The Expert Panel

This issue's team of pro artists...



IAN FAILES
vfxblog.com



Ian speaks to the geniuses of animation at Walt Disney Animation Studios to find out how they are re-rendering Wreck-It Ralph for brilliant new sequel, *Ralph Breaks The Internet* on page 24.
3DArtist username N/A



ABSKETCHES
absketches.com



AbSketches is a visual creative studio who has kindly given us a masterclass on creating stunning architecture visualisation. Their seven-page tutorial is on page 56.
3DArtist username abSketches



RAINER DUDA
welovecgi.de



Taking inspiration from *Blade Runner 2049*, studio owner and university lecturer Rainer has shown us how to make an awesome hologram in real time with UE4 on page 64.
3DArtist username rainerd



STEFAN KANG CHUN YIH
stefankang.com



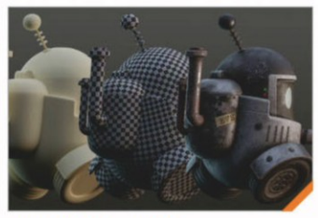
Based in LA and working at The Mill, Stefan teaches us how to create effective procedural lightning storms in Houdini this issue. You can read his guide on page 70.
3DArtist username stefankang



MILAN BOJANIC
artstation.com/25thportal



Milan completed his mentorship at Think Tank Training Centre and is trained in 3D modelling and texturing for the film industry. He teaches us how to create an expressive face on page 74.
3DArtist username 25thportal



ALEJANDRO TREVIÑO
aendom.com



Alejandro is a self-taught artist, always looking to improve his skills. He's helping us with improving our skills with UVs in Blender this issue. His six-step guide is on page 76.
3DArtist username aendom



JAMES CLARKE
N/A



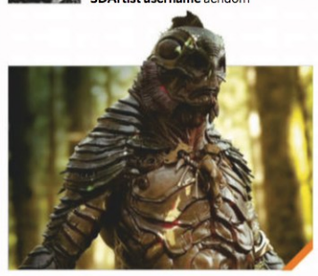
Book and film writer who has traversed the grounds of *Jurassic World* and headed into the new London digs of ILM London. Hear from behind the scenes at the studio on page 40.
3DArtist username N/A



FARID GHANBARI
renderburger.com



Farid is art director at medical animation studio 3FX. We thought of no one better to give an expert verdict on the latest Maxon release: Cinema 4D R20. Read his views on page 78.
3DArtist username Farid



MAARTEN VERHOEVEN
artstation.com/mutte



Freelance ZBrush artist and digital concept sculptor Maarten uses KeyShot frequently for his work. This issue he's taken a look at KeyShot 8 Pro on page 80.
3DArtist username mutte

6100+ **free** effects

Special offer for 3D Artist readers
75% discount
coupon: 3dartist

New Filter Forge 8.0

Powerful software for Mac and PC that lets you generate seamless textures and apply effects to your 3D scenes.

- 12,000 free filters – best offer on the market
- Create your own textures and effects in the node-based editor
- All filters are fully adjustable for your own needs
- Seamlessly tiled and resolution independent
- Real-world HDR lighting
- Free fully-functional 30-day trial version

www.filterforge.com

Bump Map
Diffuse Map
Normal Map

6400+ **free** textures

SUBSCRIBE TODAY! Turn to page 22 to learn more





STEFAN KANG CHUN YIH
stefankang.com

Bio
Stefan Kang is a VFX/CG generalist based in Los Angeles. He has worked across a wide range of media, TV, commercial, event projection mapping, motion/title design and video. Stefan is currently focusing as a Houdini generalist and using his skills to create VFX content for the entertainment industry.

