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

Expeditious Modelling of Virtual Urban Environments with Geospatial L-systems

António Coelho[1], Maximino Bessa[3], A. Augusto de Sousa[1 2], F. Nunes Ferreira [1]

[1]FEUP – Faculdade de Engenharia da Universidade do Porto, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal
[2]INESC Porto, Campus da FEUP, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal
[3]Universidade de Trás-os-Montes e Alto Douro, Quinta de Prados, 5000-901 Vila Real, Portugal

Abstract

This paper presents Geospatial L-systems, a new extension of L-systems that incorporates geospatial awareness, and shows an application of this new tool in the expeditious modelling of urban environments, integrated with a modelling system with interoperable access to data sources. L-systems have been used in Computer Graphics for the modelling of plants, and in a few experiments to model urban environments. However, the lack of geospatial awareness is a limitation and in spite of some developments like open l-systems introduced the ability to communicate with the environment, there was a need for more flexibility. A new modelling system, named XL3D, generates virtual urban environments automatically from a XL3D document with a modelling specification. This modelling system accesses data sources in a interoperable way and the modelling processes are based on L-systems. The integration of geospatial L-systems with the XL3D modelling system has increased its potential for automation and improved the potential to generate virtual urban environments with a higher level of detail and visual fidelity, with a lower level of complexity of the modelling processes. These facts are shown in a case study where a virtual urban environment taken from an area in the Porto downtown is generated by this solution.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Modeling packages; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism— Virtual Reality; F.4.2 [Mathematical Logic and Formal Languages]: Grammars and Other Rewriting Systems—Parallel rewriting systems (e.g., developmental systems, L-systems)

1 Introduction

A virtual urban environment is an important content to a diverse set of applications, like urban planning, virtual tourism, education or cultural heritage. However, generating 3D models of large urban environments poses a large number of computer graphics problems, as the amount of information needed to create realistic models increases with size and complexity. Nevertheless, some methods have been developed in order to automate this task, either by using image acquisition and interpretation techniques [FLR*95] [DTM96] [PPK*98] or by using production systems [PM01] [WWSR03]. The first set of methods is strict to reality as they rely on the image of the objects to be modelled, but the second one can be used to predict scenarios or to portray urban areas that have long cessed to exist.

The complex nature of urban environments is prone to be studied by a diverse set of professionals; geographers, economists, architects, sociologists, historians, archaeologists, etc. Some initiatives [KGP05] aim to the development of a unified model to represent every kind of urban environment. This information, collected and gathered in the form of documents or databases, can be used to drive the modelling process, but usually it's not enough. To overpass missing data it must be submitted to a process of data amplification, either by providing knowledge about the environment, or including some randomness in parts of the model where information is coarse and a great level of visual fidelity is not essential.

Production systems, namely L-systems [Lin68], have been used in Computer Graphics with large success to model individual plants and plant ecosystems [DHLM98] [LP02] [PMKL01]. The ability of L-systems to model these natural organisms is largely based on data amplification and emergence, which enables them to generate complex objects from simple data sets.

The use of L-systems do create virtual urban environments has been introduced in [PM01] demonstrating that a simple initial dataset could generate complex virtual environments, although with low visual fidelity. The XL3D modeller [CSF04] is a new solution for expeditious modelling of urban environments, based on parametric L-systems, that improves visual fidelity by maximizing interoperability with distinct types of data sources. The XL3D modeller was implemented as a Web service and operates automatically when invoked with a modelling specification. This modelling specification is a

document based on a new XML schema called XL3D [CSF03] that describes the model structure, the data sources, the modelling process, but not the geometry. This fact allows for easy reuse and expansibility of these specifications, and maximizes the maintainability of the virtual environment.

A key issue in modelling urban environments is spatial awareness, that is, the ability to perceive the spatial relations between the distinct urban elements and between them and the environment. As an example, a bus stop must be oriented to the nearest street, and buildings whose walls are very close cannot have windows or balconies.

Some extensions of L-systems have been proposed to cope with these problems, like environmentally-sensitive L-systems [PJM94] and open L-systems [MP96]. Nevertheless, spatial interaction is applied by communication with external applications and not embedded in the mathematical formalism.

In this paper, a new extension of L-systems called geospatial L-systems is presented. This extension is aimed to solve the problem of geospatial interaction between the objects represented by the different modules that compose a parametric string.

Section 2 of this paper presents the related work. Fundamental concepts in the context of L-systems and geospatial data are introduced in section 3 and section 4 defines geospatial L-systems. Section 5 describes an interoperable solution for modelling virtual urban environments based on geospatial L-systems and section 6 shows the potential of this tool to expeditious modelling of virtual urban environments. Finally, in section 5, some conclusions and future developments are presented.

2 Related Work

Generating virtual urban environments has already been addressed in several works. Image data, including photographs and video, have already been used successfully in the reconstruction of buildings [FLR*95] [DTM96] [PPK*98], and although detailed models might be obtained, substantial amount of user intervention is required for extensive scenes.

Range scanning is another technology that can be used for accurate reconstruction of the geometry of buildings in extensive urban environments [WF95] [MV99]. Nevertheless, there is a need to integrate further data, like image [HB99], in additional modelling tasks, to obtain realistic models of buildings.

Geographic information systems (GIS) can provide a large set of comprehensive data regarding urban environments. Automatic modelling of large urban areas has also been demonstrated by works such as [PBGD01] and [SZ03]. Standardization activities in the Open Geospatial Consortium have led to the development of specifications like GML (Geography Markup Language) [GML] or WFS (Web Feature Service) [WFS], that enable the development of interoperable solutions. Based on GML, a new model to describe data for any king of city [KGP05], presents a good potential to improve uniformization of geospatial data of urban environments permitting new approaches to the modelling problem.

