The Visual Computer manuscript No.
(will be inserted by the editor)

Alexandre Peyrat · Olivier Terraz · Stéphane Mérillou · Eric Galin

Generating Vast Varieties of Realistic Leaves with Parametric 2Gmap L-Systems

Abstract Creating realistic plants and trees require the ability to generate thousands of leaves with different shapes and textures for different given species. This paper presents an original method to generate large atlases of leaves with many details from a single formal grammar. Leaves are described by a parameterized 2Gmap L-System that describes their evolution in shape and texture through out their entire life cycle. Our approach automatically synthesizes the deformations as well as the color and texture changes as the leaves age, as well as defects such as holes or cracks produced by insects attacks or accidents.

Keywords Leaves · Formal Grammar · L-Systems · Aging.

1 Introduction

Modeling complex and realistic synthetic landscapes covered with vegetation such as forests, meadows or gardens is a challenging and important problem in computer graphics. The challenge stems not only from the complexity and diversity of biological species that exhibit different shapes and textures, but also from the huge number of elements that need to be modeled and rendered to create realistic images. In particular leaves, which are conspicuous in nature, play a very important part in the creation of a complex scene.

The complexity and diversity of leaf patterns, shapes and textures makes modeling every single leaf by hand impossible, whereas the replication of the same graphical elements results in a very unrealistic natural scene. Thus, there is a need for procedural techniques for generating thousands of different leaves and create realistic tree foliages. In this paper, we present an efficient procedural method to generate a vast variety of different leaves with a simple and compact grammar characterization. Leaves are defined by an original parameterized 2Gmap L-System that describes the characteristics of their species and their evolution in shape and texture through out their entire life cycle. Our 2Gmap L-System builds on classical L-System [15] and 2-dimensional generalized maps (2Gmap) [9] and operates on faces rather than on segments to automatically generate a triangle mesh representing the leaves. We extend the base grammar of L-systems so that it can handle parameters and introduce inline declarations of special actions via scripting and specific case rules.

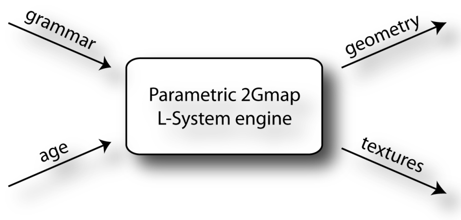

Fig. 1 Overview of the system.

Given an input set of production rules and an age parameter, our method automatically generates the venation pattern, and creates the triangle mesh and the texture of the leaf from the venation pattern as outlined in Figure 1. This approach enables us to generate complex foliages with different leaves easily. This paper is organized as follows: section 2 recalls related work on leaf generation, leaf rendering and leaf aging techniques. Section 3 presents the 2Gmap L-System and details the leaf generation process. Section 4 explains how the textures are generated to simulate aging. Finally, section 5 presents results and discusses future work.

Alexandre Peyrat
E-mail: alexandre.peyrat@xlim.fr

Olivier Terraz
E-mail: olivier.terraz@xlim.fr

Stéphane Mérillou
E-mail: stephane.merillou@xlim.fr

Eric Galin
E-mail: eric.galin@liris.cnrs.fr

2 Related Works

Leaves modeling:Many methods for leaves modeling exist: image-based modeling, particle systems, implicit contours and L-systems.

A first field for modeling leaves uses image-based modeling. Such methods propose to reproduce the real shape of the leaf provided by the user. Quan et al. proposed a semi-automatic way for modeling leaves [17]. The user provides several pictures taken at different angles, then a points cloud is built and the segmentation of individual leaves (via a graph) is done with the pictures information. Finally the user can manually refine the segmentation in order to bypass errors due to overlapping leaves. A generic and deformable leaf model is then applied.

In order to build venation patterns Rodkaew et al. in [18] use a particle system. The shape of the leaf and the base point of veins (petiole) must be given. Particles are then applied randomly on the blade of the leaf and move through it, creating a trail, to reach the petiole. Runions et al. designed a biological algorithm in order to build leaves [20]; they use the shape of the leaf, then build veins via simulation of hormones distribution inside the leave. The user must provide the shape of the leaf and the position of the petiole. Hormone sources are applied on the blade of the leaf via a dart-throwing algorithm and then migrate to the nearest vein point. When a hormone point is close enough to a vein point a new vein point is generated. This method gives biologically plausible results based on the growth of leaves. Compound and lobed leaves have also been well studied. A method designed by Hammel et al. in [8] proposed to model compound leaves with implicit contours. The skeleton, generated using an L-system, is used to build the implicit surface around the leaf. Moreover Hammel et al. can modify the radius of influence of each segment to build a more realistic contour. Pivovarov et al. in [12] designed a method to model lobed leaves via a skeleton. A 2D shape of the border of the leaf must be provided, and then a skeleton is built to fit this given shape. This method uses sticky splines [13] so that the skeleton has topological relations. Once this step has been done, the final mesh is built by scanning in the same way the skeleton and the shape.

