Eurographics Italian Chapter Conference (2007)
Raffaele De Amicis and Giuseppe Conti (Editors)

Collaborative Visualization of Sensor Data Through a Subscription based Architecture

M. Witzel & M. Andreolli & G. Conti & R. De Amicis[1], B. De Araújo & R. Jota & J. Jorge [2]

[1] GraphiTech, Salita dei Molini, 2, 38050, Villazzano (TN), Italy [2] IMMI INESC-ID, R. Alves Redol, 9, 1000-29 , Lisbon, Portugal

Abstract

In this paper we propose a collaborative prototype for the integration of Geographical Information System (GIS) sensor data into a distributed architecture for Virtual Reality. Sensor data of various origin and classified domains are provided using a topic subscription metaphor. A graphical editor, which allows creation/import/export/manipulation of sensor data, serves as a real-time administrative interface to control the provision of topics/sensorial data and, at the same time, it is used as an interface to common GIS applications. A client application, responsible for the rendering, decides according to the context which data (timevarying, static) is relevant for the actual examination. Additionally, terrain cultures like forests, buildings, streets along with their geometrical and semantical attributes are provided by using queries to a Web Feature Service/PostGIS Database and translated to a 3D representation. These queries are broadcasted to all clients in the framework to assure at all times synchronicity of the examination data. Finally Session Management is enabled by storing the messages in a relational database to achieve persistency and recalling of the analysis process.

Categories and Subject Descriptors (according to ACM CCS): I.3.2 [Computer Graphics]: Distributed/network graphics I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism - virtual reality H.5.2 [Information Interfaces and Presentation]: User Interface - graphical user interfaces (GUI)

1. Introduction

The integration of data stemming from sensors is currently an emerging field of interest in the GIS (Geographical Information System) domain for it allows the data to be visually analyzed in a 2D/3D terrain context in near-to real-time. Sensor networks/grids form a new source of information if compared to the traditionally static integration of data into GIS systems. Current trends within the GIS domain currently are moving the research and industrial focus towards what are often referred to as LBS or Location-Based Services. LBS allows GPS-enabled mobile devices to georeference ad-hoc data for further processing in distributed systems, be it either immediate or a relayed communication, by using a client-server architecture for further interoperability with additional 3rd party software. Recent improvements on the usability and the increasing computational as well as graphical power of mobile devices allows to create GIS systems capable to use a broad range of devices, ranging from cell phones, PDA's to laptops and tablet PC's. In this scenario the visualization of real-time data and in context-dependent fashion, as in the case of sensors, poses a new challenge to distributed GIS architectures. The work presented in this paper describes an extension to the IMPROVE [SSG*06] project, which aims at the development of a modular Augmented Reality system capable to exploit latest state-of-the-art hardware. The work presented in this paper enables a collaborative scenario for distributed virtual GIS. The paper described how we have successfully created an architecture for distributed sensor observers capable to deliver their observations to the other components of the AR system via XML messages. Among other features the system described allows to visually review sensor data within a real-time 3D collaborative environment, which are continuously integrated and updated through the use of a publish-subscribe approach. The paper also illustrates how the visualization process is based on a structured approach whereby sensor data is structured into domain layers to allow a context dependent access.

2. Related Works

The authors in [KLR*95] presented a real-time immersive 3D system for visualizing geographical data and providing GIS functionality. Most importantly the work combined visualization with the management of the large, complex datasets common to geographical data. As a further step towards delivering an integrated interactive GIS, [CM06] have shown, that the concept of Virtual GIS can be extended towards the direct manipulation of terrain features, creating a dynamic content. However, as illustrated by the authors, this took place in a standalone environment, where content creation and manipulation were separated. To remedy this shortcoming, Ohigashi et al. based their system on 2D legacy GIS applications and demonstrated the feasibility to link it to a 3D system via binding functional capabilities (shadow copy)[OGT06].

Traditionally, GIS systems work with 2D data which are organized into layers. To further bind 2D and 3D GIS systems, the authors in [BW05] developed a hybrid system which allows integration of 2D GIS data layers in a 3D view.

