An Incremental Declarative Modelling applied to Urban Layout Design

Sylvain Liège, Gérard Hégron
Ecole des Mines de Nantes
4 rue Alfred Kastler
44070 Nantes cedex 03, France
Sylvain.Liege@emn.fr , Gerard.Hegron@emn.fr

Abstract

In this paper, we present an interactive declarative modelling approach devoted to the design of urban layout. Declarative modelling palliates the weaknesses of imperative techniques. The scene model is described by means of properties without knowing explicitly its geometrical model. From a scene description a set of solutions fitting the properties is obtained. In our approach, the designer describes the urban layout with a description language in an hierarchical and incremental way. The scene description is translated in a constraint graph which is instanciated to compute one solution or a few solutions using a constraint propagation algorithm and a scene generation model. From the scene generation model providing a particular solution, a geometric model (2D map) is extracted. A 3D model of the urban environment can be generated for urban simulations or virtual reality applications.

Key Words : artificial intelligence, declarative modelling, constraint propagation, urban modelling.

1 Introduction

In CAD and image synthesis, the main bottleneck among modelling and rendering processes is the time spent for scene modelling. This is mainly due to the complexity of the scene in terms of number of objects (from ten or so to thousands) and in terms of geometric properties. Traditional technique, called imperative modelling, has shown its limits. For example in mechanics construction some authors [SR88] have clearly pointed these weaknesses :

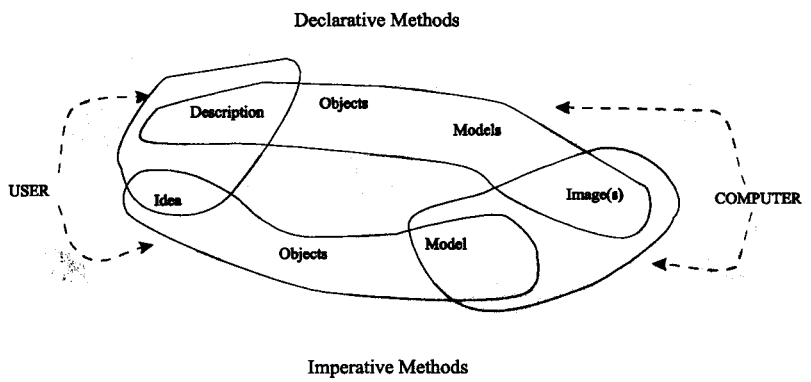

Figure 1 : Declarative modelling and imperative modelling

- There is not enough information on non-geometric aspects of the objects ;
- There is not enough abstraction level. For example, if we want to make a hole in the middle of an object, we need higher geometric information than a collection of faces to do it ;
- A complete knowledge about the object is necessary to modify it.

A few approaches have been proposed to palliate these drawbacks : parametric modelling, constraint-based and rule-based techniques ([LG82]). A new approach called declarative

modelling is meeting a constantly increasing interest in all CAD domains ([GT88], [Luc91], [VtH91], [DH93b]). The user describes the properties of the object and the system tries to compute the corresponding set of geometric models from the object's description (Figure 1). With an imperative approach only one geometric model is created and it is often very difficult to check if the desired properties of the object are verified. Declarative method provides a large number of solutions. Our system can propose one or a few solutions among the whole set. In this paper we present such a declarative method devoted to the design of urban layout.

In the first section our declarative modelling methodology and urban layout characteristics are detailed. Then the scene description model and the calculation of a scene description solution are presented. The constraint propagation algorithm used to compute the solutions is developped in section 5. Before to conclude, a few results illustrate our approach.

2 Declarative modelling and urban layout design

Declarative modelling

Declarative modelling is an attempt to get automatically the geometric models of objects corresponding to a description in terms of properties. The properties can be geometric, topologic, structural and more or less semantic. They depend on the application domain. This approach offers to the designer a more progressive scene specification, leaving the system the care of proposing more than one solution from an object specification, detecting inconsistencies and dealing with incomplete knowledge. Unlike the imperative method, declarative modelling produces more sketches than complete object models. We are applying the declarative methodology to urban layout design and mainly for the development of urban neighbourhoods.

