ISEIS

Journal of Environmental Informatics 19(2) 79-92 (2012)

Journal of Environmental Informatics

www.iseis.org/jei

Design of Topologically Structured Geo-database for Interactive Navigation and Exploration in 3D Web-Based Urban Information Systems

M. D. Robles-Ortega[1,*], L. Ortega[2], and F. R. Feito[3]

[1]*Paraje Las Lagunillas, s/n. A3-102, University of Jaen, Jaen 23071, Spain*
[2]*Paraje Las Lagunillas, s/n. A3-140, University of Jaen, Jaen 23071, Spain*
[3]*Paraje Las Lagunillas, s/n. A3-138, University of Jaen, Jaen 23071, Spain*

Received 7 July 2011; revised 9 January 2012; accepted 7 April 2012; published online 29 June 2012

ABSTRACT. Three dimensional (3D) Geographic Information Systems (GIS) are the appropriate systems for storing, manipulating and visualizing spatial data in urban environments. Providing web-based interaction to these systems is quite challenging. In this paper we present the techniques and protocols incorporated into a web-based prototype for navigation and interaction in a virtual urban model automatically generated from a cadastral 2D GIS. The input data are 2D layers with lack of topology and feature/database linkage. The process generates a new set of three-dimensional urban entities topologically integrated in the geospatial database. The city surface is modeled considering the Digital Elevation Model (DEM) for realistic visualization in pedestrian navigation, especially important in hilly cities. Thus, road surface is partitioned into different geometric entities topologically integrated with the rest of urban elements in the database. The visualization process is performed using X3D as graphical language. The real street steepness and highly detailed facades can be easily distinguished, while the scene geometry is reduced enough to support interactive navigation and exploration.

Keywords: automatic 3D generation, DEM, spatial database, virtual urban environments, 3D web-based system

1. Introduction

In GIS the spatial data are managed in digital form as simplified description of complex reality that can be stored, maintained, visualized, analyzed or distributed. Spatial and nonspatial aspects of the reality are combined together providing a powerful mechanism to analyze geographic information to cartographers (Müller, 1995), city engineers and planners (Carsjens and Ligtenberg, 2007), geologists (Abdul-Rahman et al., 2006), etc.

GIS systems can also be used in urban environments as in (Köninger and Bartel, 1998). In fact, urban GIS are nowadays extensively used in cadastral offices (Stoter, 2004; Hassan et al., 2006), urban planning (Pullar and Tidey, 2001), Building Information Models (BIM), risk management or decision support systems. However, interoperating and sharing these heterogeneous data is still a challenging task in the applications of cadastral databases (Tong and Xu, 2005). An additional problem is the large amount of information that urban information systems usually manage, which can even include 3D objects. For these reasons, a method to efficiently manage 3D models is needed since three-dimensional real entities cannot be successfully handled in 2D systems (Emgard and Zlatanova, 2008).

Systems that combine the full GIS functionality and the 3D modeling capability are the so-called 3D GIS (Abdul-Rahman et al., 2006; Alias Abdul-Rahman, 2007). These systems must be able not only to store and visualize the objects in a 3D environment, but also to handle, manipulate and analyze them. Therefore, adding the third dimension cannot be considered as a simple extension of a 2D GIS because additional structures and modules must be created to support the new functions. Next the most important features are described.

First, managing the topology and topography of the data is essential. In a GIS system, topography considers the process to obtain a terrain representation that can be considered as a map, while topology determines the connections between the different elements in the space (points, lines, polygons or solid objects) (Alias Abdul-Rahman, 2007). Topology is especially important because the analysis or visualization of any urban location must properly combine the elements of different layers, describing the relationship between each object and its neighbors. For example, thanks to the topology, the nearest bus stop to a building can be obtained, or the associated streets to a crossroad. Therefore, a complex city model must include the ability to integrate topological and attribute information for full functionality.

With regards to 3D visualization, some CAD (Computer Aided Design) tools can obtain a high level of realism in visualization and interaction (Glander and Döllner, 2009; Wergles and Muhar, 2009; Musliman et al., 2006). Despite this, CAD

* Corresponding author. Tel.: +34 953 212477; fax: +34 953 212472.
E-mail address: mrobles@ujaen.es (M. D. Robles-Ortega).

ISSN: 1726-2135 print/1684-8799 online
 doi:10.3808/jei.201200211

tools do not provide the required solution to data structuring, thematic properties or spatial relationship, essential in earth science analyzes. For this reason, adapting some techniques associated to Computer Graphics could be interesting to create the 3D models and their topological relations at the same time. As explained bellow, the method to generate 3D entities described in Section 4.2 also obtains the topological relations, which will be stored in the geo-database explained in Section 5.

An additional interesting challenge associated to these systems is providing similar characteristics to desktop applications without losing interactive navigation and exploration. Thus, in a web-based GIS, users must be able to access, recovery, visualize and analyze spatial data (Zhou et al., 2004). To achieve this objective in our prototype, the geometry has been simplified and the expensive geo-spatial queries have been replaced with simpler ones which use the topological relations among the entities in the database.

Next we review the state of the art in Section 2 and the main objectives of our approach in Section 3. In Section 4 we describe the methodology for generating 3D geometric entities from 2D GIS layers, and their integration in the spatial database. The geo-database is described in Section 5, while the interaction process is explained in Section 6. Then, we analyze the performance and visualization results in Section 7. Finally, we explain our conclusions and future work.

2. Previous Work

Most of current commercial urban GIS applications suffer from lack of functionality in emerging areas such as tourism or navigation, as stated in (Abdul-Rahman et al., 2006). Their structure in most cases is composed of several layers with thematic and geometric information. These data can appear overloaded in the visualization process as a set of abstract 2D layers, and not necessarily connected in a database. Therefore, the limitations of the current 2D GIS are related to the third dimension and the data structures, as widely reported in the literature (Abdul-Rahman, 2007).

The 3D capability in urban modeling can be provided by CAD solutions. Thus, the process to generate the geometry could be automatized using some specific solutions like, for example, CityEngine (Watson et al., 2008). This semi-automatic tool for procedural 3D urban construction can achieve interesting results in visualization and automatic generation. However, as their achievements are focused on display and the data management is not considered, they are not valid for managing urban information systems. In addition, 3D modeling of cities using real data and DEM information has some problems as described in (Robles-Ortega et al., 2009).

The standard CityGML (OpenGIS® City Geography Markup Language), developed by OGC (Open Geospatial Consortium) (http://www.opengeospatial.org/) (Kolbe et al., 2005) can be used to manage the 3D urban entities and its relations. It is based on XML language and uses hierarchical structures to cover the geometrical, topological, and semantic aspects of 3D city models (Kolbe, 2009). This multi-scale model maintains five levels of detail, from the regional landscape of LOD-0 to the interior of architectural models in LOD-4. The resulting models are rich enough to incorporate DEM, vegetation, bridges and tunnels, etc. The system also combines Geo Web services and GIS capabilities. All these features and its standardized encoding and exchange format become CityGML in a reference model for urban models management. Nevertheless, the large size of these scenes is a drawback for web-based systems, which require simplified structured models to obtain a high level of interaction. The method proposed in Section 4, however, generates a simple geometry for all the entities, that can be easily transferred via the Internet.

Besides the geographical information management, another important feature for any urban GIS is the visualization. Nowadays, geobrowser tools offer easy access to geographical and map images (Chiang et al., 2011). Some of this kind of software can manage a great amount of information and even allow the navigation and interaction in a 3D urban environment (Conti et al., 2008). Two of the most known geobrowsers have been created by the Google® company: Google Maps and Google Earth. Thanks to these tools, using together with Google Street View (Miller et al., 2009), many cities in the world are currently accessible via web. The main goal of these programs is navigation, but there is additional information like names of streets, shops or touristic information. Although the virtual city is not a real 3D scene, the system uses 360º cameras that provide the sensation of being immersed in it by means of sequence of photographs. Nevertheless, in Google Earth the scene can include 3D models generated using Google SketchUp. The main disadvantage of this tool is that municipalities have no control over the database information and neither the shops or the rest of establishments. Moreover, the view updates or the addition of new information corresponds to Google company.

