



XVIIth World Congress of the International Commission of Agricultural and Biosystems Engineering (CIGR)

Hosted by the Canadian Society for Bioengineering (CSBE/SCGAB)
Québec City, Canada June 13-17, 2010



STRUCTURAL PLANT MODELLING BASED ON REAL 3D STRUCTURAL PARAMETERS, RESULTING SIMULATION SYSTEM AND RULE-BASED LANGUAGE XL

RENTAO HE¹, JIANGPING HU¹, YONG HE¹, HUI FANG¹

¹ Rentao He, College of Biosystems Engineering and Food Science, Zhejiang University, 268 Kaixuan Road, Hangzhou, 310029, China, lanina@zju.edu.cn

³ Y. He, yhe@zju.edu.cn

⁴ H. Fang, newxfh@yahoo.com.cn

CSBE101457 - Presented at the 8th World Congress on Computers in Agriculture (WCCA) Symposium

ABSTRACT This paper presents a new method for producing a structural plant static model that simulates the details of the structure of individual plants, yet does not require detailed knowledge of underlying physiology. The example presented with this method in this study is the modelling of the tomato plant. This new structural modelling approach is based on linking a resulting simulation system representation of the static structure of the plant with real structural parameters and rule-based language XL (extended L-system language). The static model is built by applying the structural parameters of a tomato plant which is gained by using a 3D scanner, yet iteration is not under consideration in this modelling approach. The main advantage of the approach is that the 3D reflects the real structure of plant. This approach, combining real structural parameters, resulting simulation system and rule-based language XL, fills the important role of providing an intermediate level of representation between the two extremes of 3D dynamic modelling and purely 2D empirical modelling.

Keywords: simulation, 3D structural plant modelling, GroIMP, extended L-System language (XL), tomato plant

INTRODUCTION The recent rapid development of computer technology has made it feasible to simulate the structural development of plants as individuals made up of components like apices, internodes, and leaves (Room *et al.*, 1994). When these simulations run, the behaviour of a plant's apical meristems and their products are reproduced in the computer. The result 'virtual plant' captures the changes that occur in the arrangement of component parts as the plant develops or grows.

The representation of plant forms in computer scenes has long been recognized as a difficult problem in computer graphics applications. In the last two decades, several algorithms and software platforms have been proposed to solve this problem with a continuously improving level of efficiency (Prusinkiewicz *et al.*, 1988; De Reffye *et al.*, 1988; Prusinkiewicz and Lindenmayer, 1990; Weber and Penn, 1995; Lintermann and Deussen, 1999). Due to the increasing use of computer models in biological research, the design of 3D geometric models of plants has also become an important aspect of various

biological applications in plant science (Cescatti, 1997; Sinoquet *et al.*, 1997; Godin, 2000; Danjon *et al.*, 1999; Evers *et al.*, 2005). These applications raise specific problems that derive from the need to represent plants with a botanical or geometric accuracy at different scales, from tissues to plant communities. However, in comparison with computer graphics applications, less effort has been devoted to the development of geometric modelling.

It is very difficult to reveal the mechanistic details of the physiology needed for a full process-based structural model, since all plant species need to be modelled. A model including physiological knowledge is complex, and a 3D geometric model may be prerequisite for physiological model.

In this context, the most successful and widespread plant modelling system has been developed by P. Prusinkiewicz and his team since the late 80's at the interface between biology and computer graphics. They created a computer platform, known as L-Studio/VLab, for the modelling of plant growth based on L-systems. This system makes it possible to simulate the development of plants with efficiency and flexibility as a process of bracketed-string rewriting. In a recent version of L-Studio/VLab, Karwowski and Prusinkiewicz changed the original cpfg language for a compiled language, L+C, built on the top of the C++ programming language. The resulting gain of expressiveness facilitates the specification of complex plant models in L+C (Prusinkiewicz *et al.*, 2007). Farther, the programming languages L+C and XL (extended L-system) were developed to enable a more flexible integration of both modelling paradigms. Particularly, XL combines rule-based replacement with the object-oriented, platform-independent language Java. Based on relational graph growth (RGG) (Kniemeyer *et al.*, 2004; Buck-Sorlin *et al.*, 2005), the programming language XL and the open-source modelling environment GroIMP have been designed. XL extends the standard programming language Java and implements the formalism of RGGs. The resulting simulation system, GroIMP, is an open-source software that extends the chain rewriting principle of L-Systems to general graph rewriting with RGG. Similarly to L+C, this system has been defined on top of a widely used programming language (here Java). The platform GroIMP makes the modelling process accuracy, conciseness and transparency and absolutely object-oriented, rule-based. This paper presents the rule-based language XL and its application to plant modelling. The platform GroIMP was chosen to create 3D structural tomato modelling based on real parameters.