Urban layout design

Figure 2 displays the layout of Canberra city. This layout is mainly a set of streets whose configuration is a mixture of chequered pattern, rayonnant pattern and radio concentric pattern (Figure 3). Streets are the basic geometric elements of the layout. Therefore a set of properties of the layout defines specific configurations of a set of streets. The following set of properties can be distinguished ([LH96]) :

- **unary relations** (intrinsic property of a street) : short, long, broad, narrow, etc.
- **binary relations** (relationship between two streets) : with or without intersection, with or without common extremity, an extremity of a street belongs to another one, parallel, perpendicular, aligned, etc.
- **n-ary relations** (relationship between n streets) : chequered pattern, rayonnant pattern, radio concentric pattern, etc.

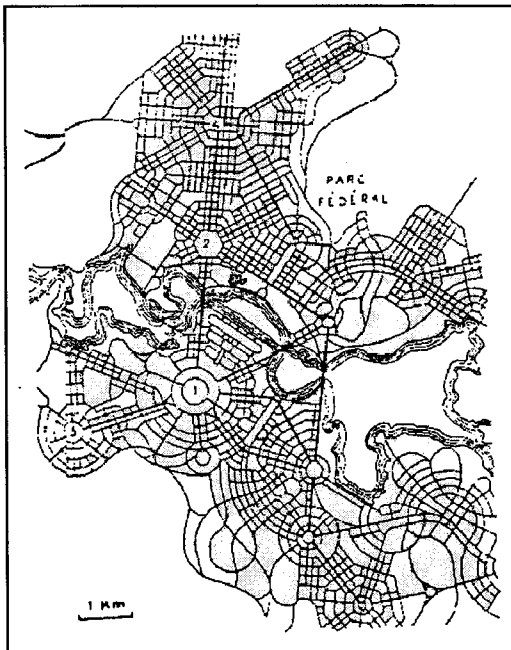

Figure 2 : Canberra city : a mixture of chequered pattern, rayonnant pattern and radio-concentric pattern.

global properties (relationship about the whole set of streets) : percentage of an n-ary relation occuring in urban layout, minimum area of the parcels delimited by the streets, etc.

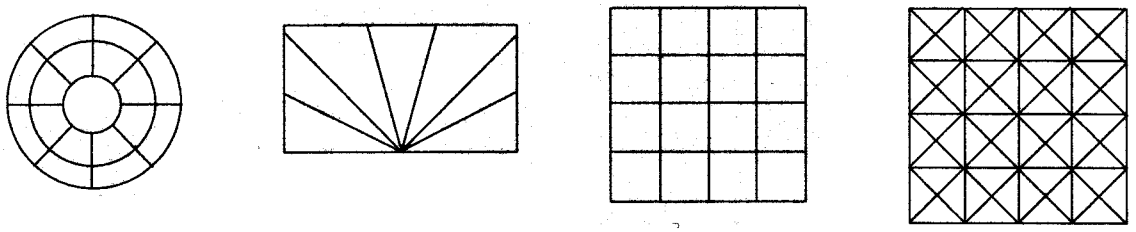

Figure 3 : Urban patterns : radio-concentric, rayonnant, chequered, diamond shaped.

Scene generation process

From a set of urban properties, the designer describes the layout in a hierarchical and incremental way. The user can start from a crude description of the scene involving only the main axes like the grand boulevards and the main characteristic points like squares and crossroads. A solution computed from this first description can be decomposed in subparts, and for each subpart a new subdescription enables the user to refine the layout model. Figure 6 illustrates this hierarchical description approach. For each description level, the designer increases step by step the layout description by introducing a set of streets and relations. From each partial description one solution or a few solutions among numerous potential ones is or are computed. If this intermediate solution does not satisfy the user, the current description can be changed or completed or another solution can be asked for. This process is presented Figure 4. At each stage of the scene generation process the current solution provided by the system can be frozen and kept.

