Ibero-American Symposium on Computer Graphics - SIACG (2006)
P. Brunet, N. Correia, and G. Baranoski (Editors)

Automatic architectural 3D model generation with sunlight simulation

A. Mas and G. Besuievsky

Graphics Group of Girona, Universitat de Girona, Spain

Abstract

Nowadays, 2D architectural design is provided with many good tools. But it lacks efficient tools that could help designers in early stages of a project for tasks such as virtual prototyping or 3D model simulation. The 3D model generation from 2D floorplans in general implies tedious manual effort. With a 3D model the designer could validate both the building structure and the space distribution. Natural lighting is also a basic feature in structures design as a correct sizing of the building openings allow a better exploration of daylight illumination, and consequently, could prevent artificial illumination needs.
In this paper we present a tool that allows to obtain, in a highly automatic process, 3D architectural models from 2D floorplans. The system also provides sunlight simulation for visualization and interactive edition operations. It allows to re-model the openings of the structure with real-time sunlight visualization. The whole module can be integrated within a standard CAD architectural platform as a complement for fast virtual prototyping and validation designs in an early stage of the project. We present results showing its application for both conceptual designs and for a real building model.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Modeling packages, Shadowing

1. Introduction

Nowadays, the use of software tools for architectural design is spreading from conventional architectural CAD products to more automatic tools where designers can easily validate their projects within interactive workflows. One of these improvements is the automatization of the 3D model generation from 2D floorplans. In this way, fast 3D visual inspection could be performed avoiding the tedious manual effort that in general involves the model generation. The illumination in virtual environments has also a very important role in architectural design, both for model visualization and for the lighting system design. The correct modeling of opening dimensions in a building allows to improve daylight uses. A tool that could model the openings using daylight simulation and visualization issues would be helpful for a better exploration of daylight needs.

In this paper we present a system conceived to be completely integrated in any architectural CAD platform as a complement for fast virtual prototyping and validation designs in a conceptual phase of the project. The goal is to provide design constructors with a set of interactive tools for automatic 3D model generation from 2D plans and opening edition with sunlight visualization. The main contributions of this work are: a method for automatic 3D structures generation from 2D architecture symbols and the implementation of a real-time shadow algorithm improved for architectural environments.

The paper is organized as follow: in section 2 the state of the art of architectural system tools is reviewed, the whole system scheme is described in section 3, in section 4 the model generation tool is presented and in section 5 and 6 our sunlight method and the opening edition module respectively are described. We present results showing its application for both conceptual designs and for a real building (section 7). The system is shown to be a suitable tool for the generation and validation of 3D models in an interactive and easy way.

2. Tools and state of art

Analysing the state of the art of architectural tools related to our proposed goals we can classify them in two categories: architectural CAD tools and 3D modeling tools with their respective features on visualization and lighting simulation. Architectural CAD tools such as AutoCAD, Architectural Desktop or AllPlan provides processes for generating 3D models but not for a complete automated generation from 2D floorplans. 3D modeling and Image Synthesis software such as Maya, VIZ or AccuRender provides advanced lighting simulation methods, but they also require computational costs over interactive rate demands.

Our work could be related to the one presented in [OWYC05]. In this work 3D architectural models are obtained from 2D floorplans. The process is semi-automatic and successful results are obtained. However, the method makes some assumptions about the input data that could be conflictive within the integration onto a CAD system. For example, they assume a restricted set of symbols in the plan, such as doors or windows, and this could be a problem if the designer uses other symbols. We think that more flexible rules can be adopted to allow a greater symbols domain.

Shadow visualization is a very important cue in order to get a realistic illumination simulation. There are a lot of methods for shadows computation and visualization [WPF90]. We analyse here only the methods that allow interactive frame-rates. The *Shadow Volumes* algorithm [Cro77] calculates the shadows from projections of shadow generator polygons. The algorithm is view dependent, so it is not suitable for interactive purposes. A better approach is the *Stencil Shadow Volumes* [Kil01] [Kwo04]. The foundation of the method is very similar to the previous one, but its efficiency is based on a hardware implementation with OpenGL devices.

