Machine Learning for a General Purpose Declarative Scene Modeller

Dimitri PLEMENOS*, Georges MIAOULIS**, Nikos VASSILAS**
*University of Limoges, MSI laboratory, 83, rue d'Isle, Limoges (France)
plemenos@unilim.fr
** Technical Educational Institute of Athens, Athens (Greece)
{gmiaoul, nvas}@teiath.gr

Abstract

In this paper we discuss about the implementation of machine learning mechanisms in declarative scene modelling. After a study of the different kinds of declarative modellers and the different cases where machine learning seems useful, we describe two implemented techniques allowing machine learning for declarative modelling by hierarchical decomposition. The first technique is based on neural networks and allows reduction of the solution space in order to generate only solutions corresponding to the user's wishes. The second one uses a genetic algorithm which, starting from a set of scenes produced by the generation engine of the declarative modeller, produces other solutions under the user's control, taking hence the place of the generation engine. The obtained results are then explained and discussed.

Keywords: *Machine learning, declarative scene modelling, neural networks, genetic algorithms.*

1. INTRODUCTION

Declarative scene modelling is a very useful modelling technique which allows the user to create scenes by simply describing their wished properties and not the manner to construct them. Scene creation is performed by the modeller, using the properties given in the user's description.
As a scene is described in a simple intuitive manner by the user of a declarative modeller, this description is often imprecise. This lack of precision can be due to two different reasons: (a) the user does not know exact properties (exact position, height, width, etc.) of the scene or of parts of the scene to be designed and uses the possibility offered by declarative modellers to make imprecise descriptions; (b) the user believes he (she) is giving exact description by using some properties proposed by the modeller, as "object A is put on the left of object B" although these properties are not precise and admit, generally, more than one solutions.
A consequence of that can be that the scene (or some scenes) proposed as solution by the modeller does not satisfy the user's desires. Several scenes or parts of scenes can be generated by the modeller, because they satisfy the user's description, before obtaining a scene matching with the user's idea about it. This problem will continue with subsequent generated scenes because the modeller does not know better the user's ideas concerning the scene's properties. The modeller does not learn during the scene generation process and it can be frustrating for the user to often obtain non wished scenes. The purpose of this paper is to study satisfactory solutions to this problem, that is to the problem of machine learning from the user's actions.
In section 2, general principles of declarative modelling will be presented together with a special declarative modelling technique, namely declarative modelling by hierarchical decomposition (DMHD). A DMHD-based modeller's prototype, MultiFormes, will be presented too. In section 3 a distinction is established between two kinds of declarative modellers, the dedicated and the general purpose ones. Section 4 presents neural networks based machine learning. A dynamically generated neural network, allowing to filter non satisfactory solutions is proposed in section 5. In section 6, another machine learning approach, based on genetic algorithms, is proposed. A brief discussion on the efficiency of the proposed machine learning techniques is presented in section 7 which concludes the paper.

2. DECLARATIVE MODELLING

Declarative modelling [2, 5, 10, 11] in computer graphics is a very powerful technique allowing to describe the scene to be designed in an intuitive manner, by only giving some expected properties of the scene and letting the modeller find solutions, if any, verifying these proporties.
As the user may describe a scene in an intuitive manner, using common expressions, the described properties are nearly always imprecise. For example, the user can tell the modeller that "the scene A must be put on the left of scene B". There exist several possibilities to put a scene on the left of another one. Another kind of imprecision is due to the fact that the designer does not know the exact property his (her) scene has to satisfy and expects some proposals from the modeller. So, the user can indicate that "the house A must be near the house B" without giving any other precision. Due to this lack of precision, declarative modelling is generally a time consuming scene modelling technique.
It is generally admitted that the declarative modelling process is made of three phases: the *description* phase, where the designer describes the scene, the *scene generation* phase, where the modeller generates one or more scenes verifying the description, and the *scene understanding* phase, where the designer, or the modeller, tries to understand a generated scene in order to decide whether the proposed solution is a satisfactory one, or not.

Declarative modelling by hierarchical decomposition (DMHD) [5, 8, 9] is a declarative modelling technique, based on top-down designing of scenes. Its principle can be described in a recursive manner:

1. If the scene can be easily described with the properties implemented in the modeller, the designer gives a description of the scene and the designing process is finished.

