DOI: 10.1111/cgf.13106

EUROGRAPHICS 2017 / L. Barthe and B. Benes
(Guest Editors)

Volume 36 (2017), Number 2

Interactive Modeling and Authoring of Climbing Plants

Torsten Hädrich[1] and Bedrich Benes[2] and Oliver Deussen[1,3] and Sören Pirk[4]

[1]University of Konstanz, Germany [2]Purdue University, USA [3]VCC, SIAT, China [4]Stanford University, USA

Figure 1: *An example of a modeling and editing session of a set of climbing plants. The user places seeds to initiate the plant growth (left). The user then draws attractors on the object surface. The plants grow on the surface and dynamically react to environmental changes (middle). The user interactively removes parts of the plant during the growth to make the windows and the lights of the car visible (right).*

Abstract

We present a novel system for the interactive modeling of developmental climbing plants with an emphasis on efficient control and plausible physics response. A plant is represented by a set of connected anisotropic particles that respond to the surrounding environment and to their inner state. Each particle stores biological and physical attributes that drive growth and plant adaptation to the environment such as light sensitivity, wind interaction, and physical obstacles. This representation allows for the efficient modeling of external effects that can be induced at any time without prior analysis of the plant structure. In our framework we exploit this representation to provide powerful editing capabilities that allow to edit a plant with respect to its structure and its environment while maintaining a biologically plausible appearance. Moreover, we couple plants with Lagrangian fluid dynamics and model advanced effects, such as the breaking and bending of branches. The user can thus interactively drag and prune branches or seed new plants in dynamically changing environments. Our system runs in real-time and supports up to 20 plant instances with 25k branches in parallel. The effectiveness of our approach is demonstrated through a number of interactive experiments, including modeling and animation of different species of climbing plants on complex support structures.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling— I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques; I.6.8 [Simulation and Modeling]: Types of Simulation—Visual

1. Introduction

Vegetation plays a key role in increasing the realism of 3D scenes in various application domains ranging from entertainment to architectural design. While plant libraries are readily available and commonly used, content creators are often confronted with the need to obtain plant models that dynamically adapt to concrete, yet changing scenes. Thus, a dynamic plant model is needed to react to the presence of other plants, to varying lighting conditions, and to the scene itself. Moreover, designers require full control over all stages of the plant development to employ plants as part of the storytelling in their applications.

A number of approaches have been proposed for realizing adaptive plants. Most of them consider plants as biological models and create the 3D geometry by growing the plants *in silico* by means of some kind of environmentally-sensitive biological simulation. Some approaches involve simulation of plant development via parallel string rewriting systems [PL90] where the exogenous flow is achieved by using environmentally-sensitive query modules [PJM94, MP96]. Other methods address the requirements of adapting plants to their environment via inverse procedural models [SPK*14], competition for resources [RLP07, PHL*09, LRBP12], or simulated adaptation [PSK*12].

An especially difficult problem are *climbing plants*, such as grapevine, clematis, or ivy, where the plant geometry needs to be considered in context of its adaptation to the geometry of the supporting objects. Previous approaches simulated them via environ-

mentally sensitive automata [AK88], competing particles in voxel space [Gre89, BM02], and recently as particles with tendrils represented as mass-springs [WC15]. However, a major open problem in vegetation modeling is the control. While highly interactive methods with great control exist [PHL*09, LRBP12], most of the existing algorithms focus on standing trees and they are usually controlled by setting input parameters and the initial location of the tree. Moreover, coupling physics with biologically-plausible behavior is arguably a challenging problem as both domains require specific knowledge that cannot be easily abstracted.

The key idea to our approach is to model a plant as an evolving system that dynamically adapts to its environment, reacts to physics, and is easy to control by the user. To simulate physical response, our approach extends *meshless deformation* as proposed by Müller and Chentanez [MC11] and the agent-based simulation of living parts as environmentally sensitive elements [AK88, Gre89, BM02]. We model plants as sets of connected anisotropic particles that communicate among themselves and with the environment. These particles store biological and physical attributes, *e.g.,* position, stiffness, and growth parameters, which drive plant development. Particles compete for resources and together let emerge the geometric shape of the plant with its distribution of organs. We simulate physical interaction with the environment, such as bending, breaking aremoval of branches, as well as the plants response to wind. We focus on interactive modeling of plants and provide means for their efficient editing.

Figure 1 shows an example of modeling by using our system. Three complex plants grow from interactively placed seeds (left). The user draws attractors on the surface of the car and the plants generate a couple of thousand of branches and leaves. The plants dynamically react to the changes in their environment and to the presence of other plants (middle). The user interactively prunes some branches to make the window and the lights of the car visible (right). We demonstrate the effectiveness of our framework by showing a number of results in the paper and in the accompanying video. In summary, the main contributions of our system are:

- We implemented an interactive method that allows for coherent modeling of climbing plants in changing environments and along the entire developmental process of the plant.
- We model climbing plants as dynamic systems that support biologically- and physically-plausible behavior; plants remain flexible and animation-ready during the modeling session.
- We couple plants with wind simulations and model advanced physical effects such as the bending and breaking of branches.
- We introduce a number of editing operations including plant seeding, dynamic branch placement, removal, and sketching of attractors on support geometry.

2. Related Work

Early approaches to vegetation modeling exploited self-similarity and fractals [AK84, Kaw82, Smi84, Opp86]. Other works use biological modeling such as L-systems that describe branching plants by string-rewriting mechanisms [PL90]. Below we focus on plant environmental response and interactive methods and we refer the reader to the recent state of the art report on vegetation modeling [PBI*16].

Environmentally-sensitive models incorporate environmental responses of plant development into the modeling process. One class of methods simulates vegetation as a set of particles that move through 3D space and compete for resources. The main disadvantage of these methods is their low controllability. The user is only able to place input seeds and adjust input parameters for the simulation. Closest to our approach are methods that simulate climbing plants [AK88, Gre89, BM02] or trees by space colonization [PJM94, MP96, RLP07, PHL*09]. While these works allow for an interactive control by positioning attraction particles, our method enables the user to interactively control various aspects of plant development including physics.

Recent approaches reuse plant development to simulate plants by either inverse procedural modeling [SPK*14], by simulating trees as real-time interactive growing systems [PSK*12, PNDN12], by simulating the effect of wind on tree development [PNH*14], or by using FEM simulations to author trees interactively [ZB13]. While these algorithms allow for various forms of control, they require to store the entire plant and its complete development or to represent it as a set of parameters of an inverse procedural model. Casati and Bertails-Descoubes [CBD13] simulate twining plants by elastic rod kinematics. Their approach allows for a precise integration of the underlying kinematics, but it is not applicable for the interactive generation of complex plants.

Interactive methods focus on user control. The Xfrog module-based system of Lintermann and Deussen [LD99] is commonly used for vegetation modeling. Ijiri *et al.* [IOI06] developed a system for flower modeling, an example-based sketching system was introduced in Okabe et al. [OOI07]. Longay *et al.* [LRBP12] present a system that integrates many aspects of interactive plant modeling and uses the space colonization algorithm [PHL*09] for plant growth. Although these methods provide control through successive iterations, our climbing plants can also interactively grow along objects with control of their growth and physics.

Climbing plants were first described and modeled by Arvo and Kirk [AK88] and by Greene [Gre89]. Both approaches define environmental effects such as light and proximity to obstacles and direct growth into suitable areas. Benes and Millán [BM02] provide a real-time implementation of an ivy simulator, but the control is limited to seeding and Luft [Luf08] later provided an ivy plant generator implementation that is currently used in many modeling systems. Their method focuses on the interaction with support structures, but it does not support physics and only provides limited control. Kutzen [Knu09] used L-system to model climbing plants that react to gravity and sunlight. Recently, Wong and Chen [WC15] generate climbing plants with a focus on procedural modeling and the behavior of tendrils that grow around objects. While we try to incorporate such behavior in our system, a particle-based approach can only do this to a limited extent. In contrast, we couple the plant growth with fluid interaction and more refined tools for artistic authoring. A more thorough overview of the biomechanic behavior of climbing plants can be found in Goriely and Neukirch [GN06].

