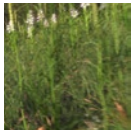
Old Gnarly Fantasy Tree Project

Genera-
ting basic
branching
structures.

Create
materials and
make use of
displacement.



Different
approaches
of modeling
roots.

WALLIS ECK

The Old Gnarly Fantasy Tree Project

First of all I want to mention, that english is not my mother tongue. So it might very well be, that you spot one or another spelling error, faulty grammar or simply bad use of english language - sorry for that!

Second I want to mention, that the information you will find on the next pages is not of scientific nature. It's just what I see and feel in my personal, every day life.

So why should you read on?

Actually I have no idea, probably some of my thoughts might be of help, when you are struggling in creating 3d models of plants - let it be grass, flowers or trees.

In some cases I recorded video tutorials, so the written part might be very short. I already mentioned that english is not my mother tongue - I hope you will be able to understand my mumbling, which I consider to be english - I heard from other opinions though. Also the sound quality is not of best quality and sometimes you will hear disturbing noise from around - I guess this is the deal you have to take, in return you´ll get this all for free.

So have fun and good luck!

P.S. Some of my models are available online - if you don't have the time to do your own plants, you might consider to visit the website of Greenworks.
Another option would be to visit the online store of Cornucopia (mostly Vue related) or Turbosquid.

Jan Walter Schliep

Table of Contents

1 Introduction

During the OGFTP (I think wrinting Old Gnarly Fantasy Tree Project all the time, is a waste of time and bytes) we will build an old and gnarly tree, often found in fantasy movies or illustrations - and of course also in nature!
This will be a longer project and I decided not to do it „Step by Step". The reason: it simply would take way to long and everybody would have the same fantasy tree as result.

During this project I will try to show you several techniques and different approaches to create the fantasy dream tree of your imagination. I will use different tools, so that as many users as possible can follow this project.
I also will combine several tools to get the best of each software.

As first step I will use Xfrog4.2 (and later Xfrog3.5) to create a base mesh for our fantasy tree. So have fun and continue with the first Step, creation of a branching structure..

Links to references

It´s always a good idea to know what you want to build. Some sketches or photographs as reference are always helpful, even if you don´t want to create a realistic plant.
I decided to model a tree like you can see on Caspar David Friedrichs painting „Einsamer Baum" (Solitary Tree). You can find the painting on Wikipedia for example: Der einsame Baum.

If you want to go for a different tree, simply use Google or another search engine of your choice - normaly you will find a lot of reference pictures.

Some other useful sites may be:

- Forest Conservation Portal
- UW-Madison Department of Botany
- Plants of Hawaii
- Digital Library Project

So now after we have some reference, we can start Cinema and Xfrog4 ;-)

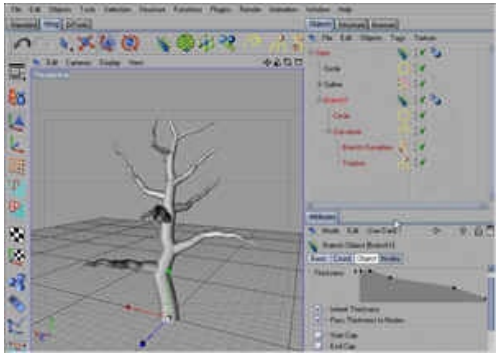
By the way - if you would like to work with the files I created during the following tutorials, you can download the scenefiles right here:
OGFTP Scenefiles

2 Modelling the tree

Step 1: Basic Branching Structure

At first we will start to build up the basic branching structure of our tree. If you want to download the scene files of this project, then you can get them here:

OGFTP scene files



We will use the branch component to build the stem and the main branches of the tree. You should keep the resolution low - this improves editor redraw and render speed. Apart from this, it´s easier to increase the resolution of needed parts then the other way around.
Use the typical tools to work on the path splines (Point Mode, Move, Scale and Rotate tools). If needed you can add more points by CTRL+Leftclicking onto the spline (you should be in Point Mode with Move tool active).

Some additional hints:

- If you want to get even more variations into your branches, then make use of the Variation object. Model several branches and set the Variation mode to random. Normally I use this method as last step - so I can copy a complete branchingstructure and do some changes, instead of completly building a new branching structure
- If you want to control one specific branch - perhaps you need one branch to look dead without leaves, then make use of the Variation object. Set the mode to Exception and put the regular branch and the exceptional branch inside
- Often you get nice results by using a Variation object (mode set to random), and putting the regular branch and a null object inside. This you can use to simulate „missing branches“

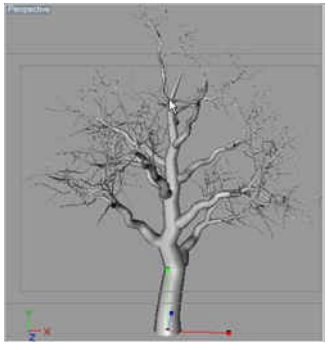
Download the Video Tutorial OGFTP - Stem and main branches.

