Database and Modelling Strategy: A Compliant Way for Display Optimisation

Nathalie Farenc, Frederic Sidler, Albert Ferrando and Daniel Thalmann
Computer Graphics Laboratory, Swiss Federal Institute of Technology, Lausanne, Switzerland
Nathalie.Farenc, Frederic.Sidler, Daniel.Thalmann @epfl.ch

In order to display huge scenes with virtual human inhabitants evolving inside a virtual city, we propose a methodology to create and manage different Level of Detail for a well-segmented scene without "re-meshing" the scene during simulation. A database dedicated to urban life simulation providing data for planning human actions and behaviour is completed in such a way to furnish also information about the most adapted displayed representation. In this article we propose a methodology for scene modelling. The scene is analysed and completed in order to create a database containing information needed for urban life simulation. Designers need and create rules for scene creation and some tools for database visualisation and manipulation are presented in this article.

Keywords : **database, modelling strategy, simulation, urban scene, optimisation, LOD.**

1. Introduction

As visual computer science is growing up, the visual scene structures are more and more complex and their sizes tend also to take more and more space memory. This problem can be partially handled using different optimisation strategies. The first one is to simplify the model of the scene by reducing the vertices number and/or the texture number and size. The display can be also optimised using hidden part of the scene, with Z-buffer technology [Catmull(1978)] e.g. The last option is to have different representations of the same scene according to a relative location of the camera (point of view in the scene). This is the level of detail concept (LOD) as existing in the VRML standard. The user builds his scene using various versions of a part of the scene according to the optimum camera distance. These tools and methodologies are sometimes not efficient enough. For simulations of virtual humans in a city we have very large scenes and the memory to display polygons and textures is dedicated both for the scene and for the virtual humans. Using a database associated to the scene for urban human behaviours [Farenc(1999)] providing geometrical data, we want to improve the simulation display. Our methodology is based on LOD concepts and a hierarchical scene decomposition in order to create the most adapted scene to our simulation.

1.1 Related Work

The field of LOD research is active and we can find several nice works generating different LOD for a polygonal model during a simulation. There are a lot of methods to refine a mesh model during or before its display. A progressive mesh method presented by H. Hoppe [Hoppe(1996)], or simplification of envelopes by J. Cohen [Cohen(1996)] are some examples of such technology. Applied on terrain surface with quadtree techniques TerraVision project [Reddy(1998)] adds a new layer above mesh representation attaching complementary information. Other techniques mix three-dimensional models with simplified representation for distant part of a scene using images in background [Aliaga(1999)]. Our approach is to have static LOD for the scene and to select the most adapted one according to the point of view. The scene is the surrounding sets for inhabited urban environment and we adopt the strategy to minimise time computation for its display, even if a lot of pre-processing design has to be done. The database attached to the environment provides useful information in order to improve our LOD management for the scene.

1.2 Overview of the Methodology

In this paper we present a complete scene construction in order to perform efficient display. All the steps in this process are aimed to offer users the possibility of modifying the scene composition for better ulterior display optimisation. The scene creation step puts labels on objects in order to create an associated database. Moreover, the designer applies rules of construction such as embedded files in order to allow modification of the scene composition after the design phase [Thalmann(1999)]. The second step concerns an automatically database construction using objects labels presented in the analysed scene. The database is mainly dedicated to provide data for simulation of urban human behaviours. This includes geometrical data such as objects to interact or to avoid collision with, and semantic notions such as sidewalks are areas for pedestrian mobiles [Farenc(1999)]. The database carries also all the files composing the scene. In this work we add a new layer to inform users about the displayed files used in the scene, to create different scene versions according to a scenario of the camera's displacement.

1.3 Organisation of this paper

In the first section we present the database linked with a scene and more precisely the spatial decomposition of the scene into entities composing the database. The second part describes the scene modelling process and the attached constraints in order to provide relevant data for the database construction. The section three presents the visual tool associated with the database allowing visualisation and interactions with users in order to create LODs and some scene plans according to camera locations. The last section concerns the conclusion and future work.

2. A virtual city database

What is required for virtual human behavioural simulation in a complex environment like a city? Bearing this question in mind, we can assume that urban displacements and behaviour depend to a large extent on geometrical data and urban features knowledge. For us an urban environment is a place where information (semantic and geometric) is dense and can be structured using rules. The notion of urban knowledge encloses urban structural information and objects useable according to a set of conventions [Farenc(1999)].

