Incorporating Legal Rules on Procedural House Generation

N. Rodrigues, M. Dionísio, A. Gonçalves*
DEI/ESTG-IPL, Portugal

L. G. Magalhães, J. P. Moura†
UTAD, Portugal

A. Chalmers‡
Warwick Digital Laboratory
University of Warwick, UK

Abstract

The increasing demand for larger and more complex virtual models arising from different areas (e.g. design of virtual cities, video games and computer animated movies) creates the need for efficient computer algorithms able to generate them automatically. This paper presents a method for automatic generation of traversable houses, using architectural legal rules and an L-system to generate a list of interior rooms. The method is established with a framework specifically conceived to generate 2D floor plans and 3D models which allow the generation of multiple houses and interactive navigation.

CR Categories: F.4.2 [Mathematical Logic and Formal Languages]: Grammars and Other Rewriting Systems; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism; I.6.3 [Simulation and Modeling]: Applications; J.6 [Computer-Aided Engineering]: Computer-Aided Design (CAD).

Keywords: Procedural Modelling, Virtual Environments, House Generation.

1 Introduction

In the context of Procedural Modelling, different methods for the automatic generation of virtual environments have been proposed in the last few years. The goal of such methods is to place most of the effort of creating a virtual environment on computer algorithms rather than human resources. Thus the latter may use the spared time to enhance the final models or thinking about new ways to increase their realism.

In this paper, a complete method is proposed for house models generation, aiming for single, one-floor, Portuguese homes, which focuses on producing the entire house (interior and exterior geometry). The method took into consideration strict legal rules provided by RGEU (Regulamento Geral das Edificações Urbanas, General Regulations for Urban Buildings) which is one of the main documents which all Portuguese Architecture projects have to comply with. Several authentic floor plans were also analysed to obtain rules that complement those of RGEU, which do not comprise some of the features found in real houses. Such is the example of separating a house into public areas (mostly formed by public rooms) and private areas (mostly formed by private rooms).

*e-mail: nunorod@estg.ipleiria.pt, ei12418@student.estg.ipleiria.pt, alex@estg.ipleiria.pt
†e-mail: lmagalha@utad.pt, jpmoura@utad.pt
‡e-mail: alan.chalmers@warwick.ac.uk

SCCG 2008, Budmerice, Slovakia, April 21 – 23, 2008.
© 2010 ACM 978-1-60558-957-2/08/0004 $10.00

The proposed method presents several innovative features, as follows:

- The use of an L-system[1] to generate a list of the different rooms of a house.
- Automatic generation of floor plans compliant with legal rules (RGEU).
- Algorithms for correct door placement including opening directions.
- Possibility to define rules referring to the paths between different rooms of the house (i.e. establish which rooms connect to any other rooms of the house).

Figure 1. Generated House.

A framework is also presented which implements the proposed method and allows the creation of floor plans in raster and vector formats as well as the creation of 3D models featuring many house characteristics, including doors, windows, roofs and baseboards. The realism of the final models is enhanced by varying features in different rooms of the house. These may be seen in the 3D models where kitchens and bathrooms present tiles in their walls and different kinds of materials are used for each type of room (e.g. wood floors for bedrooms, and tiles for kitchens and bathrooms). Some of these features are visible in Figure 1.

1.1 Related work

The reconstruction of urban environments has been a focus of previous work from different areas, spanning several types of data sources (e.g. aerial images, laser scanning data) [Martinez-Fonte et al. 2004, Weidner 1996] as well as different applications. There are also approaches that attempt to model a particular town or city [Ingram et al. 1996], to those that create purely artificial environments [Urban Simulation Team], as stated by R. G. Laycock and A. M. Day in [Laycock et al. 2003].

A different method of modelling (often referred to as Procedural Modelling) relies on algorithms to automatically generate

1 L-systems or Lindenmayer systems (named after their creator) consist of formal grammars and were introduced in 1968 by the Hungarian theoretical biologist and botanist from the University of Utrecht, Aristid Lindenmayer.

buildings. These algorithms usually aim for the generation of new worlds, rather than the reconstruction of existing worlds. They may also be used for the reconstruction of non-existing worlds for which we have some kind of knowledge (e.g. floor plans, photographs) to support the reconstruction of realistic environments (e.g. reconstruction of archaeological sites where only some features about construction techniques related with the site are known).

In the recent past many methods have attempted to address the field of Procedural Modelling. In [Parish et al. 2001] Parish and Müller presented an approach which uses shape grammars, namely L-systems to generate large urban environments. Wonka, in [Wonka et al. 2003], also used shape grammars, but concentrated however on creating detailed geometric façades on individual buildings. Later, in [Müller et al. 2006], the knowledge from the previous mentioned work was used by Müller and Wonka to propose a new method for addressing the problem based on a mass model technique.

