<table>
<thead>
<tr>
<th><strong>Project number:</strong></th>
<th>288577</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project acronym:</strong></td>
<td>UrbanAPI</td>
</tr>
<tr>
<td><strong>Project title:</strong></td>
<td>Interactive Analysis, Simulation and Visualisation Tools for Urban Agile Policy</td>
</tr>
<tr>
<td><strong>Instrument:</strong></td>
<td>STREP</td>
</tr>
<tr>
<td><strong>Call identifier:</strong></td>
<td>FP7-ICT-2011-7</td>
</tr>
<tr>
<td><strong>Activity code:</strong></td>
<td>ICT-2011.5.6 ICT Solutions for governance and policy modelling</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Start date of Project:</strong></th>
<th>2011-09-01</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Duration:</strong></td>
<td>39 month</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Deliverable reference number and title (as in Annex 1):</strong></th>
<th>D3.2 User Interface Elements Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Due date of deliverable (as in Annex 1):</strong></td>
<td>30-11-2012</td>
</tr>
<tr>
<td><strong>Actual submission date:</strong></td>
<td>see “History” Table below</td>
</tr>
<tr>
<td><strong>Revision:</strong></td>
<td>3</td>
</tr>
</tbody>
</table>

**Organisation name of lead contractor for this deliverable:**
Fraunhofer IGD

**Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)**

<table>
<thead>
<tr>
<th><strong>Dissemination Level</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>PU</td>
</tr>
<tr>
<td>PP</td>
</tr>
<tr>
<td>RE</td>
</tr>
<tr>
<td>CO</td>
</tr>
</tbody>
</table>
Title:
User Interface Elements Documentation

Author(s)/Organisation(s):
Jens Dambruch / Fraunhofer IGD, Michel Krämer / Fraunhofer IGD, Jan Peters-Anders / AIT, Ernst Gebetsroither / AIT

Working Group:
WP3 – Fraunhofer IGD, AIT

References:
Description of Work, D2.1, D3.3

Short Description:
This document specifies the user interface elements of the UrbanAPI applications. It also includes some technology background and architecture of web based applications, to explain the benefits and challenges in our approach.

Keywords:
User Interface, 3DSC, GSM, portal, urban growth simulation, web application

History:
<table>
<thead>
<tr>
<th>Version</th>
<th>Author(s)</th>
<th>Status</th>
<th>Comment</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>Jens Dambruch</td>
<td>New</td>
<td>Table of contents created</td>
<td>5.11.2012</td>
</tr>
<tr>
<td>002</td>
<td>Jens Dambruch</td>
<td>RFC</td>
<td>Reworked basic structure of document and listed of components identified in 3DVR</td>
<td>20.11.2012</td>
</tr>
<tr>
<td>003</td>
<td>Ernst Gebetsroither</td>
<td>RFC</td>
<td>Added urban growth simulation</td>
<td>18.12.2012</td>
</tr>
<tr>
<td>004</td>
<td>Jens Dambruch</td>
<td>RFC</td>
<td>Included suggestions of Maria de Santiago</td>
<td>18.12.2012</td>
</tr>
<tr>
<td>005</td>
<td>Jan Peters-Anders</td>
<td>RFC</td>
<td>Added Motion Explorer</td>
<td>18.12.2012</td>
</tr>
<tr>
<td>007</td>
<td>Jens Dambruch</td>
<td>RFC</td>
<td>Rework after review</td>
<td>9.1.2013</td>
</tr>
<tr>
<td>008</td>
<td>Ernst Gebetsroither</td>
<td>RFC</td>
<td>Rework after review</td>
<td>10.1.2013</td>
</tr>
<tr>
<td>009</td>
<td>Jan Peters-Anders</td>
<td>RFC</td>
<td>Rework after review</td>
<td>11.1.2013</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>010</td>
<td>Michel Krämer</td>
<td>Final</td>
<td>Added more software architecture diagrams and a more complete description of the software architecture</td>
<td>1.3.2013</td>
</tr>
<tr>
<td>011</td>
<td>Jens Dambruch</td>
<td>RFC</td>
<td>Update of document to revision 3 reflecting new developments: Updated figures, screenshots and component descriptions in chapter 3</td>
<td>28.7.2014</td>
</tr>
<tr>
<td>013</td>
<td>Ernst Gebetsroither</td>
<td>RFC</td>
<td>Update of document to revision 4 reflecting new developments: Updated figures, screenshots and component description in chapter 3.3</td>
<td>29.08.2014</td>
</tr>
<tr>
<td>014</td>
<td>Jens Dambruch</td>
<td>Final</td>
<td>Final rework after review</td>
<td>2.9.2014</td>
</tr>
</tbody>
</table>

**Review:**

<table>
<thead>
<tr>
<th>Version</th>
<th>Reviewer</th>
<th>Comment</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>006</td>
<td>Chiara Caranti, Zaheer Khan</td>
<td>See Review Form of Deliverables, UrbanAPI QA_D3.2 CC.doc submitted on 8.1.2013</td>
<td>8.01.2013</td>
</tr>
</tbody>
</table>
About this Document ........................................................................................................ 6

1 Introduction .................................................................................................................. 6

2 Software architecture and technology ......................................................................... 10
   2.1 Notation .................................................................................................................. 10
   2.2 Data Administration components ............................................................................. 12
   2.3 Web portal ............................................................................................................... 13
   2.4 Motion explorer application ..................................................................................... 14
   2.5 3DSC application ..................................................................................................... 15
   2.6 Urban development application ............................................................................... 17
   2.7 CityServer3D ......................................................................................................... 17
   2.8 ViewHUB services ................................................................................................. 18
   2.9 UrbanAPI Web Application ..................................................................................... 19
      2.9.1 Web servers and web portals ............................................................................. 19
      2.9.2 Web application frameworks ........................................................................... 21
   2.10 Summary ............................................................................................................... 25

3 User Interface Elements ............................................................................................... 26
   3.1 User Interface Elements for the 3DSC application .................................................. 26
      3.1.1 Content Display and basic control elements ....................................................... 29
         3.1.1.1 2D Map ....................................................................................................... 29
         3.1.1.2 3D Display .................................................................................................. 30
         3.1.1.3 Export Image from 3D-View ....................................................................... 31
         3.1.1.4 Navigation mode change ............................................................................ 32
      3.1.2 Scenarios ........................................................................................................... 32
         3.1.2.1 User Management ....................................................................................... 32
         3.1.2.2 Scenario Selection ..................................................................................... 33
         3.1.2.3 Sharing Scenario ....................................................................................... 34
         3.1.2.4 Save Changes ............................................................................................. 35
         3.1.2.5 Feedback for a scenario as a whole ............................................................ 35
      3.1.3 Object manipulation ......................................................................................... 37
         3.1.3.1 Object selection in 3D ................................................................................. 38
         3.1.3.2 Move .......................................................................................................... 38
         3.1.3.3 Re size ........................................................................................................ 40
         3.1.3.4 Delete object ............................................................................................... 40
         3.1.3.5 Deselect object .......................................................................................... 40
         3.1.3.6 History records ........................................................................................... 40
         3.1.3.7 Object integration (upload) ....................................................................... 41
         3.1.3.8 3D Interaction options and object palette (Street furniture) ....................... 42
      3.1.4 Annotations ....................................................................................................... 43
         3.1.4.1 Annotation Details ....................................................................................... 45
      3.1.5 XML Download ................................................................................................. 46
      3.1.6 Free Drawing Tool ............................................................................................ 47
      3.1.7 E-mail notification ............................................................................................. 47
      3.1.8 Route planning .................................................................................................. 48
      3.1.9 Visual compare .................................................................................................. 51
About this Document

This document specifies the user interface elements of the UrbanAPI applications. According to the description of work for UrbanAPI [1] it is the deliverable D3.2 – User Interface Elements Documentation of work package WP3: Software Platform Development: Database, Integration, Visualisation and Simulation Tools. This also includes some technology background and architecture of web based applications, to explain the challenges and benefits in our approach. This is the second improved version of the document which is updated in the following sections:

In chapter 3 all figures and screenshots are updated to show the final elements of the GUI. Descriptions of the elements are updated accordingly. Components are revised to reflect also the findings encountered during development and evaluation.

1 Introduction

Building on the requirements [2] gathered from participating cities in work package WP2 the objective of work package WP3 is to develop application prototypes to support ICT governance of cities.

The UrbanAPI tools will provide advanced ICT-based intelligence in three urban planning contexts. First, directly addressing the issue of stakeholder engagement in the planning process by the development of enhanced virtual reality visualization of neighbourhood development proposals. Second, at the city-wide scale, developing mobile (GSM - Global System for Mobile communications) based applications that permit the analysis and visual representation of socio-economic activity across the territory of the city, and in relation to the various land-use elements of the city. Finally, developing simulation tool applications in the city-region context addressing multiple challenges in responding to the simultaneous demands of both expanding city populations for certain European cities, and declining and frequently ageing populations elsewhere.

The first of these applications is the 3DSC application. The description of work for the UrbanAPI project outlines the application as follows:

Maps – as 2D-visualizations of proposed changes in an urban environment – are often judged as too abstract by lay persons, who do not have the experience required to create a mental representation of a future environment. Describing the general effects and the depicting the visual impact of urban development plans as realistically as possible with the help of 3D scenarios will support the negotiation process for urban development projects. Virtual and augmented reality visualizations depict the consequences of spatial planning processes in a virtual or real world environment. Interactive control of planning interventions and – on the fly – presentation of the new visual effects released through changes in zoning and finally building help the citizens to experience these impressions. The 3D expression is interpreted rather as architectural presentation of the urban development, which do not exhaust the citizens, interests and policy making participation. In addition to the architectural image perception people is hardly interested in a lot of additional urban parameters, such as:

- Transport capacity and transport bottlenecks;
- Air, noise, chemical pollutions;
- Urban services availability (i.e. medical, police, school, communications);
- Disaster exposure and disaster prevention.
In this context the UrbanAPI shall present the parallel capability of multiple coded layers presentation in complex with 3D presentation of buildings and green zoning. All those virtual parameters presentation shall be available both for actual infrastructure and for multiple planned scenarios of key urban parameters’ layers combinations. Virtual representations of planning decisions seem to be the most convenient and understandable solution for presenting spatial planning alternatives to the public. Allowing interactive modifications of alternatives helps stakeholders to understand the proposed actions and to endorse the anticipated impacts. This deployment of the UrbanAPI system will employ a relatively coarse client-side simulation model, but requires high quality 3D geodata and rich interaction elements, especially to provide feedback on planning in various forms. For this scenario, the 3D web client as well as the mobile app client is to be used. ([1], page 11)

The second application is the Motion Explorer (on example of the Vienna application):

The objective of the Urban Motion Explorer is, to develop an interactively controllable application to make use of GSM mobile device traffic data to explore diurnal movement patterns as well as temporal population distribution pattern in the urban built environment and - to some extent- in open spaces.