[KD02] presented in their work capabilities to represent vector data for manipulation in a 3D environment by using visualizations directly mapped to the terrain textures. It should be noted that no translation to 3D representations according to the context was undertaken. Shumilov et al. developed a prototype of GIS system which allowed the construction of 3D and 4D models. Time-dependent VRML objects were integrated through the use of TimeSensors, which animated an object by switching among the time-mapped representations it[STCK02]. However it must be noted that the exchange of these data among multiple instances in the previously mentioned works however was not possible.

This problem was efficiently resolved by Haist and Korte, who used a client-server based architecture in order to provide an adaptive delivery of 3D content over a public network[HK06]. As support for workgroup collaboration is becoming increasingly important in the 3D GIS domain, MacEachren[Mac05] extended the concept of Geovisualization to support collaborative activities of users present within the system. Another step towards collaborative analysis of spatial data in a Virtual Reality environment was done by [MTG02], who created a collaborative, interactive and location-independent working environment. Within this context, [GEH03] showed the importance and feasibility of a scalable integration of sensor data and wireless sensor networks in a distributed system whilst maintaining their temporal relationship and thus achieving a distributed, 4D enabled system.

Steed used mobile devices in order to record sensor data like air pollution. The captured data were then integrated at a later time and mapped to overlay texture of the terrain (vector to raster) therefore not providing a real-time, distributed support[Ste04]. The authors of [Sch01] emphasized further the need of a strong terrain and content visualization, particularly in the field of integrating data stemming from remote sensing sources. Complementary work of Walker et al. demonstrated the usage of a visual query language on conceptual and spatial relationships by forming an interface dedicated to non-expert users[WPM03].

3. System Design

The system here presented provides support for collaborative GIS session based on the architecture depicted in fig. 1. The several existing components are inter-connected based on the IMPROVE distributed Mixed Reality/Augmented Reality Design Review architecture. This provides a flexible communication layer named OSGA to share messages between software components and applications.

The core component OSGA (Open Source Groupware Architecture) is a distributed XML message-based integration framework developed in the scope of the IMPROVE project to overcome integration problems. This framework can be easily used for further applications and it is integrated in our system. OSGA is built upon XmlBlaster[xml06] and provides the capacity to have several clients, each of them rendering an instance of the virtual environment, receiving messages accordingly to both subscription and publishing mechanisms. Basically, all the messages sent to the system are redirected to all the clients which subscribed to a given message topic. It also enables to filter the messages for a topic taking in account additional message properties such as sender's identification, or clues about the message content.

In order to participate in collaborative virtual GIS sessions, clients can connect to the system by logging to the OSGA communication backbone with their unique identification. This is composed by username, system ID and the location where the system is running. Most importantly these clients can be themselves GIS sensor data providers or any viewing application which is able to understand parts of the communication protocol. To illustrate our system, we extend the IMPROVE framework with GIS related client applications such as the SensorBuilder, the Web Feature Service interface (WFS) and the IView clients. The SensorBuilder application is a graphical tool capable to provide the management of sensors and able to both administrate data and georeference their location and boundaries. The WFS system provides the interface in order to access PostGIS spatial databases. The clients are the collaborative GIS viewers, capable of high end visualization features, which are able to be connected to the distributed GIS session. On the other hand, all the communication traffic is stored within a relational database provided by another client application of our system named Repository.

To achieve a strong support of higher level terrain visualization, we choose to integrate the Open Source Virtual