Through a different approach Greuter, in [Greuter et al. 2003], focused on optimization techniques to present a framework capable of generating real-time virtual worlds and Finkenzeller, in [Finkenzeller et al. 2005], presents a technique for generating floor plans and resulting 3D Geometry based on a decomposition technique.

One common feature among these authors' techniques is the fact that only buildings façades are generated, which means that there is a degree of realism lacking. This also means that these approaches are not suitable for some applications (e.g. Architectural applications and some video games) since the generated buildings are not traversable. Almost all of these authors (except for Finkenzeller) focused solely on skyscraper-type commercial buildings.

In [Martin 2005] the author addressed this problem by presenting two different approaches to generate both outer and inner characteristics of houses (instead of skyscrapers). Even though the presented work lacks some realism, an interesting thing to notice is that some architectural issues were taken into account. This can be perceived from the fact that the author makes the distinction between public and private parts of a house, as laid out by Christopher Alexander. Alexander's contribution is also acknowledged by several others authors (some mentioned in the related work).

Though, there is also several other works which take as a reference the work on the construction and analysis of architectural design using shape grammars presented in [Stiny 1972 et al.]. These shape grammars became the foundation for many applications which comprise many different architectural styles [Koning et al. 1981, Knight 1981, Flemming 1987, Duarte 2002].

In this paper we present a new framework for the automatic creation of houses which addresses most of the gaps identified in the previous approaches. As such, we are the first to codify official rules, as laid out by RGEU, in the generation of houses. This may serve as a basis for the future development of tools, which may aid both Architects and Civil Engineers, in the creation of new building floor plans, as well as 3D models, in a sense that may be considered as artificial creativity.

1.2 Overview

This paper is structured as follows: Section II provides a general description of the proposed method for procedural house generation. Section III presents the L-system and user-defined ruleset used to generate the list of rooms which define the interior of the house. Algorithms for floor plan generation are described in Section IV, and in Section V we discuss algorithms for 3D geometry generation. In Section VI, some results achieved with the proposed method are discussed, and in Section VII some conclusions and plans for future work are presented.

2 Procedural House Generation

The proposed method for procedural house generation was divided into three major steps: Room Generation, Floor Plan Generation and 3D Model Generation. The first step, Room Generation, employs an L-system to produce a list of rooms (strings) which define the interior of the house and creates a tree with those rooms ensuring that a valid path exists between all of them (i.e. starting from the front door making sure that it is possible to reach every room). The next step, Floor Plan Generation, creates a 2D floor plan based on the tree produced in the first step. The last step, 3D Model Generation, is responsible for creating the three-dimensional geometry that corresponds to the 2D floor plan previously created and for mapping it to X3D[2].

The proposed method employs the rules of the user-defined ruleset (which includes RGEU and several observation-based rules, e.g. the separation of a house into public and private rooms[3]) to guide the presented steps, described in the next section. Figure 2 illustrates all of these steps.

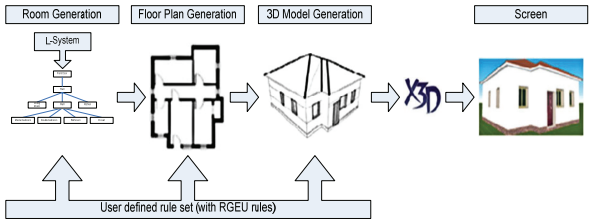

Figure 2. Method components.

3 Room Generation

The Room Generation step is similar to the method presented by Jess Martin in [Martin 2005] where the author employed a context-free grammar and user-defined ruleset to generate a graph representing the rooms in a house and the connections between them. In our method a user-defined ruleset was also used which characterizes RGEU rules and several observation-based rules, but instead of a context-free grammar the method employs a

2 X3D is an integrated 3D graphics and multimedia framework of the ISO process for Information technology — Computer graphics and image processing [Web3D Consortium].

3 The distinction between public and private rooms stems from the architectural patterns described in 1977 by Christopher Alexander's in "A Pattern Language" [Alexander et al. 1977]. In most of the related literature it is common to find references that acknowledge Alexander's contribution which may serve as a basis to several architectural applications. Nevertheless, a similar distinction has already been made in more ancient civilizations (though in different terms). One example is the Roman civilization where the literature adapted from Vitruvius (Roman Engineer and Architect) "De Architectura" refers to the function of a room [Maciel 2006].

context-free L-system in which terminal symbols represent different rooms of a house. In addition, all of the subsequent algorithms in Room Generation follow a different generation procedure, depicted in this section, in which only the ultimate goals are similar, i.e. the generation of a house floor plan where the interior structure dictates the exterior of the house.

