Solidifying of Molten Lava

This page provides a tutorial on creating a lava simulation with Chaos Phoenix in 3ds Max.

Overview

This is an Advanced Level tutorial. The workflow for setting up the shot, and the Phoenix settings involved in the simulation are explained in detail. However, creating a production quality shot of a similar nature may require some tweaks to the lighting, materials and/or the Phoenix simulation.

This simulation requires **Phoenix 4.10 Official Release** and **V-Ray Next Official Release** for **3ds Max 2017** at least. You can download official Phoenix and V-Ray from https://download.chaos.com. If you notice a major difference between the results shown here and the behavior of your setup, please reach us using the Support Form.

The instructions on this page guide you through the process of using Phoenix's Variable Viscosity capabilities in order to simulate molten lava or metal cooling and hardening over a period of time.

The **Download** button below provides you with an archive containing the scene files.

Download Project Files

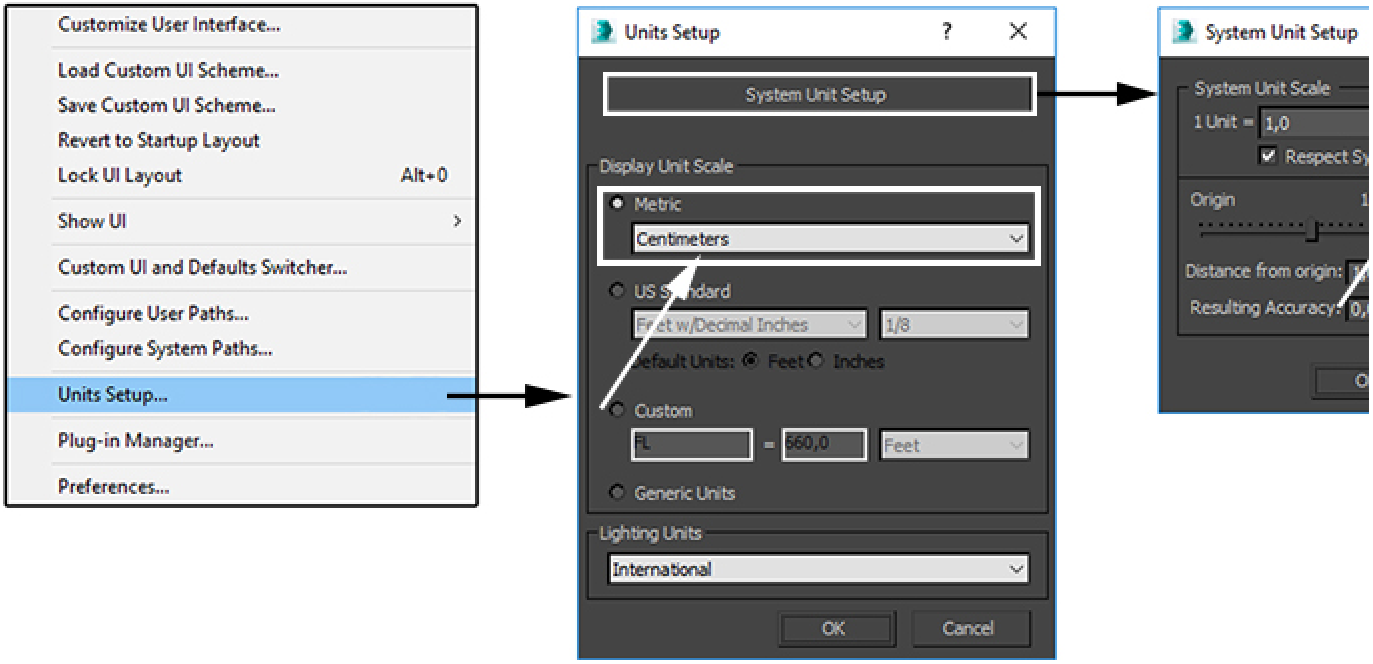
Units Setup

Scale is crucial for the behavior of any simulation. The real-world **size of the Simulator** in **units** is important for the simulation dynamics. Large-scale simulations appear to move more slowly, while mid-to-small scale simulations have lots of vigorous movement. When you create your Simulator, you must check the **Grid** rollout where the real-world extents of the Simulator are shown. If the size of the Simulator in the scene cannot be changed, you can cheat the solver into working as if the scale is larger or smaller by changing the **Scene Scale** option in the **Grid** rollout.