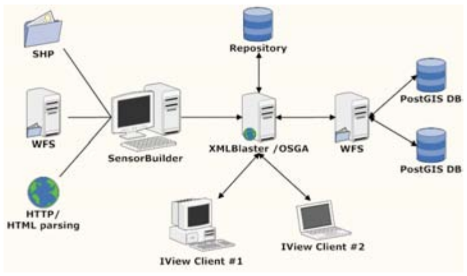


Figure 1: *System Architecture.*

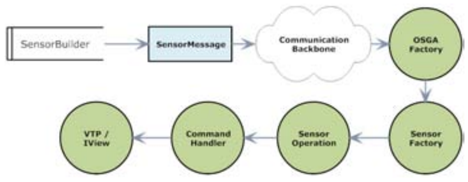


Figure 2: *Dataflow*

Terrain Project (VTP) [Dis06] functionalities into our system. Doing so, the clients take advantage of the rendering techniques like various continuous level of detail implementations and paging of elevation and imagery. Furthermore, through basing the system onto a porting of VTP system, advanced navigation schemes like terrain flyers are made available. In order to do so the visualization process relies on porting of VTP, made by the authors, from its original scene graph OpenScenegraph to the OpenSG scenegraph[Rei]. This choice has enabled our system to have support for clustering of visualization clients for use in conjunction with large scale displays as well as support for advanced stereoscopic see-through glasses as provided by the IMPROVE project research.

4. From Data Generation to Visualization

The exchange of dynamic, time varying data requires specific communication paradigms such as support for asynchronous initialization and release senders as well as the necessity to determine the systems which is the destination of the information. To tackle these issues, we decided to use a message passing middleware (MPM) which follows a publish/subscribe approach. This is the OpenSource Groupware Architecture(OSGA), which is based on XML message definition. In our system, the management of the dataflow coming from the sensor is performed as presented in fig. 2. First, the SensorBuilder generates sensor messages with updated sensor data. These messages are then published to the communication backbone using a channel identified by a *Sensor* topic which identifies the type of message being sent. Finally, the communication backbone server, which relies on XmlBlaster, redirects the messages to all the clients interested to that specific type of sensor data, i.e. any system which has subscribed to that *Sensor* specific topic.

4.1. Sensor Message Definition

The GIS message definition relies on plain XML syntax and defines the sensor information exchanged by all client applications. All messages are derived from a common structure which holds attributes such as author, system origin and its location. This enables session management for multiple users, since it identifies each user, place and system uniquely. This common structure, which is used by all the messages of our IMPROVE communication backbone, has been extended with the sensor-related information, associated sensor operation and a time-stamp. Currently, a sensor is defined by having an unique identifier, the value domain, a boundary influence area, a station name, a sensor domain and most importantly the value itself. However, the XML nature of the protocol will allow a simple extension to introduce new attributes.

When messages are sent, each message gets a time-stamp assigned to it which is important to enable further dissemination of the message or to identify the correct sequence of the work-flow during a given virtual session. Finally the message operation defines the possible actions that our system can perform over sensor data. Specifically these are one of the following types: CREATE_SENSOR to support creation of a new sensor, DELETE_SENSOR to enable deleting sensors and UPDATE_SENSOR to change any attribute of the sensor.

4.2. Communication Initialization

While navigating and interacting with the virtual scene, clients are able to send and receive individual messages containing sensor data to the communication server via a communication channel identified through the relevant topic *'SENSOR'*. Other topic-related visualization and interaction means are also supported by our IMPROVE network backbone in order to enrich the data exploration and the collaborative functionalities. Any given client does not need to know about the number of clients present within the working system. The information exchange is simply done by publishing and processing only the input data associated with a topic. This approach makes clients independent and it increases scalability, reconfigurability, and reuse of components.

However, this approach emphasizes the need for a well-defined communication protocol according to which rendering clients only need to implement parts of the communication protocol which is relevant to them. For the purpose of our work there are two types of topics available: a general communication channel for all sensorial data *'SENSOR'* and a field-specific topic related to the sensor domain, which is provided inside the message content.

This way, any client application, which is used to render a different instance of the environment, knows the presence of sensors through the arrival of data on the sensor communication channel. When a GIS message with a new domain arrives on the client, a new sensor layer is registered on the client system. Each sensor is identifiable through the usage of a Unique Universal Identifier which is created each time a sensor is added to SensorBuilder.

