Modeling Urban Environments from Geospatial Data

A Pipeline for Procedural Modeling

Diego Jesus
Faculdade de Engenharia,
Universidade do Porto
Rua Dr. Roberto Frias s/n
4200-465 Porto, Portugal
diego.jesus@fe.up.pt

Antonio Coelho
INESC TEC / DEI, Faculdade de Engenharia, Universidade do Porto
Rua Dr. Roberto Frias s/n
4200-465 Porto, Portugal
acoelho@fe.up.pt

Carlos Rebelo
3Decide / Palcos da Realidade
Praca Coronel Pacheco, 2
4050-453 Porto, Portugal
carlos.rebelo@3decide.com

Andre Cardoso
3Decide / Palcos da Realidade
Praca Coronel Pacheco, 2
4050-453 Porto, Portugal
andre.cardoso@3decide.com

ABSTRACT

In game development there is often the need to generate realistic urban environments, i.e. 3D virtual environments that replicate existing urban areas. However, modeling such spaces using traditional techniques is both too slow and too expensive. A good solution is the use of procedural modeling techniques to automate the process. However these techniques require large amounts of geospatial data, which are usually stored in Geographic Information Systems (GIS).

This paper presents a pipeline for the integration of both geometric and semantic data from GIS data sources into procedural modeling techniques used for the generation of 3D virtual urban environments. GIS data can already be used in procedural modeling tools but these do not provide an easy and uniform way to incorporate semantic information from different data sources. To solve this problem, the proposed pipeline is capable of transforming semantic and geometric information from different sources into 3D environments that replicate specific urban areas.

Categories and Subject Descriptors

I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Modeling packages, Geometric algorithms, languages, and systems*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Virtual reality*; H.2.8 [**Database Management**]: Database Applications—*Spatial databases and GIS*

General Terms

Computer Graphics, Geospatial Systems

Keywords

Procedural modeling, GIS, Virtual urban environments

1. INTRODUCTION

Procedural modeling techniques can be used for the generation of virtual urban environments, reducing both the amount of time and money that would be spent creating the 3D models populating these environments (houses, roads, urban furniture, etc). This is an obvious advantage over traditional modeling techniques for the game development process. Research in this area has provided good results in the generation of fictional urban environments. However, there are still some issues when using these techniques for the modeling of existing urban spaces. The main reason is that procedural modeling techniques require a great amount of semantic information to produce environments that resemble closely enough the real ones. Often, the lack of information is solved with the introduction of some randomness in the process. This technique may, however, result in an unpredictable set of models that may look different from reality.

City municipalities often store information about their urban space in GIS. This data can be accessed and processed in order to recreate existing urban spaces. Since different municipalities usually have different data models, feeding directly this data to the procedural modeling mechanisms leads to the need of adapting the modeling processes for each new case. Also, when generating an urban area containing elements belonging to distinct municipalities, we may need to have production rules to generate similar elements from each data source.

In order to provide a more structured solution, the proposed pipeline first maps the original data into an Unified Model. Since different municipalities may have distinct amounts of information for each feature, there might be a need to amplify some of the data. This way all elements that feed the modeling processes are mapped to an uniform level of associated semantic data. Only then, all the normalized data is fed to the procedural modeling tool in order to generate models that represent the urban area.

This paper is organized as follows: section 2 makes a brief overview of related work and section 3 presents the pipeline proposed, describing the Data Mapping, Data Conversion and Amplification, and the Procedural Modeling steps. Section 4 presents an overview of the pipeline's distributed architecture and in section 5 we discuss the results obtained. In Section 6 we provide the conclusions and future work.

2. RELATED WORK

Procedural content generation (PCG) refers to creating game content automatically, through algorithmic means. Procedural modeling of urban environments is a subset of this research area that focuses on generating 3D models of the distinct elements that compose an urban environment (buildings, roads, urban furniture, etc). This line of research originates from the work of Parish and Muller [9], where the generation of the virtual urban environments were based on L-systems. This mathematical tool, introduced in [4], was originally designed to model the growth process of simple multicellular organisms such as algae and fungi and that was later adapted to model plants [8]. As such, L-Systems are preferably used in growth processes in open spaces. Buildings, however, tend to have more strict spatial limitations like a bounding volume. To solve this limitation, split and control grammars were introduced [11] based in the concept of shape. The former subdivides the spaces in each derivation step until terminal symbols are found, representing the geometry, while the latter influences the choice of the rules to apply in the derivation process. This concept was further refined, leading to the CGA Shape Grammar (Computer Generated Architecture) [7]. This grammar is capable of producing extensive architectural models with high detail. The implementation of CGA Grammar is integrated in the CityEngine modeling tool.

A common limitation in these methods is the reduced or nonexistence of spatial awareness, meaning that these systems are not well adapted to perform queries to an object's surroundings. Geospatial L-Systems [3] were introduced to solve this limitation, as an extension of parametric L-Systems combining the potencial of data amplification provided by L-Systems with the spatial awareness of GIS. Procedural modeling techniques based on formal grammars require the development of production rules that incorporate knowledge on the modeling process. This is a complex process for which some visual authoring tools have been proposed [5].