3.1 L-system

The L-system used is stochastic, therefore, there are several rules which may be chosen with a certain probability in each iteration. On more practical terms, this means that for a certain house type (i.e. T0[4], T1, T2, T3, etc.) several room configurations may be achieved. With this method, it is possible to control derivations of the L-system, ensuring that only valid configurations are produced. When the first rule is applied, the initial symbol or axiom (Lot) is replaced by the house type. Considering the current house type, the next iteration may provide a proper list of rooms and the process proceeds until only terminal symbols are reached. To illustrate this process, Figure 3 represents the productions for a T3.

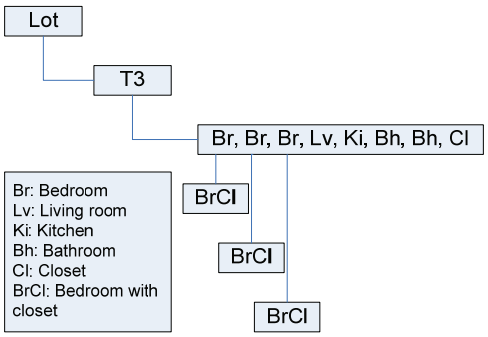

Figure 3. L-system productions for a single-floor T3.

As can be perceived in Figure 3, in the first production the house type is chosen. Next, a list of rooms is generated for the current house type and after that, each room may be replaced by a more specific room (e.g. bedroom replaced by bedroom with private closet) and so on.

The method is fairly simple, but specific enough to ensure that only valid productions are generated; and at the same time generic enough to extend to different house types and new areas of application. This is demonstrated by giving an example of extending the grammar to multi-floor houses. In this case a new level of abstraction could be added as represented in Figure 4.

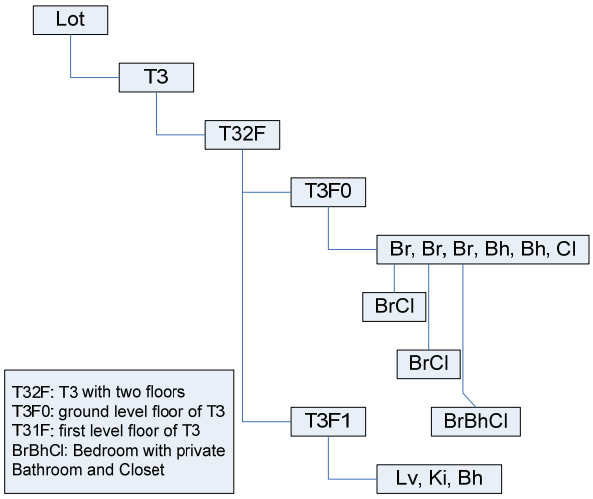

Figure 4. L-system productions for a two-floor T3.

The specified alphabet and production rules of the L-system are codified according to the RGEU rules, ensuring that some rooms must exist for different house types (e.g. in houses equalling three or more bedrooms, at least two bathrooms must exist). However the L-system is conceived to allow some extensibility so it can be used in very different situations with total control over their productions.

One important thing to notice is that any symbol of the grammar may be associated to a geometric entity for use in the generation of the floor plans and 3D models since the L-system does not involve any geometry generation.

3.2 User-defined ruleset

After determining the rooms that will make up the house, the correct linking between them must be achieved. The possible links between rooms are defined in a ruleset (as well as RGEU rules) which may be defined either at a higher level, i.e. defining the room types (e.g. public, private) which can be linked together or at a lower level i.e. defining the specific rooms (e.g. kitchen, living room) which can link to each other.

The first kind of rules offers several advantages since it prevents the definition of links between two rooms by specifically referring to each room. Links here are established by referring to the room type instead, as presented in (1).

HighLevelLink: PublicRoom, PrivateRoom (1)

With this kind of rules it is possible with just a few rules to prevent links which do not represent the reality of modern houses (e.g. a private room never serves as a link between two public rooms, meaning that in the present context a bedroom could not be used as a passage from a dining room to a kitchen).

With the second type of rules, it is possible to establish more detailed links by specifying which rooms can link to each other. One possible rule is presented in (2).

LowLevelLink: LivingRoom, Kitchen (2)

With the latter kind of rules it is also possible to override the higher level rules.

4 The house type (e.g. T0, T1, T2) where the "T" stands for the number of bedrooms in the house determines most of the legal rules from RGEU in Portuguese architectural projects.