Particle systems have been used in Computer Graphics to capture a wide range of phenomena. Reeves [Ree83] introduce particle systems to simulate fire and trees. Close to our approach is the work of [AK06] that simulates tree dynamics by particles using a

mass-spring system. However, this approach requires an enormous amount of particles that limits the size of the used model. Pirk *et al.* [PNH*14] simulate the effect of wind onto trees by coupling a particle-based fluid simulation with a model for tree dynamics. Similarly, Selino *et al.* [SJ13] integrate spherical particles that approximate a tree structure within a fluid solver to simulate the interaction between trees and wind. Both methods uses a simplified physics model that does not resolve plant collisions.

Müller *et al.* [MHTG05] simulate deformable objects by a meshless approach where objects are represented by point clouds. To restore deformed objects, generalized shape matching of a deformed state with an un-deformed rest state is used. We extend this approach by biologically-motivated attributes and use them to simulate the plant growth with full interactive control.

Becker *et al.* [BIT09] simulate deformable solids using Smoothed Particle Hydrodynamics (SPH). While being able to simulate a variety of different materials, they report a lower stability compared to shape matching approaches. Müller and Chentanez [MC11] extend shape matching by incorporating oriented particles. Particles can be anisotropic, which allows to approximate the shape of an object better than by using isotropic particles. We exploit this method for the simulation of climbing plants as they allow for an economic particle approximation of rod-like structures such as tree branches. Their robustness makes them suitable for interactive applications.

Our approach builds on and extends the previous work. However, it is the first attempt to an integrated system that allows to model realistically behaving plants in an interactive setting, which remains a challenging problem.

3. Overview

Plants grow primarily by extending their tip (apical growth) or by branching from lateral buds caused by the meristemic cells [Leo64]. The cambial (secondary) growth extends the branch radius and forms annual rings [Bai23, KSG*15]. We describe the plant as a set of environmental-sensitive oriented anisotropic particles that branch by generating new particles. Connections between particles form branches that develop at the extremities of the existing apices or by shoots formed from lateral buds. The connection of branches forms the plant skeleton and the branch geometry is represented by generalized cylinders. Plant geometry and growth can be controlled interactively at different stages of the simulation (Figure 2).

Plant modeling starts with either manual or automatic positioning of seed particles in the scene. The plant is represented by anisotropic particles that form branch segments with attached leaves. End particles correspond to branch apices and branching is achieved by generating particles that correspond to lateral branches. In the first step we update the branching structure. We simulate plant dynamics by a geometry-based method [MC11] that provides plant responses due to the external forces. In addition, we employ the physically-based Smoothed Particle Hydrodynamics (SPH) method [Mon92] to simulate fluids. A two-way coupling is achieved by jointly simulating the response of branch and leaf

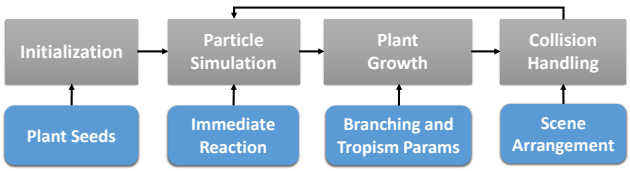


Figure 2: *Our system consists of a simulation loop (upper row) and the interaction (lower row) during each step.*

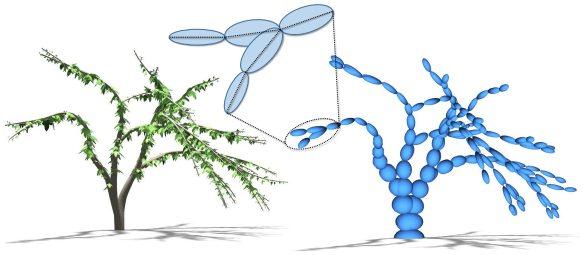


Figure 3: *The shape of a plant is defined by anisotropic particles that are attached to each other. The main axis of the particles form a graph corresponding to the plant skeleton.*

particles with the fluid dynamics. An emitter can be used to create a wind stream that affects the plant.

During growth the end particles (the apices) move in space by using a directed random walk which is influenced by environmental conditions (sunlight, proximity to support, gravity, etc.). Lateral branches are produced by generating new particles from the existing ones. This corresponds to lateral branching at the end of a shoot. The development of branches and leaves is parameterized; we can alter angle variance, growth speed, branching probability as well as environmental parameters such as light sensitivity or gravity. Leaves are also represented as particles and they are attached to branches. Collision of particles with scene geometry is detected and resolved to avoid object penetration.

We present various interaction tools that can be classified into three groups: seeding tool, direct (dragging/cutting), and indirect deformation (paint attracting area, light, and wind). The user can interact with the plant at all developmental stages and during all steps of our pipeline. New plants can be added to the scene by placing seed points on the ground or on surface meshes. We enable low level control by allowing to initiate individual branches by clicking on the existing plant geometry. All branches can be dragged, bent, and moved around, while they follow physical constraints; if the stress is too large, they break and fall down. Plants have species-dependent material properties, such as the stiffness that vary as a function of age. Moreover, the scene can be modified by inserting new or by moving existing objects. To guide the growth process, the user can sketch the preferred areas of attraction on the object surface. The simulation can be stopped and restarted at any time.

4. Climbing Plants

Climbing plants have a branching structure similar to trees. However, an important difference is that they use support structures to

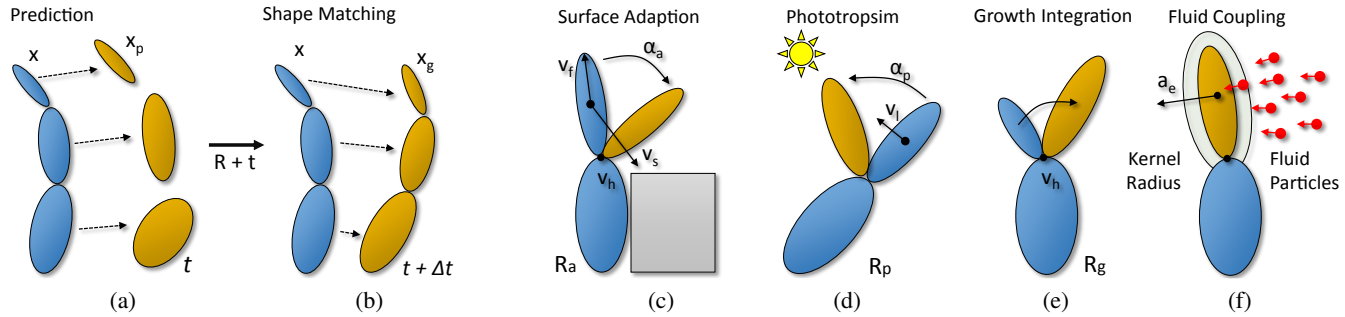


Figure 4: *Shape matching: new particle positions are predicted in the first pass (a). The deformed shape is then matched with the original shape and the particles are moved to the new positions (b). The environmental feedback of a plant simulated as anisotropic particles. The two contributions of surface adaption (c) and phototropism (d) are integrated into a new growth position and orientation (e). The coupling of plant particles with fluids is realized by computing an acceleration term from fluid simulation that adjusts the plant particle orientation according to the forces of fluid particles detected in the kernel radius (f).*

grow and thus can invest more resources into their branches than standing trees, which have to provide enough material to stand upright on their own. Most climbing plants such as ivy or grape, can be modeled by the Leeuwenberg's plant model [HOT78] in which the apical bud dies and produces two lateral buds. One of them becomes a new leading apex and the other produces the organ that attaches the plant to the supporting object. We approximate this behavior by using particles. While previous models use spherical particles that colonize space [AK88,Gre89,BM02], our branch segments are represented by anisotropic oriented particles that together define the branch shape (Figure 3).