2. Otherwise, the part of description possible at this level is made, the scene is decomposed in a set of sub-scenes and each sub-scene is described using the DMHD principle.

The main advantages of DMHD are: top-down designing; description made locally for each part of a scene without having to take into account its other parts; factorisation of properties; generation in various levels of detail.

MultiFormes is a declarative modeller's prototype working according to the DMHD principle. The tree of the hierarchical description of a scene, used in the generation phase, allows scene generation in various levels of detail and reduction of the generation's cost. To do this, the modeller uses a bounding box for each node of the tree. This bounding box is the bounding box of the sub-scene represented by the sub-tree whose the current node is the root. All bounding boxes of the children nodes of a node are physically included in the bounding box of the parent node. This property permits to detect very soon branches of the generation tree which cannot be solutions. In figure 1, the spatial relation between the bounding boxes of a scene and its sub-scenes is shown.

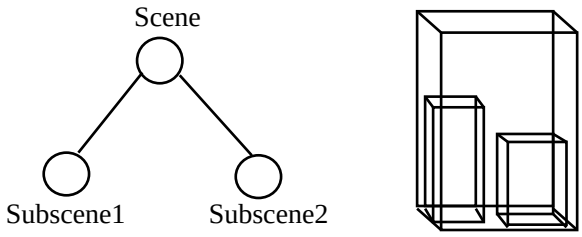


Figure 1: The bounding boxes of the sub-scenes of a scene are inside the bounding box of the parent scene.

Four kinds of properties are mainly used by MultiFormes for its descriptions:

- inter-dimensions **size** properties like "scene A is higher than wide",
- inter-scenes **size** properties like "scene A is higher than scene B",
- inter-scenes **position** properties like "scene A is put on the left of scene B",
- **form** properties like "the top of scene A is almost rounded".

There exist several improvements of the MultiFormes declarative modeller [5, 8, 11, 12]. Latest improvements are based on arithmetic and/or geometric constraint satisfaction techniques. In figure 2, one can see some scenes generated by the MultiFormes declarative modeller.

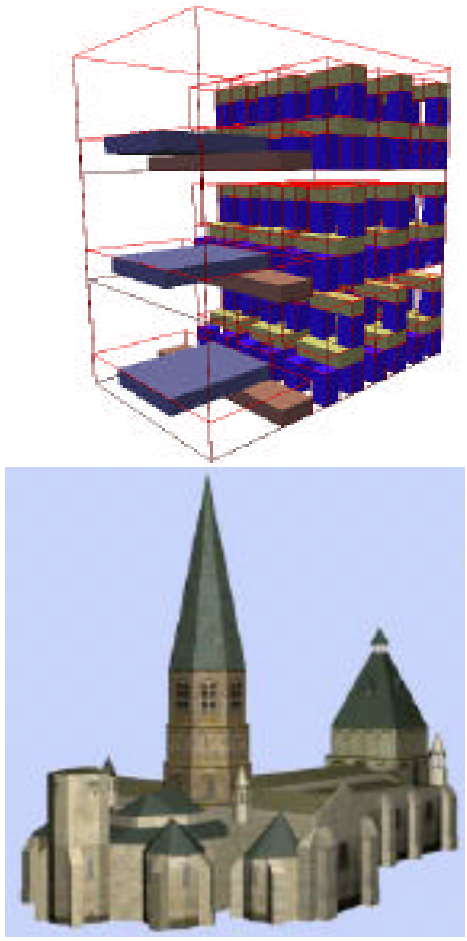


Figure 2: Two scenes generated by the last versions of the MultiFormes declarative modeller. On the left: inside a 3 floors building. On the right: Cathedral of Le Dorat (France), designed by W. Ruchaud.

3. DEDICATED AND GENERAL PURPOSE DECLARATIVE MODELLERS

There exist two kinds of geometric modellers, general purpose modellers, allowing to design almost everything, and specialised (or dedicated) modellers, offering high level modelling for limited specific modelling areas. In the same manner, there exist two families of declarative modellers: general purpose modellers, covering a large set of possible applications, and dedicated modellers, covering a specific area (architectural design, mechanical design, …).
The principle of dedicated modelling is to define a declarative modeller each time it is necessary for a well delimited modelling area (see figure 3). Thus, PolyFormes [6] is a declarative modeller designed to generate regular or semi-regular polyhedra.

