INVITED

Virtual Plants – Integrating Architectural and Physiological Plant Models

JIM HANAN
jim@ctpm.uq.edu.au

Cooperative Research Centre for Tropical Pest Management
CSIRO Division of Entomology
Private Bag # 3, Indooroopilly
Queensland, Australia 4068

Abstract Recent advances in computer technology have made it possible to create *virtual plants* by simulating the structural development of individual plants, opening up new application areas for plant modelling. The simulation of plant development has been approached from two very different angles. Architectural models tend to focus on capturing the three-dimensional structure of a single plant as a product of an internal developmental program, while crop models take aspects of plant physiology into account to estimate biomass production as a consequence of environmental inputs. Integration of these approaches will provide benefits on both sides. Architectural models will gain the ability to react to broad environmental conditions, incorporating physiological parameters of the plant parts being modelled. Physiological simulations such as crop models will gain a visualisation component and could use the added structural detail to respond to spatial variations in conditions that result from plant growth. This integration requires the resolution of the fundamentally different time scales underlying present models. Architectural models are usually based on physiological time; each time step encompasses the same amount of development in the plant, without regard to the passage of real time. In contrast, physiological models have time steps based in real time; the amount of development in a time step is dependent on environmental conditions during the period. This paper presents a case study in which an architectural plant model based on parametric L-systems is extended to allow the incorporation of crop modelling routines which respond to environmental conditions. Developmental rules that operate in real rather than physiological time are developed, based on widely used concepts of thermal time.

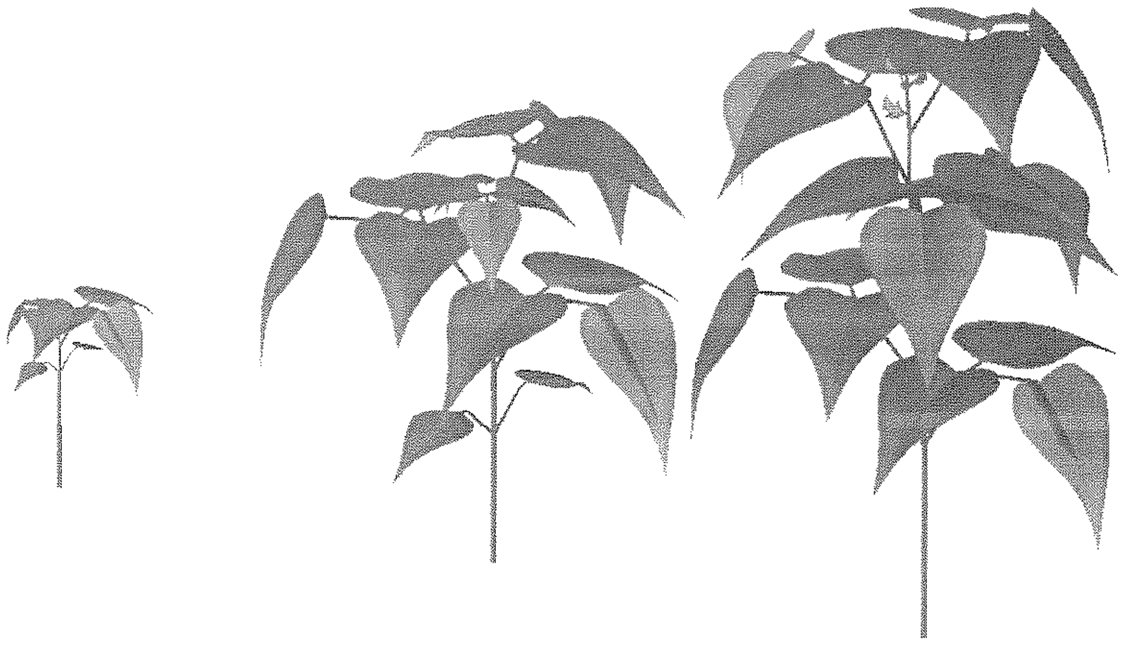

Figure 1: Visualisation of three stages of a virtual bean plant modelled using L-systems.

