Review of Tree Modeling with Real Tree-Parts Examples

Tree models are one of the special interests among computer graphics topics because of their variety in nature and they are one of the most common objects that can be implemented in almost any application. Due to their complex features, it is still challenging to create realistic trees even with the help of 3D tree modeling advancement. It is common that the programmers need to have a good knowledge of the features of the trees and their interactions with other forces or objects. Currently there are three major approaches to tree modeling, which are procedural, sketch-based, and data-driven. Procedural approaches rely on grammar-based parametric methods that allow the users to create trees with large branching details. However, it requires the user to obtain expert knowledge and very manually intensive. Sketch-based approaches are best for fast interactive design of trees, but sketching every detail, such as branches, leaves, etc…, can be frustrating and time consuming. With the help of 3D scanning, sketch-based approach has become more popular as the users can scan a real tree and reconstruct their 3D models. Data-driven approaches can produce the most realistic result as the generated models capture branching structures and geometric nuances. This article introduces a new method called example-based tree modeling. This method allows the users to design trees in an intuitive manner and generate realistic trees with complex branching structures and geometry details.

At first, a data-base of realistic 3D trees are captured. Then the trees are reconstructed and the tree-cuts, which are extracted from the tree branching structure and stored in a repository, are computed. These tree-cuts are selected and positioned in the scene and they connect to existing structures based on shortest-distance. The users can also easily select and position those tree-cuts in 3D space by computing an interpolating surface approximation of the final tree branch structure. In order to enhance the realism of the trees, the user can define branch details in a form of geometric textures onto the tree model branches. The user can also control the growth of the tree by scribbling lobes that are filled with foliage. This modeling can result in high-level of expressiveness and realism of the real trees geometrically. In addition, this tree modeling is also easy to implement as well.

In order to model a tree, tree-cuts are selected from the repository. If a specific part of the tree is missing, the system provides a sketch-based tree-cut retrieval tool. In this step,

there are many works on 3D model retrieval. The user is required to sketch the 2D shape of the object as seen from one or more viewpoints. Then these 2D sketches are stored into the repository. The distance field originating from sketches is computed. In this experiment, five different views are used as sample of the view space around the tree-cut. For each skeleton, a set of five corresponding 2D distance fields is computed and the system retrieves the top ten best matching tree-cuts with respect to the distance between their distance fields. Once a tree-cut is positioned, it is matched to other tree-cuts by computing the shortest Euclidean distance. At this point, the users can also manually specify the distances as well. The best transformation is selected by minimizing the sum of squared distances between the transformed points and their corresponding points on the other loop. Then the best matching pair is selected, and the full correspondence is computed by walking on both boundary loops in counter-clockwise direction and matching sample points. Connecting tree-cuts leads to branching bifurcations, which are modeled as superpositions of multiple connections between cuts. In order to blend and connect intersecting branches, Poisson reconstruction is applied in the intersection area. Allometric rules are best to determine branching diameters and angles. Therefore, this method uses allometric rules and manual cuts by user. The tree-cut connection by interpolating surface allows the user to twist and gnarled branches through modification of the tree-cut position and orientation. This is one of reasons why surface interpolation plays a crucial part. It can be improved by applying geometric textures from the repository, so it will play well with detailed geometry. Fitting cylinders along the tree-cut helps with computing cylindrical parametrization. Once the cylindrical parametrization of the fine geometry is obtained and scaling it to fit the target cylinder, then the geometric texture can be transferred to the tree. As a finishing touch of modeling the tree, the user can control the foliage production such as generating number of twigs and leaves. Scribbles of 2D loops can help define major lobe contours and guide foliage growth. Then the lobes are filled by randomly distributed sample points which is used to generate twigs.

After multiple experiments, this example-based method is proven to produce more realistic trees from a collection of example tree models and tree-cuts that are extracted from both existing repository and user's sketches. This method is able to generate tree models

that has geometry nuances of example models, which existing system does not have. The proposed method is only capable of creating subset of existing trees, because of limitations in scanning large tree structures. In addition, it also ignores the interrelations between branches, but it could be amended by tropisms and other global factors. The example-based method definitely has potential future to be expanded and improved. The results of this method can be improved by obtaining more data from scanning more trees from the real world. It also allows users to be creative and create their own type of trees using the existing data. By leveraging state-of-the-art procedural methods and using the underlying procedural principles to suggest the placement of new tree-cuts, this method can become more intuitive. The user interaction part can also be taken one step further by automating the suggestive positions and editing of the trees This would definitely reduce the workload of editing and defining locations of each tree-cut so the users can focus more on creativity.