One of the most noticeable contributions in plants modeling are L-systems which are commonly used. These methods are very efficient to design plants organs and growing structures. Prusinkiewicz and Lindenmayer in [15] describes a large amount of methods based on L-systems. Many improvements have been proposed to increase realism and flexibility of L-systems. Prusinkiewicz and Hanan [14], build parametric L-system to handle numerical parameters. Another way to build branching structures by fitting them to a given shape is proposed by Rodkaew et al. [19] based on genetic algorithm. In order to make plants grow depending on the later steps, Prusinkiewicz and Lindenmayer [15] design context-sensitive L-systems.

String L-systems are of 1 topological dimension even if 3D shapes can be used instead of segments ([16] and [7]). But many botanical structures need 2 or 3 topological dimensions to be modeled. Lindenmayer and Prusinkiewicz in [15] introduced Map L-systems and Cellwork L-systems. Map L-systems were introduced to work on subdivisions of surfaces such as cellular structures. A map is a set of regions constituted of borders made of segments. In the first phase, Map L-systems build segments and in the second one, they create regions with these segments. Cellwork L-systems work on 3 topological dimensions. These methods give realistic results but induce a heavy and hard to built grammar. Extension of L-systems based on 3-dimensional generalized maps that allows an easier control of the internal structure of 3D objects is proposed in [21].

In this paper, we introduce a new kind of L-system, which contrary to the models previously developed makes it possible to easily describe operations on subdivisions of surfaces. The direct use of high level operations on surfaces makes it possible to control more intuitively and more simply the evolution process. The main drawback of image-based and particles system methods is that the user must provide a picture or a shape of the leaves. Our method is free from pictures or shape input, the shape of the leaf is totally defined via rules in the L-system grammar.

Leaves rendering:Leaves are composed of several layers that behave differently when exposed to light. Bousquet et al. have studied reflectance and transmittance of leaves [3]. The authors provide several BRDF and BTDF parameters suited to leaves rendering. A method proposed by Rodkaew et al. [18] generates colors using particles previously used to also generate the veins of the leaf. A vector field is modified by moving particles and is used to fill the blade of the leaf with colors via the petiole.

Subsurface scattering is also well-suited to render leaves, due to their multiple-layered structure. Baranoski et al. [1] have proposed an "Algorithmic BSSDF Model" (ABM) where BSSDF stands for "bidirectional subsurface scattering distribution function". The ABM model consists in four layers that can absorb, transmit, or reflect the ray of light via Snell-Descartes laws. Baranoski et al. [2] also proposed an improvement of the ABM. They assume that the surface of the plant is not totally flat, thus they apply a perturbation on the rays of light. These perturbations depend on the layer crossed by the ray as well as the layer the ray comes from. Wang et al. in [22] proposed a method to render leaves with real time subsurface scattering based on a parametric leaf model. This model is a slab with rough surfaces and uses spatially-variant BRDFs and BTDFs. It also contains thickness variations and an albedo map. Subsurface propagation is insured by an experimentally validated model.

Leaves aging:Aging has been mainly studied via the concentration of pigments. In [5], Chiba et al. have studied colors of leaves getting old. The authors determine pigment concentrations against time, distance between veins and the petiole, and light. Mochizuki et al [11] also studied colors of leaves with a method accounting for solar light and experimental information on the coloration of leaves. They propose both biological and fractal models for autumn leaves. In [4], Braitmaier et al. have also proposed to characterize colors of leaves with pigments concentrations against solar light and pigments. A stochastic model, proposed by Desbenoit et al. in [6], uses an atlas of leaves built from scanned images to distribute them in a scene.

3 Leaves Generation

Leaves consist in an almost flat structure, so a 2 topological dimension structure is well suited to represent them. Our method is based on a 2Gmap L-system which builds up a 2 dimensional generalized map including operations associated with production rules. The 2Gmap is a topological model which allows to represent the topology of any 2 dimensional subdivision. This model uses a single type of element: Dart (which can be seen as half-edges) on which 3 bijections are defined: connect 2 darts to form edges, connect edges to form faces and finally sew a face to another. However, technical aspects of the implementation of generalized maps will not be describe here, all topological operations used bellow (creation, face glueing and face splitting) are traditional and formally defined in the corresponding literature (for more details readers can refer to [9]). In our method, the formal L-system grammar is used to build faces which represent veins of the leaf; this grammar has been enhanced in order to accurately design venation patterns. Then the leaf body is filled through the iteration process. Moreover, tools are added to the grammar to make designing rules simpler and permit to control the evolution more precisely. Our main goal is to be able to generate a collection of different leaves for a given species with the same grammar. The user can build a grammar using variables and random values, this grammar will give different leaves at the end of the iterative process.

