EFFICIENT VIRTUAL PLANT DATA STRUCTURE FOR VISUALIZATION AND ANIMATION

Marc Jaeger
CIRAD UMR AMAP, EPI Digiplante (joint INRIA, CIRAD, Ecole Centrale Paris project team)
TA A51 / PS2, 34398 Montpellier Cedex 5, France

Ruoxi Sun and JinYuan Jia
College of Software Engineering, Tongji University
Shanghai, China

Vincent Le Chevalier
Ecole Centrale de Paris, MAS Laboratory, EPI Digiplante (joint INRIA, CIRAD, Ecole Centrale Paris project team)
Grande Voie des Vignes, 92295 Chatenay-Malabry Cedex France

ABSTRACT

Virtual plants are interesting but costly objects in video games and virtual reality applications. Despite advances on Level of Detail approaches and increasing GPU power, previous work fails in producing convincing and generic vegetation with low bandewidth requirements for web display. The efficiency drops down considering costs of compression and inner structure redundancy is still poorly explored.

We propose here a tree graph as a generic output of simulation, easy to implement from any rule based plant generator. This graph allows to efficiently compress similar structures, that appear naturally as a result of procedural definition. Nodes carry minimal attributes, mainly related to age. Upon display requests, nodes orientation and precise geometry are computed using a simple graph exploring approach based on a very small number of parameters,. This exploration allows to generate all past growth stages and dynamic effects such as pruning. We show how this graph can be efficiently generated from L_systems, as well as from a functional-structural plant simulator. We illustrate that, thanks to this encoding and reconstruction scheme, a simple tree simulation can generate a wide range of geometrical plant exports at various ages, and is compatible with classical level of detail rendering approaches for geometric models.

KEYWORDS

Computer Graphics, Virtual plants, natural scene rendering, L systems, FSPM models

1. INTRODUCTION

In video games and virtual reality applications, natural object visualization and transfer of geometrical data are still a challenging topic, despite all the technological progress on graphics cards performance. This is especially the case of scenes composed by virtual plants seen from close views. In the past decade, a wide range of studies were held on the subject , mainly based on significant decrease of plant geometry, replacing complex geometrical structures by simple ones according to viewer distance. The main distinction found in literature is between image based approaches and compressed polygonal approaches. Among image based approaches, one can note the use of impostors [Max 1999]. Impressive leaf cluster impostors examples and an interesting review of various techniques used for realistic single tree visualization can be found in [Fernandez 2007]. Images approaches show potential for fast rendering but suffer from relative complexity on close view due to the multiplication of textures, thereby incurring high bandwidth costs if Web visualization is the target. Many authors did also work on polygonal representations, simplifying the tree structure and its mesh conversion, with specific methods on foliage [Deng 2007]. An overview of such methods is given by [Deng 2010]. However, sets of level of details models -multi resolution models or just billboards- introduce costly pre-processing constructions. And, once again, for Web visualization, sets of multi-scale representations lead to high bandwidth costs if realistic close and far views are requested on the client display. Finally, those techniques seldom address growth animation. In a natural scene with plants of the same species with various ages, such techniques, working at a low semantic level, show poor reusability. Several authors show that efficient tree generation can drop down generation complexity from exponential to linear, with automates [Yan 2002] or L-systems [Brownbill 1998], but those approaches were not explored for data transfer and rendering. While compression using grammars is popular in data analysis [Nevill-Manning 1994], it is seldom used for the encoding and transfer of virtual object.

In order to visualize natural scenes on the Web, an efficient solution consists in sending to the client the construction rules and the plant generator. Such approaches were already adopted for plants reconstructed from photographs [Sun 2009] with L_systems. But it is easy to imagine that such simulators may not be simple, low cost and compact software pieces. L_systems [Prusinkiewicz 1990] are widely used in literature, however, as described by [Deussen 2004], various techniques are currently available and popular (automates, patterns [Deussen 1997], specific grammars, fractals, …), leading to lots of specific tools on the client.

The idea we want to push here is that, for a wide range of tree plant generators, there may be intermediate structures of interest between rules (specific to each approach) and the final geometrical structure rendered at full scale (usually a oriented tree graph). We present here a tree graph, a structure open to most plant generators. We show that, if we take care of basic botanical constraints, this structure can drastically compress duplicated patterns in the tree, and can be used to generate all past growth steps of the plant. As applications, we also propose a reconstruction frame allowing to generate the growth steps geometry and show its potential on natural scenes, using tree graphs generated by different simulators.

In the next section, we detail how we encode tree structures taking advantage of redundancies deduced from biological typologies . In the third section, we show how this coding can reconstruct the geometry of plants at several growth stages. The last section illustrates example applications with their performances.

