IMapple - Functional Structural Model of Apple Trees

Hao Kang[1], Marek Fiser[1,2], Biying Shi[1], Fatemeh Sheibani[1], Peter Hirst[1], and Bedrich Benes[1]
[1] Purdue University, West Lafayette, In USA
[2] Google, Inc., Mountain View, CA, USA

***Abstract*—Computer simulation of tree development can provide important insight into real world tree behavior. Numerous models have been developed ranging from simple equations to complicated 3D geometrical models that allow for simulation of acquisition and allocation of carbohydrates. We introduce IMapple (Interactively Modeled Apple), a functional structural apple tree model that builds on the source-sink model of L-peach. Our model integrates accurate biological functionality, as well as high quality visual geometric details. The details are modeled by using subdivision surfaces that allows for fast generation of high precision geometry and direct illumination of leaves is simulated by sampling the sky and the sun trajectory. The main advantage of the IMapple is that allows for an interactive experimentation by the way it manages data. Input data can be provided interactively or by using real-world samples and the model responds immediately. Alternatively, hypothesized data can be input by drawing functional dependencies, for example the resource assimilation rate as a function of light interception can be input by drawing a graph showing the real or assumed relationship. Moreover, our model supports a variety of convenient user interactions, such as branch pruning and flower thinning. We have collected a large data set over the last few years that serves as inputs to our model. IMapple is fast and it allows for interactive or near-interactive simulation of apple trees. For example, a ten years-old tree grown with a time step of one hour can be simulated in under five minutes using a PC. Moreover, our system allows for a simulation of multiple trees and their mutual interaction. We demonstrate the results by simulating growth and fruiting of Golden Delicious apple trees realistically with data collected from a planting at the Purdue Meigs research farm.**

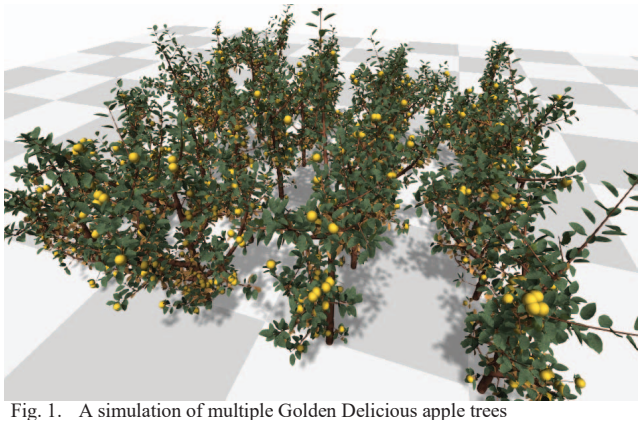


Fig. 1. A simulation of multiple Golden Delicious apple trees

Keywords—Functional-Structural Plant Modeling, Sink-Source Regulation, simulation, orchard, Malus

I. INTRODUCTION

Conducting research that involves the growth of trees in orchards is challenging because of their relatively slow rate of growth, and also the unpredictable growth environment. Conducting field experiments is both time consuming and costly limiting the possible scope of such experiments. A computer simulation of tree development greatly helps to solve such problems.

The procedural and functional tree generation is usually considered from either structural or biological aspects. This makes the resulting 3D tree models to be either realistic without biological functionality or biologically accurate but lacking good appearance. Our model brings the two approaches together and creates high quality 3D geometric models of trees that are biologically accurate, plausible, and visually appealing.

We approach our goal in two steps. We first develop a *modular framework* that allows for the realistic simulation of growth of trees that are both biologically and visually plausible. Then we use the real data collected in our field experiments into *parameters*, and apply them into the framework, in order to create a functional growth model of Golden Delicious apple trees. The assumptions of this study are:

- the geometrical and biological parameters of a tree are known,
- data measured in the field are accurate and significant, and
- trees are not affected by any pests or diseases.

An important part of our simulation framework is user interaction that allows a user to change parameters and interact with the model. Such interactions are very important because fruit trees are a highly managed crop. and being about to simulate and predict the effect of such management interactions should enable optimal management strategies to be identified. The user can prune the branches or thin the flowers, and observe the response of the tree model in a few seconds. These management interactions with the tree can greatly affect tree growth, final crop yield and fruit quality.

Our framework and functional tree model can be used to accelerate tree management research, optimize orchard layout, or it can be used for the education of farmers and students.

II. PREVIOUS WORK

One of the first models of plant development was described by Honda in [1]. The author defined a tree as a recursive structure of branches similar to each other (self-similarity). The

resulting 3D models were simple but they captured a tree-like structure. Smith [2] introduced the term *graftal* – parallel graph grammar language, which shares properties with fractals, and can be used for procedural plant modeling. A plant structure exhibits fractal appearance and features.

Reeves [3] approached the problem of procedural plant generation by introducing a particle system that was able to model *fuzzy* objects such as clouds or fire. The primary focus of that paper was not plants, however some results pertain to a grass patch, which was made by capturing the trajectory of particles coming from the ground up. The particle-based rendering system was later improved by Reeves and Blau [4]. They used probabilistic algorithms to shade particles according to light direction and approximate occlusion of other particles. The authors demonstrated their system on various examples of tree models and grassed areas. Their work also discussed the use of particle system dynamics to simulate the interaction of plant models with environmental factors such as wind. Particle systems were later used in the works of Arvo and Kirk [21] and Greene [22] to simulate environmental sensitive plants and was extended to run at interactive framerates [23].