Continue with the next step, Higher Branching Levels

Step 2: Higher Branching Levels

Trunk and main branches are done - but now we need some smaller branches.
Actually this works pretty much the same like with trunk and mainbranches. Simply use another Branch component, drag it into the third position of the main brach hierarchy and adjust parameters like Node Growth, Angle and of course the number of branches.
You also could simply copy the main branch, rename it and then adjust some parameters. It also might be a good idea to lower the resolution of the splines of the higher branching levels, this can save you 10thousands of polygons.

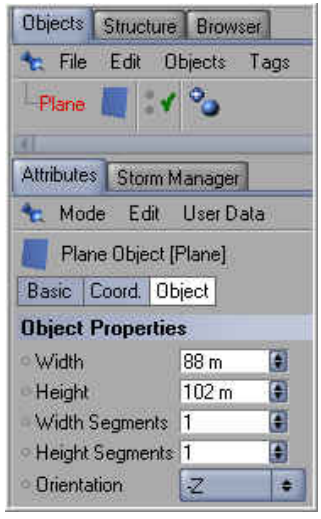
Download the video tutorial on how to build the higher branching levels: Higher Branching Levels



In case functions and equations in combination with Xfrog are new to you, you might want to take a look at the chapter about noise, u and i functions.

Step 3: Leaves on our tree

Now we will add some foliage to the gnarly tree. It´s up to you - or your tree - if you use a bitmap with a single leaf, or a texture with several leaves - or even a bitmap that replaces a complete branching level. If you want to save some polygons, then you should use the second or third option. If you really want a detailed result for closeups, then I would choose the single leaf bitmap option.



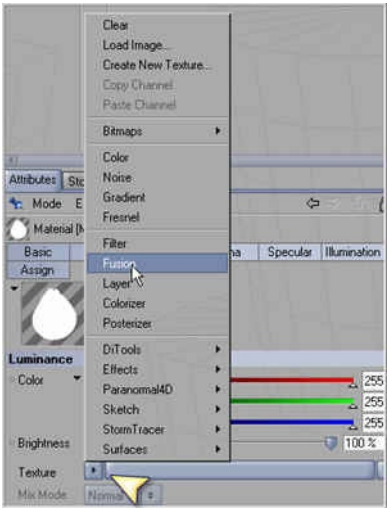
Add a Plane object to the scene. Width and Height should have the same ratio that you used for the bitmap. Otherwise the leaf will look stretched or sqashed. Don´t forget to set the number of segments to 1 - otherwise your computer will explode later on ;-)
Then set the Orientation to -Z. We are almost done with the geometry of the leaf. Use the Make Editable command (rightclick the Plane->Make Editable, or just hit „C“ on the keyboard) to convert the parametric Plane into a polygon object. I am doing this, because I want to move the pivot point (Object axis) and this doesn´t work with parametric objects.

Now after we have converted the Plane to a polygon object, we can use the Object Axis tool.

Activate the Object Axis tool, then go into Move mode and move the axis down to the base of the leaf. You could make use of Snapping to be sure, that you really „hit“ the bottom of the leaf. But the most easiest way is this: take a look at the coordinates manager and enter „-height/2“ for position Y. In our case the height of the leaf is 102 units. So I enter „-102/2“ or just „-51“ for position Y. Don´t forget to press enter (or leftclick the Apply button).

The axis should be moved down to the bottom of the leaf. Now switch back to the Model tool (Model tool) and move the leaf object up (position Y=0).
Now we need a material that we can apply onto the geometry. Go for the Material Manager and choose File->New Material.
Leftclick the material icon to take a look at the properties in the Attribute Editor. It´s also possible to doubleclick the material icon - a new window will pop up (Material Editor). It´s up to you what you prefer most!

Go for the Basic properties of the new material and



enter a new name - perhaps something meaningful like Leaf. The Basic properties also contains a list of the different material properties. By default only Color and Specular are activated, but we also need Bump and Alpha.
Then go for the Color channel. Specify the path to the leaf bitmap in the Texture field. If you downloaded the scenefiles, then this would be the file leaf.tif . If you have the feeling that the leaf texture is to blurry, you should reduce the Blur Scale parameter. In most cases I am using a value like -50%. You also could use one of the other Sampling methods like Circle. Normally this saves some memory during rendering, but despite this fact rendertimes can get higher. The reason: if the bitmap is „presampled“, then the render engine has to invest less power into antialiasing. So if memory is not critical, then MIP or SAT sampling should be your choice.