4.3. Handling Sensor Message Exchange

The sending and the creation of sensor messages takes a straightforward approach: once sensor data is added, modified or deleted, a sensor message is created and it holds the description of a sensor and its attributes. Then during the XML serialization, only the sensor object itself needs to be passed to the message object. In order to abstract future extensions of our sensor protocol, all the messaging data use XSD schema definition to automatize message translation.

Regarding the reception of sensor messages, the deserialized sensor information is delegated to a sensor_factory by the client side of our OSGA communication backbone. For each particular message, a sensor_operation is created and then queued on the application event loop as a command to be handled. The operation is processed during the idle time of the application. This ensures safe access to the local OpenSG context in order to allow smooth visualization. All operations are performed in a non-blocking manner. Hence, users do not experience interruptions neither in their interaction experience nor in their work-flow, Allowing smooth navigation and interaction with the scene. All the handling process relies on no knowledge about the sensor data content besides the message topic and its correct operation type which is extracted from the message.

As well as any other communication operation inherited from the original IMPROVE backbone, the sensor operation is determined during the serialization of the message and is bound to a specific application handler through a global reception callback. Multiple handlers, besides the default handlers, can be linked to the command through the use of function binders, specifiable anywhere within the application, even in- place as stateful function objects. Thus, (un)binding of handlers at runtime enables the application to achieve a more complex behavior according to the applications state. This makes it possible for example to enable objects within the application to react individually to a received message. The several components described for the message handling are illustrated in fig. 2.

5. Sensor Data Creation and Management

We use the concept of message exchange in order to allow two distinct approaches: the task of Sensor Observers is basically to retrieve observation values, which are published to the *SENSOR* channel. These data can be considered *raw*, as there is no way of manipulating fields of transmitted data. It is evident, that an additional tool for editing and redirecting sensor messages in near-to realtime fashion is needed to control the data to be inserted into the viewing/interaction component IView.

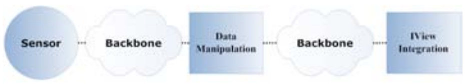


Figure 3: *Managed Data Pipeline.*

As shown in fig. 3, we allow an operator to be inserted into the flow of data which redirects manipulated messages to the *SENSOR* channel.

5.1. SensorBuiler: the Graphical editor

The standard VTP graphical editor, used for viewing and processing geospatial data (VTBuilder, 2D), has been extended by the authors through the development of the so-called SensorBuilder. This tool allows the definition and graphical manipulation of sensor data, boundaries and attributes. In fact the original VTP editor, which is a standalone component to create and manipulate data and achieves, does not allow to communicate at runtime with the client application. Because of the time-dependent nature of sensor data, a more flexible solution was required capable to manage the publishing of real-time information.

Following the architecture of the standard VTP graphical editor we use a layer-based architecture in which data are classified by the user into a specific contexts. Layers are thus assigned a context like water, structures, elevation and imagery. The possibility to create multiple layers and data sets for each domain gives an excellent opportunity to extend this concept to the administration of sensor data.

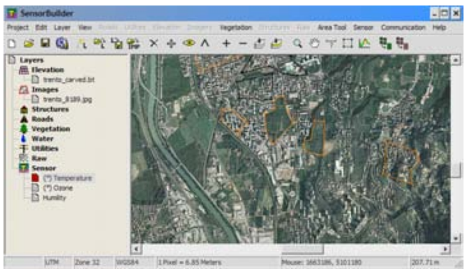
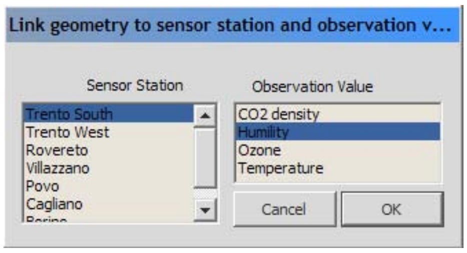


Figure 4: *Sensor boundary creation.*

For each topic, a dedicated layer is created within the sensor Context. This can draw its data from multiple sources. First, the location and boundaries of a geospatial sensor needed to be defined and visualized. This is currently possible either via a WFS request, an import via an ESRI *.shp data or a manual editing in the editor, as shown in fig. 4.