1. 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 are run, the behaviour of a plant's apical meristems and their products are reproduced in the computer's memory. The resulting *virtual plant* captures the changes that occur in the arrangement of component parts as the plant develops and grows.

Virtual plants are founded on architectural models that simulate the developing three-dimensional (3-D) structure of plants. These models focus on the plant's internal developmental program expressed as "rules of growth" applied to plant components in successive time steps. The particular rule to be used depends on the component's current state as identified by internal factors like component age and position, or the presence of signals due to hormone flow and internal resource partitioning. Environmental effects are considered to be implicit in the current state of the components. This is particularly true in stochastic models, where both internal and environmental effects are expressed in the probabilities of meristem development derived from measurements under varying conditions (Jaeger and de Reffye [1992]). More explanatory models of structural dynamics can be developed by explicit simulation of internal control mechanisms, as is possible using an approach based on Lindenmayer systems (Prusinkiewicz and Lindenmayer [1990]). This approach allows interactive development of models incorporating both deterministic and stochastic rules in the same model, and has been used successfully in modelling a wide variety of plant types. It serves as the basis for further discussion in this paper.

Besides allowing for highly realistic visualisation of plant development (Figure 1), virtual plants provide a dynamic platform for analysing how 3-D structure affects a plant's interactions with its environment. For instance, virtual plants could be used in simulations supporting research in pesticide deposition or insect behaviour. In order to do this effectively and to be useful in the widest variety of applications, the modelling system which gives the virtual plant its "life" must allow the incorporation of models of physiological processes to represent development under varying environmental conditions.

The extensive research into crop modelling (see review by Whisler *et al.* [1986]) provides a broad range of physiological models for use with virtual plants. These systems focus on predicting biomass production at a crop level as a consequence of environmental inputs, capturing the processes at work either empirically or mechanistically. They commonly take agronomic and weather data as input, and may include detailed sub-models of systems components such as soil-water processes, nutrient availability, and photosynthate production and distribution. They rarely treat plant morphology in detail other than at a collective, topological level.

Integration of architectural and physiological approaches will provide benefits on both sides. Architectural models will gain the ability to react to broad environmental conditions, incorporating physiological parameters of the structural components. In simulating plant-insect interactions, for example, it would be possible to determine the nitrogen status of plant components serving as food for the insects, in addition to having both insects and plants react appropriately to temperature fluctuations. Physiological simulations such as crop models will gain a visualisation component making their performance easier to understand and communicate to farmers and students. Crop models could use the added structural detail to respond to spatial variations in conditions which result from plant growth, for instance by incorporating more detailed models of light interception.

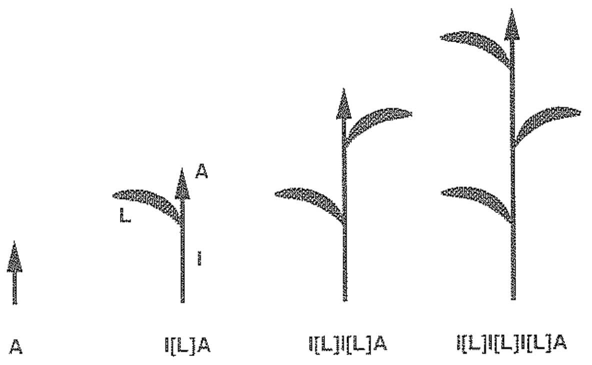

Figure 2: L-system string representation of the development of a simple plant.

This paper describes how the features of a plant modelling system based on Lindenmayer systems (L-systems) can be used to incorporate models of physiological processes into virtual plants. First, a brief background on the L-systems formalism underlying the models will be given, and the main features of the plant modelling system will be described. The following section will look at issues related to integrating the two types of models based on the commonly used concept of thermal time. Then a simple case study is used to present the application of these techniques. Finally, the paper is summed up and avenues for further research are presented.