The Phoenix solver is not affected by how you choose to view the Display Unit Scale - it is just a matter of convenience.

Go to **Customize Units Setup** and set Display Unit Scale to **Metric Centimeters**.

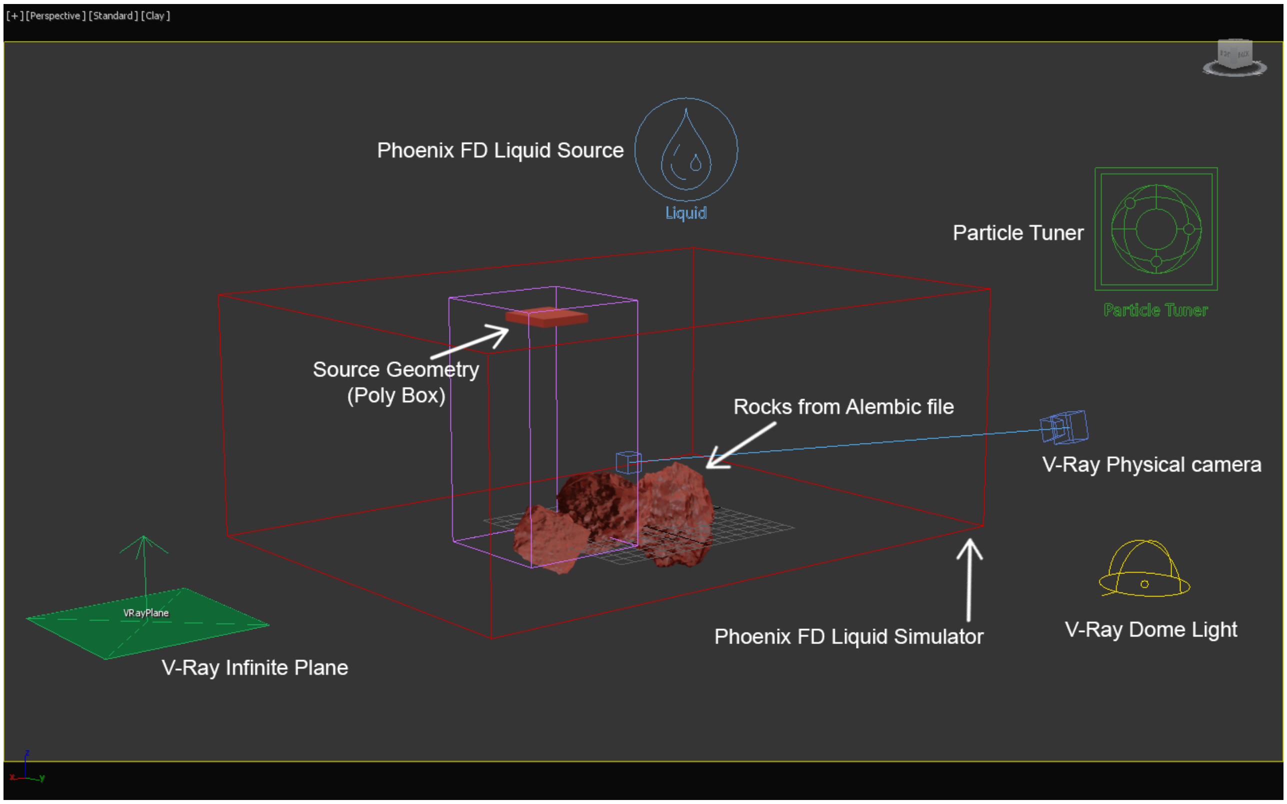
Also, set the **System Units** such that **1 Unit** equals **1 Centimeter**.



