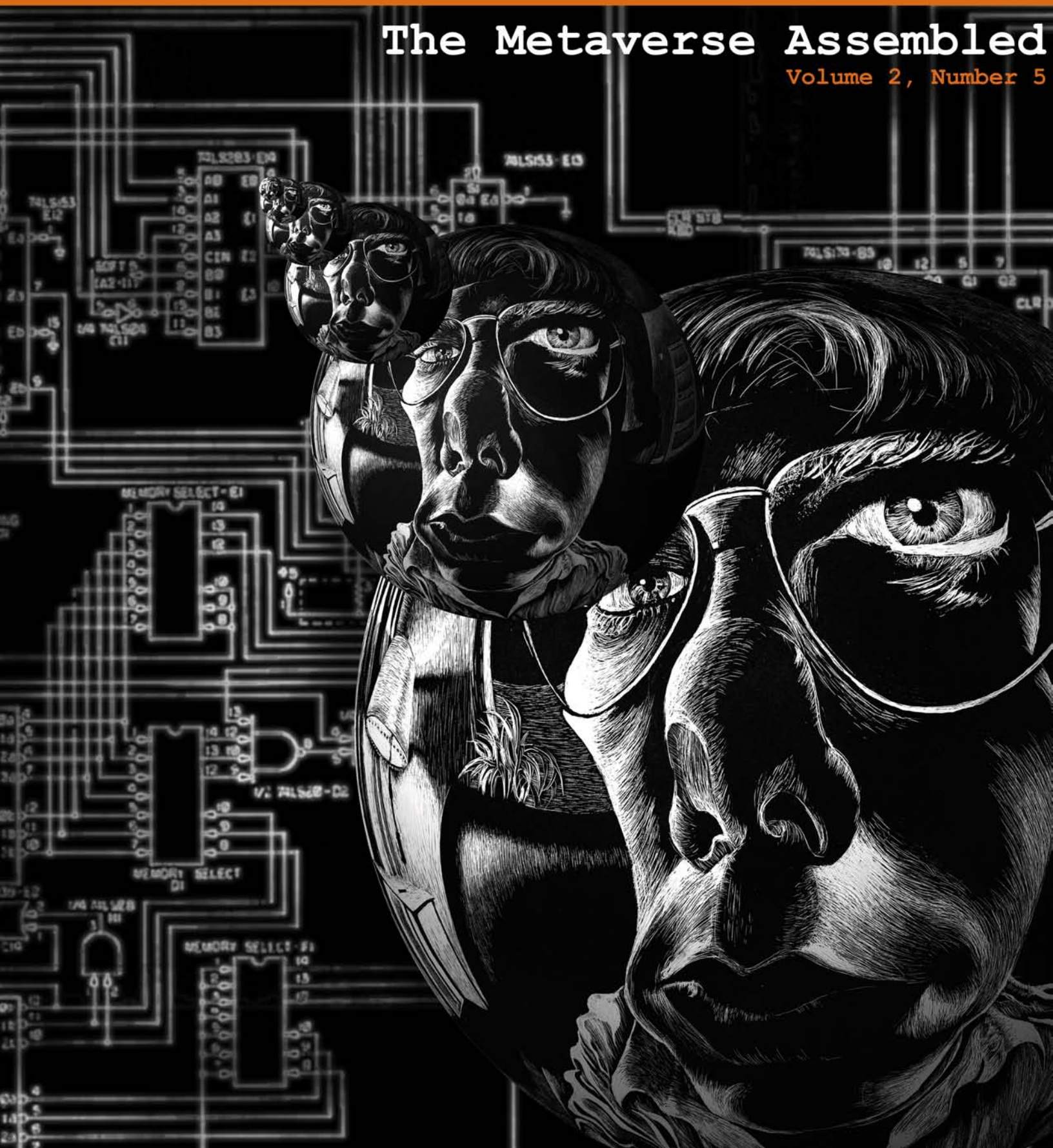
Journal of
Virtual Worlds Research
jvwresearch.org ISSN: 1941-8477

The Metaverse Assembled

Volume 2, Number 5



Volume 2, Number 5
The Metaverse Assembled
April 2010

The Journal of Virtual Worlds Research
is owned and published by the
Virtual Worlds Institute, Inc.
Austin, Texas, USA

Journal of Virtual Worlds Research
jvwresearch.org ISSN: 1941-8477

Volume 2, Number 5
The Metaverse Assembled
May 2010

ArchHouseGenerator – A Framework for House Generation

Nuno Rodrigues
Research Center for Informatics and Communications, Polytechnic Institute of Leiria, Portugal

Luís Magalhães
INESC Porto. UTAD - University of Trás-os-Montes e Alto Douro, Portugal

João Paulo Moura
GECAD - Knowledge Engineering and Decision Support Research Center, Porto; UTAD - University of Trás-os-Montes e Alto Douro, Portugal

Alan Chalmers
University of Warwick
International Digital Laboratory, WMG, University of Warwick, United Kingdom

Filipe Santos
ESECS, Polytechnic Institute of Leiria, Portugal

Leonel Morgado
UTAD - University of Trás-os-Montes e Alto Douro, Portugal

Abstract

The manual creation of virtual environments is a demanding and costly task. With the increasing demand for more complex models in different areas, such as the design of virtual worlds, video games and computer animated movies the need to generate them automatically has become more necessary than ever.

This paper presents a framework for the automatic generation of houses based on architectural rules. This approach has some innovating features, including the implementation of architectural rules, and produces 2D floor plans as well as complete 3D models, with a high level of detail, in just a few seconds. To evaluate the framework two different applications were developed and the output models were tested for different fields of application (e.g. virtual worlds). The results obtained contain evidences that the proposed framework may lead to the

development of several specific applications to produce accurate 3D models of houses representing different realities (e.g. civilizations, epochs, etc.).

Keywords: procedural modelling, virtual environments, virtual worlds, house generation.

ArchHouseGenerator – A Framework for House Generation

Nuno Rodrigues
Research Center for Informatics and Communications, Polytechnic Institute of Leiria, Portugal

Luís Magalhães
INESC Porto. UTAD - University of Trás-os-Montes e Alto Douro, Portugal

João Paulo Moura
GECAD - Knowledge Engineering and Decision Support Research Center, Porto; UTAD - University of Trás-os-Montes e Alto Douro, Portugal

Alan Chalmers
University of Warwick
International Digital Laboratory, WMG, University of Warwick, United Kingdom

Filipe Santos
ESECS, Polytechnic Institute of Leiria, Portugal

Leonel Morgado
UTAD - University of Trás-os-Montes e Alto Douro, Portugal

In the past few years, the use of algorithms to automatically generate virtual environments has become an area of growing interest for computer scientists and researchers all over the world. In fact, the idea of automatically creating environments with very little modelling effort is a fascinating idea that can lead to several benefits in different areas, such as Architecture, virtual worlds, video games and movies.