2. EFFICIENT TREE ENCODING, THE TREE GRAPH

We remind here some basic elements of rule based vegetation. Our approach uses the plant modeling formalism GreenLab [De Reffye 2003], implemented with a L_system [Prusinkiewicz 1990]. We describe then the tree graph with its topology and specify a set of node attributes.

2.1 Efficient Tree Structure Generation

As described by O. Deussen [Deussen 2004], rule based plant generation applies rules on an initiator, what ever the underlying technique is. However, to generate a wide variability of shapes, simulators must allow for variability in both the structure generation (the tree topology) and the geometry (angles), leading to a set of rules. Another important point is the age, or the number of cycles during which the rules are applied in the course of a simulated growth process. Based on these concepts, a L_system production is implemented that uses the notion of substructures, as defined by Yan et al [Yan 2002], minimizing redundancies.

2.1.1 Basis of Procedural Tree Generation: The Metamer as a Basic Node

In botany, plant architecture is defined from a typology of various axis characters [Barthelemy 2007]. Tree architecture is build from axis, defined by growth units (annual shots in temperate trees), themselves defined by nodes. Each node carries a leaf from which a lateral bud can grow a new axis. The node with its leaf is called a metamer and thus defines the basic topological unit.

The plant structure development involves two aspects [De Reffye 2003]: the appearance of new metamers (on the same axis from an apical bud or on starting a new lateral axis from a lateral bud), and the geometrical expansion of the existing metamers (usually short in time). This second aspect is mainly related to environmental conditions, while the first one is essentially caused by genetics.

New metamers appear usually in a group (the growth unit) with various characters in terms of viability and branching potential; biologists show that this potential is related to the physiological age. In practice, 2 to 4 physiological age steps explain drastic metamorphosis in plant developments and describe the basic pattern in trees: the “architectural model” [De Reffye 1988, Barthelemy 2007]. Physiological age can be seen as a morphogenetical status starting from high development potential (age 1) to low ones (transformation to flower for instance). In classical approaches, tree generators define implicitly different characters from the ramification order. To summarize, a plant can be generated from rules depending on a physiological age (or branching order), adding new sets of metamers (growth units) from end of existing axis or from metamers leaf insertions. Geometrical aspects (angles, length) can also be associated to this physiological age index.

2.1.2 GreenLab's Dual Scale Automaton and Substructure

For our biologically inspired model, Xing Zhao [Zhao, 2003] has shown that a dual scale automaton can be used to described and simulate any plant architecture. The deepest level describes the metamer successions (the micro states) in a typical growth unit at a given physiological age. The second level describes unit successions (the macro states) with possible physiological age changes, even on the same axis.
In the GreenLab model, all macrostates of a given physiological age born at the same date are similar (sharing the same attributes and the same fate) and define a subtree called a substructure by Yan et al. [Yan 2002]. Their evolution can be computed only once.

2.1.3 Encoding Dual-Scale Automaton and Sub Structures with L_systems

L-system is a grammar rule string rewriting system. Starting from a initial string (axiom), letters are replaced by production rules by words using the same alphabet, building a new string. We translate the concept of Microstate in Dual-Scale Automaton into a L-system, using the following alphabet:

Bi	*Bud at physiological age i (i >= 0)*
Mi	*Metamere at physiological age i (i >= 0)*
[]	*Used to start a new axis, as defined in classical L_systems*

The general description of Microstate (sequence in a growth unit) can then be written as follow:

Bi -> Mi ([Bj]...) Bk (j, k >= i >= 0). We note here "*(...)*" to say that there could be none or several lateral buds. Constraints on physiological age *(j, k >= i)* translates the biological assumption that the physiological age of a new element is always higher than than of its father. The Mi letters stand for internodes of the tree, and "*[*", (respectively "*]*"), stack (respectively unstack) a sub-tree from a lateral bud.

Macrostates (growth units) evolution is performed by duplicating microstates:

Macrostate i (n) {list of Microstates} where *i* stands is the macrostate physiological age, *n* the number of cycles the macrostate is replicated until it is transformed to Macrostate *i+1.*

Applying these rules, the resulting string may reach a huge size very quickly but that can be reduced thanks to sub-structuring consideration. Let's note *Rit*, a bud *Bi* with *i* as its physiological age and *t* as its birth time. Now, at cycle *t*, before rewriting the tree string, any bud appearing with physiological age *i* is replaced by *Rit*. *Rit* is associated to a string *Sij*. Rewriting does not affect *Rij*, but affects *Sij*. The principle applies recursively on all *Sij* strings. The number of *Rit* instances, called multiplicity, is recorded accordingly.