One key issue for the automation of the modelling process is data amplification, that is, the ability to generate new data by integrating knowledge and an initial set of data. Grammars, such as L-systems, are one of the technologies that have been used in this context with good results.

L-systems have been used in several computer graphics applications, especially to model natural phenomena and organisms as in [DHLM98], [LP02] and [PMKL01]. Similar works of automatic modelling systems, based on grammars, have also addressed the modelling of urban environments:

In [PM01], extended L-systems are used to derive streets and building allotments, from an initial set of data consisting of image maps, each of these containing data such as population density, elevation or land use. Parametric stochastic L-systems are also used for the generation of building geometry, although, to achieve some detail, another technique, based on functional composition, is used. In [CSF04] parametric stochastic L-systems are used to generate models of all types of features from urban environments with a similar level of detail. Furthermore, the integration of interoperable data sources containing vector graphics data allows the generation of a more realistic model.

The original definition of L-systems does not allow for spatial awareness, but some extensions have been proposed to cope with this issue. Environmentally-sensitive L-systems [PJM94] enable the environment to influence the development sequence of the L-system, but not the opposite. Open L-systems [MP96] enable bidirectional communication between the L-system and the environment, through a so called ”query module”, that instantiates a set of parameters by calling external applications.

A different approach is presented in [WWSR03], a split grammar, a special set grammar based on the concept of shape. This approach is restricted to the automatic modelling of architecture, but eases the processing of coordinates, by integrating the concept of shape in the production rules. The specification of rules is based upon a shape description of buildings, while in [CSF04] the structure of the building model is described in a tree-like form, as in a X3D scene graph.

Issues like modularity and reuse of the modelling processes are vital for the speed and efficiency of an initial solution generation. Project CONTIGRA [DHM02] is clearly an example of how XML coding of X3D makes possible the integration and reuse of components in the process of three-dimensional applications development.

3 Fundamental Concepts of Geospatial L-systems

Geospatial L-systems here presented are an extension of Parametric L-systems [PHMH95]. Each parametric module, known as geospatial module, is associated to a geospatial projection of the object that is being addressed.

This object is characterized by a set of geospatial properties that represent its geometric shape.

Although L-systems are prone to simulate communities of organisms and entities, there is a complete absence of spatial awareness in its formalism (although integration with external modules may allow this) to enable the capture of spatial interrelationships.

Geospatial systems, however, are well suited to the representation and analysis of spatial relationships between entities that populate the same space.

The merging of these two technologies can lead to the emergence of powerful tools, prone to data amplification and capable of geospatial awareness.

3.1 L-systems

L-systems (acronym for Lindenmayer systems) are string rewriting techniques developed by Astrid Lindenmayer in 1968 [Lin68]. They correspond to a technology prone to data amplification [Smi94], generating complex structures from small data sets. Another key feature of L-systems is emergence, a process in which a collection of interacting units acquires qualitatively new properties that cannot be reduced to a simple superposition of individual contributions [Tay92]. These new properties then "emerge" from the initial representation, in the same way as an ant colony looks like a superorganism and not a collection of ants.

According to [PHMH95], one of the basic concepts in the context of L-systems is the module, denoting any discrete constructional unit that is repeated as the system develops. Branches and leaves are an example of this concept, in modelling a tree. A parallel rewriting system replaces individual ancestor modules by configurations of descendent modules on a development sequence.

All modules belong to a finite alphabet of module types, thus the behaviour of an arbitrarily large configuration of module can be specified using a finite set of production rules.

In the simplest case of context-free rewriting, each production replaces a module, called the predecessor, by zero, one or more modules called the successors. In the case of context sensitive L-systems, the applicability of a production does not depend only on the predecessor module, but also on its neighbours.

The productions are applied in parallel, with all modules being rewritten simultaneously at each derivation step. The sequence of strings of modules obtained in consecutive derivation steps, from a predefined initial structure (axiom) is called a developmental sequence.

Parametric L-systems [PHMH95] extend the base concept of L-systems, by appending parameter values to each name. Parametric L-systems operate with parametric words, that is, strings of modules with parameters. In this type of system, the application of a production rule can be dependent of a condition evaluation.

3.2 Geospatial Data

Geospatial data are a subset of spatial data, which is data referencing the location of objects and entities within a given coordinate system. What distinguishes geospatial data is that it is absolutely or relatively positioned on a planet, or georeferenced. That is, it has a terrestrial coordinate system that can be shared by other geospatial data.

Coordinate systems used for georeferencing geospatial data can be classified as astronomical, three-dimensional Cartesian, ellipsoidal and cartographic.

Astronomical coordinate systems are generally used to define the position of the stars, and three-dimensional Cartesian coordinate systems are based on a coordinate system located approximately in the centre of the Earth.

Ellipsoidal coordinate systems are defined based on a surface called an ellipsoid, that represents roughly the surface of the Earth, using two angular measures (latitude and longitude), and one measure of height above the ellipsoid. The irregularity of the Earth's surface requires the use of a well known mathematical surface, whose dimensions and position are specified in an associated geodesic datum.

The planar representation of phenomena located over Earth's surface and referenced by geodesic coordinates requires the use of mathematical functions that change some metric properties of the object. Cartographic coordinate systems do objects projections on a plan using one of these functions, known as cartographic projections.

Although there are a lot of different geospatial coordinate systems, mathematical tools allow the transformation between any two coordinate systems.

Geospatial data are classified into graphic data (or called geometric data) and attributes (or called thematic data).

4 Geospatial L-systems

Geospatial L-systems operate with geospatial words consisting of characters (or symbols) associated to a geospatial object and a set of parameters, so they can be considered as an extension of parametric L-systems. As such, a geospatial module representing a building is associated to a geospatial object representing the building geometry and a set of parameters describing its properties.