The goal will be fulfilled by developing an easy to use tool which transforms the data delivered by mobile phone service providers into spatio-temporal representations by means of web-based dynamic maps linked with topographic and planning related information. Spatial joining of the mobile device distribution peaks and sinks with various urban characteristics allow planners as well as the public to explore the relationships between adoption of urban infrastructure and urban design or public space quality. Detecting correlations will help to improve the urban space, making it more attractive places for the population and allow more efficient usage of infrastructure. This will help stakeholders to explain their interests through action without verbal communication.

The user interface cannot cope with “original – individual mobile phone user data but requires pre-processed mobile phone location data to take into account privacy concerns and error correction of input data.

Pre-processing work refers to:

1. Identification of the users’ “home raster cells” by exploring night and early morning cell occupation
2. Extraction of all single trips per day
   + sorting of users with home raster cell within case study areas
   + sorting of uses’ trip sections by time stamp and location coordinates
   + correction of false locations within a trip because of redirection of cell tower connections to other towers if the nearest cell is lacking connection capacity.
      - check of virtual high speed trips due to cell tower redirection
      - re-location of the users to the “previous cell tower” with new time stamp.
3. Aggregation of users with valid trips by raster cells
4. Generating dynamic origin-destination matrices for time slices
   + aggregating users by origin raster cells
   + aggregating target cells by the users of each origin cell during a day
5. Feasibility study: Extrapolation of user numbers to population totals per origin cell
+ estimation of mobile phone user-inhabitant ratios per origin raster cell serving as extrapolation quotas for all destination cell totals
+ extrapolation of user totals to inhabitant totals
+ validation of extrapolation results through referencing the extrapolated numbers to population totals of (a) the entire population, (b) population counts e.g. along traffic routes

The public motion exploration features to be applied are:

1. Visualisation of the population distribution pattern - over all raster cells in time steps
2. Visualisation of the populations raster cells’ occupation – through diagrams per raster cell
3. Dynamic interaction matrix exploration
   (a) Visualisation of population motion pattern of single source raster cells exploring the target raster cells per time step
   (b) Visualisation of population motion pattern of single target raster cells addressing all exploring the source raster cells per time step
4. Aggregated results for Vienna
   - day-time interaction matrix (aggregated time steps)
   - night time interaction matrix (aggregated time steps)

Figure 1: Population distribution dynamics in the Vienna Region during morning hours (Oct 18, 2011) shown through A1 mobile phone user distribution patterns (Source: data A1, processing: AIT - Austrian Institute of Technology GmbH)
The third application is the Urban Development Simulation (UDS):

City region development simulation helps to understand the large scale consequences of spatial planning decisions in a complex urban system, including representation of socio-economic activity across the territory in relation to various land-use elements of the city. Interactive control of proposed planning interventions and associated impacts generated by these interventions assists the public interactively engaging in the planning processes and contributing to planning decisions. Detailed and easy understandable information about planning decisions and full transparency about the expected impacts will support the negotiation activities during a participatory planning process and will finally increase public commitment to these decisions. ([1], page 13)

The main focus of this application is on support for local urban planners to evaluate certain planning decisions. Simulation results from different planning scenarios for the future development will be visualized and distributed via internet for a broader public discussion to support a participatory planning process. Therefore it is aimed to distribute certain scenario results from the UDS in 3D format via web servers and a web portal, see section 2.9.1 and 2.9.2.

In chapter 2 the basic architecture and technology aspects of these applications are explained, focusing on web based technologies. Chapter 3 explains the actual user interface elements derived from the use cases and requirements (see [2]) and how these are composed to make up a web-application.
2 Software architecture and technology

In this chapter we define the software architecture of the UrbanAPI applications and the technology that we use to implement them. The description of work ([1], pp. 46) outlines several aspects regarding what shall be achieved within this project beyond the state-of-the-art. What is new in respect to 3DSC, visualization and simulation can be summarized as the integrated approach using emerging new web-based viewing technologies.

The UrbanAPI toolset will be capable to process large amounts of rich 3D models, but also little and coarse 3D information from standard CAD and GIS data bases and selected elevation views (photos, renderings) as texture information. This will allow easy and widespread applications, by providing a simple and intuitive graphical user interface (GUI) to introduce alternative planning scenarios. Data will be provisioned through web-based services which can be tailored to application-specific requirements. In order to support simulations, web services will deliver data also based on time and different planning scenarios. In comparison with ongoing projects that also plan to develop ICT toolboxes for policy modelling such as OCOPOMO, PADGETS and IMPACT, UrbanAPI focuses on data integration and harmonisation and the delivery of seamless 2D/3D AR/VR applications. ([1], page 18)

The main focus of the architecture described here lies on the web technology that is used to implement the use cases defined in the requirements documentation [2] that are targeted to a broad audience including non-domain experts such as citizens of a city or other city departments. This approach gives novel opportunities for public participation and a more direct communication channel to communities that share some special interest such as preserving cultural heritage in the city.

The technology designated for this will be detailed in the following sections. The major goal is that the people should have an easy accessible opportunity to get involved directly and interact with the scenarios.

We start with describing the different components of the UrbanAPI tool set, ranging from administrative tools to the web portal that will be used by stakeholders such as domain experts, decision makers or citizens. We then describe existing technologies that we use to speed up the development of the UrbanAPI tool set. Finally, we give an overview over existing third-party web-frameworks that we investigated in order to choose the right one for the UrbanAPI web portal.

2.1 Notation

The figures presented in this section have been created using the “Fundamental Modeling Concepts” (FMC) block diagram notation (http://www.fmc-modeling.org). This notation has originally been defined by the Hasso-Plattner Institute in Potsdam, Germany. It is now maintained by Intervista AG in Potsdam. It can be used very well to describe complex software architectures with a simple set of symbols and rules. FMC has been used in the industry to describe the architecture of several commercial software products including SAP R/3 (cf. http://www.fmc-modeling.org/fmc-and-tam). Basically, an FMC diagram consists of the following elements:

- **Stickmen**
  Active human actors
- **Rectangles (boxes with straight edges)**
  Active components that serve a well-defined purpose (e.g. controllers, web services, etc.)

- **Boxes with round edges**
  Passive systems, components channels or storage (e.g. files, databased, communication channels)

- **Arrows and connecting lines**
  Access type, read or write (arrows) or both (lines)

- **Circles**
  Communication channels with a directed request direction

A more complete description of all elements that can be used in an FMC block diagram is depicted in Figure 2.

---

**Figure 2: Description of elements in a FMC block diagram. Source:**
http://www.fmc-modeling.org/notation_reference, last visited 2013-03-01
2.2 Data Administration components

The UrbanAPI description of work [1] defines three applications:

- **The Motion Explorer application** deals with mobile device data. This data is analysed to infer information about how citizens move through the city at certain times of the day.

- **The 3DSC application** deals with visualising existing data sets (such as city models, digital terrain models, etc.) as well as simulation results in a scenario-oriented manner.

- **The urban development simulation** simulates the effects of structural change in cities.

The three applications deal with heterogeneous data that has to be integrated and processed in different ways (see deliverable D3.5 “Data integration components”). We therefore develop three different software tools to manage the various data sets. Each of them is tailored to one application only. Compared to a single generic solution, this allows us to implement specialised administration features for each application without complicating the user interface too much.

Note that the tools described in this section are for data administration only. They are supposed to be used by data administrators and expert users. Besides, we also develop tools for the end-users. They will be described in the sections following this one.

Figure 3 shows the three data administration tools that we develop. Data collected in the **Motion Explorer application** is managed by the mobile device data administrator. He or she interacts with the mobile device data toolset in order to process mobile device data, to perform analyses and to store this data as well as analysis results into a database. The mobile device database keeps this data so it can later be used by the mobile device data service (see section 2.4). The user interface for the mobile device data toolset is described in section 3.2.

In order to speed up development of UrbanAPI tools, for the **3DSC application** we make use of the existing software CityServer3D. This software has been developed by the Fraunhofer IGD for several years and is already used in practice in several municipalities. In figure 3 we therefore marked existing components with a grey background color. The CityServer3D is described in more detail in section 2.7. To summarise, the CityServer3D consists of a server application and a client tool (AdminTool). The client can be used by data administrators to import data from different data formats, to edit it and to finally send it to the CityServer3D server. The server keeps a connection to a database where it can save 3D geodata.

In the UrbanAPI project we extend the CityServer3D by several new features that allow expert users to create scenario-specific web apps (cf. the term “scenario” described in section 3.1.2). The expert user can upload apps to the CityServer3D. They will then be displayed in the web portal. The whole process is described in section 2.5. The UI elements that will be added to the CityServer3D AdminTool are described in detail in section 3.1.

The **urban development simulation** does not need a special service to access the database. It consists of a rich-client desktop application that can be used to import urban growth data, to process it and to store it into the database. The database stores this data so it can later be visualised in the urban growth web app (see section 2.6).
2.3 Web portal

In the previous section we described the three data administration applications developed in UrbanAPI. The end-users access the data that is stored in the three databases through an integrated web portal that runs in the user’s web browser. An end-user can be one of the following:

- Domain expert working at the municipality
- Decision maker (for example heads of municipal offices, mayors, etc.)
- Other stakeholders such as citizens

Figure 4 shows the web portal that contains several “portlets”. More information about web portals and current state-of-the-art technologies in this respect can be found in section 2.9.

