Graphical Models 71 (2009) 1–21

Contents lists available at ScienceDirect

Graphical Models

journal homepage: www.elsevier.com/locate/gmod

PlantGL: A Python-based geometric library for 3D plant modelling at different scales

C. Pradal [a,1], F. Boudon [a,*,1], C. Nouguier [a], J. Chopard [b], C. Godin [b]

[a] CIRAD, Virtual Plants INRIA Project-Team, UMR DAP, TA A-96/02, Avenue Agropolis, F-34398 Montpellier, France
[b] INRIA, Virtual Plants INRIA Project-Team, UMR DAP, F-34398 Montpellier, France

ARTICLE INFO

Article history:
Received 13 August 2007
Received in revised form 6 October 2008
Accepted 9 October 2008
Available online 26 October 2008

Keywords:
Graphic library
Virtual plants
Crown envelopes
Plant architecture
Canopy reconstruction
Plant scene-graphs

ABSTRACT

In this paper, we present PlantGL, an open-source graphic toolkit for the creation, simulation and analysis of 3D virtual plants. This C++ geometric library is embedded in the Python language which makes it a powerful user-interactive platform for plant modeling in various biological application domains.

PlantGL makes it possible to build and manipulate geometric models of plants or plant parts, ranging from tissues and organs to plant populations. Based on a scene graph augmented with primitives dedicated to plant representation, several methods are provided to create plant architectures from either field measurements or procedural algorithms. Because they are particularly useful in plant design and analysis, special attention has been paid to the definition and use of branching system envelopes. Several examples from different modelling applications illustrate how PlantGL can be used to construct, analyse or manipulate geometric models at different scales ranging from tissues to plant communities.

1. Introduction

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, e.g. [1–9]. 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, e.g. [10–16]. 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 systems adapted to the requirements of biological applications.

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 designed a computer platform, known as L-Studio/VLab, for the simulation of plant growth based on L-systems [17,1,3]. This system makes it possible to model the development of plants with efficiency and flexibility as a process of bracketed-string rewriting. In a recent version of LStudio/VLab, Karwowski and Prusinkiewicz 18 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 [19]. An alternative implementation of a L-system-based software for plant modeling was designed by W. Kurth [20] in the context of forestry applications. This simulation system, called GroGra, was also recently re-engineered in order to model the development of objects more complex than bracketed strings. The

* Corresponding author.
E-mail address: frederic.boudon@cirad.fr (F. Boudon).
[1] These authors contributed equally to this work.

1524-0703/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.gmod.2008.10.001

resulting simulation system, GroIMP, is an open-source software that extends the chain rewriting principle of L-Systems to general graph rewriting with relational graph growth (RGG), [21,22]. Similarly to L+C, this system has been defined on top of a widely used programming language (here Java). Non-language oriented platforms were also developed. One of the first ones was designed by the AMAP group. The AMAP software [2,23] makes it possible to build plants by tuning the parameters of a predefined, hard-coded, model. Geometric symbols for flowers, leaves, fruits, *etc*., are defined in a symbol library and can be modified or created with specific editors developed by AMAP. In this framework, a wide set of parameter-files has been designed corresponding to various plant species. In the context of applications more oriented toward computer graphics, the XFrog software [5,24] is a popular example of a plant simulation system dedicated to the intuitive design of plant geometric models. In XFrog, components representing basic plant organs like leaves, spines, flowerlets or branches can be multiplied in space using high-level multiplier components. Plants are thus defined as graphs representing series of multiplication operations. The XFrog system provides an easy to use, intuitive system to design plant models, with little biological expertise needed.

Therefore, if accuracy, conciseness and transparency of the modeling process is required, object-oriented, rule-based platforms, such as L-studio/VLab or GroIMP, are good candidates for modelers. If interactive and intuitive model design is required, with little biological expertise, then component-based systems, like XFrog, or sketch-based systems are the best candidates. However, if easiness to explore and mathematically analyse plant scenes is required, none of the above approaches is completely satisfactory. Such an operation requires high-level user interaction with plant scenes and dedicated high-level mathematical primitives. With this aim, our team developed the AMAPmod software [25] several years ago, and its most recent version, VPlants, which enables modelers to create, explore and analyse plant architecture databases using a powerful script language. In a way complementary to L-Studio/VLab, VPlants allows the user to efficiently analyse plant architecture databases and scenes from many exploratory perspectives in a language-based, interactive, manner [26–29]. The PlantGL library was developed to support geometric processing of plant scenes in VPlants, for applications ranging from computer graphics [30,31] to different areas of biological modeling [32–34,15,35,36]. A number of high-level requirements were imposed by this context. Similarly to AMAPmod/VPlants, the library should be open-source, it should be fully compatible with the data structure used in AMAPmod/VPlants to represent plants, *i.e.* multi-scale tree graphs (MTGs), it should be accessible through an efficient script language to favor interactive exploration of plant databases, it should be easy to use for biologists or modellers and should not impose a particular modelling approach, it should be easily extended by users to progressively adapt to the great variety of plant modelling applications, and finally, it should be interoperable with other main plant modelling platforms.

These main requirements lead us to integrate a number of new and original features in PlantGL that makes it particularly adapted to plant modelling. It is based on the script language Python, which enables the user to manipulate geometric models interactively and incrementally, without compiling the scene code or recomputing the entire scene after each scene modification. The embedding in Python is critical for a number of additional reasons: (i) the modeller has access to a powerful object-oriented language for the design of geometric scenes, (ii) the language is independent of any underlying modelling paradigm and allows the definition of new procedural models, (iii) high-level manipulations of plant scenes enable users to concentrate on application issues rather than on technical details, and (iv) the large set of available Python scientific packages can be freely and easily accessed by modelers in their applications. From a contents perspective, PlantGL provides a set of geometric primitives for plant modelling that can be combined in scene-graphs dedicated to multiscale plant representation. New primitives were developed to address biological questions at either macroscopic or microscopic scales. At plant scale, envelope-based primitives have been designed to model plant crowns as volumetric objects. At cell scale, tissue objects representing arrangements of plant cells enable users to model the development of plant tissues such as meristems. Particular attention has been paid to the design of the library to achieve a high-level of reuse and extensibility (e.g. data structures, algorithms and GUIs are clearly separated). To favor the exchange of models and databases between users, PlantGL can communicate with the other modelling platforms such as LStudio/VLab and is available under an open-source license.

In this paper, we present the PlantGL geometric library and its application to plant modelling. Section 2 describes the design principles and rationales that underly the library architecture. It also briefly introduces the main scene graph structure and the different library objects: geometric models, transformations, algorithms and visualization components. Then, a detailed description of the geometric models and methods dedicated to the construction of plant scenes is provided in Section 3. This includes the modeling of organs, crowns, foliage, branching systems and plant tissues. A final section illustrates how PlantGL components can be used and assembled to answer particular questions from computer graphics or biological applications at different levels of a modelling approach: creating, analysing, simulating and assessing plant models.

2. PlantGL design and implementation

A number of high-level goals have guided the design and development of PlantGL to optimize its reusability and diffusion:

- *Usefulness*: PlantGL is primarily dedicated to researchers in the plant modelling community who do not necessarily have any *a priori* knowledge in computer graphics. Its interface with modellers and end-users should be intuitive with a short learning curve.