The goal is to place all, or most of the effort of creating an environment in computer software. In the context of the examples presented (i.e. Architecture, virtual worlds, video games and movies), the effort required by human resources (architects, computer artists and animators), could be greatly reduced and applied to more useful tasks, thereby leading to the development of more realist models.

This paper describes a framework – ArchHouseGenerator – conceived on the study of real structures, to demonstrate that computer algorithms for house generation can be used, even on areas where there are strict legal rules to be respected. Such is the example of Architecture where all Portuguese projects have to comply with RGEU (*Regulamento Geral das Edificações Urbanas*, General Regulations for Urban Buildings)[1]. This is just one example, since the framework may be used in other areas (e.g. virtual words and video games). Indeed, in this paper we also show some tests where the generated models were imported inside a multiuser virtual world that was built with a Virtual World SDK, OpenCroquet, This raises some interesting possibilities as the generation of *Real Virtual Worlds*, i.e. virtual worlds generated according to authentic Architectural rules. These may be contemporary worlds but also ancient worlds, representing structures lost in time, but where some kind of information about the rules of construction is available. This last reality is demonstrated with the representation of heritage structures.

[1] RGEU is one of the main documents by which all Portuguese architectural projects have to comply.

The proposed framework implements an algorithmic approach for the automatic creation of a multitude of different house geometries in just a few seconds. This includes external geometry as well as interior geometry, with a high level of detail. An application – HouseGen – was developed over the framework which allows the creation of many features present in typical houses including doors, windows, roofs and baseboards. At the same time, the features generated for the different rooms of the house reflect the characteristics of a real house. This stems from the results, where the final models (considering the example of modern houses) present tiles, both in kitchens and bathrooms, and where different kinds of materials are represented for the floors of different rooms (e.g. mosaic for kitchens and bathrooms and wood floors for bedrooms).

The following contributions are presented:

- ArchHouseGenerator produces models according to architectural and legal rules.
- The generation of models that may match different architectural styles, thus allowing the representation of buildings from different civilizations, places and epochs.
- The generation of 2D floor plans as well as complete 3D models, with a high level of detail, in just a few seconds.
- The possibility to define rules that regulate the possible paths between different parts of the house (i.e. define which parts of the house can connect to any other part of the house).

A dedicated exporter capable of generating 3D models suitable to different types of applications.

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 Laycock & Day (2003).

A different method of modelling, entitled Procedural Modelling, relies on algorithms to automatically generate the physical geometry. These algorithms usually aim for the generation of new worlds, rather than the reconstruction of existing worlds. Although they may also be used for the reconstruction of non-existing worlds for which there is 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 urban environments, where most of them are discussed by Watson et al. (2008) in addition to several other aspects, advantages and practical applications of this promising area. One initial approach to the construction and analysis of architectural design is based on shape grammars, presented by Stiny (1975). These have become the foundation for many

applications comprising different architectural styles, e.g. (Koning & Eisenberg, 1981; Knight, 1981; Flemming, 1987; Duarte, 2002). Indeed, most of the related work concerning the generation of urban virtual environments relies on grammars, e.g. L-Systems (Parish & Müller, 2001), split grammars (Müller et al., 2006).

Parish & Müller (2001) presented an approach which uses shape grammars, namely L-Systems to generate large urban environments. Wonka et al. (2003), also used shape grammars, but concentrated however on creating detailed geometric façades on individual buildings. Later, Müller et al. (2006), used the knowledge from the previous mentioned work to propose a new method for addressing the problem based on a mass model technique.

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

One common feature among most of 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.

Martin (2005) 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[2].

Hahn et al. (2006) also address interior generation, in real time, by dividing rectangular floors, corresponding to buildings' interiors, into rectangular rooms and hallways. Although, the divisions are performed randomly and have no architectural patterns into account (which doesn't seem suitable for the representation of real buildings).

The use of real rules to support the generation of virtual environments has been recently explored by the authors of this paper, either on the generation of modern structures (Author, Date) or on the generation of ancient civilizations, e.g. Roman civilization (Author, Date; Author, Date). This is demonstrated in this paper where modern structures as well as heritage structures were generated by the framework. As such, official rules, as laid out by RGEU were codified. 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.

[2]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. 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).

Likewise, Vitruvius' rules[3] were also taken into account for the generation of Roman structures, and an example of a 3D reconstruction of a Roman House is presented.

Overview

The rest of the paper is structured as follows. "Framework Architecture" section provides a general description of the framework. Examples of applications built over ArchHouseGenerator, as well as experiments are presented in the "Results" section and the paper concludes with the "Conclusion and Future Work" section.

Framework Architecture

The framework is composed by several modules represented in Figure 1. These are set off by the user design choices which guides each module until the final result is reached. The whole process is demonstrated with the case study of modern Portuguese houses.

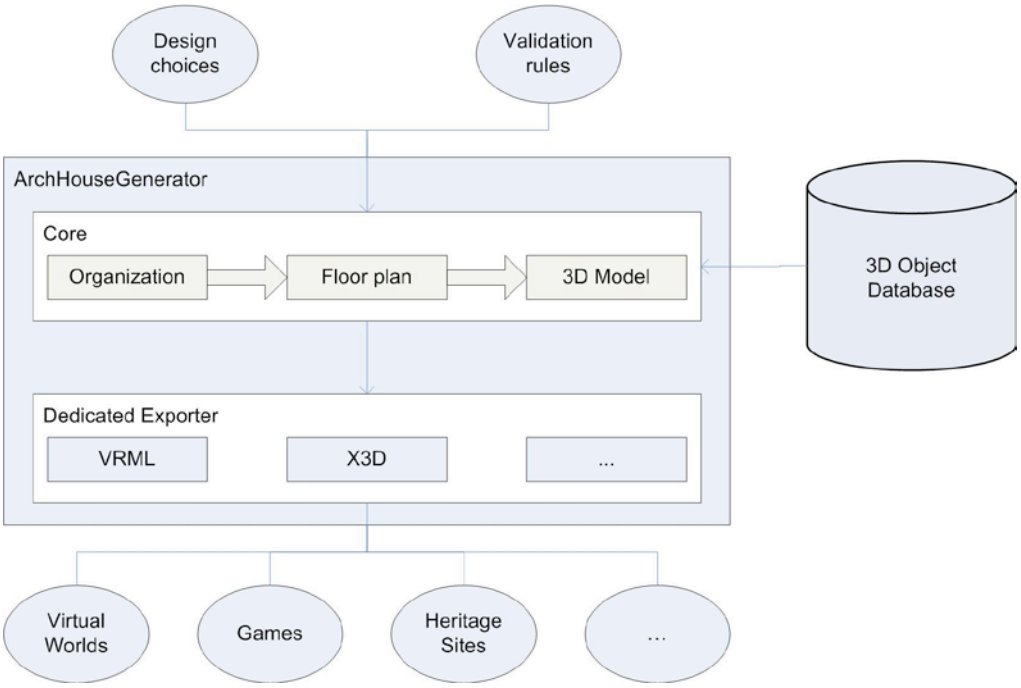


Figure 1: ArchHouseGenerator Architecture.

