Smoky Forest: Iyal Suresh and Varun Nambiar

Motivation

Setting out on this project we observed that volumetrics are a powerful tool in composing attractive imagery. They can be used to set the tone of a scene, and are capable of heightening the feeling of depth in a fundamentally two dimensional image. Having chosen a broad category of effect to look at, we then looked for visually impressive photographs to inspire us. After many hours we came across this striking image of a redwood forest and decided that a similar shot featuring crepuscular rays would be a great way to show off the beauty of volumetrics

Approach

From examination of our “target” image, as well as what we’d learned in class, we chose a few elements to focus on for our project. First we observed that a scene making heavy use of volumetrics would likely need to an efficiency boost to be renderable in a reasonable amount of time. We also observed that the geometric complexity of the tree branches were critical to creating visually interesting shafts in the light beams. As we needed to create enough trees to give the illusion of a forest, it was decided to create them procedurally rather than by hand. This gave us a convenient work distribution, with Iyal focusing on creating and composing the scene geometry in Blender, and Varun focusing on improving the efficiency of volumetric rendering in pbrt.

Scene Creation -Iyal-

Tree Creation

One of the references we found while planning our project was **this paper** from Hewitt describing various methods of procedural tree generation. As a full dissertation, this paper had way too many details, and could account for many diverse types of tree. The primary takeaway was the rough workflow for parametric tree generation based on the work of Weber and Penn. This template can be described as follows:

1. Create the trunk using a beveled curve object
2. Taper it
3. Add branches recursively, also as beveled curves
4. Add Leaves as meshes

As it happens, the bulk of what Weber, Penn, and Hewitt contributed is good parameters for describing trees in general. However, as we mostly only needed redwood trees, we adopted the general recursive approach outlined above and simply added ad hoc parameters as we went. The most “reality based” parameters in our model is the function describing the tapering of the tree trunk with height. This was taken from an article on tree taper[1]. The first parameter added was a “gravitropism” term. Stems and branches tend to grow away from the direction of gravity, and roots tend to grown downwards. After implementing this and looking at our reference image we realised that many of the branches on a redwood are thin enough that they bend downwards due to their own weight. This same term was thus inverted to approximate this droop. The next parameter implemented was a term to point sub-branches somewhat in the same direction as their parent branch. This parameter was tweaked manually to give results that looked reasonable. The last two “behaviors” added into the script were regarding placement of branches and subbranches on their parent. Going from trunk to tip along a branch, sub branches tend to alternate sprouting between two sides. This was explicitly added as a rigid behavior for all subbranches. The second behavior was for branch placement around the trunk. Initially this was done entirely at random, resulting in trees such as those in the below.

This looked weird and unnatural, as it was too unstructured. Many objects in nature exhibit spiral behavior with pitches as a ratio of fibonacci numbers. Some examples are pineapples and pinecones. As we’d seen reference images of redwood trees with some spiralling to their branches we decided to have all of the branches follow a spiral pattern.