3.1 2Gmap L-system

In our model, faces will be characterized by a label and will be referred to as an upper case letter. Each edge has an automatically assigned label (figure 2). E edge represents the extremity of a face, O edge represents its origin and C_x edges represent its side edges. We use the following notation : A_{C_1} for a C_1 edge of a A face. The grammar consists in the description of faces used, an axiom, definitions of variables used and finally production rules.

Faces are defined with their number of edges N, their translation coefficient t, three angles defining their orientation, denoted as α, β, γ, length l and width w. The axiom of the grammar must be specified to determine the first face where other faces will be glued. Finally the list of rules is given. The 2GMap L-system rules will shape faces rather than build segments or word as in classical L-systems. Our system is built on three production rules: growing, glueing, splitting.

Growing:The first base rule is the growing operation. It creates a new face and glues it on another one using a given edge label: $A \rightarrow A[B]_E$. Here a face B is created and glued on A_E (figure 2). New faces are always glued using their O edge.

Glueing:Labels provide the user with the ability to glue one edge with a given label to another one. Glueing can be applied to two existing edges. This operation consists in glueing two adjacent edges: $A : A_{C_1} < B_{C_2} \rightarrow A_{C1}|B_{C2}$. In this rule, if an edge A_{C_1} is adjacent to an edge B_{C_2}, then these two edges are glued (figure 2). This also allows us to introduce a context-sensitive aspect on our rules.

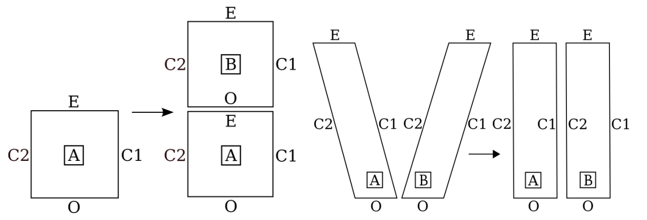

Fig. 2 On the left, face B grows from edge A_E. On the right, if A_{C_1} is adjacent to B_{C_2}, they are glued.

Splitting:A face can be split in two other faces. The face is cut parallel to the desired edge $A \rightarrow {}_O BC$. Here A is split into B and C parallel to A_O (figure 3).

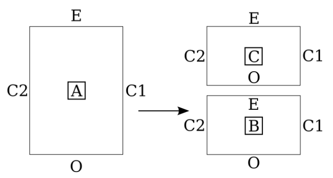

Fig. 3 Face A is divided into face B and C parallel to A_O.

3.2 2Gmap parametric L-system

We extend the previously described 2Gmap L-systems by introducing new features: scripting inside the grammar, overloading faces attributes, declaring conditional actions and reusing attributes of faces.

Scripting system:Variable can be initialised and can then be used in the rules. In order to use a variable, we introduce scripting blocks providing the ability to get values from arithmetical operations on variables.

Overloading faces attributes: To enhance control on faces we include three tools which allow us to build a more precise structure. Our first improvement is the ability to modify faces attributes from inside the rules. This overload of face attributes is directly used when we create a new face; all the attributes can be defined with values different from the ones used in the face definition.

Grammar 1

```
@Faces, variables and axiom definition :@
#define A(4,1,0,0,−90,1,1)
#define B(4,1,0,0,0,1,1)
#define C(4,1,0,0,0,1,1)
#define nl = 2
#define nw = 1

#axiome : A

@Rules :@
p01 A → A[B(,,,,,2,)]_E
p02 B → B[C(,,,,,< nl >,)]_C1
p03 C {nw = nw + 1} {step > finalstep} {nl = nl + 1}
→           C[B(,,,,,< nl >,< nw >)]_C1
p04 B(nbe,t,rotx,roty,rotz,l,w) { } {l < 10} {nl = nl + l}
→           B[C(,,,,,< nl >,)]_C1
```

We can see in (*Grammar* 1 : *p*01) how overloading is conducted. *B* length is changed from 1 to 2. Every parameter, presented in the face definition, can be altered. Moreover a scripting block can be set in place of an attribute so that variables and functions can be used in the overloading (between < and >) as shown in (*Grammar* 1 : *p*02).

Conditional actions: We introduce three blocks, the first one contains an action done every time, the second one contains a conditional rule and the last one contains an action done only if the condition is met. These blocks are similar to the ones used in CPFG syntax ([10]). This improvement is shown in (*Grammar* 1 : *p*03). If *step* is greater than *finalstep* then we increment *nl* and the rule is applied. Regardless of the condition satisfaction, we increment *nw*. We can also see two undeclared global variables used to know current and final step of the iteration.