Bloomenthal [5] made a significant improvement by modeling the trunk and branches of a maple tree using generalized cylinders over space curves that are interpolated between branch segment ends. A realistic bark structure was achieved by bump mapping. Weber and Penn [6] focused their research on improving the realistic look of trees with an emphasis on their overall geometric structure rather than their biological correctness. Their method dynamically adjusts the level of details of the displayed trees based on the distance from the camera. This approach makes it feasible to render large outdoor scenes with possibly hundreds of trees.

A major step forward in procedural plant modeling was presented in the book *The Algorithmic Beauty of Plants* [7], which contained a mathematical formalism describing L-system in a comprehensive way together with an extensive collection of examples of use in procedural plant modeling. L-system is a formal language invented by Lindenmayer for the purposes of cellular subdivision [8]. The parallel string rewrtiting system consists of an alphabet of symbols (also called modules), set or production rules, and an axiom – an initial string of symbols. The production rules are applied iteratively to the axiom in parallel fashion – all symbols are rewritten at once. Finally, the plant is constructed by interpreting symbols from the last iteration as turtle graphics – commands like “move forward”, “turn left”, or “branch” as introduced in [24]. L-systems have been extended in many directions. Prusinkiewicz et al., [9] extended the capabilities of L-system to handle the discrete development of the resulting model. They called the extension Differential L-systems (or dL-systems). So called Open L-systems were presented by Měch and Prusinkiewicz [10] and they introduce environmental modules that serve as a feedback loop for an environment state. Recently, inverse procedural modeling of trees has been introduced [28].

One of the first works combining botanical knowledge with computer graphics was the paper of de Reffye et al., [11]. They covered basic plant structure that included branch (internode), leaf, and buds as well as basic plant growth models. Chiba, et al., [12] presented a growth model that takes into account several biology-based growth phenomena such as withering, heliotropism (attraction to light) and geotropism (tendency to grow against gravity). Their model simulates estimates the amount of sunlight reaching individual plant parts.

A review by Forshey and Elfving [13] summarized the existing knowledge about the biology, growth and fruiting of apple, trees. The authors included a detailed description of the processes in the tree including shoot growth, leaf development, flower bud initiation, fruit growth, and branch thickening. A previous study presented data on apple trees but they did not focus on using that data for simulation of the growth. An apple tree developmental simulation was presented by Costes et al. [14], who called their model MAppleT. This simulation used both biomechanical and stochastic approaches to achieve realistic results. The underlying simulation used L-systems for tree generation and hidden semi-Markov chains for randomization of tree topology and geometry. Similar growth simulation for peach trees called L-PEACH was conducted by Lopez et al., [15]. Their model incorporated important concepts such as carbon assimilation, allocation, and utilization. They simulate resources within a tree using the analogy of an electrical circuit. The system also allowed simulating pruning and its impact on tree growth and fruiting. Both MAppleT and L-PEACH have many common features with the work described in this paper, however our work uses detailed and realistic 3D meshes for the simulation and allows simulation of plant colonies.

The space colonization algorithm described by Runions et al. [16] simulates the competition for space between growing tree branches. This algorithm generates models with a wide range of tree forms and one of its major advantages is that it simplifies the modeling process with few parameters. It also improves the visual realism of tree models. Achieving the realism by simulating growth of a tree may be slow and not interactive. Work by Pirk et al. [17] is achieving the interactivity by transforming input polygonal tree model based on the environment. Their method allows the user to place trees next to obstacles such as houses or other trees and observe the change of their shape immediately. Recently, Longay et al. [25] introduced a fully interactive model that extends the work [16] by providing detailed interactive control by sketching.

Most of the previous work focus on either geometrically precise models and does not focus on the underlying biological methods, or focuses on the biological approaches and uses simplified geometry. We present a model that follows biological rules for tree growth as well as producing realistic-looking trees. Moreover, our model allows for detailed data control of the input parameters.

III. SIMULATION SYSTEM

This section describes our simulation system. The system is modular and was designed with extensibility in mind. It contains a strong foundation of generic functionality so the user can focus on the important parts of the growth model rather than writing utility code. A part of the system is resources allocation, source-sink transport, and light simulation model. Although we use it for simulation of apple trees, the system is designed to be flexible enough to allow modeling other trees and plants.