2. L-SYSTEMS

L-systems were developed by Lindenmayer [1968] as a mathematical formalism for modelling the development of the branching architecture of plants at a topological level. With the extension of the formalism to include continuous parameters quantifying component attributes and the addition of procedures for 3-D graphical interpretation based on turtle geometry, they became suitable for the creation of virtual plants. For a history and formal description of parametric L-systems with turtle interpretation see Prusinkiewicz and Lindenmayer [1990]. The remainder of this section will provide an overview.

In the L-system formalism, symbols are used to represent particular plant components. For instance, in Figure 2, the letters I, L and A stand for internode, leaf and apex, respectively. To represent an entire plant, symbols for

Command	Description
+ -	turn left and right
\ /	roll left and right
^ &	pitch up and down
;	set colour
#	set line width
[	save turtle attributes
]	retrieve turtle attributes

Table 1: Sample turtle commands.

Operator	Description
* /	multiplication and division
+ -	addition and subtraction
< <=	less than, less than or equal to
> >=	greater than, greater than or equal to
== !=	equality and inequality
&& ||	logical *and* and *or*

Table 2: Arithmetic and logical operators

each component making it up are assembled into a sequence (or string) capturing the topological connections between components. Branching is handled by enclosing all components of a branch between matching brackets, [and], marking its beginning and end. This is positioned between the two symbols in the string that mark its point of attachment.

In parametric L-systems, the representation of a component is expanded to allow an arbitrary number of numeric parameters to be associated with a symbol. These parameters can be used to capture aspects of a component's state, such as age, length, or chemical composition. This representation, termed a module, is expressed as the symbol followed by a comma-separated list of parameters enclosed in parenthesis. The modeller chooses the symbol to represent a component and assigns meaning to the parameters according to the order that they appear in the list. A leaf might be represented by `L(4.3,2)` where the number 4.3 at the first position represents its length and the number 2 at the second position represents its age. An internode of age 3 with half its possible concentration of a hormone might be represented by `I(3,.5)` and an apex not requiring any parameters by the module `A`.

In order to create a virtual plant, the topological information inherent in the L-system string representation must be complemented by geometric information. Since growth or bending of an internode at the base of a plant will affect the positions of all components above it, it is most appropriate to specify geometry in a relative way. Rather than cluttering individual modules with parameters to describe their geometric placement with respect to their neighbours, a predefined set of modules (Table 1) is used to encode commands for an interpreter, called a turtle due to the origin of this technique in LOGO-style turtle geometry. As the commands are encountered in a left to right pass of the string, they cause the turtle to modify its position and orientation and to change attributes such as width and colour. The first parameter of the module quantifies the change. Branching is handled by saving the turtle's attributes when a branch is encountered and restoring them when the branch has been completely interpreted. A component's position and orientation are then defined by those of the turtle when the module representing the component is encountered in the string. For example, the third plant in Figure 2 could be represented as

```
I(3)[+(45)L(3.5,2)]I(3)[-(45)L(3.5,2)]A(2)
```

where the `I` modules represent internodes with a length parameter, `L` modules represent leaves with length and width parameters, and the `A` module represents an apex of age 2. The command `+(45)` causes the turtle to be turned left by 45 degrees and the command `-(45)` causes it to be turned right by 45 degrees. For clarity, turtle commands will be ignored for the remainder of the discussion in this paper.

The development and growth of a component is defined by rules or *productions* describing how it will change in a time step. The rules are expressed as a replacement or rewriting operation written as

predecessor : *condition* --> *successor*

where the *predecessor* is the module to be replaced and the *successor* is a string of zero or more modules representing the result of development for that time step. The module in the predecessor has a formal parameter, a unique variable name, for each positional parameter appearing in the actual module. The *condition* is a logical-valued expression and the parameters of successor modules are arithmetic expressions which may be formed by combining these variable names and/or constants using a standard set of operators and syntax (see Table 2).