In the following we use the generic term “app” instead of the more implementation-specific term “portlet”. The web portal consists of several apps serving different purposes. For example, there are apps for the motion explorer application (section 2.4), for the 3D VR application (section 2.5) and for the urban growth application (2.6). Apart from that, we also use some general purpose apps such as user management, web content, etc. These apps are typically pre-defined by the web portal framework (see section 2.9).
The web portal and the apps can save arbitrary information in a common database. For example, a 3D VR app may save the current camera position in this database, so the user can close the browser window and later return to exact the same position. We call this kind of information “state” of an app.

The apps can also communicate with each other. For example, the urban development app generates simulation results that should be visualised in 3D. It therefore sends the information to the 3D VR app. Modern web portal frameworks (as they are described in section 2.9.1) support the Java portlet specifications JSR 168, JSR 162, JSR 167, and JSR 286 which already define APIs for portlet development and inter-portlet communication.

In the following sections we describe the three applications and how we integrate their web apps into the UrbanAPI portal. We use a service-oriented architecture in order to achieve low coupling. According to the GRASP software development patterns this helps to separate concerns and to enable good reusability of the single components.

### 2.4 Motion explorer application

Besides the motion explorer administration toolset described above we also develop a motion explorer web app that will be included in the UrbanAPI web portal (see figure 5). The motion explorer app displays results generated by the mobile device data processing service. Its user interface elements are described in section 3.2.

The app communicates with the 3D VR app in order to display processing results in 3D. The motion explorer app loads its content from a web service called “Content service”. This service provides data from both, the mobile device data processing service (that is administered with the mobile device data toolset described in section 2.2) and the CityServer3D (see section 2.7).

The motion explorer app can access the “Converter service” which is able to convert various file formats to a web-compatible representation. For example, the app sends its processing results as JSON objects to the converter service. The service then converts them to X3D, a format that can easily be displayed by the X3DOM framework (http://www.x3dom.org, see section 2.9.2) that we use in the 3D VR app. It sends the X3D data back to the app which then sends it to the 3D VR app which displays it on the screen. The converter service will be described in deliverable D3.7 “Processing components documentation”.

---

*Figure 4: The web portal consists of several apps (portlet). It can store state and other information in a common database.*
The 3D VR app may also make use of existing, static “background” data such as digital terrain models, 3D city models, etc. The app loads such information through the content service which gets the data from the CityServer3D (see section 2.7). Since data like 3D city models and terrain models can be quite huge, a special optimisation step is needed so WebGL-enabled web browsers can handle it. We therefore make use of the existing technology “ViewHUB Transcoder” which is described in section 2.8. In the UrbanAPI project we will extend the ViewHUB Transcoder service whenever needed.

2.5 3DSC application

Within the web portal the 3D VR related application user interface components are deployed as scenario-specific applications. These applications make use of the following services (see figure 6):

- **Converter service** to support uploading of 3D models. For example, in the Vienna application (see [2]) architects should be able to upload their own 3D models and to integrate them into the 3D visualization that has been provided as a 3D VR app by the municipality. In order to support a wide range of file formats, the 3D VR app may send the file to the converter service which converts it to X3D, the format needed for the X3DOM-enabled web visualization.

- The **Content service** provides data for the 3D visualization. Other than the ViewHUB Transcoder service it provides non-optimized data. Such kind of data is needed if a high level of interactivity is required. Optimized models often do not provide such interactivity as they are highly compressed and the optimizer has likely changed their structure significantly.

- The **ViewHUB Transcoder Service** is an existing component that can provide 3D data that is highly optimized for display in the web. The service is described in section 2.8.
The Content service and the ViewHUB Transcoder Service get input data from the CityServer3D which saves its geodata in its own database. It can be managed by expert users through the AdminTool (see section 2.2). The AdminTool is also used by the expert users to define scenarios (cf. Section 3.1.2). Such scenarios are used to integrate all necessary data to create an application-specific 3D visualization. The AdminTool transfers the scenario to the CityServer3D which saves it in its database.

The ViewHUB app producer is a component that is currently developed in cooperation with the UrbanAPI project. It creates the scenario-specific 3D VR apps that are added to the web portal. It therefore requests HTML templates from the App service. These templates define how a scenario-specific 3D VR app looks like. Templates contain placeholders that will be filled with the data defined in the scenario that has been stored in the CityServer3D’s database and that has been created by the expert user in the AdminTool. The HTML templates are pre-defined by the system's developers, but since they are separated from the App service's code, they can easily be customized.

As described above, the web portal has access to an Application database. It is used to store the apps’ state as well as other related data. For example, most of the applications described in [2] require the possibility to enter feedback for the presented visualization. For example, citizens should be able to comment on urban plannings. This feedback is also saved in the application database, so the end-user can later access it and edit it.

The web portal provides user and rights management. Each end-user will get access to its own data stored in the application database. The urban planners who provided the 3D visualization and who want to get feedback from the end-users will be able to access the information of all users in the application database.

Figure 6: The 3D VR apps are scenario-specific. They are created by the App Producer. They make use of several web services.
2.6 Urban development application

For the urban development application we develop three apps (cf. figure 13):

- The **urban development app** is the main app in this respect. It has access to the urban growth database that has been filled with urban growth simulation results (see section 2.2 above). The app displays these results in a textual, tabular or graphical representation. It also makes use of the converter service (just like the motion explorer app, section 2.4) to convert simulation results to a format compatible to the web visualization (i.e. X3D). The converted data is then sent to the 3D VR app that displays the results in 3D.

- The **3DSC app** can display simulation results received from the urban development app in 3D. It integrates this data in an existing 3D scene that is provided by the content service and the ViewHUB transcoder service as described in section 2.5.

- The **LimeSurvey app** makes use of the existing Open-Source framework LimeSurvey to provide a survey for the simulation results currently displayed by the urban development app and the 3DSC app.

All these components are described in detail in section 3.3.

![Diagram](image.png)

*Figure 7: The urban development app uses the LimeSurvey portlet and the 3DSC app to display surveys and to render 3D visualizations of the current simulation results respectively*

2.7 CityServer3D

The CityServer3D technology is the basis for the management of spatial data and processing within the applications to be developed. On the web site [3] the CityServer3D is described as follows:
The technology of the CityServer3D consists of a geo-database, a server with numerous interfaces for the import and export of the data and applications for the development of landscape models. An administration software allows to process the data and the web viewer takes them by Internet onto the user's screen. [...] The CityServer3D can be used to manage 3D geo information. It offers access to geodata via standardized interfaces. When connected to already existing data infrastructures, the CityServer3D helps you to efficiently manage your 3D city models. The integrated continuation processes grant a sustainable use of your information. Integrating multimedia content can positively affect the presentation of your geodata. The CityServer3D can be used for urban planning, simulation and data analysis.

The CityServer3D and the corresponding AdminTool are essential elements in the 3D content delivery and content preparation stage. Regarding to Figure 3, the CityServer3D will be the host for the 3D-city-model and other spatial data like digital terrain models. For the data storage a standard database like MySQL [4] can be used. CityServer3D also offers standard web services to other applications. An example can be a spatial query for 3D-data for display, conversion of various data formats to a common representation or export of data to a specialized format like ESRI-shapefiles.

The setup of scenarios and the data preparation should be done with a highly specialized tool, that is available to administrative users only. Typically these users have a sound understanding of GIS technologies and are domain experts. To craft such a scenario editor is one key aspect that will be elaborated during the UrbanAPI project lifetime.

Some more detailed information on AdminTool and CityServer3D is available in the deliverable 3.3, Rule User Interface Documentation [5] and on the web site [3].

2.8 ViewHUB services

The ViewHUB Transcoder shown in Figure 6 will be the link between 3D-web-visualization and the models in CityServer3D or other sources. In order to be useable the 3D-web-visualization needs to be responsive and visually compelling. The 3D geographical data in the servers is typically not directly suitable for this as datasets may be too large, too detailed or in a format that is not very efficiently usable in a web application. As of this the ViewHUB Transcoder is responsible to optimize the 3D-data for the web display. The ViewHUB is a new approach to prepare and deliver web based 3D applications and a topic of research at Fraunhofer IGD. The project is in an early stage and the development of the UrbanAPI application prototypes will drive the ViewHUB developments by its requirements.

The data sets that will be provided to the 3D-display are optimized for best experience and viewing performance. Several aspects of optimization include restructuring of the declarative 3D content like building walls, roofs etc., compression and streaming of content. Also caching of static content in scenarios can be done here as most of the 3D data will not be subject to change frequently. This also reduces the load on the CityServer3D.

The ViewHUB Application Producer is a template based system to provide web applications. In short this service has a set of templates for application use cases that will be parametrized with scenario and optionally user specific data.

This component is another aspect of research and developments in the project. Based on the results of the web application prototypes elaborated using the technology explained in section 2.9 an easy to use and high performance automated application generator shall be conceived and implemented as a prototype.
2.9 UrbanAPI Web Application

A web application is an application that is accessed by users over a network such as the Internet or an intranet. The term may also mean a computer software application that is coded in a browser-supported programming language (such as JavaScript, combined with a browser-rendered markup language like HTML) and reliant on a common web browser to render the application executable. [6]

Today web applications can be considered the state-of-the-art for occasional users. Internet access is available to nearly everyone and the use of web browsers is common to more and more people. Social networks, blogs, wikis and other web applications pioneered some new ways for public participation on a broad and cost effective level. On the other hand the developments that enable high performance 3D games together with the availability of 3D APIs for browsers offer new possibilities to create exciting new interactive applications also for expert users with demanding 3D graphics requirements to the software.

Applications that needed special software and special hardware some time ago, nowadays can be run in a standard web-browser on an inexpensive computer.

2.9.1 Web servers and web portals

The basis for a web application is a web server that delivers the contents to the client browser and even smartphones. In the context of the UrbanAPI applications we recommend using a “web portal” that also includes server components but also has many ready to use components to build a complete application.