4.1 Geospatial String

Data operated on a geospatial L system is represented by a parametric string, known as a geospatial string. In accordance to its geospatial nature, each of these parametric modules is associated to a geospatial object, thus named geospatial module.

Geospatial L systems allow the integration, in the same developmental sequence, of data supplied from different data sources. All geospatial objects associated with modules represented by the same name belong to the same spatial theme (Figure 1).

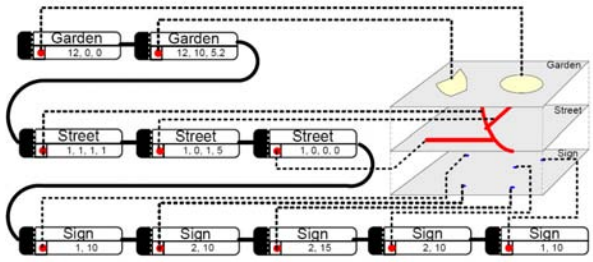

Figure 1 - *Geospatial string*

The geospatial concept is extensible to the control of the development sequence, either in the applicability of the production rules, or in the definition of the multiple successor's sets. The applicability of the production rules can be dependent of geospatial conditions between the geospatial modules that compose the modular string.

4.2 Geospatial Module

A geospatial module is the discrete constructional unit of the modular string that is operated by a geospatial L-system. A geospatial module is thus a parametric module that is associated with a geospatial object mapped over a geospatial reference system. Each of these modules is composed by a name, a geospatial component and a sequence of parameters.

The parameters belong to the set of the real numbers and the geospatial object belongs to a two-dimensional space (a plan), although Z values may be used to encode height.

The name of a geospatial module corresponds to a character string, from a specified alphabet. This identifies unanimously each type of geospatial module in any parametric modular string over the entire developmental sequence of the L-system.

For example, a geospatial module with symbol *A* and parameters a_1, a_2..., a_n that is associated to a geospatial object *G*, can be represented by *A(G,a1,a2...,an)*.

4.3 Geospatial Object

A geospatial object is a representation of an entity or phenomena that is located or occurs on a specific position over the surface of the earth.

The concept of a geospatial object closely follows the OGC Simple Features Specification [SF], since this is a result of important standardization activity, which led to the development of GML 2.1 [GML], and is extended by another set of spatial operators [Shn97]. In order to be more flexible, allowing for the description of height, the coordinate definition of Simple Features is extended to $\Re^3$.

4.4 Production Rules

Production rules are the core component of geospatial L-systems. They replace certain modules by sequences of others, throughout the developmental sequence.

A production rule numbered *m* is represented as follows:

$$m: \quad P : cond \rightarrow S_1 \ldots S_n$$

P is the predecessor module, *cond* is a condition and $S_1 \ldots S_n$ is the string of successor modules.

The applicability of a production rule depends on the verification of the following condition: the name and the number of arguments in the module, must be equal to the name and number of formal parameters of the predecessor of the production rule.

Additionally, if the production rule has an associated condition, this must also be evaluated as true. This condition is a mathematical expression containing arithmetic, logic and spatial operators and predicates.

After evaluating all predecessor modules, the production rules are applied simultaneously to those modules where at least one production rule is applicable. For each application of a production rule, the predecessor module is replaced by the successor modules. The actual parameters of the successor modules are instantiated by evaluating expressions of the formal parameters of the predecessor, or also from spatial operators returning numbers. The geospatial objects of the successor modules are obtained from the predecessor, either by cloning or as a result of spatial operations. A spatial operation is defined as a function of spatial arguments returning spatial objects or real values

4.5 Collections

Geospatial objects can be associated to a single geometry (a point, a line or a polygon) or to a collection of geospatial objects, which in the Simple Features specification is also a Geospatial Object. All modules with the same symbol are mapped on the same geospatial theme and, are also considered a collection.

Operations on collections can be done using two special modules: The query module and the fragmentation module.

The query module inserts a collection of objects from a query to the geospatial database and is represented by the symbol *?* followed by a text string containing a SQL query.

The fragmentation module is represented by the symbol % followed by the name of the new modules. It fragments the geospatial object associated to the geospatial module located at the left of this fragmentation module. The module to the left is replaced by a sequence of modules with the specified symbol, containing each of the segments and the same parameters. On a collection, fragmentation returns each of the individual geometries, but if the object is a single geometry, it returns components of this geospatial object. For instance, if the geospatial object is a polygon, fragmentation returns its boundary, if it's a line string it returns line segments, and if the object is a line it returns its points.

4.6 Developmental Sequence

An initial string of geospatial modules, called the axiom, is operated by a set of production rules, throughout a succession of discrete steps, named derivation steps. A developmental sequence of geospatial L-systems is thus an iterative process that transforms an initial axiom into a final string of modules, over a sequence of derivation steps. At each derivation step, a set of production rules replace modules from the predecessor string by strings of modules forming the successor string. This process is repeated iteratively until no more production rules are applicable or a predefined maximum number of derivation steps is reached.

At each derivation step a sequence of three steps is performed:

i. The geospatial string is searched for query and fragmentation modules. When found a fragmentation module, the module on the left is replaced by a set of modules obtained from fragmentation of its geospatial object. If a query module is found, then the modules resulting from the query are inserted in the module's position;

ii. The geospatial string is searched sequentially again to test each module for the applicability of a production rule;

iii. For each geospatial module where a production rule is applicable, the rule is applied to the module replacing the predecessor module by a sequence of successor modules.

4.7 Stochastic Geospatial L-systems

A great variety of systems, like natural ecosystems or urban environments, denote randomness on its visual appearance. Stochastic geospatial L-systems are a development of geospatial L-systems based on the definition of stochastic L-systems [PL90] that enable more than one production rule to be applied simultaneously to a specific geospatial module. In this case, only one production rule must be randomly selected. In stochastic L-systems for each production rule, a probability value can be associated, thus changing the probability of being selected.