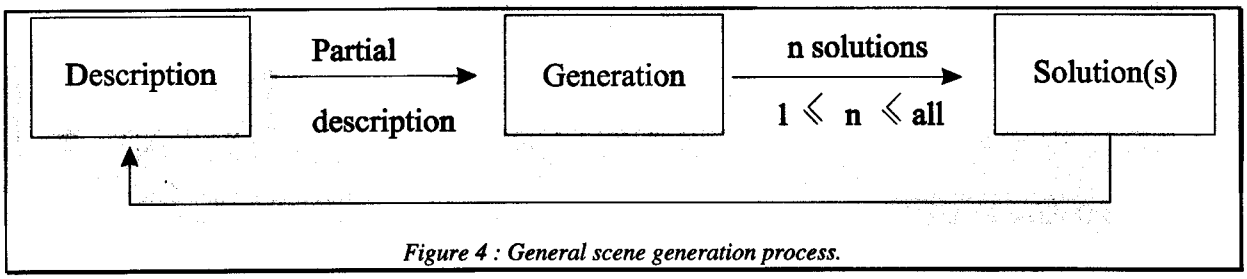

Figure 4 : General scene generation process.

The final geometric model of the urban layout is genarated by the system through the following stages :

1. The designer provides a scene description by means of a description language.
2. The current scene description is translated in a constraint graph which is instanciated to produce a solution using a constraint propagation algorithm and a scene generation model.
3. From the current scene generation model providing a particular solution, a geometric model (2D map) is extracted.

The scene description model

From a description language, the designer describes the urban layout. The geometric primitives are line segments (streets). The properties are unary, binary or n-ary relations, or global properties. A line segment can be specified by an imperative method providing the coordinates of its extremities. The scene description is then represented by a constraint hypergraph whose nodes are primitives (streets) and whose arcs are properties. An example of such a graph is given Figure 5

```
A intersects B
B intersects C
C has a common extremity with A
D does not intersect A
D does not intersect B
D has a common extremity with C
C is large
D is narrow
```

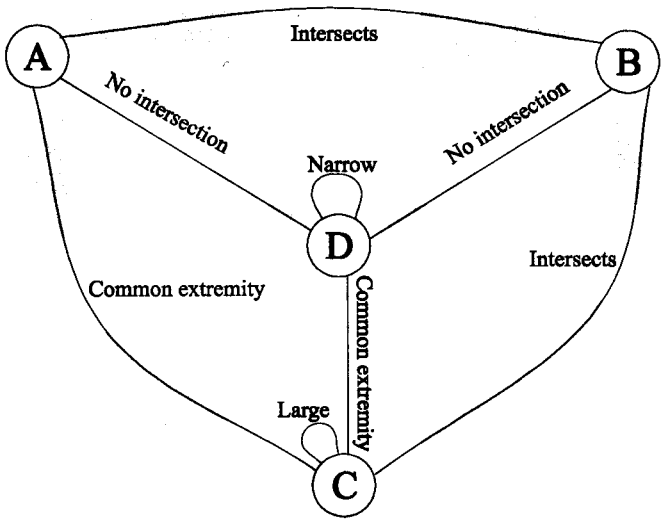

Figure 5 : A description and the associated constraint graph.

3 The calculation of a scene description solution

The scene generation model

The way to compute a solution from the scene description depends on the type of space in which the scene model is generated. If the space is continuous the number of solutions is infinite and no countable. In this case optimization techniques are necessary to get a solution [DH93a]. If the scene is generated in a discrete space, a countable and finite set of solutions is obtained [PD94]. In this case it is more a combinatorial problem.

Our work takes place in the second case. The street configuration (set of line segments) is drawn on a grid of points. The grid can be regular or not, depending on the result we want and the existing environnement. It represents the working space in which we will search a solution of the description. The street extremities are either points of the grid or intersection points between streets, or can belong to another street. If during the scene generation process two line segments intersect, the intersection point is added to the set of points used for line segment extremities.

The calculation of a solution

The calculation of a solution is achieved by instanciating the constraint graph which represents the scene description. This instanciation is performed by a constraint propagation algorithm and by means of the scene generation model on which line segment configurations are generated till they fit the properties. During the constraint propagation algorithm, for each node a set of line segments verifying the properties with the connected nodes is computed and finaly one line segment is kept to compute a final solution. From this configuration of line segments, the set of intersections between line segments is extracted, then a 2D map is generated. The 2D map is useful to compute properties about the layout (shortest path between two points, area of parcels, size of parcels, etc.) and to build a 3D model of the city.