Scene Layout

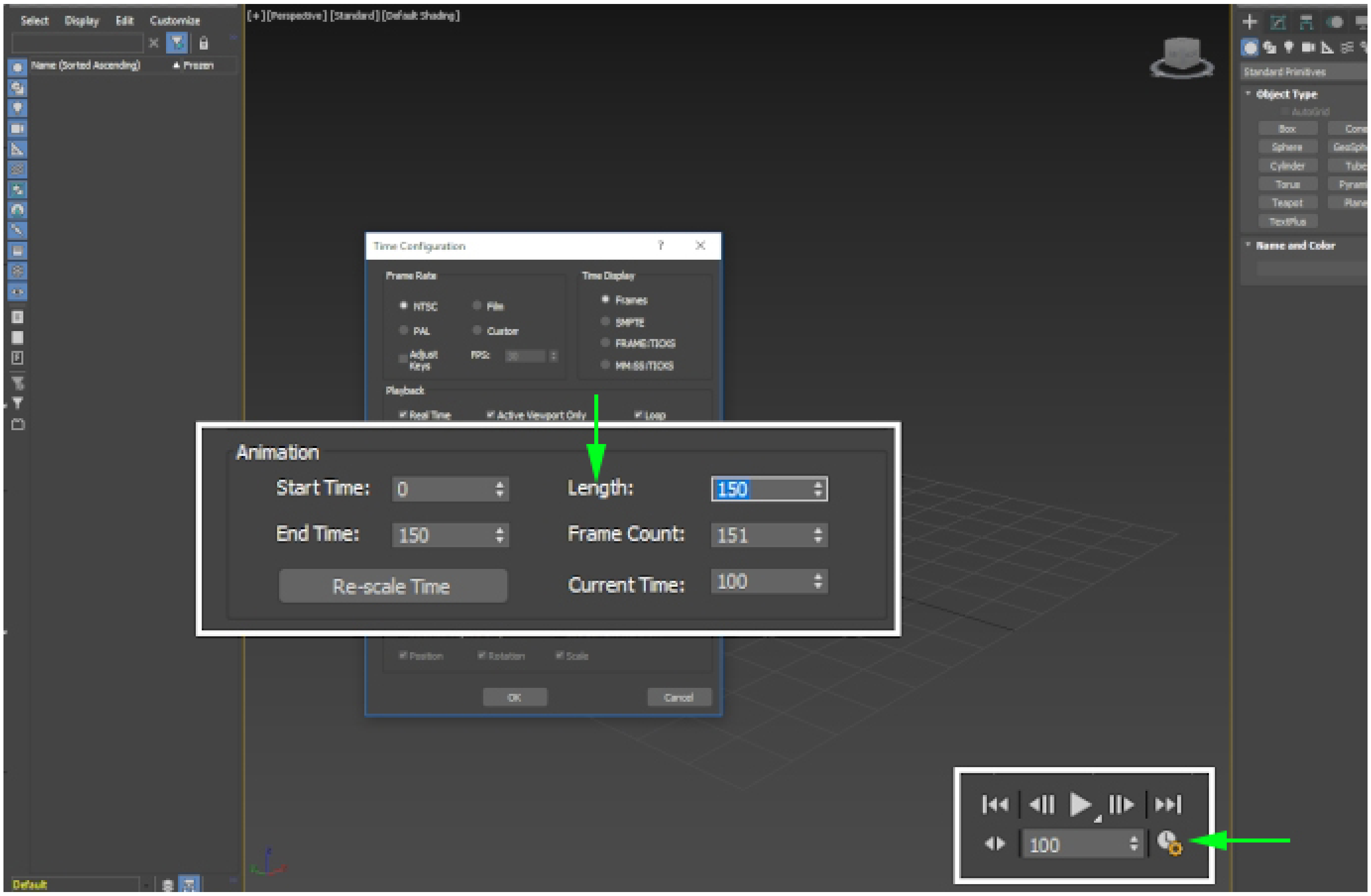
The final scene consists of the following elements:

1. A Standard Primitives **Box** used as the source geometry for the liquid. An animated **Noise** modifier is applied to the geometry to break up the emission.
2. A set of rocks provided with the **rocks.abc** file.
3. A **Phoenix Liquid Source** with the **Box** in its **Emitter Nodes** list. The Source is in **Surface Force** mode and **Noise** is enabled for extra randomization.
4. A **Phoenix Liquid Simulator** with some tweaks in the Grid, Dynamics and Rendering rollouts.
5. A **Particle Tuner** used to tweak the **Viscosity** of the liquid during the simulation.
6. A **V-Ray Physical camera** with minor tweaks for final rendering.
7. A **V-Ray Dome Light**.
8. A **V-Ray Plane** used as an infinite ground surface.