2.2 The Tree Graph: a Simple, Compact Tree Structure Encoding

The proposed encoded virtual tree structure is based on a classical graph description where nodes stand for the tree nodes : the metamers and links for the topological relations between them, similar to [Lintermann and Deussen 1999] work. Original point is that each node is also given time and age attributes.

2.2.1 Principles: Using the Metamer Description Level

A graph description build from metamere nodes can easily be designed from its sons; in our case we differentiate the son on the same axis from the lateral ones. We have implemented this graph from a simple list of nodes. Each node is given an id, its number of sons, the node id of the son on the axis (set to -1 if not existing) and the list of the lateral son(s) node(s) ids. We illustrate here an example of a single binary tree (Leeuwenberg tree model) of age 2, with units also defined by 2 microstates (Figure 1).

Figure 1. A simple binary tree

Using the L_system approach as defined earlier, the corresponding grammar for this model is:

Macrostate 1 (1) B0 -> M0 B0
B0 -> M0 [B0] [B0] B0
Macrosate 2 (1) B1 -> F (F stands for a flower)

At age 2 (ie after two iterations), the corresponding string is:

M0 M0 [M0 M0 [B0] [B0] B0] [M0 M0 [B0] [B0] B0] F

Assuming that each node is described on a line, the resulting topological encoding can be written as :

Node id	Number of sons	Successor	Axillaries	
0	1	1		
1	3	-1	2	4
2	1	3		
3	3	-1	-1	-1
4	1	5		
5	3	-1	-1	-1 (*)

(*) an alternate way could be to specify no sons at all

This encoding leads to a tree is easy to define for nearly all kind of ramified patterns, created from generators or simply measured on real world objects.

2.2.2 Efficient Topology Encoding: Avoiding Redundancies

We have shown that in most of procedural generated virtual plants, the simulators generate similar patterns at a given cycle(duplicate nodes). In our graph, we will just refer to the same.

On our exemple using the L_system approach, the corresponding grammar for this model is of course still the same but at age 2 (ie after two iterations), the corresponding string can now be written as:

Macrostate 1 (5) { M0 M0 [R01] [R01] F
S01 (2): M0 M0 [B0] [B0] B0 }

where *S01* stands for the substructure *R01*. In our typology we chose explicit time and aging; the substructure comes from a bud at physiological age 0 and did appear at year 1. The number (2) in the bracket is the multiplicity of this substructure.

The resulting topological encoding can now be written as (-1 as node id means no successor):

Node id	Number of sons	Node successor	Axillary Nodes	
0	1	1		
1	3	-1	2	2
2	1	3		
3	3	-1	-1	-1

2.2.3 Encoding Geometrical Metamer Properties

Each graph node can also be attached geometrical attributes. It is understood that in case of multiplicity, these attributes are the same for each instance, except origin, initial direction and mechanical behavior. Typical geometrical parameters are the length and the diameter. In our implementation we chose to encode volumes, since our plant model is a functional one and computes masses. More precisely we encode the pith volume and the rings volume of the corresponding metamer. Since each metamer carries leaves (see 2.1 section), we add the leaf volume. Note that the branching and phyllotaxy angles are not stored, they are implicit to physiological age.

2.2.4 Encoding Time and Aging Metamer Properties

In many plant simulators, aging attributes can be set from the generator. The time of apparition (or the cycle) can be stored as a birth date. Nodes typology such as physiological age is also an aging attribute to which additional parameters such as angles or graphical material properties are related.

3. RECONSTRUCTION AND VISUALIZATION

We can reconstruct the simulated plant from its encoded graph simply by defining branching and phyllotaxy angles and traversing the graph. Moreover, with time and aging attributes, the encoded graph allows to generate all past growth stages.

3.1 A Fast and Easy Reconstruction Way

3.1.1 The Reconstruction Pseudo Code

When geometrical representation is requested, the tree graph and its attributes are loaded in memory. The tree graph is a classical oriented graph, that can be traversed in several ways, in prefix order or not, using a stack or a recursive implementation. In our case, we traverse the graph by oldest node first, axis by axis. In other words starting by the trunk, order by order. We chose this way in order to generate easily Level of Detail models: merging metamers of the same direction together, or stopping rendering at a given branching order.

A pseudo code algorithm is given below (non recursive):

```
stack ← first node  // first node is the tree origin, at bottom trunk
while (stack not empty)  {
    current node <- stack
    if  ( node already seen)
        draw substructure generated from current node
      else  {
            while (current node exists)    {
                compute node position and draw (export) node geometry
                set node as seen
                for each lateral node
                    compute lateral node orientation and stack the lateral node
                current node ← next node  //on the same branch
                }
            }
     while (top of stack node seen)
            unstack
    }
```

3.1.2 Building the Tree Geometry

A fast way to define the geometry is to assign a unit length to each node. For the diameter, a polynomial law function of metamer rank can be used [De Reffye 1988]. Lengths can also be parametrized from physiological age or correlated to the existence of lateral nodes (usually shorter). Usually length and diameters values are computed during tree generation and can be attached to each node attribute. In functional structural models, biomass volumes are defined for each node (for the pith and the rings). Using allometric rules, length can be deduced as well as diameters (See annex for this conversion).