Other alternatives have been proposed to enable designers to concentrate on stating what they want to create instead of on describing how they should model it. These approaches range from interactive system for synthesizing urban layouts by example [1], designing an underlying tensor field for editing the graph representing the street network [2] and to interactive procedural sketching using the framework SketchaWorld [10].

CityEngine allows the creation of attribute layers that associate a value to a specific position in the terrain. These values can be associated with different production rules parameters which in turn influence the appearance of the urban space. With these layers the user is able to control the growth of road networks, selections and attributes of the production rules of the CGA grammar [7]. But the limitation of using values in raster maps reduces the amount of semantic information that can be used. However, this tool is also able to import GIS files with geometry and semantic data directly, feeding such data to the rules' attributes. This is also the approach followed by Geospatial L-Systems [3]. Nonetheless, this implies the creation of specific rules for each case, or at least the modification of existing ones, leading to an increased effort.

To facilitate the procedural modeling of urban spaces from different GIS data, an Urban Ontology was introduced in [6]. Furthermore, since each municipality might have different amounts of semantic data for the same type of urban elements, this led to the creation of the Level of Mapping (LOM) concept, which indicates the minimum required information for each type of urban elements.

The work herein described provides a set of tools and processes capable of gathering GIS data from several sources, and efficiently generating 3D environments through the use of procedural modeling techniques. By using GIS data maintained by city municipalities, it is possible to produce virtual urban environments that closely resemble real cityscapes using procedural modeling techniques.

3. PIPELINE FOR GEOSPATIAL DATA

To meet the research goals, we propose a pipeline capable of generating accurate 3D environments based on GIS data incorporating semantic attributes. This pipeline can be implemented in a distributed architecture and can be easily integrated into procedural modeling tools, provided they have an interface that allows such integration. The modeling tool selected as a case study was CityEngine. The pipeline consists in the sequential execution of three processes: Data Mapping, Data Conversion and Amplification, and Procedural Modeling. Figure 1 shows an overview of the proposed pipeline and its several steps.

The Data Mapping stage is responsible for mapping the original GIS data into an unified data model based on an Urban Ontology [6]. This is a necessary step due to the fact that different municipalities might have different data models containing distinct sets of information. Without this mapping, the rest of the pipeline would become very vulnerable to such differences, leading to the need of creating different processes for every municipality.

After the conclusion of the Data Mapping stage, the semantic and geometric information regarding the municipality's urban elements is stored in the unified model. However, as stated previously, the amount of information present in the original data can be different for each municipality, which leads to different LOMs [6]. A LOM is a level of modeling, i.e. a measure of the potential of the data to achieve a specific level of visual fidelity. To maintain the same procedural modeling processes in the next stage of the pipeline, these should be provided with the same amount of semantic information for every element, i.e., a specific LOM. As such, the need to convert and amplify existing data arises.

The final step is responsible for the procedural modeling

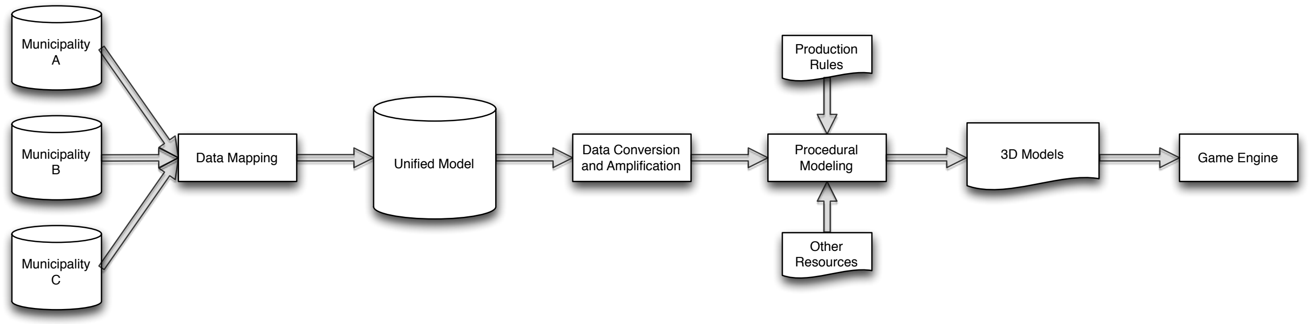

Figure 1: Overview of the proposed Pipeline

of the urban space itself and is conducted by a procedural modeling tool such as CityEngine. This takes the geometric and semantic data provided by previous steps and generates three dimensional models representing the city environment. Several procedural modeling tools could be used as long as they provide a means of external manipulation (e.g. through some sort of *API*).

Each of these steps in the pipeline will be discussed in more detail in the following sections.