3.3 Tree generation algorithm

The algorithm receives as input the list of rooms created by the L-system and starts by adding the first room to the tree of rooms. Then corridors are created and added to the list of house rooms. After that, all the other rooms from the list are added to the tree and finally circular connections between rooms are created.

Adding the first room

The first step of the algorithm (adding the first room) determines which room will be linked to the front door. This is defined in the ruleset, where only public rooms may link to the front door, since it does not make much sense linking a private room, such as a bedroom, to an exterior door.

Creating corridors

The second step of the algorithm consists in determining the necessary number of corridors to link all the rooms of the house. This does not mean that corridors must exist, since there are some rooms which may link directly to each other (e.g. living room linked to kitchen) and there are also other rooms which may be used to serve as a link between rooms (e.g. hall, atrium). However corridors are a predominant feature of Portuguese houses and were emphasized in the algorithm.

The number of necessary corridors is estimated following the *ad hoc* formula in (3).

$$Number\ corridors = \frac{number\_rooms}{max\_corridor\_rooms} \quad (3)$$

In the formula, the variable *number_rooms* represents the number of rooms which can be linked to a corridor and *max_corridor_rooms* the maximum number of rooms which a corridor can link to. After this calculation, the necessary number of corridors are created and added to the list of rooms.

Adding remaining rooms

After all the corridors have been created and the initial room of the tree added, the next step consists of iteratively removing each of the rooms from the list and adding them to the tree until the list is empty. The process starts by searching in the tree for all rooms which may serve as a source room to the current room, to be added. Then one of these rooms is randomly chosen and the corresponding link between the source room and the current room is defined. Then, this last one is removed from the list of rooms and the process repeats itself for the next room to be added.

The whole process is performed following the rules defined in the user-defined ruleset.

Adding circular connections

The presented algorithm allows circular connections to be created in the tree, i.e. connections between rooms which have the same origin (e.g. a corridor which links to a living room and to a dinning room that are also linked between each other). This is the last step of the algorithm, where only the rooms which are set to allow for circular connections in the user-defined ruleset are considered.

Figure 5 represents a tree generated by the tree generation algorithm for a T2 house.

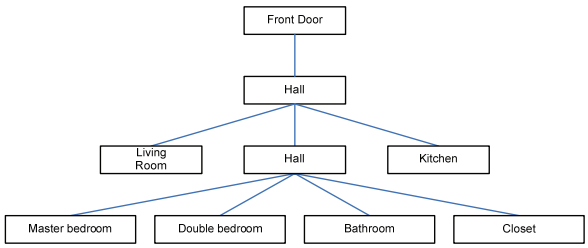

Figure 5. Algorithm generated tree.

The output of this step is the list of rooms that will constitute the house and their links.

4 Floor Plan Generation

The Floor Plan Generation step occurs in two smaller steps. In the first one, the total area available for the house is distributed through each room and then, the rooms are geometrically placed and properly dimensioned to their previously given size.

4.1 Area distribution

There are three possible ways to define the area of a house:

1. Specify particular dimensions for each room of the house (e.g. living room dimensions: 3 m (width) x 5 m (length)).
2. Specify a particular area for each room of the house (e.g. kitchen area: 16 m^2).
3. Specify the total area of the house.

This third possibility is more explicitly described since if only a total area is specified, an algorithm to distribute this area through all the rooms of the house is necessary.

The proposed algorithm makes use of different priorities for each room. Each room is given an initial size corresponding to the minimum allowed for the given room (according to the RGEU). Then the difference between the total desired area and the sum of the current areas of the rooms is distributed proportionally to each room of the house, i.e. according to their priority. This is easily explained with the example of a living room which has a greater priority than a bedroom, implying that it will receive a bigger portion. This is due to its public and social function, which means that it normally has a bigger area. This observation was obtained from authentic floor plans.

If any of the rooms exceed their maximum size, the excess area is removed and distributed through the other rooms that have not yet exceeded their limit. This last step is repeated until the total desired area is distributed through all of the rooms or until all the rooms have achieved their maximum area.

4.2 Room placement and sizing

The placement and proper sizing of the rooms is done using a greedy algorithm. This is a technique to solve problems where at each step of the algorithm the locally optimum choice is made with the hope of finding the global optimum. There are two issues to take into account with this kind of algorithms. One is that they may fail to find the best solution or a solution may not even be found (even if one exists). The other is that the algorithm is not

complete, meaning that not all of the search space is contemplated in the search for a solution. This last aspect however contributes to enhance the performance of the algorithm.