Before proceeding to the graphical representation level, the sensor location or area has to be linked to a time-dependent or static field. Then the selected numerical attribute is continuously requested at specifiable intervals and in case of changes published to a sensor channel along with additional attributes like observation station, measurement unit and geographical position. As mentioned earlier, this component is only aware of the sensor data and it does not need to know about other clients how have subscribed within the distributed virtual session. The strength of building on top of the standard VTP tool is in the support for import and export from/to common GIS formats and software. For example, the authors have showed how an ESRI shape file provided by a local authority can be used in order to use lake boundaries as boundaries for a sensor definition, linked to manually specified observation value. As depicted in fig. 5, the user can choose which observation value and station links to the boundaries of a sensor definition.


Link geometry to sensor station and observation v...

Sensor Station: Trento South, Trento West, Rovereto, Villazzano, Povo, Cagliano, Borino

Observation Value: CO2 density, Humility, Ozone, Temperature

Figure 5: *SensorBuilder station & domain assignment.*

5.2. Managed Sensor Stations

The concept of administrating sensor stations via the SensorBuilder relay application requires us to support two distinct types of messages:

1. *Direct sensor messages (SM)*
2. *Managed sensor messages (SSM)*

After the user logs in to the communication backbone, SensorBuilder listens on the *SENSOR* channel for the presence of sensor stations and sensor domains and it keeps track of changes by storing in a dictionary, for each station, available data sets . In case the user links an observation value, e.g. humility, to a sensor boundary, SensorBuilder acts as a managing component for this particular station #1('Trento South') and the observation chosen value domain. The pair {station, value domain} is then marked as managed on the sensor observer. This is done (fig. 6) via publishing a 'marking message' on the sensor channel which carries the sensor identifier, the observation pair and the operation tag 'MANAGE'. Upon receiving, station #1 will then mark all future outgoing sensor messages related to this particular pair as marked (SMM). These marked, managed messages are then disregarded by all connected clients. However, unmarked observations of station #2 via unmarked messages (SM) are still integrated.

The SensorBuilder thus is assigned the task to send out unmanaged messages, updated with the assigned boundary and/or sensor specification. These are then handled by rendering clients like the data coming from station #2, and integrated.

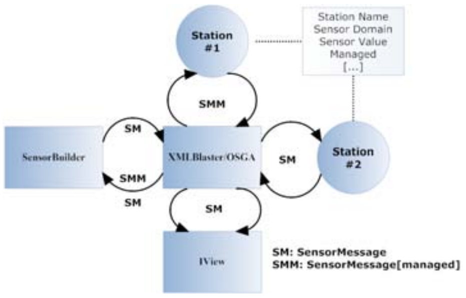


Figure 6: *Managed sensor stations.*

This way, we allow both direct input from a sensor source and additionally supervised, modified input via an administration component.

6. Visualization

In order to bridge the gap between 2D GIS and a 3D interactive component, a set of spatial data described by a sensor domain is integrated using an analogously layer-based approach, with layers providing thematic access to the actual dynamic 3D content.

A sensor manager forms the central management facility for the sensor data and their visualization. It holds an arbitrary number of layers accessible via a dictionary of domains and sensor layers.

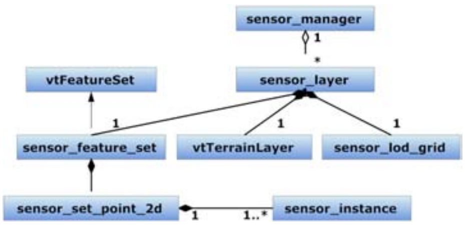


Figure 7: *Sensor Containers.*

The layers consist of a sensor feature set with all sensor data of a specific spatial dimension (fig. 7). Further, it provides basic query facilities like 'find the closest point to' and 'find all points at location'. The application provides the content interface to perform common VTP operations like tagging the underlying feature set with rendering and contextual attributes. Further, a Level-of-Detail Grid gives access to the actual location within the scenegraph. Since VTP implements a level of detail grid (LOD grid) for its terrain positional 3D terrain cultures (like streets and buildings, and vegetation), similarly we have provided a LODGrid for the visualization of sensor data. These are inserted into a LODGrid which determines the according terrain LOD cell, where the sensor geometry is placed onto the terrain. Only content present within the current range of sight dependent on the viewpoint of the user are rendered, thus increasing performance and displaying only content within the current area of interest.