3. System overview

The proposed system can be integrated into any architectural environment platform composed by other CAD systems. In this way, it may be useful as a complement for fast virtual prototyping. The 2D floorplan input is in *DXF* format, which is one of the most standard formats used by 2D CAD packages. After the 2D model is parsed and processed a 3D model is created and stored in *VRML* format, also a standard format accepted as input in most CAD packages. For the sunlight simulation, condition parameters such as the date, time and geographic position must be specified. A 3D navigator allows to visualize the model modifying interactively both the condition parameters and the openings dimensions and positions (see Figure1). In the following sections we describe details of the system components.

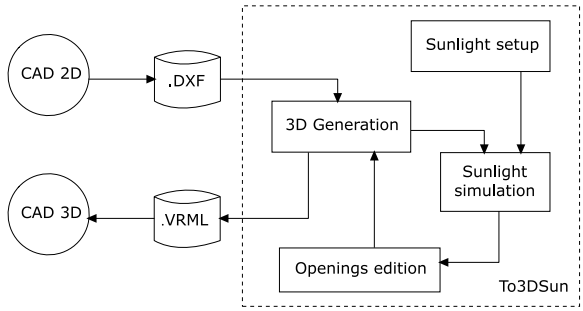

Figure 1: *General system scheme.*

4. 3D model generation

A 2D architectural design is composed by vectorial information representing structure elements like walls, doors, windows and other symbols [Aut]. In our approach we classify some of this vectorial elements based on their meaning and the structure representation that is created in the 2D design process through the layer assignment. Usually the designer creates a set of layers in the design and each layer is associated with a type of object based on their meaning. Also, some symbols can be identified with labels. The user must select manually, and in a semi-interactive way, the layers and labels of interest. Once the 2D elements are identified, the generation of 3D objects is automatic. We consider the following basic objects : walls, doors, windows, ground, ceiling, stairs and furniture. Except for furniture, that is instantiated form a 3D library, all of these objects have specific algorithms to be generated.

4.1. Walls

The method to create wall objects has three steps. First, we identify the vectorial objects that represent walls, which could be represented in one or more layers, both as lines and polylines. In a second step we search for the wall profiles. Each wall profile must be a closed polygon with its vertices sorted in a positive direction. Finally, we specify the floor height and we create the walls from the extrusion of each line in the polygon. Note that the positive vertices orientation allows the correct orientation for the normal of these planes. (see Figure 2).

Two methods have been considered to obtain the profiles. The first one assumes that each wall is represented by two parallel lines. In this case we only need to search closed polygon sequences. The second one assumes that each wall is represented by only one line with a thickness value. Here we have to create two parallel lines over the wall line and separate them by the same distance than the thickness value. Joining and closing the new lines we obtain closed polygons (see Figure 3). If there are two consecutive walls we can join them into one closed polygon.

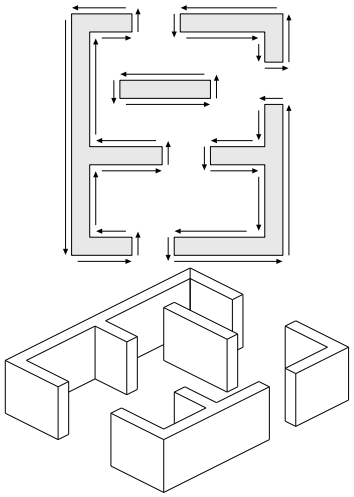

Figure 2: *Walls construction.*

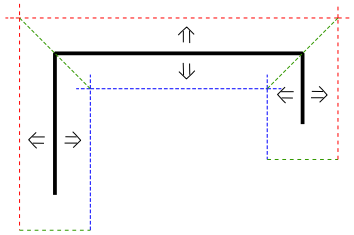

Figure 3: *Wall profile creation from walls defined by only one line. Red and blue lines are the exterior and interior wall profile lines. Green lines are the walls joining and closings.*

4.2. Openings

In the context of this work we only consider doors and windows as openings of a building structure. In general, they are represented with symbols composed by vectorial elements. Because of the lack of a defined standard pattern to represent openings (see Figure 4), we decided to use label information instead of a vectorial recognition approach. In a first step, we select all doors and windows using the labels that identify them or using the layer where they are placed. Then, we search for the two nearest walls from the insertion point symbol. Using the walls as bounds, we generate a rectangle that fills the space between walls. In this way we define the opening size. Finally, we generate the 3D structure by extrusion of the rectangle according to their category : door (with only top part) or window (with top and bottom part) (see Figure5).

4.3. Ground and Ceiling

We can create the ground and ceiling by simply creating bounding polygons that close the walls and all openings.