Review of Real-Time Interactive Tree Animation

This article is about introducing a new method to capture the complex and mesmerizing natural motions of trees. The proposed method is inspired by the position-based dynamics and the O(N)-style algorithms for articulated rigid body simulation. The new method can be achieved by using the analytic solutions to the spring dynamics equations in order to model rotations about joints. By leveraging the simplicity of a position-based dynamics approach, this method is believed to be faster than the Featherstone-style approaches and Gauss-Seidel-style approach because it does not require large number of small time steps, expensive implicit time integration, or repeated iterations. The proposed method can also utilize parameters and simulation of tree models with many degrees of freedom.

For the geometry of a tree, its trunk, branches, and twigs can be modeled as a series of conical frustums which results in linear approximation of the curved branch. Its leaf is a thin triangle prism, while its fruits are spheres, so it is more efficient to utilize collision detection techniques. The body of the tree is represented by a vector in maximal coordinates. This information can be applied to its parent, its children, its root, and all forces being applied

to it. Each joint consists of three one-dimensional rotational springs that resist the angular displacements about the corresponding axis of the joint frame.

The proposed method is based on O(N)-style algorithm, but with several aggressive assumptions that reduces the need for small time steps or costly iterative solvers without the restriction of emulating tree dynamics. It also robustly and efficiently takes only a single time step per rendered frame, which results in simulations in real time. The composite body updates are achieved by rigidifying the entire outboard subtree emanating from that body inclusive of the body itself. For each composite rigid body, the mass, the world space inertia tensor, and the total external force are computed while applied torque to the composite body is evaluated about its parent joint. The calculation is performed from the leaf-level rigid bodies and traveling backward towards the root of the tree.

The forces that are applied to the tree are represented by Newton-Euler equations of motion about the center of mass of rigid body. These forces are call fictitious force. There are also other “free” rigid bodies that exist, such as collision. This occurs when some forces are affect the fruits and make them fall off their parent branches. This can be simulated by computing the impulse that makes the velocity of the free rigid body match the velocity of the tree and apply the impulse when it pushes the fruits away from the tree. After running the collision detection, it turns out that this approach might need a smaller time step when the penalty forces are large. Therefore, the collision detection is sped up by building an acceleration structure based on the reachable workspace. The effect of wind is also considered in this approach. The tree is immersed in a spatially varying wind velocity by orthographically projecting a two-dimensional uniform square grid that encompasses the tree and its surroundings. The wind velocity is defined outside of the projection where each grid has spatially constant wind velocity where it contains its own number of octaves, persistence of amplitudes in higher octaves, and starting phase angle. The wind that affects the branches is defined by using a pressure-based force. For the leaves, they can fall from the tree by detaching them from their parent branches when the torque on the parent joint exceeds a threshold. The approach for simulating the leaves is inspired by a data-driven approach and randomly sample prescribed spline curves.

Overall, the proposed method reduces the drawback of other methods that do not consider the two-way relationship between the wind and the tree. This algorithm can produce more realistic motions. Sometimes the trees can react unexpectedly, which is the reason why it is important to consider how different types of trees shield each other from the wind. This method has potential room to be expanded and improved. The force-based interaction that is used in this method could be improved to consider the effects of rain hitting leaves or snow accumulating on branches. On the collision part of this method can also be expanded to consider an impulse-based treatment of the rigid bodies that builds up the tree. Another main limitation of this method is the visualization. The result does not seem to be as realistic as a real tree. As an improvement, this algorithm can be implemented with parameters for the real tree. With the combination of the realistic models and the proposed simulation algorithm, there is no doubt that the future work can produce more visually rich trees with more realistic motions.

Comparison between the Two Literatures

The main focus of these two literatures is about creating trees. The first article is about a method that can create a more realistic tree called example-based method. This method relies on existing data of tree models. These data are collected from the scanning of actual real trees. The method works like a puzzle as it puts together the pieces of trees together. Allometry rules play a huge part in this method as it is responsible for ensuring relations between adjacent tree branches. The second literature is about a method that can make the motions of the created trees more realistic. This method is a combination of position-based method and O(N)-style algorithm which results in less small time steps, less time integration, and less iterations. Overall, with the combination of the example-based method to create visually rich trees in the first literature and the realistic simulation method in the second literature, a tree can be produced with realism of a tree with a real-life like motions in any application.

Citations

- E. Quigley, Y. Yu, J. Huang, W. Lin, R. Fedkiw, “Real-Time Interactive Tree Animation”, IEEE Trans. Vis. Comput. Graph., vol. 24, no. 5, pp. 1717–1727, May. 2018.
- K. Xie, F. Yan, A. Sharf, O. Deussen, B. Chen, and H. Huang, “Tree modeling with real tree-parts examples,” IEEE Trans. Vis. Comput. Graph., vol. 22, no. 12, pp. 2608–2618, Dec. 2015.