4.8 Context-sensitive Geospatial L-systems

Context sensitive L-systems [PL90] allow modules to be aware of other modules located on its left and right sides of the geospatial string, called left and right context, respectively.

In context-sensitive geospatial L-systems a production rule is represented as follows:

$$m: \quad L < P > R : cond \rightarrow S_1 \ldots S_n$$

Where *m* is the number of the production rule, *P* is the predecessor module, *L* is the left context and *R* is the right context. *Cond* is a condition and $S_1 \ldots S_n$ is the string of successor modules.

In the case of context sensitive geospatial L-systems, the applicability of a production rule requires also that the modules located on the left and right sides of the predecessor module must also match the left (L) and right (R) context of the production rule.

The formal parameters of the geospatial modules composing the context of a production rule, including its geospatial object, can be used on the expressions that instantiate the successors.

4.9 Interpretation

Interpretation of the final string of modules, obtained after the developmental sequence of the geospatial L-system, is a necessary task in order to generate the corresponding scenegraph in X3D format [X3D].

Each parametric module is replaced by a segment of a scenegraph, from which a certain number of parameters are instantiated according to the actual parameters of the geospatial module being interpreted, including the geospatial object. The final string of modules is thus interpreted as a hierarchical structure, shaped as a tree, in which the first module (on the left) is the root, and the last is a leaf. In order to allow the creation of a branching structure, an additional set of symbols (brackets - [and]) delimit each branch.

In the current work, we followed the approach described in [CSF04]. It is based on the interpretation of a hierarchical structure, shaped as a tree, in order to generate a X3D scenegraph by replacing each parametric module by a prototype. Prototypes are like words from a vocabulary, which the user must create (or import) in order to produce meaningful sentences.

5 Expeditious Modelling of Urban Environments

The generation of three-dimensional models of urban environments is often a very complex task. The main reason is the great diversity of man-made edifications and natural organisms that populate these environments in large numbers besides their extensive nature.

In order to achieve better results, the XL3D modelling system [CSF04] was upgraded by the integration of geospatial L systems in the modelling process. The added value of geospatial awareness will lead to the generation of virtual urban environments with a higher level of complexity and visual fidelity.

5.1 The XL3D Modelling System

The XL3D Modelling System [CSF04] is a three-dimensional modelling system based on interoperable access to data in diverse formats and digital support.

Data integration is achieved by the use of XML documents or invoking Web services in a distributed architecture. The modelling processes and data sources are specified in a declarative way, using documents based on an XML-schema called XL3D [CSF03] and the modelling processes are automated by the use of geospatial L-systems.

This approach potentiates the use of large amounts of digital information about urban environment already processed by other professionals that study these environments. This abstract data is then further amplified by the geospatial L-systems based modeller, leading to the emergence of detailed three-dimensional urban environments. The realism obtained by this solution is actually dependent on the quality of the geospatial data available.

5.2 Modelling Urban Environments

Modelling urban environments comprises a diversity of different kinds of entities, and because urban environments diverge according to the different countries and cultures, modelling projects may differ slightly. Some initiatives [KGP05] aim to the development of a unified model to represent any kind of urban environment and all objects that compose it. This model describes, at a higher level, different levels of urban objects like land uses, buildings, transportation, urban furniture, vegetation and water bodies.

6 Results

To test the power of geospatial L-systems for the expeditious modelling of urban environments, a small area in the city of Porto was used as a case study. A rich set of data was collected for this area to enable the attainment of a high level of visual fidelity.

To generate the virtual environment for this case study, a single XL3D document was created containing four different entities (pavements, buildings, vegetation and urban furniture), four distinct modelling processes and a single data source.

Pavements are important features as they are the base of the model, defining land uses of urban spaces. For instance, sidewalks and roads limit the space for people and cars to circulate. Buildings are key urban entities as they are the goal of urban movements and their visual appearance is of great importance for orientation. Vegetation is also relevant and appears as gardens or isolated trees on sidewalks. Finally, urban furniture is composed by a set of equipments of public use, distributed by specific places around urban environments. Examples of these objects are bus stops, advertising panels, phone cabins, among others.

The modelling processes for these four urban entities are explained in the following sections. The geospatial L-systems described are examples that generate simple styles in modelling urban environments. They can be refined to incorporate additional rules and more data to produce different urban styles, for example according to the city area or the buildings age.

6.1 Pavements

For this case study, the available data to model pavements comprised polygons with a code identifying the type of pavement, so the proper modelling process can create polygons, textured with images of the pavement. Listing 1 shows the three production rules of the geospatial L system that models the pavements.

```
1: Pavement(gPolygon,id,level) →
   InstPav1(gPolygon,id,level,
      Max(gPolygon.EnvelopeWidth(),
      gPolygon.EnvelopeHeight()),
      gPolygon.ExteriorRing().NumPoints(),"-1")
2: InstPav1(gPolygon,id,level,length,nPoints,coord)
   : nPoints>0 →
   InstPav1(gPolygon,id,level,length,nPoints-1,
      Str(nPoints-1)+","+coord)
3: InstPav1(gPolygon,id,level,length,nPoints,coord)
   : nPoints=0 →
   InstPavement(gPolygon,id,level,length,coord)
```

Listing 1 - *Modelling of pavements*

The first production rule determines the maximum size of the polygon in order to configure the texture mapping, so that the dimensions of the distinct shapes that compose the pavement become realistic. *Pavement* is the predecessor module and contains a geospatial object (*gPolygon*) of type polygon and two parameters, designating the type of pavement (id) and the level of the pavement (*level*). This last parameter is important to simulate levelling of pavements, that occurs, for instance, in roads and sidewalks (higher level). If a module with the same name and number of parameters appears in the geospatial string (actually modules *Pavement* compose the axiom) the production rule is applied and this module is replaced by its successor, *InstPav1*. This module is associated with the same geospatial object and has 5 parameters, whose values are obtained by the expressions enclosed in parenthesis and separated by commas. The first two parameters are the same as the predecessor, parameter 3 is the result of a function (the maximum) between two values returned by geospatial operators applied to the geospatial object (*EnvelopeWidth()* and *EnvelopeHeight()*) which return the maximum length of the bounding box. The last parameter is a text string.