The hierarchical approach

As we mentioned in section 2, the description of the scene can be achieved in a hierarchical way. For each description level of the urban layout, the solution generation process of this subdescription uses the related information included in the different representations of the scene of the previous description level :

- constraint graph : subgraph including the streets (nodes) belonging to the subpart of the layout and the associated relations (arcs) ;
- scene generation model : set of streets, points of the grid and intersection points belonging to the subpart of the scene ;

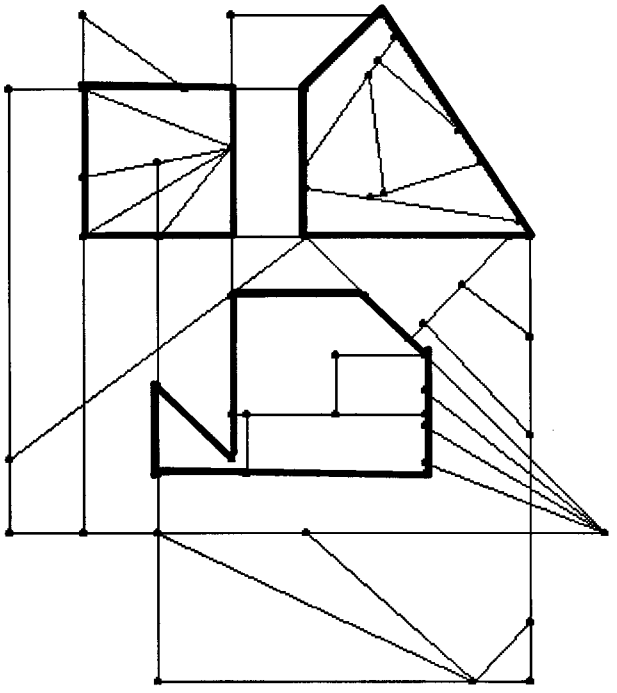

Figure 6 : A map with neighbourhoods.Three of them are in bold.

- 2D map : subgraph containing the streets belonging to the subpart and intersection points between these streets.

4 The constraint propagation algorithm

The number of solutions of a description depends on the resolution of the workspace (number of grid points). We are looking for only one solution or a few solutions, but a lot of intermediate potential solutions are computed.

A node is randomly chosen in the graph. The node is instantiated, that is to say a line segment whose extremities belonging to the grid points and having its unary properties is set up. Then a constraint propagation starts from this node. The propagation goes through the connected nodes. The system tries to instanciate each node. The algorithm randomly chooses a set of line segments whose extremities belong to the grid or to the set of intersection points. A few line segments are kept if they verify the intrinsic properties given at this node, those of the relation with the previous instanciated node and finaly the global properties of the layout. The propagation stops when all the nodes are instanciated.

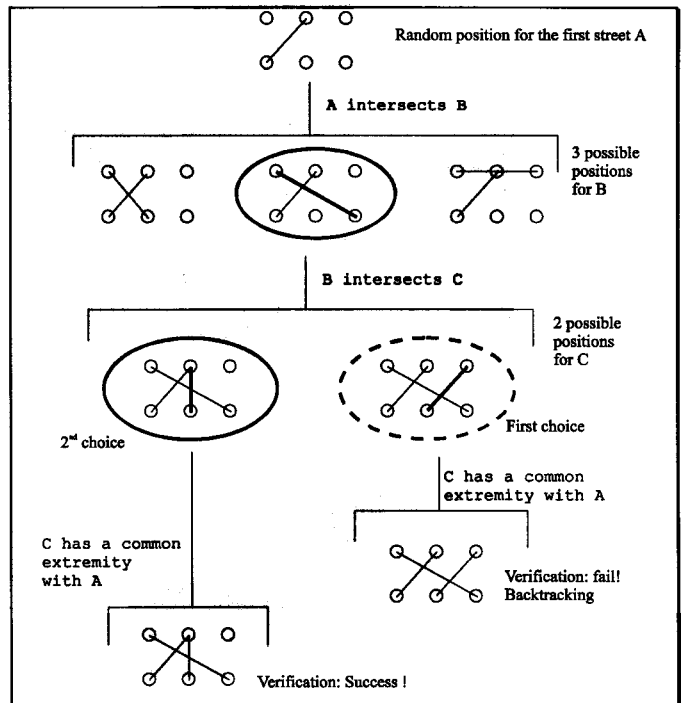

Figure 7 : Example of constraint propagation algorithm in a 2 × 3 grid.