MATERIALS AND METHODS

Preparation of tomato plants In this study, we chose tomato plant as our initial modelling subject, 5 plants which grew in flowerpots were placed in a green house where ventilation was eligible enough and temperature was also strictly controlled with $25\pm 1^{\circ}\text{C}$. These plants were termly irrigated by water and nutrient fluid from seedling stage in order to assure that plants can grow healthily.

The rule-based language XL The programming language XL is an extension of Java, which supports the specification and execution of relational growth grammars, a variant of parallel graph grammars. Like L-systems, relational growth grammars are just a formalism, not a concrete programming language. Relational growth grammars are a recent accomplishment to address the needs of structural plant modelling. They extend the proven L-system formalism that has been widely used for plant modelling. The concrete programming language XL is a complete extension of Java within which

relational growth grammars can be specified. The general syntax of L-system rules is retained in the simple rule:

$$A(x) \implies F(x)[RU(60)A(x*0.6)]RH(90)A(x*0.8);$$

In this formula, $A(x)$ is a user-defined module with one parameter x , $F(x)$ is a cylinder of length x , RU and RH specify a rotation around the up and head axis of the turtle with the amount of rotation in degrees specified in parentheses, and the brackets define a branch; a complete description of the available commands is available in the API documentation installed with the software GroIMP. However, more complex rules that make use of true graphs and arbitrary context (not just left and right-hand side of the predecessor like in L-systems) can be specified. In fact, the definition of XL is quite common and covers not only relational growth grammars, but also allows for rule-based implementation of other graph rewriting formalisms, such as vertex–vertex algebras (Smith et al., 2003; Knemeyer, 2008), for the modelling of the growth of surfaces (e.g. an apical meristem). In addition to L-system-like rules, other kinds of rules can be specified by replacing the operator \implies with another derivation operator. Currently, XL provides the operator $\implies\implies$ for general graph replacement rules (in which no automatic embedding is carried out as is the case for \implies), and the operator $::\implies$ for specification of execution rules (in which no structural changes are made to the graph).

Within XL, the current structure is represented as a graph whose nodes are Java objects and whose edges show specific relationships. Nodes generalise the symbols of L-systems and edges generalise the adjacency of symbols in an L-system string.

The GroIMP software provides a set of standard geometric node levels whose instances play the role of turtle commands of L-systems, but are now conceptually geometry nodes of a 3D scene graph.

The 3D laser scanner FastSCAN FastSCAN is the ultimate laser scanner. With a simple sweep of the FastSCAN wand, you can create instant real time 3D points cloud and databases--anytime, anywhere. FastSCAN is lightweight and ultra-portable. The entire system is supplied in a briefcase, and setup only takes a few minutes. As a result, you can readily scan objects onsite, in their natural environments. The first laser scanner you can easily travel with, FastSCAN sets up almost anywhere, indoor or out. FastSCAN instantly acquires three dimensional surface images when you sweep the handheld laser scanning wand over an object, in a manner similar to spray painting. FastSCAN works by projecting a fan of laser light on the object while the camera views the laser to record cross-sectional depth profiles. The object's image immediately appears on your computer screen. Because FastSCAN provides real-time visual feedback, monitoring and controlling the scan process is straightforward. Unlike other scanners, FastSCAN automatically stitches your scans together, saving a great deal of time. The sweeps list enables turning individual sweeps on and off to facilitate optimizing the amount of data in your final output. However, strong magnetic field, bright light and metals have a great influence on the result of scanning. Thus a dark frame house which has little magnetic field was chose to scan the plant tomato of different periods during the experiments. Then, the point clouds of every plant can be obtained.

Data processing software Rapidform The software needed to process 3D scan data is often more crucial than the hardware that collects the data, and 3D scanning software has

traditionally been too difficult to use or too simplistic to be effective. Rapidform is a high quality, sophisticated and user-friendly software that unlock the true potential of 3D scanning.

Some useful functions for data processing, such as filter noise, filter redundancy, smooth points, thin points, were applied to make point clouds more clear. Moreover, Rapidform admirably triangulate point cloud to 3D surface hence the parameters we need including area of leaf, height of stem and angle between branches can be easily measured.