3.1 Data Mapping

As previously mentioned, different municipalities might have different data models. On the other hand, it is also possible that the information contained in their Geographic Information Systems is incomplete or in a format that makes the Data Mapping process complex. There may also exist the need to relate several pieces of information or geometries in order to obtain new sets of data that are richer and more complete. For this reason, a flexible method to work on different scenarios is demanded.

To solve this problem, we introduce the concept of Mapping Module. Mapping Modules are capable of accessing GIS data sources and convert the stored data relative to a single type of urban entities into a format compatible with the Unified Model and store the new information in such model. They are also responsible for providing the transformed data to other modules depending on it. This kind of dependency requires the modules to be executed in a specific order, where one module can only depend on information provided by previously executed Modules. Which modules are executed, and in which order, is specified externally by the user, as it is specific to each situation.

As mentioned, one module is responsible for mapping one, and only one, type of urban elements. However, these elements aren't always stored in the same manner, leading to the need that different modules process the same kind of entity. The set of mapping modules that execute the mapping of the same type of entities is called a Family of Modules. Each family provides methods that allow other Modules to query the processed data. There are several Families in the system, corresponding directly to the types of entities in the Unified Model, specified in [6]. There are, for example, Families dedicated to mapping of buildings, roads, vegetation and other urban entities.

Consider, for example, the data relative to the city terrain. One municipality might store such information as a cloud of points with height data associated, while another might store contour lines. There is clearly a need for two different algorithms to process each format and, as such, two modules must be implemented. Since they map the same kind of data, they belong to the same Family of Modules, in this case the Terrain Family. After processing the respective data, both Modules can be queried, in the same fashion, for the height of a specific point in the terrain.

The Family of Modules concept creates a logic separation between different modules, since one module does not need to know about the specific implementation of the possible modules on which it depends. All it needs to know is what Family it belongs to and then, call the respective methods. The system maintains a registry of modules allowing the search for Modules based on their family. Since it would make little sense mapping the same kind of elements more than once, the system only allows, at most, one module of each Family to be executed at a given time.

The execution of a module starts with the retrieval of all the information from the data source. Afterwards, it processes this information, communicating with other modules if necessary. Finally it stores the created elements in the Unified Model. Although it is possible to process the data while reading from the source, it may be desirable to treat the information as a whole and infer relations between all the urban elements. The processing phase works on both the geometric data (e.g., building lots) and the semantic data, and creates new entities with an axiom – which will be fed to the procedural modeling on a following stage – and a set of attributes. However, in simple cases where a complex algorithmic component is not needed, the mapping module can be externally configured to create new attributes based on one of two actions: Copy and Set. The former, simply copies some data from the original data source to a new attribute, while the latter sets a new attribute with a specified value.

Sometimes, municipalities may store data that may prove to be uninteresting, such as information about buildings that are not yet built. If this data was allowed to be mapped into the Unified Model, it could lead to unexpected results like overlapping buildings or other incorrect urban elements. As such, mapping modules can be configured with filters to remove those elements from the process.

The mapping modules can also be configured by means of a *XML* file, allowing that an arbitrary number of parameters to be passed to the specific algorithms and filters. This way, even if there is a need to develop new modules for different situations, if the algorithms are general enough and can be configured through the *XML* file, in the long term we expect to gather a collection of modules for the most common scenarios.

By the end of the Data Mapping stage, the Unified Model contains all the new urban elements needed to correctly model the urban environment. Each of these elements will be in a specific LOM, based on the amount of information that could be extracted from the GIS data source.

3.2 Data Conversion and Amplification

To facilitate the procedural modeling processes, these expect to be fed with the same amount of information for a given type of urban element. In other words, they expect the same LOM for all elements of a given type. However, as has seen before, these may be stored in the Unified Model with an arbitrary Level of Mapping. Thus, a mechanism to convert entities between different LOMs is required and, as such, the concept of LOM Converter is introduced.

LOM Converters are responsible for converting data between different levels of information, guaranteeing that the entities affected are output with a specific Level of Mapping. This is, they ensure the entities will have at least the minimum semantic information required by the given LOM. Figure 2 shows a possible building modeled in LOM 1 and the same building after the LOM Converter was applied. LOM 1 only contains information regarding the building's volume while LOM4 contains information that is much more complete.

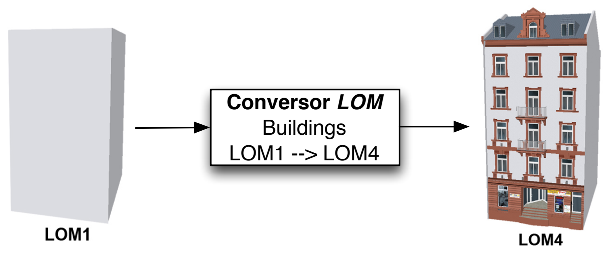

Figure 2: Building LOM 1 to LOM 4 Converter

Given that the definition of Level of Mapping [6] does not impose a maximum level of information for a given LOM, the conversion from higher LOMs to lower ones does not make sense. However converting from lower LOMs to higher ones requires the amplification of the existing information. This can be achieved through the introduction of randomness in this process, or with the use of heuristics.