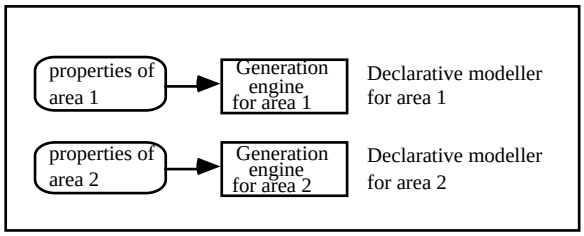


Figure 3 : Dedicated declarative modellers

The main advantage of the dedicated declarative modellers is efficiency because their solution generation engine can be well adapted to the properties of the specific modelling area covered by the modeller. On the other hand, it is difficult for such a modeller to evolve in order to be able to process another specific modelling area.

The aim of the general purpose modellers is generality. These modellers include a solution generation engine which can process several kinds of properties, together with a reduced set of pre-defined properties, as general as possible. General purpose declarative modellers could normally be specialised in a specific modelling area by adding to them new properties, corresponding to the specific modelling area we want to cover (see figure 4). In this sense, general purpose modellers can be seen as platforms to generate dedicated declarative modellers.

The main advantage of general purpose declarative modellers is generality which allows to specialise a modeller in a specific modelling area without having to modify its solution generation engine. On the other hand, general purpose modellers suffer from their lack of efficiency, because of the generality of the solution generation mechanism.

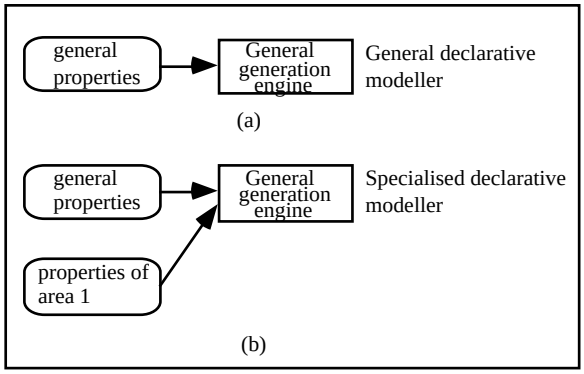


Figure 4 : general purpose declarative modeller (a) and its specialisation (b)

The declarative modeller's prototype MultiFormes is a general purpose declarative modeller.

4. MACHINE LEARNING AND GENERATION MODES

A declarative modeller can be used in two different modes: *exploration* mode and *solution search* mode.

In exploration mode, the declarative modeller, starting from a user's description, performs a full exploration of the solution space and gives the user all found solutions. This mode can be used when the designer has insufficient knowledge of a domain and wants to discover it by an exhaustive exploration or when the designer is looking for new ideas and hopes that the modeller could help him (her) by exploring a vague description. The use of imprecise properties increases the richness of the solution space and allows the user to obtain concrete answers for a vague mental image. So, the use of imprecise properties is very important for the designer. As the exploration mode is based on the use of imprecise properties, it is very important to have techniques to reduce exploration cost by reducing the number of useless tries during the solution search process [9, 1, 11]. A problem with the exploration mode is that the use of general imprecise properties can produce a very important number of solutions and make very difficult the management of these solutions. Furthermore, some families of solutions can be of no interest for the designer and he (she) would like to avoid generation of such solutions in subsequent generations. As the modeller does not know the designer's preferences, machine learning can be used to teach the modeller what kind of scenes are, or are not, interesting.

In solution search mode, the designer has a relatively precise idea of the kind of scenes he (she) would like to obtain. Thus, the designer would like to obtain a solution immediately or very quickly from a description using less imprecise properties. Unfortunately the semantic of a property is often ambiguous and several solutions not satisfactory for the user can be faced by the modeller. In such cases, the knowledge of the designer's preferences can guide the modeller in its search. So, if some proposed solutions do not satisfy the designer, it would be interesting to teach the modeller not to examine this kind of solutions. This learning decreases the solution space because, for a great number of scenes verifying the properties of the initial description, some scenes will not satisfy the intuitive idea of the user and these scenes will be avoided.