Lateral nodes orientation and position is set according to the father node direction with two angle values: the insertion angle and the phyllotaxy angle. Both can be considered as related to the physiological age. These values are therefor not interesting to set as node attributes, since they are the same for each node of same physiological age (1 to 4 different values for each angle for the full tree).

Once node positions and orientation are known, exports to various format can be performed, generating components display lists [Jaeger 1992] or a skeleton base with its radii for the woody part.

Note that, in the case of substructures, this formalism takes its full advantage when orientations and positions are computed relative to the father node. Geometrical operations can then be encoded as affine transformations, easy to convert to graphical libraries instructions. On first call, substructures can be stored as nested graphical object definitions, to be instantiated when substructures need to be drawn again.

3.1.3 Leaves

As specified in the previous section, each node (metamer) carries leaves, and lateral buds grow at insertion of leaves. The orientation and the position of leaves on the axis is thus clearly defined: on the top of the metamer, with the orientation of the lateral axis. In practice, in order to avoid self collisions, the insertion angle is smoothly increased, and the leaf origin pushed to the branch surface (not the center). Similar to node length, the organ size can be defined from an empirical law, or computed from the simulator.

This approach can be extended to all organs: flowers and fruits. An alternate way is to specify a type attribute to each node (for instance in the L_system examples in section 2, production involves the letter F for flower apparition) and thus allow the simulator to allow specific location for organs.

However, aging is a key point for organs, since they are usually not persistent. As described below, self pruning can easily be performed when node birth date is given as an attribute.

3.2 Building a Wide Range of Tree Geometries from a Single Tree Graph

So far, given a tree graph computed at a given age and a limited set of parameters (length law, branching and phyllotaxy angles, drawing organs or not), a user can interactively design several geometries for the tree and generate display or exports. Taking advantage of aging and time information in node properties, we can produce new geometrical outputs, using the temporal dynamics of plant development.

3.2.1 Self Pruning

Self pruning is easy to implement. Let us first consider the specific case of leaf organs. Tree species growing in temperate areas have leaves that are alive during one single cycle (one year); leaf display is thus decided based on a single threshold on the birth date. More generally, self pruning can appear after a given pruning period following death. During this period, geometrical or material properties attached to the organ can be different in order to simulate metamorphosis (from flower to fruit) or senescence. For a given tree age, a death status can be computed as:

death status = tree birth date + tree age – organ birth date – organ life span + pruning period

If the organ is still alive, the status is positive, the geometry has to be generated. If the status is greater then the pruning period, the organ is functional, otherwise it is not -but still on the tree-, its color or shape attributes may be changed. If the status is negative, no geometry has to be generated. An example of such a pruning is shown in Figure 2, leaves are getting darker before getting pruned on older stage.

The method can be applied on nodes as well. In this case the major difficulty lies in the definition of the axis life span. This information can be retrieved exploring the substructure born from the node, tracking the nearest birth date of the substructure. This is easy to realize but can be a little bit costly. In our implementation we proceed in two steps. We retrieve first the life span for a given physiological age (this is supposed to be constant), since this period is supposed to decrease with the age value, the substructure exploration can be optimized. Application of self pruning on axis is illustrated on Figure 4 (axes drawn in light gray) and on Figures 5 and 6 where small dead branches on older trees are pruned.

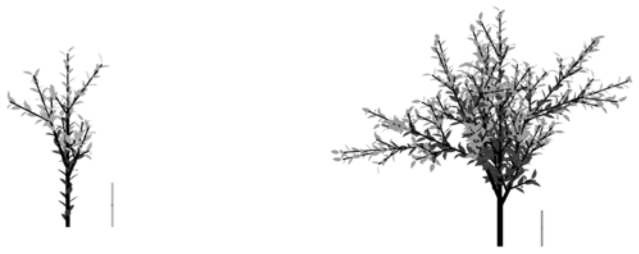
Figure 2. Leaves life span and pruning

3.2.2 Growth Animation

The basic idea is to explore the graph with constraints over the birth date. The reconstruction algorithm remains the same, including the self pruning effect, just ignoring nodes born after the specified date, and setting the final tree age to the appropriate one. For the geometrical evolution, we suppose that the expansion duration is one growth cycle (such is the case for temperate trees). Under this condition, node length is stable as well as leaf volumes (ie areas). But axes diameters are not, they increase during tree growth because of ring accumulation. We use here a simple quadratic interpolation from metamer age to its final age. Figures 4, 5 and 6 show the interest of this approach. Two different tree graphs are used in figures 4 and 5, and both graphs used in Figure 6.