A rule is considered for application to a particular module if the symbol in the predecessor matches that in the module being considered. The variable names are assigned a value from the corresponding position in the actual module parameter list in much the same way as in a subroutine call in an ordinary programming language. For example, the rule

```
I(len,age) : (age<6) --> I(len*1.1,age+1)
```

describes exponential growth of an internode until it reaches age 6. Given the module `I(7,5)`, the variable `len` is assigned the value 7 and the variable `age` is assigned the value 5. The condition is then evaluated and, if its value is true, as it is in this case, the replacement takes place. The expressions appearing in the successor are evaluated to determine the values appearing in the resulting modules. In this example the module `I(7,5)` would be rewritten as `I(7.7,6)`. In the next time step, this rule would not be applied, since the condition does not evaluate as true when the value 6 is assigned to `len`. If no other rule matches a module it is rewritten as itself. For the purposes of this paper, details of modelling information flow in the structure using context-sensitive rules and the possibilities of stochastic production application will be ignored.

Plant development is modelled by taking an *axiom* or starting string of modules representing an initial stage of the plant and rewriting it to produce a new string representing the next stage of growth. A string is rewritten by applying the appropriate rules to each module in the string, conceptually in parallel, *ie.* all at the same time. The rewritten modules maintain their topological neighbourhood in the string. This *derivation* process is repeated to give a series of strings representing successive stages in the plant's life. For example the following L-system would produce the sequence of strings in Figure 2 in three time steps.

```
Axiom: A
A --> I[L]A
```

Note that turtle commands and attribute parameters such as length have been left out.

3. L-SYSTEMS AS MODELLING LANGUAGE

The L-system formalism as described to this point can be considered as a special-purpose language for expressing structural models of plant development. This language can be made more flexible and useful by adding features commonly found in general-purpose programming languages (Hanan [1992]). A detailed description of the current implementation of the language can be found in the User's Manual (James *et al* [1993]).

The most important extension is the introduction of global variables, along with an assignment statement allowing modification of their value. Global variables are accessible from within all productions in an L-system. These variables allow the expression of environmental effects or the collection of statistics related to the complete plant. Syntactically, a list of statements is specified in braces after the condition, and is evaluated only if the condition is true. For example, the following production captures the transformation of a bud B into a flower F:

```
B(age) : (age>2) {b=b-1; f=f+1;} --> F
```

The global variables b and f are updated once for each occurrence of the module B with parameter value greater than 2 in the current string, thus capturing the effect of this rule on the count of buds and flowers represented in the next string.

Standard programming constructs such as built-in functions, input/output procedures, arrays, character strings, and flow-of-control statements are also added to the modeller's set of tools. Of particular interest for this paper is the input procedure `read();` which allows data to be input to the model from a separate file.

In order to allow initialisation, update, and output at the appropriate times, the basic processing cycle of step after step of parallel production application is modified to include sequential processing segments. Each segment is composed of a list of statements enclosed in braces, labelled to indicate its position in the cycle. The sequence is illustrated by the following pseudocode

```
set axiom
process Start statements
loop over derivation steps
    process StartEach statements
    parallel derivation
        - process statements in productions
    process EndEach statements
endloop
process End statements
```

The Start statements are processed before the L-system derivation begins, in order to initialise global variables used for accumulating counts over the complete development cycle or representing environmental factors. The StartEach statements are processed before each derivation step, in order to initialise variables used within a step and to modify environment variables over time. Statements appearing in a production are processed each time the rule is applied during the parallel derivation of the next L-system string. The EndEach statements are processed at the end of each derivation step, to output current step results or store them in global variables. Finally, the End statements are processed at the end of the entire derivation process, to calculate and report overall statistics.

4. INCORPORATING PHYSIOLOGY

There are two possible approaches to integrating physiology into the architectural model underlying a virtual plant. New or existing modules expressing physiology can be added to the architectural modelling system or a generic interface allowing communication between the architectural system and a crop model could be developed, avoiding duplication of computer code. In either case, the key to integrating physiology into architectural models is to resolve the differences between the time scales underlying the models.