In each time step we first simulate plant dynamics by a geometry-based approach (Section 4.1). This step modifies the existing plant geometry according to user interactions and external forces such as gravity, or fluids. New geometry is added by simulating the primary growth of the plant's shoots. This is achieved by growing apical particles by either extending the existing branches or by adding new lateral branches. During growth, each particle reacts to environmental conditions (Section 4.2). Moreover, secondary (cambial) growth is simulated by increasing the diameter of the particles inside the branch skeleton. Another view to the branch thickening is provided by the pipe theory [Hol94], where each branch is considered to be a collection of strands. Branches thicken by splitting an individual strand into two and they branch by splitting a collection of strands into two or more sets.

The plant geometry can be represented on the simplest level by a *plant skeleton* that is a set of polylines with branching. For example L-systems [Pru86] encode this as a linear list of modules with parenthesis representing branching. In our representation, the plant skeleton corresponds to a chain of connected anisotropic particles. Each particle has its position and orientation and the main axis form the plant skeleton (inset of Figure 3).

4.1. Plant Dynamics

Our particle-based plant representation is based on the *shape matching approach* introduced by Mueller et al. [MHTG05]. Particles carry quantities such as the position and orientation for their current state as well as the rest state. We update the particle attributes in each time step. The plant shape can be modified, for example by pulling the branches to a different location, and the *shape matching* algorithm restores the initial plant shape.

Plants are represented as oriented particles that implicitly define a graph of the branching structure. For each particle we maintain a particle group that connects a particle to its adjacent particles. The particle group contains the current particle, the parent particle, and each of its successors, which produces a hierarchical relationship for the branching structure. A particle usually appears in multiple groups as it is in the particle groups of its adjacent particles. Updating particle positions is performed sequentially and once per time step in the following order: for each particle we first compute a predicted position and orientation which only depend on local particle quantities. Then a rotation matrix that matches the current positions is computed using generalized shape matching. We calculate the target position after a modification for the particle itself as well as for its particle group. The final goal position of a particle is a weighted sum of the target position computed by the particle itself and target positions computed for the evaluated particle by members of the particle group (Figures 4 (a, b)). The computations are performed sequentially and the particles do not need to be sorted.

In the first step we compute a predicted position $\mathbf{x}_p$ for all particles as:

$$\mathbf{x}_p = \mathbf{x} + \mathbf{v}\Delta t + \frac{(\mathbf{a}_g + \mathbf{a}_e)\Delta t^2}{2}, \tag{1}$$

where $\mathbf{x}$ is the particle position, $\mathbf{v}$ is the particle velocity, Δt is the simulation step, $\mathbf{a}_g$ and $\mathbf{a}_e$ are the gravitational acceleration and external acceleration caused by fluid particles. The predicted orientation $\mathbf{q}_p$ of a particle is computed by using Equation 16 (Appendix).

In the next step the *shape matching* computes the optimal rotation $\mathbf{R}$ that matches the rest state to the current state of each particle group. The rotation is computed via a polar decomposition from the moment matrix $\mathbf{A}$ (Equation 14). The moment matrix depends on the mass m of each individual particle, which is computed from an ellipsoid with volume V and density ρ

$$m = V\rho = 4\pi abc\rho/3, \tag{2}$$

where a, b, c represent the axes of the ellipsoid. Having computed the optimal particle rotation, we can calculate the target position

for each particle group as

$$\mathbf{x}_t = \mathbf{R}(\bar{\mathbf{x}} - \bar{\mathbf{c}}) + \mathbf{c}, \tag{3}$$

where $\mathbf{c}$ is the center of mass of the particle group, $\bar{\mathbf{c}}$ the rest center of mass, and $\bar{\mathbf{x}}$ the rest position. The final goal position of a particle is the weighted sum of the target positions of the particle itself and its adjacent particles:

$$\mathbf{x}_g = \sum_i w^i \mathbf{x}_t^i / W, \tag{4}$$

where W is the sum of all weights that are proportional to the mass and the position in the branch and w is the individual weight. The goal position is computed once per time step. Parent particles are given a higher influence than the child branches in order to let particles always follow their parents. Other particle attributes, such as velocity, angular velocity and orientation, are updated from their predicted values by an integration scheme as described in [MC11].

A key advantage of a physics-based simulation is that it facilitates to model the ability for plants to attach to objects. For this we provide anchor points for particles that are attached to the surface of a support structure. Anchor points are created dynamically when a growing particle touches an object with an attraction strength linearly proportional to the particle size. The particle position is corrected to the new position $\mathbf{x}'_p$ by nearby anchor points so that it sticks to the surface:

$$\mathbf{x}'_p = \mathbf{x}_p + \phi(\mathbf{x}_{anchor} - \mathbf{x}_p), \tag{5}$$

where $0 < \phi < 1$ controls the attachment strength that increases over time as the particles get bigger. If a particle position exceeds a distance threshold from an anchor point the anchor point is released.

It is important to note, that unlike to previous methods, such as Jirasek et al. [JPM00] or Taylor-Hell [TH05], our implementation does not provide a biomechanically realistic simulation of plant dynamics. Instead we focus on visually plausible behavior that can be computed in real-time.

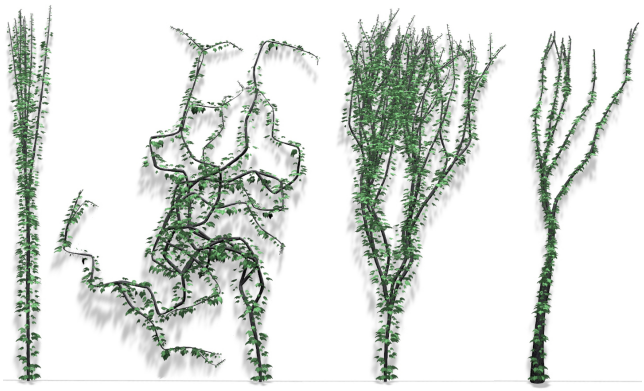
4.2. Plant Growth

After the particle dynamics is evaluated, the growth of the apical and lateral particles is calculated [Leo64]. Within each time step all particles at the end of the plant's shoots increase their size until a maximal size is reached. The growth rate depends on the amount of light at the particle position that can additionally be controlled by the user.

Surface Adaption. The plant searches for a nearby support structure and directs its growth towards this direction. If a plant particle approaches an object, the plant orients itself parallel to the surface. The axis $\mathbf{a}_a$ and the rotational angle α_a (Figure 4 (c)) are:

$$\begin{aligned} \mathbf{a}_a &= \hat{\mathbf{v}}_s \times \hat{\mathbf{v}}_f \\ \alpha_a &= (\hat{\mathbf{v}}_s \cdot \hat{\mathbf{v}}_f)\tau\Delta t, \end{aligned} \tag{6}$$

where $\mathbf{v}_s$ is a vector pointing to the closest surface, $\mathbf{v}_f$ the current forward vector of the particle, and $0 \leq \tau \leq 1$ is a user-defined parameter that controls the surface adaption strength.