Reusing attributes of faces: Our last improvement in the rules is the possibility to reuse attribute of the mother face. In order to get the attributes of the face, we have to specify names for these attributes as shown in (*Grammar* 1 : *p*04). If the length *l* of the mother face is lower than 10 then *newlength* is increased by *l* and applied to the glued face.

3.3 Leave generation process

All this rules and enhancements allow us to build the faces structure used to simulate veins. Modifying variables and random values allows building a collection of different leaves from the same species with only one grammar. Nevertheless, random values must be defined in a range fitting attributes of the species. For example veins with an angle of 45 degrees may be defined between 40 and 50 degrees in order to respect the shape of the species. The number of steps allows building leaves with various levels of details in order to have a lot of polygons in the foreground and less in the background. However, the leaf body is empty and we must now build faces between veins to simulate the limb.

During the iteration process, our method fills holes between veins by glueing faces. Building the limb with rules inside the grammar can be done but leads to heavy and complex grammars. Therefore, we proposed to fill in the limb of the leaf with an automatic process.

This filling is done at each step of the iteration, when faces have been glued and/or divided. Firstly we find an extremity (free edges with label "E") and then we travel through the border of the veins from one extremity to another using the 2Gmap topology. While doing so we save each found edge to build a new face and we topologically sew them to the border. The new face is then completed with an edge between the two free extremities.

Figure 4 shows result of leaves generation for a given species with only one grammar.

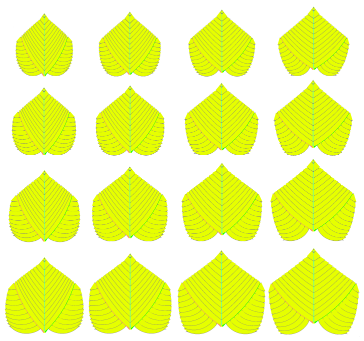

Fig. 4 This figure shows a family of leaves from the same species (lime tree) generated from the same grammar. We can see from left to right an increase in veins angles and from top to bottom an increase of the veins length. These parameters can be used randomly inside the grammar to generate the intra-class variation.

We are now able to generate efficiently a large amount of different leaves for a given species. These leaves can be finely designed and thus fit the real leaves shapes. Therefore, a natural scene can be filled with leaves using only one grammar by species. We also have veins and body information embedded in our faces structure.

The user has the possibility to design leaves at chosen steps of the iteration, this feature enables to generate leaves with low details (figure 5). Therefore, scenes can be generated with level of detail for shorter rendering times.

In order to have realistic looking leaves we need to apply colors and deformations on them. Moreover, we need to build textures that will fit the leaf and take in account veins information. In addition these textures must be generated to present the leaf at a given time of its life cycle.

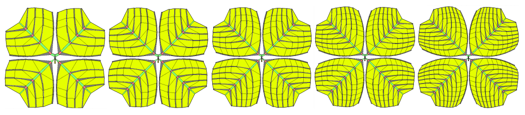

Fig. 5 A clover leaf at step 4, 5, 6, 7 and 8 of the iteration. Low level of detail will be used in the background and high level of detail in the foreground. Rendering engines which use our method to generate leaves can adapt the level of detail for each leaf according to the camera.

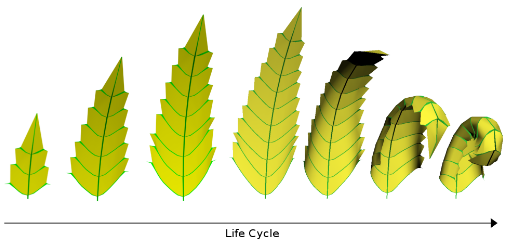

Fig. 6 This figure present the life-cycle of an European chestnut leaf. Angles and length are defined in the grammar by the user in order to give the different aspects of the leaf.

4 Textures generation and aging simulation

When leaves get old, their pigment concentrations roughly change. The color of the leaf changes from green to red as the concentration in chlorophyll diminishes. In addition, leaves shapes are deformed through time due to their drying (this process is even accelerated after leaves have fallen off). In our method we handle both these geometric and color perturbations. We also introduce damage on leaves body from insects attack. Information given by the faces are essential because they permit to change colors in a botanic-inspired way and to damage the leaf body while preserving the veins structure.