4. RESULTS AND PERFORMANCES

4.1 Architectural Models

As specified in section 2.1, plant topology and geometry is defined by its architectural model. Figure 3 show that the proposed L_System based on the dual scale approach is able to generate such architecture. An encoding graph does not manage the geometrical arrangement of the axis, such as the phyllotaxy. Using this approach, there is no difference between Rauh and Massart models. They differ in the spatial organization of the branches, orthotropic for Rauh and lying in the plane for Massart, managed in the reconstruction tool.

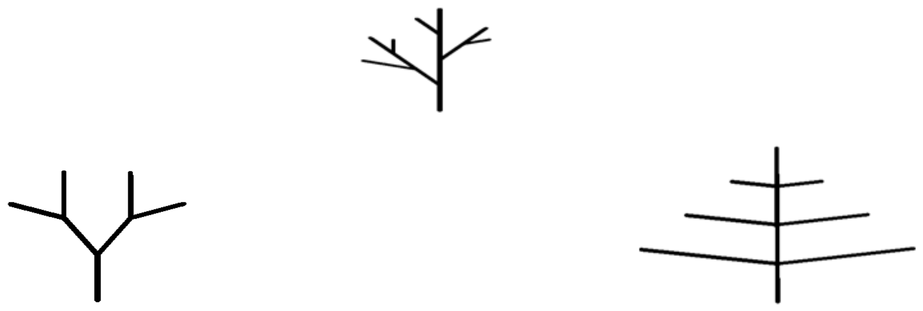
Figure 3. Simulation of three architectural models. From left to right Leeuwenberg, Rauh, and Massart models

4.2 Simulated Plant Examples and Application to Landscape Scenes

The approach is applied here on two “realistic” cases and used in a small forest scene. Geometrical export defines a component display list as specified in [Jaeger, 1992]. Four components are generated: woody parts (one per node standing for branch segments) and 3 kinds of leaf components, sharing the same geometry but with different graphical materials (diffuse, specular and rugosity contributions) according to their age. Each node in the graph satisfying the condition on age thus generates a set of wood components (whose cardinal is equal to the node multiplicity) and its corresponding leaves (if not pruned). Realistic view is generated from a classical polygonal OpenGl based renderer with shadow depth maps, fog and depth of field.

The first example reproduces the example given in [De Reffye, 2003]. The generator is a simple L_system production tool. In this example, an unique plant of age 8 is computed, and no geometrical attributes were simulated. All nodes show the same length. An empiral law defines the final diameter from the plant age and the node age. Phyllotaxy is 147.5 degrees and insertion angles varies from 60 to 75 degrees according to the branching node physiological age.

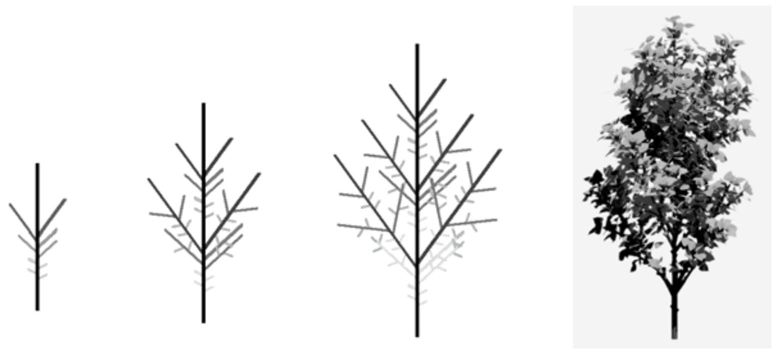

Figure 4. A realistic case. Left, flat sketch views plant reconstruction at age 2, 3 and 4. (grey levels stands for physiological ages). Right, rendered plant reconstruction with leaves at age 7 and phyllotaxy angles set to 147.5 degrees.

The second example illustrates a structural fonctional simulation output, using the continuous GreenLab simulator [Li, 2010]. The tree was computed with a daily simulation (no discrete growth cycles) for a duration of 245 months (20 years and a half). Time and aging attributes are thus expressed in floating point values. Node length and diameter are computed from volumes as defined in section 3.