(a) (b) (c) (d)

Figure 5: *The user modifies branching probability and the direction of the lateral branches. Low branching direction probability (a), high branching probability and direction variance (b), high branching probability with a low direction variance (c), low branching probability and medium direction variance (d).*

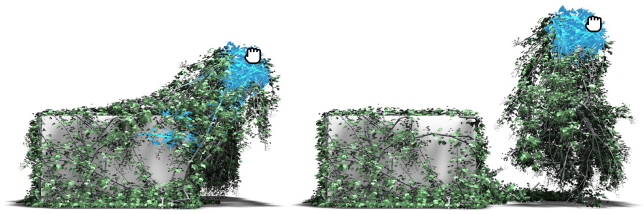


Figure 6: *Branch bending and breaking by user interaction. The plant is dragged by the user until branches break (branches under large amount of stress are highlighted in blue). The separated branches maintain plausible behavior and are re-used for modeling or animation purposes.*

Phototropism is the plants' response to light that orients plant organs (leaves and flowers) towards the light direction and help the apices reach areas with more intensive illumination. Similarly to the previous case, the rotation axis $\mathbf{a}_p$ and angle α_p are:

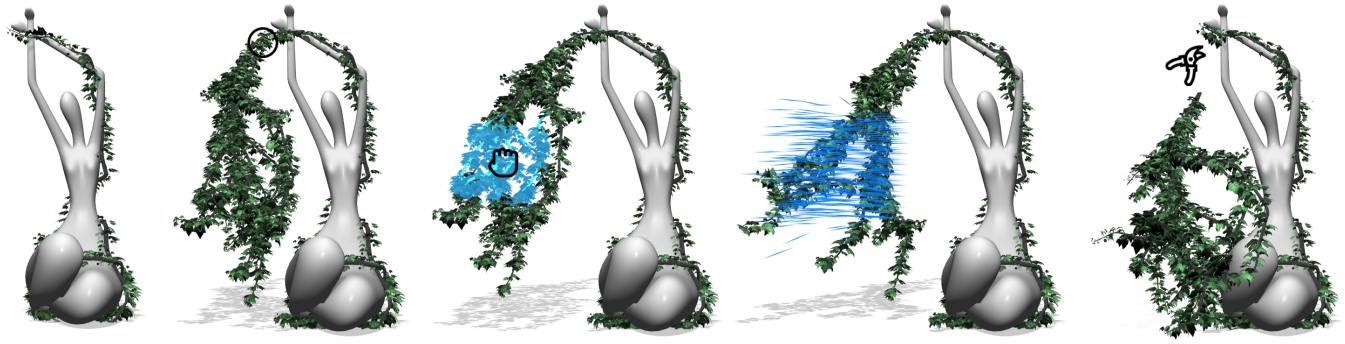
$$\begin{aligned} \mathbf{a}_p &= \hat{\mathbf{v}}_l \times \hat{\mathbf{v}}_f \\ \alpha_p &= (1 - O)\eta\Delta t, \end{aligned} \tag{7}$$

where $\mathbf{v}_l$ is the vector to the light source, O is the light occlusion at the particle location, and η is a parameter that controls the phototropism response strength (Figure 4 (d)).

Growth Integration. In the last step we integrate all changes and compute the new particle position. The dynamic simulation approach (Section 4.1) uses the shape of the plant in a rest state to partially recover it from deformations. Therefore, to incorporate plastic changes into the plant, the rest states as well as the current state have to be updated; we do this similarly to the growth deformation proposed by Pirk et al. [PNH*14]. An accumulated rotation matrix $\mathbf{R}_g$ incorporating the orientation changes is computed from the rotation matrices using the previously computed axes:

$$\mathbf{R}_g = R(\mathbf{a}_a, \alpha)\, R(\mathbf{a}_p, \alpha_p), \tag{8}$$

where $R(\mathbf{a}, \alpha)$ returns a rotation matrix for the given axis $\mathbf{a}$ and



(a) (b) (c) (d) (e)

Figure 7: *A set of interactive editing operations performed with our framework: a plant grows after placing a plant seed on the ground close to on obstacle (a). The user manually adds a branch by clicking on the plant (b). The entire plant remains animation-ready at all stages, branches can be grabbed (c) and suffer from stress, while the plant is coupled with fluid dynamics (d). Finally, the user decided to remove the branch by using a cutting tool (e), separated branches fall on the ground and disappear.*

angle α (Figure 4 (e)). Finally, all particle orientations are updated by integrating growth rotations and size changes into the current and the rest state:

$$\begin{aligned} \mathbf{R} &= \mathbf{R}\mathbf{R}_g \\ \bar{\mathbf{R}} &= \bar{\mathbf{R}}\mathbf{R}^{-1}\mathbf{R}_g, \end{aligned} \tag{9}$$

where $\mathbf{R}$ and $\bar{\mathbf{R}}$ are the current and rest orientations represented as matrices. Similarly, the particle positions in its current state as well as its rest state are updated by

$$\begin{aligned} \mathbf{x} &= \mathbf{x}_h + \bar{\mathbf{R}}\mathbf{R}\mathbf{u}_f \\ \bar{\mathbf{x}} &= \bar{\mathbf{x}}_h + \bar{\mathbf{R}}\mathbf{u}_f, \end{aligned} \tag{10}$$

where $\mathbf{x}_h$ and $\bar{\mathbf{x}}_h$ are the head positions of the parent particle and the forward vector $\mathbf{u}_f = \begin{bmatrix} 0, & 1, & 0 \end{bmatrix}^T$. If a resting position or current particle position change the center of mass for a particle, each group that contains the particle has to be recomputed.

4.3. Species and Material Properties

Although we do not primarily focus on a biologically correct plant modeling, our method includes a number of features that are biologically-inspired and they showcase the versatility of our framework. We model different species and material properties, branch and leaf development, and time-variant material properties.

Branches. To model different species the user can influence the growth direction and the maximum number of new particles that are generated at each branch. This defines the main growth behavior and affects the resulting shape. Moreover, the user can set the probability for the number and direction of lateral branches that are generated. Each particle can grow a number of child particles that are initiated with a random orientation relative to the parent. The branching variance parameter, with the range [0, 1], controls the maximal change of orientation and is maximally 90 degrees. The thickness of a new branch is controlled by a falloff parameter that defines the thickness w.r.t. its parent: $t_c = t_p.f$, where t_c and t_p are the thickness of the parent and child branch respectively and f is the falloff parameter. Figure 5 shows examples of specific settings that control the branch development.

Leaves. We implemented two modalities for modeling and animating leaves. For plants with small branching structures (under 10k particles) we model each individual leaf with a single particle, which allows for more realistic animation and collision response. Leaf particles are integrated with the rest of the simulation and affect the motion of branch particles. Updates of branch particles are propagated to their associated leaf particles. In more complex plants each leaf is assigned to a branch particle that updates the leaf with the updates of its transformations.

Leaf growth and position (phyllotaxis) are controlled by two parameters for the angle and step size on the parent branch. Additionally, we alter the orientation of leaves randomly by a small amount to create more variance. The growth speed of a leaf is relative to the growth speed of the branch particle. We also adjust the leaf orientation according to the incoming light to simulate phototropism (Section 4.2).

Stiffness and Branch Breaking. Our position-based approach allows to directly modify particle positions. We exploit this to model the stiffness of a branch by defining how far a particle can move from its predicted position to its goal position. We expose a stiffness parameter that controls this behavior: $\mathbf{x}'_g = \mathbf{x}_p + s(\mathbf{x}_g - \mathbf{x}_p)$, where $\mathbf{x}'_g$ is the updated goal position, $\mathbf{x}_p$ the predicted position, $\mathbf{x}_g$ the goal position, and the stiffness parameter s controls the elasticity of the plant. We automatically vary the stiffness according to the age of the branch during simulation [RBTT07]: $s = t_l/t_m$, where t_l is the lifetime of a given particle and t_m the time a particle reaches its maximum stiffness, s is clamped to the range $[0,1]$. For our experiments we initialized $t_l = 0.2$ and increase it over 10 seconds.

Breaking can occur because of the environment such as gravity or as a result of user interaction. The user can select one or multiple branches and direct the growth by bending them. The stress on a branch segment is measured by the difference of the distance between two particles in their rest state and the current state invoked by the force. If the stress exceeds a user-defined threshold that reflects the material properties, particles are removed from each other's particle group list and the corresponding branch breaks.

5. Authoring

Our particle-based representation allows for the modeling of external effects that can be induced at any time without prior analysis of the plant structure. Our system exploits this representation to provide powerful editing capabilities that allow to edit a plant with respect to its structure and its environment while maintaining a biologically plausible appearance. Moreover, we couple plants with fluid simulation by using Lagrangian systems, and model advanced effects, such as the breaking and bending of branches.