The web portal shall support the following requirements:

- It is easy to create working prototypes
- It is easy to use with respect to the use cases and project requirements from D2.2 [2]
- Must fit the technologies described here that we want to use
- Is open source or free-of-charge for development and later operation
- Shall have good support and a sound community
- Must be customizable
- Shall provide user management solution
- Support for different languages and internationalization

Considering these requirements we have several products to choose from:

- JEE Portal (Application) Servers
  Offer integration of small modules (Portlets) and customization on end user level. For example:
  - Liferay
  - JBoss Portal
- Content Management Systems (CMS)
  As the name says the system supports content management but is targeted more for editorial uses. For example:
  - Drupal
Hand crafted by using elementary tools like Tomcat

All of the features above could be covered by the Liferay portal [7] as the web infrastructure. As of that we will use Liferay as the platform for our prototypes. The basic technology behind Liferay is Java EE. There are also several implementations of this standard to choose from: as working hypothesis we use the reference implementation Apache Tomcat (see [8]), that also is available as a bundle together with Liferay. User related data like alternative scenarios and their annotations to the scenarios will be stored in a separate application database, that is also accessible via standard clients and uses standard formats. We use MySQL for this.

A page with interactive 3D contents created with X3DOM (see also next chapter) could for example be integrated with a special portlet. Even out-of-the-box portlets that are shipped with a standard installation of Liferay can be used, like an iframe¹ portlet² as shown in Figure 8. Also components for user management, content management, wiki, forum and blog are available in this software. The technical evaluation within the project will then assess the usability of this solution based on the requirements from [2].

---

1 Iframe: This HTML construct enables to integrate HTML pages directly into another page.
2 Portlet: A module in a portal application. A portlet can be arranged on a page and can serve several purposes. For example a portlet can be a blog, a forum or just simple HTML content display. These modules can be customized and configured by the users and give lots of possibilities for personalization of a portal.
It is clear, that one can achieve the best results by completely custom developments, but on the other hand experience shows that not reusing existing solutions is an expensive choice and needs a sound understanding of all technical details involved. So taking into account the limitations imposed by the project budget and the time frame of the project it is advisable to rely on proven application toolkits or frameworks. Apart from that, the existing solutions mostly provide other useful features like management tools or other components like wikis or blogs that a custom solution cannot offer as an additional benefit.

2.9.2 Web application frameworks

Having chosen a platform for the web portal, there is still an open point to decide. There are several ways how a web application can be implemented. From the bottom up perspective HTML-pages can be created as files and served with a web server. On the other hand sophisticated high level frameworks as JSF can be used.

The portal platform chosen in the previous section leaves a lot of opportunities and we have to select a suitable application toolkit. Direct use of Servlets/JSP and JavaScript libraries is very low level programming which makes the further development and maintenance very expensive and tedious and we want to have a cleaner and aesthetically compelling approach for this.

The communication with the server should be as simple as possible. A handcrafted AJAX solution cannot provide this. Also we don’t want to mix many different languages like HTML/XML, JavaScript, Java, PHP,... for defining our application as this complexity makes the template mechanisms in the ViewHub very complicated too. Also we want to use proven advanced rich client functionalities as in the GoogleWebToolkit [9]. A possible approach would be to use a Model Driven Architecture that generates artifacts on the technical level from a high-level domain model. This would require to set up a domain model for the code generation which should offer means to describe basic user interface elements on an high-level; for example a text field that accepts only certain types of input like numbers or date selection boxes. An abstraction on that level is already available as web application frameworks. A rough overview is available in [10]. So it is sensible to use these instead of doing this another time.

From the vast possibilities compared in [10] we chose the vaadin framework as our working hypothesis. Vaadin is a rich internet application framework that offers high-level constructs to programmers, which enables a programming model that is closer to GUI software development rather than a HTML/JavaScript based web development. Behind the scenes vaadin is using GoogleWebToolkit to generate JavaScript and HTML for the realization of high-level constructs. For the communication with the generated client side artifacts with the server vaadin also generates AJAX encapsulations so that the programmer needs not to deal with this and can work on a server-side application model, which simplifies development. Additionally the use of Vaadin and GoogleWebToolkit encapsulates most of the peculiarities of different web browsers and therefore the programmers need not to adapt their developments to those.

The main features of vaadin (excerpt, a complete list can be found on the web site of vaadin [11])

Comprehensive Component Framework

- A large set of user interface components, controls and widgets
- Rich and interactive widgets with lazy-loading
- Drag and drop support
- Mobile touch event support
• Data binding using MVC (model-view-controller)
• Build layout
• Create new components with composition and inheritance

Web Compatibility
• Based on Google Web Toolkit (GWT)
• No browser plugins needed
• Browser window and tab support
• Back button support
• Deep-linking support
• URL parameter and fragment handling

Customizable Look and Feel
• Powerful CSS based component styling
• Good looking built-in themes and styles
• Build custom application themes
• Embed to any web page

Java Web Development
• Java-only, object-oriented web development
• Powerful server-side programming model
• Simple single-Jar deployment

Secure Web Application Architecture
• Server-side state management
• Application code runs in the server
• Secure parameter and request validation
• Build-in input validation framework

Extensible Component Architecture
• Extensible widgets based on GWT

The major challenge is the integration of 3D interactive content in the web portal and application framework. In 2009 a new API for 3D graphics in web browsers was initiated:

WebGL (Web Graphics Library) is a JavaScript API for rendering interactive 3D graphics and 2D graphics[2] within any compatible web browser without the use of plug-ins. WebGL is integrated completely into all the web standards of the browser allowing GPU accelerated usage of physics and image processing and effects as part of the web page canvas. WebGL elements can be mixed with other HTML elements and composited with other parts of the page or page background. [12]

The WebGL API is a low-level API which requires some extra efforts to transform the content format that is typically used in 3D-city-models, like CityGML. For 3D computer graphics there are already some standardized high level file formats that are different in that they represent a so called scene graph. A scene
graph is a hierarchical graph of objects that make up a particular scene. An important and widely accepted format established by web3d consortium is the X3D [13] standard.

X3D is the ISO standard XML-based file format for representing 3D computer graphics, the successor to the Virtual Reality Modelling Language (VRML). X3D features extensions to VRML (e.g. Humanoid animation, NURBS, GeoVRML etc.), the ability to encode the scene using an XML syntax as well as the Open Inventor-like syntax of VRML97, or binary formatting, and enhanced application programming interfaces (APIs) [14].

There are various clients that already use X3D. One of them is the X3DOM framework developed by Fraunhofer IGD that renders X3D content in the browser using WebGL. The X3D format is also supported by the CityServer3D and therefore a complete content delivery chain can be established starting from CityGML over CityServer3D to X3D and then in the browser via X3DOM to WebGL.

The main features and ideas of X3DOM are given in [15] as follows.

X3DOM (pronounced X-Freedom) is an experimental open-source framework and runtime to support the ongoing discussion in the Web3D and W3C communities how an integration of HTML5 and declarative 3D content could look like. It tries to fulfill the current HTML5 specification for declarative 3D content and allows including X3D elements as part of any HTML5 DOM tree. The goal here is to have a live X3D scene in your HTML DOM, which allows you to manipulate the 3D content by only adding, removing, or changing DOM elements. No specific plugin or plugin interface (like the X3D-specific SAI) is needed. It also supports most of the HTML events (like “onclick”) on 3D objects. The whole integration model is still evolving and open for discussion.

We hope to trigger a process similar to how the SVG in HTML5 integration evolved:

- Provide a vision and runtime today to experiment with and develop an integration model for declarative 3D in HTML5
- Get the discussion in the HTML5 and X3D communities going and evolve the system and integration model
- Finally it would be part of the HTML5 standard and supported by every major browser natively

In short, it brings interactive 3D to the browser using the WebGL technology [12], that became available in several web browsers recently. No other plugins or special software installation is needed. Apart from that this abstraction also has the advantage that the applications need not make assumptions on the capabilities a web client supports.

So for example if no advanced WebGL or 3D Hardware is available the framework can use fallbacks as server-side-rendering or Adobe Flash. The application needs not to be adapted for this and is easier to maintain. Also no plugins or other third party software is needed to be installed on the clients.

The UrbanAPI applications use therefore the X3DOM framework to integrate the 3D content in the web context. Interactive components will be realized with JavaScript code and a X3DOM encapsulation for the Vaadin framework will be developed. The use of existing solutions and proven components gives us the freedom to put an emphasis on the research aspects of the project. Nevertheless it is a challenging task to integrate all the different tools into one usable portal application from a technological point-of-view. Some components like the ViewHUB are in an early development stage and can be influenced by the experiences
made in this project. From a research perspective it is interesting to find out, if and how the exiting tools can support urban planning by application of proven solutions to a new domain.
2.10 Summary

The software architecture presented in the previous sections integrates software components from all three applications into one single web portal. Modern portal software is quite flexible so the UrbanAPI portal can be configured for the different applications. Apart from that, we support scenario-specific 3D visualizations that can be configured through the CityServer3D AdminTool.

The integrated portal makes use of a range of web-services that are developed in the UrbanAPI project. The projects partners in work package 3 (“Software Platform Development”) work together to achieve best reusability of each of the web services. The description above reflects this as most services appear in more than one application.

We also make use of existing services in order to speed up our development. This allows us to quickly access technology that is already there without the need to re-develop them again. The service-oriented architecture ensures low coupling, so the services developed in UrbanAPI have no direct dependency to the existing services apart from their well-defined service interfaces.

Putting all this together the UrbanAPI web applications use a technology stack depicted in Figure 9. The applications will open up new possibilities to interaction within a city for example by using 3D city-models and some other data like zoning information or GSM-tracks that is visualized directly within the 3D-City-Model. Based on this the people can directly give feedback or even propose new ideas and thoughts.

An implementation should consider that the cities may already have existing infrastructure that serves similar purposes like CMS or portal software. The cities also may have policy constraints that they are obliged to use the existing infrastructure for all projects that may have some public attention or even need to fulfill internal special requirements like security. Therefore we must avoid any kind of vendor or technology lock-in to be able to adapt the core of our solution to different environments. As a consequence, the core of the application need to be agnostic of special environments and basically should work in different environments. This aspect is to be covered by the ViewHUB services.
3 User Interface Elements

Here we describe the elements that apply to all implemented use cases of the applications. The intended target audience for these may be different depending on the application. If some element is very specific for a use case or city, this will be mentioned within the description. Screen shots, descriptions and figures represent the current state of the final prototype. Feedback provided by the users and stakeholders is also considered to improve the user interface. In the case of CityServer3D several other components already exist and are reused to contribute to the project. These are not explained here and the basics are covered in deliverable 3.6, chapter 3.1.1. Closer details about the difficulties encountered during development are given in [16].