Urban scenes can also be renderized using languages which transmit 3D models through the Internet, such as Java3D (Huang, 2003; Xiao et al., 2009) or VRML (Zlatanova and Gruber, 1999). However, in order to make the application more extensible, a standard format as KML, COLLADA or X3D is desirable. Next these three possibilities based on XML are briefly described.

KML, Keyhole Markup Language, (http://www.opengeospatial.org/standards/kml) can be used to show geographical data in geobrowsers like Google Earth or Google Maps. It uses a tag-based structure with nested elements and attributes. COLLADA (COLLAborative Design Activity) (Arnaud and Barnes, 2006) defines an XML database schema that enables 3D authoring applications to freely exchange digital assets. It includes features such as geometry, animation, effects and multiple versions of the same resource. X3D (Brutzman and Daly, 2007), the last alternative, is the VRML successor to visualize 3D scenes in the Internet, as explained in Section 4.3. The models are structured using an scene graph, which is composed by a set of nodes. X3D can interoperate with other web technologies like Ajax or COLLADA.

Once the main tools to be used in a 3D GIS implementa-

tion have been explained, next we describe the most signifycant GIS systems in the literature. In 1998, for instance, Köninger and Bartel (1998) proposed a system to manage the 3D city model and its thematic information. This GIS provides an effective data storage and administration, with additional planning analysis functionality. Another interesting application is explained in (Brooks and Whalley, 2008). It consists on an 2D/3D hybrid system which integrates both 2D and 3D views of the data. In other cases, as in Verbree et al. (1999), virtual reality techniques are also used to improve the interaction with the 3D GIS.

3. System Specification and Objectives

In this section we describe the main goals of our application. As commented before, our prototype includes some of the most significant functionalities in a 3D urban GIS. In particular, our system can manage real 3D urban data not only for visualization, but also for navigation and interaction. The subject areas of our approach are: realistic 3D visualization, interaction and web capability. Next we describe the main objectives, depicted in Figure 1.

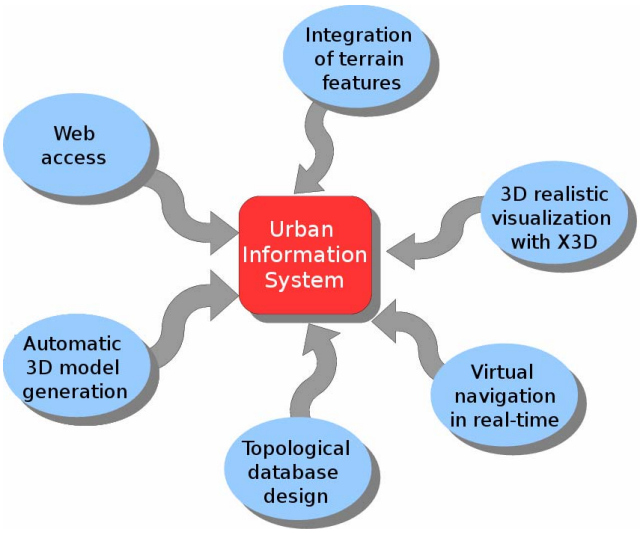


Figure 1. System features.

Regarding the first feature, realistic 3D visualization, we have implemented an automatic method to generate the 3D geometry of an urban scene from 2D GIS data (objective 1). In order to increase the realism of the resulting model, the approach also considers the DEM information as input data (objective 2). The 3D visualization (objective 3) is performed using the X3D language.

Users can move freely around the urban scene (objective 4) using two main navigation modes: pedestrian or flight. In order to allow the interaction with the objects in the 3D scene (objective 5), we have implemented a geo-spatial database to store both topological and thematic information about the urban entities, and a method to query it. As explained in Section 5, this approach is based on open-source technologies, and creates a communication process between the 3D model and the database.

Thanks to the reduced geometry of the scene and the geo-database, our system can be accessible from the Internet using a client-server architecture (objective 6).

Next we describe the main objectives:

1) Automatic generation from 2D GIS. Nowadays most cities have cadastral information on a Geographic Information System. For this reason, we have created an automatic method to generate a 3D urban model of a city using 2D GIS information as input data. Besides a simplified 3D geometry, this approach generates a set of topologically related entities which will be stored in the geo-database. In Section 4 we describe this process in detail.

2) Integrating DEM. Although city modeling has been widely considered in the literature, most of the proposed solutions assume that cities are located on a flat surface. Nevertheless, modeling real hilly cities as Jaén (in Spain) requires additional information to achieve realistic results. Our method uses the DEM information to assign correct heights to the different urban elements. This solution improves the classical TIN triangulation requiring less space and defining surface geometric entities that match with real entities for visualization and location of urban elements.

3) 3D realistic visualization. Similarities between real and virtual cities in a 3D GIS is an essential feature in order that users can identify the urban elements easily. In our application, the visualization process is performed using X3D, the ISO standard XML-based file format for representing 3D graphics in the web (Brutzman and Daly, 2007). The main reasons to choose this language are described in Section 4.3.

4) Navigation. A large environment needs an efficient navigation process to allow the movement and interaction along the whole scene. In our method, only the visible portions of the city are renderized. However, an approach to obtain the new visible elements is required since users can move freely around the scene. In our case, we have used the topological relations between the different urban entities stored in the database. Thus, we avoid expensive spatial queries based on proximity and distance calculation.

5) Spatial database interaction. Spatial data management is the most important functionality of GIS. It is desirable that a GIS system uses a geo-database to store the thematic, geometric and topological data of the city and its entities. This information must be accessible not only for the traditional forms or straight queries to the database, but also by means of the interaction with the 3D scene. For example, users must obtain a brief description or the opening hours after clicking on the model of a public organization building. Other spatial queries such as determining the bus stops in a specific street and the distance between them must be allowed as well. The design of the geospatial database for our application is described in Section 5.

6) As described above, this repository stores the information and the topological relations among the geometric entities (buildings, streets, etc.) and the rest of street furniture.

7) Client-Server architecture. Web-enabled GIS services have already shown great potential in relation to geo-information. For instance, cadastral information is now available for any Internet user. However, the third dimension cannot be considered sufficiently exploited by web applications. The difficulties are the same as those of 3D GIS and also those related to the network bandwidth and interactivity protocols. The major challenge is to achieve an interactive response and a high level of renderization in the client side. Our solution comes by means of drastic reduction of the overall geometry of the scene by defining specific geometric entities, and establishing appropriate client-server communication protocols as described in Section 6.

All the objectives described above could be obtained by means of different technologies. In this work, we propose a solution using open-source technologies. Next we describe the technical features of our method.

4. The Method for Automatic Urban Modeling

The process for automatically modeling large cities has promoted two main different strategies: using data from LIDAR technology (Sohn and Dowman, 2007), or from 2D GIS (Abdul-Rahman et al., 2006). The first approach is being extensively used due to the high accuracy of the resulting models and the level of automation recently reached (Poullis and You, 2009), even managing facade texturing (Frueh et al., 2005). The major drawback of this data acquisition method is the difficulty for determining topological relations from the point cloud data. In addition, the huge amount of information should be processed and drastically reduced to be supported by web-based systems. In contrast, urban GIS has the advantage of using existing accurate geo-databases, making possible the modeling automation from their thematic and geometric information. Two-dimensional data are transformed into 3D objects, and new entities are created as well as the required topology to provide connectivity. In this methodology some post-processing should also be necessary for higher performance, but this can be gradually carried out afterwards, by texturing facades or incorporating additional urban elements.

4.1. Definition of Geometric Entities

City modeling is a complex task due to the great amount of different urban entities that can be considered.