4.1 Geometric aspect of aging

We propose to characterize the age of leaves using the following characteristics, that are parameters of the grammar:

- The number of veins the leaf contains. This specific point is a rough approximation of the biological growth process as real primary veins are all created inside the bud. However, the controls provided by our grammar permits to concentrate on leaves shape by using only a few relevant veins. Thus, we empirically assimilate young leaves as leaves containing only a few veins. This parameter is directly controlled by the number of iterations.
- The mean length of these veins. The growth process of leaves makes primary veins length increase to reach the adult leaves size. This parameter is function of both the number of iterations and the face length l (see 3.2);
- The rotation of each vein. From their adult state, leaves start to wither and deteriorate. This aging cycle implies a shrinking process. After a certain amount of time (that depends on environmental factors such as moisture), the whole leaf curls up. Veins are not made of the same material as the leaf body. Therefore, these changes in shape follow the veins structure: the curling process occurs around the main veins. Vein rotation is a function of the previously defined angles α (twisting angle), β (curvature), γ (curling angle).

These characteristics permit to handle the full life-cycle of leaves: young leaves have a few short veins, adult leaves have the maximum number of veins (corresponding to their species), and old leaves are deformed, as illustrated on figure 6.

4.2 Colours generation and aging

Another aspect of leaves aging is the colors modification. Leaves are known to be mainly green, however, in autumn they will have a brown or red color. These changes are caused by the decomposition of chlorophyll which makes leaves look green. The other pigments are red anthocyanins and yellow or orange carotenoids. They protect the leaf from light at lower temperatures while chlorophyll is decomposed. Chlorophyll quickly disappears near the veins because of the flowing sap but disappears slowly inside the body. Nevertheless it can happen in random regions of the leaf.

Our method is based on a few parameters and user's inputs. We used:

- Two colors provided by the user. These colors are used to define the old and the young state in the color aging process. Specifying a young and an old color permits to generate any color of leaves at any time of their life. Moreover, the user can set colors depending on the species he wants to generate;
- A blending parameter depending on the desired age of the leaf. This parameter is used to determine the amount of blending between the young and the old color provided;
- Distance to the veins. The age of the color is directly linked to this distance as the sap in the veins slowly migrates into the tree. We can see on figure 7 how the distance acts to blend provided colors;
- Blending masks. These masks are automatically computed during generation and are applied during the rendering phase. They permit to produce more complex colors via mixing multiple colors in order to have different local ages inside the leaf (as shown on figures 7 and 11).

In order to simplify the use of the previous parameters, we use a variable *age* that acts as an input to simple functions that computes gemoetrical and colors aging. This variable is used to determine angles, length (inside the grammar) and the blending parameter.

4.3 Leaves damage

During their life leaves may experience some damage to their tissues. Insects, diseases, parasites, all these problems

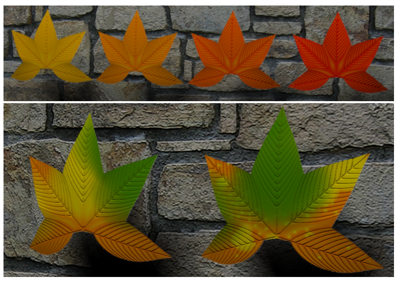

Fig. 7 Evolution of autumn's colors. We can see the transition between younger (right) and older (left) leaves. On the second line, masks are used to mix generated color textures in order to have local variation in the colors aging process.

have repercussion on the appearance of the leaf. With the numerous nutrient inside the leaf body, caterpillars and more generally larva attack leaves (figure 8). However they avoid eating veins because of the lack of nutrients and the hard fibre they are made of.

We produce an alpha channel texture to make holes; this texture is calculated with random values in order to have realistic holes as larva spread out on the leaf and eat at their current positions. Holes parameters are computed with three parameters :

- The radius of the hole which is relevant of the size of the larva;
- A direction where the larva will next move to eat the leaf;
- The number of time the larva will eat the leaf;

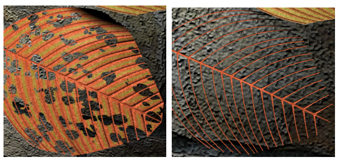

Fig. 8 Insects eat the leaf body and avoid veins. Leaves can be more or less damaged by insects.

The user can define extreme for all these values so that he can influence size of the holes. In figure 8 we can see an old leaf which has been eaten by several insects and one totally eaten so that only veins remains (figure 8).