The design choices may match different architectural styles thus allowing the representation of buildings from different civilizations, places and epochs. The architectural style is enforced by several Validation Rules, coded in the form of a formal grammar. The object database, composed by 3D geometry representing different house features (e.g. ornaments, window styles, etc.), along with the Validation Rules may be used in this fashion to create computer programs using the framework to generate specific house types. This is demonstrated in the "Results" section, where an application – HouseGen – was developed to generate modern single, Portuguese houses.

[3] (Marcus Vitruvius Pollio – Roman architect and engineer who lived in the 1st century BC, author of "De architectura").

In the ArchHouseGenerator Core there are three major components: Organization, Floor Plan and 3D Model. These are in charge of the generation of house floor plans and corresponding 3D models according to a specific organization (i.e. room types, connections between rooms, room dimensions, etc.) specified by the user.

The output of the framework is produced by a dedicated exporter module capable of producing several formats according to the purpose of the models. Furthermore this exporter module is more than an all-purpose exporter that simply maps the geometry primitives to a specific format. Indeed, it includes a sub-module for each format which has to be capable of exploiting the optimizations of that specific format. For example if the desired format is VRML or X3D the final model ought to include these format optimization capabilities, e.g. cloning, inlining, billboards, etc. This way the exporter model is a specialized tool, which produces clever models where performance may be a critical issue, making them suitable to different fields of application (e.g. virtual worlds, games and heritage sites).

Organization

The first step performed by the ArchHouseGenerator Core is the definition of the Organization of a house which will guide the generation of the floor plan. This Organization includes the creation (when the rooms are specified by the user) or generation (when the rooms are randomly generated) of the house rooms as well as the way in which they connect to each other.

Room creation and generation

When a new house is to be built there are several issues to account for. These may have a different priority considering the person who is dealing with the problem. On one hand, if we are talking about a common person (e.g. the person who wants to order a project for a new house), the first thing that usually occurs is the determination of the desired rooms. On the other hand if the person is the one who will develop the house project, one of the first things to account for, may be the total area of the house.

In this framework an effort to include the most common choices associated with the creation of a new house was made, considering that there may be different profiles, which may have different goals, relative to the final result. Having this in mind we considered both high level decisions, as well as low level decisions, regarding the options to be chosen on a house. When considering the rooms to be created this resulted in three different modes of creating a house:

1. Constrained Random Creation – in which there is no need to specify any room of the house since they are randomly created.

2. Partial Specification – the user may specify only some of the rooms of the house (e.g. three bedrooms, one living room and a dining room) and the framework will generate a house adding more rooms to the ones already specified.

3. Full Specification – the user specifies all of the rooms of the house. The only additional rooms (beside the ones chosen by the user) may be small add-on rooms to fill small spaces in the house (e.g. closet).

In each of the previous cases the generation will take into account rules which were specified in the framework to prevent the generation of invalid houses (houses where the room lists do not represent common Portuguese houses, in the present case study). This stems from the application of the RGEU rules and rules obtained from the observation of "real" floor plans.

Connecting the rooms

Once all the desired rooms of the house are generated, there is still the need to establish the proper links between them, guaranteeing a valid path, i.e. making sure all the rooms are reachable starting from the front door. This is ensured in this step where one can specify all the valid links between the different rooms of the house. The result of this step is a graph representing the connections between the rooms.

The links may be defined either in a low level kind of way, i.e. defining the rooms which can link together (e.g. "A living room may link to a kitchen") or in a higher level, i.e. defining the room types which can be linked together (e.g. Allowed Links: Public Room, Private Room). The latter allow the creation of rules such as "A private room never serves as a link between two public rooms", meaning that a bedroom could not be used to pass from a dining room to a kitchen, which seems to make sense in today's houses.

Figure 2 represents a connection graph generated for a T2[4] house.

[4] The "T" stands for the house type, where the following number represents the number of bedrooms in the house. This number determines most of the legal rules from RGEU in Portuguese architectural projects.

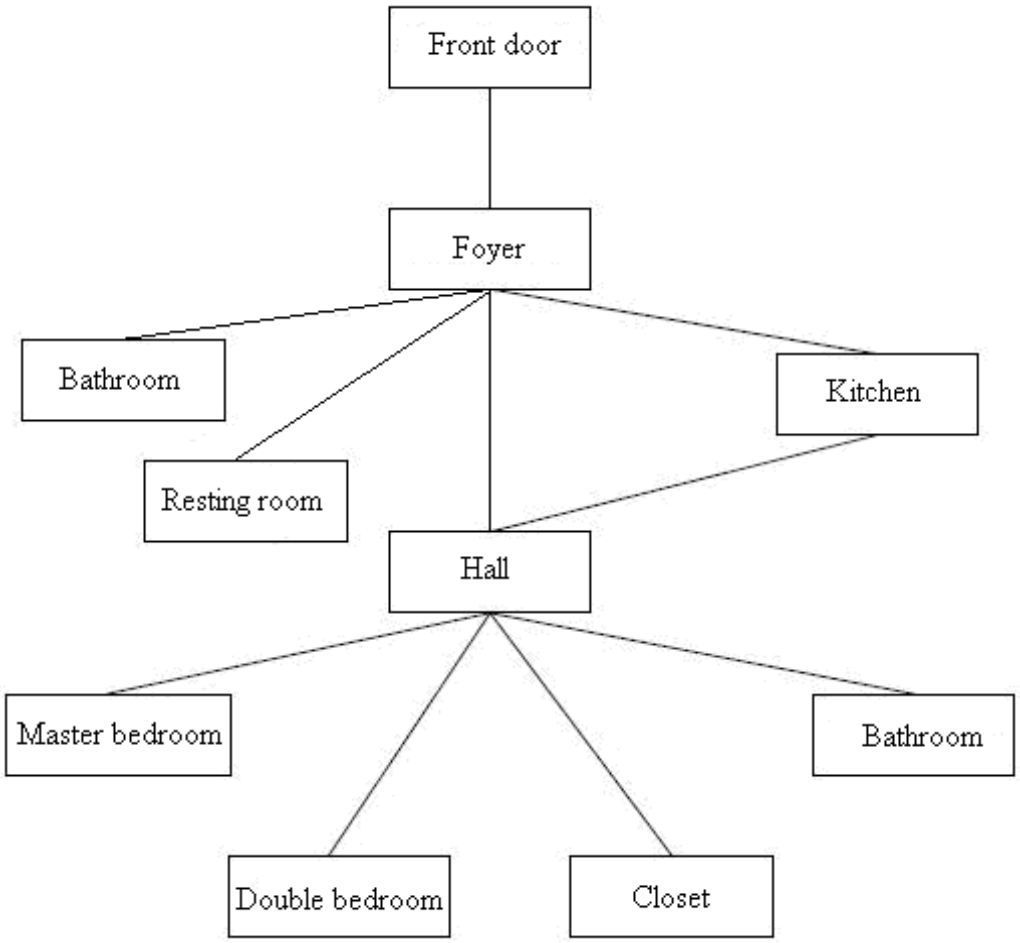


Figure 2: House graph.