The modelling environment GroIMP GroIMP (Growth Grammar-related Interactive Modelling Platform) is a 3D-modelling platform.

GroIMP distinguishes itself from other modelling software by the modelling potential of growth grammars. This potential is made accessible by the integration of the modelling language XL.

Two screenshots of the GroIMP application window are shown in the Fig. 1.

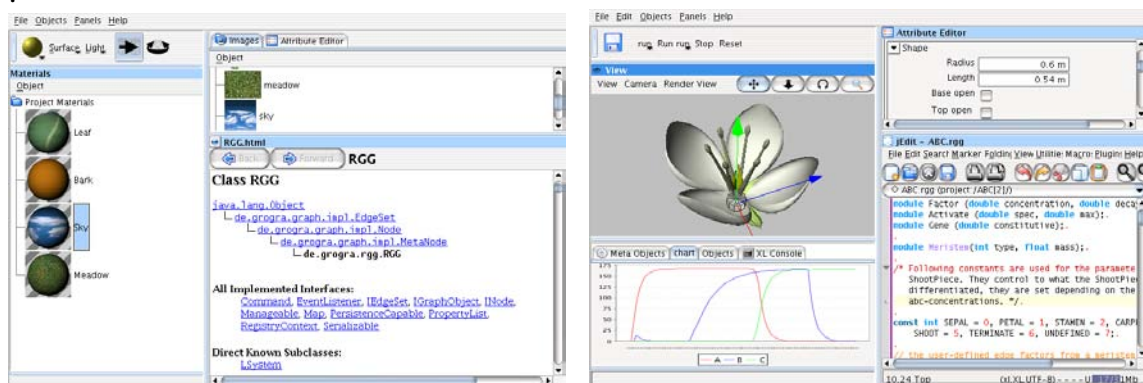


Figure 1. These two screenshots show GroIMP as a modelling platform for growth grammars.

The growth grammar model can be created in an integrated text editor, its three-dimensional outcome is visualized in GroIMP's 3D view. Interaction with the model is possible via the attribute editor or 3D tools.

According as the structural parameters that already measured, the corresponding static model can be created in this modelling environment.

MODEL CONSTRUCTION This model is a 3D structural static model of tomato plants that based on real parameters hence it can reflect the growth tendency factually. It would allow for its use to precisely predict tomato growth. Rather, this model illustrates some important novel features of XL and GroIMP, while using the basic principles of L-systems and extended L-systems.

Iteration is not under consideration in the simulation, thus each plant was divided into several parts in order to simplify corresponding program. Fig.4 shows how to divide the plant.

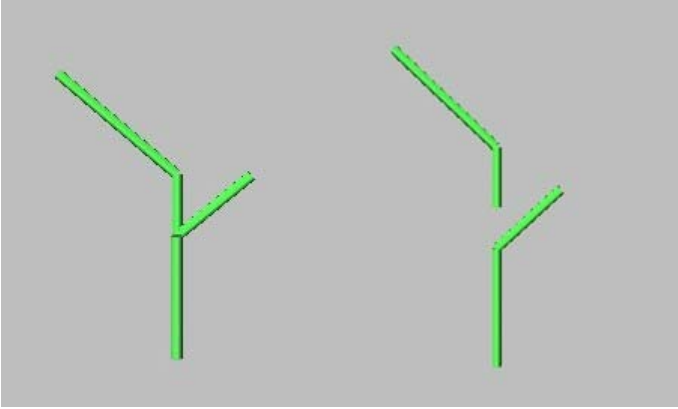


Figure 2. A length of stem and its conterminous branch were divided into one part.

Entities of the model We used two specific classes for the entities of the tomato plant model. The First class is the stem which should be divided into several parts. We use the method that we have explained above. The stem was simulated as a cylinder with a final length as well as its initial and final radius. The program of each part is similar:

```
Vertex(0.025)M(0.48f)RH(137.5f)Vertex(0.015)Surface(newCircle())
(setShader(branchmat))
// Main stem.
[Mark Vertex(0.011)RU(47)RV(0.1)M(0.12f)Vertex(0.008)for
((1:5))(RV(0.1)M(0.03f)Vertex(0.008))Surface(newCircle()).(setShader(branchmat))]
//Branch.
```