Architectural models are usually based on physiological time; each time step capturing the same amount of development in the plant, without regard to the passage of real time. This can be thought of as an internal view of time, with intervals beginning and ending at significant events in a plant's life. The units of the time step depend on the basic structural unit of the model. Some models increment growth by the addition of a metamer, an "internode+leaf+axial bud" complex, in each time step. In this case, the time step will typically be some fraction of a *plastochron*, the interval between the formation of two successive leaf primordia on the apical meristem. If the model works at the level of branch complexes, the time step will be some portion of a growth cycle, for instance the duration of a growth flush for a tree.

In contrast, physiological models have time steps based in real time, with the amount of development that occurs dependent on the environmental conditions during each successive period. This is an external view of time, with intervals measured by a clock and related to the plant's internal processes through environmental effects on development and growth rates. Researchers have found that duration of growth is directly proportional to temperature for many plants, and techniques based on summation of temperatures over time have been applied in crop models with great success. These techniques have been variously termed growing degree-days, heat units, heat sums, and thermal time, the latter term probably best expressing the concept. A detailed discussion of the value and potential problems of using thermal time to predict plant development, as well as methods for determining temperature response functions, is presented by Ritchie and NeSmith [1991].

There are several ways of calculating thermal time, the simplest being by accumulation of daily differences between mean air temperature and the base temperature below which the plant's development stops. This method is appropriate if the plant's developmental response is linear over the range of mean temperatures experienced and if the daily temperature does not fall below the base temperature or rise above a peak developmental threshold for significant portions of the day. This method will be used in the case study in the next section, though there is no reason why more complicated approaches that give a more precise estimate of thermal time could not be implemented.

To convert an architectural model from physiological to real time, the temperature summation approach is used to determine the amount of development that should occur in a real time interval, usually a day. In each step, average or maximum and minimum temperatures are read from the appropriate weather data file and the daily contribution to thermal time is calculated. All productions expressing development are modified to trigger when the appropriate amount of thermal time has past. For instance, the main stem apex will produce a new metamer after the passage of the amount of thermal time representing a plastochron. All rules expressing growth are modified to reflect the variable passage of thermal time in a step, with growth rates expressed as a function of thermal time. An example of an implementation using the L-system approach is given in the next section.

It is important to note that the relationship between development and time and between growth and time are not necessarily the same. In particular, development durations and growth durations differ. In the case of internodes, one is initiated in each plastochron, while individual internodes may expand over a much longer period of time. Growth rates during a time period may also depend on the availability of resources, which may be handled by determining the optimal rate according to the temperature, then reducing it according to limiting factors.

The thermal time approach provides a foundation for incorporating other effects. Some may require modelling of other physical processes, for example to determine water availability in the soil. Others may simply be signaled by environmental events, such as photoperiod effect on initiation of flowering. The final example in the next section will present a simple model of this phenomena.

5. CASE STUDY

In this section, a series of L-systems illustrate the conversion of a model from a physiological to a real time basis. The example is a simplification of a prototype bean (*Phaseolus vulgaris* L.) model. A visualisation of some stages of development of a complete 3-D model of a bean can be seen in Figure 1. The plant forms a main stem with lateral leaves and buds, following a monopodial growth habit. At some point in the growth of the axis, a developmental switch occurs and the apex stops vegetative growth, subsequent development forming a monopodial inflorescence. Details of internode growth, leaf, lateral branch and flower development and commands expressing geometry have been left out of this discussion to focus on the implementation of thermal time.