5.1. Dynamic Editing

One of the key objectives of our method is to support user-friendly interactive plant editing. The idea is to modify a plant while it obeys biological and physical constraints. This allows to easily produce plant models according to artistic requirements while maintaining a realistic appearance. The user can place a plant seed anywhere in the scene and it immediately starts to grow and interact with its environment.

To further expose plant growth as a means for editing, we implemented the seeding of new branches. The user can click on an existing branch and define a branch seed that initiates the growth of a new shoot from that location. If the user seeds a branch while the plant is growing the new branch grows jointly with all the other branches, otherwise the new branch keeps growing until the button is released. The plant geometry (width of the trunk, branching angles, etc.) is modified accordingly (Figure 7).

To guide the developmental process we provide a brush tool that allows to paint regions on obstacles that either attract or repulse the plant's growth. The user can directly sketch on the 3D surfaces (Figure 8). Each sketches defines sample points associated with a distance function that specifies the radius and strength of the attractor. A growing plant particle checks for attractors in its vicinity and adjust its growth direction based on the weighted average of the nearby attractors (Section 4.2). The coupling of initiating new branches while using the sketch tool provides an interesting means for modeling and authoring plants. A new sketch in the vicinity of a fully grown particle automatically triggers a new branch seed for this particle, causing the plant to develop into the sketched region.

Finally, it is possible to delete branches and branch parts. By clicking on a segment, we identify the consecutive particles of the branching structure and remove them from the simulation. Moreover, we define an undo operation that allows to revert the previous operations.

5.2. Collision Response

A key advantage of the particle-based modeling of plants is that it allows to easily resolve collisions. Unlike the previous approaches, our systems resolves the collision of a plants with obstacles, other plants, and its own organs. Plant-plant collisions are resolved by the position-based approach described in Mueller et al [MC11]. We compute the contact point of two particles and displace them along their normal until they no longer intersect.

To resolve the collision of plants and obstacles, we assign each shape a signed distance field (SDF). A collision is resolved by tracking how plant particles approaches the surface of an obstacle. We used the longest axis of a particle and compute the distance of its position and the surface stored in the SDF. If the distance is smaller then the length of the longest axis, we move the particle by computing the reflection vector of the particle direction and the surface normal. Affine transformations of the obstacle can also be applied to the distance field, allowing to freely move objects around in the scene. Moreover, we simulate friction and elasticity by adapting the particles change in velocity when it collides with an obstacle. We decompose the particle velocity in a tangential part and multiply it with a friction value in the range of $[0,1]$. To model the effect of elasticity we multiply the normal part with another value in the range of $[0,1]$. In both cases zero represents the lowest and one the highest response to these effects.

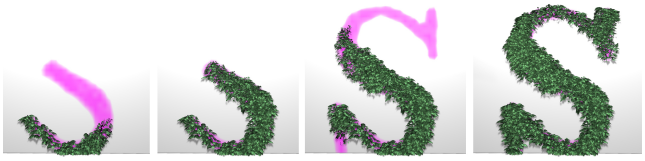


Figure 8: *An example of an attracting layer sketched on the geometry of a support structure (here a wall). The attractors guide the plant growth and is used to author plants to artistic requirements, while maintaining a biologically plausible appearance.*

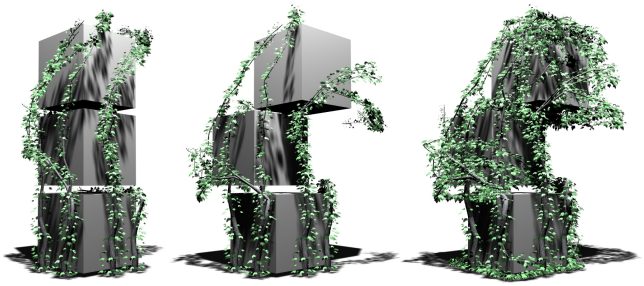


Figure 9: *The interaction of plants and cubes: each plant resolves collisions with the object, other plants, and itself. The objects can be moved and the plants dynamically adapt to changes in their environment while they continue to grow.*

Figure 9 shows an example of the collision response in our framework. A user assembles a scene of obstacles. Plants start to grown and use the obstacles as support structures. The user can move the objects and the plants dynamically adapt.

5.3. Two-way Fluid Coupling

Representing plants as connected sets of particles facilitates the coupling with Lagrangian simulations. We couple our plants with a fluid simulation; in particular a wind field is simulated by using Smoothed Particle Hydrodynamics (SPH) [Mon92]. Collisions are handled by particle-cylinder intersection tests. Macklin et al. [MMCK14] incorporated isotropic solid and fluid particles into a unified solver, avoiding additional collision tests. Similarly we incorporate branch and fluid particles into the SPH solver and extend this approach by taking into account anisotropic branch particles.

SPH represents the movement induced by a fluid through advec-

Figure 10: *An object covered by three climbing plants with thousands of leaves. The user can grow, grab, and cut individual branches or groups to author the plant to her artistic needs. Branches that suffer from stress are highlighted in red.*

tion of a set of independent particles that carry physical quantities, *e.g.,* pressure and mass. The acceleration $\mathbf{a}_i$ of the $i-$th particle is computed as

$$\mathbf{a}_i = \frac{d\mathbf{v}_i}{dt} = (-\nabla p + \mu\nabla^2\mathbf{v})/\rho, \tag{11}$$

where $-\nabla p$ is the pressure, $\mu\nabla^2\mathbf{v}$ is the viscosity, and ρ the density. Density ρ is a user-defined parameter that corresponds to the densities of wood and the fluid. We used values in the range of $0.3 - 1.0 \cdot 10^3$ kg/m^3 for wood and $1.3 - 1.4$ kg/m^3 for air. The acceleration is added as an external influence within the position prediction step (Equation 1) (see Figure 4 (f)). New fluid positions are computed by using Eulerian integration. Interacting forces are only computed between fluid particles and full grown branch particles to avoid big density differences which can lead to instability within the simulation. Particles are influenced by other particles within their support radius h. Fluid quantities $A(\mathbf{x})$ at a certain location $\mathbf{x}$ are computed as a weighted sum of neighboring particles j:

$$A(\mathbf{x}) = \sum_{j=1}^{N} V_j A_j \, W(\eta), \tag{12}$$

where V_j is the volume and W is a smoothing kernel with normalized position vector $\eta = (\mathbf{x} - \mathbf{x}_j)/h$. However, for ellipsoidal particles the support radius is not a scalar. As proposed by [OVSM97] we map the position vector from real to normalized position space by

$$\eta = \mathbf{G}(\mathbf{x} - \mathbf{x}_j), \tag{13}$$

where the tensor $\mathbf{G}$ performs a linear transformation. Forces are computed in the same way as in Müller et al. [MCG03].

Pirk et al. [PNH*14] introduced an integrated systems that simulates the effect of wind on trees by using a SPH-based fluid simulation. Unlike the previous approaches, our contribution is in extending plant-fluid interaction to a position-based physics simulation. Operating only on particles, as opposed to coupling particles with rods or even rod hierarchies, is more robust and enables the concurrent processing of plant structures. Moreover, jointly simulating plant and fluid particles natively supports a two-way coupling of fluid and plant dynamics. Our unified solver simulates the bilateral interactions without abandoning the SPH-paradigm and thus allows for a more efficient collision handling.

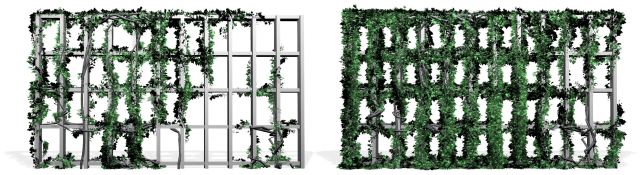


Figure 11: *Plants growing on a fence not only exploit the features of the support structure, but also interact with each other by competing for light coming from the upper left corner. Collision are resolved for plant organs, neighboring plants, and obstacles.*

6. Implementation and Results

Our system is implemented in C++ and uses GLSL and OpenGL for rendering. All results were generated on a desktop computer with an Intel i7 processor clocked at 4.0 Ghz, 16 GB of RAM, and a NVIDIA GeForce GTX 970 graphics card. The particle simulation is calculated on the GPU by using CUDA 7.5.