YOUR FREE DOWNLOADS from **filesilo.co.uk/3dartist**
- Houdini file
- Tutorial screenshots

HOUDINI

Create procedural lightning in Houdini

In this tutorial, I'll take you through the entire process of creating procedural lightning in Houdini for use in your own projects. We'll start by creating source points for your lightning sim. Then we'll create some attributes to help you control the lightning. Next we'll jump into VOPs to set up our lightning trigger frame and duration – this is an essential step in the process. We'll animate noise on our lightning to give it movement and add additional lightning branches. Finally, I will show you how to create a mask for the lightning and also create custom AOVs so you can enhance the look in compositing.

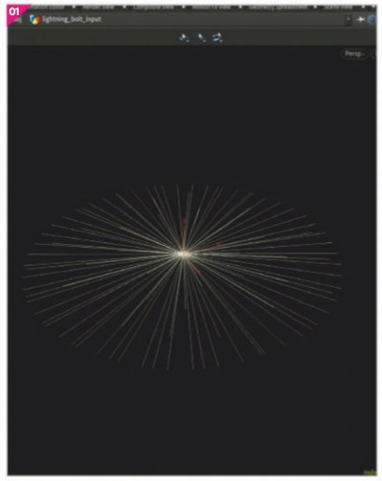
There are a handful of lightning tutorials online, but this setup gives you more freedom to art direct the effect and use the asset in different scenes. As a bonus, I'll talk about node dependencies and also the importance of building HDAs after you've completed an asset like this. By using HDAs you can quickly render different iterations and use one asset for multiple shots in a sequence.

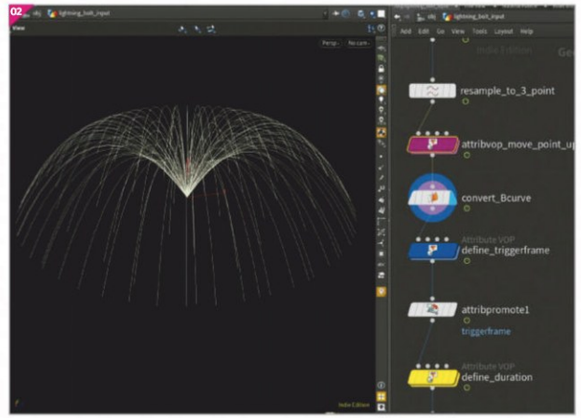
This lightning effect was inspired by Peter Claes' (FX supervisor at the Mill) lightning FX work in the movie *Thor*. I always look to big-budget FX movies for inspiration and Peter's work on *Thor* is worth studying if your goal is to create successful lightning effects.

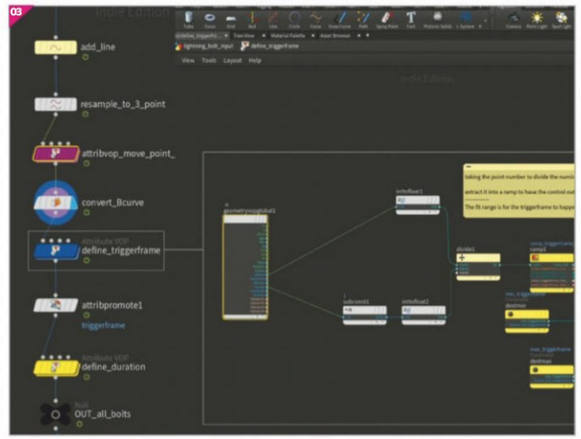
01 Lightning source setup We will start by creating two source points: the start point and the end point of our lightning bolts. The start point has all of the points in the centre of the grid. The end points are points that I scattered on an animated circle shape, which I animated to scale over time. With the end point and start point defined, we will connect them and create lines that will serve as the source for the lightning.