It is also possible to generate several graphs until the desired structure is achieved. This step also allows circular connections to be created in the graph (which is actually why the connections are represented in a graph instead of a tree), i.e. connections between rooms which have the same origin (e.g. a foyer which links to a kitchen and to a hall which are also linked between each other – see Figure 2).

Floor Plan

After all the desired rooms of the house are generated and the links between them established, it is up to the Floor Plan to create a floor plan with all these elements.

For each room of the house, the corresponding 2D polygon is created and the final result assures that the previously established path is respected. The rooms are randomly placed, but in accordance with the links previously established between them. The generated floor plans also respect several rules such as the minimum area of each room and the correct placement of doors (including correct opening directions) and windows. Wall thickness is also taken into account during room placement.

The boundaries of the floor plan are dictated by the position and geometry of the rooms making up the house. The user can also generate several different floor plans. Therefore multiple different geometries can be obtained and the user can choose the one that pleases most. An example of a result from this process is illustrated in Figure 3.

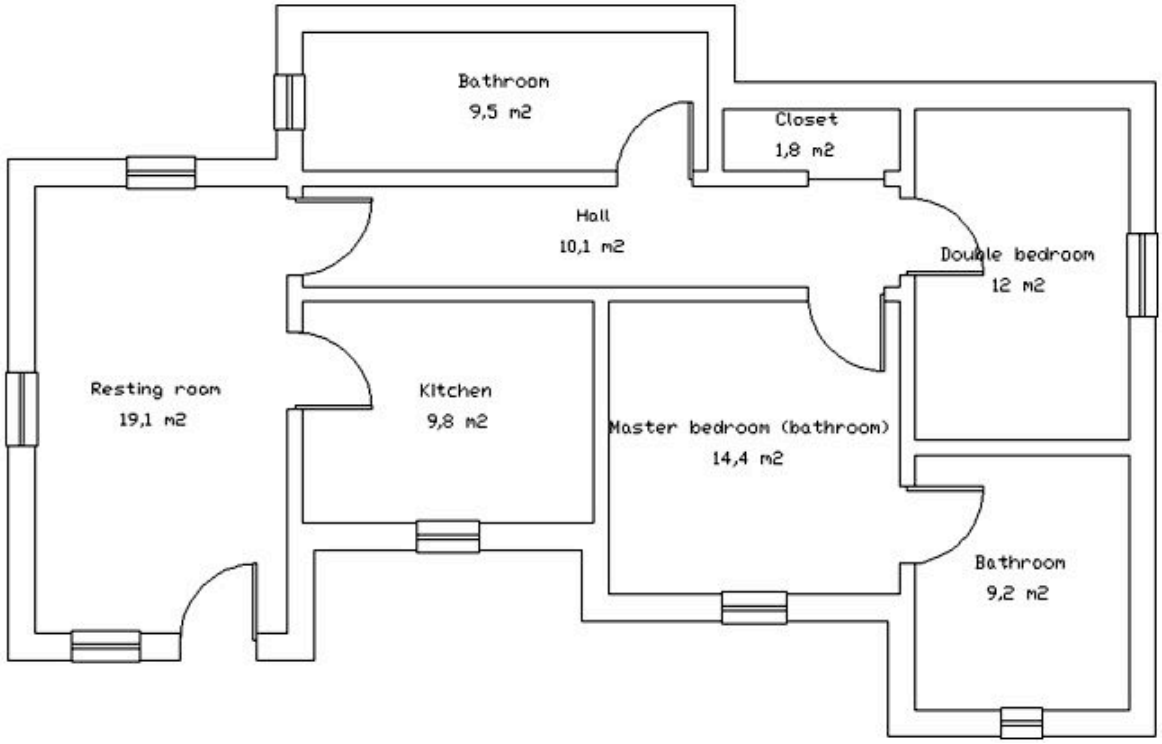


Figure 3: Generated floor plan.

3D Model

The floor plan may serve to illustrate the different parts of the house, the connections between them, the sizes of each one and even the outer geometry, but still does not give a real perspective of the house. This emerges by adding the third dimension into it.

A great deal of features may be specified for the 3D models, although this step is also capable of using appropriate default parameters for each feature of the houses.

To enhance the realism of the scene several aspects were included in the final models, like different light sources, appropriate textures for each part of the house and transparencies to represent glass surfaces (e.g. windows and doors). All of these can be individually specified by the user.

In Figure 4 a generated model representing the exterior of the house may be observed. Figure 5 represents the same house viewed from the inside.

Figure 4: Exterior of the house.

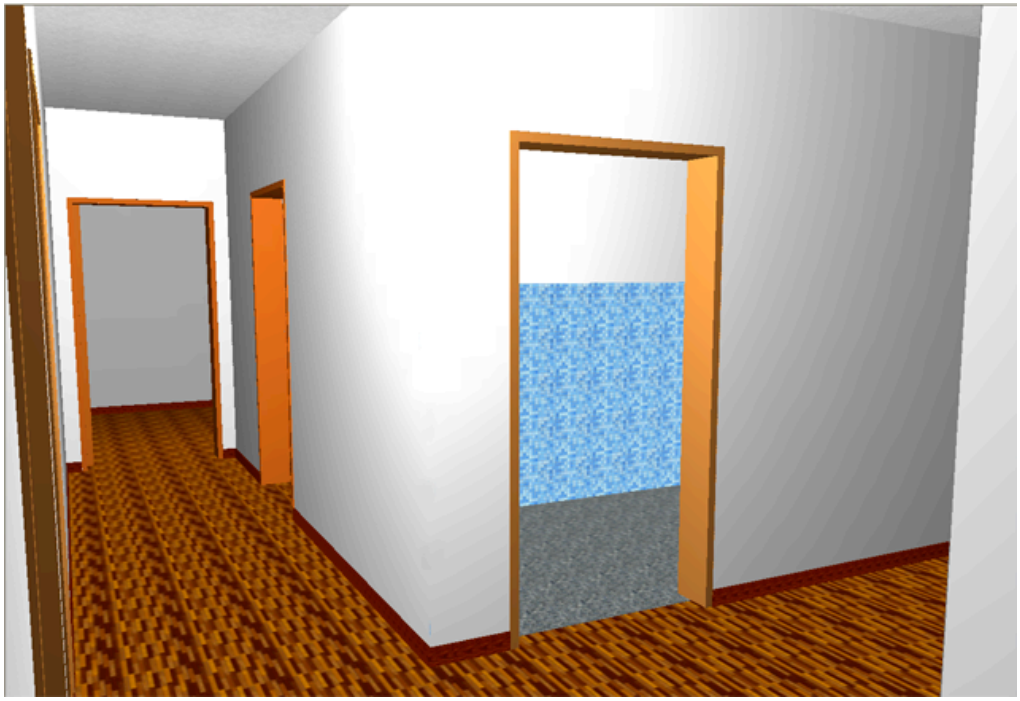


Figure 5: Interior of the house.