When generating a floor plan, the search space (i.e. all possible configurations for a certain connections tree between rooms of the house) is too vast to be taken into consideration. This is one of the reasons why the algorithm may fail to find a solution, since the solutions may be in the search space that was not analyzed. However, when this happens the presented method simply starts from the beginning, and in some cases the final user will not even be aware of the additional time taken by running more than once the algorithm.

Due to these issues, the algorithm uses a grid to deal with room placement and sizing. This makes the number of possible solutions (positions and dimensions) for each room finite and also makes possible to control the performance of the algorithm by changing the parameters of the grid.

The general idea of the algorithm is to navigate through each node of the tree, and for each room represented by that node, create the corresponding polygon, ensuring the appropriate links between rooms dictated by the tree branches, by placing it in correctly. After this procedure, the rooms of the house are displayed in the floor plan and then windows and doors are added. This general description of the algorithm is fairly simple to explain, since it hides the more complex aspects of the underlying sub-algorithms necessary to generate the final floor plan. These include several metrics and formulas necessary to enforce the achievement of the desired results. There are also some rooms which need some special consideration, such as corridors which are created depending on the characteristics (e.g. size, geometry) of the rooms linking to it.

Some of the sub-algorithms include door placement (taking also into account for opening directions, to avoid conflicting doors, i.e. doors which intersect each other when opened) and window placement algorithms. In both cases several RGEU rules as well as observation-based rules were accounted for, to assure the realism of the final floor plans. One example of an observation-based rule relates to door placement. In common rooms (e.g. living rooms, kitchens, bedrooms) doors are usually located near one of the extremities. On other kind of rooms (e.g. corridors) they are usually placed in the centre.

Finally there are also some other implemented algorithms with the goal of achieving better results, such as an algorithm for edge smoothing to avoid sharp edges, i.e. rooms with many edges. Wall thickness was also an aspect under consideration where different wall thicknesses were used for inner and outer walls.

In Figure 6, the generated floor plan for a T2 house may be seen.

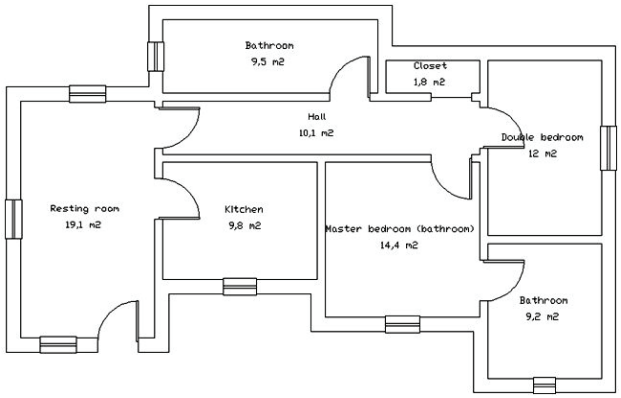

Figure 6. Floor Plan.

5 3D Model Generation

3D Geometry is achieved in three steps. First, walls are raised from the floor plan. Then doors and windows are created and finally roofs are also created and added to the final model. Each of these steps is further detailed in the next sections.

5.1 Creating walls

To generate the walls, polygons are created from the floor plans. For each edge of the floor plan representing the exterior walls, a 2D polygon is created with a length equalling the edge length and a height equalling the height defined for the walls. For the interior walls the method is similar, but follows a room approach, i.e. each room of the house is browsed and for each one, all edges are accounted for to create the polygons that compose the house.

After the walls are "parsed", the second step of the algorithm consists in adding the floor and roof planes, one for each room of the house.

Inner tiles (e.g. kitchens, bathrooms) are added for each 2D polygon based on the polygon length and mosaic height parameters and then extruded a proper thickness into a 3D volume. After this step, baseboards are created, in a similar way where the difference resides in the height parameter.

5.2 Creating doors and windows

The integration of doors and windows starts with the creation of rectangular holes for each window and door represented in the floor plan. This is done by subtracting rectangles from the inner and outer walls (that correspond to the locations of the doors and windows). The subtraction process accounts for baseboards and tiles too. The following step consists of filling the holes with the corresponding doors and windows.

Windows may be of two types: surrounding frame windows or bottom frame windows. See Figure 7.

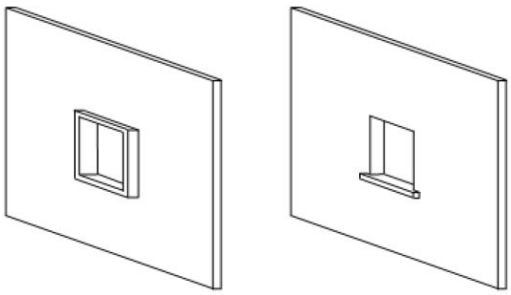

Figure 7. Window types (left: surrounding frame window, right: bottom frame window).