Scene Setup

Set the **Time Configuration Animation Length** to **150** so that the Time Slider goes from 0 to 150.

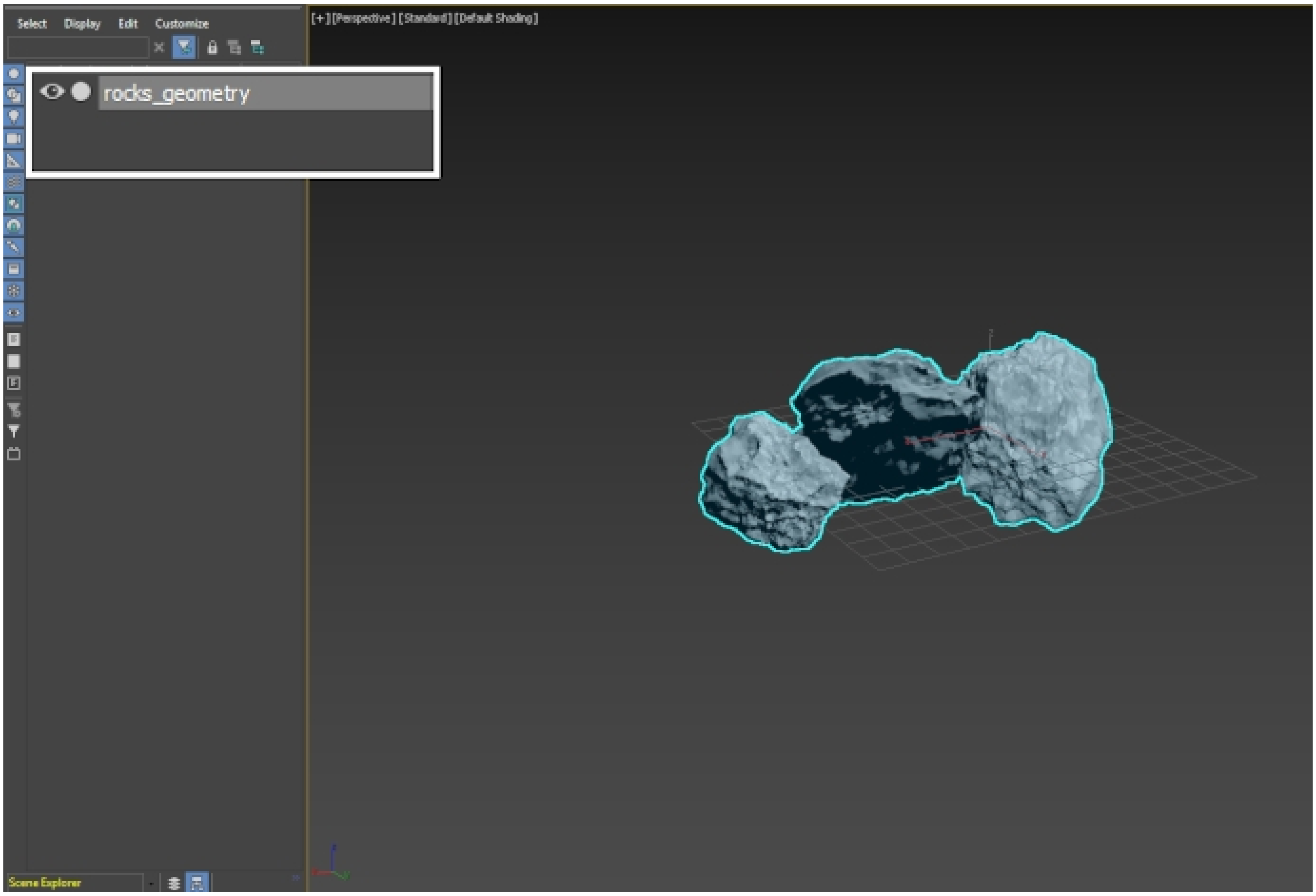


Import the *rocks.abc* geometry by going to **File Import Import...**

We start off with pre-built geometry to save time setting up the scene. Feel free to use your own personal models.

The size of the bounding box of the three rocks used in this example is:

X/Y/Z: [155/60/75].