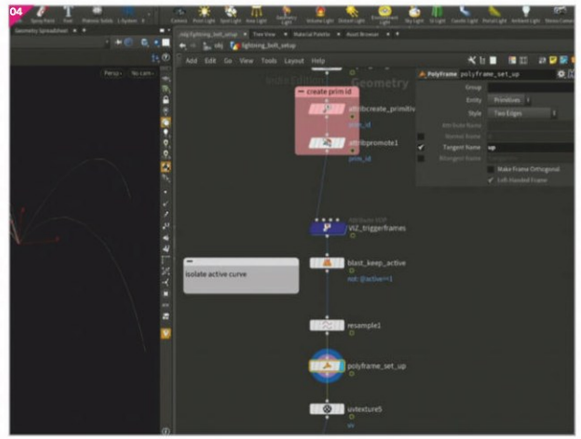
02 Convert to NURBS curve based on neighbour count Now, we will resample the lines so there are a total of three points on each line. We'll also set a condition so that if the points' neighbour count is greater or equal to two, it will move the point upwards. To move it upwards just add a vector parameter to the Y value. Next we'll convert the curves into NURBS curves which will bend nicely as you can see from this step's image.

03 Set up trigger frame and duration Next we'll work on the duration of our lightning and when it is triggered. Create a VOP network and take the ptnum and divide it by the numpt. Then we'll extract the result into a ramp so we can find the minimum and maximum trigger









> Since the operation is on the point level, we need to promote the trigger frame to the primitive level

frame based on the ramp value. The result of the trigger frame is a range of colour from black to white. Since the operation is on the point level, we need to promote the trigger frame to the primitive level. As for the duration, it is based on a random ptnum that fits between five to eight frames so each lightning bolt will be triggered on a different frame and have a random duration.

04 Set up active lightning and the up vector We will continue our project by creating a new attribute – prim_id. We'll use this attribute later to set our mask for the lightning.

Since we have already defined the trigger frame and the duration for each primitive, we will now set up a condition: if the primitive is triggered, then set the primitive to a new attribute: active. Remember to run this VOP network over the primitives. Next we will use a Blast node to delete any primitives that are not active. If you scrub through the timeline, you should be able to visualise the lightning.

Finally, we will resample the curve to get more points, set the 'up' vector as the tangent in the Polyframe node so we can copy the lightning branches onto the lines and use the UVTexture node to UV the lines.

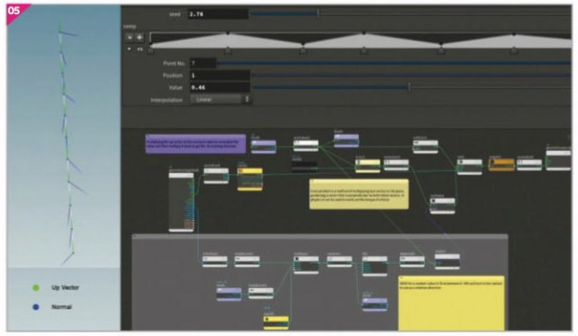
Creating noise as a HDA asset

In this project I added a lot of noise and blends to animate the lightning. We will save a lot of time in the future by saving this setup as a digital asset. Every studio has their own library of digital assets and I strongly suggest you start building your own and creating your own personal libraries. Having these assets on hand will help you speed up your workflow.

Node dependencies setup

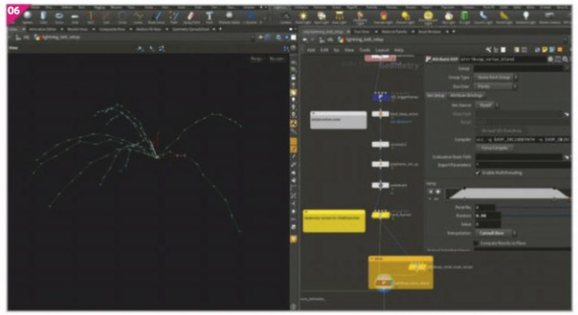
I like to break the setup into three sections. The first section is Input – it's usually the source geometry. This is where you can swap out your geometry and swap in a final model or a higher-resolution object. The second section is setup – inside the setup is where I keep all the effects and simulations. The last section is the render. I like to keep this separate from the other nodes, so that everything looks clean and organised. When I render I will only make this node visible. Most of the time I will import all the final caches and clean up the unused attributes so that the computer doesn't have to spend extra memory to calculate the render before importing into this node.