3.1 User Interface Elements for the 3DSC application

In this section the elements that apply to all use cases of the 3DSC application are described. The target audience for the elements is not the technical expert user, but the occasional user that has a high level view to the scenario. For these users a web based application is suited very well as the installation and maintenance efforts on the client side are slim to none. Furthermore their skill level must be considered in the application design. For example the usage of suggestive visual idioms and simple actions should be favoured.

In the following the basic concepts of navigation, menus, popups, toolbars and visual display in a web context are presented. Existing examples will be given if possible and useful, otherwise a sketch or screen shot of similar applications will be used to explain the visual elements and to give impression how the final may look like. The use cases from the requirements documentation [2] can be grouped after some common goal and may use a more or less common set of visual elements. Therefore a common schema for the layout of visual elements can be defined as in Figure 11. If some use case does not need all elements, the component will not be displayed at all in the context. For example if a user is not allowed to edit a scenario but merely can put a common feedback, the elements other elements are not needed and will not be there. The free space will be used by the remaining components in the most useful way. For example the 3D view may be enlarged, but the “Button bar” not, as the buttons will be of a static size.

One aspect of research is how a 3D city model can help in explaining planned actions and what are the differences and benefits compared to classical 2D GIS applications. Especially the question how 3D interaction elements must be designed to be easy to use by laypersons is to be considered. The principles for the interface design and criteria for evaluation can for example be taken from [17].
As the web portal is based on Liferay there are already several tools for managing page contents, look-and-feel and other basic things. The most important aspect is that power-users or administrators can create new pages and adapt the design in self-service without the need to dive into web application programming. The most important aspect herein is the arrangement of components on a page. Liferay offers a lot of ready-to-use templates, for example column-based page layouts (Figure 10). This was used to arrange an overview of the Components for the 3DSC Application in Figure 11.
The starting point of the application from an end-user perspective is the selection of a page which represents a scenario. After the user selected a scenario he gets to the main page based on the layout in Figure 11, where the scenario can be viewed or edited. The layout arranges up to 6 main components of the application:

- **3D-View**
  The 3D interactive display for a scenario. The viewpoint can be changed directly by using the mouse

- **2D-map**
  A map representation, like OpenStreet Maps

- **Object repository**
  Contains objects the user can use to modify the scene, like city furniture, trees, etc.

- **Feedback/comment list**
  An area where the user can manage the annotations he or other users made

![Figure 11: Three-column layout schema for the 3DSC application with several components.](image)
• Button bars – either within the 3D display or where appropriate as separate component
Contains buttons that enable special functions like uploading files, switching manipulation modes, content depends on use case context

The components outlined above will be detailed in the following chapters. Which components are actually available depends on the use case or scenario.

According to the needs of the scenario different components can be put on the page, for example to upload new designs. Navigation between the pages is possible by clicking on the tab bar provided by Liferay. Modal dialogs\textsuperscript{3} are avoided where possible.

3.1.1 Content Display and basic control elements

3.1.1.1 2D Map

For the implementation of 2D maps and the navigation within these maps the OpenLayers [18] toolkit is used. This toolkit provides navigational components for the navigation like click able arrows to move the visible area in the map. Also a zooming component is available to have a closer view on the map. In Figure 13 an example is shown: On the left side you can see the components mentioned above. These will be used for all 2D maps. Being a JavaScript library it is easy to integrate in a web application. Also it is very flexible and can be easily adapted and extended with the functionality required by the UrbanAPI tools and supports standard data sources like WebMapServers. It can be considered as the de-facto standard for integration of map data in a web application.

The configuration dialogue supports the following parameters:

• \textit{epsg3857north} - Northing
• \textit{epsg3857east} - Easting
• \textit{zoomlevel} – The zoom level which determines the scale of the map, e.g. how close to the area of interest the initial display should zoom

EPSG3857 is the google pseudo mercator spatial reference system\textsuperscript{4}. To obtain the position in this format the 2D map of AdminTool can be used for example. Press \textit{save} to accept the entries.

\textsuperscript{3} A modal dialog is a dialog that pops up and disables all interactions with any other application elements. This restriction in interaction options should be avoided as it naturally limits the user in his work. There are some situations where this is appropriate for example a file save dialog. On the other hand a dialog that shows the properties of a selected object should never be modal, as for example a property has to be edited based on information that is available on another dialog element in the system. This other element can then not be made visible as the modal dialog may lock the user interface until it is closed.

\textsuperscript{4} \url{http://wiki.openstreetmap.org/wiki/EPSG:3857}
3.1.1.2 3D Display

Figure 12: Configuration of 2D map component

Figure 13: OpenLayers 2D map example showing navigational components
The 3D display is basically implemented through the X3DOM [15] framework. An example is shown in Figure 14. Movement and interaction within the 3D scene is possible directly via the mouse. For example you can move directly in the scene by clicking the left button and moving the mouse. You also have the option to select objects by clicking on them with the left mouse button. Additional functions can be triggered by buttons outside or inside the 3D-view on the selected objects. X3DOM has several movement modes which are described in detail in annex 4.1 or [19]. For most scenarios we assume the use of the so called “Helicopter-mode”, which is active by default. To look downwards/upwards and to move higher/lower use the keys (8/9 and 6/7). Another useful mode is the pedestrian mode, where you take the perspective of a person walking around. You can switch between the modes with the shortcuts described in annex 4.1 or [19] or, with the buttons visible in the lower right corner of the 3D map.

3.1.1.3 Export Image from 3D-View

A button on the button bar enables to export the 3D-view area as an image that can be downloaded (Figure 15). Supported image format is PNG and the resolution of the image is depending on the client, e.g. the browsers window will determine the size of the exported image.
3.1.1.4 Navigation mode change

Figure 15 also shows two buttons to change the user navigation mode from helicopter (left button) to pedestrian (middle button). By clicking on one of these buttons this mode change becomes active when the user makes the next movement.

3.1.2 Scenarios

A scenario in the context of the UrbanAPI toolset combines a specific part of a 3D city model with other aspects like illumination, point-of-view and optionally some advanced visualizations of data. It focuses on a specific topic or goal in some restricted model area. Along with that a scenario can define interaction options for users like annotating or changing something in the scenario.

Some special administrative user of the city prepares the scenarios available optionally together with some technical domain experts. This task is mostly facilitated by the CityServer3D-Admintool. After the scenario has been prepared it can be published for web-access. A scenario has an owner assigned to it that initially created it. So the initial version is created by some city representative and made public to some groups of users or the public. The groups will now find the scenario in their scenario selection dialogs. With that knowledge an effective notification mechanism for tracking participation by other users can be deployed (see 3.1.7). A tracked change can be the movement of objects.

Based on the initial scenarios created by the cities the users authorized to do so may amend the scenarios in several ways. The can just comment on a scenario by adding annotations or they can be allowed to create their own version of a scenario by adding objects or including completely new things they crafted themselves to the scene. They can also share their version with others and give the same options to change or comment.

3.1.2.1 User Management

Users that want to change something or want to give some feedback must be known to the system and therefore be registered. A complete user management is beyond scope of this project and is provided by using existing Web-Technologies like internet-portals such as Liferay (see [21]). These provide a complete infrastructure for user management and integration of various information sources in one website. For the 3DSC application we assume that user-management is provided as a service to the application.

Such a service must be at least able to deal the following information.

---

5 See [5] and section 2.7 for information about the AdminTool.
Every scenario has

- an owner (creator)
- groups that can access it

Some examples for groups can be:

- public
  All people that can access the website, even anonymously
- departments in the city
- city council
- decision makers
- external partners like architects or planners

All these groups are managed by the portal administrators of the cities and will be assigned to the scenarios with the AdminTool application.

A scenario also should have different access rights per group

- View only
- Create new version
  The right to change something can later be implemented separately for every change action. For now a user needs to have his own version to simplify the scenario management
- Comment

So it is possible to share a scenario with a closed user group that can make changes to it and at the same time the scenario is visible to the public. An owner can always do anything.

The user management is used so that it is abstracted by from particular portal application frameworks, so that we can easily exchange different implementations. This enables us to integrate the tools on different platforms that participating cities may use.

For the final portal prototype the user management is delegated to the services Liferay offers. Please see D3.6 section 3.1.2.1 for closer details on how this is set up.

### 3.1.2.2 Scenario Selection

All scenarios that are available for a user are available as pages in Liferay, an example is shown in Figure 16. The pages are filtered according to the user’s rights, e.g. a user can only see scenarios which have a group assigned he is assigned to.

Depending on the users rights, a user can view, annotate or create his own scenario based on the ones provided by cities and make his own change. Only owners are allowed to edit their own scenarios.
3.1.2.3 Sharing Scenario

To share a scenario a user assigns permissions for user groups to access the page. This is done via Liferay permission options (Figure 17). This opens a dialog where users can assign groups that can view, change or comment on the scenario. Whenever someone amends an existing scenario with his own changes, the owner of the original scenario is notified about the change. Annotations on the other hand will not lead to new scenarios.

If a scenario should no longer be available to other people it is also possible to revoke the sharing of a scenario by removing the selection on all groups shown. There are two reactions on this, depending on the hierarchy.

- If a root / administrator scenario is revoked from sharing, all versions of it, no matter who owns them, are no longer visible to the removed group. Even owners of a version cannot see them.
- All other versions: only the exact version is no longer available, all descendants are still visible.
3.1.2.4 Save Changes

A “Save changes” dialogue is not needed, as there is no transactional behaviour between user actions. This means that every change is made persistent in the moment the user finishes a direct manipulation on an object or presses an optional “save” button. In a web context this makes more sense, as a user might not finish his changes but just goes to another page or just stops to interact. This finally will lead to an application timeout resulting in a total loss of changes. The dialogue requested as a requirement was therefore not implemented.