To represent the final models any of the Export formats provided by the framework may be used, and in a near future more formats may be added to allow a broader range of applications. Depending of the format used, as well as the goal of the models, different features are also available. For example if the format is VRML or X3D this allows the easy and widespread distribution of the results obtained with the framework over the Internet (see WebHouseGen, described in the "Results" section). Likewise, the VRML/X3D models allow the visualization of all of the elements of the house in a 3D perspective, where the user can navigate through the exterior and interior of the house. The generated models also include transparencies, collision detection, proximity sensors for turning on lights and opening and closing doors, to increase the realism of the scene.

Results

To test the functionality of the framework two applications were developed: HouseGen and WebHouseGen. Both applications have the same general goals, i.e. allowing the user to generate floor plans of houses and their respective 3D models. However the specific goal of each is somewhat the converse, since in one hand HouseGen runs on a Windows operating system and was developed to exploit most of the features of the framework. On the other hand WebHouseGen was developed to allow the widespread divulgation of the framework over the Internet and as such, offers a smaller bundle of functionalities.

HouseGen

Most of the present framework features were exploited in HouseGen. This application, developed in C#, was used to evaluate the framework usability but also to measure some performance issues.

Even though HouseGen is not a complete production-ready application but merely a prototype, there is still a great deal of options available to the user. Tasks such as defining the total area for the house or for specific rooms, configuring the rooms that will make up the house, generating floor plans which can be exported to some common formats[5] and specifying new types of houses can be performed in the application.

There are also detailed customizations available to the user that enables the configuration of the 3D models. These customizations allow the configuration of both exterior and interior features of the house, such as:

- Material selection.
- Colours.
- Roof types.
- Windows linings.
- Light source positioning.

The interior customizations (e.g. materials, colours, textures) may be applied to either all the rooms of the house or individually.

Figure 6 shows a screenshot of HouseGen, where some of the features available to the user are visible.

[5] Including both raster formats (bmp, jpeg, tiff, gif, png, wmf) as well as vectorial formats (dwg).

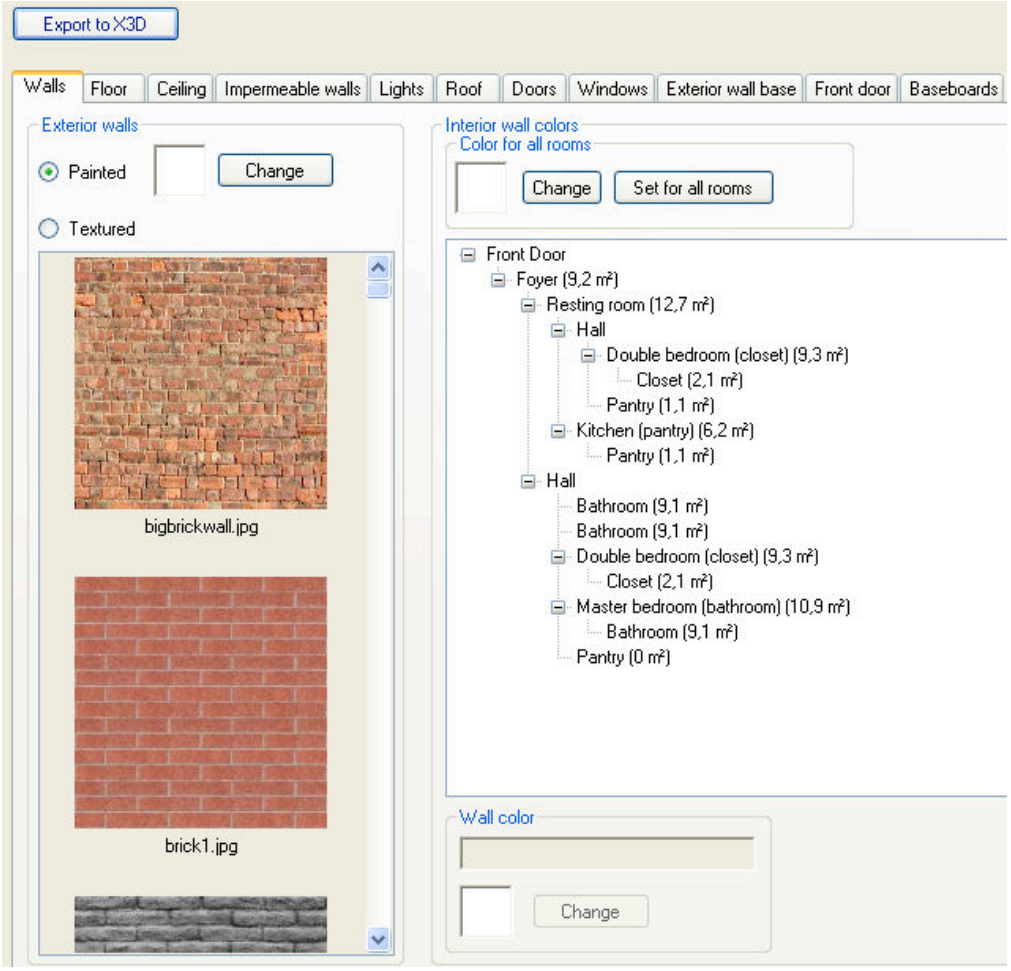


Figure 6: HouseGen.

With this application it is also possible to generate environments composed of multiple houses (whose types can also be selected).

Figure 7 shows a simple virtual city consisting of different types of houses with all of their interior rooms created. The city has 60 different houses and consists of about 170940 polygons generated in about 18 seconds[6]. 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.

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

Figure 7: Generated city with 60 houses.

Figure 12 at the end of this paper shows some of the infinite different house geometries which may be achieved with the framework.

WebHouseGen

In WebHouseGen only a few features of the framework were exploited. This application, developed in ASP.NET, was intended to show the simplicity associated with the generation of a new house, which can be done in only three simple steps: (1) Building type and room list setup, (2) Floor plan setup and (3) 3D model generation. Figure 8 presents the first of these steps.