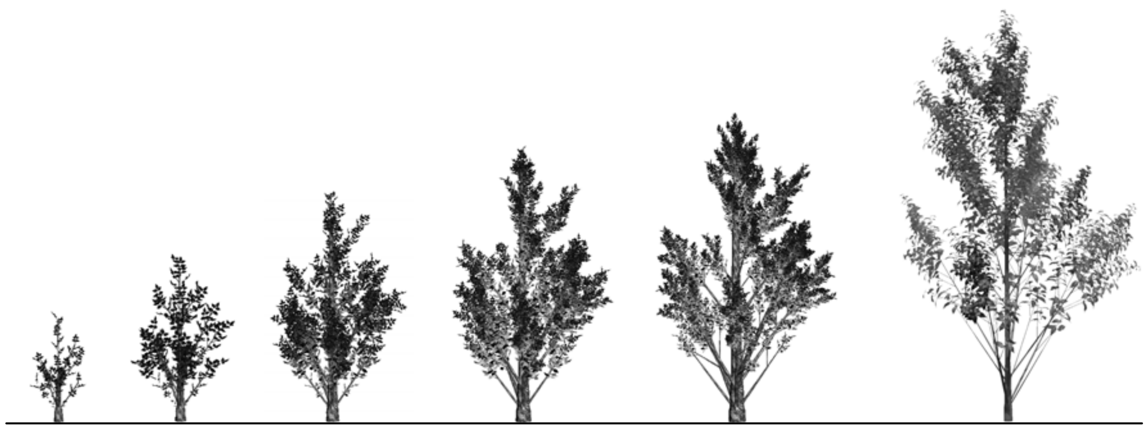

Figure 5. A structural functional tree. From left to right, the five first views show the tree reconstruction and rendering at ages 3, 5, 7, 8, and 10. Right, realistic view at age 11, with an smaller allometry factor, leading to thinner axes.

We finally build a small scene using both encoded structures. The scene contains nearly 200 trees corresponding to 14 different trees. In fact only two tree graphs where used to build six reconstructions from ages 3 to 9 of the first plant (see Figure 4) and 8 reconstructions (Figure 5) of the second tree. The scene description is a simple ascii file listing the positions and rotations of the various plants, similar to a VRML with transform nodes and inline calls. The full description scene is therefore very compact. The 3D renderer is adapted from a simple polygonal rendering tool, calling the reconstruction tool on the first instance of each plant. Shadow fog and depth of field effects are post processed using depth maps techniques (Figure 6).

4.3 Performances

In our examples, graph generation has no measurable memory and CPU cost, since close to inner structures generation,. On our examples, reconstruction user time is low, it reaches 2 seconds on the biggest tree (Figure 5, at age 50), mainly due to export to disk.

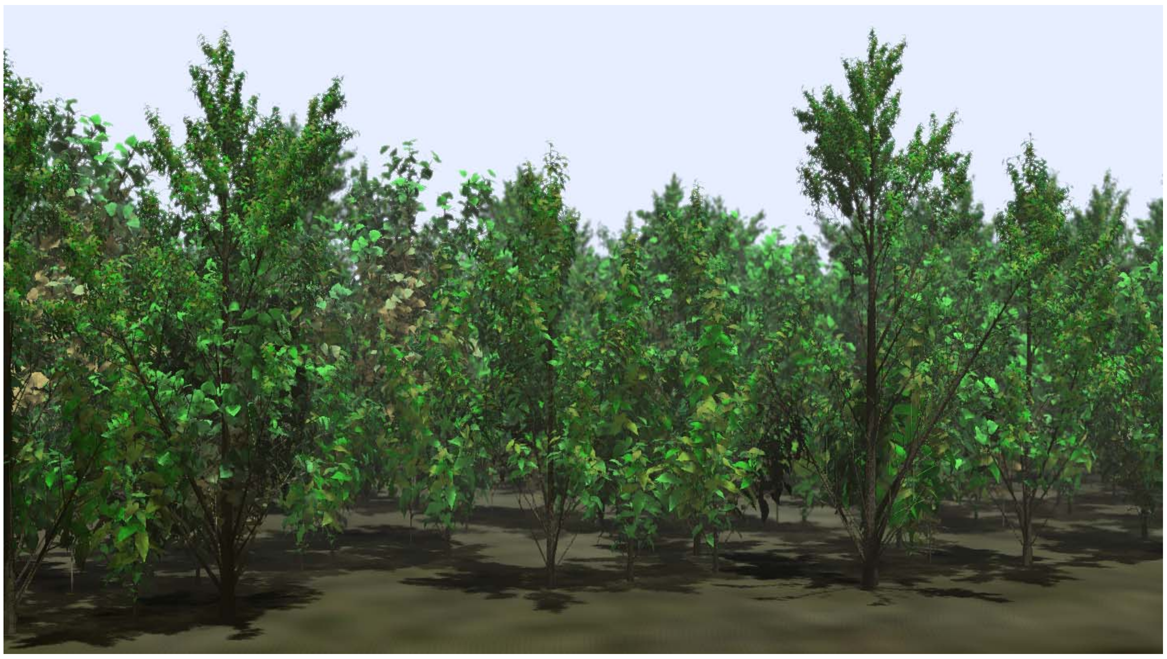

Figure 6. A small natural scene build from a tree position list and two tree graphs.