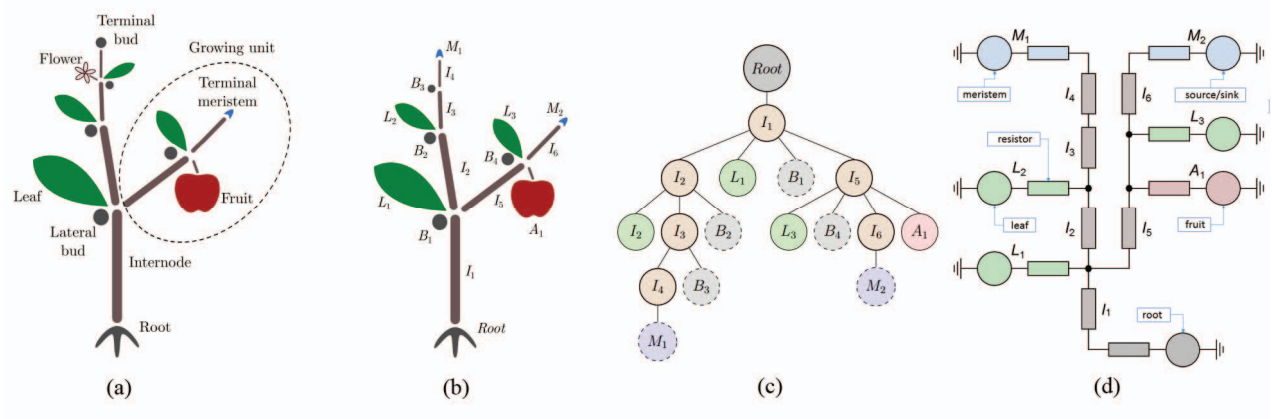


Fig. 2. Biological Tree Representation. Biological model (a), geometric model (b), data structure (c), and circuit model (d) (A=apple, B=bud, I=internode, L=leaf, and M=meristem)

A. Biological Tree Representation

A tree can be represented on many scales ranging from individual cells to large structures such as a trunk and a crown and even to communities of trees. We represent a tree using abstraction shown in Fig. 2(a) and (b). The basic building block of a tree is an *internode* which is an uninterrupted part of the stem. Other organs can only be attached at the ends of internodes. The term *branch* refers to a group of consecutive internodes. A *growing unit* is a set of consecutive internodes that grew in the same year. *Leaves* located at internodes grow from *lateral buds* and those growing at the terminal apex of a branch grow from *terminal buds*. Other organs such as the *flower* of the *fruit* can be also represented based on the needs of the user.

The system represents a biological tree using a rooted tree data structure (Fig. 2(c)). This representation will be called a graph to avoid confusion between data tree structures and biological trees. Every node of the graph represents a tree organ. Nodes can be defined by the user but there are some common nodes predefined in the system, such as internode, leaf, apple, bud, or flower.

The resource transport system (Fig. 2(d)) is motivated by previous research on peach trees [15]. They simulated the flow of carbohydrates in a tree using an electrical circuit analogy. An electrical circuit is built from two logical components: connections and source/sinks. A connection is a resistor and a source/sink is resistor with a voltage source in series, as shown in Fig. 2(d). Every tree node may be either of the two logical parts or ignored. For example, an internode is both a connection and sink, a leaf is a source, and an apple fruit is a sink. Some nodes of a tree graph may not have any electrical properties, thus, the electrical circuit is a sub-graph of the tree graph. There is no need for the explicit construction of the electrical circuit. Fig. 2(d) shows a simple tree and the corresponding electrical circuit. Notice that in this example buds have no electrical properties, therefore are not represented in the electric circuit.

B. System Overview Identify the Headings

The tree growth simulation is performed iteratively by discrete time steps (Δt). The number of time steps per day is a user specified parameter that controls the fineness of the simulation. In our experiments the time step $1 \leq \Delta t \leq 8$ hours. The shorter the Δt, the more precise the simulation becomes, but the more time necessary to run the simulation. The tree simulation system was designed to be general and extensible, thus, it has no fixed pipeline. Fig. 3 shows the system overview.

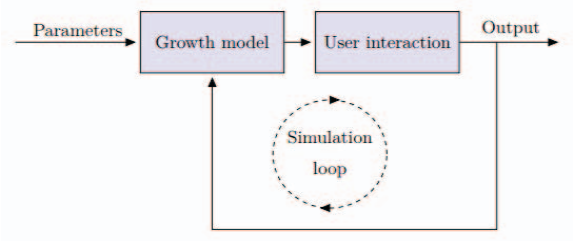


Fig. 3. The growth simulation system overview

C. Growth Model

The growth model represents the logic necessary to perform the tree growth simulation. The simulation is performed by invoking user defined *actions*.

The growth simulation system is not one uniform block, but is divided into modules. Every module may perform any logic in any of the actions shown as blue rectangles in Fig. 4 (the resource transport is described later in this section). Three main actions are performed at every simulation time step: time-step-start, resource transport, and time-step-end (Fig. 4). Time-step-start and time-step-end are actions that perform the growth logic that must occur before and after the resource transport, respectively. For example, a time-step-start action computes the amount of resources produced by a leaf based on the amount of incoming light being calculated in the environmental simulation. The time-step-end action can then use the transported resources for growth simulation.

There are three additional actions that are performed only during certain simulation steps. The initialization action is invoked only once during the very first simulation step of the simulation and initializes the simulation system. Actions year-start and year-end are performed before the first and after the last

simulation step of a year, respectively. Those actions are used for processes that are difficult to simulate on a day-to-day basis.

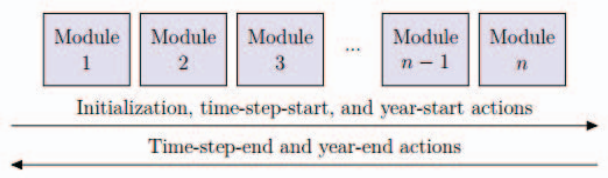


Fig. 4. Order of action invocation on a list of modules based on action type

Division of modules was made from a maintainability point of view. The whole growth model could be theoretically developed as a single module but maintenance and extensibility would be compromised.

Each module usually has a single responsibility. For example, a *leaf module* simulates leaves; an *apple module* simulates apple fruit, etc. Modules easily communicate with each other. For example, some modules serve quite a generic role such as a *temperature module* that provides weather information to all other modules.