It is easy to see that machine learning often lightens the designer's work and increases the efficiency of generation. In the following sections we present two methods to implement machine learning for declarative modelling by hierarchical decomposition in order to improve modelling in both exploration and solution search mode.

5. A DYNAMICAL NEURAL NETWORK FOR FILTERING UNSATISFACTORY SOLUTIONS IN DMHD

In order to improve exploration mode generation, but also solution mode generation, we have implemented an interactive machine learning mechanism based on the use of neural networks [3, 4, 7, 1], applied to DMHD. This mechanism is used by the modeller in the following manner:

- During a learning phase, some scenes, generated by the modeller from the initial description, are selected by the user to serve as examples of wished scenes. Each time a new example is presented, the modeller learns more on the user's preferences and this is materialized by a modification of the values of weights associated to the connections of the network. At the end of the learning phase, an acceptation interval is calculated and assigned to each decision cell.
- After the end of the learning phase, the modeller is in the normal working phase : the weights of connections calculated during the learning phase are used as filters

allowing the choice of scenes which will be presented to the user.

The used machine learning mechanism takes into account only relative dimension and position properties of a scene. Form properties are processed by the scene generation engine after selection of dimensions and positions of the bounding boxes of each sub-scene.

5.1 Structure of the used network

The neural network is created dynamically from the description of the scene and its structure is described in the following lines.
To each node of the scene's description tree are assigned:

- A network whose input layer is composed of two groups of neurons (see figure 5) : a group of two neurons whose inputs are w/h and w/d where w, h and d are respectively the width, the height and the depth of the scene associated with the current node; a group of neurons whose inputs are the results of acceptance of the other nodes of the scene. The network contains another layer of two neurons which work in the following manner: the first one evaluates the quantity $i_1*w_1 + i_2*w_2$, where w_k represents the weight attributed to the connection between the k-th neuron of the input layer and the intermediate layer and i_k the k-th input; it returns 0 or 1 according to whether this weighted sum belongs to a given interval or not. The values of weights w_k can be modified during the learning phase. The second neuron computes the sum of acceptance results coming from the second group of neurons of the input layer. Its output function returns 0 or 1 according to whether this sum is equal or not to the number of acceptance neurons. The decision layer of the network contains one neuron and computes the sum of the outputs of neurons, returning 1 (acceptance) or 0 (refusal) according to whether this sum is equal to 2 or not.

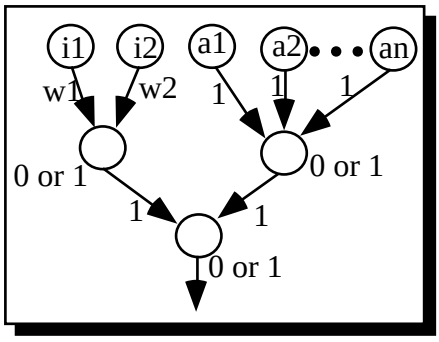


Figure 5 : local neural network

- n*(n-1)/2 networks, corresponding to all possible arrangements of two sub-scenes of the current scene, where n is the number of child-nodes of the current node. These networks have the same structure and the same behaviour as the others except that the inputs of the first group of neurons of the input layer are dx/dy and dx/dz, where dx, dy and dz are the components of the distance d between the two sub-scenes. The values of weights w_k can be modified during the learning process.

Let us consider the scene *House*, described by the following Prolog-like pseudo-code:

```
House(x) ->
    Habitation(x1)
    Garage(x2)
    PasteLeft(x1,x2);

Habitation(x) ->
    HigherThanWide(x)
    HigherThanDeep(x)
    Roof(x1)
    Walls(x2)
    PutOn(x1,x2);

Garage(x) ->
    TopRounded(x,70);

Roof(x) ->
    Rounded(x,60);

Walls(x,p) ->;
```

The generated neural network for this description is shown in figure 6.