The following L-system represents a typical plastochron-based model for the switch from vegetative to reproductive development in this plant.

```
#define FSTART 5

Axiom: A(3)

A(age) : (age<FSTART)  --> I[L]A(age+1)
A(age) : (age>=FSTART) --> I[F]A(age+1)
```

The `#define` statement is used to give meaningful names to constants appearing in the L-system. The plant's apex or meristem is represented by the module `A` with a single parameter, age. The time step is assumed to be a complete plastochron. The vegetative stage of growth is controlled by the first production. During this time, marked by apex age less than the constant `FSTART`, the apex produces a new internode and a leaf in each time step, along with an aged version of itself. Flowering is initiated once the apex reaches the age specified in the constant `FSTART` and the apex produces lateral flowers from that point on according to the second production. The following strings capture the first few stages of development for this L-system:

```
A(3)
I[L]A(4)
I[L]I[L]A(5)
I[L]I[L]I[F]A(6)
I[L]I[L]I[F]I[F]A(7)
```

Graphical representations for apex ages 5 to 7 are shown in Figure 3.

In order to convert this model to real time, development must be expressed as a function of thermal time. In the following L-system, the time step is a day and the weather dataset contains daily minimum and maximum temperatures.

```
#define DURATION 30
#define BASETEMP 8
#define FSTART 5

Start: { tt=0; }

StartEach: {
    read(min,max);
```

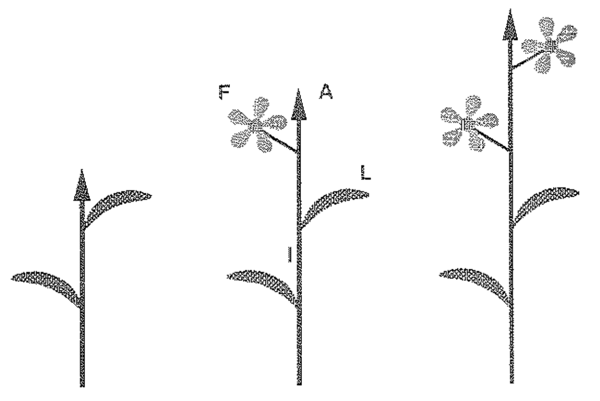

Figure 3: Visualisation of a developmental switch.

```
    tt = tt + (max+min)/2 - BASETEMP;
}

Axiom: A(3,0)

A(age,start) : tt-start>=DURATION && age<FSTART
    --> I[L]A(age+1,start+DURATION)
A(age,start) : tt-start>=DURATION && age>=FSTART
    --> I[F]A(age+1,start+DURATION)
```

where DURATION is a constant equal to the thermal time between the initiation of successive internodes, BASETEMP is the base temperature for bean development, and FSTART has the same meaning as in the last L-system. The module A representing the apex now has two parameters, the first recording age in plastochrons as before and the second recording the time when this apex was initiated in thermal time units.

In this L-system, the global variable tt accumulates thermal time, acting as the plant's internal clock. It is initialised to zero in the Start: code section, and is updated in the StartEach: section by the appropriate calculation using the given base temperature and the daily minimum and maximum temperatures read from the weather file.

The productions are only applied if the duration in thermal time since the last start time is greater than the required amount, as specified in the tt-start>=DURATION clause in both conditions. The switch to flowering is controlled by age using the same mechanism as in the previous example. The start time of the new A module representing the continuing apex is determined by adding DURATION to the previous start time. This accounts for any thermal time units in the daily increment that are in excess of those required for the initiation of the new metamer.

Care must be taken when designing rules based on thermal time. For example, this model will become unsynchronised if the increment in thermal time in a single day is greater than twice the duration of a plastochron. Only one new internode will be initiated that day although two should appear. This problem can be overcome by introducing another production initiating two internodes for this case.