In the first type of windows (surrounding frame), the frame is applied to all edges, and in the second type of windows (bottom frame windows) the frame is only applied to the bottom edge.

The doors creation process is similar to the previous one (windows), differing only in the geometrical variation between these two elements. Notwithstanding there is also a difference between interior and exterior doors: conversely to interior doors, exterior doors usually do not have similar frames, i.e. the outer frames (usually in stone) are different from the inner frames (usually in wood). This is visible in Figure 8.

Figure 8. Exterior door.

5.3 Creating roofs

The final step of the algorithm consists in adding the roof structure. In a first stage, all edges of the roof are created according to the floor plan. The remaining roof is created by following the "Straight Skeleton Implementation" proposed by Felkel and Obdržálek in [Felkel et al. 1998]. With this method, a hip roof may be created following four major steps which receive as input the building floor plan, as stated by Laycock and Day in [Laycock et al. 2003]:

1. Construct the Straight Skeleton.
2. Determine the distance, d, from each vertex is to its supporting edge.
3. Perform a boundary walk, using the least interior angle, to determine the roof planes.
4. Raise the vertices according to their distance from the supporting edge.

Figure 9 illustrates this process.

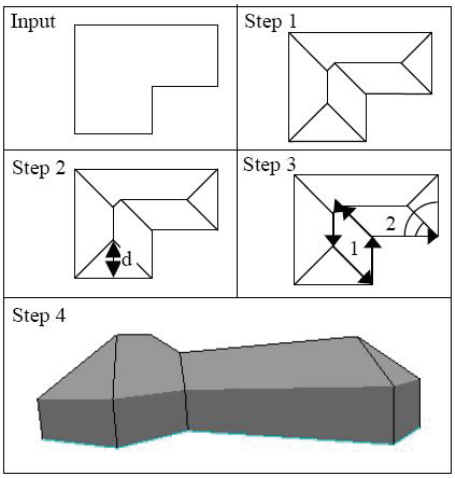

Figure 9. Hip roof modelling [Laycock et al. 2003].

Finally, to enhance the realism of the roofs, 3D half-cylinders are placed in each of the Straight Skeleton edges, to represent the roof spines.

Figure 10 presents a house roof created by this method.

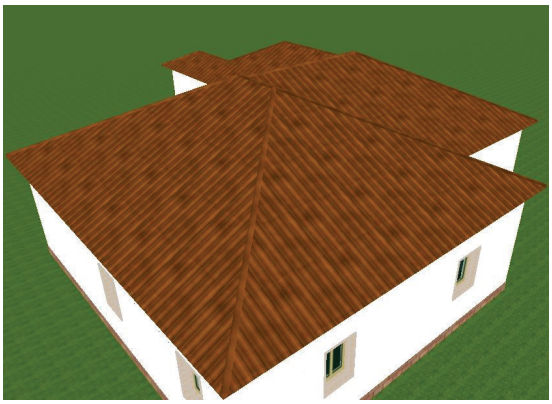

Figure 10. Hipped roof.

After the execution of all previous stages, the geometry is modelled into X3D. This technology allows the visualization of all the elements of the house through a 3D perspective, where the user can navigate through the exterior and interior of the created house. The generated models also include collision detection, proximity sensors for turning on lights and opening/closing doors, to increase the realism of the scene.

In Figure 11 a screenshot of a generated model representing the house exterior is shown. Figure 12 shows the same model viewed from the inside.

Figure 11. House exterior.

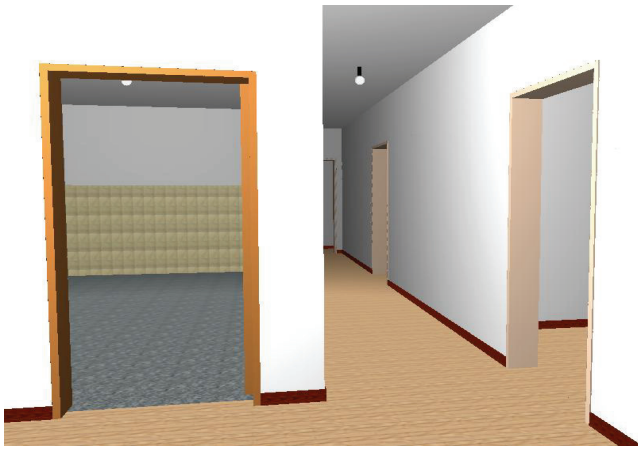

Figure 12. House interior.