The branch mesh is dynamically generated in the geometry shader and is built on a frame-to-frame basis by creating a cylinder mesh between two adjacent particles, where the radius of the particle defines the radius of the cylinder. The orientation of two subsequent particles defines a reference frame for the start and the end of the cylinder. Leaves are represented as points that are trans-

formed into curved surfaces in the geometry shader during rendering. The size of leaves and branches can be dynamically adjusted and the entire tree mesh can be read-back from the GPU for offline rendering purposes. Unlike other methods [PHL*09] we do not explicitly generate a tree graph for the simulation, but instead directly operate on the connected particles. If necessary, it is possible to generate the graph from the adjacency lists for storing the generated plants or for other applications.

All particles are stored in a 1D array on the GPU and labeled as active and inactive. When an active particle reaches its maximum length we pick a previously allocated, but inactive, particle from the buffer and connect it as a successor to the current particle. This updates the particle groups of the participating particles. The newly enabled particle is initialized and then updated as part of the simulation.

Shadows are computed by using variance shadow maps [DL06] that are also used to compute the light particle occlusion for phototropism (Section 4.2). A signed distance field is computed for each obstacle in the scene to resolve collisions with particles and to retrieve the closest surface position for the growth adaption.

In our system we advance the simulation of fluids and physics response with a time step of $t = 25ms$. Plants grow very slowly and real-time growth does not make much sense. Therefore, the *growth speed* for plants is a user-defined multiple of t.

Table 1 provides performance measurements for the figures shown in the paper. Our methods supports real-time performance for the simulation of plant behavior that makes it well-suited for applications where interactivity and direct response are important. Moreover, it enables the efficient content creation as the plant model can be exported as static mesh at any time. Table 2 lists the parameters used for the results shown in the paper. The user seeds a number of plants (N) and adjusts the parameters to produce plants with the desired appearance. It is possible to control the maximum number of lateral buds (B), the branch probability (BP), the branching variance (V) and a weight for the applied Phototopism (Ph). The stiffness for branches was set to a value of $s = 1.0$ for most of our tests.

6.1. Results

Our simulation enables various ways for the user to interact with the plant (Figure 2, lower row). The user can guide plant development by modifying parameters of the underlying growth model or by changing the scene setting.

In particular, the user can influence the strength by which the plant sticks to the support structure, how quickly plant growth diverges from the surface, the number of leaves, and the amount of fluid emitters. Furthermore, plants can be edited by either bending or breaking their branches or by seeding new branches anywhere on the plant. To efficiently author a plant our system provides a sensitivity threshold $t_s \in [0, 1]$, to control the breaking of branches. Setting $t_s = 0$ disables the breaking of branches and allows users to refine the position of a grown branch; a value of $t_s \geq 0$ enables the branch breaking with the specified sensitivity.

The effect of tropisms and material properties can be seen in Figure 12. Various seeds were planted in different light conditions.

Figure 12: *The effect of tropisms and material properties: phototropism causes plants exposed to more light to grow faster and also towards its direction (top). Gravity pushes branches downwards, but phototropism and an increased branch stiffness lead to an upward growth allowing the plant to grow away from the support structure (bottom left). If the stiffness is low, branches need the support and thus can only grown on the surface (bottom right).*

Fig.	NP (k)	T (ms)	R (%)	P (%)	G (%)	C (%)
1	10	18,16	27,65	29,38	42,42	0,55
7	plant: 5 fluid: 1	20,40	26,01	7,83 43,85	22,12	0,19
9	5	9,66	44,42	14,11	41,07	0,40
11	10	17,05	29,46	29,68	40,37	0,48
10	7	16,82	26,87	34,17	38,46	0,50
14, l	10	16,87	29,77	33,55	35,08	1,60
14, r	25	51,76	11,63	39,51	48,33	0,53

Table 1: *Performance measurements for our experiments, where NP denotes the number of particles, T the total simulation time, and the respective percentage for rendering (R), physics (P), growth (G) and obstacle collision handling (C). The timings indicate that modeling and authoring can efficiently be done in real-time.*

Phototropism causes some branches to grow slowly due to the lack of light and others to grow faster and towards the light source (Figure 12, top). The effect gravity and stiffness acting upon a set of plants is shown in Figure 12 (bottom, left). Branches grow on the surface and eventually reach out of the support area, which causes them to bend down because of gravity. Increased branch stiffness allows them to grow without support. The branches still follow the light because of phototropism, once their weight exceeds the breaking limit, they break and fall down. If the branch stiffness is low, branches need the support and thus can only grow directly on the surface (bottom, right)

Figure 6 shows two frames from an interactive sequence where a user takes an existing plant (left) and drags its branches (right). As soon as the breaking threshold is reached the particles disconnect, the plant breaks, and the pieces fall down due to gravity. The sensitivity of the branch breaking can be individually controlled for each plant, which enables an interesting new way to model vegetation.

Fig.	N	B	BP	V	Ph
1	6	2	0.4	0.2	0
7	1	1	0.1	0.1	0.1
9	4	1	0.1	0.2	0.5
11	8	1	0.2	0.1	0.5
10	5	1	0.2	0.2	0.5
14, l	3	1	0.1	0.1	0.7
14, r	20	1	0.3	0.2	0.5

Table 2: *Parameter settings for the results shown in the paper. N denotes the number of plants, B the maximum number of lateral branch particles, BP the branching probability, V the branching variance, and Ph the weight for the Phototropism. For most of our experiments we used a stiffness of s=1.*

The example in Figure 11 shows a scene where multiple branches compete for a support structure and for light coming from the upper left direction. Similarly, examples in Figures 10 and 14 show complex scenes with multiple influences applied at the same time generated by using our system. These results illustrate that our system allows the efficient modeling and rendering of complex climbing plants while interacting with intricate support geometry.

The accompanying video demonstrates several examples of modeling climbing plants with our system. Specifically, the two-way coupling of plants and wind and the interaction with complex support geometry allow for interesting effects that could not be realized with previous approaches. A plant tends to grow attached to a supporting object but the growth is being disrupted by wind coming from the left which results in an updated dramatic plant shape.

6.2. Evaluation

Trees and plants are dynamically evolving systems that continuously react to their environment. Understanding the morphogenesis and stress response of plants are actively studied topics in botany and forestry research. However, to our knowledge no dataset exits that would allow to perform an evaluation of our growth model and physics response to real-world data.

To show that our system provides an efficient means for the realistic modeling of climbing plants we compare our results to photographs of real climbing plants. Figure 13 (top row) shows three photographs of climbing plants in different settings and the bottom row shows virtual scenes produced with our system. The example on the left shows a real grape vine plant growing on a wall. Using the physics response of the plant model and its interaction with obstacles, our system allows to replicate realistic virtual scenes. In the center and on the right we show two examples of plants growing around support structures at different scales. The combination of modeling plant growth, physics response, and artistic tools (local branch control) allows to efficiently control the plant behavior and quickly achieve artistic intent. The user can dynamically control the appearance of the plant by defining the size of branches and leaves and by selecting different textures. For the center and the right scene we used the photographs as a background image. All scenes in this figure were rendered with an offline renderer by storing the plant mesh after completing the modeling process.

Figure 13: *A visual comparison of photographs of real climbing plants (top) and a similar scenes produced with our system (bottom). The combination of modeling plant growth, physics response and artistic tools allows to efficiently control plant behavior to replicate realistic looking scenes. Photographs courtesies of G. Jarkovska (bottom left), F. Kuipers (bottom center), public domain (bottom right).*

Compared to the work of Wong and Chen [WC15] we do not model the effect of tendrils searching for support structures and the pulling of branches upon contact. However, as shown in Figure 13 our method allows modeling coiling effects similar to their approach.