Actions are invoked serially over all modules – one action is invoked on all modules before invoking the next action. Serial processing is important to ensure the determinism of the process. For example, a random generator module may provide semi-random numbers to all other modules and if the order would not be defined then the results of the same simulation even with the same value of the seed of the random generator would result in different outputs.

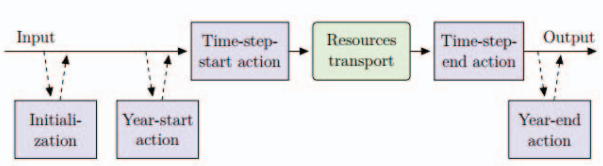


Fig. 5. Actions performed during one simulation step

The order of invocation of actions depends on the type of action. For start actions the order is from the beginning to the end of the list of modules. For end actions the order is reversed; from end to beginning as shown in Fig. 5. Note, that some actions are not necessary in each step and they are shown as optional connected with dotted line. This allows modules earlier in the pipeline to prepare data for later modules in *-start actions. In *-end actions the earlier modules are invoked after the previous ones so that they can effectively pass the data.

Currently, the modules are implemented as a functions in our implementation (C#) and their communication is enabled via data structures. While a more generic approach would be possible, e.g., where the modules would be programmable from some scripting langue, we prefer to sacrifice the generality for having a fast execution and quick response time of the system that allows for interactive experimentation.

D. Light Simulation

The goal of light simulation is to estimate illumination of every leaf. We assume that the light that passes through the leaves or is reflected from other tree parts represents is only a very small part of the total amount. Thus, we compute only direct illumination which can be done efficiently by using a Monte-Carlo technique [18, 19]. Indirect illumination could be calculated by the model of [26].

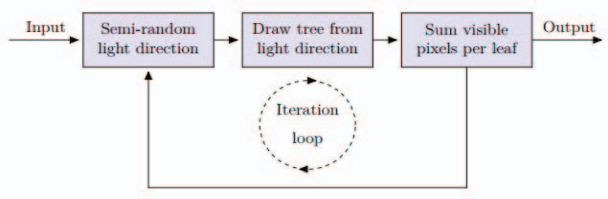


Fig 6:Light simulation

The overview of the light simulation algorithm is shown in Fig. 6. At the start of each iteration a light direction is chosen semi-randomly from possible directions of direct light based on position on Earth latitude and time of year. A point on a hemisphere has a probability to be chosen proportional to an average amount of light emitted from that point of sky over the day. This means that points near the sun's zenith have a high probability of being chosen while points near the horizon have a very small probability.

After choosing the direction, the tree is rendered from the light direction such that individual leaves can be distinguished. This is done by rendering every leaf with a different color. There are 2^{24} (16,777,216) different colors (without using a transparency channel) which is enough for any tree size. The background and all other parts of the tree such as branches or apples are transparent.

Finally, from the rendered image the number of visible pixels per leaf is computed and summed with all previous samples. When all samples are processed, tree resources (carbohydreates) are calculated from the total cosine-corrected pixel counts and saved to leaf nodes in the graph. In particular, the normal vector to each pixel is known by interpolation from the corresponding vertices in fragment shader during rendering. We also know the direction to the light sources. The incoming radiance is then multiplied by the cosine of the angle between the normal and the direction to the light source.

E. Resources Simulation and Transport

The growth simulation system contains a simulation of resource allocation and a transportation sub-system. This is mainly to allow realistic simulation by allowing the user to implement analogy between hydraulic and electric terms growth mechanisms that are driven by resources. The resources may include water, carbohydrates, or nutrition and the user makes the decision.

Resource Analogy	**Hydraulic Entity**	**Electric Entity**
amount of resources	mass or volume	charge
resource flow	flow rate	current
resource concentration	pressure	voltage (potential)
resource flow resistance	hydraulic resistance	resistance

Table I: Resource mapping to electric entities

We use the analogy of an electrical circuit for simulating the flow of carbohydrates in a tree using an earlier method proposed [15]. This is possible due to analogies between hydraulic and electric current flow as shown in Table I.

The analogy with electrical circuits is beneficial in many ways. For example, the role of the source and sink may change based on local conditions. A small growing leaf that is heavily shaded may require more resources to grow than it can produce (i.*e.,* below the light compensation point). Thus, it behaves as a sink even though leaves are formally considered a source. This effect may be difficult to simulate with other approaches and it will occur naturally in the electrical circuit simulation because the voltage of the shaded leaf will be lower than the voltage in the adjacent branch. A similar situation may occur in branches that are considered as a sink due to their growth; however, in the spring they provide resources saved from the last year to initiate the growth. This means that the terms source and sink are merely suggesting the primary role of the organ, however, their actual role can change over time based on the voltage in the network.

Electric circuits need to be normalized to simplify the algorithm for solving currents. In the normalized form, every connection has a single source/sink connected to it. There are two rules used for normalization. 1) If there is no source/sink connected to a connection, an artificial one is created with resistance set to infinity (voltage does not matter). 2) If there is more than one source/sinks connected to an internode then they are combined to one using the standard rule of parallel circuits.

All currents in a normalized circuit are solved by the technique of folding and unfolding described in [19]. Currents are computed iteratively because electrical components may be nonlinear – their voltage/resistance may depend on the current or potential.