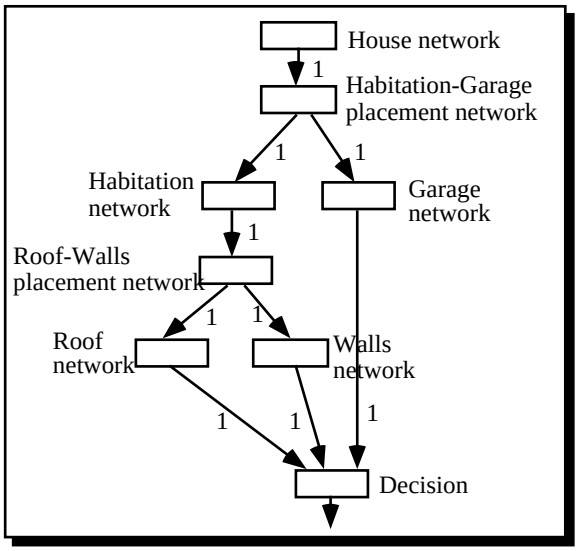


Figure 6 : Neural network generated for the "House" scene description

The acceptance process of a scene generated by the modeller is the following:

- If the current node of the generated scene is accepted at the current level of detail and relative placements of all the sub-scenes of the node at the next level of detail are also accepted by the associated neural networks, the scene is partially accepted at the current level of detail and acceptance test is performed with each child-node of the current node. A scene is accepted at the current level of detail if it is partially accepted for each node up to this level of detail.
- A scene generated by the modeller is accepted if it is accepted at each level of detail. Otherwise, the scene is

refused and it is not presented to the user.

5.2 The machine learning process

For each neural network of the general network, machine learning is performed in the same manner.
From each new example of scene presented by the user among the scenes generated by the modeller, each local network learns more and more and this fact is materialised by modifying the weights of connections between the association (intermediate) layer and the decision (output) layer. The formula used to adjust a weight w is the following:

$$w_{i+1} = w_0 - w_i \, V(X_{i+1})$$

where

w_i represents the value of the weight w at step i,

w_0 represents the initial value of the weight w,

X_i represents the value of the input X at step i,

$V(X_i)$ represents the variance of values $X_1, ..., X_i$.

This formula permits to increase, or not to decrease, the value of the weight when the corresponding input value is constant and to decrease the value of the weight when the corresponding input value varies.
When, at the end of the learning phase, the final value of the weight w has been computed, the value of the quantity (w - w0) / m is estimated, where m is the number of examples selected by the user. If the value of this quantity is less than a threshold value, the weight w takes the value 0; otherwise, it takes the value 1.
The quantity computed by the output function of the first neuron of the intermediate layer of each local network is:

$S = w_1 X_i + w_2 Y_i$,

where X_i and Y_i are the input values at step i.

Thus, at the end of the machine learning phase, each output function of the first neuron of the intermediate layer of each local network will compute the quantity:

$S_a = w_1 A(X) + w_2 A(Y)$

where A(X) represents the average value of the input values $X_1, ..., X_n$.

During the phase of normal working of the modeller, a solution is partially accepted by a local neural network if the output value computed by the output function of the first neuron of the intermediate layer belongs to the neighbourhood of the value S_a.

Let us consider three cases for each local network:

1. $w_1 = w_2 = 0$.

The value of S_a is then equal to 0 and all solutions are accepted.

2. $w_1 = 0, w_2 = 1$.

The value of S_a is then equal to A(Y). Only the input value associated with w_2 is important and only scenes which have, for their corresponding to the current local network part, input values close to the value of A(Y) are selected.

3. $w_1 = w_2 = 1$.

The value of S_a is then equal to A(X) + A(Y). The two quantities associated with w_1 and w_2, are important. The parts of scenes selected by the current local neural network will be the ones whose input values associated with w_1 are close to A(X) and input values associated with w_2 are close to A(Y). However, such a scheme does not prevent acceptance of other undesired scenes for which the sum of entry values is equal to A(X) + A(Y).