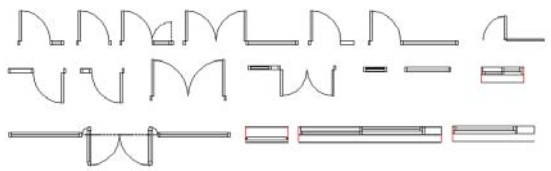

Figure 4: *A small representation of opening symbol possibilities.*

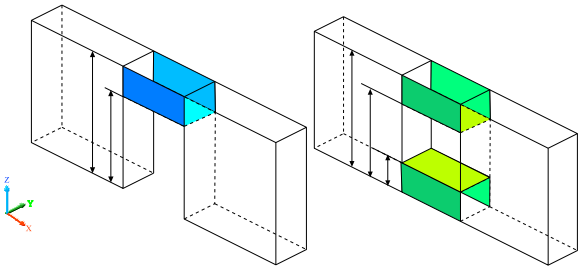

Figure 5: *Openings constructions.*

But this approach could represent a problem if the resulting 3D model is exported for its usage in a rendering system based on radiosity or finite elements methods (see Figure 6). This kind of systems are frequently used in architectural interior building visualization where global illumination is required. We propose a solution, called *Rooms Search*, which allows the identification and segmentation of interior room zones. The algorithm begins with the 2D bounding box created from the walls and opening profiles found before. First, the profiles of the wall are modified to join them with the openings profiles to obtain the closed rooms. Then, an intersection boolean operation is applied between the 2D bounding box and the structure profiles (see Figure 7). The result are the segmented interior rooms zones (see Figure 8).

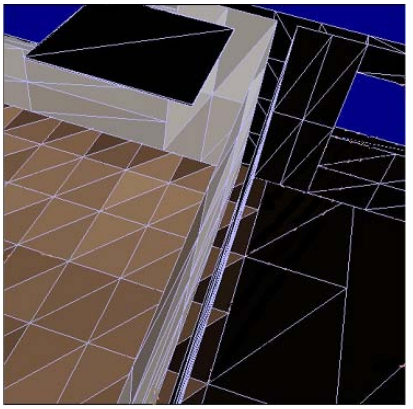

Figure 6: *Problem joining walls and ground. From this top plan view you can note the illumination error under the wall*

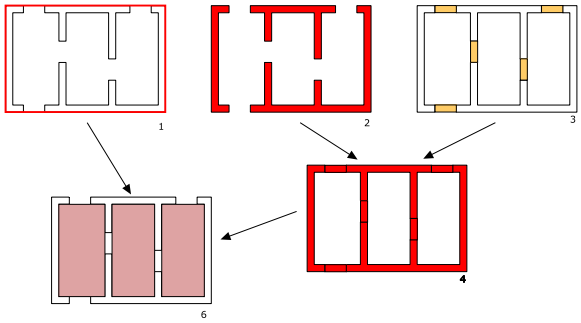

Figure 7: *Rooms Search algorithm.*

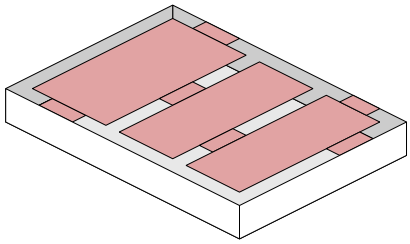

Figure 8: *Final roof or ceiling construction.*

4.4. Stairs, Open spaces and Multiple floors

Stairs are identified with a symbol that provides the growing orientation and the planar rest of the stair with their corresponding size (see Figure 9). With this data we generate ramps for each stair flight, and planar surfaces for each landing. For the stairs placements it is also necessary to make holes in the ground or in the ceiling. For this step we use a simple boolean operation between the ceiling or the ground, and the 2D bounding box of the stairs.

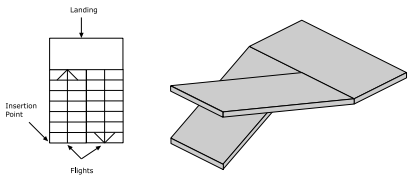

Figure 9: *Stairs 3D generation from 2D floorplan symbol.*

Multiple floors are created by replicating them with the corresponding transformation of the 3D model generated. Open spaces, used to connect consecutive floors, are indicated with a rectangular symbol in 2D plans. To create the 3D holes we perform the same boolean intersection operation applied for stairs. For both stairs and open size elements no user intervention is required, the 3D modeling operation is completely automatic.