Imagine the case where the production rules for buildings need information regarding roof types but there is no such information stored in the Unified Model. We can associate a random roof type to every building. However, this may lead to small houses with flat roofs or sky scrapers with gabble roofs. Although it is possible, it may seem unfamiliar. A better approach would be to define an heuristic and take into account the building height.

On the other hand, it might be necessary to create new production rules (e.g., because there is a need for a specific architecture) which may expect to be fed with more information. It is also possible that some attributes don't belong to any Level of Mapping. To accommodate for these scenarios, this stage in the pipeline can be configured through a XML file specifying how new attributes can be created. This file allows to specify random values for the attributes, copy them from those present in the Unified Model and relate the existing information to infer new data through the use of Data Amplification Operators which resemble those of a programming language, including relational operators, *if-then-else* statements and a domain specific operator which allows to specify a building's facade. This last one works by allowing the user to indicate the number of existing openings (windows and doors) in every floor and side of the building.

This way, the user can specify new data amplification mechanisms for specific cases. However, this file cannot alter the attributes created by the LOM Converters. In the same fashion, this file can also specify the file containing the production rules to model an element. This is particularly important when modeling monuments for instance.

This stage of the pipeline also takes the responsibility of manipulating CityEngine, turning it into a tool of the pipeline thus reducing human interaction. In other words, this step will control what operations CityEngine will execute with the information it is consuming. Among these operations we can name, for example, the creation of a shape or a street segment, geometry generation, or the terrain alignment to a set of shapes. These were called CityEngine Manipulation Operators. However, different cases may need different operations to take place, in order to correctly generate the urban environment. As such, these operations and their execution sequence can be configured externally using a file.

3.3 Modeling Processes

The last stage in the proposed pipeline is the execution of the procedural techniques which will generate the three dimensional models representing the urban space. These processes are fed with information from the previous steps. Normally this information consists of a geometric axiom (the shape) and a set of semantic attributes, but can also be a terrain or an attribute map. An example of the latter is the municipality's land use map which can control the types of buildings.

To create the three dimensional models representing the space, the procedural techniques are controlled by a set of CGA rule files, one for each type of urban elements. These, as mentioned before, expect a set of attributes to be passed from the earlier stages of the pipeline.

Often it is desirable to control the Level of Detail (LOD) of the produced geometries, either for efficiency reasons or to speed up the modeling process. As such, the production rules provide an attribute to control the LOD. Also, several levels were defined for each of the types of urban elements.

As seen before, it is possible to define the file responsible for modeling an element. However, this can lead to CGA code being duplicated among files. Imagine, for example, that a city's characteristic building is being modeled and, therefore, needs a different rule file. Even though it may be quite different from other buildings, it may contain similar elements such as windows, doors or arcs, given they share the same architectural style. These elements could have the same code but are duplicated in both files. To prevent this, a library for each type of element capable of being reused was created. These libraries are simply CGA files which have a rule that takes at least two parameters: the style and the Level of Detail. The former can be the architectural style in the case of building elements, or can be a type of urban furniture for example. The latter indicates the Level of Detail specified in the current element.

4. ARCHITECTURE

As mentioned before, the proposed pipeline can be implemented in a distributed architecture, where each of the discussed steps can take place in a different computer. Figure 3 illustrates this architecture.