6 Results

To test the proposed method a framework – ArchHouseGenerator – was developed which implements all the algorithms presented in this paper. Two different applications were also developed to demonstrate the framework.

The first application – HouseGen – was developed in C# and exploits all the available features of the framework. It includes a wide collection of options available to the user ranging from the initial specifications (e.g. house type, rooms, areas) to some more specific customizations of the final models (e.g. materials selection, colours and textures, window linings, light source positioning). A screenshot of HouseGen may be observed in Figure 13.

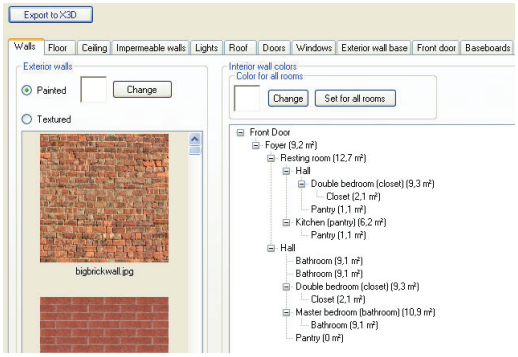

Figure 13. HouseGen.

The second application – WebHouseGen – was developed in ASP.NET, has a smaller range of options available to the user and was conceived with two different goals: demonstrate the simplicity associated with the generation of a new house and allow the widespread dissemination of the framework over the Internet. This application is available at: http://www.dei.estg.ipleiria.pt/projectosOnline/geradorEdificios/.

6.1 Performance tests

Numerous tests were conducted in order to evaluate the efficiency of the framework in generating virtual environments. Figure 14 presents a test which involved the random generation of different house types (ranging between T0 and T6) measuring the performance from 10 to 100 houses considering intervals of 10[5]. Note that the decrease in the computing times (when the number of houses increases) is due to the random generation, i.e. a significant number of simpler house types were generated.

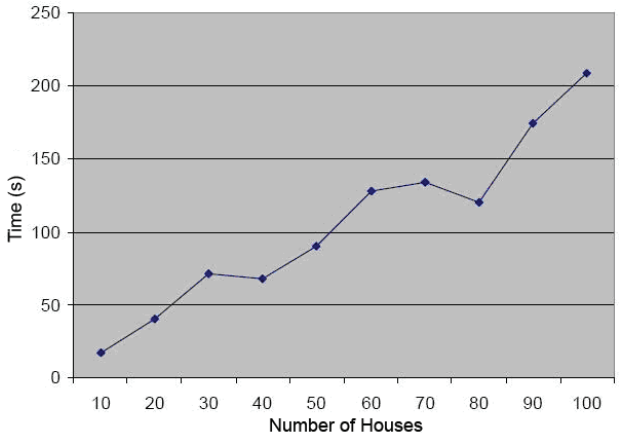

Figure 14. House generation times.

Figure 15 shows a simple virtual city consisting of different types of houses with all of their interior rooms created. The city was generated using the same hardware than the above test and has 60 different houses and consists of about 170940 polygons generated in about 18 seconds. Note that no performance issues were taken into consideration. The city may be explored from a first person perspective and the houses may be traversed.

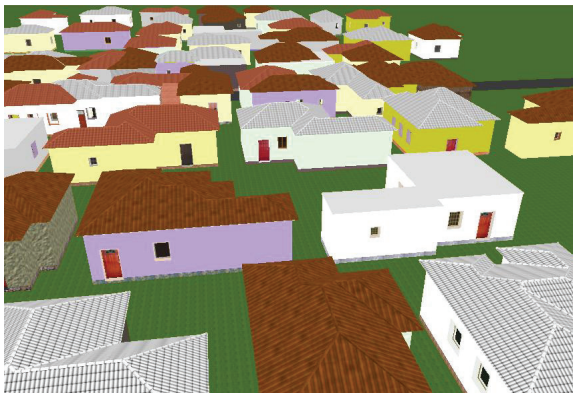

Figure 15. Generated city with 60 houses.

[5] All tests were conducted on a system equipped with an Intel Pentium 4 running at 3.2 GHz with 1 GB of RAM.

7 Conclusion and Further Work

The method presented in this paper was conceived for the procedural modelling of houses under architectural rules. A framework was implemented which is capable of generating an infinite number of different houses with a high level of detail. The system was conceived to allow enough flexibility to be extended to other areas, which may me achieved either by allowing the control of the L-system or the user-defined rule set which codifies the RGEU rules. The incorporation of these rules represents a first step into the creation of architectural applications, which may serve to produce legal floor plans.