2.1 A Hierarchical Decomposition

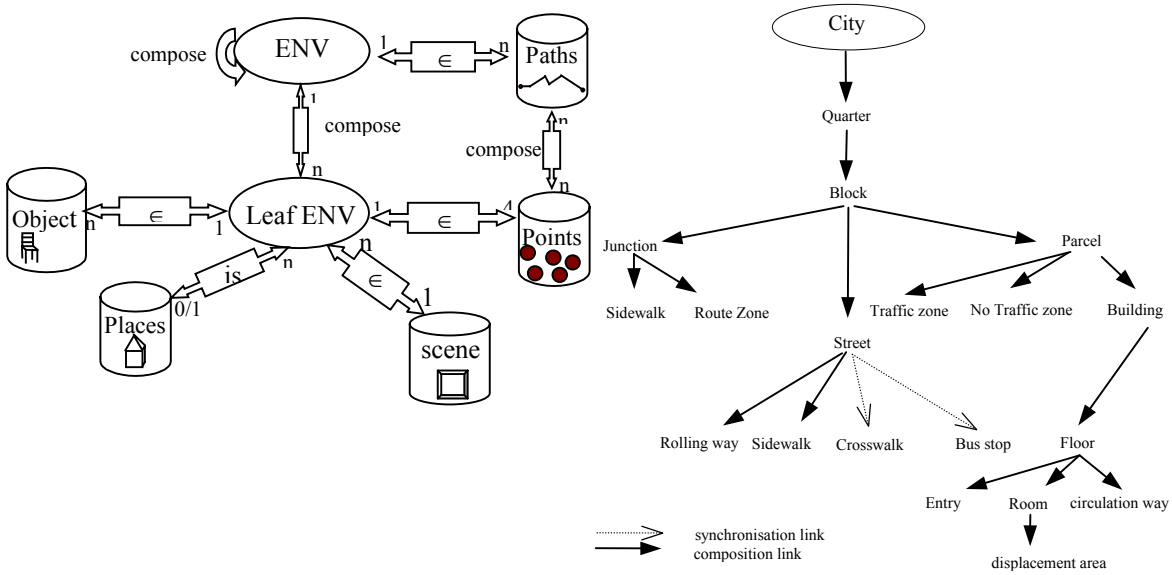


Figure 1 – The database schema (left) and the city spatial hierarchical decomposition (right)

The semantic attached to a place defines a space decomposition of the scene. For example, in a city, we can denote sidewalks as areas for pedestrian entities, so that in such a place they may walk

or have social encounters [Doyle(1997)]. The city database has to inform all “mobiles” (objects with mobility such as pedestrians, cars, buses, and bicycles) where they may circulate. How might one organise them to access information efficiently? Our approach is to define areas. The areas can be subdivided into sub-areas, or grouped, depending on the ‘level of information’. This decomposition tidies up and links the semantic data. The city is decomposed into several areas, depending on their geographical and functional properties. At each level in the environmental representation, the accessible information corresponds to a level of abstraction. At a town level we are interested in knowing the name of different quarters and not the position of the next pedestrian crossing. Specific areas of the scene are defined as “Environment Entities” (*ENV*) because they possess geometrical and semantic information. Figure 1 (left) presents the city decomposition into “quarters”, the quarters are decomposed into block levels. We define a block as a part of a quarter, composed of streets, junctions and “parcels”. Junctions are crossroads that connect streets. We consider as a parcel a portion of land that has no streets or junctions inside, like a park or a piece of land with buildings. In this way, all the space covering a block can be marked out and all points on the surface of a block can be associated with an ENV. The database stores these ENV and links the ENVs with other types of information (i.e. paths to cross the scene, objects for collision avoidance, file names composing the scene) according to the draw in Figure 1 (right). This set of data, available through request, are used for the action/path planning and the display optimization.

2.2 The Database Creation

The database is able to provide different services to applications that need information about the environment such as behaviour for virtual humans or the names of the visualisation files containing the part of the scene observed for a display optimisation. The scene and its internal hierarchy is analysed, then the objects names and the labels found are used to create the database. The database is based on a set of ENVs structured in a hierarchy and containing different levels of information. The designers can have previously associated some semantic notions to the generic model of ENV and the database can be automatically more completed during the construction. The information stored is about the scene's geometry, the location of objects but also location of the places, and information inherent to the structure of the city e. g. "here is a location on a sidewalk, belonging to a street and after a quarter". Analysis of the scene provides the main part of the database. The database contains all the ENVs with the links between ENVs and the files list composing the scene. These files can be embedded in order to construct the scene using reusable modules (see section concerning the scene modelling). Labels are attached to the different parts of the scene (surfaces) on the basis of a generic model informing the scene according to the simulation to perform.