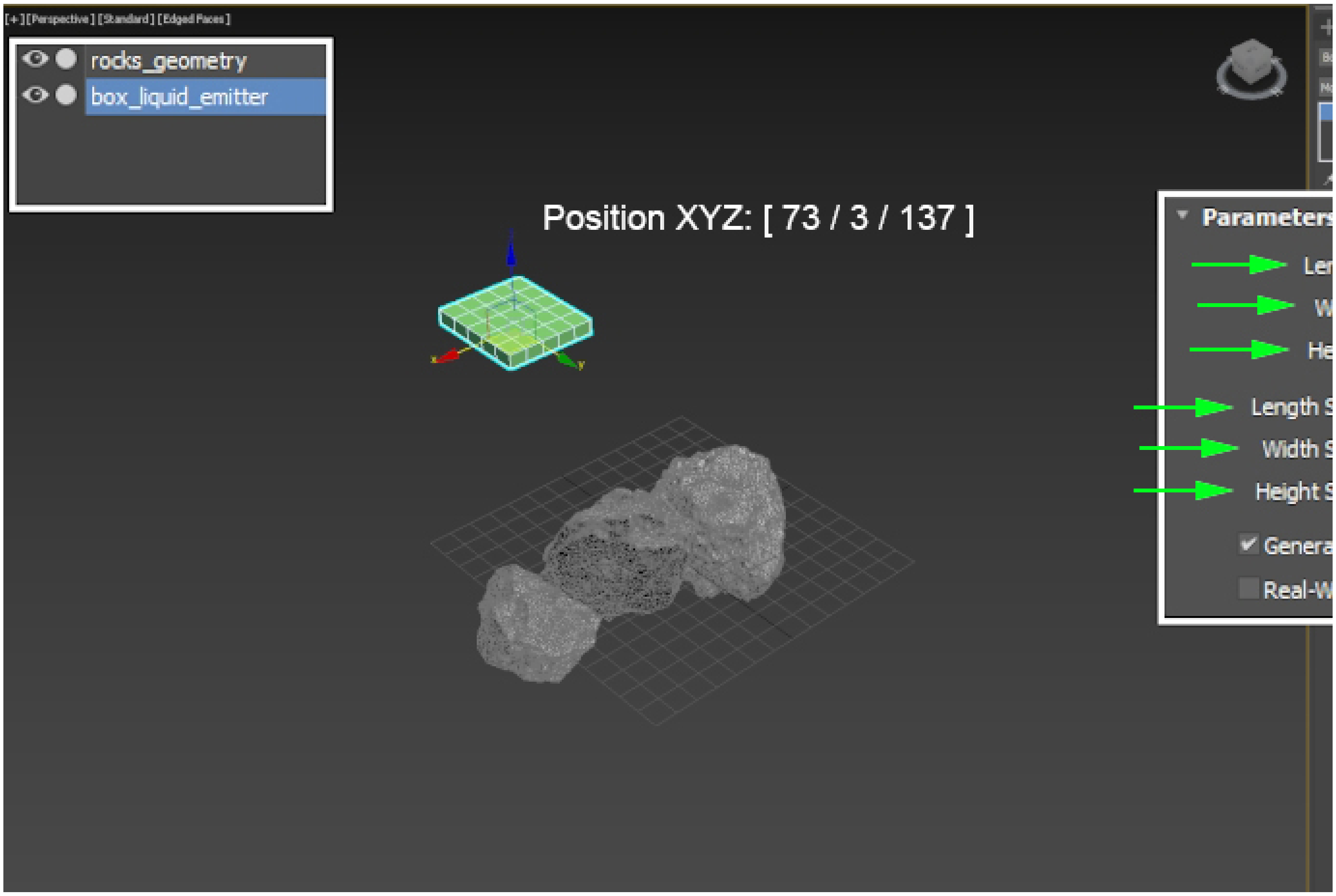
Let's add the geometry used to emit the lava.

Create a **Geometry Standard Primitives Box**.

Use the **Move** tool to place the box above the rocks. The exact transformation values are **X/Y/Z: [73/3/137]**.