3.1.2.5 Feedback for a scenario as a whole

For each scenario as a whole a feedback can be given. This could be thought as something that is displayed like a blog entry and not associated to a particular element that is portrayed in the scenario. For this a tabular area besides the maps is used to display the comments in a short form. On clicking on them a popup will show the whole text. The contents also adhere to the rules imposed by the user rights management, which means users can see the comments of others if they are allowed to and can give of course their own feedback to the scenario. It is not intended to use this as a platform for intense discussions between users. For this a more appropriate component like a web-forum or blog should be used. If this is needed, a simple link or something alike depending on the technology used can be introduced to refer to the discussion. For
example portal software mostly has web-forum and blog components available out of the box and easy to use. A notification that a new comment has been created (see 3.1.7) will also be sent to the owner of the scenario.
3.1.3 Object manipulation

This section describes the elements that can be used to modify objects in the scenario. If it is allowed in the scenario buttons and handles for direct manipulation will be shown. It is possible to navigate within the 3D scene as described in section 3.1.1.2 in helicopter or pedestrian style. All interaction options are directly located within the component, including a context dependant button bar on the upper left corner and a navigation toolbar in the lower right corner. The most prominent manipulation options are available as handles drawn on the object for direct interaction.

Figure 18: View or edit scenario with edit functions shown.
3.1.3.1 Object selection in 3D

To manipulate an object the first step is to select an object. This can be done by clicking on an object in 3D view. To indicate that an object is selected, many visual hints can be used. A transparent yellow bounding box is drawn around the object as shown in Figure 18. Once the object is selected, users can carry out some actions with it as described in the following sections. Direct manipulation in the 3D scene is made available in the latest 1.6 release of X3DOM via sensors. This is a major step for improving the usability as compared to the buttons implemented in the first version of the 3D component.

3.1.3.2 Move

After selecting the object users can directly interact with it by using the handles. To move a selected object just click on the yellow box around it, hold the mouse button and drag it to the new position. The object will only move horizontally (X-, and Y-Axis).

![Figure 19: Horizontal movement can be achieved by clicking on the yellow bounding box.](image)

To move vertically, another handle located above the object can be used (Figure 20). Click on the red cones above the object, hold the mouse button and drag to the desired height.
Rotate To rotate a selected object click on the green band above the object, hold the mouse button and drag the mouse to the desired angle.

Figure 20: Red cones for horizontal movement (up/down)

Figure 21: Green band handle geometry for rotation
3.1.3.3 Re size

After selecting the object users click on the button “scale” on the toolbar. To re-size (scale) a selected object they click on larger or smaller. This function is rather questionable as it takes not into account that real objects cannot adapt the size easily, even worse it creates a kind of delusion that a design, probably made up by an architect can be adapted easily. This takes not into account that it is not sensible to scale floor heights or door sizes in such a simple way. Therefore this function is only available as an additional option via buttons, not directly as handles.

3.1.3.4 Delete object

A selected object can be deleted after selection by clicking on the delete object button (Figure 22).

3.1.3.5 Deselect object

A click on this button removes all handles and the upper button bar.

3.1.3.6 History records

Every movement and rotation is tracked by the object in history records. With this, the users can see what was changed and keep track of it in an easy visual way. Every change is shown as a red cone (Figure 23), but only when an object is selected these cones will be displayed only for this particular object.

Figure 22: Buttons for scaling, right side
3.1.3.7 Object integration (upload)

There are two specialised components for data upload.

For architects or users that want to upload just a small design, building or an extension to the object palette the “User Upload Portlet” is available (Figure 24). The object has to be uploaded as a compressed ZIP-file containing a valid model in X3D format. The component unzips the file and analyses the contents and finally supplies a list of selectable objects. On clicking the “select” button, the object is transferred to the 3D scene the same way as from the component palette. A click in the 3D scene places the object.

Figure 23: Selected object with history records, indicated by red cones
Administrative users, which need to setup a scene as a whole should use the “Upload portlet” (Figure 25) which works the same way, but does not offer a “select” button. For now the url to be used in the 3D-display has to be assembled manually as pointed out in the component itself.

For special scenarios the administrator can prepare a selection of objects that can be placed in the scene. These objects are displayed in a special area with a thumbnail representation on a button. On pressing that button a new instance of the object chosen is placed at the mouse position of the current 3D-view. By selecting “Navigation” only navigation in the 3D scene is possible, “Annotation” enables to annotate on a design and “Get Distance “ enables measuring.

3.1.3.8 3D Interaction options and object palette (Street furniture)

For special scenarios the administrator can prepare a selection of objects that can be placed in the scene. These objects are displayed in a special area with a thumbnail representation on a button. On pressing that button a new instance of the object chosen is placed at the mouse position of the current 3D-view. By selecting “Navigation” only navigation in the 3D scene is possible, “Annotation” enables to annotate on a design and “Get Distance “ enables measuring.
3.1.4 Annotations

Annotations are different to “feedback as a whole” (see 3.1.2.5) as they allow creating comments with some direct spatial reference. The basic idea is quite similar to the “REDLINER” concept in [21].

The Redliner interface is simple. It can be described in a few sentences and intuitively understood even by novice users. It contains a window with two frames: one containing users’ text annotations and the other for 3D interaction with the design model. Comments listed on the left side of the Redliner window are sorted by time and by author. The right side displays the 3D model. (Screen resolution issues discouraged us from embedding the comment text directly into the 3D model.)

In contrast to the concept outlined above the annotated objects will directly show the annotation as an overlay in the 3D-scene as screen resolution is not an issue on modern high resolution devices for example the retina displays from Apple. If the number of annotations is too big only an annotation mark is shown which will pop-up the details of the annotation on hovering or clicking on it (Figure 27). If users click on an existing annotation, it should be possible to set the camera to the very same position and viewers angle that the creator of the annotation used at the time of creating it. Then a reviewer can see exactly what the user saw and understand better what he is concerned with.
Figure 27: Annotation display in the 3D scene
Annotations to Objects in the 3D-view can be created by clicking the “annotate” button from the toolbar. Figure 28 shows some annotation that is directly added within the 3D display.

### 3.1.4.1 Annotation Details

The details of an annotation can be also displayed in a tabular for using the Annotation Details component shown in Figure 29. The columns of the table can be sorted by clicking on the sort symbols next to the column caption. The table shows who and when the annotation has been created, some unique key number and title and details.
3.1.5 XML Download

The object placements and the changes to it can also be obtained in XML format. A special component is available to download the data for a particular scenario. Figure 30 shows the component which only offers a link to click on to get the data. Configuration for this component is shown in Figure 31: the base URL of the scene and the optional scenario identifier are to be configured. These values are typically the same ones as in the 3D display component configuration.

![Figure 30: Component to download changes in XML format](image)

![Figure 31: Configuration of XML Download](image)

In Table 1 an example of such an XML file is given. The following information is available:

- `createdon` – the date when the object was created
- `id` – a unique surrogate key as number to identify the item
- `location` – the 3D coordinates of the object-oriented
- `objecturl` – the type of object
- `rotation` – if the object was rotated the rotation as quaternion
- `scale` – the scaling vector for the object
• scenarioid – the scenario the object belongs to
• userid – userid as Liferay id of the user who made the change