Rules 2 and 3 create a text string that lists the index of points that form the polygon. The applicability of these two production rules are dependent on conditions. Production rule 2 is applied iteratively until the number of points reaches 0, when production rule 3 is applied.

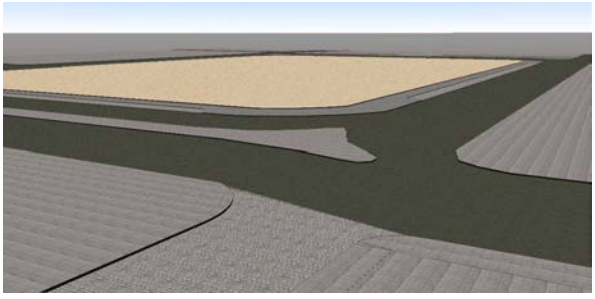

Figure 2 - *Pavements*

The interpretation of the final geospatial string is done by a prototype, instantiated with the polygon points obtained from the geospatial object, the text string indexing

those points, together with the number of the pavement that is used to index the image file to be mapped as a texture. Figure 2 shows the result.

6.2 Buildings

The modelling process of buildings is, no doubt, one of the most complexes. Geospatial data available for modelling buildings is generally restricted to the polygon representing the building projection on the surface of the earth, and some data regarding its height or number of floors. Using this simple axiom (together with roof configuration) and the production rules of listing 2 a prismatic model is obtained representing the volume of the buildings (Figure 3).

```
1: Building(gPolygon,nfloors,roof) : nfloors>0 →
  Building1(gPolygon.ExteriorRing(),nfloors*3,roof)
2: Building1(gLineString,height,roof) →
    [ Block(gLineString,height) ]
    [ Roof(gLineString,height,roof) ]
    [ Gutter(gLineString,height) ]
3: Roof(gLineString,height,roof) : roof=1 →
    InstTerrace(gLineString,height)
4: Roof (gLineString,height,roof) : roof>1 →
    InsRoof1(gLineString,height,
    gLineString.NumPoints(),
    Str(gLineString.NumPoints())+",1,0,-1")
    RoofTop(gPolyline.Centroid())
5: InsRoof1(gLineString,height,npoints,coord)
:npoints>1→
    InsRoof1(gLineString,height,npoints-1,
       Str(npoints-1)+","+Str(npoints)",0,1,"+coord)
6: InsRoof1(gLineString,height,npoints,coord)
    > RoofTop (gPoint) : npoints = 1 →
    InsRoof(gLineString,gPoint.GetX(),gPoint.GetY(),
       height,2.5,coord)
```

Listing 2 - *Modelling of prismatic buildings*

Production rule 1 determines the boundary of the building and approximates the building height by multiplying the number of floor by 3 meters. Production rule 2 composes the structure of the model with a building block, a gutter and a roof. Production rules 3 models a terrace and rules 4, 5 and 6 model a pyramidal roof from the boundary of the building and its centre.

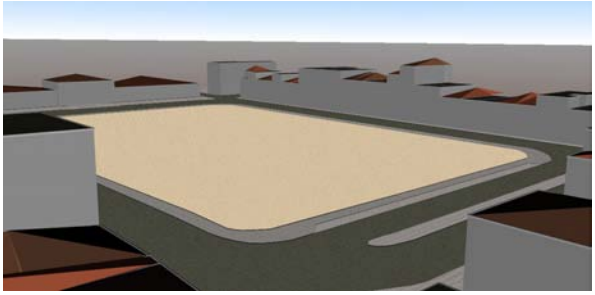

Figure 3 - *Prismatic models of buildings*

With the same axiom but with an increase in the complexity of the geospatial L-system, it is possible to create much more detailed models for buildings, by texturing the building blocks and roofs and instantiating windows, balconies and doors on the façade (Figure 4). In the absence of this kind of data, randomness must be introduced. The buildings walls may be covered by a kind of tile, typical of the buildings age, and the façades must be selected from those that don't have buildings in the nearby. Listing 3 describes four production rules that select the walls to be instantiated by balconies and windows. Production rule 1 extracts the boundary of the building and segments it into distinct walls. This is done by the fragmentation module (*%"Wall"*) which replaces the building boundary by the line segments corresponding to its walls. Production rule 2 determines the exclusivity zone for each wall, by creating a 3 meter buffer around each wall and intersecting it with the rest of the buildings. The query module (?"select * from Building where id=$id") returns, from the geospatial database, a collection (section 4.5) of all buildings except the one under consideration. It works by comparing its identifier with the value if the parameter "id" (the *$* refers to a parameter) of the geospatial module. Production rule 3 selects the walls that don't intersect any other building and production rule 4 eliminates all other walls. Without the use of geospatial L-systems, none of these geospatial operations would be available. So, information about façades would have to be inserted by human intervention or randomly generated, in this last case, with the risk, for example, of façade elements to intersect other buildings.

```
1: Building(gPolygon,id,nfloors,style) →
    Boundary(gPolygon.ExteriorRing(),id,nfloors,
       style) %"Wall"
2: Wall(gLine,id,nfloors,style) →
    CandFacade(gLine,id,nfloors,style)
    Neighbor(gLine.Buffer(3))
3: CandFacade(gLine,id,nfloors,style)
> Neighbor(gPolygon) : !gPolygon.Intersects(
       ?"select * from Building where id<>$id") →
    Facade(gLine, npisos, estilo)
4: CandFacade(gLine, id, npisos, estilo) → ε
```