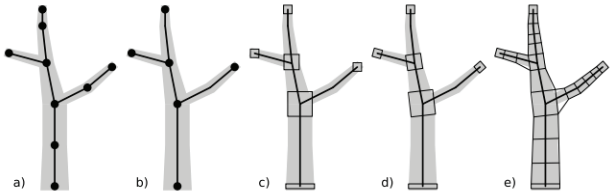


Fig 7: 2D example of steps of the reconstruction algorithm. a) Initial tree graph. b) Simplification of the graph. c) Initial cubes creation. d) Cube alignment based on the branches. e) Connection of the cubes

F. 3D Geometric Model Generation

We generate a 3D manifold watertight mesh representing the tree geometry. This mesh is one solid object with a smooth surface, without holes, self-intersections, or other undesired features. The mesh has a good topology for texturing and editing and it is 3D printable.

The generation algorithm has two main steps: initial mesh creation and subdivision. Fig. 7 shows the subdivision algorithm at work. The tree graph is simplified and cubes are placed at the center of each node. The size of each cube is based on the branch diameter of each node. The cubes are then aligned based on the incoming and outgoing branches to minimize errors. Finally, the straight segments between the boxes are filled.

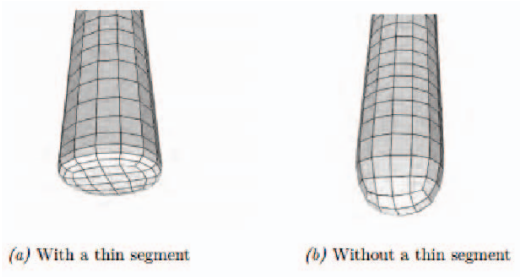


Fig 8: Effect of a thin box at a base of a tree

The implementation uses a face extrusion that ensures that the mesh is manifold. It starts with a thin box at the root and then keeps extruding the faces. The root is represented as a thin box instead of a cube to prevent a round shape that would emerge after the subdivision. Fig. 8 shows the comparison between the two. The mesh from the first step is rough and it needs to be subdivided to obtain good quality results. We use the Catmull–Clark algorithm for the subdivision [20]. Our experiments demonstrated that one or two iterations of the subdivision algorithm are enough to generate smooth shaded 3D meshes. More iterations might be required to achieve a smooth surface for other applications such as 3D printing. Fig. 9 shows the entire process of reconstruction on a simple tree and Fig. 10 shows detail of the subdivision process.

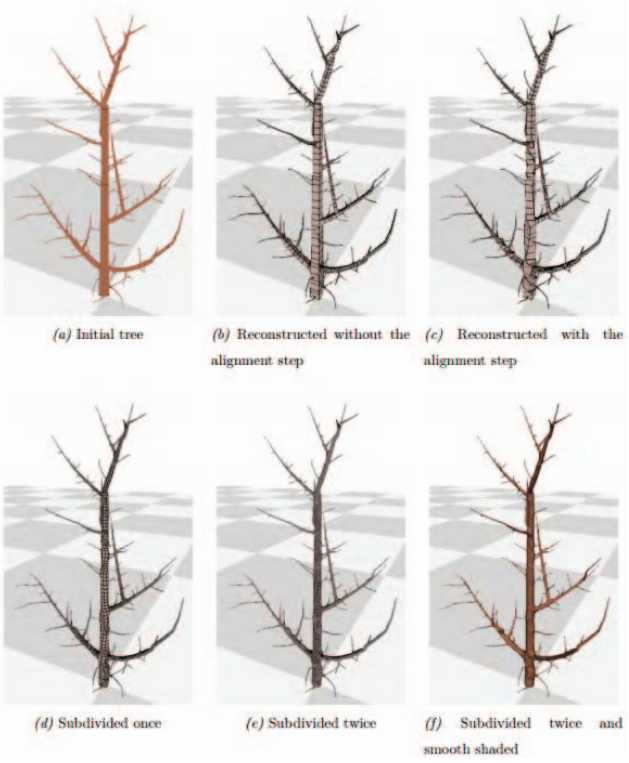
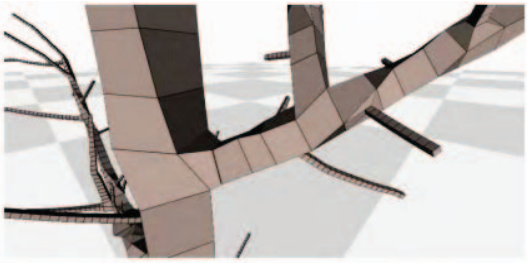
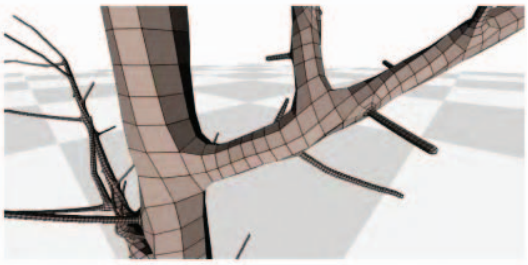


Fig 9: An example of a complete tree construction.