2.3 A Generic Model: The database Schema

This model of hierarchical decomposition is defined in a generic way before the database creation stage. We chose this decomposition in view of the simulation we have in mind, with autonomous humans in a virtual city. For other types of simulation, other semantics may be more appropriate. For this, a script describes the decomposition model in a file defining a set of prototypes, the ENV prototypes. Thus, it is easy to define a new decomposition hierarchy for the ENVs. As we do not use any commercial GIS system we also define how to store the ENVs, the file names, the type of the ascendant ENV, the names of the sub-ENV and the ENV type. This ENV definition yields, in the case of the parcel the following information:

Parcel	***name of the entity***
type : 1	***type of the entity***
nb_under : 3	***number of entities composing the entity***
parzci	***list of the sub_entity names***
notraffic_zone	
building	
type_eng_up : 0	***type of the up entity, in this case the block***
label : **INT**	***name found in the scene for the parcel***
name : PAR_#2#_INT	***name after re-naming***

rename_from : 3	***type of information used to rename the ENV ,here a label found above the object in the hierarchy***
XPAR_#2#	
name file : 1	***number of file where we can find this ENV***
PAR_#2#.wrl	***name of the previous files***
store_file_name : DEF_FILE_PAR	
type_mobile : 5	***mobile entity able to use this ENV (pedestrians in this case)***

The names in bold, in this hierarchical decomposition model correspond to labels. The designer inserts them in the scene during the scene modelling presented in the next section.

3. The scene modelling

For 3D modelling, we are working with [3DStudio Max]. This software includes everything we need to produce vrml files (vrml importer/exporter, vrml nodes like Inline [Vrml]). We can then convert these files into Open Inventor format via Casus Presenter [Casus]. To save time at the design step, the reuse of objects with same geometry and texture is very important. To be able to import objects previously constructed in a scene, designers use a vrml node called “Inline”. This function is implemented in 3dStudio and allows embedding of files. An "Inline" corresponds to a link with a pre-designed object file giving its location. This file location can be absolute (directly from the root) or relative (to the location of the created scene). If the designer needs to modify an object, the modification will be done only once and will be pass to all the instances linked via "inline" nodes. In order to handle the object management, the designers store the entire scene in a file structure. This structure tidies up all the object file representation and is necessary for handling LOD notions. Using the notion of decomposition needed for the database construction, we sort objects in relative directories in order to reuse models and textures. The higher level is the quarter level, which can be decomposed into blocks. This decomposition is the same for directories and at the quarter level we found the texture directory (/mp) with the different blocks, buildings, and objects directories. For the texture management, there is a specific directory where all the textures are stored. The designer points to this directory in order to reuse the textures as much as possible. Making loading test for embed files, the "DEF-USE" tool seems to be efficient. This tool has the advantage of reusing objects several times with only one load (with some loaders). We have established that a "DEF-USE" is efficient only on objects not directly analysed for the database construction. The database is composed of ENVs such as sidewalks. They are "fixed" for an easier computation of their geometrical characteristics, whereas a tree or a public bench will be described via "DEF-USE" links. In the same manner, the textures are handled with "DEF-USE". All the rolling ways are mapped with the same asphalt texture, which could be loaded only once. As we structure the object directories, we structure the fixed and linked objects inside files. To be processed and converted into an informed database of objects, the designer has applied the generic model described previously onto object names. Objects with same properties are grouped.