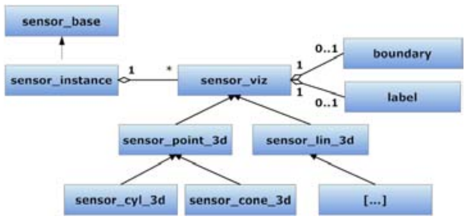


Figure 8: *Sensor class hierarchy.*

Fig. 8 defines the class hierarchy in order to separate data and their visualization. sensor_instance extends the basic sensor data-holding class by providing the means for the visualization and giving access to the sensor sub scene graph. Each visualization component has a label to be displayed on top of the sensor, a boundary geometry and a representation of the sensor value itself. A sensor is enabled to have multiple representations according to the context of the current examination.

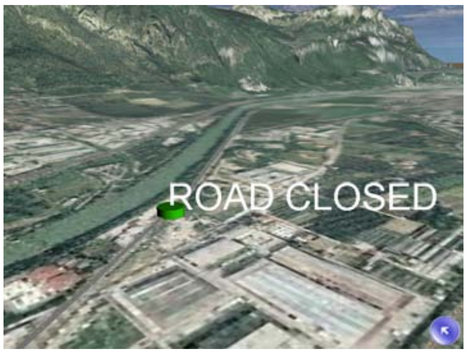


Figure 9: *Generated from http source.*

In fig. 10, a sensor is shown using a mapped color gradient and a height modification to represent a value. Further, the station name is shown always residing on top of the cone. It should be noted that SensorBuilder acts mainly as a managing application. The sensor observers can be distributed as well, for they are simple applications that publish sensor messages to the sensor channel. To show this, an observer which retrieves georeferenced attributes of streets from a website, has been implemented (see fig. 9).

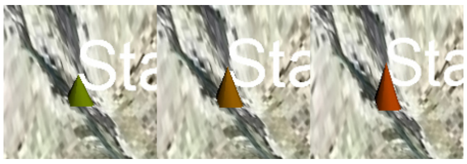


Figure 10: *Update of values.*

7. User Interface

The user interface of rendering client adopted a 2D interaction technique with buttons displayed on the OpenSG overlay. For each sensor layer, a new button is presented with the possibility to turn on/off the visual representation if clicked. The sensor layer buttons are created each time a sensor of a new domain is received and ordered in a stack-wise manner on the right side of the viewport. A ring menu on the top-right corner of the viewport allows to access additional functionality like placing of annotations in the scene which are attached to the present terrain cultures. A minimize button allows to switch between pure navigation and interaction mode.

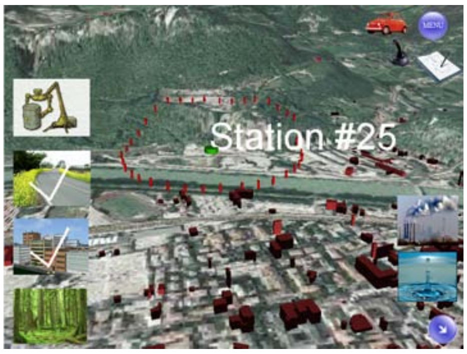


Figure 11: *IView Desktop Interface.*

As a first step towards interaction of sensor data and terrain cultures, the user is given the opportunity to retrieve main terrain features like streets, forestal roads, railways and buildings by composing visually a WFS query. This is possible via a list of available WFS feature sets arranged in buttons on the left side of the overlay interface. Queries to the WFS are currently restricted to selecting a culture type by its button and to performing a bounding box selection of an area. A polygonal selection was disregarded due to the fact that it is only supported by few WFS implementations. Instead the selection of an area is done using a rather simple approach: first the user selects a center point, and then he/she specifies interactively the bounding rectangle of the features to be requested.