This way of organising your geometry node will help you to keep track of the node dependencies.



05 Random point normal In this step we'll use the normals to define the direction of the child branches. Firstly we have to take the cross product of the up vector and a constant vector of (0,1,0) to get a perpendicular vector. Then we take the prim_id and multiply it by the ptnum to get a random orientation value between 0 and 360 degrees. Then we will multiply it by the perpendicular vector to get the direction.

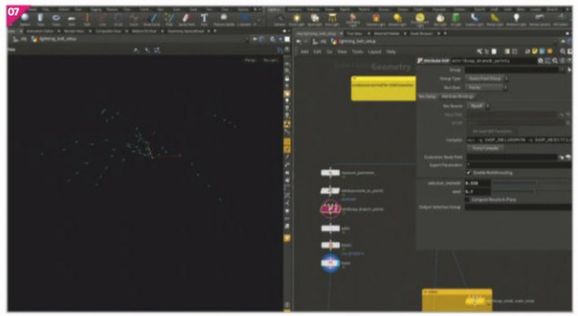
Next we will use the UVs to control both the orientation and perpendicular vectors of the normal.



06 Apply noise on lightning Let's continue by adding some movement to the lightning. Here we have two VOP networks. In the first network we'll add anti-aliased noise into the P (position) and animate it over time. If you look at the lightning the position is all offset.

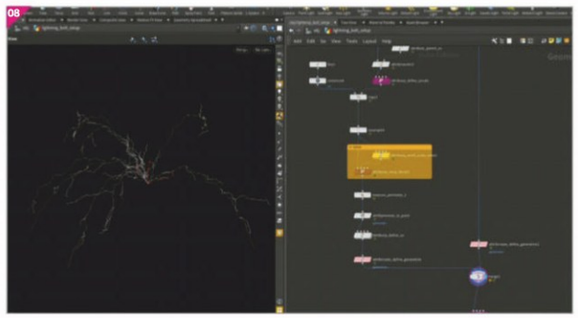
In the second VOP network, we will correct this by blending the original line and the offset line using the UVs.

This way we can keep the start and end point positions when applying noise.



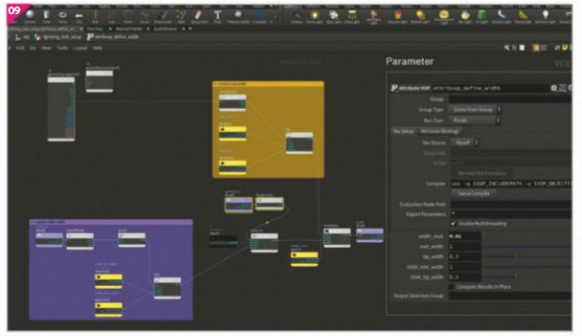
07 Set up branch points To create our branches we'll need to start by measuring the perimeter. The perimeter value will be used to multiply and set the pscale value for the child branches. Let's create a new VOP network to randomly colour our line black and white using the prim_id. This way we can remove some points and keep only a few points as our branch points.

If you refer to the screen grab you can see that I set a threshold and a seed in order to create iterations of branch points based on the seed.



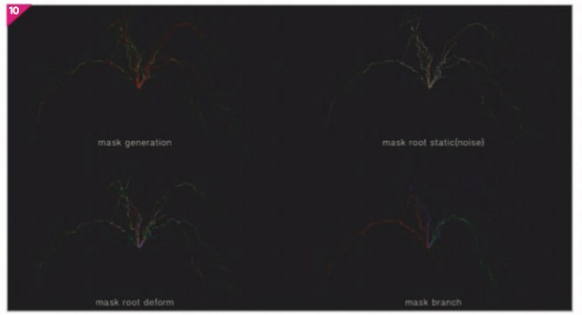
08 Apply noise and UV the child branches Next we will create a line and UV it to copy onto the branch points that we just created. We will also set up a random pscale for each child branch to have different length branches. Next we will resample the lines and apply the same noise technique that was used previously to animate the line shape. Once again, we'll measure the perimeter of the child branches so we can normalise them and determine the correct UV attribute.