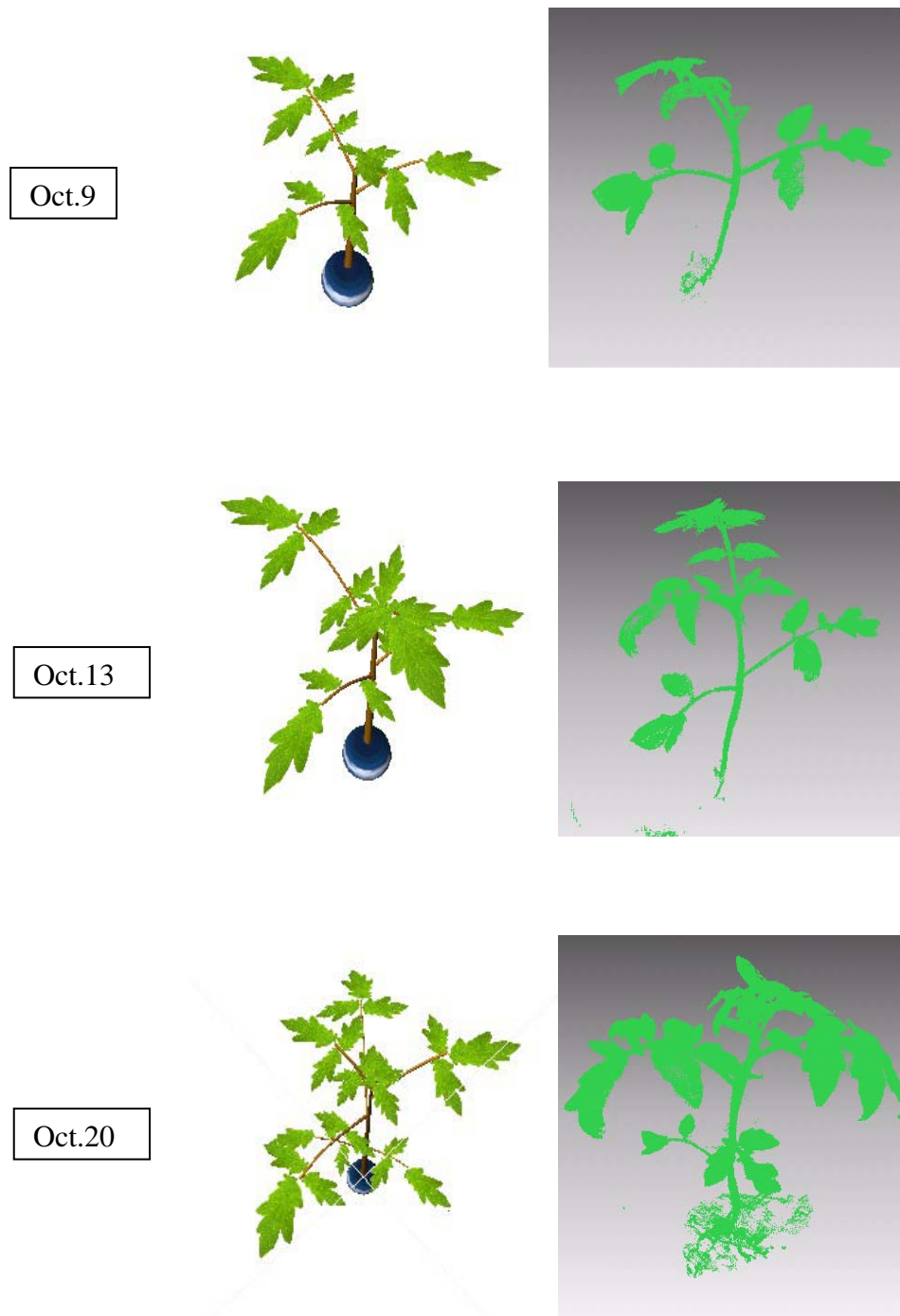
As indicated in foregoing code, Vertex (r) is the symbol of drawing vector, r represents the radius. M (l) means moving a distance of l . Thus Vertex (0.025) M (0.48f) Vertex (0.015) means drawing a vector with the length of 0.48, and its initial and final radiuses are 0.25 and 0.15, respectively. RH (r) represents the turned angle on the plane which is perpendicular to main stem. RU (r) represents the angle to the plane which is parallel to main stem. RV (r) controls the curvature of the stem. RH (r) and RU (r) is both relative, the curvature RV (r) refers to is in the direction of gravity.