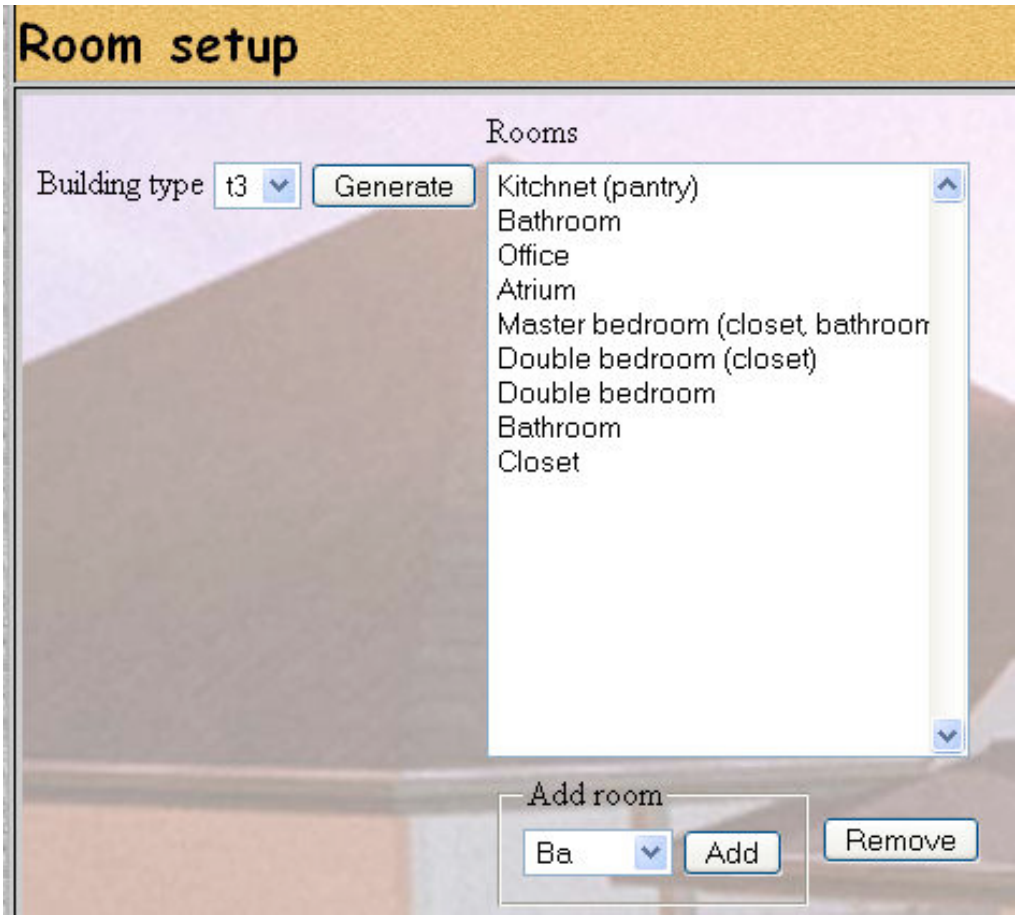


Figure 8: WebHouseGen.

As outputs the user receives the floor plans as well as the 3D models. The website is available at: http://www.dei.estg.ipleiria.pt/projectosOnline/geradorEdificios/.

Performance Tests

To measure the performance of the framework, two different tests were conducted. The first test measured the efficiency of the generation of a single house, considering different house types from a T0 to a T6. The generation times range from 0,064 to 2,913 seconds respectively as perceived on Table 1.

Table 1. Generation times for a single house.

House Type	Generation time (s)
T0	0,064
T1	0,211
T2	0,247
T3	0,982
T4	1,785
T5	2,542

T6	2,913

For the second test the goal was to measure the efficiency of the framework in generating virtual environments, i.e. several houses. The selection of houses types to generate was done randomly (between T0 and T6) and the tests accounted from 10 houses to 100 houses, considering intervals of 10. Test results are presented in Figure 9.

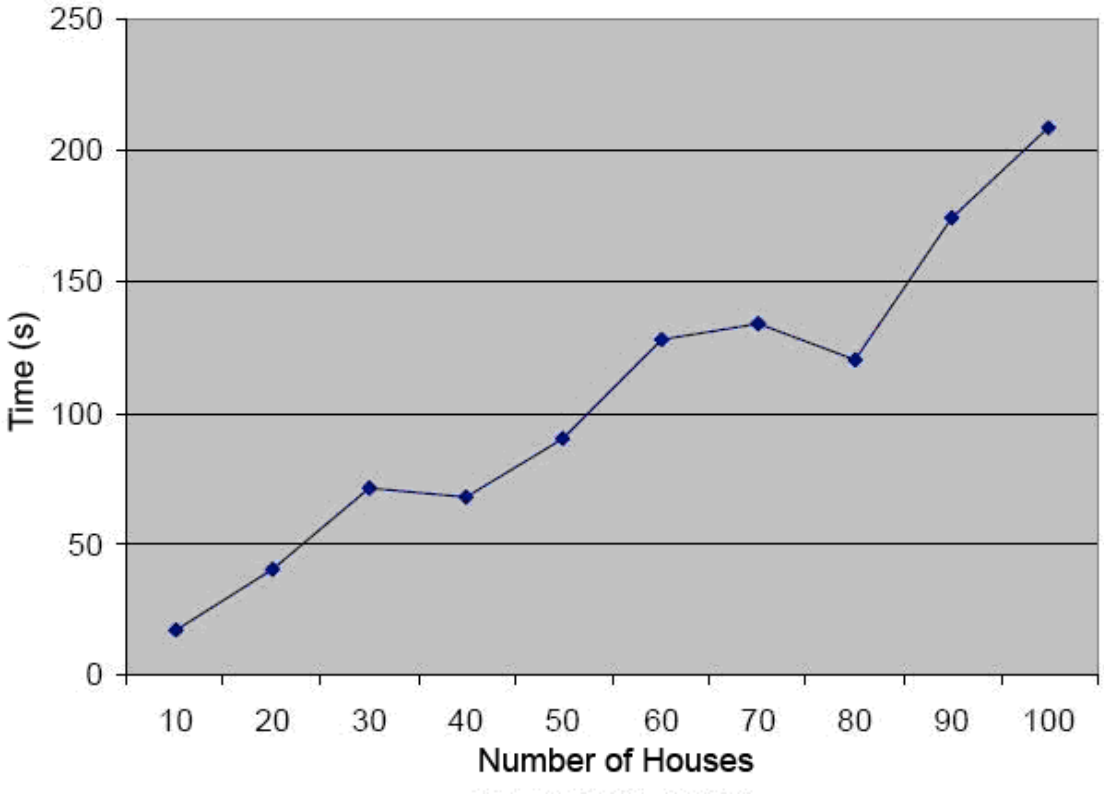


Figure 9: Generation times for multiple houses.

Note that any performance issues were taken into consideration.

Virtual Worlds

In recent years we have been seeing an increase use and academic interest in Virtual Worlds. Virtual Worlds are generally characterized as collaborative virtual environments that give their users the sensation of "being" in a space (generally 3D) and the sense of "co-presence", i.e. "the sense of others" (Bartle, 2003). Their use is consolidated in some areas, as games (where the MMORPG kind is the most common use of these environments (MMOG chart, 2009)), virtual spaces for socializing, collaborating and digital creation (being the Second Life virtual world the most popular example (MMOG chart, 2009)). Likewise, a growing number of SDKs (as OpenCroquet (OpenCroquet, 2009), OpenSimulator (OpenSim, 2009) and Sun's Project Wonderland (Project Wonderland, 2009)) for their creation is arising.