We summarize the performance gained on several examples shown in this paper in Table 1. It contains the graph file size and the number of nodes (metamers). The age of the reconstructed tree is then given together with the final simulated age. The number of components (internodes and leaves) of the reconstruction is given. Line Tree space is indicative, and could be understood similar to a geometrical polygonal size storing 50 to 80 bytes per component. The sequence size is the size of all Line trees generated from the tree graph from age 1 to final age.

Table 1. Tree graph size versus geometrical output size.

Tree model	Graph size	Nodes	Age vs Final age	Components	Line tree size	Sequence size
Fig. 2 right	2.0 Kb	225	6 on 10	26162	1.2 Mb	32 Mb
Fig. 4 right	2.2 Kb	252	7 on 10	5570	251 Kb	2.5 Mb
Fig. 5 middle	16.7 Kb	2006	7 on 19	3287	156 Kb	23.8 Mb
Fig. 5 right	16.7 Kb	2006	10 on 19	10049	474 Kb	23.8 Mb
Fig. 5 (+)	18.9 Kb	2258	19 on 19	59161	2.54 Mb	23.8 Mb
Fig. 5 (++)	96.7 Kb	8234	50 on 50	1798357	65.7 Mb	918 Mb

(+) Tree in fig. 5 was reconstructed at age 19 but not shown in the paper.

(++) Tree in Fig. 5 was simulated and reconstructed at age 50 but not shown in the paper.

Comparing graph sizes and components number show that substructure encoding leads to drastic compression. In Fig 6, the size of the scene description is close to 60 Kb, including the two graph size. The total number of the components is over 3 millions.

5. CONCLUSION

Virtual tree geometries have high bandwidth and rendering costs when natural scenes are to be displayed on client machines. Compared to image based techniques, rule based approaches seems to show new interests but lead to heavy specific developments on the client. However, we underline that rule based virtual plants approaches share common basic elements in their definition of virtual plants. Elementary patterns, rules of developments leading to the addition of new components, and, indeed, duplicated similar structures.

We proposed a structure encoding focused on the plant topology and these elementary patterns. We described an efficient tree graph structure to encode rule based virtual plants, suitable to various plant generators: L_systems, dual scale automaton, patterns. Based on node links (related to biological metamer notion), the graph avoids similar structure encoding, referring simply to a tree axis typology and the birth date (cycle) of the first element of the sub tree. We showed how such a structure can be implemented with grammars and gave the generic reconstruction frame. Adding time and aging attributes allows to reconstruct geometry of the plant at any growth stage, in a simple, fast graph exploration. Organ reconstruction is easy to add including aging effects such as self pruning.

The approach is suitable for a wide range of applications from multimedia applications to scientific studies; and is really efficient for Web applications.

However, since the approach takes advantage of inner similarities, geometrical deformations related to absolute position or orientations are ignored. Such are the cases of mechanical behaviors (bendings, twists, …). These deformations should be post processed, perhaps using the GPU capabilities.

Diameter evaluation while reconstructing the former ages is currently performed using an interpolation, giving graphical satisfaction, but biologically not founded. Rigorously, the diameter is function of the past accumulated leaf area, that can be evaluated from the graph but at a higher cost.

Finally, we are currently exploring two interesting tracks using this tree graph.

The first one is to use the graph exploration as a LoD tool: attributes such as physiological age, or number of nodes or volume of the underlying substructure gives interesting criteria in terms of feature importance. The second one is even more captivating: environmental pressure, on small cultivated plants mainly changes the production, and the development speed, but the topology remains quite stable. It means, that, for cultivated plants such as rice, wheat, sun flower... the same tree graph could be used with various environmental conditions. Qualitative variability in virtual crops could thus be efficiently visualized.

ACKNOWLEDGEMENT

This work is supported by the following projects: ARCUS ChongMing Agropolis Region Languedoc Roussillon, and french National program support ANR in project ANR 3Words

REFERENCES

Barthelemy, D. and Caraglio, Y., 2007. Plant architecture: a dynamic, multilevel and comprehensive approach to plant form, structure and ontogeny. *In: Annals of Botany,* Vol.1, No 33, pp. :1-33.

Brownbill, A. 1998. Reducing the storage required to render L-system based models, PhD thesis of University of Calgary. http://dspace.ucalgary.ca/bitstream/1880/29367/1/20821Brownbill.pdf Calgary, Alta., Canada, 134 p.