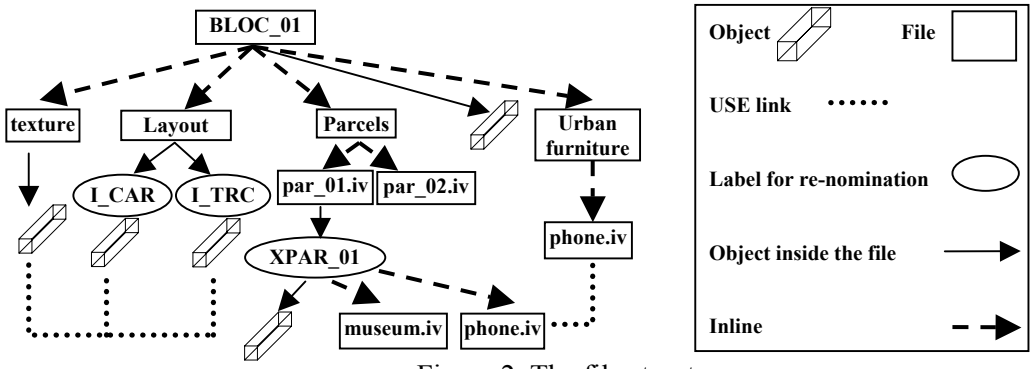


Figure 2. The file structure

For example at the block level, the scene is composed of an entity corresponding to the surface of the block (the layout of streets, parcels and segments) plus three embedded files corresponding to: The parcel file (containing the list of all the parcels files of the block), the urban furniture (such as public telephone) and the list of the used textures in the displayed scene. These three parts

correspond to three “inline” nodes, the names associated permit to exchange the included part easily in the scene graph just by modifying the name of the linked file. Thus for the layout file, we can exchange a heavy representation of streets (with details and precise textures mapping) by simple coloured cubes. The nomination of the different LOD is directly linked to the database. Using labels, it is possible to construct the scene adding building inside a parcel for example [Farenc(1999)] and to associate different representations according to parameters such as location of a relative point of view. Figure 2 shows an example of this structure of scene decomposition. Dote lines represent the "DEF-USE" links, hatched lines the "Inlines" and full line the included objects. For time rendering optimisation, we use LOD notion in order to have different representations of a same object, or some tricks to simulate visual perception.

3.1 A Standard: LOD of VRML

LOD technique is a good way for increasing scenes rendering. Because a complex geometry takes longer to be rendered, the designer models different versions of the scene. The LODs are associated with a range of distance and the "visualiser" tool switches between the different models. Because further is the viewpoint, lower details can be displayed then time rendering is optimised. This node is implemented in VRML and Open Inventor languages [Vrml][OpenIventor].

3.2 Some rules to link LOD and future database

We have chosen to define some rules in order to allow the user to easily find different LOD for a same part of the scene. In the directory dedicated to a specific object (parcel, building or telephone see Figure 2) the designer creates different LOD for an object. For an object named "myobj1.iv", the designer creates different versions using some extension about the refinement associated to the considerate version. For the object "myobj1.iv", there could be in the directory containing "myobj1.iv", "myobj1_F1.iv", "myobj1_F2.iv", "myobj1_F3.iv" (extension F for far) or "myobj1_N1.iv", "myobj1_N3.iv" (extension N for Near). Higher is the number, more the precision is accurate/rough. In a text comment at the top of the file, the designer informs the user about the scale of the scene (metre, centimetre or km), the ideal range for the distance for the camera and the dimension on the ground covered by the part of the scene of the LOD. An LOD inline file could have this type of configuration for a Building_station_N2.iv file:

```
{       # scale metre       # ground surface 20*100
        # distance 20 - 50…}
```

To provide user with larger scene, we have explored others techniques.

3.3 Trompe l'oeil technique and scene part displacement

For simulation situated in the centre of a block, surrounding three-dimensional block representations are not necessary and two concepts can be used to enlarge the urban view without loading unnecessary objects. The first one relies on trompe l'oeil technique, simulating other surrounding blocks views. The second one consists in moving a block situated behind the camera in front of the current block in order to give a sensation of infinite perception. Both are dependent on the geometrical decomposition of the urban model. We have defined two versions on which these techniques could be applied on: the ideal geometric decomposition and the generic one. We present these models and then the techniques. The ideal geometric decomposition is made of nine cubic blocs forming a quarter and a bloc is composed of nine cubic parcels. With this ideal model there is always a central unit surrounded by eight others units. Even if this seems unrealistic, concrete examples exist in reality as in Manhattan centre with a ratio of 1/3 between width and length for all the parcels of this quarter. The most generic model of decomposition on which the techniques of enlarged view presented here can be applied is an n faces convex polygon. Having in mind an ideal/generic geometric model decomposition we present the techniques of ″trompe l'oeil″ and infinite model.

3.3.1 Trompe l'oeil technique

It concerns simulations of a centralised view of neighboured ENVs, from inside an ENV such as a block. There are two methods to display pictures of surrounding blocks.