The created models reveal that a high variety of houses can be generated in a few seconds. The results achieved lead us to believe that the presented method is suitable for generating virtual environments in some areas of application. Though the framework was initially conceived for the Architecture field, it can be easily extended to some other fields where a high degree of realism is required (e.g. virtual cities or video games).

For Architecture, specifically, even though the framework produces results according to legal rules, some of the generated floor plans reveal some lack of realism, which may be improved in the near future to make them suitable for architectonical planning. Some of the features which need improvement and new features to be added, which involve further work and the development of new algorithms, are enhanced as follows:

- Adding of rules to assure more realistic results.
- Extension to multi-floor houses and allowing the creation of enhanced façade features (e.g. balconies, ornaments, porches).
- Generation of buildings (e.g. skyscrapers).
- Allowing new geometries for the rooms of a house (e.g. circular, octagonal).
- Allowing geometric operations (e.g. rotation) on rooms of a house as well as extending the interactivity by allowing the user to operate on the different features of that house.
- Generation of furniture.

The presented topics represent only a portion of the future work involving the field of Architecture. In a near future, the aim is to extend this work to other areas, namely to the generation of time specific sites where there is some knowledge about the types of buildings and their architectural features.

8 References

ALEXANDER, C., ISHIKAWA, S., AND SILVERSTEIN, M. 1977. A Pattern Language., Oxford University Press.

DUARTE, J. 2002. *Malagueira Grammar – towards a tool for customizing Alvaro Siza's mass houses at Malagueira*. PhD thesis, MIT School of Architecture and Planning.

FLEMMING, U. 1987. *More than the sum of its parts: the grammar of Queen Anne houses*. Environment and Planning B: Planning and Design 14 (1987): 323-350.

FELKEL, P., AND OBDRŽÁLEK, S. 1998. Straight Skeleton Implementation. In *Proceedings of the 14th Spring Conference on Computer Graphics*, Budmerice, Slovakia, 210-218.

FINKENZELLER, D., BENDER, J., AND SCHMITT, A. 2005. Feature-based Decomposition of Façades. In *Proceedings of Virtual Concept*, Biarritz, France.

GREUTER, S., PARKER, J., STEWART , N., AND LEACH, J. 2003. Undiscovered Worlds – Towards a Framework for Real-Time Procedural World Generation. *Fifth International Digital Arts and Culture Conference*, Melbourne, Australia.

INGRAM, R., BENFORD, S., AND BOWERS, J. 1996. Building Virtual Cities: applying urban planning principles to the design of virtual environments. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST96)*, 83-91.

KNIGHT, T. W. 1981. *The forty-one steps*. Environment and Planning B 8 (1981): 97-114.

KONING, H. AND EISENBERG, J. 1981. *The language of the prairie: Frank Lloyd Wright's prairie houses*. Environment and Planning B 8 (1981): 295-323.

LAYCOCK, R. G., AND DAY, A. M. 2003. Automatically Generating Roof Models from Building Footprints. In *Proceedings of WSCG*, Poster Presentation.

MACIEL, M. J. 2006. Vitrúvio – Tratado de Arquitectura. IST Press.

MARTIN, J. 2005. *The Algorithmic Beauty of Buildings: Methods for Procedural Building Generation*. Honors Thesis, Trinity University.

MARTINEZ-FONTE, L., GAUTAMA, S., AND PHILIPS, W. 2004. An Empirical Study on Corner Detection to Extract Buildings in Very High Resolution Satellite Images. In *Proceedings of ProRisc, IEEEProRisc*, 25-26, Veldhoven, The Netherlands. 288-293.

MÜLLER, P., WONKA, P., HÄGLER, S., ULMER, A., AND GOOL, L. 2006. Procedural modeling of buildings. *ACM Transactions on Graphics* 25, 3.

PARISH, Y. I. H., AND MÜLLER, P. 2001. Procedural modeling of cities. In *Proceedings of ACM SIGGRAPH 2001*, ACM Press / ACM SIGGRAPH, New York, 301-308.

STINY, G.1975. *Pictorial and Formal Aspects of Shape and Shape Grammars*.Birkhauser Verlag, Basel.

Urban Simulation Team, http://www.ust.ucla.edu/ustweb/ust.html

Web3D Consortium, http://www.web3d.org/x3d/specifications/

WEIDNER, U. 1996. An Approach to Building Extraction from Digital Surface Models. In *Proceedings of the 18th ISPRS Congress*, Comm. III, WG 2, Vienna, Austria, pp. 924-929.

WONKA, P., WIMMER, M., SILLION, F., AND RIBARSKY, W. 2003. Instant architecture. ACM Transactions on Graphics 22, 3, 669–677.