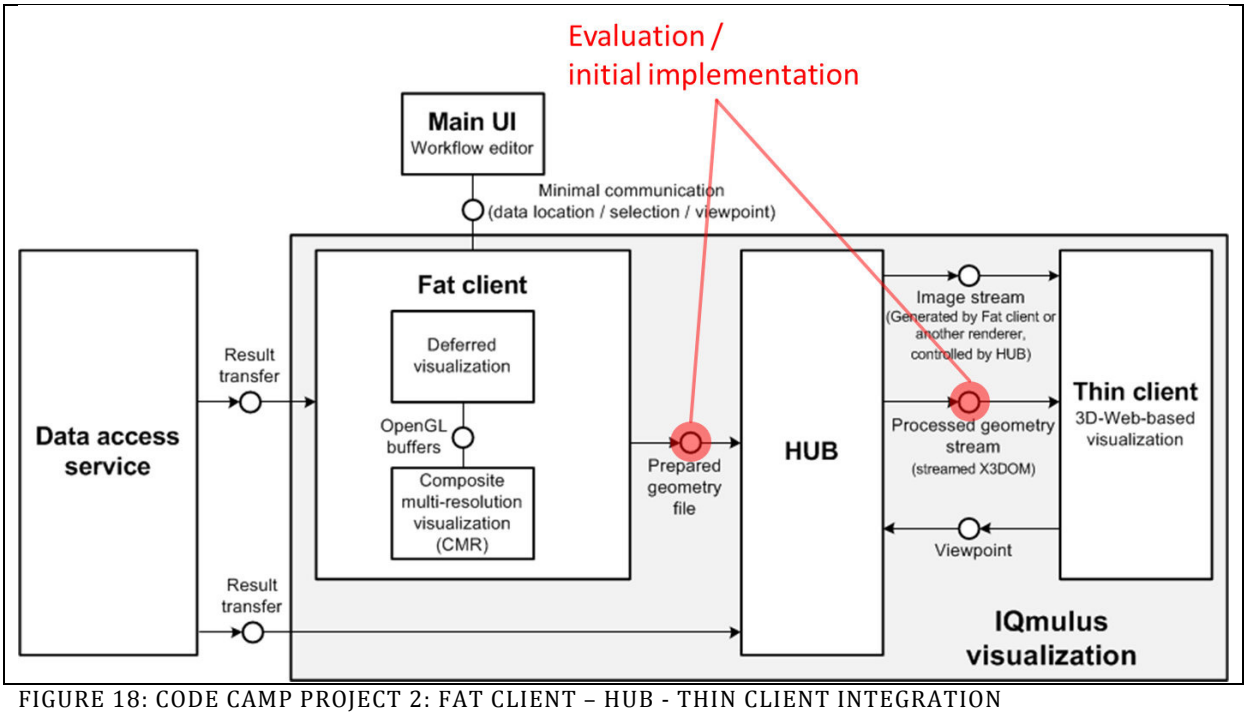


FIGURE 18: CODE CAMP PROJECT 2: FAT CLIENT – HUB - THIN CLIENT INTEGRATION

The project succeeded. A file was loaded into RPE, exported to the OBJ format (*.obj), passed to the HUB, transcoded by the HUB to X3DOM, and finally visualized in the thin client.

Furthermore, the transcoding progress was tracked by RPE. It turned out that writing .obj files is rather slow. Therefore, work on an LAS writer / reader has been performed and the “liblas”

library was integrated into the fat client. Figure 19 shows an example file displayed in fat client/RPE (left) and thin client/3D-web browser (right).

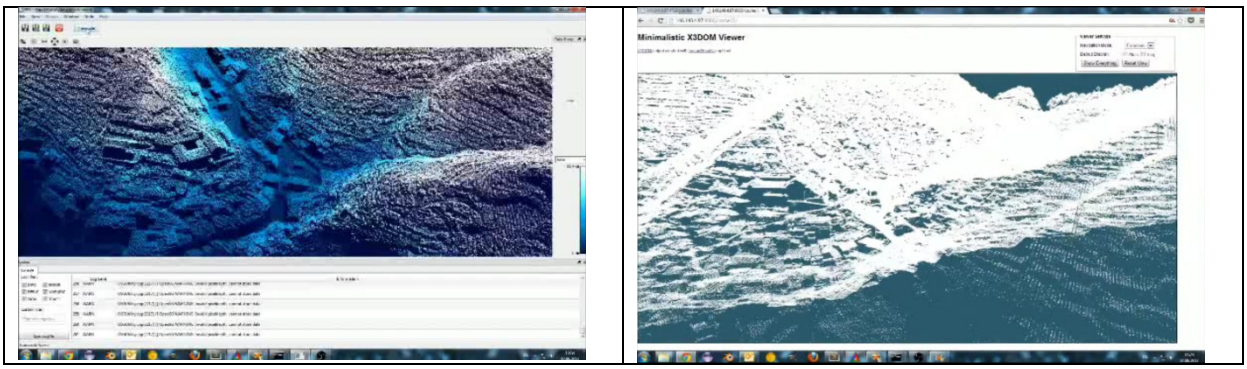


FIGURE 19: RESULT OF CODE CAMP PROJECT 2: ORIGINAL POINT CLOUD IN RPE (LEFT) AND POINT CLOUD TRANSCODED BY HUB IN 3D-WEB BROWSER (RIGHT)

After the Code Camp, deeper integration work has started to enhance the file-based data transfer by direct network communication. This work concentrated on initial integration of WebSocket functionality into the fat client (HERMES network library integration, preparation to provide / use REST interfaces). This work in progress will be completed during the next project period.

6.2 DATA SETS

6.2.1 Fele Data set

Provided by partner:	FOMI
Dataset/service title	Digital Surface Model (DSM)
Country(/ies)	Hungary, Eger
Bounding box: East Bound Longitude	20.41394
Bounding box: South Bound Latitude	47.86223
Bounding box: West Bound Longitude	20.33154
Date of creation	2012-10-18
Spatial resolution: Resolution distance (m)	0.4
Sensor type: Platform category	Airborne
Data characteristics: Data type	Point cloud
Data characteristics: File format	Las
Data characteristics: Reference system	LSR
Attributes	Elevation/Depth
Number of points	8 tiles about 14 million points per tile
Filename:	las_fele.zip First tile: tiled_283257_749349.las

6.2.2 Middelburg data set

Provided by partner:	TU Delft
Dataset/service title	Digital Surface Model (DSM)
Country(/ies)	Netherlands
Location	Historic city center of Middelburg

Bounding box:	1km x 1.25km
Spatial resolution:	14 points per square meter
Sensor type: Platform category	Airborne laser scanner
Data characteristics: Data type	Point cloud
Data characteristics: File format	xyz
Attributes	Elevation/Depth
Number of points	17.5 million
Filename	05r_31000_391250.zip 05_31000_391250.xyz

6.2.3 Vernazza Data Set

Provided by partner:	Liguria
Dataset/service title	LIDAR (XYZ)
Country(/ies)	Italy
Location	Vernazza, Liguria, Italy
Data characteristic: Data type	Point cloud
Data characteristic: File format	xyz
Attributes	Elevation/Depth
Filename	LIDAR_XYZ.zip