The other class is the leaf. A TomatoLeaf is a rectangular parallelogram with a scanned image of a tomato leaf as texture. The size of (the enclosing rectangle of) a new leaf is determined by the real area of that leaf. The simulation of leaf includes two aspects, the controlling of leaf's position and angle with its branch:

```
Mark Vertex(0.025)RU(33)RV(0.1)M(0.70f)Vertex(0.017)for((1:8))(RV(0.1)M(0.15f)
Vertex(0.017))Surface(newCircle()).(setShader(branchmat))
[RU(40) AdjustLU leaf(0.76, 0.38).(setShader(leafmat))]
//The leaf was on the apex.
[M(-0.06)RL(90)RU(15)AdjustLUleaf(0.8,0.4).(setShader(leafmat))]
[M(-0.06)RL(-90)RU(15)AdjustLUleaf(0.76,0.38).(setShader(leafmat))]
//The leaf was within easy reach of apex.
[RU(33)RV(0.1)M(0.7)for((1:3))(RV(0.1)M(0.15f)Vertex(0.017))RL(90)RU(10)
AdjustLU leaf(0.68, 0.34).(setShader(leafmat))]
//To position the leaf according as the curvature of branch.
```

RESULTS AND DISCUSSION

Simulation Figure 3 shows the outcome of five simulation runs of tomato plant model which was one of the five experimental plants at different times within one month, in addition, the corresponding point clouds were also shown by comparison. Actually, the plants were scanned every two or three days, yet in view of space limitations, we just show five simulation runs in this paper.



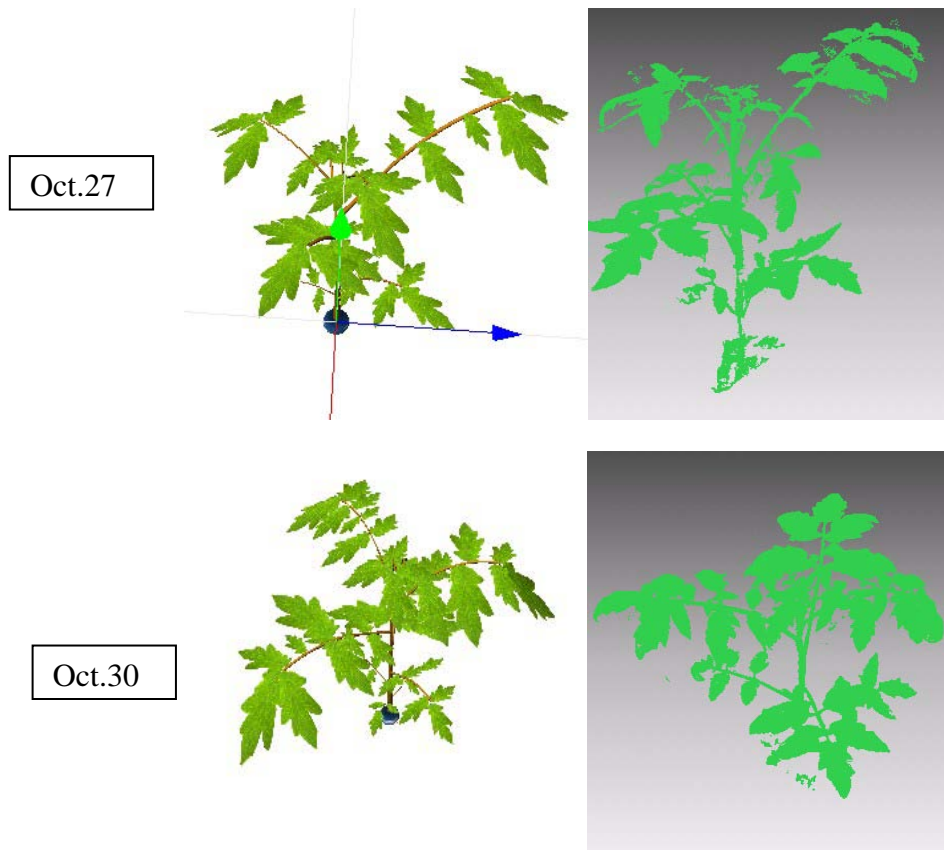


Figure 3. The comparison of tomato plants models and corresponding point clouds at different times from Oct.9.2009 to Oct.30.