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<objectRefs>
  <objectRef>
    <createdOn>2014-079T13:53:49.026Z</createdOn>
    <id>332</id>
    <location>1354.6088186667564 -32.26064471782498 -579.1824979900169</location>
    <objectUrl>/urbanapi.X3Drepository-portlet/fs/components/tree/tree.x3d</objectUrl>
    <rotation>0 0 0 0</rotation>
    <scale>1 1 1</scale>
    <scenarioId>/urbanapi.X3Drepository-portlet/fs/bol2/scene.x3d</scenarioId>
    <userId>10405</userId>
  </objectRef>
</objectRefs>
```

*Table 1: XML download example*

### 3.1.6 Free Drawing Tool

The current development state of the X3DOM technology is not supporting a “Free Drawing Tool” but this is considered for future releases. A closer inspection shows that this would require some substantial efforts which could not be covered by the project, mainly in terms of time needed. Therefore this functionality was rejected from development and deferred to future releases.

### 3.1.7 E-mail notification

Whenever users change a scenario, by adding annotations or objects, the owner of the original scenario will be notified by email. Annotations do not lead to new scenarios, but the owner of a scenario will be notified on any new annotation that has been created by someone. There will be no GUI component for this in the UrbanAPI toolset, as standard Internet E-Mail mechanisms are used and any Client like Thunderbird or iMail can be used.
3.1.8 Route planning

*Special elements for Bologna*

The user interface for route planning will use a custom layout as indicated by Figure 32. Most of the actions will be triggered from within the 2D-map as for route drawing along some roads it is much easier to use a 2D-map than doing it in 3D. However the 3D-view will display the routes as well.

Whenever a route has been drawn users can name it. A popup as shown in Figure 33 allows users to enter a name for it. The same popup will be shown on clicking on an existing route when users are the owner of the scenario. A „Delete“ button will also be shown if users want to remove the route. The routes drawn can be categorized by vehicle type like car, bicycle or bus. For this a drop-down selector will be added to the popup which allows choosing a type.

*Figure 32: Basic layout for route planning. Both maps are synchronized. After selection of an area in the 2D-Map the corresponding 3D scene will be shown.*
The 2D-map as shown in Figure 34 has several interaction buttons located to the upper right and the lower right corners. In the upper right corner users have buttons for

1. **Selection of routes**
   On pressing this button the user can select routes by clicking on them. An appropriate edit dialog is brought up that is essentially the same as in Figure 33. Pressing “Delete” on this dialog removes the annotation or route.

2. **adding a route**
   On pressing this button the user can now create new routes by clicking on the appropriate place on the map. Each click will create a new line segment for the route. When the user wants to finish the route he simply needs to double-click on one location. When the user finished the route definition, a dialog like in Figure 33 is brought up in which a name can be assigned to the route.

3. **selection of an area to zoom in**
   When this button is pressed, the user can select an area to zoom in by dragging a rectangular area in the map. The user clicks and holds the left mouse button on one corner and drags the mouse to the cater-cornered other corner and releases the mouse. Then the map zooms in directly.

4. **moving the viewpoint of the map with the mouse**
   On pressing this button the user can pan the map, e.g. move the visible area. For this the left button must be clicked and held somewhere in the map. The move with the mouse now moves the visible area. On button release this area will remain the current area.

In the lower right corner users have buttons for

- **toggle visibility of routes**
  On pressing this button all routes will be made visible or invisible. The previous state is just toggled.
Figure 34: Example 2D-route display
3.1.9 Visual compare

**Figure 35: Scenario comparison**

**Special elements for Vitoria-Gasteiz**

Visual comparison can be aided by a dual scenario display. No special components are needed for this – just two 3D-displays with different data setup can be used. The scenarios must cover the same spatial extend for this. Then a 2D map for example with OpenStreetMap data can be shared between both scenarios to aid the navigation. The main part is made up by the two 3D-views with the scenarios to compare. The navigation within these 3D-views will be synchronized so that the position and viewing angle will be the same in both displays. Based on the observations made, one can now give feedback on the whole or annotate both scenarios. The set up of the scenarios that are available to compare for end users is configured with the AdminTool.
3.2 User Interface Elements for the Motion Explorer application

In this section the GUI elements of the Motion Explorer application will be depicted and the reader should get an overview over the various functionalities of the application. At the end of the project there will be a separate user manual to describe the final GUI in its entirety.

3.2.1 Applied open source software and libraries

The Motion Explorer application developed for the UrbanAPI project will be part of the overall framework applicable through a web interface but the application shall be able to function as stand-alone web-applications as well. 3D-presentation and interactive control features will be integrated by making use Fraunhofer 3D functions related to CityServer3D.

The Motion Explorer is based on a server-side data-base (PostGreSQL\(^6\)) storing pre-processed mobile phone location data as well as population data to enable user-defined calculation based on these data layers. The client side of the application will be an HTML-based web-page using state-of-the-art Javascript\(^7\) libraries such as Open Layers [18], Dojo\(^8\) and jquery\(^9\). The server-side will be a Tomcat [8] server which processes the user requests and deliver the results to the web-client. For the 3D part of the GSM data visualization an X3DOM interface [15] will be used to also enable the use of the mobile phone location data (aggregated to spatial entities like raster cells in the City3DServer software). The Motion Explorer application should be as generic as possible to be flexible with respect to further cities which will be using the application for their own purposes but to make it also capable of being integrated more easily into web-platforms like the the UrbanAPI one.

---

http://www.postgresql.org/
http://dojotoolkit.org/
http://jquery.org/
3.2.2 GUI Overview

Although the Motion Explorer is a stand-alone web-application, closely linked to the other UrbanAPI components (CityServer3D and UDS) via the database it is getting its data from. The underlying database is capable of serving the data visualised in the Motion Explorer application to the other applications as well, because most of the calculations are being done within the database via database functions.

The GUI consists of a intuitively controllable interface securing high usability for easy understanding of the visualised content. Figure 36 shows the main page design of the application.

It consists of the following graphical user elements:

The application’s screen is split into four frames:

1. control menu for layers (left hand side)
2. viewing area (centre)
3. legend (right hand side)
4. control panel for control of actions (bottom)

The control panel (bottom frame) allows changing multiple variable values which results in dynamically updated map in the view area (centre frame).

Figure 36: Motion Explorer Main Window

Although the Motion Explorer is a stand-alone web-application, closely linked to the other UrbanAPI components (CityServer3D and UDS) via the database it is getting its data from. The underlying database is capable of serving the data visualised in the Motion Explorer application to the other applications as well, because most of the calculations are being done within the database via database functions.
GUI elements:

Menu elements

1. Drop-Down list of actions
   The action drop-down list enables the user to choose from a list of possible actions (Figure 37). It is then possible to click into the map in the centre frame to choose a desired area of interest (AOI), i.e. a raster cell of interest.

2. The time range slider
   The time slider lets the user choose which target time they would like to explore. E.g. it is possible to choose a time later in the day to see where people at a certain AOI are going to (Figure 38).

Figure 37: Action Drop-down list

Figure 38: Time slider
3. **Action buttons**

The action buttons at the bottom let the user perform additional functions, i.e. playing an animation, choosing a specific time of the day, exporting data and printing the map frame (Figure 39).

- **“Download OD-Matrix”:** This button will send the query to the database on the server which will send back a map of the desired AOI and times chosen.
- **“Download .shp”:** This button lets the user download an ESRI shapefile of the data for further use in a desktop GIS e.g..
- **“Play”:** This button will create an animation for the chosen time.
- **“Time boxes”:** These boxes allow for choosing a specific time of the day.
- **“Background Opacity”:** These sliders enable the changing of the background opacity of the map.
- **“Create PDF”:** This button lets the user export the map to a PDF file.

4. **Legend**

The legend shows the explanation of the different map elements depicted on the right hand side (Figure 40).

---

**Figure 39: Action buttons**

- “Download OD-Matrix”: This button will send the query to the database on the server which will send back a map of the desired AOI and times chosen.
- “Download .shp”: This button lets the user download an ESRI shapefile of the data for further use in a desktop GIS e.g..
- “Play”: This button will create an animation for the chosen time.
- “Time boxes”: These boxes allow for choosing a specific time of the day.
- “Background Opacity”: These sliders enable the changing of the background opacity of the map.
- “Create PDF”: This button lets the user export the map to a PDF file.

**Figure 40: Legend**
Map GUI elements

5. Main Map
   The main map frame shows the user query's actual map (Figure 41).

6. Drag and rubber band zoom button
   The drag and rubber band zoom button let the user navigate on the map more precisely.

Figure 41: Main map

Figure 42: Drag and rubber band zoom button
7. **Zoom controls**
The zoom button enables the user to zoom in and out on the map (the same is possible with the mouse wheel)

![Zoom controls](image)

*Figure 43: Zoom controls*

**Motion Explorer Mapping Modes**
The Motion Explorer lets the user choose between two mapping modes, where the second is still experimental:

1. **A 2D static view for time steps:** This view is used for plain 2D map visualization for display of static data like the sum of users at a certain date and time
2. **A 3D static view for time steps:** Analogue to the 2D view this mapping mode is used to display 3D content for a certain point in time.
1. **2D View**

The 2D view is a plain map showing the desired borders and AOIs (Figure 44)

*Figure 44: 2D map view*
2. **3D view**

The 3D view is analogue to the 2D view but is showing a 3D representation of the user’s query. (Figure 45)

*Figure 45: 3D View*
3. **Mouse Click-Chart**

In order to be able to visualize the occupation intensity over time, the application features a hover-chart at the bottom of the application frame presenting the database query results for each particular raster cell chosen (Figure 48). The chart is created on the fly via parsing the database request results and creating the chart via the Ext-JS chart functionality\(^\text{10}\).

---

\(^{10}\) [http://docs.sencha.com/extjs/3.4.0/#/api/Ext.chart.Chart](http://docs.sencha.com/extjs/3.4.0/#/api/Ext.chart.Chart)
3.3 User Interface Elements for the Urban Development Simulator application

Following the general architecture as well as the main features of the UI will be depicted. It should be noticed that here only main features of the Urban Development Simulator (UDS) are described within this deliverable. To teach a user how the UDS can be used video tutorials are provided, depicting different use cases of the UDS.

3.3.1 The general architecture of the UDS

The UDS will not be a classical web application using a web browser like Firefox, Safari or Chrome as a Client. It was clear from the very beginning of UrbanAPI project that the UDS concentrates on the simulation of the behaviour of the agents and their consequences on the future land-use, which is a complex issue requiring a lot of data and different behaviour rules of the agents. These rules have been explored and identified through statistical analysis and expert judgement during the model development and validation process. The model will be built with MASGiSMo (Multimethod Agent-based, System Dynamics and GIS modelling platform) as the basis platform developed within the last years at the AIT enabling to create stand alone models running on a desktop computer or notebook. Within the UrbanAPI project this platform was enhanced to allow different users to interact with the same model hosted on a server. While the CityServer3D application focuses on the high-resolution presentation of a static urban environment, predefined through a detailed 3D object model, focuses the UDS on simulating the urban dynamics over time at a block scale, thus a high-resolution of e.g. building textures will not be provided within this context. This means that in the UDS, 3D objects as different buildings can only be represented in a very rough way. Nevertheless the 3D presentation available is assumed to improve the impression of the urban dynamics and helps to engage with the local stakeholder by enhancing the recognition value of the local conditions. Certain simulation results will therefore be provided to a broader audience via a web portal with the help of the X3DOM technology (cf. section 2.9).