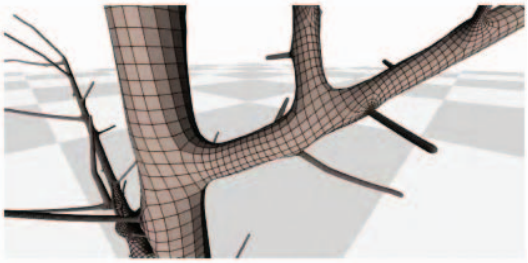
The modeling requires geometric parameters, such as branch diameter, internode length, branching angle, etc. Our system allows for input of those parameters interactivelly from a menu or it uses procedural values as defined in [27]. Geometric atributes influence the simulation, for example wider branches cast shadow on the leaves, etc.



(a) Initial extruded and aligned mesh



(b) Mesh after one subdivision step



(c) Mesh after two subdivisions steps

Fig. 10: Detailed mesh of a tree

IV. APPLYING REAL DATA INTO THE SYSTEM

Real-world data collected from trees and orchards can be easily input into the system to specify the tree model we want to simulate. The growth model of Golden Delicious apple tree can be created using the simulation system covered in the previous section. The model is based on previous work and data collected in an apple orchard during the years 2014 and 2015.

The growth model is composed of modules that can be divided into four main categories: 1) resource generation, 2) growth, 3) branching, and 4) flowering.

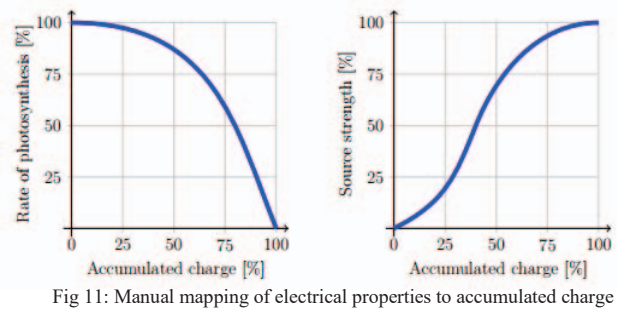


Fig 11: Manual mapping of electrical properties to accumulated charge

Here, we convert illumination of leaves to charge from resource generation as an example to demonstrate how real data is incorporated into the system. Every leaf converts received illumination to resources – represented as electric charge. This conversion is not linear and it depends on several factors. We use relationships described in [19] and a leaf has charge capacity proportional to its size. Fig. 11 shows how the resources assimilation rate and source voltage depend on the amount of accumulated charge. The motivation behind the functions is that the more unused resources within a leaf, the lower the resource assimilation rate but the higher the voltage of its source. Higher source voltage will cause higher current flowing from the leaf.

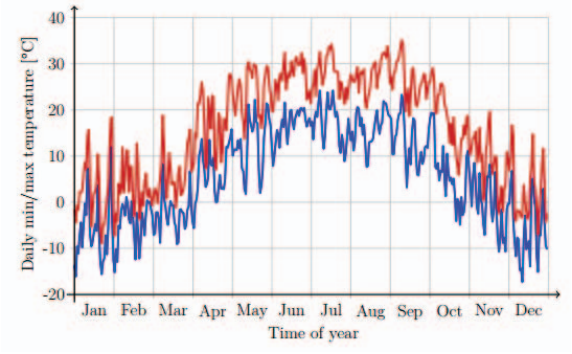


Fig. 12: Temperature data from Meigs farm for the year 2014. Red is daily maximum temperature and blue is daily minimum. Fig. 13 shows values of GDD and CGDD (cumulative GDD) computed from this data.

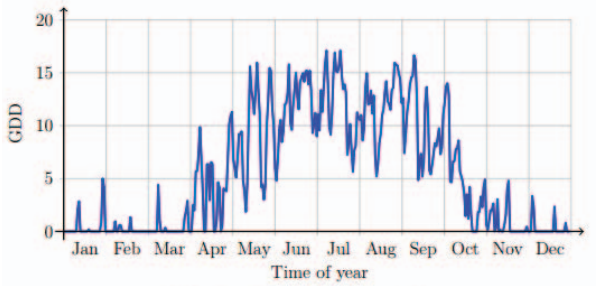
Resource assimilation rate also depends on current temperature. The temperature is converted to *growing degree days (GDD)* as

$$GDD = \frac{T_{max}+T_{min}}{2} - T_{base} \qquad (1)$$

where T_{max} and T_{min} are, daily maximal and minimal temperatures, and T_{base} is base temperature. For Golden Delicious $T_{base} = 10^o\ C$ and both T_{max} and T_{min} are limited to range $[T_{base}, T_{top}]$ (T_{top} is usually $30^o C$). Thus, values of GDD have range $[0, T_{top} - T_{base}]$. Figures 12 and 13 show measured temperatures and values of GDD for the year 2014 at Purdue Meigs farm.

For the growth control we use a percentage of *GDD* which is computed as

$$GDD_{\%} = \frac{GDD}{T_{top} + T_{base}} \quad (2)$$



(a) Growing degree days over year 2014

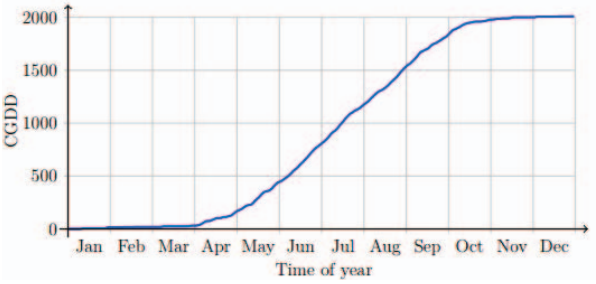


Fig 13: GDD and CGDD computed from temperature data in Fig. 12

The total charge assimilation is computed using the function shown in Fig. 14.