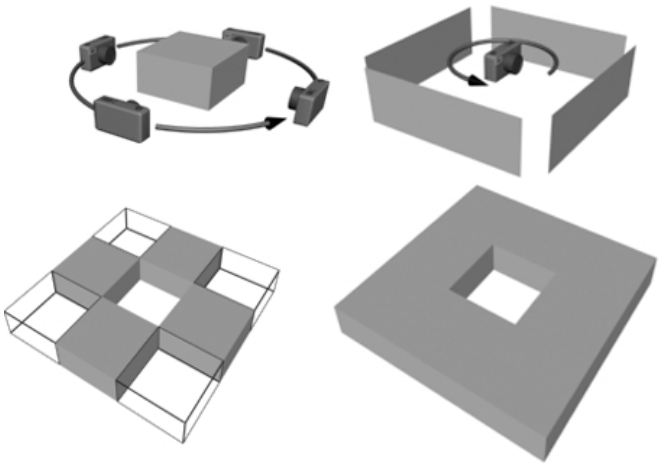


Figure 3. Trompe l'oeil techniques

The first one is to turn around the block, and to take perpendicular pictures of all the edges of the block. The cube (ideal decomposition) or a more complex polygon simulates the viewed block with these pictures mapped on the external faces. For a simulation all surrounding blocks are replaced by these lures. This can be a LOD for the block. The main drawback of this technique, regarding the ideal decomposition is the non-realistic effect created because the diagonally opposite blocs aren't seen and there is a huge feeling of being in a box. This impression is worst in a generic modelling (Fig. 3 left part). On other hand it is possible to compose a cubic panorama of the eight surrounding blocs (ideal model) from the simulation centre bloc, and wrap these cubic maps on the inside face of a global cube covering the simulation scene (Fig. 3 right part). D. Aliaga presents an example of this kind technique in order to accelerate time rendering [Aliaga(1999)]. The difficulty of such a method is to load and visualise all the blocs (without the central one), and to compute the maps (due to inline visibility and space memory). Under 3DsMax, if it is possible to view such a scene, a tool PANORAMA [Panorama] is accessible and generates automatically the maps and the cube (with the ideal model). For a generic model the problem is more complex and problem of non-perpendicular views mixed with different points of view can create visual artefacts.

3.3.2 Infinite model

This solution consists in moving parts previously loaded and hidden in front of the camera during a camera movement. The board effect when the user arrives near the end of the scene is removed, and the user doesn't have the feeling to be movement restricted in a box. The film "The Cube", presents this type of technique [Cube]. The main drawback is the constraints emerging for the scene design. All board connections have to be planned in order to connect all the sides with each other. Moreover the virtual autonomous human has to be "transferred" and repositioned inside the scene when the scene is reconfigured. The database also has to be updated at each scene part movement in order to assume data consistency.

4. Visualisation tool and LOD management

In order to verify and complete the database, we have developed a database visualisation tool. This visualisation tool has been divided in two principal data 's display through two windows: the main window for the scene visualisation and the second one for database representation. The main window shows the scene and provides to the users the possibility to select part of the scene. The database window can display the database content, or the scene file decomposition via inlines structure, the generic model. Furthermore we add a tool to create the LOD database complement which will be described at the end of this section. According to [Smith(1999)], every description of the interaction techniques has different drawbacks. An informal approach of the description makes more inconsistent the model, and S. Smith and al show that there is no obvious way to describe it. If an environment is not full comprehensible, it makes difficulties to interact with it.

4.1 Inlines composition

We begin the description of the database visualisation window with the simplest visualisation: the inlines composition. In fact, this notion has no attribute to associate with, so we determine to illustrate an inline as a sphere. The inlines are sorted in a hierarchical structure, which is displayed in the visualisation window. The main goal of this representation is not only to show the hierarchy but also to bring an easier way to manipulate the LODs characteristics. We have a couple of manners to achieve it. The LODs are linked to the inline files, not directly to an object. We have two modes of selection: concrete object selection via the scene window, using a mouse and the "inline" selection. The selection via the scene visualisation can be just a part of the object or just a part of the inline-selected content. The second one allows the user to select directly the inline (a file), without idea of what objects it contains.