Listing 3 - *Generating façades*

The next step, following determining the façades, involves instantiating the different elements that compose it. Instantiating elements on the façade is not direct, as unnatural compositions would result. Building architecture is normally based on horizontal coherence (each floor has the same type of element) or vertical coherence (each vertical band has the same element). The first one is predominant in Portuguese architecture of the beginning of the 20th century and the second one from more recent buildings. This modelling process results in the model shown in figure 4. The model is much more detailed and realistic, but randomness lowers the level of visual fidelity, although some randomness, like the state of the windows (open or closed), can give the model a more realistic look, by appearing to be inhabited.

Figure 4 - *Random generation of buildings*

To increase the level of visual fidelity more data about buildings must be introduced in the same modelling process. The type of coating of the building block, the type of roof and, if possible, the type of elements of the façade will produce a model with the same level of detail but much more level of visual fidelity (figure 5).

Figure 5 - *Modelling buildings from rich set of data*

To obtain an ultimate level of realism, either 3D geometry must be available or images of the façades must be used. By using a texture for each building, it was possible to obtain the more realistic model in figure 6 with just one production rule (listing 4) to produce the façade and the rules of listing 2 to generate the building blocks, with textures.

```
1: Facade(gLineString,image,nfloors) →
       InstFacade(gLineString,image,nfloors*3)
```

Listing 4 - *Instantiating textured façades*

Figure 6 - *Buildings with textures of the façade*

6.3 Vegetation

Vegetation is an important element of urban environments, and appears in the form of gardens or as isolated trees on sidewalks. Modelling vegetation can be done by texturing grass or flowers on appropriate areas and instantiating 3D models of trees on their exact position. Furthermore, if data about the species and dimension of the tree exists, the corresponding model can be instantiated and scaled.

This was the circumstance for this case study, where geospatial data was only available for a few special trees. The rest was generated randomly for each area covered with grass. The model obtained is shown in figure 7.

```
1: Tree(gPoint,spec,height,radius) : spec>0 →
    InstTree(gPoint,spec,height,radius)

2: Garden(gPolygon) →
    InstGarden1(gPolygon,
    Max(gPolygon.EnvelopeWidth(),
        gPolygon.EnvlopeHeight()),
        gPolygon.NumPoints(), "-1")

3: InstGarden1(gPolygon,length,npoints,coord) :
points>0 →
    InstGarden1(gPolygon,length,npoints-1,
        Str(npontos-1)+","+coordIndex)

4: InstGarden1(gPolygon,length,npoints,coord) :
npoints==0 →
    InstGarden(gPolygon,length,coord)
    InstTrees1(gPolygon.Difference(?"flowerbed"))

5: InstTrees1(gPolygon) →
 InstPlantTrees(gPolygon,Round(gPolygon.Area()/60))

6: InstPlantTrees(gPolygon,nTree) : nTree>0 →
    InstPlantTrees(gPolygon,nTree-1)
    Tree(gPolygon.PointOnSurface(),
    Round(Random(1,5)),Random(5,15),Random(2,6))
```

Listing 5 - *Modelling vegetation*

Listing 5 shows the set of production rules used to instantiate trees. Production rule 1 places trees for which there is geospatial data available, and production rules 2, 3 and 4 create the areas with grass. The same process is used for bed flowers, although for simplicity it was not visualized in this paper. Production rule 4 initiates the process of tree plantation on areas with grass but excluding bed flowers (the spatial difference). Production rule 5 defines the number of trees to instantiate in each area of grass, by dividing its area by 60 m^2; production rule 6 randomly places each new tree on the selected polygon (method *PointOnSurface()*) and randomly determines the species (5 different species), the height of the tree, between 3 and 15 meters, and its width between 2 and 6 meters.

If no data were available except for the bounding polygon, then geospatial L-systems are able to create a plausible and similar garden automatically. Listing 6 shows the production rules for the automatic generation of gardens.

Figure 7 - *Modelling of vegetation*

Production rule 1 initiates the modelling process by calculating the radius of the reference circle for the garden, and production rule 2 uses this value to determine the three

concentric sectors that define the grass areas. Production rules 3, 4, 5 and 6 create the eight crossways of the garden. Production rule 8 subtracts the crossways to the sectors of gardens, production rule 9 creates flower beds on areas with more than 500m^2 and finally production rule 10 creates notable trees in the centre of the flower beds.

```
1: GardenArea(gPolygon) →
    ConstGarden(gPolygon,SQR(gPolygon.Area()/2*Pi())

2: ConstGarden(gPolygon,radius) →
    GardenCirc(gPolygon.Difference(
       gPolygon.Centroid().Buffer(radius/2)))
    GardenCirc(gPolygon.Centroid().Buffer(radius-8)
        .Difference(
          gPolygon.Centroid().Buffer(radius/2-8)))
    ConstCrossways(gPolygon.Envelope(),
       gPolygon.Centroid().GetX(),
       gPolygon.Centroid().GetY())
    Garden(gPolygon.Centroid().Buffer(radius/4))

3: ConstCrossways(gPolygon,xc,yc) →
    Boundary (gPolygon.ExteriorRing(),xc,yc)
    %"BoundarySeg"

4: BoundarySeg (gLine,xc,yc) →
    PointCrossway(gLine.StartPoint(),xc,yc)
    PointCrossway(gLine.Centroid(),xc,yc)

5: PointCrossway(gPoint,xc,yc) →
    Crossway(Line(gPoint,Point(xc,yc)).Buffer(8))

6: Crossway(gPolygon) > Garden(gPolygon) →
    RadialCrossways(gPolygon)

7: Crossway(gPolygon1) > RadialCrossways(gPolygon2)
→  RadialCrossways(gPolygon1.Union(gPolygon2))

8: GardenCirc (gPolygonG) >
RadialCrossways(gPolygonC)→
    RadialCrossways(gPolygonC)
    Garden(gPolygonG.Difference(gPolygonC))
    FlowerCand(gPolygonG.Difference(gPolygonC))

9: FlowerCand(gPolygon) : gPolygon.Area()>500 →
    Flowerbed(gPolygon.Centroid().Buffer(8),
       Round(Random(1,4)))
       Tree(gPolygon.Centroid(),6,Random(15,20),
            Random(6,8))
```