Footprints of buildings, blocks and sidewalks, as well as street polylines, are the only geometric input to the whole system. Streets, which are originally given as polylines in a 2D layer, must be converted into polygonal entities such as street stretches, crossroads and sidewalks which provide surface to the streets. Buildings are normally part of building blocks, and both entities are also provided as polygonal lines. Using these geometric 2D information and the rest of thematic fields of the input data tables, an entire 3D virtual environment is generated following a systematic process, with the entities described in Figure 2. The figure shows 2D entities, but they maintain height information to build the final 3D model. Crossroads and stretches are surface lacking entities obtained from the streets polylines of the cadastral GIS, but they are linked with three-dimensional points to determine their geometry in 3D.

Figure 2. Spatial entities of the database.

Next, we summarize these geometric entities, which will be stored in the geo-database:

• *Street*: is an original entity from a cadastral layer; the geometry is given as a set of sorted points representing a polyline. The name is also maintained, but there are no links between neighboring or intersecting streets.

• *Block building*: is obtained from a 2D GIS layer and consists of the outline block geometry and the cadastral code. The value of their altitude is assigned from the DEM. This entity is 3D-reconstructed for visualization.

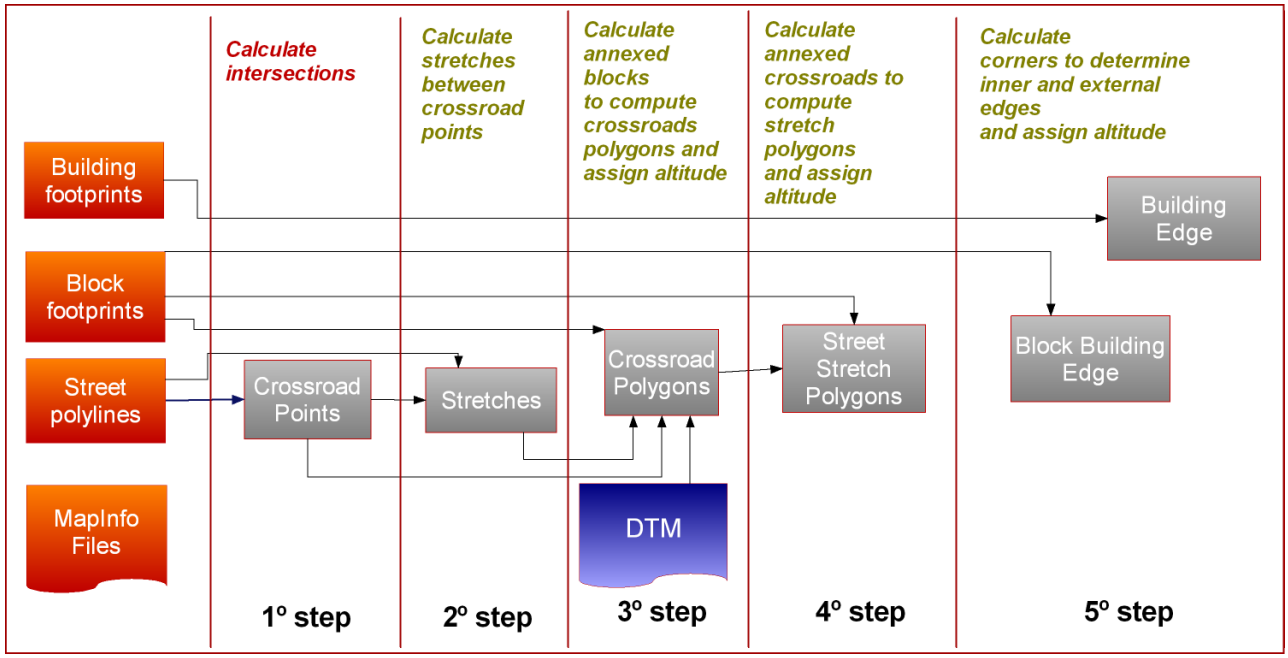


Figure 3. Phases in the geometric entities construction.

• *Block building edge*: individual edges of blocks are also maintained for being assigned with real heights in the geo-database.

• *Building*: as in the case of blocks, this entity is given as the outline geometry of a building and its cadastral code. This code maintains some correspondence between a block and the buildings inside.

• *Building edge*: individual building edges are kept apart to distinguish inner and outer sides of the building for future texturing.

• *Street stretch*: the portion of polyline between two consecutive cross-points is a street stretch. A block is totally surrounded by different street stretches.

• *Stretch polygon point*: the area representing a street stretch polygon consists of a set of vertices obtained from the stretch polylines. This region is limited by the neighboring blocks and the annexed crossroad polygons. The real inclination is obtained from the pair of annexed crossroad polygons (this value is already assigned). The union of all crossroads polygons and street stretch polygons generates the complete street surface. For an accurate visualization in X3D these polygons must not overlap each others.

• *Crossroad*: crossroads are the set of intersecting points among all the street polylines. These points are the seeds to generate crossroad polygons once neighboring blocks are computed. Firstly, the DEM is used to assign them altitude values, and afterwards the other entities take their respective altitudes.

• *Crossroad polygon*: the polygonal surface representing a crossroad is a crossroad polygon. The crossroad point lies into their associated crossroad polygon, and the altitude of this point is assigned to the whole polygon.

• *Sidewalk*: represents sidewalks polylines associated to blocks of buildings by linking sidewalk vertices.

• *Sidewalk polygon points*: each sidewalk vertex is stored to maintain different height to fit to the steep streets.

4.2. Geometric Entities Generation

Once the urban entities of the database have been defined, in this section we summarize the process described in Robles-Ortega et al. (2012) for their modeling. In real cities with thousands of buildings, an automatic methodology is required for performing this process in a reasonable time.

As input data our prototype uses the polygonal footprints of buildings and blocks, from a cadastral 2D GIS in MapInfo format (Daniel et al., 2001). This 2D geometry layer is used to build new entities in 2.5D. The height of buildings is an estimated value considering the real number of floors from the geodatabase. The result is a 2.5D object with similar shape to the original for most of the apartment buildings, particularly from a pedestrian perspective in which roof details are irrelevant. The Cathedral, as well as some other historic or architecturally significant buildings require CAD modeling process. The terrain altitude is an additional attribute assigned to buildings after combining with the digital terrain model (DEM). Facades texturization is the only process that should be manually applied for each building if a realistic view is required. Nevertheless, the method described in Robles-Ortega et al. (2009) using genetic algorithms for automatic texturing by combining facade cuts, can be extended to manage such facades in steeply sloping streets.

The three-dimensional city surface follows a more complex method to be automatically modeled from the 2D geometry layer. In fact, there is not any entity representing the streets geometry in the input data, except a set of polylines. These linked segments are the central edges of the free area between two