7. Discussion and Limitations

One limitation of the implementation of our system is the effect of global control. While the parameters provide good local control and they are fairly intuitive, their mutual interaction and global effects are difficult to predict. The effect on the entire plant, which is a complex system with hardly predictable emergent phenomena, may not be obvious. Targeting plant shape is a known difficult problem [TLL*11, LRBP12].

Another limitation is that our current growth model does not offer the same complexity as state-of-the-art work in this field. Although our system can produce a variety of climbing plants, it does not allow to model more nuanced developmental properties of similar species. Moreover, our system does not provide a biomechanically-plausible simulation of tree behavior, *e.g.,* as proposed by Jirasek et al. [JPM00] or Taylor-Hell [TH05]. Instead, we carefully balance artistic requirements with a plausible physics response. This allows to efficiently produce visually plausible, but not always realistic effects. In our system this limitation is apparent for the breaking of branches; applying stress to the branches may result in an unrealistic stretching. The stretching can be reduced by sequentially fixing predicted positions, but in order to ensure interactive frame rates for bigger plants we only perform it once.

Apart from the visual comparison to real plants, we did not conclusively evaluate our approach. While we aimed at developing an interactive method, our framework is also inspired by biology and it would be interesting to provide a more thorough evaluation and comparison to real plants w.r.t. the different time scales used in our system. However, to the best of our knowledge no dataset exist to provide a baseline for plant growth and physics response for climbing plants.

8. Conclusion and Future Work

We have presented a novel system for the interactive modeling of growing climbing plants. Our approach provides an efficient means for the control over plant development by allowing the user to affect growth parameters and physical properties of the plant. This is enabled by a meshless formulation of plant geometry as a set of connected anisotropic particles. We show that our approach handles efficient modeling of external effects that can be induced at any time without prior analysis of the plant structure. We exploit this representation to provide powerful editing capabilities that allow to modify a plant with respect to its structure and its environment while maintaining a biologically plausible appearance. We show the efficiency of our approach on a wide variety of interactive examples.

There are several possible avenues for future work. First, particle-based and meshless deformation methods offer intriguing advantages for modeling trees and plants. In applications that require to balance realistic behavior and artistic requirements, such as in video games and movies, particle systems seem like a plausible representation, especially, as the required computations can efficiently be performed in parallel. Therefore an interesting avenue of future research is to explore these techniques with a stronger focus on biological and physical plausibility. Second, particles provide a means for locally controlling physical and biological quantities. Our system does not fully exploit the possibilities of this setting and it would be interesting to use particles for the efficient modeling of secondary growth, e.g. the development of growth rings, or more advanced physics, such as the cracking of bark.

Acknowledgements

We thank the anonymous reviewers for their constructive suggestions. The work was supported by the Stanford AI Lab-Toyota Center for Artificial Intelligence Research, the Max Planck Center for Visual Computing and Communications, and the National Foreign 1000 Talent Plan (WQ201344000169) and Leading Talents of Guangdong Program (00201509).

References

[AK84] AONO M., KUNII T.: Botanical tree image generation. *IEEE Computer Graphics and Applications 4(5)* (1984), 10–34. 2

[AK88] ARVO J., KIRK D.: Modeling plants with environment-sensitive automata. In *Pro. of Ausgraph* (1988), pp. 27–33. 2, 4

[AK06] AKAGI Y., KITAJIMA K.: A study on the animations of swaying and breaking trees based on a particle-based simulation. *Journal of WSCG 20*, 1 (2006), 21–28. 2

[Bai23] BAILEY I.: The cambium and its derivative tissues. The increase in girth of the cambium. *Amer. J. Botany 10*, 9 (1923), 499–509. 3

[BIT09] BECKER M., IHMSEN M., TESCHNER M.: Corotated sph for deformable solids. In *Proceedings of the Eurographics Conf. on Nat. Phenomena* (2009), pp. 27–34. 3

[BM02] BENES B., MILLÁN E.: Virtual climbing plants competing for space. In *IEEE Proc. of the Comp. Anim.* (2002), IEEE Computer Society, pp. 33–42. 2, 4

[CBD13] CASATI R., BERTAILS-DESCOUBES F.: Super space clothoids. *ACM Trans. Graph. 32*, 4 (July 2013), 48:1–48:12. 2

[DL06] DONNELLY W., LAURITZEN A.: Variance shadow maps. In *Proc. of I3D* (2006), I3D '06, ACM, pp. 161–165. 9

[GN06] GORIELY A., NEUKIRCH S.: Mechanics of climbing and attachment in twining plants. *Phys. Rev. Lett. 97* (Nov 2006), 184302. 2

[Gre89] GREENE N.: Voxel space automata: Modeling with stochastic growth processes in voxel space. *SIGGRAPH Comp. Graph. 23*, 3 (1989), 175–184. 2, 4

[Hol94] HOLTON M.: Strands, gravity and botanical tree imagery. *Computer Graphics Forum 13*, 1 (1994), 57–67. 4

[HOT78] HALLÉ A., OLDEMAN I., TOMLINSON J.: *Tropical Trees and Forest: an Architectural Analysis*. Springer–Verlag, Berlin/Heidelberg/New-York, 1978. 4

[IOI06] IJIRI T., OWADA S., IGARASHI T.: Seamless integration of initial sketching and subsequent detail editing in flower modeling. *Comp. Graph. Forum 25*, 3 (2006), 617–624. 2

[JPM00] JIRASEK C., PRUSINKIEWICZ P., MOULIA B.: Integrating biomechanics into developmental plant models expressed using l-systems. h.-ch. *Spatz and T. Speck, Plant Biomechanics, Georg Thieme Verlag* (2000). 5, 10

[Kaw82] KAWAGUCHI Y.: A morphological study of the form of nature. *SIGGRAPH Comp. Graph. 16*, 3 (1982), 223–232. 2

[Knu09] KNUTZEN J.: *Generating Climbing Plants Using L- Systems*. Master's thesis, Chalmers University of Technology, 2009. 2

[KSG*15] KRATT J., SPICKER M., GUAYAQUIL A., FISER M., PIRK S., DEUSSEN O., HART J. C., BENES B.: Woodification: User-controlled cambial growth modeling. *Comp. Graph. Forum 34*, 2 (2015), 361–372. 3

[LD99] LINTERMANN B., DEUSSEN O.: Interactive modeling of plants. *IEEE Comput. Graph. Appl. 19*, 1 (1999), 56–65. 2

[Leo64] LEOPOLD A. C.: *Plant growth and development.* New York, San Francisco, Toronto, London: McGraw-Hill Book Co., 1964. 3, 5

[LRBP12] LONGAY S., RUNIONS A., BOUDON F., PRUSINKIEWICZ P.: Treesketch: interactive procedural modeling of trees on a tablet. In *Proc. of the Intl. Symp. on SBIM* (2012), pp. 107–120. 1, 2, 10

[Luf08] LUFT T.: An ivy generator. `http://graphics.uni-konstanz.de/~luft/ivy_generator/`, 2008. 2

[MC11] MÜLLER M., CHENTANEZ N.: Solid simulation with oriented particles. *ACM Trans. Graph. 30*, 4 (2011), 92:1–92:10. 2, 3, 5, 7, 13

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), SCA '03, Eurographics Association, pp. 154–159. 8

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless deformations based on shape matching. *ACM Trans. on Graph. 24*, 3 (2005), 471–478. 3, 4, 13

[MMCK14] MACKLIN M., MÜLLER M., CHENTANEZ N., KIM T.-Y.: Unified particle physics for real-time applications. *ACM Trans. Graph. 33*, 4 (July 2014), 153:1–153:12. 7

[Mon92] MONAGHAN J. J.: Smoothed particle hydrodynamics. *Annual review of astronomy and astrophysics 30* (1992), 543–574. 3, 7