As such there is also an increase demand for tools that can generate "massive" 3D content automatically to quickly populate a virtual world (ex: buildings for a whole city). This requires automatic solutions as manual creation methods are no longer sufficient.

This framework does not intend to replace the work of computer designers or replace the creativity of virtual world users who want to design every single detail of their models. Instead, the goal is to provide means of rapidly and efficiently produce models, which may include user choices and several sets of rules, which may also be enhanced and freely modified by them.

We can foresee a set of contexts where automatic 3D modelling mechanisms based on a set of rules are useful. For example, a virtual city with hundreds or thousands of buildings can be implemented for real life simulations (ex: earthquake damages scenarios) as virtual worlds may also have powerful physics engines integrated. These can also be useful for games and movies as new scenarios could easily be seen and tested since it would only be needed to change the pre-programmed rules for building creation.

Although some worlds give their users tools to develop 3D content (with scripts for behaviour if desired) there are others that rely on 3rd party tools for this task, as Blender or 3D Studio Max, giving the user "import content" tools. As this may be desirable, giving the user the liberty to use its favourite modelling tool, the lack of a common 3D modelling format, for the several virtual worlds, could also be a challenge as some problems of conversion between formats still exist (demonstrated shortly). Due to these issues, some virtual worlds are opening their platforms to more creators by using open-standards. One example is Vivaty, a "web-based virtual world platform that is built entirely on open standards" (Vivaty, 2009). Indeed, Vivaty allows the users to create worlds in standards like X3D and VRML (supported by ArchHouseGenerator), also providing a tool – Vivaty Studio – which allows the user to import several other formats.

We have tested some of our models in a virtual world developed with virtual World SDK, OpenCroquet, a tool that "can be used by experienced software developers to create and deploy deeply collaborative multi-user online virtual world applications on and across multiple operating systems and devices" (OpenCroquet, 2009). As this SDK is still on an early release it does not give great support for the most common 3D formats. Indeed, though this SDK supports some formats provided by ArchHouseGenerator, such as VRML, the authors have experienced some problems in the correct import of the model. This led to rely mainly in the ASE format (3D Studio Max ASCII Scene Export) as this is one of the formats that it is easily imported by this SDK.

Although the ASE models have rendered generally well in the virtual world (see Figures 10 and 11) it was stated that using some of the most common conversion tools available to convert the original model (VRML) to the ASE format (that was imported into the virtual world) would sometimes give unpredictable results as some features of the original format were lost. This is visible in Figure 10 where the columns and flowers, previously formed by billboards and transparent textures in VRML, lost their veracity when converted to the ASE format and imported into the virtual world. The figure shows the rendering of one model using a virtual world developed in OpenCobalt (OpenCobalt is currently being developed by the OpenCroquet community as a future "metaverse browser" (OpenCobalt, 2009)) corresponding to the virtual reconstruction of the "House of the Skeletons", a Roman house, for which presently only the ruins exist at the heritage site of Conimbriga, Portugal.

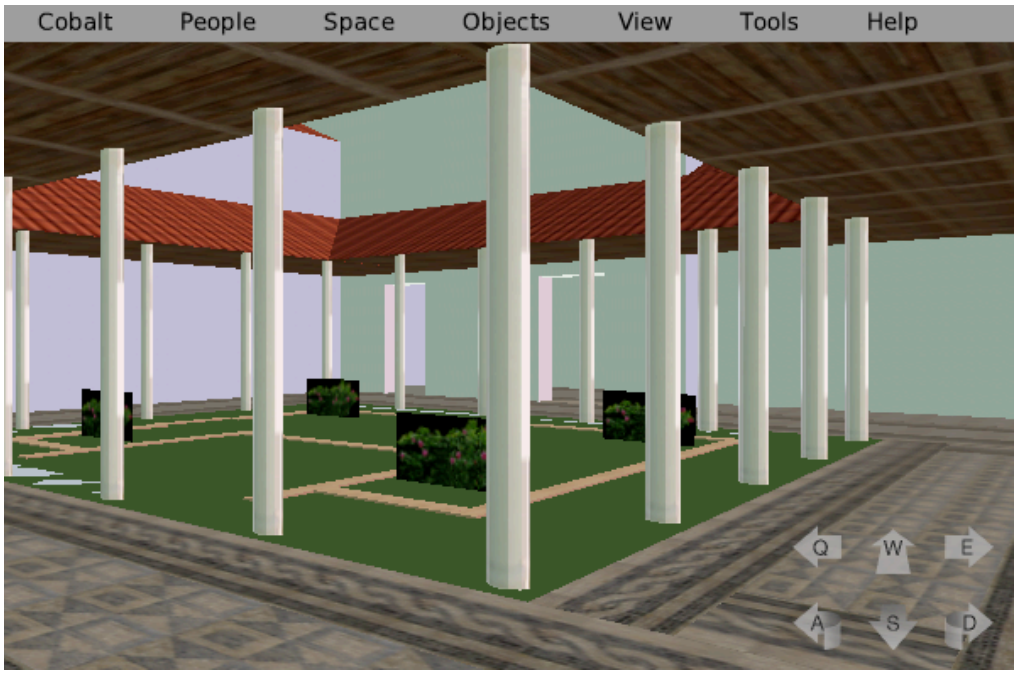


Figure 10: House of the Skeletons (in OpenCobalt).

The reconstruction considered rules derived from the knowledge left by Vitruvius, mostly trough the reading of the Portuguese adaptation of "De architectura" from M. Justino Maciel "Tratado de Arquitectura" (Maciel, 2006). A study of Roman Architecture, along with the director of the Monographic Museum of Conimbriga – Dr. Virgílio Hipólito Correia, with the goal of determining several options regarding the reconstruction of the House of the Skeletons, is also responsible for most of the options to produce the results presented. This example also serves the purpose of demonstrating the use of the framework to generate heritage structures.

Figure 11 shows our own designed OpenCroquet virtual world where a house produced with HouseGen was imported and where its multi-user capabilities were also tested.

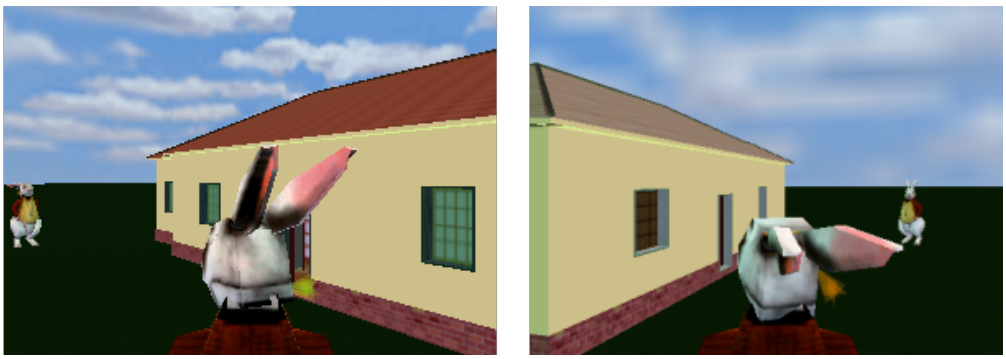


Figure 11: T3 House (in an OpenCroquet multiuser world).