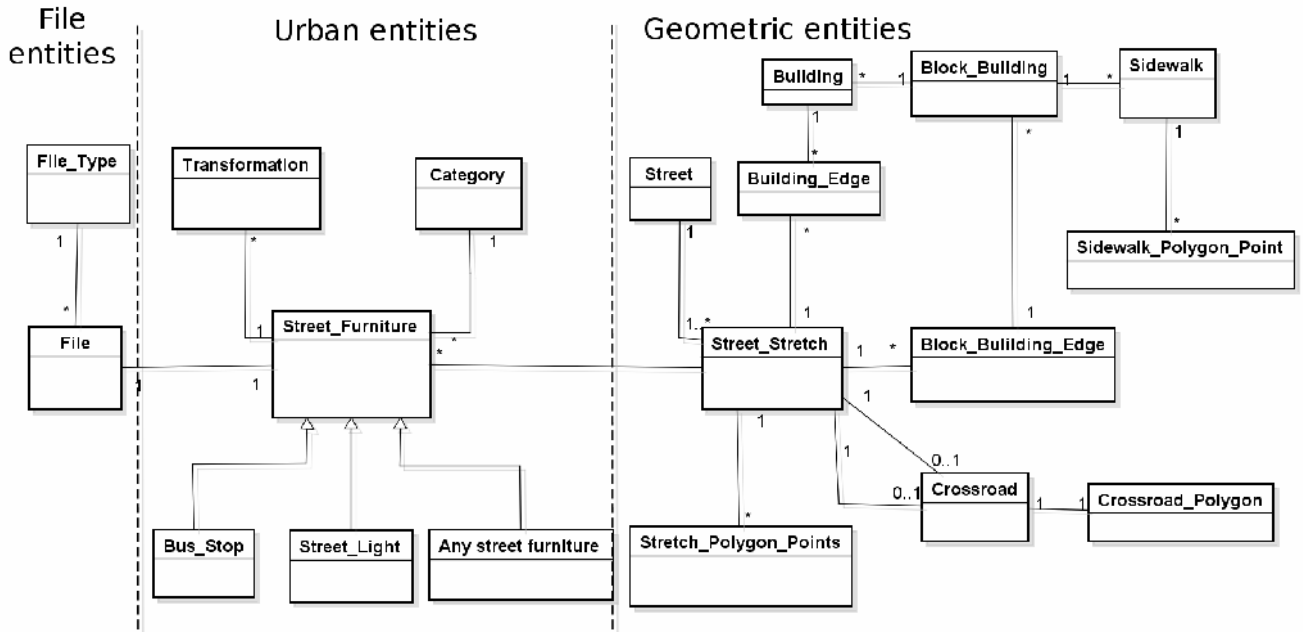


Figure 4. UML diagram of the DBMS model.

annexed blocks (see the Street picture in Figure 2). Street names are also associated to streets polylines as thematic information.

Most of the available approaches for modeling terrain surfaces use a triangular irregular network (TIN) or a tetrahedral network (TEN) (Said Easa, 1999). The visual result is normally appropriate for panoramic views or flight navigation. Nevertheless, a pedestrian walkthrough requires higher level of accuracy in the visualization, and triangulated surfaces of steep streets may show the triangles shape on the asphalt. Higher GIS resolution does not solve the problem, triangles just appear smaller. Furthermore, a TIN model is more expensive in terms of data storage, implying a lower performance in a client-server interaction.

The advantage of this new approach is not only the accuracy in the visualization. Geometric entities coincide with real world entities, then placing street furniture using this spatial database is straightforward. For instance, benches, street-lamps or phone booths are associated to particular streets, or more specifically to street stretches, instead of a triangle of a TIN. Geospatial and thematic data are then connected providing the database with the appropriate topology.

The phases for automatic generation of urban entities are summarized in Figure 3. New geometric entities have been obtained from 2D layers by means of computational geometry methods.

In the first step, street polylines are intersected to obtain crossroad points. A street stretch is the resulting polyline between two consecutive crossroad points (step 2). Next, crossroad polygons are computed as the polygon around a crossroad point whose vertices belong to the annexed blocks. Adjacency information is not part of the input data, what cannot be solved only using the proximity criterion. There can be buildings close to a crossroad point and however not be adjacent. Altitude is a value assigned to crossroad polygons by determining the nearest DEM cell to the crossroad point. In the fourth step the street stretch polygon is computed as the polygon formed by the nearest vertices of the two annexed crossroad polygons and the centroid of the adjacent blocks. The lack of overlapping between polygons is important in order not to detect vibration in the visualization during the navigation process. Edges of buildings and block buildings are easily obtained by decomposing in vertices the original polygons. The result is the city surface as 3D geometric entities, as depicted in Section 7.2.

4.3. Visualization Language

Visualization language is an important feature in a 3D urban model. For this application, a main requirement is that the 3D scene can be transmitted via the Internet. Next the most significant tools available at the moment are detailed.

WebGL is a web technology which includes 3D models in websites using HTML5 without any additional plugin (Behret al., 2009). Another possible alternative can be O3D, implemented on WebGL. However, this tool cannot visualize and interact efficiently with large models.

Google SketchUp is the proposal of Google Company Google®. The models generated with this tool can be integrated in Google Earth. Nevertheless, it does not consider sloping streets, and building location can cause non-realistic situations.

Traditionally VRML has been considered as a standard to transmit 3D models through the Internet. It has been replaced by its successor, X3D. This specification, developed by the Web3D consortium (http://www.web3d.org/x3d), increases the VRML functionality and can interchange data with other web technologies like Ajax and MySQL. Renderizing an X3D model in a browser requires a plugin installation. However, according to HTML5 specification, X3D nodes will be directly inserted in DOM content without any additional software. Some models have been developing to achieve this objective (Behr et

al., 2009). Moreover, according to OpenGIS® Discussion Paper with reference number OGC 09-104rx (Shilling and Kolbe, 2010), the Web 3D Service (W3DS) shall support X3D as mandatory format. A Web 3D Service (W3DS) is a portrayal service for three-dimensional geodata such as landscape models, city models, textured building models, vegetation objects, and street furniture.

For all these reasons, X3D has been chosen as rendering language for the urban environment described in this paper. All models have been tested using BS-Contact plugin.

5. Geo-database Design

GIS systems store large amounts of data that must be available to multiple users (Frank, 1988). For this reason, a database management system (DBMS) is essential to provide an efficient access to the information. In this section we describe the geo-database implemented to store all geometric and thematic data of the urban entities.

Initially, GIS systems were based on a dual architecture, with a relational DBMS to store the thematic information and a separate subsystem to save the spatial data. This dual architecture reduces the performance because an object with both thematic and spatial components has parts in both subsystems that are linked by a common identifier (Vijlbrief and Vanoosterom, 1992). Nevertheless, DBMS systems have significantly changed in the last several years (Zlatanova, 2006) and nowadays most of them allow the spatial data manipulation and storage. Thanks to this new functionality, the geo-databases could be implemented as platforms to integrate 2D maps, 3D geo-scientific models, and other geo-referenced data (Breunig and Zlatanova, 2011).

Among the different DBMS, we choose MySQL because it is open-source and supports spatial extensions. It is also compatible with web technologies like PHP. Moreover, OGC establishes SQL as an option to implement the simple feature access in the OpenGIS standard (http://www.opengeospatial.org/standards/sfs).

Our geo-database has three types of tables:file, geometric and urban elements. The first category includes the required tables to store the information about the different kind of files of the urban elements, while the second stores the spatial entities generated in Section 4 considering their relations. Finally, the third category includes the rest of urban elements like the street furniture (bus stops, street-lamps, etc.). Evidently, street furniture cannot be considered as an unique entity owing to the different existing classes. As a result, a new approach is needed to include new elements efficiently.

As can be seen in Figure 4, we have included two tables in our prototype in order to achieve this purpose: street furniture (which stores the common data to any urban object like position, description, etc.) and category (to classify the urban element in several classes). Although the first version of the geo-database considers street-lamps, bus stops and trees as categories, the system can be easily extended simply by including new entities which inherits from street furniture.

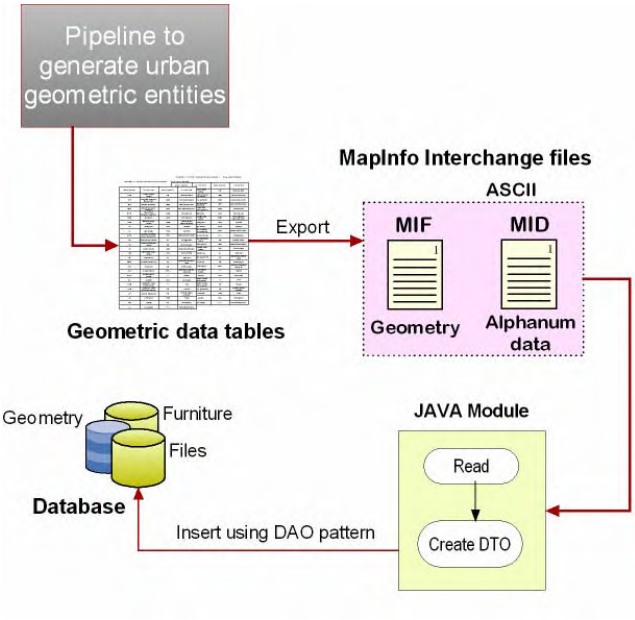


Figure 5. Data transformation from data tables to the databases.