[MP96] MĚCH R., PRUSINKIEWICZ P.: Visual models of plants interacting with their environment. In *Proc. of SIGGRAPH* (1996), ACM, pp. 397–410. 1, 2

[OOI07] OKABE M., OWADA S., IGARASHI T.: Interactive design of botanical trees using freehand sketches and example-based editing. In *ACM SIGGRAPH Courses* (2007), ACM. 2

[Opp86] OPPENHEIMER P. E.: Real time design and animation of fractal plants and trees. *Proc. of SIGGRAPH 20*, 4 (1986), 55–64. 2

[OVSM97] OWEN J. M., VILLUMSEN J. V., SHAPIRO P. R., MARTEL H.: *Adaptive Smoothed Particle Hydrodynamics: Methodology II*. 1997. 8

Figure 14: *Left: a complex support structure is used by a number of plants with thousands of leaves. Note the branch bending due gravity (Water Tank cc-by R. Edwards). Right: a large scene showing plants favoring the illuminated side of a building (High-rise building cc-by S. Keckeis).*

[PBI*16] PIRK S., BENES B., IJIRI T., LI Y., DEUSSEN O., CHEN B., MĚCH R.: Modeling plant life in computer graphics. In *ACM SIGGRAPH 2016 Courses* (New York, NY, USA, 2016), SIGGRAPH '16, ACM, pp. 18:1–18:180. 2

[PHL*09] PALUBICKI W., HOREL K., LONGAY S., RUNIONS A., LANE B., MĚCH R., PRUSINKIEWICZ P.: Self-organizing tree models for image synthesis. *ACM Trans. Graph. 28*, 3 (2009), 58:1–58:10. 1, 2, 9

[PJM94] PRUSINKIEWICZ P., JAMES M., MĚCH R.: Synthetic topiary. In *Proceedings SIGGRAPH* (1994), SIGGRAPH '94, ACM, pp. 351–358. 1, 2

[PL90] PRUSINKIEWICZ P., LINDENMAYER A.: *The Algorithmic Beauty of Plants*. Springer-Verlag New York, Inc., 1990. 1, 2

[PNDN12] PIRK S., NIESE T., DEUSSEN O., NEUBERT B.: Capturing and animating the morphogenesis of polygonal tree models. *ACM Trans. Graph. 31*, 6 (2012), 169:1–169:10. 2

[PNH*14] PIRK S., NIESE T., HÄDRICH T., BENES B., DEUSSEN O.: Windy trees: Computing stress response for developmental tree models. *ACM Trans. Graph. 33*, 6 (2014), 204:1–204:11. 2, 3, 5, 8

[Pru86] PRUSINKIEWICZ P.: Graphical applications of l-systems. In *Proceedings on Graphics Interface* (1986), pp. 247–253. 4

[PSK*12] PIRK S., STAVA O., KRATT J., SAID M. A. M., NEUBERT B., MĚCH R., BENES B., DEUSSEN O.: Plastic trees: interactive self-adapting botanical tree models. *ACM Trans. Graph. 31*, 4 (2012), 50:1–50:10. 1, 2

[RBTT07] RUELLE J., BEAUCHENE J., THIBAUT A., THIBAUT B.: Comparison of physical and mechanical properties of tension and opposite wood from ten tropical rainforest trees from different species. *Annals of Forest Science 64*, 5 (2007), 503–510. 6

[Ree83] REEVES W. T.: Particle systems—a technique for modeling a class of fuzzy objects. *SIGGRAPH Comput. Graph. 17*, 3 (1983), 359–375. 2

[RLP07] RUNIONS A., LANE B., PRUSINKIEWICZ P.: Modeling trees with a space colonization algorithm. In *EG Nat. Phenom.* (2007), Eurographics Association, pp. 63–70. 1, 2

[SJ13] SELINO A., JONES M. D.: Large and small eddies matter: Animating trees in wind using coarse fluid simulation and synthetic turbulence. *Comput. Graph. Forum 32*, 1 (2013), 75–84. 3

[Smi84] SMITH A. R.: Plants, fractals, and formal languages. In *Proc. of SIGGRAPH* (1984), ACM Press, pp. 1–10. 2

[SPK*14] STAVA O., PIRK S., KRATT J., CHEN B., MĚCH R., DEUSSEN O., BENES B.: Inverse procedural modelling of trees. *Comp. Graph. Forum 33*, 6 (2014), 118–131. 1, 2

[TH05] TAYLOR-HELL J.: Incorporating biomechanics into architectural tree models. In *Proceedings of the XVIII Brazilian Symposium on Computer Graphics and Image Processing* (Washington, DC, USA, 2005), SIBGRAPI '05, IEEE Computer Society, pp. 299–. 5, 10

[TLL*11] TALTON J. O., LOU Y., LESSER S., DUKE J., MĚCH R., KOLTUN V.: Metropolis procedural modeling. *ACM Trans. Graph. 30* (2011), 11:1–11:14. 10

[WC15] WONG S.-K., CHEN K.-C.: A procedural approach to modelling virtual climbing plants with tendrils. *Computer Graphics Forum* (2015). 2, 10

[ZB13] ZHAO Y., BARBIČ J.: Interactive authoring of simulation-ready plants. *ACM Trans. Graph. 32*, 4 (2013), 84:1–84:12. 2

Appendix

The *oriented particles* approach introduced by Müller and Chentanez [MC11] allows the efficient modeling of rigid, plastic or even soft bodies. A key component of their method is the *shape matching* algorithm [MHTG05] that determines the least squares optimal rotation $\mathbf{R}$ that matches the rest state to the current state of each particle group. The rotation $\mathbf{R}$ is computed via a polar decomposition $\mathbf{A} = \mathbf{RS}$, where the symmetric part is $\mathbf{S} = \sqrt{\mathbf{A}^T\mathbf{A}}$ and the total moment matrix $\mathbf{A}$ is computed as

$$\mathbf{A} = \sum_i \left(\mathbf{A}_i + m_i \mathbf{x}_i \bar{\mathbf{x}}_i^T - m_i \mathbf{c}_i \bar{\mathbf{c}}_i^T \right). \tag{14}$$

m_i is the particle mass, $\mathbf{x}_i$ and $\bar{\mathbf{x}}_i$ are the current and rest particle positions, and $\mathbf{c}_i$ and $\bar{\mathbf{c}}_i$ are the current and rest centers of mass per particle group. The moment matrix $\mathbf{A}_i$ of an ellipsoid particle with radii a, b, and c is computed as

$$\mathbf{A}_i = \frac{1}{5} m_i \begin{bmatrix} a^2 & 0 & 0 \\ 0 & b^2 & 0 \\ 0 & 0 & c^2 \end{bmatrix} \mathbf{R}. \tag{15}$$

The predicted orientation $\mathbf{q}_p$ for each particle can be computed through an Explicit Euler integration as

$$\mathbf{q}_p = \left[\hat{\boldsymbol{\omega}} \sin\left(\frac{|\boldsymbol{\omega}|\Delta t}{2}\right), \cos\left(\frac{|\boldsymbol{\omega}|\Delta t}{2}\right) \right] \mathbf{q}, \tag{16}$$

where $\boldsymbol{\omega}$ is the angular velocity of the particle and $\mathbf{q}$ is the current particle orientation.

The following integration scheme is used to update particle velocity $\mathbf{v}$ and position $\mathbf{x}$, angular velocity $\boldsymbol{\omega}$, and orientation $\mathbf{q}$:

$$\mathbf{v} = (\mathbf{x}_p - \mathbf{x})/\Delta t \tag{17}$$

$$\mathbf{x} = \mathbf{x}_p \tag{18}$$

$$\boldsymbol{\omega} = axis(\mathbf{q}_p \mathbf{q}^{-1}) \cdot angle(\mathbf{q}_p \mathbf{q}^{-1})/\Delta t \tag{19}$$

$$\mathbf{q} = \mathbf{q}_p \tag{20}$$

where *axis* and *angle* return the normalized direction and angle of a quaternion respectively.