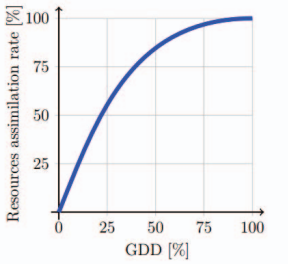


Fig 14: Resources assimilation rate as a function of growing degree/days percent

V. USER INTERFACE AND INTERACTIONS

The user interface was designed to be intuitive and simple to understand and use. There are three main components: the main menu on the top, the modules list on the left, and the 3D viewport on the right (Fig. 15).

All parameters of all modules are automatically displayed in the list. Many parameters of modules are expressed as curves and the built-in curve editor shown in Fig. 16 is a powerful tool allowing interactive editing of parameters as cubic Bezier curves. The curve editor supports sampled functions as well as histograms. The user can copy-paste values in a text form directly to the interface, for example measured values of certain properties, or the values can be estimated based in the user experience and insight. This allows experimentation and "what-if" scenarios.

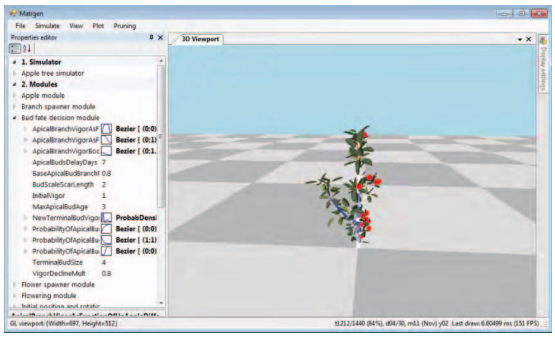


Fig 15: Graphical User interface of the IMapple application

An important part of the user interface is pruning. A user can interactively select any branch and prune it as shown in Fig. 17. The system also supports flower thinning interaction which is handled in a similar way to pruning.

VI. CONCLUSIONS AND FUTURE WORK

In this paper we present a modular framework - IMapple for the realistic simulation of trees. Our framework provides powerful features such as fast light simulation and resource transportation so that the user can focus on the implementation model of a tree. With help of data collected in the years 2014 and 2015 from a planting at the Purdue Meigs Farm, we implemented a functional model of a Golden Delicious apple tree that demonstrates features of our simulation framework.

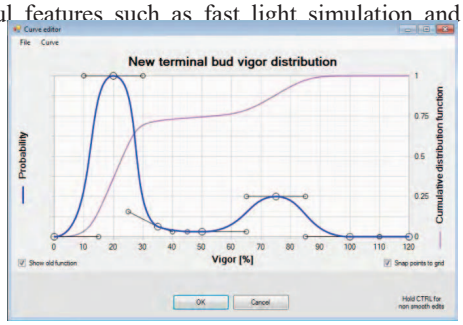
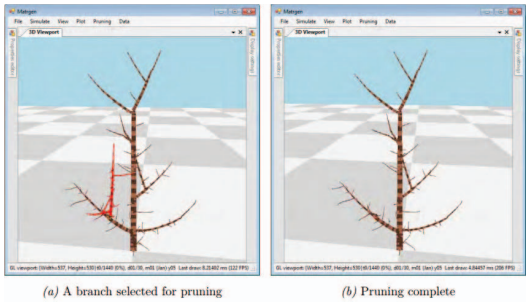


Fig. 16: Curve editor for editing probability distribution

There are some limitations in our system. First, the developed model of Golden Delicious that has been fitted for Lafayette, IN, USA. If used in different conditions, the input parameters should be modified accordingly. We consider only direct illumination. It would be interesting to measure and evaluate indirect illumination (secondary bounces, diffuse-transmission, and subsurface scattering) as well. Other environmental factors, such as water supply and nutrients in the soil, are not taken into consideration. Bending of branches as result of gravity and crop load is not considered.

The data we collected over the two years in the orchard was instrumental in developing the model. However, we found some parameters remain unmeasured and unknown. In future work, we plan to focus on the measurement of missing parameters, replacing assumptions with real data, and thereby improve the model. However, plants are complex systems and the actual dependence of mutual parameters is difficult to estimate.

Gravity is important because it bends branches under the load of fruit. This changes the tree shape in a significant way,



(a) A branch selected for pruning *(b)* Pruning complete

Fig 17: Our user interface for pruning. The user can choose from a variety of informative visualization to assist them with pruning

including the illumination of leaves. Implementing a physics simulation to our model is another area for future work.

Given the realistic tree model, an automatic pruning algorithm can be developed and trained using the model. Automatic pruning can target the highest fruit weight or even the highest fruit revenue as an optimization target. Human pruners can be also trained with the model which is faster and cheaper than training on real trees. The quality of fruit is correlated with illumination. Research of light distribution in a tree crown and its impact can be studied and modeled with our tool. Results in this area may have an impact on pruning strategies.

Our simulator can model not just single trees, but entire orchards of trees (Fig 18). By iterative refinement of tree positions an optimal orchard layout can be found for a particular tree cultivar. This may increase productivity of orchard per unit of land area.

The source-sink model that has been used in L-peach and is also used in our model is an analogy and must have some limitations. It would be interesting to address them by a rigorous comparison of the results with real measurements.