Topology and neighboring relations are needed to connect geometric and thematic entities in the database. Thanks to this feature, the following spatial queries can be executed efficiently:

• Obtaining the nearest elements to a point. These objects are found after georeferencing the position and obtaining their associated street stretch. Then, the rest of urban elements can be obtained using a SELECT sentence.

• Determining the associated street furniture for a street. Users can fill an HTML form to indicate the street name, which will be used to obtain the street stretches. Afterwards, these entities will be used to query the urban elements.

• Accessing to additional information stored in the database about any urban element. For example, users can obtain the opening hours of a public building or the bus routes and timetables associated to a bus stop.

• Determining the new visible elements during the navigation process. This objective is obtained thanks to the interconnection between all elements. Thus, the known user position in the scene can be used to obtain the street stretch.

For instance, a query to find out the set of surrounding blocks and buildings to a given street stretch cST, requires the steps performed in Algorithm 1. First, the entity geomStretch with code cSTis retrieved from the geo-database. Next, the set of BLOCK_BUILDING_EDGE entities with codeST==cST is secured. Each of these block edges is associated to a block, then the set of surrounding blocks are also retrieved. As the pseudocode shows, once the blocks entities have been obtained, the buildings inside can also be directly accessed.

Algorithm 1 SelectAdjacentGeometry (cST)
1. SELECT STgeom

FROM STRETCH
WHERE codeST = cST **INTO** *StretchGeom*
2. Send *StretchGeom* for visualization
3. SELECT codeBB
FROM BLOCK_BUILDING_EDGE **WHERE** codST = cST **INTO** arrayBB
4. For i **IN** arrayBB.FIRST. arrayBB.LAST **LOOP do**
a. SELECT *Bgeom*
FROM BUILDING
WHERE codeBB = arrayBB(i)
b. SELECT *SWgeom*
FROM SIDEWALK
WHERE codeBB = arrayBB(i)
c. Send *Bgeom* and *SWgeom* for visualization
5. End_For

Another important feature in the urban element table is that any urban element can have associated more than one file. In the case of the 3D models, the original file contains only the geometry, while the location data are stored in a different table. Therefore, 3D models can be reused, which reduces the size of the scene. In X3D this solution is implemented using the tags DEF and USE.

Once the geo-database has been defined, we explain the process to include new data. As depicted in Figure 5, the results of the approach proposed in Section 4 are stored in MapInfo tables. These tables are exported to the ASCII interchange format MIF/MID, which is well supported by almost all GIS products. Then, this information is the input to a Java module which transforms these table-structured data into MySQL format. We use the DAO/DTO pattern (Anderson and Anderson, 2002) to allow the separation of detailed operations for managing the database and the rest of the system management.

6. The Interaction Process

Integrating web 3D applications and geo-databases for interaction is an important challenge in 3D urban information systems. Nowadays there is not a standard solution to this problem using open-source software and protocols (Behr et al., 2009). We describe the procedure to achieve such an objective joining several technologies, programming languages and open source libraries.

Our prototype follows a client-server architecture in which the client device visualizes and interacts in an urban virtual world, while the server provides the geometric and thematic information. In the next subsections we describe the protocols for retrieving and visualizing the scene information (urban geometry and street furniture).

6.1. Interaction with Urban Geometry

The process for retrieving and visualizing the urban geometry follows the scheme of Figure 6. First, the user can fill a HTML form to determine the urban elements to visualize (also a simple 2D city map can be clicked at any position). For example, we introduce the name and number of a street. This position is transferred to the server and the portion of surrounding three-dimensional city is displayed, considering this point as the position in which a virtual pedestrian visualizes the scene. We have used an own algorithm based on polar diagram tessellation to select the visible buildings, described in (Robles-Ortega et al., 2009). The city sector is then reconstructed in an X3D file, and finally transferred to the client device. The visualization corresponds significantly with the reality in the shape of streets and buildings.

Once the request is sent by the client, an Ajax module opens a Servlet, a Java object that receives a request and generates the appropriate response. The DTO (Data Transfer Object) is used in conjunction with the DAO (Data Access Object) to retrieve each of the entities obtained from the SELECT statements described in Algorithm 1.

The geometric data provided by the database must be transformed into an X3D file which is sent to the client-side using Ajax. The advantage of using this Ajax module is the asynchronous communication with the server in the background without interfering the visualization process and increasing the interactivity with dynamic content.

6.2. Interaction with Street Furniture

Certain urban elements are sensible to users' interaction by generating a database query. The general scheme of this interaction process is depicted in Figure 7. For instance, if a bus stop panel is clicked, a window in the client side is shown with the bus routes and timetables in a straightforward way (step 3 in the figure). To conduct such communication and data interchange between the three-dimensional model in X3D and the MySQL database, the following sequence of steps are performed: 1) inclusion of the X3D model in a website; 2) generation of X3D events and processing with external JavaScript code; 3) database access with JavaScript code; 4) three-dimensional visualization in the website.

Next, we describe in detail each of these procedures.

X3D Model Inclusion in the Website: As commented before, X3D has been chosen as visualization language. In particular, we have used BS-Contact, a rendering plug-in and client software that performs the interactive visualization of virtual 3D objects. When the user clicks on a point position in the initial city map, the server generates an X3D file representing the virtual 3D model around this position.

Event utilities and scripting in X3D: X3D files use scene graphs for representing objects of the virtual world, their shapes, textures and relations. The hierarchical scene graph is composed of nodes, which are divided into different fields for additional information storage. The communication among the nodes in the scene is performed using a message-passing mechanism by sending events. Sensors and interpolators are special types of nodes that generate these events and that are connected with the rest of the nodes of the scene via ROUTE statements. There are different ways to generate an event: for example, touch sensors are activated when clicking on specific scene objects. Proximity sensors generate events when the avatar is next to objects, and time sensors are specially interesting in animation.

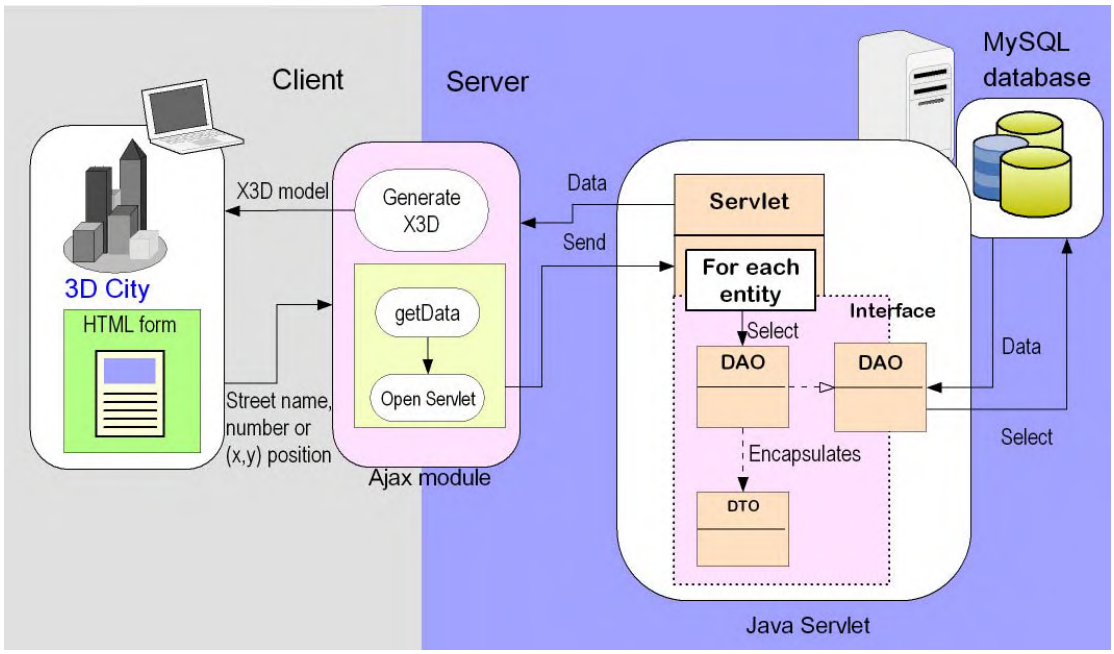


Figure 6. Interaction process for urban geometry.