Practical Notes

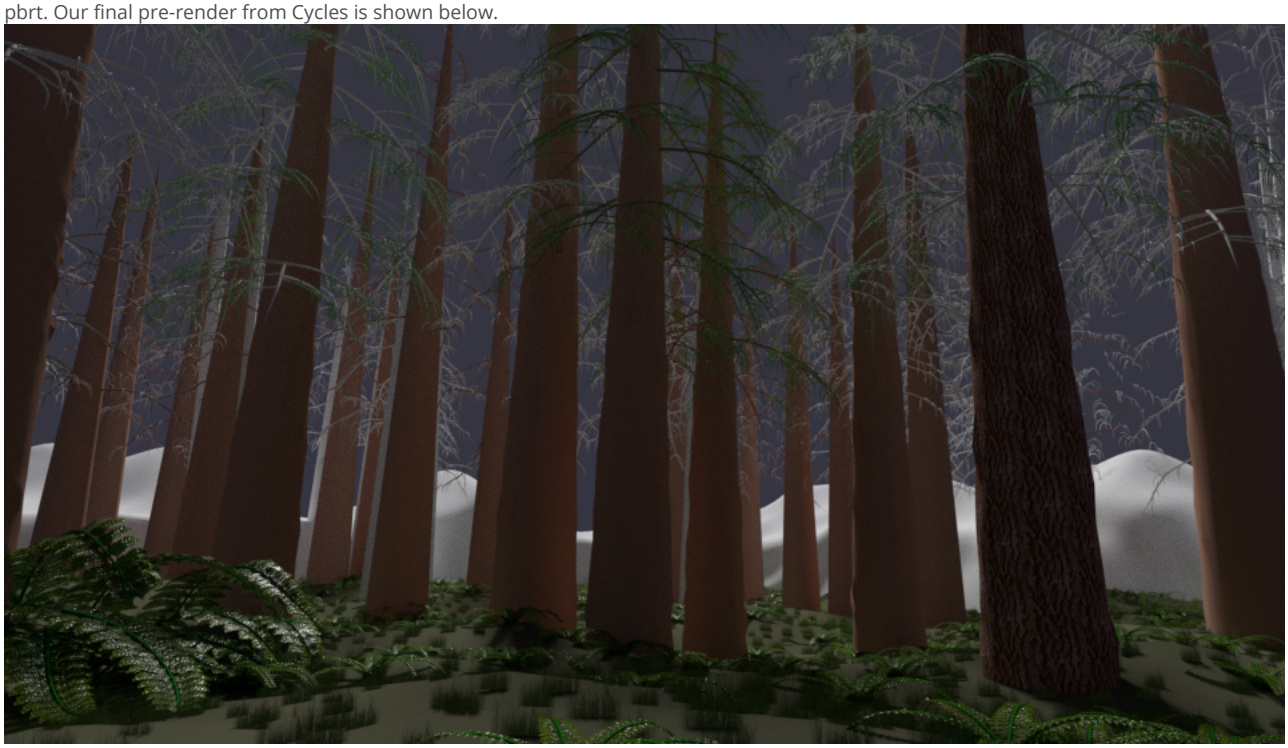
The parameters and methods described above capture the bulk of how our redwoods were created, but there were a few other layers of nuance added to enhance the effect. The first is that noise terms were added to almost every stage of the trunk and branch creation processes. This noise was small relative to the overall patterns enforced by our parameters, but served to break up the regularity of the trees a little bit. These parameters were also varied for different levels of recursion using special cases. Secondly, the curves resulting from the last level of recursion (the finest and most numerous branches) were separated out as a separate object. This was done so that they could be separately manipulated by hand if necessary. Lastly, the trunks of trees in the foreground were displacement mapped to break up their silhouette. Other than these post processed trunks, all trunks and branches are perfectly round.

Non-Procedural Geometry

After the trees were created, the rest of the scene was filled in around them. Ferns loosely inspired by those in our reference image were manually constructed using a variant of the procedure used for the trees. One fern model was instanced around the ground as minimal ground cover. Scaled up fronds from this fern were then copied and used to create a larger fern in a conspicuously empty patch of ground. The leafs of the fern are taken from an image of a fern and alpha-masked onto simple geometry to fake higher quality meshes[2]. A similar technique is used with billboarded squares on the ground for grass. The ground and mountains were created using simple noise applied to approximate representations of their shape. Lastly, a large diffuse plane was added behind the camera to emulate the indirect illumination that should be coming from the nonexistent forest behind the camera.

Scene Composition

Except for the volumetrics, all the geometry and surfaces in the scene were set up in Blender. Due to the high number of trees, they were randomly placed by our script. “Weird areas” created by the pure random nature of this were corrected by hand.Camera angles and field of view were also tested using this. There was some differences in surface texture that came from translating materials in Blender’s Cycles renderer to pbrt, but the quick iterations it enabled in scene composition were vital. Lucas Silva’s **exporter** from Blender was used to transfer the scene into pbrt. Our final pre-render from Cycles is shown below.