Following that, we will create a new attribute 'generation' for the child branches and set the value to 1 and 0 for the attribute on the parent root. This generation attribute will be used to create the mask for the lightning.

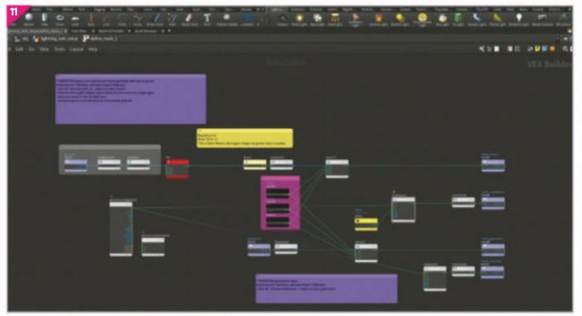


09 Adjust the lightning width Before we drop down a polywire for the lightning, we will set up the width attribute to have better control over the thickness of the lightning. First let's take the UV and convert it from a vector to float and then connect it with a fit node. The fit node will re-range a min and max value as the root width and tip width. For the child branches, we will use a bind to bring in the child UV and then set up the same controls on the child branch's tip width.

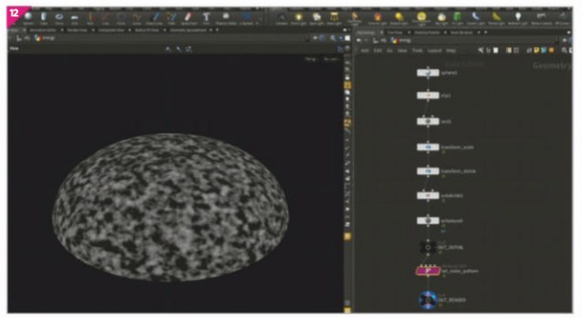
Next, let's bind in the 'generation' attribute. Remember we have two separate values on 'generation' for the child branch and root. Here we will be using this attribute to switch between a constant value of 1 and the child UV and then multiply the result to the root UV. Finally, we'll bind export the result as the width attribute. This way we'll have better control over the lightning bolt thickness.



10 Customise lightning AOV for rendering Lastly we will create four different masks to use in compositing: mask generation, mask root static (noise), mask root generation and mask branch. The mask generation separates the child branches from the root. The mask root static is a grain on the lightning. The mask root deform will be used to mask the lightning emission density and allow for post glow effect. The mask branch will be used to isolate individual groups of the lightning.



11 Lightning mask setup For the mask branch we will take the prim_id, randomise the value and then input the value into a switch over three different constant vectors (R,G,B). Next, for the mask root deform apply a 3D turbulence noise and link the position to the global geometry. Make sure to normalise the output vector at this point. We will also animate the offset to provide some subtle motion to it. For the mask generation, take the generation attribute and input it into the switch node to pick up the colour from different generation values. Finally, the mask root static is similar to the root deform, the only difference here is that it is static.



12 Set up the shield texture In the last step, we will create a sphere and clip it in half to make the outer layer of the energy shield. Use the transform node to animate the scale over time. We will apply a rest position and then take that rest position and add it to the anti-alias noise to get the texture pattern on the surface. We will then keyframe the offset value to animate the texture.

> "The mask root deform will be used to mask the lightning emission density and allow for post glow effect"

Distortion texture map

In the final compositing I added a distortion map to manipulate the lightning since it's surrounded by an energy shield. You can set up the distortion map using the normal position of the energy shield and distort it in Nuke. This will distort the lightning and make it seem more natural. Be careful not to distort it too much – a little goes a long way.