5.3 Discussion

Although neural networks are fully efficient for linearly separable problems, their application to declarative modelling for a non linearly separable problem, selecting scenes close to those wished by the designer, gives results globally satisfactory for exploration mode generation because it lightens the designer's work by filtering the major part of non interesting solutions. Machine learning is performed with little information and already learnt knowledge can be used for continuous machine learning where the modeller avoids more and more non interesting scenes.
Figure 7 shows some rejected scenes during the machine learning phase for the above "House" scene description. The purpose of this phase was to teach that the "Habitation" part of the scene must be wider than the garage part. In figure 8, one can see some generated scenes for the "House" scene description, after the machine learning phase.

Of course, this kind of interactive machine learning reduces the number of solutions shown to the user but not the number of tries; the total exploration time remains unchanged. Another drawback of the method is that learning is not efficient if the number of given examples is small and that learning can be avoided if the given examples are not well chosen.

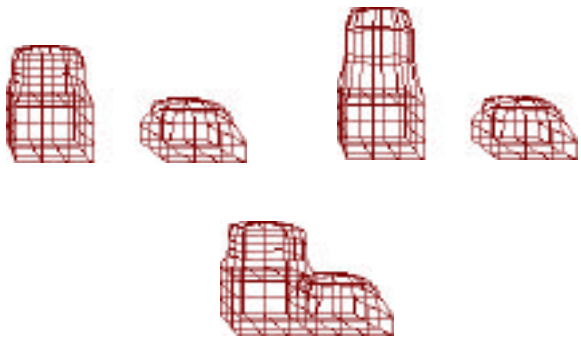


Figure 7: Scenes generated from the "House" description and rejected by the user.

We think that it is possible to use the same neural network to reduce the number of tries during generation. To do this, after the machine learning phase, the neural network has to be activated as soon as possible, after each enumeration phase in the constraint satisfaction process. In this manner, the next step of the resolution process will take place only if the scene goes through the neural network filter.

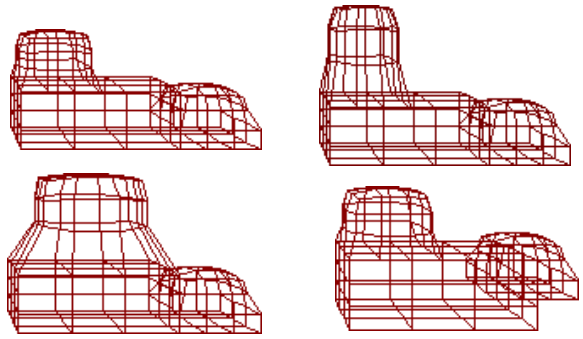


Figure 8: Machine learning for the "House" description. Only generated scenes with the wished shape are presented to the user.

6. A MACHINE LEARNING APPROACH USING GENETIC ALGORITHMS

Genetic algorithms [14] can also be used to perform machine learning because they are supposed to follow natural evolution lows, based on selection of individuals (chromosomes). Chromosomes are selected, according to their importance, to participate to the next generation. Finally, an important number of generated chromosomes has the properties corresponding to the evaluation function.

Genetic algorithms based machine learning has been implemented on MultiCAD [13], an information system for CAD, which uses the MultiFormes' generation engine to generate scenes.

In this implementation, the initial generation is composed of an arbitrary chosen number of scenes generated by the generation engine of MumtiFormes and the purpose of machine learning is to generate other scenes verifying the properties wished by the user. Each initial scene is a chromosome, encoded in a special manner.

6.1 Encoding chromosomes

Information taken into account to encode a scene generated by the MultiFormes engine is the one concerning bounding boxes of sub-scenes corresponding to terminal nodes of the generation tree. Each sub-scene is represented by pertinent information describing its bounding box. This information is the following:

- coordinates of the bottom-front-left corner of the bounding box,
- width, depth and height of the bounding box.

Assuming that each coordinate, as well as width, depth and height of a bounding box are integer numbers, each bounding box of sub-scene needs 6 integers to be described (figure 9).

x	y	z	w	d	h
0	1	3	5	3	6

Figure 9: sub-scene's representation

As genetic algorithms use chromosomes encoded as binary strings, each information describing a sub-scene is encoded as a binary string. In the current implementation, the length of each binary string is limited to 3, in order to represent quantities from 0 to 7. So, the terminal sub-scene of figure 9 will be encoded as shown in figure 10.