5. Sunlight

A complete daylight simulation could be a complex task when considering the sun and sky as sources and the global illumination process with all the related parameters. The use of a complete simulation generates realistic results [BM05] but is not suitable for interactive changes. In order to achieve interactive updates rates we consider only the sunlight simulation with the condition parameters such as hour, date and geographic location, and the shadows produced in the architectural environment.

5.1. Sunlight source

An interactive visualization algorithm was developed considering only sun direct illumination. The Sun could be though as a kind of omnidirectional source light because it emits uniformly in all directions, although usually it is considered as a directional light source due to the distance from the Earth. Thus, the incoming sunlight only depends on the angle of incidence of the rays to the Earth and the intensity. We used the model proposed in [PSS99] to calculate this incidence, which depends on the longitude, latitude, day of the year and time. Directional direct illumination can be computed efficiently using hardware implementations [SWND03].

5.2. Shadows

For shadows simulation we used an approach of the Stencil Shadow Volumes algorithm [Kil01] [Kil99] [Kwo04], which bases its efficiency on the correct use of the graphics library OpenGL. The Stencil Shadow Volumes is based on the original idea of Shadow Volumes [Cro77]. This algorithm proposes the construction of volumes generated by the polygons that cause the shadows. The volumes split the space in two regions: the first for shadowed objects and the second for unshadowed ones (see Figure 10).

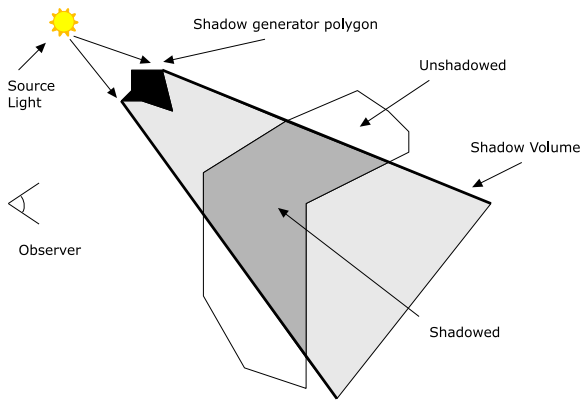

Figure 10: *Shadow Volume sectional view.*

The algorithm strategy has two stages. In the first one the

volumes are generated (Shadow Volumes) and in the second one the scene is divided according to shadowed or unshadowed polygons. The core of the algorithm is well documented with examples in [Kil01] and [Kwo04]. In this work we develop an adaptation of this algorithm to architectural environments. In the following sections we describe the proposed modifications to the original algorithm in order to achieve efficient results within our goals.

5.2.1. Computing the volumes

The volumes are created from the projection of the polygons that generate shadows and from the incident sunlight rays. However, not all polygons are necessary to be considered as shadow generators. For example, the ground of the building will not generate shadows. Also, it can be observed that not all the polygons of the wall are shadows generators, approximately just the half. Applying a simple visibility algorithm with the viewer placed in the source light direction, we can discard visible polygons, and consider only the non-visible polygons to generate the shadows (see Figure 11).

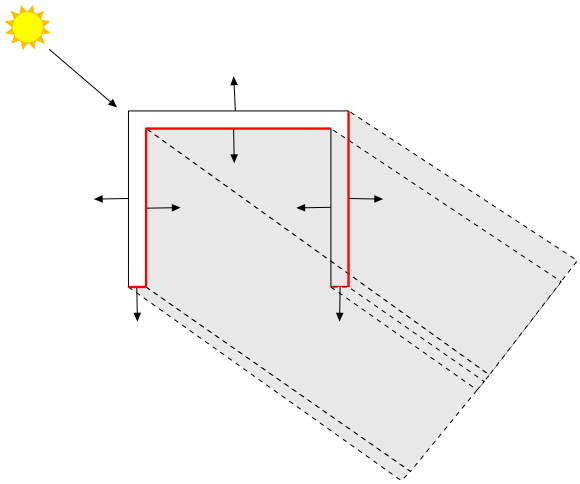

Figure 11: *Selection of polygons that generate shadows. The red polygons are selected.*