Perhaps you also want to use the bitmap inside the Bump channel. An alternative solution would be to use a noise shader.
The bitmap leaf.tif contains a premultiplied alpha channel, so we can use the same file for color channel and for alpha channel.
So use leaf.tif inside the Texture slot, then deactivate Soft (the edges of this leaf are sharp, so I want a sharp alpha).
Activate Premultiplied and don´t forget to use Invert - you´ll notice why you have to do this.

Again you can make use of lower Blur Scale values to avoid a blurry look of the leaves. In my eyes, it´s a good idea to use the same values for Color and Alpha channel. Especially when you are working with bitmaps that don´t offer a premultiplied alpha channel.

Okay, we created a leaf material. Perhaps you want to work on the specular too. There are no rules for this channel, it always depends on the plant and the lighting.

Some leafs are dull, so you would have to use a low

and wide specular, other´s are more shiny (higher and narrow specular).

Now apply the material to the leaf geometry. This you can do by dragging the material icon onto the object (either in the viewport or in the objects list). That´s it, we are finished with the leaf.

The video tutorial shows you, how to add the leaf object to the gnarly tree:

Download Add leaves to our tree video tutorial

3 Bark material and Displacement

Part1, basic setup

We already have a nice tree - but it still lacks a proper surface. There are different possibilities to create convincing surfaces and one of the most simple approaches is to make use of photographed or painted bitmaps.
Procedural textures also are powerful and they have some advantages over bitmap textures - resolution independet to name just one of them.
But it´s not easy to create convincing procedural bark materials, and therefor I´ll spend an extra chapter on this topic. For now we will go with bitmap textures.

The first clip shows how to create a simple material for the higher branching levels. We´ll use a bitmap for the color channel and a second one for the bump channel. In Cinema you can overide most maximum settings of parameters, so instead of using only a bump strength of 100%, you can go with values much higher! You just have to enter those numbers manually.

After creating the Branch material, we´ll concentrate on the bark of trunk and main branches. Cinema4D R9 offers a nice feature called Sub-Polygon Displacement. An object (which can be pretty rough) is smoothly subdivided and displaced at rendertime.

Download the videotutorial Bark material and Displacement Part1.

Part 2, finetuning

When taking a closer look at the displaced surface, then you´ll notice a problem: the displacement uses the same strength for all underlying parts, no matter if the strong base of the tree is effected or thin parts of the branches. Often the result is a badly deformed mesh.
But there are several possibilities to overcome this problem. One option would be to use materials with different displacement settings. Another would be to use gradients together with the displacement textures.
The videotutorial Bark material and Displacement Part2 covers this topic!

Additional hints:

- you could make use of displacement to create roots for the tree
- use your painting software to paint displacments into a black/white version of the color texture
- you can get nice displacement results with 3D noise shaders
- Cinema4D R8 users will have to increase the resolution of the models manually, C4D R9 users simply should use Sub-Polygon Displacement
- if not necessary, don´t use Sub-Polygon Displacement on higher branching levels - this increases rendertime!

We already built a nice base for our old and gnarly tree. At a later stage I´ll also use Xfrog3.5 to build a base object.

The upcoming (mini-) tutorials will show you how to finetune this base tree. I´ll try to present different techniques to create roots for example.
Also we will learn how to reshape the surface and how to create smooth transitions between stem and main branches.

So please keep you eyes open for upcoming tutorials.

4 We need some roots!

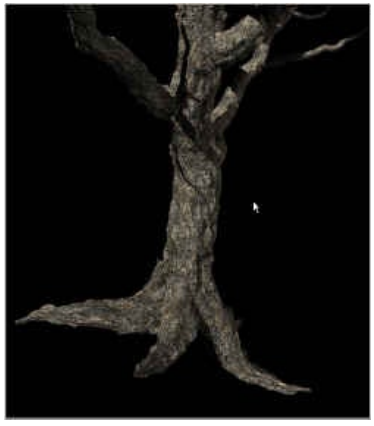
Roots with Xfrog 4

This tutorial will show you two possible approaches, how to use Xfrog4 to create roots for a tree.

Roots - the first approach...

...is very simple - just build an additional, inverse tree!
Xfrog4 offers all parameters to create branching structures. Roots are branching structures, they only grow into the opposite direction!
As first step add a base Branch component to the scene, it will serve as base for the roots. Of course the base should be much shorter than the regular tree. Just be sure, that the base follows the „path" of the tree.
Then add another Branch component to the scene and use it as child object for the „base roots". You can either use the Cinema tools to move the spline points to create a more irregular look, you also could make use of the Curvature spline. This is exactly the same what we did before for the regular branches of the tree.
Now use the Node Angle parameter of the base to change the angle in wich the roots emerge from the base.