Deng Q, Zhang X, Gay S, Lei X. 2007. Continuous lod model of coniferous foliage. *International Journal of Virtual Reality*, 6(4): 77–84.

Deng, Q. and Zhang, X. and YANG, G. and Jaeger, M. 2010. Multiresolution foliage for forest rendering, *COMPUTER ANIMATION AND VIRTUAL WORLDS* , 20, 1, pp. 1-23

De Reffye, P. and Jaeger, M. and Edelin, C. and Françon, J. and Puech, C. 1988. Plant models faithful to botanical structure and development. *Proceedings of Computer Graphics Siggraph 1998,* Atlanta, USA, 22, p. 151-158.

De Reffye, P. and Goursat, M. and Quadrat J.P. and Hu, B., 2003. The dynamic equations of the tree morphogenesis GreenLab model. *Proceedings of Plant Modeling and Applications (PMA03)*. Beijing, Springer Tsinghua, 2003. pp. 108-117

Deussen, O. and Lintermann, B. 1997. A modelling method and user interface for creating plants. *Proceedings of Graphics Interface 97*, Morgan Kaufmann Publishers, pp. 189-197

Deussen, O. and Lintermann, B. 2004 *Digital Design of Nature: Computer Generated Plants and Organics*. SpringerVerlag.

Fernández, I.G.. 2007. *Generation and Interactive Visualization of 3D vegetation*. Master Thesis, Master in Computing. http://www.gametools.org/archives/publications/mastertesi_isma.pdf Universitat de Girona, 2007.

Jaeger, M. and De Reffye, P. 1992. Basic concepts of computer simulation of plant growth. *Journal of Biosciences,* 17, 3, p. 275-291.

Li, Z and Le Chevalier, V. and Courbède, P.H., 2010, Towards a Continuous Approach of Functional-Structural Plant Growth, *Proceedings of Plant Modeling and Applications (PMA09)*. Beijing, IEEE, CPS, pp. 334-340

Lintermann, B. and Deussen, O. 1999. Interactive structural and geometrical modeling of plants. *IEEE Computer Graphics and Applications*, 19, 1 pp. 56-65

Max, N and Deussen, O and Keating, B. 1999. Hierarchical image-based rendering using texture mapping hardware. *Proceedings of the 1999 Eurographics Workshop on Rendering,* 1999, p. 57–62.

Nevill-Manning, C.G. and Witten, I.H. and Maulsby, D.L. 1994. Compression by induction of hierarchical grammars. *Proceedings of IEEE Data Compression Conference DCC94.* Publisher: IEEE Comput. Soc. Press, Pages: 244-253

Prusinkiewicz, P. and Lindenmayer, A. 1990. *The Algorithmic Beauty of Plants.* Springer Verlag, New York, NY, USA.

Sun, R. and Jia, J. and Li, H. and Jaeger, M. 2009. Image-based lightweight tree modeling. *Proceedings of VRCAI '09*, New York, NY, USA: ACM, 2009. pp. 17-22.

Sun, R. and Jia, J. and Jaeger, M., 2009. Intelligent tree modeling based on L-system. *Proceedings of CAID and CD'2009*. IEEE Computer Society, 2009. pp. 1096-1100.

Zhao, X. and de Reffye, P. and Barthelemy, D. and Hu, B., 2003. Interactive simulation of plant architecture based on a dual-scale automaton model. *Proceedings of Plant Modeling and Applications (PMA03)*. Beijing, Springer Tsinghua, 2003. pp. 144-153.

Yan, H. and Barczi, J.F. and De Reffye, P. and Hu, B. and Jaeger, M. and Le Roux, J. Fast Algorithms of Plant Computation Based on Substructure Instances. 2002 *Proceedings of WSCG'2002 – the 10-th International Conference in Central Europe on Computer Graphics, Visualization and Computer.* Vision'2002 in cooperation with EUROGRAPHICS. Vol. 3, no. 10, 145-153

Annex

Volume to length/section conversion is performed thru following equations:

Definitions
Vp : volume of pith
Vr : volume of rings
H : internode height
Ri : internode radius
Rp : pith radius
Parameters for pith allometry are :
Href : reference length (set to 5 in our application)
*Vref : reference volume (set to P * Href so nearly 15.71)*
b: allometry parameter (set to 1 in our application)
Dimensions from volumes conversions:

$H = Href\ .\ (Vp / Vref)^{b/(1+b)}$

$Rp^2 = (Vref / P\ .\ Href)\ .\ (Vp / Vref)^{1/(1+b)}$

$Ri^2 = Rp^2 + (Vr / P\ .\ Href) = (Vref / P\ .\ Href)\ .\ (Vp / Vref)^{1/(1+b)}\ .\ (1 + Vref / Vp)$