000	001	011	101	011	110

Figure 10: binary string encoding for sub-scene of figure 9.

The whole scene is encoded with a chomosome including all encoded terminal sub-scenes of the scene.

6.2 The genetic algorithm

The purpose of the genetic algorithm is to create new generations of chromosomes, i.e. solution scenes, from an initial population by applying genetic operations to the chromosomes of the current generation.

The genetic operations used are the classical ones, that is *cloning*, *crossover* and *mutation*. The probability for a chromosome to be submitted to a genetic operation is proportional to the importance of this chromosome. The importance of a chromosome is determined interactively by the user who assigns a fitness score to each chromosome of a new generation.

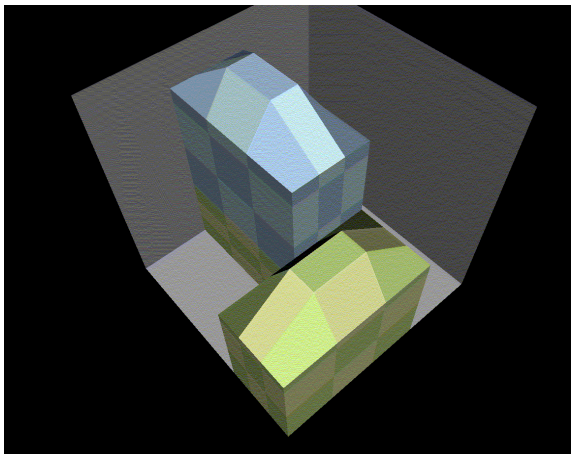
The genetic algorithm starts with an initial set of solutions created by the generation engine of MultiCAD, which is the one of MultiFormes. The user assigns a fitness score to each solution and the genetic algorithm based scene generation process starts.

At each step of generation, chromosomes are selected according to their importance and the genetic operations are applied to them, in order to generate the next generation of solutions. Unlike in many genetic algorithm applications, mutation is applied to each step of the generation process. The whole process can be stopped when at least one of the solutions of the new generation satisfies the user.

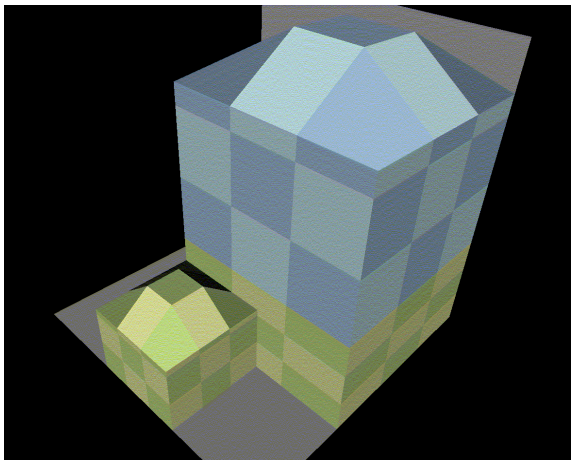
6.3 Discussion

Figure 11 shows two solutions generated by the implemented genetic algorithm, for the scene "House" described in section 5. Solutions are visualised through a VRML file.

Genetic algorithm based learning is more suitable with the solution search generation mode. The obtained results are quite interesting and the generated scenes are progressively closer to the desires of the user. The main advantage of the genetic algorithm approach is that only some solutions, the initial population, are generated using the time consuming constraint satisfaction techniques, used by the main generation engine of Multiformes. The final solutions are obtained by the evolution process used in genetic algorithms, with the search being guided by the user on more promising parts of the search tree.



(a)



(b)

Figure 11: scenes generated by the genetic algorithm for the scene "House"

An additional advantage is that the user has to evaluate solutions by groups, rather than one by one as in the main generation engine of MultiFormes. This possibility permits a relative comparison of scenes and improves, generally, results.

On the other hand, as relative positions of the terminal sub-scenes of a scene are not encoded, it is possible to get undesired overlappings between these sub-scenes, especially during the first steps of the generation process. Moreover, some potentially interesting solutions may never appear, due to the choice of the initial population.

7. CONCLUSION

Machine learning can be very interesting in scene modelling, especially whenever declarative modelling is used. We have proved that it is possible to implement machine learning in DMHD and that the obtained results are generally in accordance with our desires. However, there are some drawbacks with these kinds of machine learning.