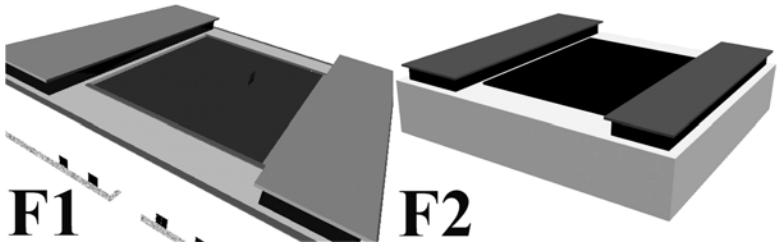


Figure 4. Different low LOD for a museum

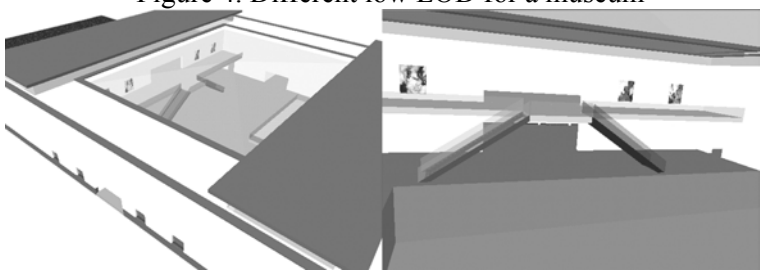


Figure 5. Views of a high LOD for a museum

Figure 4 and 5 present different LOD for a same building: a museum. F2 corresponds to three boxes with only material color whereas F1 is a little bit more complex but without rooms, inside stairs and pictures on the walls as shown in Figure 5. Using the inlines present in the scene and in the database, we construct a simplified scene named Bdbscene corresponding to the inlines's computed bounding boxes. This simplified scene is stored in the database and is used to perform optimal computation of viewed inlines according to a camera location and configuration. Thus for a point of view, with set-ups camera, the optimisation process computes only using boxes the visible inlines and their associated distance. This will be important for the LOD management. In case of embedded inlines such as a stair inside a building, the distance does not determine alone the inline observed. The "semantic" location of the camera using the database (for example inside the building) can enhance the computation of visible inlines.

4.2 Visualisation of the generic model

How we have explained, the generic model uses a hierarchical structure; hence, its representation follows the generic model’s hierarchy with 3D icons. The main raison to choose 3-dimension icons against 2-dimension icons is to symbolise as near as possible the displayed scene. The user can manipulate the symbolic representation of the generic model and have different views (profile or top) in order to have a better perception of the model structure. Figure 7 shows different views of the database display. A draw represents an up view with the ENVs surface in the front plan and the star corresponds to the block decomposition with the location of each block's sub_ENVs. B draw corresponds to the multilayer notion, a side view, for parcel decomposition. C draw is an up view of the generic model representation with different icons such as building, street or block icon.

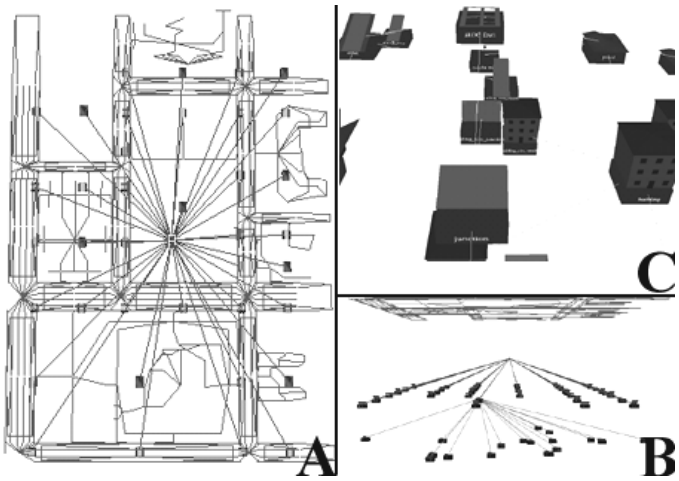
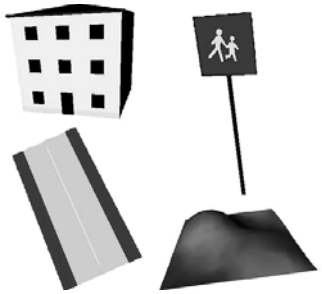