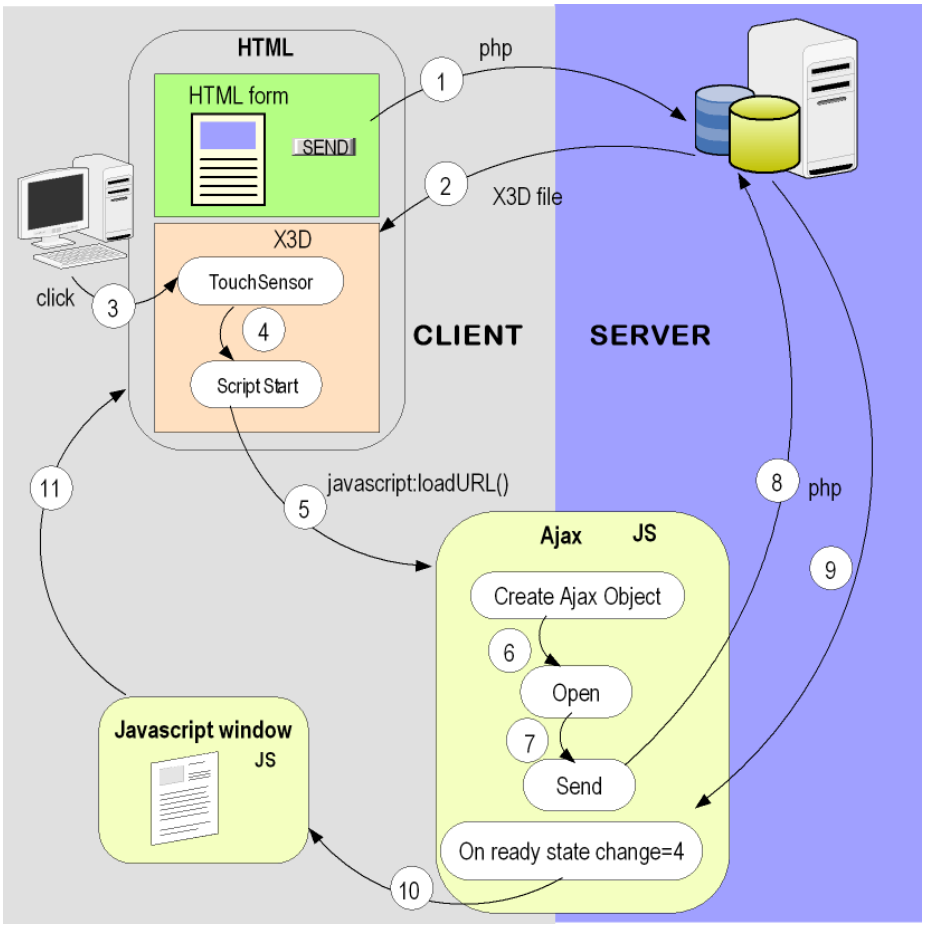


Figure 7. Interaction process for urban furniture.

In our application we use touch sensors to create an event when the user clicks on objects such as the timetable panel. As this node is connected with a script, an external JavaScript function is executed in order to query the geo-database (steps 4 and 5 of Figure 7).

Database access: After clicking sensible virtual objects in the scene like the bus stop, or after filling a HTML form, the geo-database must be queried. As for security reasons it is not allowed to directly access to the database, the client sends a JavaScript function to the server side. An intermediate mechanism is responsible for this communication process using Ajax technology (Crane and Pascarello, 2005) and PHP code (Welling and Thomson, 2008) (steps 5 ~ 9). In web applications, Ajax can retrieve data from the server asynchronously in the background, making possible an increase in interactive or dynamic interfaces on web pages. PHP is a scripting language executed in the server allowing the integration of MySQL sentences for accessing to the database.

Three-dimensional visualization: Finally, the data obtained in the previous stage is renderized. In our example a JavaScript window with the bus timetable information (steps 10 and 11) is shown. JavaScript is compatible with any browser and the interaction has been tested with Internet Explorer and Firefox.

6.3. Comparing with Other Client-server Systems

In our system, the client application requests desired information from the web server, which obtains it through queries in the geo-database. This process starts when the user selects any of the clickable objects in the X3D scene. As commented previously, the functionality can be increased by adding new clickable elements or creating sensors for the existing objects. This application fulfills the goals described in Section 3 and it is implemented using open-source technologies. It also obtains good performance and visualization results.

Another possible architecture for our system is the proposed by OGC. The OGC WMS specification (http://www.opengeospatial.org/standards/wms) offers a standard client-server interaction protocol in which each map server has a common interface for accepting requests and returning responses. Thus, an OGC web map server implements three functions: GetCapabilities, GetMap and GetFeatureInfo. The first function provides the client with a map server's service metadata, specifying its capabilities. The GetMap function specifies map request parameters that enable the client to request an image map. Finally, the GetFeatureInfo function allows the client to request more information about features at a specific location in the map. Some examples of web map servers implemented using this standard can be found in the literature (Ding et al., 2002).

This standard is very useful because it allows the interaction of any client with any server that implements the WMS service. In this initial stage of our work, we have not used this standard since we have focused on creating an efficient and easy interaction process with the X3D scene using open-source technologies. In future work we want to increase the functionality of our application to fulfill this specification.

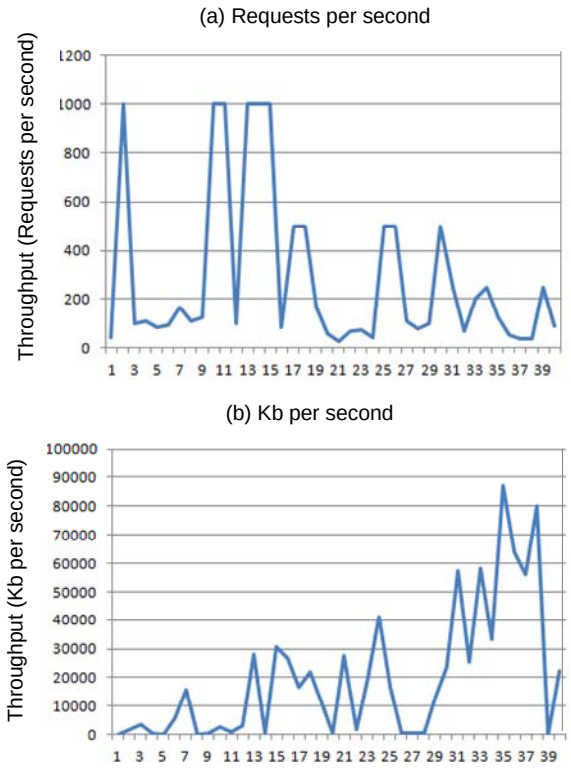


Figure 8. Performance results.

This change is not difficult since both GetMap and GetFeatureInfo functions are similar to our interaction process described above. In any case, we should include a GetCapabilities procedure to implement the full interface.

7. Performance and Visualization Results

In this section we describe the results of our application regarding to performance and visualization.

7.1. Performance

A high performance is essential to obtain an efficient web-system. For this reason, we have tested the performance of our application using Apache JMeter (http://jakarta.apache.org/-jmeter/). Specifically, the number of request and the data transmission per second have been recorded during an execution of our system. Results are summarized in Figure 8.

When the web-system is loaded at first, the entire X3D scene must be downloaded. Therefore, the number of requests from the client to the server must be higher, as can be seen in Figure 8a. Nevertheless, once the scene has been loaded, only the queries to obtain additional data are required.

Besides the number of requests, it is also important evaluating the data transmission. Figure 8b shows the results of our test for this feature. In general, the network traffic is reduced, apart from two peaks during the seconds 35 and 38, which is owing to the transmission of an image queried by the user.