During the constraint propagation two problems have to be solved :

- A node is already instanciated : that happens when there is a loop in the graph. In this case, we check if the connected nodes verify the new constraints of the relation. If they do not, a backtracking is performed to explore new possibilities ;
- A node instanciation fails : that is to say no new line segment can verify its corresponding properties (unary, binary, n-ary and global). Then a backtracking is also performed : each node knows all its possible instantiations on the grid. If an instanciation achieved during the previous graph traversal leads to a failure, a new one is picked up in the set of possible instanciations of the node, and a new constraint propagation starts from this node.

The main algorithm of constraint propagation is given Figure 8. It is called PlaceSegment because its work consists in calling the rule which computes possible position for the current segment (current node in the graph), then assigning one of these possible positions to the current segment and finally propagating the information to the connected nodes in the graph. It performs backtracking if needed.

In this algorithm, we do not use a classical generation tree ([Tsa93]) from which all the solutions are computed and in which we verify a posteriori if the line segment configurations fit the properties or not. For instance, in a 3×2 grid, with a classical generation tree, at the most $15 \times 14 \times 13 = 780$ configurations containing three line segments can be computed. In this case the algorithm provides all the solutions. Our system computes only a few solutions and is guided by the properties. At each leaf of our generation tree, a new constraint is solved. Figure 7 illustrates the constraint propagation algorithm computing in a 2×3 grid a solution of the following description : *A intersects B ; B intersects C ; C has a common extremity with A* . In this example, only 8 configurations are generated to find a good one. This number depends on the random choice of the new line segments at each step.

```
PlaceSegment (SegmentToPlace, ReferenceSegment,Relation)
/* Place SegmentToPlace respecting the Relation between
SegmentToPlace and ReferenceSegment. ReferenceSegment is already
placed */
BEGIN
    IF ReferenceSegment=nil THEN /* 1st Segment*/
        RandomPlacing(SegmentToPlace)
        FOR ALL Segment in relation with SegmentToPlace DO
            PlaceSegment(NewSegment, SegmentToPlace,NewRelation)
        ENDFOR
    ELSE
        CASE relation OF
            PossiblePositions ←ApplyRelation(SegmentToPlace,Relation)
            IF there is a global property to verify THEN
                remove from the set the solutions which do not verify this
                        property
            ENDIF
        ENDCASE
        needToContinue ← True
        WHILE possible position exists and needToContinue DO
            AssignPosition(SegmentToPlace, a position in the set)
            result ← succes
            WHILE result=success DO
                result ← PlaceSegment(NewSegment,
                            SegmentToPlace,NewRelation)
            ENDWHILE
            IF listOfResults contains a fail THEN
                needToContinue ← True
            ELSE
                needToContinue ← False
            ENDIF
        ENDWHILE
        IF needToContinue = true THEN
            /* All the Segments did not succeed in looking for a place */
            return(fail)
        ELSE
            return(success)
        ENDIF
    ENDIF
END
```

Figure 8 : Main constraint propagation algorithm.

few points computed along the segments already placed. By this way, we allow the future segments to start from these new points out of the regular original grid. An example of a segment computed with this method is given Figure 9. E is located between B and D. The extremities of the segment E belong to B and D but not to the regular grid.

This method allows us to guide the research for some properties

Improving the realism of the solutions

As the line segments are generated in a discrete space, the system does not produce satisfactory solutions in two cases :

1. Some properties as « a street *Starts In another one* » and « a street is *Between* two other streets » cannot most often beverified because line segments seldom pass through the points of the grid.
2. The number of dead-ends (street extremities without connection with another street) would be too important to get a realistic solution.

So, we have developped a method to solve both problems. Each time the system computes the possible positions for a segment, a part of these positions is computed in the continuous space : we add to the set of the grid points where the segment extremities can take place, a