Neural networks based machine learning presented in this paper allows a reduction of the solution space for scene generation, but it leaves the search space unchanged. So, the generation time is not affected by this kind of learning. Moreover, machine learning is performed only for the current scene description and it is unusable with other scene descriptions.

Genetic algorithms based machine learning allows to reduce the search space but it is also usable with only the current scene description. Moreover, this kind of machine learning can produce intermediate solutions which do not fit with the initial description.

We think that it is possible to implement another kind of machine learning for declarative modellers, complementary of those presented in this paper, allowing to customise the modeller according to the user practices. Each user of a declarative modeller has probably a particular interpretation of some imprecise properties like "A *is close to* B", "A *is higher than* B", etc. This means that the search space to get solutions is not the same for different kinds of users. Machine learning of the user's manner to interpret imprecise properties can reduce the search space for solutions and, thus, the solutions generation time.

8. REFERENCES

[1] D. PLEMENOS, W. RUCHAUD, K. TAMINE, Interactive techniques for declarative modelling. 3IA'98 International Conference, Limoges (France), 28-29 of April 1998.

[2] M. LUCAS, D. MARTIN, P. MARTIN, D. PLEMENOS, The ExploFormes project: some steps towards declarative modelling of forms. AFCET-GROPLAN Conference, Strasbourg (France), November 29 - December 1, 1989. Published in BIGRE, no 67, pp 35 - 49 (in French).

[3] W. S. Mc CULLOCH, W. PITTS, A logical calculus ofthe ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5, 115 - 133, 1943.

[4] F. ROSENBLATT, The perceptron: a perceiving and recognizing automaton. Project Para, Cornell Aeronautical Lab. Report 85-460-1, 1957.

[5] D. PLEMENOS, A contribution to study and development of scene modeling, generation and display techniques - The MultiFormes project. Professorial Dissertation, Nantes (France), November 1991 (in French).

[6] D. MARTIN, P. MARTIN, An expert system for polyhedra modelling. EUROGRAPHICS'88, Nice (France), 1988.

[7] D. PLEMENOS, Techniques for implementing learning mechanisms in a hierarchical declarative modeller. Research report MSI 93 - 04, Limoges (France), December 1993 (in French).

[8] D. PLEMENOS, Declarative modeling by hierarchical decomposition. The actual state of the MultiFormes project. International Conference GraphiCon'95, St Petersbourg, Russia, 3-7 of July 1995.

[9] D. PLEMENOS, K. TAMINE, Increasing the efficiency of declarative modelling. Constraint evaluation for the hierarchical decomposition approach. International Conference WSCG'97, Plzen (Czech Republic), February 1997.

[10] M. LUCAS, E. DESMONTILS, Declarative modellers. Revue Internationale de CFAO et d'Infographie, 10(6), pp. 559-585, 1995 (in French).

[11] P.-F. BONNEFOI, D. PLEMENOS, Constraint satisfaction techniques for declarative scene modelling by hierarchical decomposition. 3IA'2000 International Conference, Limoges (France), May 3-4, 2002.

[12] W. RUCHAUD, Study and realisation of a geometric constraint solver for declarative modelling. PhD Thesis, Limoges (France), november 15, 2001 (in French).

[13] N. VASSILAS, G. MIAOULIS, D. CHRONOPOULOS, E. KONSTANTINIDIS, I. RAVANI, D. MAKRIS, D. PLEMENOS, MultiCAD-GA: a system for the design of 3D forms based on genetic algorithms and human evaluation. SETN02 Conference, Thessaloniki (Greece), April 2002.

[14] D. E. GOLDBERG, Genetic Algorithms, Addison-Wesley, USA, 1991.

About the authors

Dimitri PLEMENOS is professor in The University of Limoges (France) and director of the MSI research laboratory.
E-mail: plemenos@unilim.fr

Georges MIAOULIS is professor in the Technical Educational Institute of Athens (Greece).
E-mail: gmiaoul@teiath.gr

Nikos VASSILAS is professor in the Technical Educational Institute of Athens (Greece).
E-mail: nvas@teiath.gr