5.2.2. Inside or outside of Shadow Volume

In order to know if a polygon is inside or outside of a Shadow Volume, we use the position and the normal vector and check if the polygon is visible from the sunlight direction. There are some proposed approximations [Kil99] [Kil01] that has been considered to adapt the Shadow Volume algorithm to architectural environments and direct sunlight. We used a modified version of an algorithm called *Z-fail* [Kil99]. This algorithm casts rays from the infinite to the viewer, using an integer counter for each ray, which is increased by one each time the ray enters in a volume, and it is decreased when the ray exits (see Figure 12). Each ray corresponds to one pixel on final render image, thus, all pixels with counter different to 0 are in shadow. This assumption is also valid when the viewer is inside the volume, once the counters are initialized at the infinite. This is a very frequent case when performing walkthroughs in architectural interior buildings.

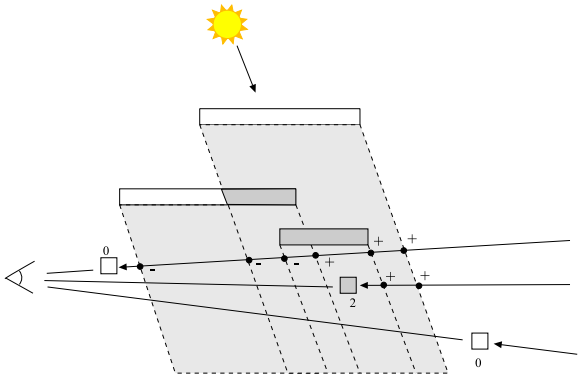

Figure 12: *Z-fail example.*

5.2.3. OpenGL implementation

The original Shadow Volumes algorithm [Cro77] was a view dependent method, and, in consequence, not suitable for interactive purposes. The Stencil Shadow Volumes [Kil99] [Kil01] improves the Shadow Volumes in order to achieve efficiency for interactive tasks. The main idea behind this method is a specific implementation using a current graphics hardware library as OpenGL [SWND03].

We describe here the three implementation steps of the algorithm using OpenGL. In the first step, all internal pipeline buffers are initialized and the scene is rendered with the sunlight source enabled.

```
EnableSunLight();
glEnable(GL_DEPTH_TEST);
glDepthFunc(GL_LEQUAL);
glDepthMask(1);
glDisable(GL_STENCIL_TEST);
glClearStencil(0);
glColorMask(1,1,1,1);
glClear(GL_COLOR_BUFFER|
     GL_DEPTH_BUFFER|
     GL_STENCIL_BUFFER);
glEnable(GL_CULL_FACE);
DrawScene();
```

The second step disables the sunlight source and applies the Z-Fail algorithm using the *Stencil Buffer* by processing the volumes and the intersections between them with sunlight rays. In this process the output rendering is disabled.

```
DisableSunLight();
glEnable(GL_DEPTH_TEST);
glDepthFunc(GL_LESS);
glDepthMask(0);
glEnable(GL_STENCIL_TEST);
glStencilFunc(GL_ALWAYS,0,0);
glStencilMask(~0u);
glColorMask(0,0,0,0);
glEnable(GL_CULLFACE);
glStencilOp(GL_KEEP,GL_INCR,GL_KEEP);
glCullFace(GL_FRONT);
DrawShadowVolumes();
glStencilOp(GL_KEEP,GL_DECR,GL_KEEP);
glCullFace(GL_BACK);
DrawShadowVolumes();
```

Finally, all the scene is rendered again, but only the shadowed pixels are updated. To show the pixels in shadow, the sunlight source is disabled.

```
glEnable(GL_LIGHTING);
glEnable(GL_DEPTH_TEST);
glDepthFunc(GL_EQUAL);
glDepthMask(0);
glEnable(GL_STENCIL_TEST);
glStencilFunc(GL_NOTEQUAL,0x0,~0u);
glStencilOp(GL_KEEP,GL_KEEP,GL_KEEP);
glColorMask(1,1,1,1);
DrawScene();
```

The use of Stencil Buffer is essential in this algorithm. It is used to implement the counters for each pixel ray. Some improvements to the basic algorithm have been applied, such as a more efficient use of *Display Lists* to obtain a better performance in geometry edition and volumes calculations. Also the *Blending* functionality has been used properly to avoid pure-black shadowed pixels.

6. Opening edition

For the edition operation in openings, we allow to change the length, the heights and the position insertion in the wall (see Figure 13). Note that the adjacent walls could also suffer modifications as a consequence of an operation edition. All processed changes cause geometry modifications and consequently illumination changes. Using the algorithm described above updates after a given edition can be done at interactive rates.