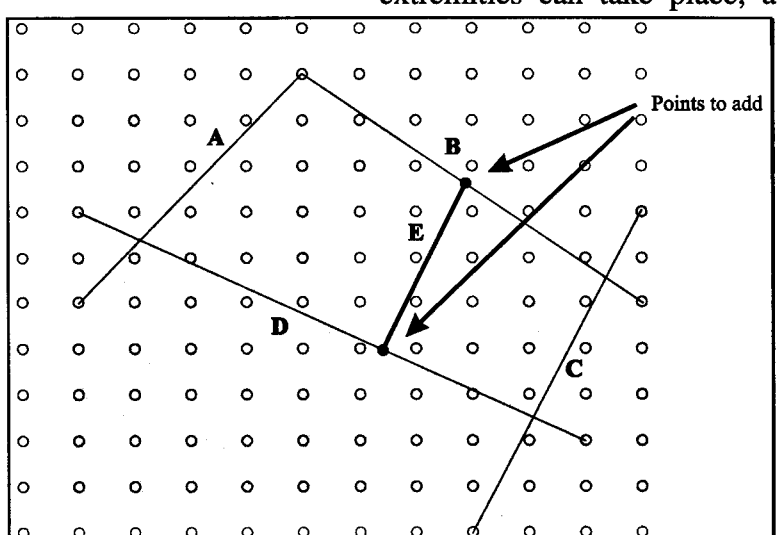

Figure 9 : Instanciation of a segment placed between two other segments.

very difficult to be verified in a discrete space and to improve the realism of the solutions by reducing the number of dead-ends.

5 System Implementation and Results

The system is written in Visualworks 2.0 (Smalltalk). The user interface is mainly composed of a description zone and a display zone as shown Figure 10. The description is made using an interactive interface with buttons and other common widgets. The user gets an echo of his description in a text window. All the properties are available by this way. Step by step, the user enriches the description and the system computes a new current solution. If a solution satisfies the user, he can *freeze* and go on with the description. So, the part of the constraint graph which is frozen will not be altered during the computation of the next solutions. If the solution does not suit the designer, another one can be asked for. The system can also provide more than one solution and display them. The current description can be saved in a file with its solution and restored in order to carry on the design process.

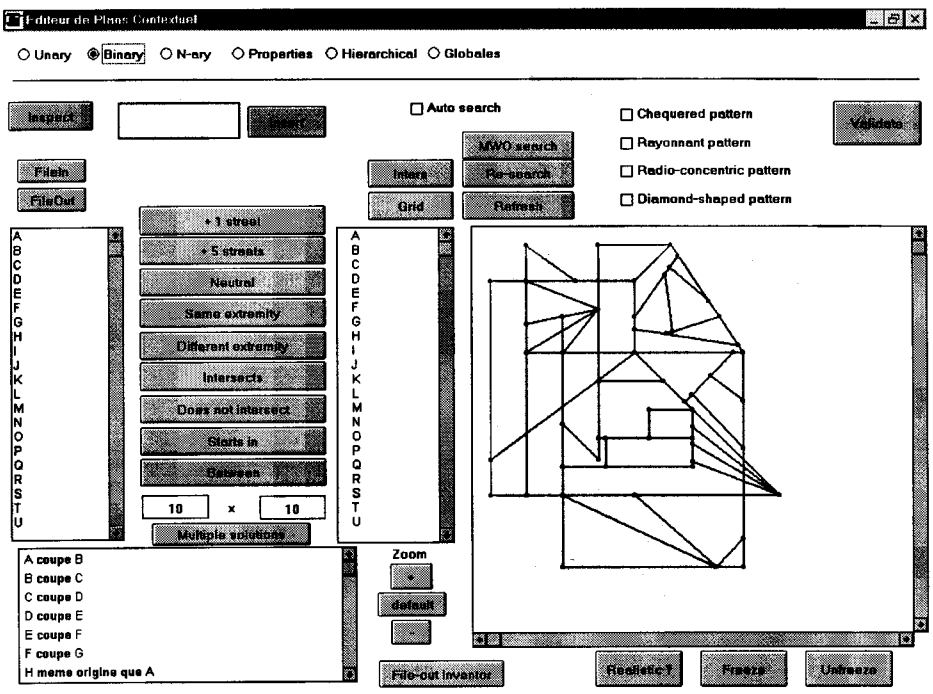

Figure 10 : A view of the main window of our application.