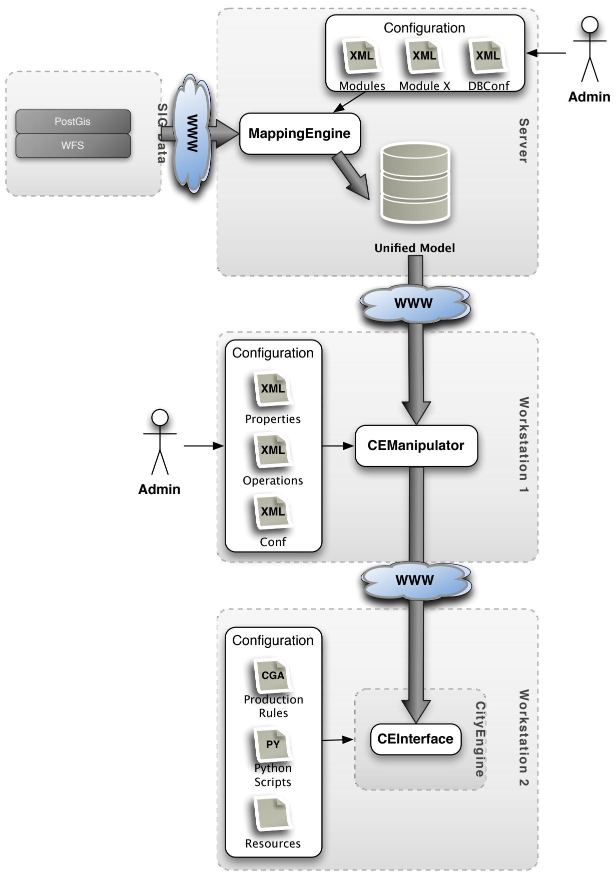

Figure 3: Pipeline Architecture

Here, the information flows through all the components from the municipality's servers where GIS data is stored, to CityEngine where the transformed data is converted into three dimensional models reenacting the city's urban space. The Data Mapping stage is conducted by the application named *MappingEngine*. This is responsible for the sequential execution of the Mapping Modules which can connect to *WFS* and *PostGis* databases. Then, they transform the original data into an Unified Model compatible format. Afterwards, the transformed data is stored in the Unified Model.

To configure this step, the application relies on a set of *XML* files. The file *Modules.xml* contains information about which Modules to use and in which order should they be executed. It also specifies an extra configuration file for each Mapping Module. This is represented by the file *Module X.xml*. These files contain the connection parameters the Module needs to access the original data source, an arbitrary number of additional parameters to configure the specific algorithm, filters to remove unwanted data and information about how to map simple data. The file *DBConf.xml* contains connection parameters to access the Unified Model database.

On a different host machine, the application *CEManipulator* is responsible for the Data Conversion and Amplification step. This application establishes a bridge between the Unified Model, CityEngine and the procedural modeling processes. Here is where LOM Converters are applied to every element retrieved from the Unified Model, effectively amplifying the existing data. It is also possible to generate more attributes, other than those specified in the different Levels of Mapping, by applying the Data Amplification Operators. These are stored in the file *Properties.xml*. To control the procedural modeling processes that take place in CityEngine, this application counts with CityEngine Manipulation Operators defined in the file *Operations.xml*. The file *Conf.xml* contains connection parameters to access both the Unified Model and the plugin *CEInterface*, described next.

To allow the external manipulation of CityEngine, a plugin was developed using the *Python* scripting interface. This plugin implements a server that receives the operations from *CEManipulator* and executes the actions. Because it is a server, it allows the CityEngine to be a tool for the pipeline, thus reducing the need for human interaction. This way, it is possible to remotely execute the procedural modeling processes and retrieve the generated models without physical access to the computer running CityEngine.

5. RESULTS

In this section we present the results obtained using the proposed pipeline to model the urban space of the municipality of Santa Maria da Feira, in Portugal. We also present screenshot images displaying the models generated being used in the Unity 3D game engine in an interactive application.

5.1 Efficiency

It is possible that urban environments may have thousands of elements that one may wish to model. For this reason, if care is not taken, the pipeline's execution may take a lot of time to produce results. As such, it is important to have some kind of metric regarding the time spent by the pipeline to generate urban environments, measuring the individual times for each step.

Table 1 shows the times relative to the Data Mapping stage. This table shows information regarding the mapping of buildings, roads and the whole city. Here, *N^o Read* indicates the number of elements read from the original data source, *N^o Written* indicates the number of elements written to the Unified Model and *Time* represents the time spent on this process. We can see that the number of read items and the written items isn't the same for each type of entities. This is caused by the automatic removal (using filters) of non interesting data.

Table 1: Times relative to the Data Mapping.

Type	N° Read	N° Written	Time
Buildings	1822	1640	116.8s
Roads	282	228	178.58s
Whole City	14631	9899	359.4s

These values depend a lot on the format of the original data and the algorithms involved in processing of such data. From this point on, the rest of the pipeline is not significantly affected by this format.

In Table 2 the times for the Data Conversion and Amplification are shown. As in the previous table, this one presents information for buildings, roads and the entire city. *N^o Read* represents the number of elements read from the Unified Model, *Imported* indicates the number of CityEngine elements (Shapes and Graph Segments) created, and *Time* represents the time spent on this process for each type of entity. The number of roads imported differs from the number of roads retrieved form the Unified Model because of the different representations of this type of data. The Unified Model treats roads as one *Polyline* element, while CityEngine treats them as a set of graph edges and nodes.

Table 2: Times for Data Conversion and Amplification.

Type	N° Read	Imported	Time
Buildings	1640	1640	24.2s
Roads	228	4972	395s
Whole City	9035	10941	1221s

Table 3 contains information about the time spent by CityEngine modeling buildings, roads and the whole city. *N^o Elements* indicates the number of elements to be modeled by CityEngine, *N^o Polygons* indicates the total number of polygons generated and *Time* indicates the time spent in the procedural modeling.

Table 3: Times for procedural modeling.

Type	N° Elements	N° Polygons	Time
Buildings	1640	182574	76.1s
Roads	4972	5352	1.9s
Whole City	10941	4847969	165.4s

Summing the times spent on all three steps for the entire city it is possible to conclude that the whole process takes around thirty minutes to complete for a city of 14631 elements. This is an average of 0.16 seconds per city element.