The presented algorithm process the scene three times for each render update. The process can be optimized using Display Lists for the geometry that does not changes. In this way, only the modified geometry and volumes are recomputed. This means that the computational cost can be reduced avoiding processing stages because in general only a few polygons need to be updated in each edition.

7. Results

In this section we present applications of our system tool. Figure 14 and 15 show the 2D floorplan and the 3D generated models for two conceptual design building containing all the element structures described. The generation process is automatic and allows to produce a fast virtual prototyper 3D inspection. Figure 16 shows one of the 2D floorplan of a three-floor real building of the *Universitat de Girona* campus and the generated 3D model. In this case the floors are generated separately and joined to compose the building.

Figure 17 and 18 show results for both sunlight interactive visualization at different times of the day and an opening edition session respectively. In order to check the performance in today's current hardware, we used a computer with one of the most simplest OpenGL graphics cards. The only requirement the graphic card needs is the OpenGL buffers implementation, such as Stencil Buffer and Depth Buffer, currently available in most cards. The minimum requirements aren't so high, needing only a Stencil Buffer with 8 bits of precision or more. Note that this bit precision indicates the number of intersected objects by each ray [Kil01].

Attached to the paper file there is an animation where all of this processes and results are demonstrated.

8. Conclusions and future work

We have presented an architecture system tool that offers automatic modeling that can complement conventional architectural CAD packages. It allows the automatic generation of 3D models from the original 2D floorplans and the interactive edition of the openings on 3D model with a sunlight simulation and visualization. This allows to design, test and validate the correct dimension of the openings in a fast way. The tool, which is conceived to be integrated with other CAD systems, had been successfully applied for both modeling real building and for visualization conceptual design projects.

Future work of this project include automatic roof generation methods and also fast global illumination simulation including the sky contribution.

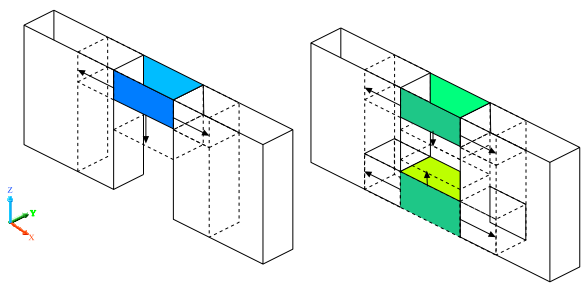

Figure 13: *Opening edition parameters.*

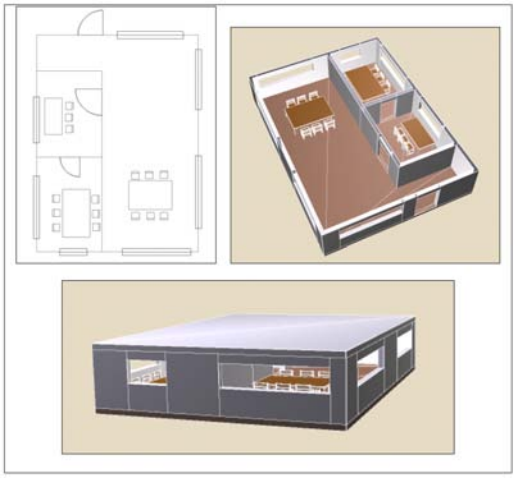

Figure 14: *Example of 3D model generation.*

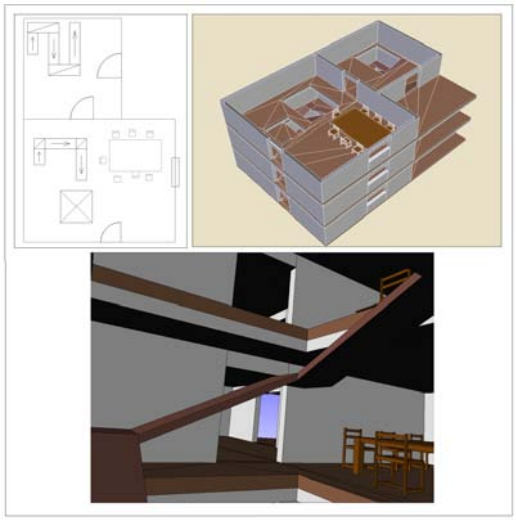

Figure 15: *Example of 3D model generation with stairs, open spaces and multiple floors*

Figure 16: *Above, the University of Girona P4 building 2D floorplan. Below, the 3D generated model*

9. Acknowledgments

This work has been financed partially by TIN 2004-07672-C03-00, CERTAP (Generalitat de Catalunya) project, PICS 2005 (DURSI), and by the Grup de Recerca Consolidat 2001/SGR/00296.