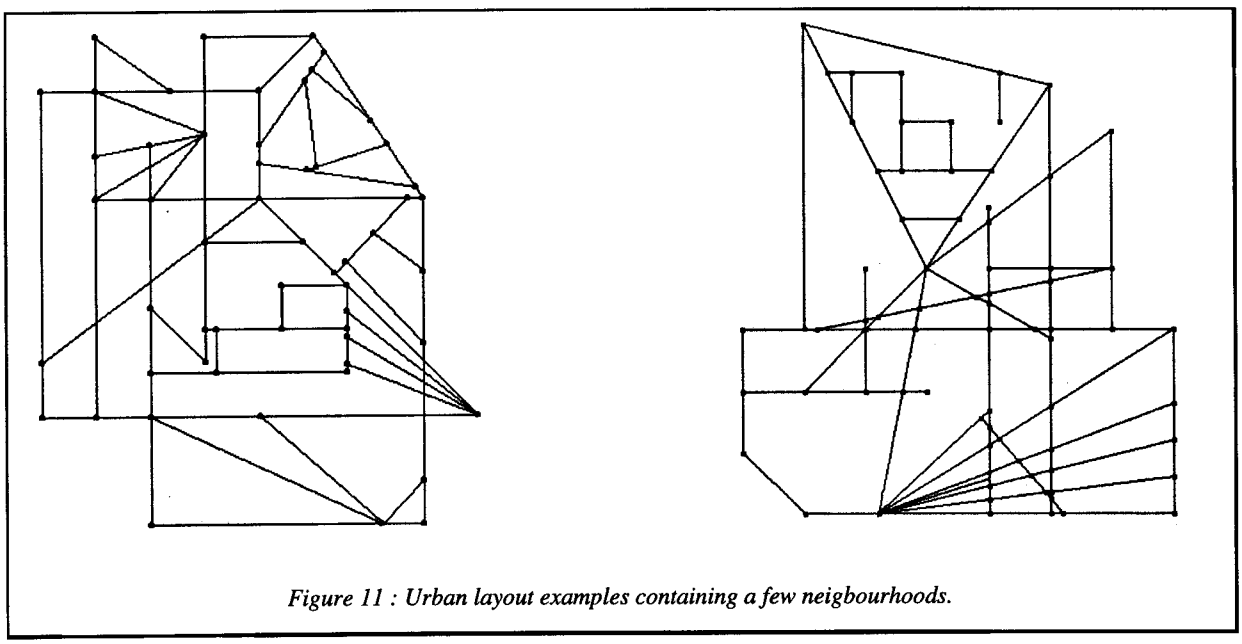

Figure 11 : Urban layout examples containing a few neigbourhoods.

Basically, the system is devoted to the production of 2D models. The creation of 3D models is then the next step for the design and the simulation of comprehensive virtual urban environment. This problem is out of the scope of this paper but Figure 12 presents a 3D example of the urban layout of the right map of Figure 11. This 3D model respects the Inventor syntax and a real time walk through the city can be performed. For virtual reality applications, this declarative method would be very efficient to create very quickly many different cities with the desired properties.

6 Conclusion

In this paper a new method for declarative modelling applied to urban layout design has been presented. The system provides a high level and friendly man-machine interaction. The description of the layout in terms of properties is done step by step and solutions are proposed in an incremental and hierarchical way. An original constraint propagation algorithm has been presented which enables the user to get a solution or a few solutions among a large number potential ones. To compute these solutions in a reasonable running time, we have implemented some powerful prune methods wich guide the research and give more realistic solutions in the same time.

The 3D extensions let the user to have a walk in the city with a VRML browser. This is a first step to a virtual reality modelling system. We can create a sketch of a city in a few minutes without thinking in terms of geometric coordinates.