Listing 6 - *Automatic modelling of gardens*

Figure 8 - *Automatic generated garden*

Figure 8 shows the result of automatic modelling of gardens, that by comparison to figure 7 looks very similar. This is another example how geospatial L-systems can be a significant advantage to the automation of modelling processes.

6.4 Urban furniture

Urban furniture objects are generally on a geospatial database as inventory. Since its orientation is important for some of them, like bus stops or street lights, their orientation must also be recorded. With this set of data available, the modelling process is straightforward, and the production rule listed in listing 7 produces the result visualized in figure 9.

```
1: UrbanFurn(gPoint,code,model,direction) →
    InstUrbanFurn(gPoint,code,model,direction)
```

Listing 7 - *Instantiating urban furniture*

Figure 9 - *Virtual environment with urban furniture*

If orientation of urban furniture is not available, geospatial L-systems are able to determine the orientation towards an associated object, like a road, a wall or a crossway.

This is an important use of geospatial L-systems: determine orientation of objects to make them coherent with the environment. In the case of buildings, for example, this is an important facility that guaranties the correct orientation and placement of façades towards the streets where they belong.

```
2: UrbanFurn(gPoint,code,model) →
    InstUrbanFurn1(gPoint,code,model)
    ?"Centreline" %"RoadSeg"
    EndUrbanFurn()

3: InstUrbanFurn1(gPoint,code,model) <
RoadSeg(gLine)→
    Road(gLine,gPoint.Distance(gLine))

4: InstUrbanFurn1(gPoint,code,model)
Road(gLine1,dist) < RoadSeg(gLine2) →
    Road(gLine2,gPoint.Distance(gLine2))

5: Road(gLine1,d1) < Road(gLine2,d2) : d2>=d1 → ε

6: Road(gLine1,d1) > Road(gLine2,d2) : d2<d1 → ε

7: InstUrbanFurn1(gPoint,code,model) >
    Road(gLine, dist) EndUrbanFurn() →
InstUrbanFurn(gPoint,code,model,gLine.Direction())
```

Listing 8 - *Orienting urban furniture*

Listing 8 expands listing 7 to orientate any object to the road network. Production rule 2 fragments the roads' centreline and production rules 3 and 4 determine the distance of the object to each of these lines. Production rules 5 and 6 select the closest line to the object (by eliminating all others) and finally production rule 7

generates the geospatial module with the correct orientation of the closest road.

7 Conclusions and Future Work

Parametric L-systems have already been used for the modelling of urban environments [PM01] [CSF04] with satisfactory results but the absence of spatial awareness difficults the specification of the modelling process and the attainment of high levels of visual fidelity.

In this paper, we presented a new extension to the above L-systems, named geospatial L-systems, that operate on geospatial modules, providing geospatial awareness, not only at the modular level, but also at the control of the developmental sequence. This tool can be used to model highly detailed urban environments, but unlike other similar tools, if a rich set of data is available, a high level of visual fidelity can be reached.

Two other extensions of L-systems, open L-systems [MP96] and environmental L-systems [PJM94], have been used in the same context. Nevertheless, these two solutions have the drawback that they require the interaction between modules to be done by external software modules, and are not integrated in the definition of the production rules.

In order to justify and evaluate the use of geospatial L-systems a case study was conduced to model part of an urban environment. Compared to [CSF04] these results show a large decrease in the complexity of the modelling processes and clear gains in the level of visual fidelity.

Geospatial L-systems can play a major role in the modelling of virtual environments as shown on this paper, but it can also be of great value to areas like urban planning, biology and ecology were its ability to data amplification and geospatial awareness can be used to preview scenarios. Another important area of application is the game and entertainment industry where the possibility for a user to play a game, his own city would be clearly delightful.

As a future work way, there is a need for moving in authoring tools to create XL3D specifications based on geospatial L-systems. Paradigms like visual tools, modelling by example or sketch-based interfaces should improve productivity of these tools.

CityGML [KJP05] is an application schema recently developed for the representation of urban environments in a effort to reach standardization of this data. The development of modelling processes based on this specification would improve reusability to different urban environments and may lead to a certain standardization of the modelling processes. These developments should be done in conjunction with urban architects, to specify the architectural rules that model the distinct urban styles of an urban environment.

Spatio-temporal databases allow the recording of geospatial entities evolution. Integrated with geospatial L-systems this would allow for the modelling of virtual environments over time.

Acknowledgements

This work is partially supported by POSI, the European Union and FEDER through the project POSI/CHS/48220/2002 entitled “3D4LBMS - Three-dimensional Urban Virtual Environment Modelling for Location Based Mobile Services”.

References

[CSF03] COELHO A., SOUSA A., FERREIRA F.: 3D Modelling of Large Urban Scenes from Diverse Sources of Information. In Proceedings of the 7th ICCC/IFIP International Conference on Electronic Publishing (2003), pp. 278-287.