5 Results

The output from our method gives a satisfying mesh and several textures in order to render leaves. Leaves from a same species can be also generated with the same grammar and will produce a realistic family. Figure 9 shows a CG image generated by applying textures of colors and veins on our output meshes. This figure shows two models of plants, one with the same shapes of leaves and one with all different leaves. The right plant looks more realistic because of the

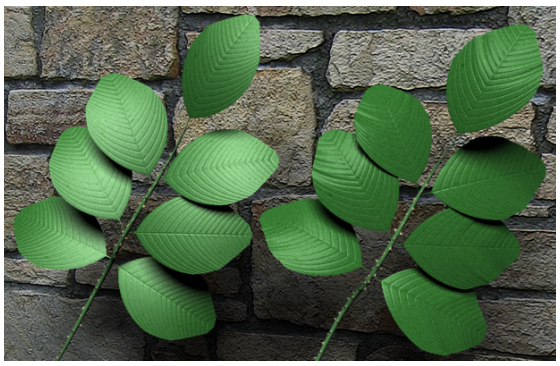

Fig. 9 Young, healthy leaves of rose tree. The left leaves have all the same shape, the right ones have all a different shape.

differences between leaves. The left plant looks less realistic because of the ”perfection“ of its leaves.

Moreover, our method permit to build leaves at various ages. Figure 10 shows the aging simulation of a rose tree through time. We can see from the left to the right (clockwise) different steps in leaves colors. This figure also shows insect attacks which can have any intensity through time and can be light or heavy. All these details added to the leaf increase realism.

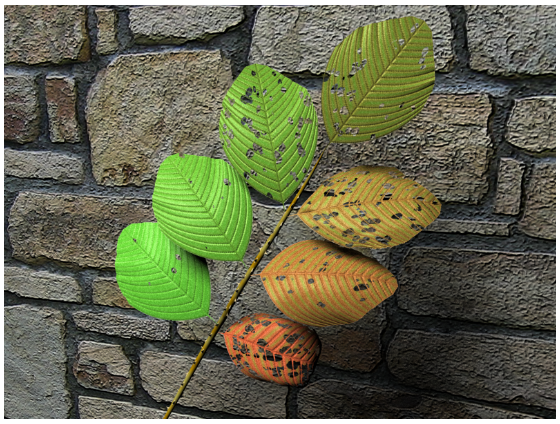

Fig. 10 Degradation of leaves from a rose tree. From left to right, leaves are affected by color changes, insects attacks, and geometry shrinking.

We can see in figure 11 a table showing various leaves and their representation with our method. The first column shows a picture of the desired leaf, the second column shows the output faces generated by the grammar, the third column contains the output mesh, alpha channel texture, bump map, young/old colors and the blending mask. Finally the last column show rendered leaves within a 3D engine.

Variety of leaves is a key component of realistic scenes. In figure 12 we can see the same scene with diminishing number of different leaves. The left picture is obtained with a set of 40 different leaves, the one in the middle has a set of 10 different leaves and the right one has only 2 different

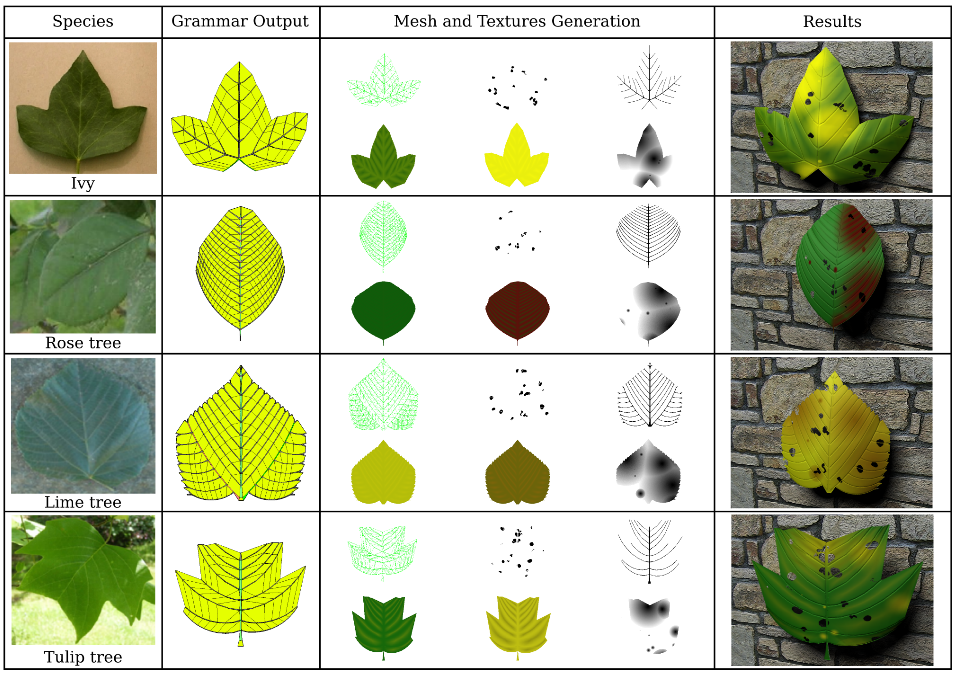

Fig. 11 Results of our method with various leaves. Grammars are easily written once operations on faces are mastered, moreover generic grammar depending on the number of main veins can be used.