5.2 Procedural Modeling

The primary goal of this pipeline is to procedurally generate three dimensional models that resemble an existing urban space. As such, it is necessary to analyze these processes regarding the level of detail, visual fidelity and geospatial contextualization.

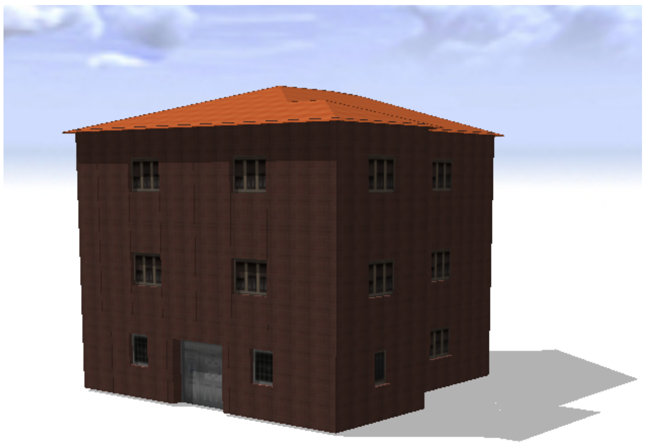

Figure 5: House modeled with CityEngine

To achieve a certain level of visual quality, it is important that procedural generation tools are capable of creating elements with an adequate level of detail. In Figure 5 we present a building modeled with CityEngine. The model in the Figure, represents a house with detailed windows and doors. Also, CityEngine is capable of incorporating previously modeled elements into its models allowing the combination of procedural techniques with traditional ones, increasing the level of detail provided by this tool.

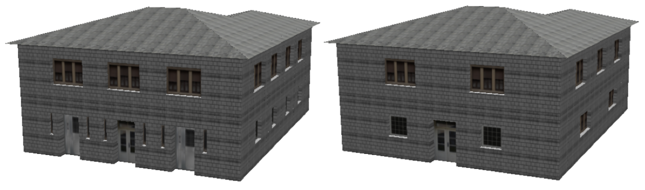

Figure 6: Same house with different openings.

Visual fidelity is a very important factor in the representation of existing urban environments. It also plays a critical role by allowing users to recognize the environments. The correct positioning and spatial relation between elements and their height and shape contributes a lot to establish some relation with the real spaces. In the case of buildings, the production rules created allow some control over the aspect of their facades, by allowing the number of openings (doors and windows) to be specified for each floor in every facade, by changing an attribute. Figure 6 shows the same building with different openings by floor and wall. In this image it is possible to see that, on the left, all facades and floors have three openings, while on the right the same house has two windows on the top floor of the front side, and one door and two windows on the first floor. Moreover, the right one looks more realistic and has a better visual appearance. By allowing such control, it is possible to model certain buildings to look more like their real counterparts.