References

[Aut] AUTODESK DEVELOPMENT: *DXF Reference, v. U14.1.04*.

[BM05] BESUIEVSKY G., MARTÍN I.: A hierarchical algorithm for radiosity daylighting. *XV Congreso Español de Informática Gráfica* (2005), 75–84.

[Cro77] CROW F.: Shadows algorithms for computer graphics. *Computer Graphics (Proc. SIGGRAPH'77) 11*, 2 (jul 1977), 242–248.

[Kil99] KILGARD M. J.: Improving shadows and reflections via stencil buffer. *Advanced OpenGL Game Development course notes, Game Developer Conference* (Mar. 1999), 204–253.

[Kil01] KILGARD M. J.: Robust stencil shadow volumes. CEDEC Presentation, Tokyo, Sep. 2001.

[Kwo04] KWOON H. Y.: The theory of stencil shadow volumes. In ShaderX2, Introductions and Tutorials, W. F. Engel, Ed. Wordware Publishing, pp. 197–278.

[OWYC05] OR S. H., WONG K. H., YU Y. K., CHANG M. M. Y.: Highly automatic approach to architectural floorplan image understanding and model generation. *Vision, Modeling, and Visualization* (Nov. 2005), 25–32.

[PSS99] PREETHAM A., SHIRLEY P., SMITS P.: A practical analytic model for daylight. *Proc. SIGGRAPH'99* (1999), 91–199.

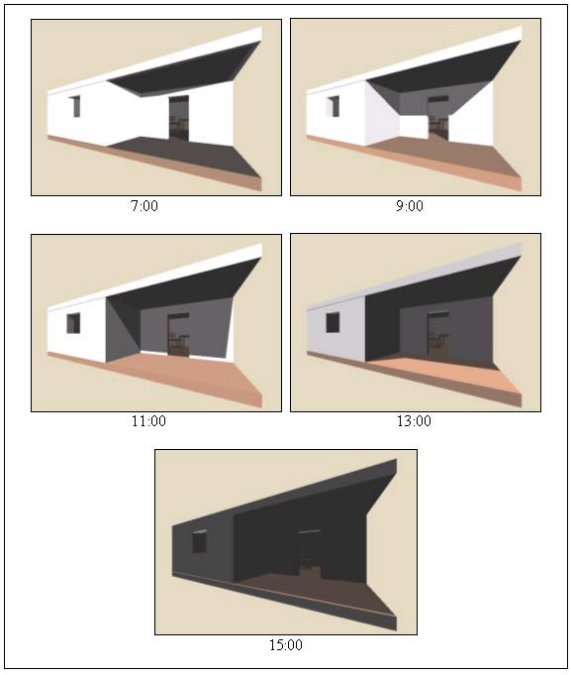

Figure 17: *Example of sunlight simulation at different day hours*

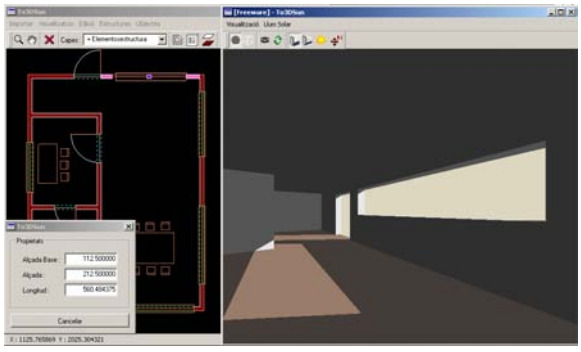

Figure 18: *Opening edition and sunlight simulation updating*

[SWND03] SHREINER D., WOO M., NEIDER J., DAVIS T.: *OpenGL Programing Guide.* Adisson-Wesley Professional, 2003.

[WPF90] WOO A., POULIN P., FOURNIER A.: A survey of shadow algorithms. *IEEE Computer Graphics and Applications* (Nov. 1990), 13–32.