Figure 6. Database display

Continuing with the properties of the generic model, the entities overlapping is represented via links with different colours depending on the entity depth degree from the root entity in the hierarchical structure. Inherent entity information is reflected through: the icon type associated with, its spatial location (plan location plus depth location) and via its name display. We present the strategy adopted for these attributes meaning/choice. In order to represent the hierarchical decomposition we position an ENV and all its sub-ENVs relatively to the ascendant entity position. To represent the notion of depth in the hierarchical model, we adopt a multilayer approach, which can be viewed from a profile view (Fig. 1). Once the position in the hierarchical decomposition model is represented, the icons are simply arranged around a circle. By writing the type of the object on all the faces of a cubic base, the user can perceive this information from all points of view. The colour of the name informs about a possible decomposition into sub-ENV. The icon is attached on the top face. The icon symbolises an entity (a concept or a viewed object). Its drawing reflects as near as possible the semantic attached to the entity it represents. We classify the entities from their attached semantic notion in order to choose the most relevant icon (Fig. 7).



- Building icon
- Pedestrian walk area icon
- Street icon
- Parcel icon

Figure 7. Icon Representations

4.3 Visualisation of the database

The dissimilarities between the generic model visualisation and the concrete scene visualisation are few. The concrete scene visualisation respects the majority of the properties explained in the precedent section. Nevertheless, there are some key differences: the child's spatial position and additional information not coming from the generic model. The database contains notion of spatial position, which doesn't appear in the conceptual generic model. This spatial location is important for the association between the scene and the icon representation. Taking in account spatial location information makes the symbolic representation not only richer but also assures a more coherent representation [Catarci(1999)]. Location of ENVs is important and the visual representation must reflect such data. The Database provides the centre of the entity and instead of using a circle model for child's position, the location in the plane corresponds to the location of the entity in the scene. This approach gives us a clear and understandable scene, with a lot of semantics, making easier to recognise an object within the hierarchy. Other important difference is

an information window. An information window appears when an object of the hierarchy is selected. It informs about the object selected, concerning the object's name, number of children and other additional information stored in the database such as the list of actions commonly performed by humans in a place linked with an entity. Fig 8 represents a parcel visualisation, with icons, spatial location, name display and its information window.

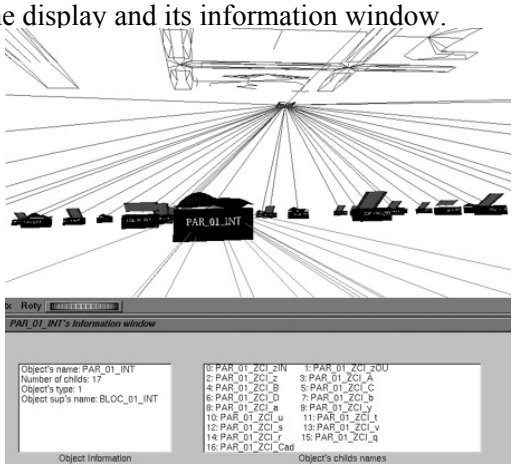


Figure 8. Database Hierarchy with textual info.

4.4 GUI interface to complete the database with LOD associated to the inlines

In order to add some LOD to the inlines composing the scene, a graphical user interface allows users to complete the scene and the database with different representation of an inline entity. When a user selects an inline, he can add a new LOD representation. The tool opens the repertory corresponding to the inline file and the user can select other representation (using the file nomination defined previously). This information is stored both in the database (with even some comments) and in the scene file model adding a "switch" node above the inline node to allow LOD exchange. The "switch" node displays only the selected representation and hides the other ones [ref VRML/IV]. The database-display-tool shows the LODs via horizontal lines connecting the different LODs. The user defines the default LOD and named others in order to be able to switch easily from one version to another one. In the case of buildings with inside furniture, the loading time can be quite big. A possible solution is to load, when the camera is exterior, all the external scene with low LOD and to hide inside parts with some "switch". The simulation starts with all LOD scene parts hidden, by computation we determine the more appropriate ones.