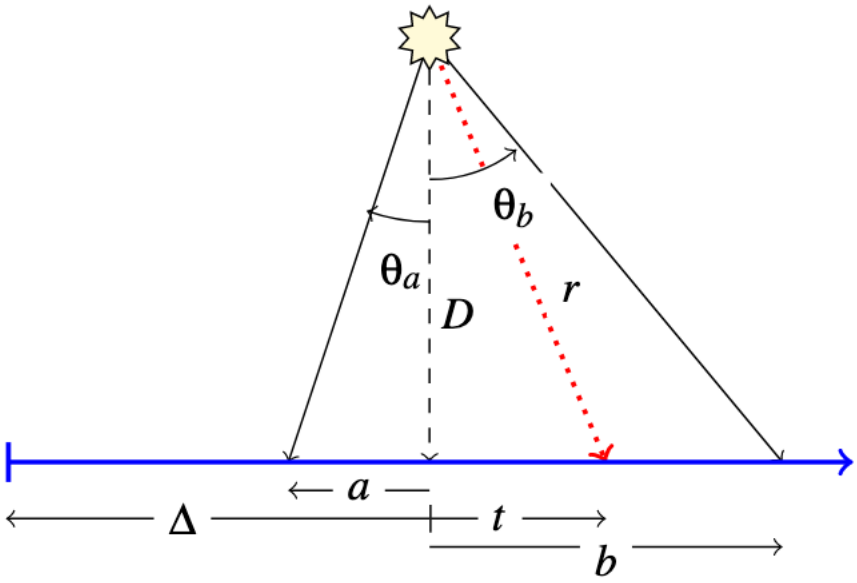
Roadblocks and Difficulties

The largest difficulty in this was learning how to utilise the scripting system within Blender to achieve our goals. Documentation of all of the functions exist, but it is non obvious *which part* of the system has the functions that are needed. This was complicated by the fact that the recommended way of learning Blender scripting is to do operations by hand and to use the provided readout of what python function that operation called. I learned that many of these functions rely on the mouse being in the relevant portion of the screen which led to many elaborate workarounds. Additionally, these functions are very inefficient. I discovered by accident the “correct” way of scripting in Blender on the Blender forums while looking for a solution for a different problem I was having. This happened roughly a day before the rendering competition. This change in how I was using Blender resulted in generating one tree taking milliseconds instead of a minute or two. This drastically increased my ability to iteratively tweak the look of the trees. As a result, most of what I accomplished was completed right at the end of the project. Many other planned features such as leaves and other types of trees were not done due to time constraints.

Lighting and Volumetrics -Varun-

In order to render crepuscular rays in our scene we needed to implement some form of volumetrics. PBRT implements a distance based sampling technique in order to render higher transmittance at boundaries of the volume. However, if the light source is within the medium distance sampling tends to fall short. We opted to use a technique known as equi-angular sampling which is highlighted in a paper by Kulla et al.[3]

Sampling Homogeneous Media



For a given point along a ray that interacts with homogeneous media, we compute the transmittance of that point by sampling another point by the offset $\delta + t$. The point is selected by setting a and b as bounds of integration and randomly picking a point in between. The equations below are used in the implementation:

$pdf(t) = \frac{D}{(\theta_b - \theta_a)(D^2 + t^2)}$

$t(\zeta) = Dtan((1-\zeta)\theta_a + \zeta\theta_b)$

$\theta_x = \taninv(\frac{x}{D})$

Implementation

The ideal way to implement this new sampling technique is by subclassing pbrt's Medium class. Unfortunately our method requires the scene state in order to sample the correct point since it takes into account the light position with respect to the ray. As a work around we implemented the sampling technique in a new integrator and made the absorption, scattering, and transmission coefficients from the Medium public members.

In order to calculate the correct transmission value we first sample a point using the exponential function as defined in the base Medium class. Given the point, we check if the point is interacting with media to see if we should sample another point within the media. If it is, we compute bounds along the ray that intersects the homogeneous medium and randomly pick a point to sample. In order to accurately weight the transmission value in the image we needed to divide by the pdf defined in the paper. However, since we sampled our original point using exponential sampling, we need to reweight our equiangular pdf by dividing it by the exponential sampling pdf. This way, we can guarantee our image is physically accurate.

Final Image

Git

The Team:

Iyal Suresh:

Iyal is a second year MS in Mechanical Engineering graduating this quarter. He enjoys filling his free time with cooking, Ultimate, and now computer graphics. As he leaves Stanford he hopes to design and build some really cool robots.

Varun Nambiar:

Varun Nambiar: Varun is a first year MS in Electrical Engineering who is excited about robotics and graphics. When he's not working on psets or in the lab, you will see him road-tripping or playing his favorite retro videogames.

---

1. Ormerod, D.W., 1973. A simple bole model. Forestry Chronicle. 49:136-138. ↩

2. Source image of a fern courtesy kisspng ↩

3. Kulla, Christopher, and Marcos Fajardo. "Importance sampling techniques for path tracing in participating media." Computer graphics forum. Vol. 31. No. 4. Oxford, UK: Blackwell Publishing Ltd, 2012. APA ↩