As it can be seen we show how the world was rendered in two different machines synchronously – allowing the simultaneous presence of two users (here represented by rabbits – each one is seeing the other). This is one of the main advantages of shared virtual worlds as

by allowing the presence of multiple users they allow a great set of collaborative activities (by the simultaneous presence and interaction of several users).

Conclusion and Future Work

The framework presented in this paper was initially conceived to demonstrate the use of computer algorithms for ruled based house generation. The results achieved are in accordance with the initial goals, whereby most of the 3D models produced present many features, with a high level of detail, comparable to real houses.

ArchHouseGenerator also revealed to be capable of generating several distinct house types, with different characteristics (e.g. rooms, areas, paths), which seem adequate for the creation of virtual environments. Figure 12, at the end of the paper, shows different houses types representing the variety which may be achieved with the proposed framework.

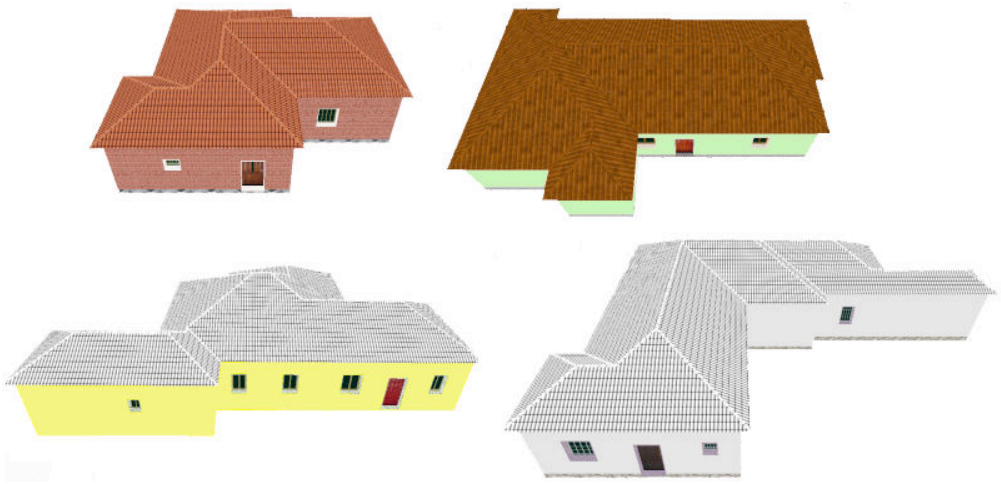


Figure 12: House variety.

We can conclude that the proposed framework is suitable for some areas of application, where Architecture and virtual worlds were emphasized. Nevertheless it may also be used, in the near future, in other areas where some level of realism is required such as video games and movies.

There are some details which need improvement, and new features to be added, either on the framework itself or the presented applications, such as:

- Allowing different geometries for the rooms of the house (e.g. circular, octagonal).
- Allowing geometric operations (e.g. rotation) on room geometries.
- Extending interactivity by allowing the user to manipulate different characteristics of the house.
- Generation of multi-floor houses.

- Generation of furniture which allows an easier identification of the different rooms of the house.
- Creating enhanced façade features (e.g. balconies, ornaments, porches).
- Add new export formats.

These represent a fraction of the identified future work topics, since from here there are several new features to be added as well as new areas of application for which the framework may be used in a near future.

Bibliography

Alexander, C., Ishikawa, S., & Silverstein, M. (1977). A Pattern Language. Oxford University Press.

Bartle, R. (2003). Designing Virtual Worlds, ISBN 0-1310-1816-7. New Riders Publishing.

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.

Finkenzeller, D., Bender, J., & Schmitt, A. (2005). Feature-based Decomposition of Façades. Proceedings of Virtual Concept, Biarritz, France.

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.

Greuter, S., Parker, J., Stewart, N., & Leach, G. (2003). Undiscovered Worlds – Towards a Framework for Real-Time Procedural World Generation. Fifth International Digital Arts and Culture Conference, Melbourne, Australia.

Greuter, S., Parker, J., Stewart, N., & Leach, G. (2003). Real-time Procedural Generation of 'Pseudo Infinite' Cities. International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia.

Hahn, E., Bose, P., & Whitehead, A. (2006). Persistent Realtime Building Interior Generation. In Proc. of ACM Siggraph Symp. on Videogames, ACM Press, pp. 179–186.

Ingram, R., Benford, S., & Bowers, J. (1996). Building Virtual Cities: applying urban planning principles to the design of virtual environments. Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST96), 83-91.

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

Koning, H., & Eisenberg, J. (1981). The language of the prairie: Frank Lloyd Wright's prairie houses. Environment and Planning B 8 (1981): 295-323.

Laycock, R., & Day, A. (2003). Automatically Generating Roof Models from building Footprints. Proceedings of WSCG, Poster Presentation.

Maciel, M. (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., & Philips, W. (2004). An Empirical Study on Corner Detection to Extract Buildings in Very High Resolution Satellite Images. Proceedings of ProRisc, IEEEProRisc, Veldhoven, The Netherlands. 288-293.

MMOG Chart, http://www.mmogchart.com/, retrieved on 29-07-2009.

Müller, P., Wonka, P., Hägler, S., Ulmer, A., & Gool, L. (2006). Procedural modeling of buildings. ACM Transactions on Graphics 25, 3.

OpenCobalt, http://www.opencobalt.org/, retrieved on 29-07-2009.

OpenCroquet, http://www.opencroquet.org/, retrieved on 29-07-2009.

OpenSim, http://opensimulator.org/ , retrieved on 29-07-2009.

Parish, Y., & Müller, P. (2001). Procedural modeling of cities. In Proceedings of ACM SIGGRAPH, ACM Press / ACM SIGGRAPH, New York, 301-308.

Project Wonderland, https://lg3d-wonderland.dev.java.net/, retrieved on 29-07-2009.

Second Life, http://www.secondlife.com/, retrieved on 29-07-2009.

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, retrieved on 29-07-2009.

Vivaty, http://www.vivaty.com/, retrieved on 29-07-2009.

Watson, B., Müller, P., Veryovka, O., Fuller, A., Wonka, P., & Sexton, C. (2008). Procedural Urban Modeling in Practice. IEEE Computer Graphics and Applications, Vol. 28, No. 3, page 18-26.

Weidner, U. (1996). Proceedings of the 18th ISPRS Congress, Comm. III, WG 2, Vienna, Austria, pp. 924-929.

Wonka, P., Wimmer, M., Sillion, F., & Ribarsky, W. (2003). Instant architecture. ACM Transactions on Graphics 22, 3, 669–677.