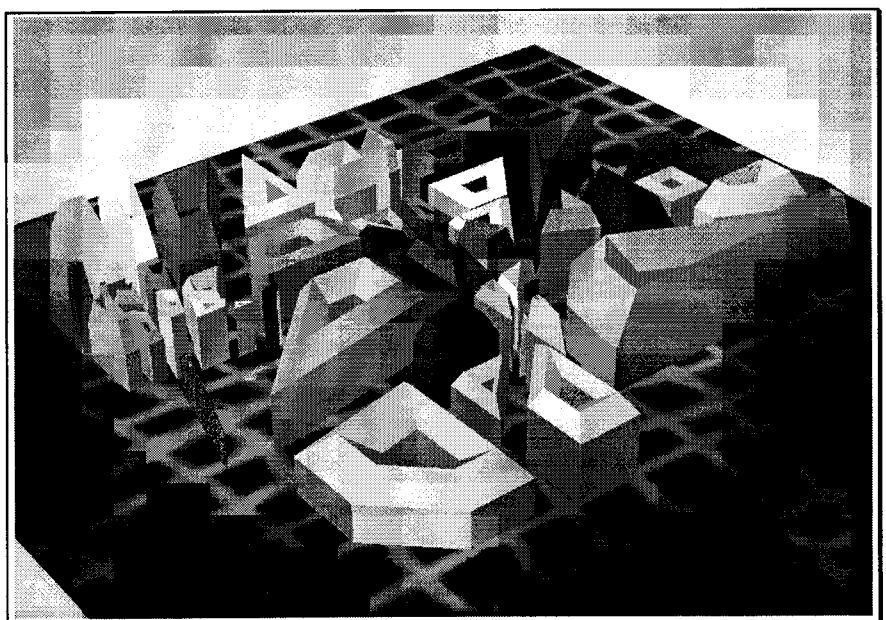

Figure 12 : 3D scene corresponding to a 2D map of Figure 11.

Right now a few properties have been implemented. New properties have to be introduced to make the solutions more and more realistic. In order to detect if there are contradictions in the scene description, we also have carried out the verification of logical inconsistencies, but the presentation of this work is out of the scope of this paper.

This declarative approach can be helpful not only for urban layout modelling but also for every kind of layout based on line segment configurations like the 2D design of gardens. The only change would be the introduction of a new set of properties depending on the specifity of the application.

A The description language grammar

Here is a light version of the language grammar of our application.

Description ::= descriptionName spaceSize ruleList .
descriptionName ::= caracter | word | string
spaceSize ::= ***Space*** *(*integer, integer*) ;*
ruleList ::= rule ruleFolowing
rule ::= property | absolutePosition
property ::= unaryProperty | binaryProperty | naryProperty | globalProperty
ruleFolowing ::= nil | *;* ruleList
absolutePosition ::= streetName *:* point , point
point ::= number @ number
unaryProperty ::= ***narrow****(*streetName*)* | ***normal****(*streetName*)* | ***large****(*streetName*)* | ***veryLarge****(*streetName*)*
binaryProperty ::= ***intersects****(*streetName, streetName*)* | ***doesNotIntersect****(*streetName, streetName*)* | ***sameExtremity****(*streetName, streetName*)* | ***differentExtremity****(*streetName, streetName*)* | ***startsIn****(*streetName, streetName*)* | ***doesNotSartIn****(*streetName, streetName*)*
naryProperty ::= ***Rayonnant****(*streetList*)* | ***Chequered****(*streetList*)*
globalProperty ::= ***Rayonnant****(*streetList*)* | ***Chequered****(*streetList*)*

References

[DH93a] Stéphane Donikian and Gérard Hégron. Constraint Management in a Declarative Design Method for 3D Scene Sketch Modeling. In *First Workshop on principles and practice of constraints programming*, April 1993.

[DH93b] Stéphane Donikian and Gérard Hégron. A declarative design method for 3d scene sketch modeling. In *Eurographics 93*, Barcelona, Spain, September 1993.

[GT88] F. Giunchiglia and E. Trucco. Object by incremental ill-described spatial constraints. Technical report 400, Departement of Artificial Intelligence, University of Edinburgh, 1988.

[LG82] R. Light and D. Gossard. Modification of geometric models through variational geometry. *Computer Aid Design*, 14(4) :209-214, July 1982.

[LH96] S. Liège and G. Hégron. An Interactive Declarative Modelling for Urban Layout Design. *Internal report 96-2-INFO*. Ecole des Mines de Nantes. February 1996.

[Luc91] M. Lucas. Equivalence Class in Object Shape Modelling. *IFIP TC5/WG 5.10 Working Conference on Modelling in Computer Graphics*. Tokyo, 1991.

[PD94] Laurence Pajot-Duval. *Modélisation déclarative de configurations de segments de droites : le projet FiloFormes*. PhD thesis, Ecole centrale de Nantes, June 1994.

[SR88] J.J. Shah and M.T. Rogers. Expert form feature modelling shell. *Computer Aid Design*, 20(9), November 1988.

[Tsa93] Edward Tsang. Fondations of Constraint Satisfaction. Academic press. 1993.

[VtH91] P. Veerkamp and P.J.W. ten Hagen. Qualitative reasoning about design objects. In *5th International Conference on the Nanufacturing Science and Technology of the Future*, Enschende, Pays-Bas, 1991.