According to the type of culture requested, the retrieved *.gml features are translated to native VTP representations (PlantInstance/RoadMap/Building).

The 2D desktop interface technique has been adopted for usage indoors, specifically for usage with a TabletPC in front of a Tiled Display System. Further developments will be done using 3D interaction techniques to enable usage of Head Mounted Displays and to support an Augmented Reality setup.

8. Collaboration

Collaboration is enabled by the exchange of all actions executed by an user, such as e.g. performing queries to a Web Feature Service (WFS). Thus, we created the new communication channel *WFS_REQUEST* and new message definition *WFS_MESSAGE*. A WFS query message carries the base WFS URL, its version, the typename and the bounding box of the query. According to the typename, an appropriate visualization metaphor is chosen to represent e.g. a 2d polygonal area within the query box as a forest for the feature type topp:forest. This ensures synchronicity amongst all instances of the client application, without the need for transmitting particular geometries (that is, the result of the queries) over the communication backbone. The message exchange is simple yet effective: once a query is performed on a client, the according WFS query fields are embedded into a WFS_query message and published to the WFS channel topic. Except for the administrative component (SensorBuilder), the repository and other instances of rendering client receive this message and query the WFS for the geometrical features and their attributes stemming from a PostGIS database. Since this will put heavy load on the WFS, future developments will implement a more efficient approach to include geometrical data in messages and to ensure that the feature data set is only retrieved once.

Due to the similarity to sensor messages w.r.t. handling and exchange, we omitted here a detailed description.

9. Repository

The repository acts as a relational datastore, where all actions of a user, i.e. messages sent out by a client, are stored. It relies on a mySQL database accessed by a dedicated OSGA client. This concept was extended in order to support sensor data and WFS queries exchanged via the communication backbone. In effect, the timely variations of data can be retrieved along with the user action performed during a system session for further analysis.

Contrary to the collaborative handling of WFS messages, the repository only inserts the WFS query string into the relational database. It is intended to use our framework for an incremental refinement of queries. For this reason, our system was freed from storing all geometrical data multiple times.

A further benefit when using a repository is that when a new client arrives, it will allow to recover information in order to be synchronized, both from the data and from the graphical point of view, with the sensor data and other content that is viewed by other users (catch-up).

10. Discussion and Future Work

In this paper, we showed the feasibility of a visual integration of distributed sensor data structure into VR-based client applications.

The main features of the developed system are:

- Distribution of sensor stations.
- Graphical administration of distributed sensor stations.
- Collaborative visualization of sensor data.
- Continuous integration of sensor data.
- Integration of terrain into an immersive environment.
- User friendly, easy-to-use desktop interface.
- 3D contextual visualization of WFS data.

In our system, a unique *SENSOR* channel was created and the types of sensors are only distinguished through an additional attribute. Future developments will focus on how to use dedicated channel topics to transmit data of various domains. Our system uses prototypically defined sensor stations, which are not described appropriately w.r.t. their dimensions and measurements.

Because of this, a further improvement will be the usage of SensorML as a standard XML exchange format for sensor data. SensorML is part of the OpenGeospatial Consortium (OGC) evolving standards for Sensor Web Enablement and allows for a detailed specification of sensors and processes.

An additional research focus will be to scrutinize through a graphics-drive approach the effects of sensor data on terrain cultures which will eventually lead the framework to a Visual Analytics system. As for now, the terrain cultures and the dynamic, time-dependent sensor data are not related to each other. For this reason, a way of composing incremental visual queries needs to be found.

The system setup was dedicated to the use of desktop systems, using a 2D interaction metaphor. In order to allow more advanced, direct interaction techniques, the GUI will be adopted for immersive systems yielding a Augmented Reality setup. Hence, a distributed outdoor / indoor scenario with users working jointly on an analysis will be made possible.

11. Acknowledgements

The authors want to acknowledge the EU IST programme for funding the IMPROVE project.