The tutorial video Roots with Xfrog4 Part1 will show you some more details.



Even more roots - with Xfrog 4 and standard tools

The last tutorial showed you how to use the branching capabilities of Xfrog4 to create roots. But sometimes we need a very special look. Think of roots following a rough surface for example.
The following tutorial will show you some simple but effective techniques to create „custom" roots.

We will use some of Cinema4D´s spline tools - most other applications should offer similar tools. It´s up to you, which spline tools you use. I personaly prefer to „draw" the path spline of the roots with a freehand spline. In most cases I set the Spline Type to Cubic instead of Bezier, but again this is a matter of taste.
Don´t care about the surface right now, just draw

the splines in the top view. Then use the Project Spline command (Structure->Edit Spline->Project) to project the splines onto the ground. Be sure to turn the visibility of other objects or obstacles off, otherwise the splines will be projected onto those objects too.

Of course you always can finetune the „flow“ of the splines with the regular tools. When you are satisfied, use the pathsplines inside Xfrog Branch components - or inside a Sweep Nurbs object.

And here you can download the video tutorial of this course. It shows the basic steps to draw the path splines, the projection and how to use them inside a Xfrog4 Branch object:

Roots following the ground video tutorial

As you can see, it´s pretty simple to create roots that follow the ground! I am sure that most applications support similar tools like shown above, so when you are on Xfrog3.5 and a different 3D application, you should be able to get similar results.

A little note:
When creating a lot of single roots with different splines, then you have to change every root object on it´s own. This is very flexible, but sometimes you might prefer one single root object.

It´s possible to merge the different splines - simply select all of them and choose Function->Connect. A new „combined“ spline object should appear (you can delete the originals if you want to). You can use this merged spline inside a Sweep Nurbs or inside a Xfrog4 Branch object.
Now you have one single Rootobject. Sometimes you´ll get „artifacts“ at start- or endpoints of the spline segments, but normaly you can hide those problems under the ground or inside the stem. Also trying a different interpolation mode might help.

5 Some math for fun

Excursus: about noise, u and i

One of the new features of Xfrog4 is the noise function. It seems that this function is a little bit mystical to a lot of people, so here some hints.

About noise(0) - it´s not a bad argument. I just often use(i+100) or something similar instead of just i, because in most cases the result looks better.
I think the reason is, that the higher the input into noise, the higher also the frequency of the noise. u=0 is not critical, because it´s only 0 at the absolut bottom of the Curvature spline. 1 inch away it´s bigger than 0 and so a noise is generated.

But if you take i, then for the complete first branch the noise will be 0 , at least if you use u*i. When using u+i, then there would be a noise - I just prefer to multiply those parameters, because then you get more variations. Therefor it´s a good idea to use i+something, so that also the very first branch also get´s some noise.

A little summary:

- u is some sort of gradient and produces 0 as output at the base of the Curvature spline and 1 at the top.
  You could use u to create „waves“ ,for example if you use u together with cos or sin: cos(u*6.28).
  Or to control the strength of a function along the Curvature spline: cos(x)*u.
  Or inside the noise function to generate different noise values along a control curve: noise(u).
- i stands for iteration and returns a unique number for every branch. The first branch gets the iteration number 0, the second becomes 1, the third returns two and so on.
  If you wanted to build just one branch, but along the tree this branch should look different, then make use of i inside the noise function: noise(i)
  Every branch would get a different curvature.
  But if the curvature itself should change „inside“ one branch, then use something like noise(u+i) or noise(u*i).
  As described above, it´s a good idea to replace i by i+something, for example noise(u*(i+10)) or just use noise(u+i+20).
- use the control curve to control the strength of the noise function. The formula could look like this for example: noise(u+i*5+10)*x.
  And if you want to go with higher frequencies, then you could additionaly multiply the stuff inside the noise function: noise((u+i*5+10)*50)*x, or noise(u*100+i*100+100)*x. Please do your own tests and check the results, more worth then thousands of words ;-)

When working with Xfrog3.5 , then you will have to use random (rnd) instead of the noise function. It simply returns a random number - so please take care when using this function!
For example a plant that makes intensive use of the random function could look different on different machines!
Also when adding a new random parameter or removing a random function of the plant, this already

might change the appearance of this object - even if the values inside random are 0 and the output of this function is 0! The different rnd functions seem to be conncected somehow...I guess this is one of the Xfrog 3.5 mysteries ;-)

The End...for now

I hope you had some fun going through the hints and tricks of this little tutorial series.
At a later point I also want to add some more approaches, especially for Xfrog 3.5 users.

Jan Walter Schliep, Tucherstr. 17d. ¤ 90562 Heroldsberg ¤ tel +49 911 8919867 ¤ www.wallis-eck.de