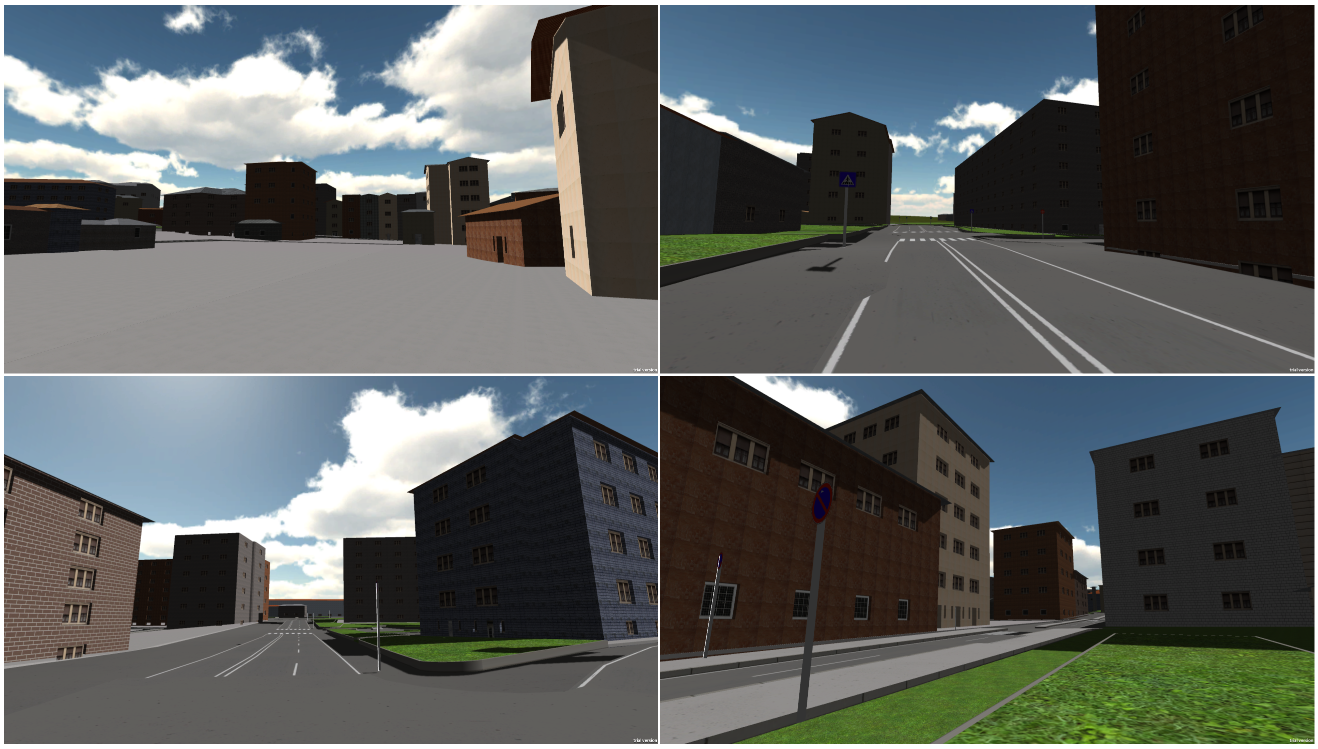

Figure 4: Interactive application in Unity 3D Game Engine.

Geospatial awareness allows to create relations between different elements in an urban space, allowing to infer new properties from the surrounding elements. To this date, however, CityEngine only provides means of verifying if an element intersects another one and to check if the side of a building is oriented towards a street. In the latter case, this spatial awareness is only automatically possible when it is CityEngine itself creating the building lots. In other cases, however, it is possible to use *Python* scripts to define which are the sides nearest to a street. For more complicated cases, where a more complex algorithmic component is needed, it is preferable to use mapping modules to solve geospatial awareness problems.

5.3 Interactive Application

As a proof of concept, we have developed an interactive application using Unity 3D Game Engine, allowing the user to freely navigate the generated urban virtual environment. This Game Engine was chosen for its capability to operate on large terrains. Also, it is quite easy to integrate the generated models and terrain with Unity. However, some terrain areas might require minor modifications, as it is possible that it is not leveled with the remaining models. It's fairly easy to detect these situations as the models may appear partially buried in the ground or floating above it. Figure 4 presents some screenshots obtained in Unity 3D.

6. CONCLUSIONS AND FUTURE WORK

In this paper we propose a three stage pipeline for the procedural modeling of real urban environments, based on information stored in Geographic Information Systems. Such pipeline aims to create a bridge between GIS maintained by municipalities and procedural modeling techniques, harvesting the power of both technologies. On one hand, GIS provide information that can be fed into procedural modeling, allowing to generate virtual urban environments with an higher level of visual fidelity, as can be seen in figures 7 and 8 from different case studies on the cities of Nantes (France) and Porto (Portugal). On the other hand, procedural techniques can accelerate the modeling of such areas, since they require less human interaction while also reducing the costs associated with such a task.

Distinct data models are mapped into an Unified Urban Model according to a discrete number Levels of Mapping. Therefore it is possible to generate virtual environments of distinct municipalities with the same modeling processes, just by adjusting the mapping of semantic information at the first stage of the pipeline. Even if we have data from different data sources classified in distinct LOM, we can amplify these data into an unique resulting LOM on the second stage of this pipeline. Therefore reusability is promoted for distinct GIS data models or the integration of distinct data sources into an unified virtual urban environment.

The Mapping Modules were developed specifically for this case study and, as such, there is a need for more generic and configurable modules. In the same way, the development of LOM Converters for all entities and mapping levels is also required. Due to the large amount of configuration required, the development of a graphical user interface for that purpose would be interesting. Further research on metadata will provide the automation of the mapping process.

Acknowledgements

This work is partially supported by the Portuguese government, through the National Foundation for Science and Technology - FCT (Fundação para a Ciência e a Tecnologia) and the European Union (COMPETE, QREN and FEDER) through the project PTDC/EIA-EIA/108982/2008 entitled "3DWikiU - 3D Wiki for Urban Environments".

7. REFERENCES

[1] D. G. Aliaga, C. A. Vanegas, and B. Benes. Interactive example-based urban layout synthesis. *ACM Trans. Graph.*, 27(5):160, 2008.

[2] G. Chen, G. Esch, P. Wonka, P. Müller, and E. Zhang. Interactive procedural street modeling. In *ACM SIGGRAPH 2008 papers*, SIGGRAPH '08, pages 103:1–103:10, New York, NY, USA, 2008.

[3] A. Coelho, M. Bessa, A. A. Sousa, and F. N. Ferreira. Expeditious modelling of virtual urban environments with geospatial l-systems. *Computer Graphics Forum*, 26(4):769–782, 2007.

[4] A. Lindenmayer. Mathematical models for cellular interaction in development: Parts i and ii. *Journal of Theoretical Biology*, 18, 1968.