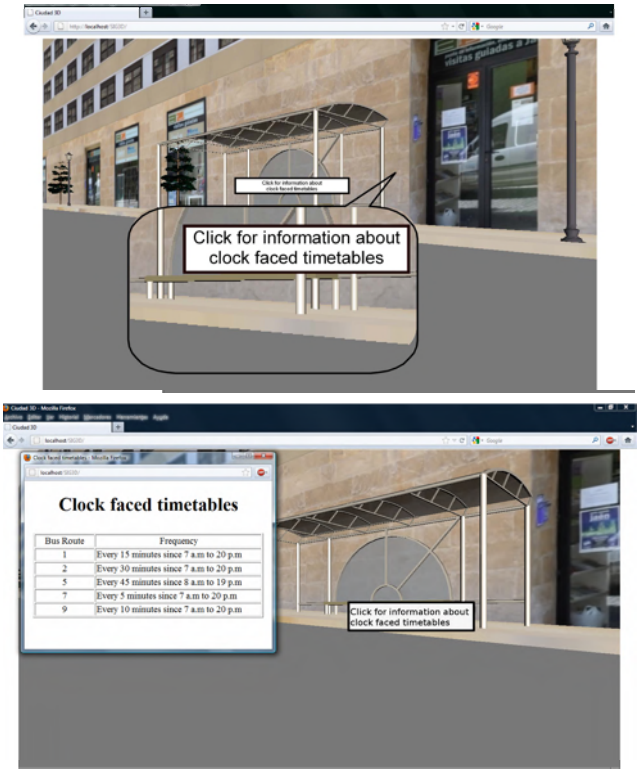


Figure 9. Interaction when clicking the bus panel.

Figure 10. City top views.

The rest of the time the amount of data is small, even when the X3D scene is downloaded. This shows that the reduced size of the scene generated with our method improves the performance of the web-system.

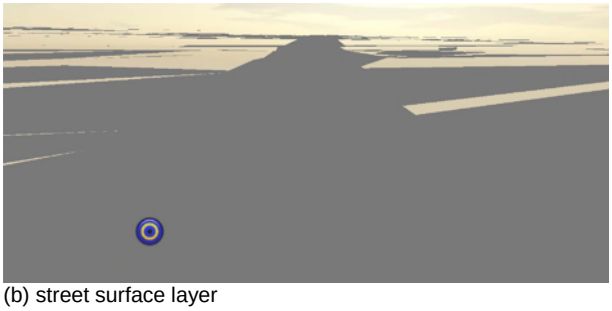
Definitely, our application obtains a good performance result which allows an interactive navigation and exploration of the scene. Next we describe the results regarding the realism of the models.

Figure 11. Street slope in hilly cities.

7.2. Visualization

The greatest challenge in 3D urban systems developing is to make the visualization of spatial data as realistic as possible, allowing at the same time the interaction with such information. In our prototype certain urban elements have become sensitive to the interaction as depicted in Figures 9a and 9b. During the navigation process the bus stop canopy shows a notice announcing that the timetable is available when the mouse is over. After clicking on it, an info window appears on the client device screen. The process follows the scheme of Figure 7 using Ajax

(a) Building layer



(b) street surface layer

Figure 12. Visualization of different layers.

technology for client-server data transfer.

The system has lots of fields of application, specially for Internet users. This example is clearly useful in a tourist portal. The information associated to urban elements can be easily modified in the server side. Since this approach holds interaction with any entity in the geo-database, possibilities of applications are then endless.

The navigation that has prevailed in this project is thought for pedestrians, since it allows a more realistic perspective for a potential visitor of the city. However navigation in flight mode as shown in Figures 10a and 10b is also possible, even in a web-based system. The simplicity of the scene geometry allows fast transfer of geometry with no need of managing different geometric LOD levels.

Although the prototype generation follows an automated process, the visualization achieves high level of realism, like the resulting images show. The slope of the street corresponds exactly to the reality. Street surface, sidewalks and buildings are fully integrated. In Figures 11a to 11d some snapshots show the asphalt tilt along the buildings of the street. DEM information determines the terrain slope, and the cadastral database the number of floors in buildings. The applications on urban planning and decision making on real cities are straightforward, for example the visual impact that involves the construction of a building with many floors.

Despite the geo-database is strongly connected, the layers visualization can be simulated as shown in Figures 12a and 12b, where buildings and street surface appear in different views.

8. Conclusions and Future Work

3D GIS is a powerful tool for visualizing and managing urban data. In this paper we propose some techniques and protocols to make possible most of its functionality. The 3D city model is automatically generated from 2D GIS data with little effort. Navigation and interaction are ensured in web-based systems thanks to a drastic geometry reduction: buildings are 2.5D objects and the city surface is formed by non-overlapping polygons. In addition, this road surface is assigned with real inclination for realistic virtual navigation. During the navigation process new geometry and thematic information is retrieved from the geo-database as the movement occurs. This process avoids expensive spatial queries since the concept of adjacent or attached entities is present in the database.

The only process that is not fully automated is the texturing. A previous work manages this automation in flat cities (Robles-Ortega et al., 2009). However, placing facades in sloping streets should consider some additional requirements For instance, results are non-realistic if a gate is located above ground level and does not allow the natural entrance to the buildings, or if a front door or a frieze are crossed by the asphalt street. This texturization task is going to be developed in the next future.

Our prototype is faster and lighter than other systems in the literature. For example, our geometry is simpler than CityGML standard, which is more appropriate for web-systems.

We want to improve the algorithm to select the set of visible buildings from a position by considering the building occlusion and the height of the terrain. Thus, this method will obtain the exact set of visible buildings from a point of view.

The field of application of this prototype can be easily extended by simply feeding of the database or adding touch sensors to any other entity. Starting from 2D layers, the described methodology allows a quick and economic generation of 3D interactive systems for any municipality.

Thanks to the features described above, our system obtains a high performance transmitting realistic 3D scenes. Nevertheless, our approach is not compatible with other map servers. To avoid this problem, we want to fulfill the OGC standard for the next versions of our system. We also want to use the CityGML standard to store our models, since this representation of city models considers the geometrical and topological relations or urban elements, and also its graphical representation.

Acknowledgments. This work has been partially supported by the Consejería de Innovación, Ciencia y Empresa of the Junta de Andalucía Spanish, the Ministry of Education and Science and the European Union (via ERDF funds) through research projects TIN2011-25259 and P07-TIC-02773.

References

Abdul-Rahman, A., Zlatanova, S., and Coors, V. (2006). *Innovations in 3D Geo Information Systems, First International Workshop on 3D Geoinformation*, Springer-Verlag, New York. http://dx.doi.org/10.1007/978-3-540-36998-1

Abdul-Rahman, A., and Pilouk, M. (2007). *Spatial Data Modelling for 3D GIS*, Springer-Verlag New York, Inc., NJ, USA. http://dx.doi.org/10.1007/978-3-540-74167-1

Anderson, P., and Anderson, G. (2002). *Enterprise JavaBeans*

Components Architecture: Designing and Coding Enterprise Applications, Prentice Hall Professional Technical Reference.

Arnaud, R., and Barnes, M. C. (2006). *Collada: Sailing the Gulf of 3d Digital Content Creation*, Peters, A.K./CRC Press.

Behr, J., Eschler, P., Jung, Y., and Zöllner, M. (2009). X3dom: a dom-based html5/x3d integration model, in 'Web3D' 09, *Proc of the 14th International Conference on 3D Web Technology*, ACM, New York, NY, USA, 127-135.

Breunig, M., and Zlatanova, S. (2011). 3d geo-database research: Retrospective and future directions. *Comput. Geosci.*, 37(7), 791-803. http://dx.doi.org/10.1016/j.cageo.2010.04.016

Brooks, S., and Whalley, J.L. (2008). Multilayer hybrid visualizations to support 3d gis. *Comput. Environ. Urban*, 32(4), 278-292. http://dx.doi.org/10.1016/j.compenvurbsys.2007.11.001

Brutzman, D., and Daly, L. (2007). X3D: *Extensible 3D Graphics for Web Authors*, Morgan Kaufmann.