4.5 Scenario composition depending on camera location or simulation location

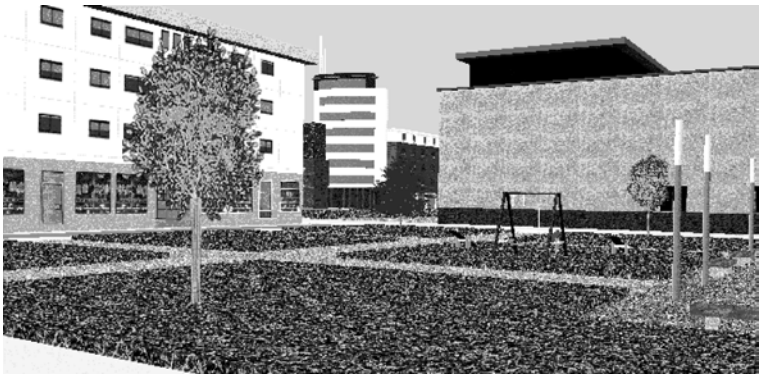


Figure 9. View of a city

The user can perform some pre-simulation computation storing some camera locations and scene visualisation in the database. The ENV associated to the camera movement surface are computed and stored in a list. The connective links between ENVs help to determine the visibility at the first level. The scene corresponding to the camera displacement surface is the sum of the entire part scene observed with the higher LOD computed. In the database we store the surface of displacement of the camera, its configuration and all the inlines to load with this scenario. If our

camera is moving passing from one area to another one, the scene can be or a list of camera scenarios or a total sum of the "inlines" linked to the camera displacement surface. This method is efficient if the camera does not cover a big area. Conversely, the scene constructed could be too huge and the best solution is to have different LOD handled with the “switch” nodes. Figure 9 shows a picture of a scene with low LODs for the museum (Figure 5/F1) inside our virtual city.

5. Conclusion and Future Work

In this paper, we have presented an integration of an Informed Environment and a tool for display optimisation. Improvement of this work could concern complement of information to the user concerning the optimum method adapted to its simulation. A new step for this work will be the texture management. According to the location of the camera the user can specify a geometrical model and the LOD will only deal with the sharpness of the textures. Another possibility could be to fix the texture and change the complexity of the geometrical model. This kind of choice depends on the application and user wishes, and on the hardware capability.

Acknowledgements

This research was sponsored by the Swiss National Research Foundation. The authors would like to thank C. Babski for its help concerning VRML standard/tests.

References

Aliaga, D.G. and Lastra, A."Automatic (1999). Image Placement to provide A Guarantee Frame Rate. *Computer Graphic Proceedings SIGGRAPH 99.*

Catarci, T., Santucci, G., Costabile, M. F. and Cruz, I. (1999). Foundations of the DARE System for Drawing Adequate Representations, *DANTE'99 proceeding*, Tokyo, Japan, 53-62.

Catmull, E. (1978) “A hidden-surface algorithm with Antialiasing”, ACM Computer Graphics, Vol 12, N 3, Aug 1978, 6-11.

Cohen, J., Varshney, A., Manocha, D., Turk, G., Weber, H., Agarwal, P., Brooks, F. and Wright, W. (1996). Simplication Envelopes, *Computer Graphics SIGGRAPH'96 proceedings*, New Orleans USA, 119-128.

Doyle, P. and Hayes_Roth B.(1997), Agents in Annotated Worlds, *Report No KSL 97_09* Knowledge Systems Laboratory Stanford University California 94305.

Farenc, N., Boulic, R. and Thalmann, D. (1999), An Informed Environment dedicated to the simulation of virtual humans in a urban context, *Eurographics'99*, Milan Italy, C309-317.

Hoppe, H. (1996), Progressive Meshes, Computer Graphics *SIGGRAPH'96* proceedings, New Orleans USA, 99-108.

Open Inventor C++, Reference Manual, *The official Reference Document for Open Inventor, release 2*, Addison Wesley 1994, ISBN 0_201_62491_5.

Reddy, M., Leclerc, Y. G., Iverson, L., Bletter, N. and Vidimce, K. (1998), TerraVision II: Visualizing Massive Terrain Databases in VRML, *AIC Technical report N. 559*, SRI International, Menlo Park, CA.

Smith, S., Duke, D. and Massink, M (1999), The Hybrid World of Virtual Environments, TACIT report TR-005.

Thalmann, D., Farenc, N. and Boulic, R. (1999),Virtual Human Life Simulation and Database: Why and How, *DANTE'99* proceeding, Tokyo, Japan, Nov. 1999,63-71.

3Dstudio Max tool home page, http://www2.discreet.com/products/products.html?prod03dsmax.

Panorama tool home page http://www.habware.at/duck3.htm#PANORAMA.

WorldView VRML browser. http://www.intervista.com/ and VRML 2.0 specifications. http://www.vrml.org.

Casus Presenter home page. http://www.igd.fhg.de/CP.

Cube (film). http://cubethemovie.com/cube.html.