The L-systems presented so far have modelled the developmental switch as a preprogrammed event in the plant's life cycle. Once the apex has produced enough vegetative metamers, as indicated by its age in plastochrons, it begins reproductive development. This process may also be influenced by environmental events. In the case of beans, the switch is often controlled by photoperiod. Since the model is now running in real time, the photoperiod can be determined for each day of the simulation, and flowering induced on the appropriate day. If the simulation is run with different weather inputs, this may result in a different number of main stem nodes being produced before flowering occurs.

```
#define DURATION 30
#define BASETEMP 8
#define FINIT 14

Start:{ tt=0; }

StartEach: {
    read(min,max,period);
    tt = tt + (max+min)/2 - BASETEMP;
}

Axiom: A(0)

A(start) : tt-start>=DURATION && period>FINIT
    --> I[L]A(start+DURATION)
A(start) : tt-start>=DURATION && period<=FINIT
    --> I[F]A(start+DURATION)
```

In this case, the model assumes a short day variety. The length of the daily photoperiod is read into the variable period. When its value falls below the threshold specified by FINIT, flowering will be initiated. Since the age of the apex is no longer required, the A module has a single parameter recording the initiation time of the metamer as in the previous example.

6. CONCLUSIONS AND FUTURE RESEARCH

This paper has presented a preliminary step in integrating architectural and physiological plant models. A plant modelling language based on L-systems was described as an example of an architectural system capable of creating virtual plants. The underlying parametric L-system formalism provides a natural way of expressing plant structural development as the consequence of rules of growth for individual plant components. Extensions to the formalism added programming language features to create a flexible system for expressing plant models of all kinds.

In order to integrate physiological models with L-systems, both must operate on the same time scale. The concept of thermal time can be used to translate from the internal view of time commonly used in architectural models to the external, real-time base required for physiological modelling, as illustrated by the case study in this paper.

Future research will look at integrating more detailed physiological models into virtual plants. The most promising approach appears to be the development of a generic interface capable of communicating between L-systems and various crop models. There are some major hurdles to be overcome. Techniques must be developed to partition biomass accumulation specified at a collective level by the crop model to the individual parts of the virtual plant. Crop models often express phenology in the average, resulting in the appearance of fractions of parts in a time step. This must be translated into the necessarily discrete events of the structural model, possibly by modelling a number of plants in an area rather than an individual plant. Once crop models can be used to supply physiology to virtual plants, more detailed analysis will be required to determine whether the dynamic structural information available in the virtual plant can be used to improve the crop models.

The integration of physiology into virtual plants is only part of a larger story. Research is in progress using L-systems to model plant responses to insect damage at a structural level in beans, cotton, parthenium, and other crops and weeds. A 3-D digitiser is used to collect structural data as the plant develops and the data are analyzed to determine rules of growth for an L-system model. It is hoped that the resulting virtual plants can be used to gain a deeper insight into plant-insect interactions, resulting in advances in pest management techniques.

7. REFERENCES

Hanan, J., Parametric L-systems and their application to the modelling and visualization of plants, Ph.D. thesis, University of Regina, Regina, Saskatchewan, Canada, 1992.

Jaeger, M. and de Reffye, P., Basic concepts of computer simulation of plant growth, *Journal of Bioscience, 17(3)*, 275–291 , 1992.

James, M., Hanan, J. and Prusinkiewicz, P., CPFG version 2.0 users manual, Living Systems Simulations, Calgary, 1993.

Lindenmayer, A., Mathematical models for cellular interactions in development, Parts I and II. *Journal of Theoretical Biology, 18*, 280–315, 1968.

Prusinkiewicz, P. and Lindenmayer, A., *The Algorithmic Beauty of Plants.* Springer-Verlag, 228 pp., New York, 1990.

Ritchie, J. and NeSmith, D., Temperature and crop development, in *Modeling Plant and Soil Systems*, edited by Hanks, J. and Ritchie, J., American Society of Agronomy Inc., 5–30, Madison, Wisconsin, 1991.

Room, P., Maillette, L. and Hanan, J., Module and metamer dynamics and virtual plants, *Advances in Ecological Research, 25*, 105-157, 1994.

Whisler, F., Acock, B., Baker, D., Fye, R., Hodges, H., Lambert, J., Lemmon, H., McKinion, J. and Reddy, V., Crop simulation models in agronomic systems, *Advances in Agronomy, 40*, 141-208, 1986.