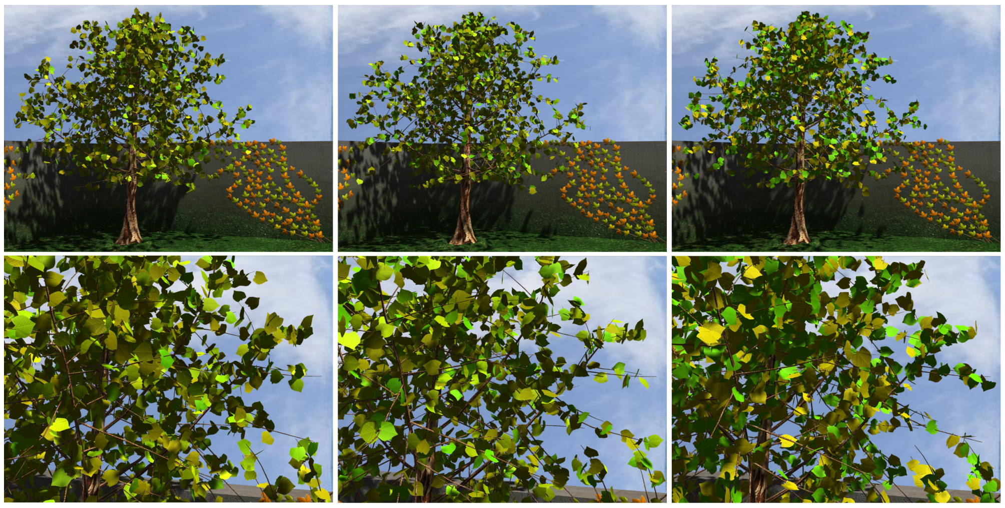

Fig. 12 Variety results. This scenes have (from left to right) 40, 10 and 2 different leaves.

leaves. Realism is increased in the first picture because of the large number of different leaves.

Figure 13 shows a realistic scene populated with leaves. The lime tree and ivy are shown at four ages of their lives, the first picture represents spring and summer, the second one represents autumn and the two next pictures represent the aging process through winter. Moreover, we used leaves with less details than these shown in figure 11 because of the large number of leaves in the scene (over 6500 leaves for the whole scene), we used the level of detail aspect of our method in order to have leaves with less faces. These pictures show that even if there are fewer details for leaves, the main shape is respected and gives a realistic result depending on the distance of the point of view.

6 Conclusion and perspectives.

We proposed a method to model leaves and then to handle aging and damage aspects. This method based on a 2Gmap

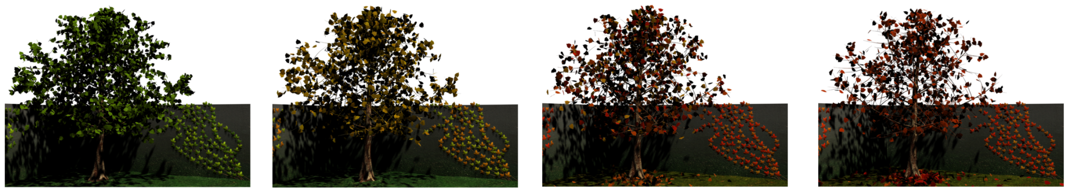

Fig. 13 A realistic scene through seasons.

L-system offers a powerful and easy to create grammar in order to build leaves veins. The grammar is also very efficient to precisely design the shape of the leaf; scripting blocks and mathematical functions give a great field of shapes and structures (examples of grammars are available on-line[1]). We also provide the user with the possibility to render leaves within any rendering program by exporting the textures and the generated mesh. A key feature of our method is the ability to create a large collection of leaves from the same species at any age from a single grammar. This allows the generation of large scenes composed of a large number of leaves with a minimum work on the grammar. One limitation of the current method is that only two colors are specified to generate textures. A possible improvement consists in giving a texture describing colors of the leaf instead and to age it with veins information and global pigments concentration. Another functionality yet to be implemented is to give the ability to generate a texture with fine veins; this will be useful to render more realistic leaves without including high level veins in the grammar. For degradation purpose, diseases should be a great improvement to geometrical deformation and colors. Finally leaves generation may be simplified by creating a grammar designer where the grammar is automatically built from some given data. The number of iterations needed to have a good level of detail may also be defined automatically so that leaves can be generated on the fly during the rendering of the scene. Regarding our model, all these additional features can be applied to it.

Acknowledgements This research is supported by the French National Agency for Research (ANR) under agreement ANR-06-MDCA-004-01.