Fig 18: An orchard simulated on a desktop computer in 10 minutes.

References

[1] H. Honda, 1971 Description of the form of trees by the parameters of the tree-like body, *Jour. of Theor. Biology,* vol. 31(2), pp. 331 - 338.

[2] A. R. Smith, 1984 Plants, fractals, and formal languages, ACM SIGGRAPH Comp. Graph., vol. 18(3), pp. 359-375

[3] W. Reeves, 1983 Particle systems-a technique for modeling a class of fuzzy objects, ACM SIGGRAPH Comp. Graph., vol. 17(3), pp. 359-75

[4] W. Reeves, and R. Blau, Approximate and probabilistic algorithms for shading and rendering structured particle systems, ACM SIGG. Comp. Graph., vol. 19(3), pp. 313-322, Jul. 1985.

[5] J. Bloomenthal, Modeling the mighty maple, ACM SIGGRAPH Comp. Graph., vol. 19(3), pp. 305-311, Jul. 1985 .

[6] J. Weber, and J. Penn, 1995 Creation and rendering of realistic trees, ACM SIGGRAPH Comp. Graph., pp. 119-128, 1995.

[7] P. Prusinkiewicz, and A. Lindenmayer, The Algorithmic Beauty of Plants, New York: Springer-Verlag. 1990.

[8] A. Lindenmayer, 1968 Mathematical models for cellular interactions in development I. *Jour, of Theor. Biology,* vol. 18(3), pp. 280-299.

[9] P. Prusinkiewicz, M. S. Hammel, and E. Mjolsness, Animation of plant development, ACM *SIGGRAPH,* pp. 351-360, 1993.

[10] R. Měch, and P. Prusinkiewicz, Visual models of plants interacting with their environment, ACM SIGGRAPH Comp. Graph., pp. 397-10, 1996.

[11] P. de Reffye, C. Edelin, J. Françon, M. Jaeger, and C. Puech, Plant models faithful to botanical structure and development, ACM SIGGRAPH Comp. Graph., vol. 22(4), pp. 151-158, Aug. 1988.

[12] N. Chiba, K. Ohshida, K. Muraoka, M. Miura, and N. Saito, A growth model having the abilities of growth-regulations for simulating visual nature of botanical trees, *Comp. & Graph.,* vol. 18(4), pp. 469-79, 1994.

[13] C. Forshey, and D. Elfving, 1989 The Relationship Between Vegetative Growth and Fruiting in Apple Trees *Hort. Rev.* 11,, pp. 229-87.

[14] E. Costes, C. Smith, M. Renton, Y. Guédon, P. Prusinkiewicz, and C. Godin, 2008 MAppleT: simulation of apple tree development using mixed stochastic and biomechanical models, *Functional Plant Biology,* vol. 35(10), pp. 936-950

[15] G. Lopez, R. R. Favreau, C. Smith, and T. M. DeJong, L-PEACH: a computer-based model to understand how peach trees grow, *Hort Technology,* vol. 20(6), pp. 983-990, 2010.

[16] A. Runions, B. Lane, and P. Prusinkiewicz, Modeling trees with a space, *Eurographics,* pp. 63-70, 2007.

[17] S. Pirk, O. Stava, J. Kratt, M. A. M. Said, B. Neubert, R. Měch, B. Benes, and O. Deussen, Plastic trees: interactive self-adapting botanical tree models, *ACM Trans. Graph.,* vol. 30(4), July 2012.

[18] B. Benes, Visual simulation of plant development with respect to influence, *Computer Animation and Simulation'97,* pp. 125-136, 1997.

[19] B. Benes, Direct illumination of dense foliage using Z-buffer, *Proceedings of SCCG'98,* pp. 237-246, Apr. 1998.

[20] E. Catmull, and J. Clark, Recursively generated B-spline surfaces on arbitrary topological meshes, *Computer-Aided Design,* vol. 10(6), pp. 350-355, Nov. 1978.

[21] N. Greene. 1989. Voxel space automata: modeling with stochastic growth processes in voxel space. ACM SIGG. Comp. Graph. 175-184

[22] J. Arvo and D. Kirk. 1988 Modeling plants with environment-sensitiveautomata. In Proceedings of Ausgraph'88, pages 27 – 33, 1988.

[23] B. Benes and E. U. Millan, 2002 Virtual climbing plants competing for space, Computer Animation, Proceedings of, Geneva, 2002, pp. 33-42.

[24] P Prusinkiewicz. 1986. Graphical applications of L-systems. In Graphics Interface, 247-253.

[25] S. Longay, A. Runions, F. Boudon, and P. Prusinkiewicz. 2012. TreeSketch: interactive procedural modeling of trees on a tablet. Proceedings of SBIM, 107-120.

[26] C. Soler, F. X. Sillion, F. Blaise, and P. Dereffye. 2003. An efficient instantiation algorithm for simulating radiant energy transfer in plant models. ACM Trans. Graph. 22, 2 (April 2003), 204-233.

[27] J. Weber and J. Penn. 1995. Creation and rendering of realistic trees. In Proceedings of the 22nd annual conference on Computer graphics and interactive techniques (SIGGRAPH '95)

[28] Stava, O., Pirk, S., Kratt, J., Chen, B., Mech, R., Deussen, O., and Benes, B., (2014) Inverse Procedural Modeling of Trees , in Computer Graphics Forum, Vol 33(6), pp. 118-131,