[5] M. Lipp, P. Wonka, and M. Wimmer. Interactive visual editing of grammars for procedural architecture. *ACM Trans. Graph.*, 27:102:1–102:10, August 2008.

[6] T. Martins, P. B. Silva, A. Coelho, and A. A. Sousa. An urban ontology to generate collaborative virtual environments for municipal planning and management. In *Proceedings of GRAPP 2012 – 7th International Conference in Computer Graphics Theory and Applications*, pages 507–510, 2012.

[7] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool. Procedural modeling of buildings. *ACM Trans. Graph.*, 25:614–623, July 2006.

[8] R. Měch and P. Prusinkiewicz. Visual models of plants interacting with their environment. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 397–410, New York, NY, USA, 1996. ACM.

[9] Y. I. H. Parish and P. Müller. Procedural modeling of cities. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 301–308, New York, NY, USA, 2001. ACM.

[10] T. Tutenel, R. M. Smelik, R. Lopes, K. J. de Kraker, and R. Bidarra. Generating consistent buildings: A semantic approach for integrating procedural techniques. *IEEE Trans. Comput. Intellig. and AI in Games*, 3(3):274–288, 2011.

[11] P. Wonka, M. Wimmer, F. Sillion, and W. Ribarsky. Instant architecture. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 669–677, New York, NY, USA, 2003. ACM.

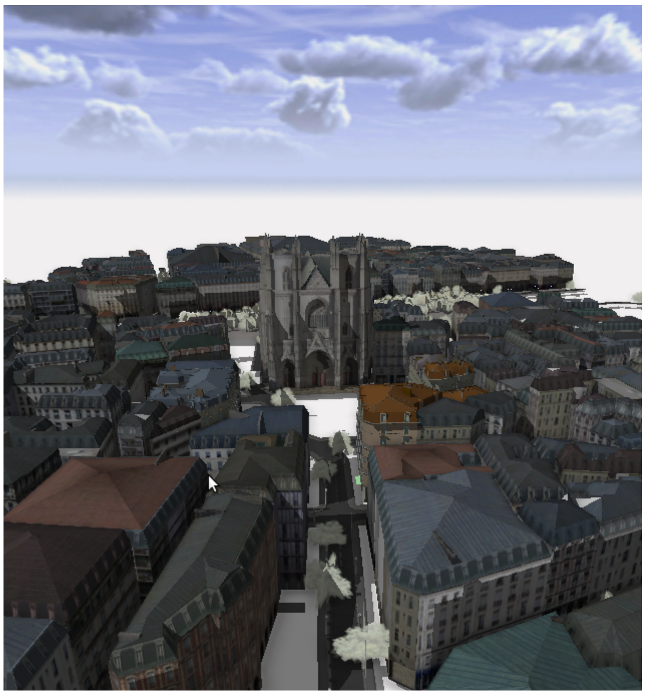
Figure 7: Landscape from the city of Nantes.

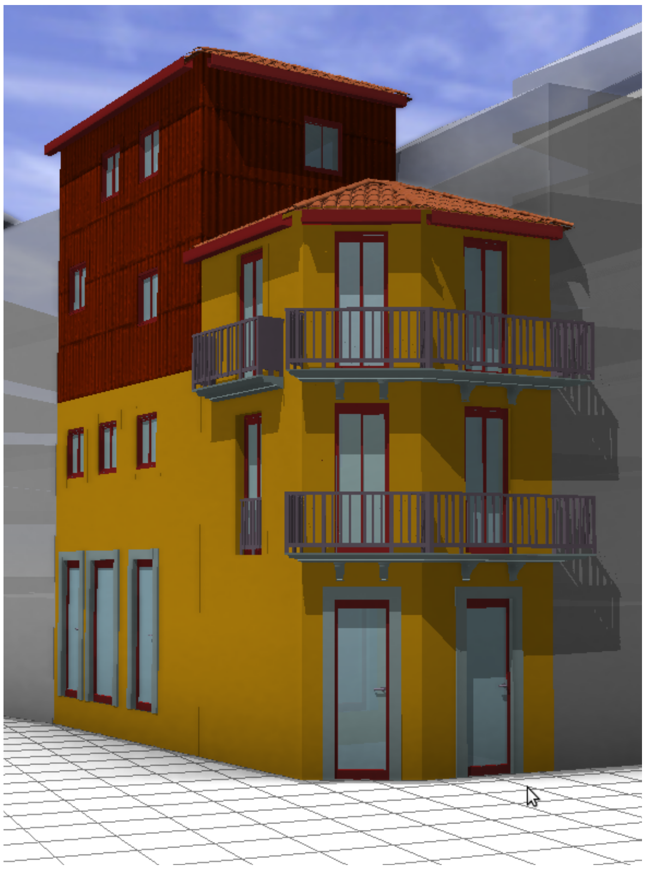
Figure 8: Detailed building from the city of Porto.