References

1. Baranoski, G.V.G., Rokne, J.G.: An algorithmic reflectance and transmittance model for plant tissue. Computer Graphics Forum **16**, 141–150 (1997)
2. Baranoski, G.V.G., Roknef, J.G.: Efficiently simulating scattering of light by leaves. In: The Visual Computer, vol. 17, pp. 491–505 (2001)
3. Bousquet, L., Lavergne, T., Deroin, T., Widlowski, J., Moya, I., Jacquemoud, S.: Multispectral and multiangular measurement and modeling of leaf reflectance and transmittance. Second international symposium on Recent Advances in Quantitative Remote Sensing (RAQRS 2) (2006)
4. Braitmaier, M., Diepstraten, J., Ertl, T.: Real-time rendering of seasonal influenced trees. In: Procceedings of Theory and Practice of Computer Graphics 2004, pp. 152–159. Eurographics UK (2004)
5. Chiba, N., Ohshida, K., Muroaka, K., Saito, N.: Visual simulation of leaf arrangement and autumn colors. The Journal of Visualization and Computer Animation **7**, 79–93 (1996)
6. Desbenoit, B., Galin, E., Akkouche, S., Grosjean, J.: Modeling autumn sceneries. In: 26th International Conference on Eurographics, pp. 107–110. Eurographics Association (2006)
7. Fuhrer, M., Jensen, H., Prusinkiewicz, P.: Modeling hairy plants. Computer Graphics and Applications. 12th Pacific Conference on pp. 217–225 (2004)
8. Hammel, M., Prusinkiewicz, P., Wyvill, B.: Modelling compound leaves using implicit contours. Proceedings of Computer Graphics International '92 pp. 119–212 (1992)
9. Lienhardt, P.: N-dimensional generalized combinatorial maps and cellular quasi-manifolds. International Journal on Computational Geometry and Applications **4**(3), 275–324 (1994)
10. Mech, R.: CPFG version 3.4 user's manual. Department of Computer Science, University of Calgary (1998)
11. Mochizuki, S., Cai, D., Komiri, T., Kimura, H., Hori, R.: Virtual autumn coloring system based on biological and fractal model. In: Proceedings of the 9th Pacific Conference on Computer Graphics and Applications, p. 348. IEEE Computer Society (2001)
12. Mundermann, L., MacMurchy, P., Pivovarov, J., Prusinkiewicz, P.: Modeling lobed leaves. In: CGI '03: Proceedings of Computer Graphics International, pp. 60–65 (2003)
13. van Overveld, C.W.A.M., Viaud, M.L.: Sticky splines: definition and manipulation of spline structures with maintained topological relations. ACM Trans. Graph. **15**(1), 72–98 (1996)
14. Prusinkiewicz, P., Hanan, J.: Visualization of botanical structures and processes using parametric l-systems. In: S. Visualization, G. Simulation (eds.) D. Thalmann, pp. 183–201. J. Wiley and Sons (1990)
15. Prusinkiewicz, P., Lindenmayer, A.: The algorithmic beauty of plants. Springer-Verlag New York, Inc. (1996)
16. Prusinkiewicz, P., Mundermann, L., Karwowski, R., Lane, B.: The use of positional information in the modeling of plants. Proceedings of SIGGRAPH'01 pp. 289–300 (2001)
17. Quan, L., Tan, P., Zeng, G., Yuan, L., Wang, J., Kang, S.B.: Image-based plant modeling. In: SIGGRAPH '06: ACM SIGGRAPH 2006 Papers, pp. 599–604 (2006)
18. Rodkaew, Y., Chongstitvatana, P., Siripant, S., Lursinsap, C.: Modeling plant leaves in marble-patterned colours with particle transportation system. In: 4th International Workshop on Functional-Structural Plant Models, pp. 391–397. C. Godin et al. (2004)
19. Rodkaew, Y., Lursinsap, C., Fujimoto, T., Siripant, S.: Modeling leaf shapes using l-systems and genetic algorithms (2002)
20. Runions, A., Fuhrer, M., Lane, B., Federl, P., Rolland-Lagan, A., Prusinkiewicz, P.: Modeling and visualization of leaf venation patterns. In: SIGGRAPH '05: ACM SIGGRAPH 2005 Papers, pp. 702–711 (2005)
21. Terraz, O., Guimberteau, G., Mérillou, S., Plemenos, D., Ghazanfarpour, D.: 3gmap l-systems: An application to the modeling of wood. The Visual Computer **To appear** (2008)
22. Wang, L., Wang, W., Dorsey, J., Yang, X., Guo, B., Shum, H.: Real-time rendering of plant leaves. pp. 712–719. ACM Transactions on Graphics (TOG) (2005)

[1] `http://www.msi.unilim.fr/~peyrat/GreenLeaves/`