References

[BW05] BROOKS S., WHALLEY J. L.: A 2d/3d hybrid geographical information system. In *GRAPHITE '05: Proc. of the 3rd international conf. on Computer graphics and interactive techniques in Australasia and South East Asia* (2005), ACM Press, pp. 323–330.

[CM06] CLARK R. W., MAUER R.: Visual terrain editor: an interactive editor for real terrains. *J. Comput. Small Coll. 22*, 2 (2006), 12–19.

[Dis06] DISCOE B.: The virtual terrain project. (VTP) `http://www.vterrain.com`.

[GEH03] GANESAN D., ESTRIN D., HEIDEMANN J.: Dimensions: why do we need a new data handling architecture for sensor networks? *SIGCOMM Comput. Commun. Rev. 33*, 1 (2003), 143–148.

[HK06] HAIST J., KORTE P.: Adaptive streaming of 3d-gis geometries and textures for interactive visualisation of 3d city models. In *9th AGILE International conf. on Geographic Information Science.* (2006).

[KD02] KERSTING O., DOELLNER J.: Interactive 3d visualization of vector data in gis. In *GIS '02: Proc. of the 10th ACM international symposium on Advances in geographic information systems* (2002), ACM Press, pp. 107–112.

[KLR*95] KOLLER D., LINDSTROM P., RIBARSKY W., HODGES L. F., FAUST N., TURNER G.: Virtual gis: A real-time 3d geographic information system. In *VIS '95: Proc. of the 6th conf. on Visualization '95* (1995), IEEE Computer Society, p. 94.

[Mac05] MACEACHREN A. M.: Moving geovisualization toward support for group work. In *Exploring GeoVisualization* (2005), Elsevier, pp. 445–461.

[MTG02] MANOHARAN T., TAYLOR H., GARDINER P.: A collaborative analysis tool for visualisation and interaction with spatial data. In *Web3D '02: Proceeding of the seventh international conference on 3D Web technology* (2002), ACM Press, pp. 75–83.

[OGT06] OHIGASHI M., GUO Z.-S., TANAKA Y.: Integration of a 2d legacy gis, legacy simulations, and legacy databases into a 3d geographic simulation. In *SIGDOC '06: Proc. of the 24th annual conf. on Design of communication* (2006), ACM Press, pp. 149–156.

[Rei] REINERS D.: Opensg 1.8. `http://opensg.vrsource.org/`.

[Sch01] SCHEEPERS F.: Landscape visualisation. In *AFRIGRAPH '01: Proc. of the 1st international conf. on Computer graphics, virtual reality and visualisation* (2001), ACM Press, pp. 49–52.

[SSG*06] STORK A., SANTOS P., GIERLINGER T., PAGANI A., PALOC C., BARANDARIAN I., CONTI G., DE AMICIS R., WITZEL M., MACHUI O., JIMÉNEZ J. M., DE ARAÚJO B. R., JORGE J. A. P., BODAMMER G.: Improve: An innovative application for collaborative mobile mixed reality design review. In *Virtual Concept 2006* (2006). `http://www.improve-eu.info`.

[STCK02] SHUMILOV S., THOMSEN A., CREMERS A. B., KOOS B.: Management and visualization of large, complex and time-dependent 3d objects in distributed gis. In *GIS '02: Proceedings of the 10th ACM international symposium on Advances in geographic information systems* (2002), ACM Press, pp. 113–118.

[Ste04] STEED A.: Data visualization within urban models. In *TPCG '04: Proc. of the Theory and Practice of Computer Graphics 2004 (TPCG'04)* (2004), IEEE Computer Society, pp. 9–16.

[WPM03] WALKER A. R., PHAM B., MAEDER A.: A framework for a dynamic interactive 3d gis for non-expert users. In *GRAPHITE '03: Proc. of the 1st international conf. on Computer graphics and interactive techniques in Australasia and South East Asia* (2003), pp. 283–284.

[xml06] Xmlblaster. `http://www.xmlblaster.org/`.