Model analysis

By combining the modelling capabilities of GroIMP with the generative rule-based approach of XL, the virtual tomato plants were simulated. From Fig.3 we can see that the models were approximately similar with real plants. However, it also has several aspects should be improved in the future work. For example, the leaves in these models were just a scanned image thereby the shape of each leaf was same with others. A tomato leaf simulation method is greatly necessary. One more, the stem was simulated with cylinder yet the real tomato plant stem might approaches semicircle even more. Lastly, the curvature of stems or branches was hardly to measure, it's very difficult to simulate stems or branches complete conformably with real entities.

CONCLUSIONS Structural plant modeling based on real parameters using modeling environment GroIMP and rule-based language XL got a well-pleasing outcome of simulation runs of tomato plants model. Entities of models were approximately similar with real plants. It commendably shows the growth tendency of tomato plants. However, more new methods should be created to simulate each entity to improve the model effect. The study also shows that GroIMP is a versatile modelling tool, providing an integrated developmental environment for XL-based models and the possibility of an easy visualisation of 3D simulation results at plant level. Comparing XL with L-systems, the true graph representation is advantageous.

ACKNOWLEDGMENTS This study was supported by the Natural Science Foundation of China (Project No: 60802038), the National High Technology Research and Development Program of China (2006AA10Z234).

REFERENCES

- Buck-Sorlin, G.-H., O. Kniemeyer, and W. Kurth. 2005. Barley morphology genetics and hormonal regulation of internode elongation modelled by a relational growth grammar, *New Phytologist* 10 (4), 413-431.
- Cescatti, A. 1997. Modelling the radiative transfer in discontinuous canopies of asymmetric crown. I. model structure and algorithms, *Ecological Modelling* 101, 263-274.
- Danjon, F., H. Sinoquet, C. Godin, F. Colin, and M. Drexhage. 1999. Characterisation of structural tree root architecture using 3d digitising and amapmod software, *Plant and Soil* 211 (2), 241-258.
- De Reffye, P., C. Edelin, J. Francon, M. Jaeger, and C. Puech. 1988. Plant models faithful to botanical structure and development, in: *SIGGRAPH'88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*. New York, NY, USA, pp. 151-158.
- Evers, J.B., J. Vos, C. Fournier, B. Andrieu, M. Chelle, and P.C. Struik. 2005. Towards a generic architectural model of tillering in gramineae as exemplified by spring wheat (*triticum aestivum*), *New Phytologist* 166 (3), 801-812.
- Godin, C. 2000. Representing and encoding plant architecture: a review, *Annals of Forest Science* 57 (05-juin), 413-438.
- Kniemeyer, O. 2008. Design and implementation of a graph grammar based language for functional-structural plant modelling. PhD Thesis, BTU Cottbus.
- Kniemeyer, O., G.-H. Buck-Sorlin, and W. Kurth. 2004. A graph grammar approach to artificial life, *Artificial Life* 10 (4), 413-431.
- Lintermann, B., O. Deussen. 1999. Interactive modeling of plants, *IEEE Computer Graphics and Applications* 19 (1), 56-65.
- Prusinkiewicz, P., A. Lindenmayer, and J. Hanan. 1988. Development models of herbaceous plants for computer imagery purposes, in: *SIGGRAPH'88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, ACM. New York, NY, USA, pp. 141-150.
- Prusinkiewicz, P., and A. Lindenmayer. 1990. *The Algorithmic Beauty of Plants*, Springer-Verlag, New York.
- Prusinkiewicz, P., R. Karwowski, and B. Lane. 2007. The L+C plant modeling language, in: J. Vos et al. (Eds.), *Functional - Structural Plant Modelling in Crop Production*, Springer,
- Room, P., Maillette, L. and Hanan, J. 1994. Module and metamer dynamics and virtual plants. *Advances in Ecological Research*, 25, 105-157.
- Sinoquet, H., P. Rivet, and C. Godin. 1997. Assessment of the three-dimensional architecture of walnut trees using digitising, *Silva Fennica* 31 (3), 265-273.
- Smith, C., P. Prusinkiewicz, FF. Samavati. 2003. Local specification of surface subdivision algorithms. *AGTIVE 2003*, vol. 3062 of *Lecture Notes in Computer Science*, pp. 313-327. (Springer-Verlag: Berlin)
- Weber, J., and J. Penn. 1995. Creation and rendering of realistic trees, in: *SIGGRAPH'95: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press. New York, NY, USA, pp. 119-128.