For a smooth operation of the UDS it is hosted on a server at the Austrian Institute of Technology (AIT). The users can attach to the UDS through the internet using the Java webstart Interface (http://www.java.com/en/download/faq/java_webstart.xml ). Improvements of the UDS will be directly handled by AIT. In principle is it possible to host the UDS on a different server e.g. managed by the respective city, but this could be difficult due to technical reasons and thus is it not planned so far. An example for such technical reasons is that within the respective city the “wrong” operating system of the server currently could be used. The user of the UDS can run their simulations, create new scenarios and export the results to their own destinations (to local directories). The following figure depicts the main architecture of this process. The arrows in the figure show the possible interaction between the different parts of the client-server architecture. The UI enables to steer the core UDS (e.g. create new scenario results), it furthermore allows directly to interact with the PostgreSQL Database (DB) to export former simulation results or to change the simulation input stored in the DB.
At the user's local computer conducting the UDS only minimal system requirements are necessary like installing a specific JAVA version or further open source software and running Windows 7 64-bit version.

### 3.3.2 General UI features

The graphical user interface (UI) of the UDS has been developed and will be extended by enhancing the simulation platform MASGISmo with specialized features required for the UDS. The most important components of the UI are related to different main functionalities which are:

- spatial simulation features conducted through GIS functions,
- interactive control features,
- scenario simulation analysis and
- visualisation features.

The UI is shown in the following figures.
Figure 48: UDS-UI Main Management View

Figure 49: UDS-UI Main Map View (START SCREEN)
The top row provides the generic simulation controls, enabling to perform several simulations (multi-run simulation) with determined parameter variation, a single simulation and the possibility to simulate just one time step. The top row depicts the current tick (time step) during the simulation and the number of the simulation during a multi-run simulation, too. The Button Open_DB_Browser opens the data base browser. The tab Results-Graphs contains predefined result graphs and the Results-Tables tab shows predefined result tables. Beyond these, during the UDS development (for the respective city) defined, standard results will it be possible to extract further results via a so called DB-Browser, see below figure and chapter 3.3.6.

In general from the UI the user will be able to retrieve a wealth of spatial information as well as interactively change the simulation input to create new simulation scenarios.

Figure 50: UDS-DatabaseBrowser

3.3.3 GIS viewing and layout functionality

The following screenshot depicts the implemented Open Jump connection used to provide additional GIS features. How in detail the features can be used will be shown in the above mentioned tutorial videos as this is assumed to be more convenient for the user of the UDS to learn how to work with the UDS.
Some original GIS features working on ASCII grid layers, which are part of MASGISmo, are hidden in the last version of the UDS to simplify the usage, as these are not necessary for the urban development simulation.

Within the UI in general a short description of the function e.g. of buttons, slider etc. will be provided through tooltip text, appearing when the user hovers the pointer/cursor over an item.

3.3.4 Interactive controls provided within the UI

For the applications developed within the UrbanAPI project interactivity of their users is a major requirement. For the UDS application the used simulation platform MASGISmo already includes some interactivity features, which are mainly focused on changing input parameter values or the appearance of the simulation results. The extension by connecting Open Jump enabled to increase the list of features regarding GIS data handling (import, export, visualization…). Thus a new so called master plan for the development for Ruse can be imported or the existing updated. How this works is shown within the tutorial videos. Furthermore additional tools like the Geoserver Exporter interface have been developed for the UDS enabling the user easily to extract further results.

Main interactive features:

- adding of new elements or deleting of elements (as streets, residential areas …)
- changing parameter values with sliders and input fields
• importing and exporting of new GIS layers (of different format)
• changing the style of the layer
• editing and adapting existing layers

These new or enhanced interactive features are currently still under development what means that for a future version of the UDS additional features can be provided.

3.3.5 UI for simulation analysis

The following features can be used during a simulation e.g. to generate a snapshot of a certain situation during a simulation run, or to further analyse the simulation results stored in the PostgreSQL database. They either can be used by simply clicking a button as in the case of taking a snapshot or with the use of Interfaces as in the case of generation of new layers.

**Main features for scenario simulation, documentation and analysis:**
• snapshots at individual time steps of the whole area (spatial map) or of a selected area,
• parameter charts and tables,(Results-Graphs and Results-Tables (see Figure 52)
• export of simulation results as tables or maps (csv files, png files, ESRI-shape files, kml files)

![Figure 52: UDS-Results Graph Example](image-url)
3.3.6 Data handling

The UDS will store the tabular data within a PostgreSQL database (DB). To retrieve information from this DB an interface (the so called Database Browser a part of MASGIsmo) was enhanced. To simplify the extraction of the most important result files a method was created which can be activated by clicking a button in the main management view (see Figure 3-42). Beyond the possibility to extract content from the DB the Browser will enable importing new data sets to the DB to allow additional and new simulations using new parameter values. To access the PostgreSQL DB open source software as pgAdmin\textsuperscript{11} could be used but this is not very convenient for the UDS user, as they would have to install the software and learn how to use.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{geoserverExporter.png}
\caption{Screenshot GeoServer Exporter}
\end{figure}

The Geoserver Exporter Window (figure 53) and the moreCharts Window (figure 54) have been developed to enhance the data extract capabilities. Both can be started from the main management view. How to use them is documented in the Tutorial videos.

\textsuperscript{11} http://www.pgadmin.org/ tested 09.01.2013
Figure 54: Additional Graphs Interface
4 Annex

4.1 X3DOM Camera Navigation

X3DOM provides some generic interaction and navigation methods. Interactive objects will be handled by HTML-like events. Navigation can be user-defined or controlled by specific predefined modes.

Currently X3DOM supports the following interactive navigation modes:

- Examine
- Walk
- Fly
- Look-at
- Look-around
- Game
- Helicopter

Non-Interactive movement encompasses the functionality of:

- Resetting a view
- Showing all
- Upright view

**Interactive camera movement**

**Examine**

Activate this mode by pressing the "e" key.

<table>
<thead>
<tr>
<th>Function</th>
<th>Mouse Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rotate</td>
<td>Left / Left + Shift</td>
</tr>
<tr>
<td>Pan</td>
<td>Mid / Left + Ctrl</td>
</tr>
<tr>
<td>Zoom</td>
<td>Right / Wheel / Left + Alt</td>
</tr>
<tr>
<td>Set center of rotation</td>
<td>Double-click left</td>
</tr>
</tbody>
</table>

**Walk**

Activate this mode by pressing the "w" key.

<table>
<thead>
<tr>
<th>Function</th>
<th>Mouse Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move forward</td>
<td>Left</td>
</tr>
<tr>
<td>Move backward</td>
<td>Right</td>
</tr>
</tbody>
</table>
Fly
Activate this mode by pressing the "f" key.

<table>
<thead>
<tr>
<th>Function</th>
<th>Mouse Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move forward</td>
<td>Left</td>
</tr>
<tr>
<td>Move backward</td>
<td>Right</td>
</tr>
</tbody>
</table>

Helicopter
Activate this mode by pressing the "h" key.
To look downwards/upwards and to move higher/lower use the keys (8/9 and 6/7).

<table>
<thead>
<tr>
<th>Function</th>
<th>Mouse Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move forward</td>
<td>Left</td>
</tr>
</tbody>
</table>

Look at
Activate this mode by pressing the "l" key.

<table>
<thead>
<tr>
<th>Function</th>
<th>Mouse Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move in</td>
<td>Left</td>
</tr>
<tr>
<td>Move out</td>
<td>Right</td>
</tr>
</tbody>
</table>

Game
Activate this mode by pressing the "g" key.
To look around (rotate view) move the mouse.

<table>
<thead>
<tr>
<th>Function</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move forward</td>
<td>Cursor up</td>
</tr>
<tr>
<td>Move backward</td>
<td>Cursor down</td>
</tr>
<tr>
<td>Strafe Left</td>
<td>Cursor left</td>
</tr>
<tr>
<td>Strafe Right</td>
<td>Cursor right</td>
</tr>
</tbody>
</table>
Non-interactive camera movement

<table>
<thead>
<tr>
<th>Function</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset view</td>
<td>r</td>
</tr>
<tr>
<td>Show all</td>
<td>a</td>
</tr>
<tr>
<td>Upright</td>
<td>u</td>
</tr>
</tbody>
</table>

4.2 Glossary

The glossary is taken or summarized from Wikipedia. For closer information see http://www.wikipedia.org/

X3DOM

X3DOM (pronounced X-Freedom) is an experimental open-source framework and runtime to support the ongoing discussion in the Web3D and W3C communities how an integration of HTML5 and declarative 3D content could look like. It tries to fulfill the current HTML5 specification for declarative 3D content and allows including X3D elements as part of any HTML5 DOM tree.

JSF

A Java-based web application framework. Defines an architecture with resusable components and is a standardized part of the Java Enterprise platform.

Apache Tomcat

The reference implementation of Java Servlets and Java Server Pages technology developed by the Apache Software Foundation and available as open source. Tomcat is an application server that provides software applications with services such as security, data services, transaction support, load balancing, and management of large distributed systems.

AJAX

Acronym for Asynchronous JavaScript and XML. Through the use of AJAX, web applications can send data to, and retrieve data from, a server asynchronously (in the background) without interfering with the display and behavior of the existing page.

GWT

Google Web Toolkit (GWT /ˈɡwɪt/) is an open source set of tools that allows web developers to create and maintain complex JavaScript front-end applications in Java. Other than a few native libraries, everything is Java source that can be built on any supported platform with the included GWT Ant build files.

Liferay

Liferay Portal is a free and open source enterprise portal written in Java and distributed under the GNU Lesser General Public License and proprietary licenses. It is primarily used to power corporate intranets and extranets.

Liferay Portal allows users to set up features common to websites. It is
fundamentally constructed of functional units called portlets. Liferay is sometimes described as a content management framework or a web application framework. Liferay's support for plugins extends into multiple programming languages, including support for PHP and Ruby portlets.

**FMC**

Fundamental Modeling Concepts (FMC) provide a framework to describe software-intensive systems. It strongly emphasizes the communication about software-intensive systems by using a semi-formal graphical notation that can easily be understood.

**ESRI**

ESRI is the leading supplier of Geographic Information Systems (GIS)

**Shapefile**

The Esri shapefile or simply a shapefile is a popular geospatial vector data format for geographic information systems software. It is developed and regulated by Esri as a (mostly) open specification for data interoperability among Esri and other software products. Shapefiles spatially describe geometries: points, polylines, and polygons. These, for example, could represent water wells, rivers, and lakes, respectively. Each item may also have attributes that describe the items, such as the name or temperature.

**JEE**

Java Platform, Enterprise Edition or Java EE is Oracle's enterprise Java computing platform. The platform provides an API and runtime environment for developing and running enterprise software.

**CMS**

A content management system (CMS) is a computer program that allows publishing, editing and modifying content as well as maintenance from a central interface. Such systems of content management provide procedures to manage workflow in a collaborative environment. These procedures can be manual steps or an automated cascade.

**Servlets/JSP**

A Servlet is a Java-based server-side web technology. JavaServer Pages (JSP) is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types. JSP is similar to PHP, but it uses the Java programming language.
References
1: J.Rix et al., Grant agreement: Collaborative project Annex I - Description of Work, Project: Interactive Analysis, Simulation and Visualization Tools for Urban Agile Policy Implementation, 2010
2: Zaheer Khan, David Ludlow, D2.1 - User Requirements Definition, 2012
3: http://www.cityserver3d.de/en/
4: www.mysql.com
5: Michel Krämer, Andreas Stein, D3.3 - Rule User Interface Documentation, 2012
6: http://en.wikipedia.org/wiki/Web_application
7: http://www.liferay.com/
8: http://tomcat.apache.org/
9: https://developers.google.com/web-toolkit/
11: https://vaadin.com/home
12: http://www.khronos.org/webgl/
13: http://www.web3d.org/x3d/
14: http://en.wikipedia.org/wiki/X3D
15: http://www.x3dom.org/
16: Jens Dambruch, Michel Krämer, Leveraging public participation in urban planning with 3D web technology, Proceedings of the 19th International Conference on Web 3D Technology, 2014
17: Jens Wandmacher, Software Ergonomie, 1993, de Gruyter
18: http://x3dom.org/docs/dev/navigation.html
19: http://openlayers.org/