[CSF04] COELHO, A.; SOUSA, A.; FERREIRA, F.; Expetitious Modelling of Urban Scenes, Revista VIRtual (ISSN: 0873-1837, http://virtual.inesc.pt/) - edição especial "Advances in Computer Graphics in Portugal", 2004.

[DHLM98] Deussen O., Hanrahan P., Lintermann B., Mech R.: Realistic modelling and rendering of plant ecosystems. In Proceedings of SIGGRAPH 98 (1998), pp. 275-286.

[DHM02] Dachselt R., Hinz, M., Meissner, K. 2002. Contigra: an xml-based architecture for component-oriented 3d applications. In Web3D ’02: Proceeding of the seventh international conference on 3D Web technology (2002), ACM Press, pp. 155–163.

[DTM96] DEBEVEC P., TAYLOR C., MALIK J.: Modeling and Rendering Architecture from Photographs: A hybrid geometry-and image-based approach. In In Proceedings of SIGGRAPH’96 (1996), pp. 289-300.

[FLR*95] FAUGERAS O., LAVEAU S., ROBERT L., CSURKA G., ZELLER C.: 3-D Reconstruction of Urban Scenes from Sequences of Images. Rapport de Recherche nº 2572, 1995.

[GML] GML - Geography Markup Language. `http://www.opengis.net/gml/`

[HB99] HAALA N., BRENNER C.: Extraction of buildings and trees in urban environments. In ISPRS Journal of Photogrammetry and Remote Sensing (1999) vol. 54, 2-3, pp. 130–137.

[KGP05] KOLBE T., GRÖGER G., PLÜMER L.: CityGML - Interoperable Access to 3D City Models. In Proceedings of the International Symposium on Geoinformation for Disaster Management (2005).

[Lin68] Lindenmayer A.: Mathematical models for cellular interaction in development, Parts I and II. In Journal of Theoretical Biology, Vol. 18 (1968), pp. 280-315.

[LP02] LANE B., PRUSINKIEWICZ P.: Generating spatial distributions for multilevel models of plant communities. In Proceedings of Graphics Interface (2002), pp. 69–80.

[MP96] Mech R., Prusinkiewicz P.: Visual Models of Plants Interacting with Their Environment. In Proceedings of SIGGRAPH’96 (1996), pp. 397-410.

[MV99] MAAS H., VOSSELMAN G.: Two algorithms for extracting building models from raw laser altimetry

data. In ISPRS Journal of Photogrammetry & Remote Sensing (1999), vol. 54, pp. 153-163.

[PBGD01] PIMENTEL J., BAPTISTA N., GOES L., DIONÍSIO J.: Creation and management of urban 3D scene complexity in immersive virtual environments. In, Proc. of the 10th Portuguese Meeting on Computer Graphics (2001), pp. 165-174. (in Portuguese)

[PHMH95] Prusinkiewicz P., Hammel M., Mech R., Hananl J.: The Artificial Life of Plants. In Artificial life for graphics, animation and virtual reality, volume 7 of SIGGRAPH '95 Course Notes (1995), pages 1--38.

[PJM94] Prusinkiewicz P., James M., Mech R.: Synthetic Topiary. In Proceedings of SIGGRAPH'94 (1994), pp. 351-358.

[PJM94] PRUSINKIEWICZ P., JAMES, M.: MECH, R.: Synthetic Topiary. In ACM Computer Graphics (1994), vol. 28, pp. 351-358.

[PL90] Prusinkiewicz P., Lindenmayer A.: *The Algorithmic Beauty of Plants*. Springer-Verlag, 1990.

[PM01] Parish Y., Muller P.: Procedural Modeling of Cities. In Proceedings of SIGGRAPH 2001 (2001), pp. 301-308.

[PMKL01] Prusinkiewicz P., Muendermann L., Karwowski R., Lane B.: The use of positional information in the modeling of plants. In Proceedings of SIGGRAPH 2001 (2001), pp. 289-300.

[PPK*98] POLLEFEYS M., PROESMANS M., KOCH R., VERGAUWEN M., GOOL L.: Acquisition of Detailed Models for Virtual Reality. In Virtual Reality in Archaeology – CAA'98 Proceedings, British Archaeology Reports; Editores: Barcelo J. et al.; BAR International Séries 843, Barcelona, 1998.

[Sch97] Shneider M.: *Spatial Data Types for Database Systems*. Springer-Verlag, 1997.

[SF] Simple features specification `http://www.opengeospatial.org/specs/`

[Smi94] Smith A.:, Plants, fractals and formal languages. In Proceedings of SIGGRAPH'84 (1994), Vol. 18, Num. 3, pp. 1-10.

[SZ03] SCHILLING A., Zipf A.. Generation of vrml city models for focus based tour animations: integration, modeling and presentation of heterogeneous geo-data sources. In Web3D'03: Proceeding of the eighth international conference on 3D Web technology (2003), ACM Press, pp. 39–ff.

[Tay92] Taylor C.: 'Fleshing out' Artificial Life II, Editors C. Langton, C. Taylor; J. Farmer, S. Rasmussen, Artificial Life II, Addison Wesley (1992), pp. 25-38.

[WF95] WEIDNER U., FORSTNER W.: Towards Automatic Building Extraction from High Resolution Digital Elevation Models. In ISPRS Journal of Photogrammetry & Remote Sensing (1995), vol. 50, pp. 38-49.

[WFS] Web Feature Service. `http://www.opengeospatial.org/specs/?page=specspdf`

[WWSR03] WONKA P., WIMMER M., SILLION F., RIBARSKY W.: Instant architecture. In ACM Trans. Graph (2003), vol. 22, 3, pp. 669–677.

[X3D] Extensible 3d. `http://www.web3d.org/x3d/specifications/x3dspecification.html`