Carsjens, G.J., and Ligtenberg, A. (2007). A gis-based support tool for sustainable spatial planning in metropolitan areas. *Landscape Urban Plan.*, 80(1), 72-83. http://dx.doi.org/10.1016/j.landurbplan.2006.06.004

Chiang, G.-T., White, T.O., Dove, M.T., Bovolo, C. I., and Ewen, J. (2011). Geo-visualization fortran library. *Comput. Geosci.*, 37(1), 65-74. http://dx.doi.org/10.1016/j.cageo.2010.04.012

Conti, G., Andreolli, M., Piffer, S., and de Amicis, R. (2008). A 3d web based geographical information system for regional planning, *WEBIST: Proc. of the fourth International Conference on Web Information Systems and Technologies*, 2, 155-160.

Crane, D., and Pascarello, E.W.D.J. (2005). *Ajax in Action*, Manning.

Daniel, L., Loree, P., and Whitener, A. (2001). *Inside MapInfo Professional*, OnWord Press, Santa Fe.

Ding, H., Pascoe, R., and Churcher, N. (2002). Implementing OGC Web Map Service Client Applications Using JSP, JSTL and XMLC, *Proc. of SIRC-The 14th Annual Colloquium of the Spatial Information Research Centre*, New Zealand.

Emgård, L., and Zlatanova, S. (2008). Design of an integrated 3D information model, in Coors, Rumor, Fendeland & Zlatanova (eds.). *Urban and Regional data Management, Taylor and Francis Group*, London, 143-156.

Frueh, C., Jain, S., and Zakhor, A. (2005). Data processing algorithms for generating textured 3d building facade meshes from laser scans and camera images. *Int. J. Comput. Vision*, 61(2), 159-184. http://dx.doi.org/10.1023/B:VISI.0000043756.03810.dd

Glander, T., and Döllner, J. (2009). Abstract representations for interactive visualization of virtual 3d city models. *Comput. Environ. Urban.*, 33(5), 375-387. Geo-information Generalisation and Multiple Representation. http://dx.doi.org/10.1016/j.compenv urbsys. 2009.07.003

Hassan, M.I., Abdul-Rahman, A., and Stoter, J.E. (2006). Developing malaysian 3d cadastre system-preliminary findings. *Innovations in 3D GEO Information Systems*, 519-533. http://dx.doi.org/10.1007/978-3-540-36998-1_40

Huang, B. (2003). Web-based dynamic and interactive environmental visualization. *Comput. Environ. Urban.*, 27(6), 623-636. http://dx.doi. org/10.1016/S0198-9715(02)00063-7

Kolbe, T.H., Gröger, G. and Plümer, L. (2005). L.: Citygml â "interoperable access to 3d city models. *Proc. of the first International Symposium on Geo-Information for Disaster Management*, Springer Verlag, 21-23.

Kolbe, T.H. (2009). *Representing and exchanging 3d city models with citygml*. In Lee, J. and Zlatanova, S., editors, 3D Geo-Information Sciences, Lecture Notes in Geoinformation and Cartography, 15-31 .

Köninger, A., and Bartel, S. (1998). 3d-gis for urban purposes. *GeoInformatica*, 2(1), 79-103. http://dx.doi.org/10.1023/A:1009797106866

Miller, F.P., Vandome, A. F., and McBrewster, J. (2009). *Google Street View*, Alphascript Publishing.

Müller, J. (1995). TI-GIS cartography: visual decision support for spatio-temporal data handling. *Int. J. Geogr. Inf Syst.*, 9(6), 637-645. http://dx.doi.org/10.1080/02693799508902061

Musliman, I.A., Abdul-Rahman, A., and Coors, V. (2006). 3d navigation for 3d-gis-initial requirements. *Innovations in 3D GEO Information Systems*, 259-268. http://dx.doi.org/ 10.1007/978-3-540-36998-1_20

Poullis, C., and You, S. (2009). Automatic reconstruction of cities from remote sensor data, *CVPR09*, 2775-2782, 2009.

Pullar, D.V., and Tidey, M.E. (2001). Coupling 3d visualisation to qualitative assessment of built environment designs. *Landscape Urban Plan.*, 55(1), 29-40. http://dx.doi.org/10.1016/S0169-2046(00) 00148-1

Robles-Ortega, M.D, Coelho, A., Sousa, A., and Ortega, L. (2009). Modelling a virtual urban environment with realistic terrain features, *Proc. of 17 Encontro Português de Computação Gráfica*, Covilhã, Portugal, 313-314.

Robles-Ortega, M.D., López, J., Ortega, L., and Feito, F. (2009). Texturización automática en entornos urbanos utilizando algoritmos genéticos. (automatic texturization of urban environments using genetic algorithms), *Congreso Español de Informática Gráfica (CEIG)*, 133-144.

Robles-Ortega, M.D, Ortega, L., and Feito, F (2009). An Exact Occlusion Culling Method for Navigation in Virtual Architectural Environments. *IV Iberoamerican Symposium in Computer Graphics (SIACG)*, 23-32.

Robles-Ortega, M.D, Ortega, L., Coelho, A., Feito, F., and de Sousa, A. (2012). Automatic Street Surface Modeling for Web-Based Urban Information Systems. *J. Urban Plann. Dev.* http://dx.doi.org/10.1061/(ASCE)UP.1943-5444.0000131

Said Easa, Y.C. (1999). *Urban Planning and Development Applications of GIS*, American Society of Civil Engineers.

Shilling, A., and Kolbe, T.H. (2010). *Draft for Candidate OpenGIS® Web 3D Service Interface Standard*, OpenGIS® Discussion Paper. Ref. number: OGC 09-104rx

Sohn, G., and Dowman, I. (2007). Data fusion of high-resolution satellite imagery and lidar data for automatic building extraction. *ISPRS J. Photogramm.*, 62(1), 43-63. http://dx.doi.org/10.1016/j.isprsjprs.2007.01.001

Stoter, J.E. (2004). *3D Cadastre*. PhD thesis, Delft University of Technology.

Tong, X.H., and Xu, G.S. (2005). Modeling Cadastral Spatial Features Based on Geography Markup Language in GIS: A Case Study in Shanghai. *J. Environ. Inform.*, 6(2), 103-110. http://dx.doi.org/10.3808/jei.200500060

Verbree, E., van Maren, G., Germs, R., Jansen, F.W., and Kraak, M.-J. (1999). Interaction in virtual world views-linking 3d gis with vr. *Int. J. Geogr. Inf. Sci.*, 13(4), 385-396. http://dx.doi.org/10.1080/136588199241265

Watson, B., Müller, P., Veryovka, O., Fuller, A., Wonka, P., and Sexton, C. (2008). Procedural urban modeling in practice. *IEEE Comput. Graph.*, 28(3), 18-26. http://dx.doi.org/10.1109/MCG.2008.58

Welling, L., and Thomson, L. (2008). *PHP and MySQL Web Development* (4th Edition), Addison-Wesley Professional.

Wergles, N., and Muhar, A. (2009). The role of computer visualization in the communication of urban design-a comparison of viewer responses to visualizations versus on-site visits. *Landscape Urban Plan.*, 91(4), 171-182. http://dx.doi.org/10.1016/j.landurbplan.2008.12.010

Xiao, D., Xiao, Y., Lin, H., Fu, X., Xu, L., and Yang, J. (2009). A Service Stack for 3D Visualization of GIS Based Urban Pipe Network, *Proc. of International Forum on Information Technology and applications*, 1, 342-346.

Zhou, G., Tan, Z., Cheng, P., and Chen, W. (2004). Modeling and visualizing 3d urban environment via internet for urban planning and monitoring. *International archives of photogrammetry remote sensing and spatial information sciences*, 35(2), 341-346.

Zlatanova, S., and Gruber, M. (1999). 3d urban gis on the web: Data structuring and visualization, *ISPRS Commission IV Syposium on GIS-Between Visions and Applications*, Stuttgart, Germany, 32(4), 691-699.