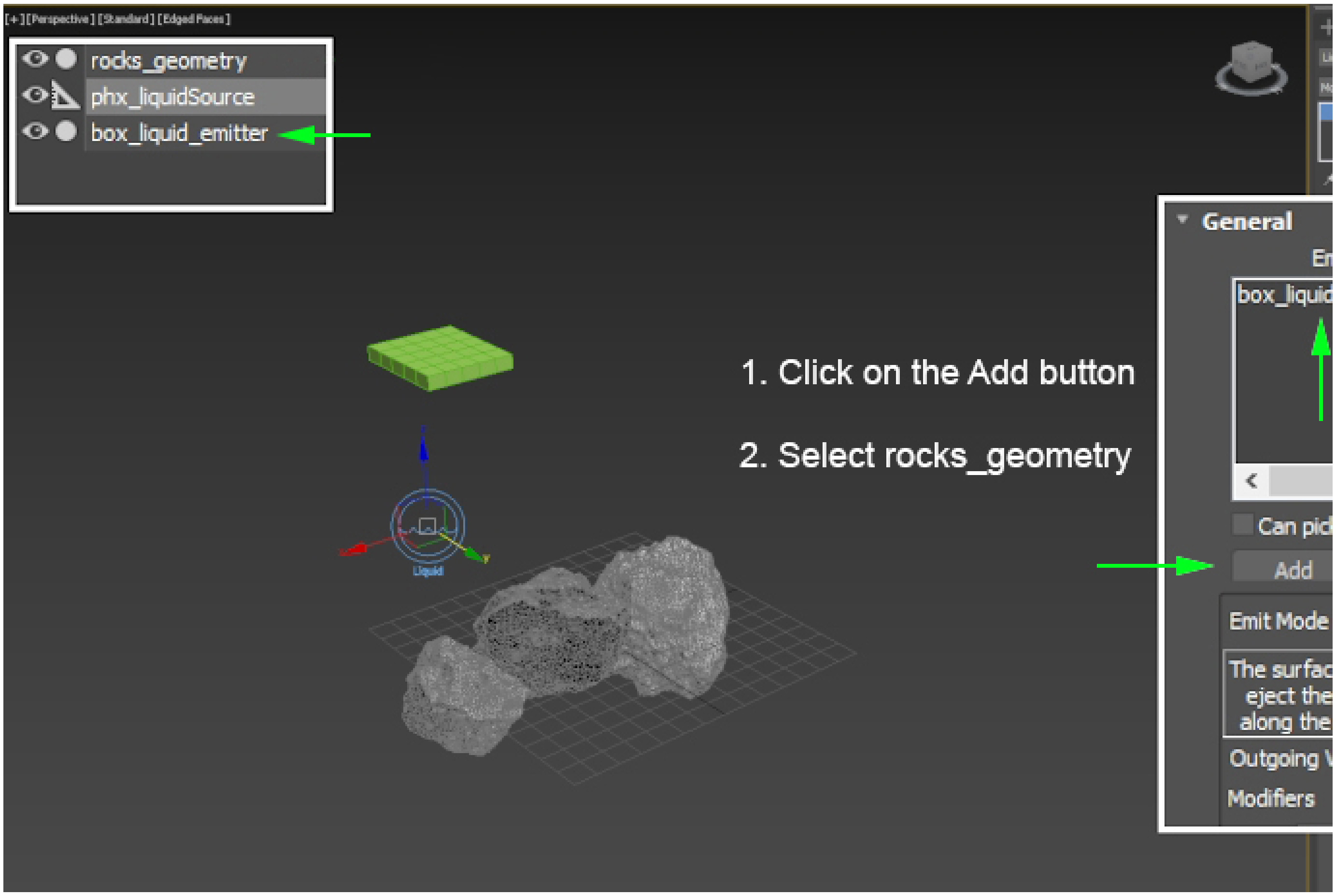
Set the **Length/Width/Height** to **[38/33/5]**, and the **Subdivi sions** along the **Length/Width/Height** to **[5/5/1]**.

The subdivisions are needed later in this tutorial when a Noise modifier is added to the Box to randomize the emission pattern.



Add a **Helpers Phoenix FD Liquid Source**. The Liquid Source is a Phoenix helper node used to tell the Simulator which objects in the scene will emit, how strong the emission will be, etc.

Add the **Box** geometry to the **Emitter Nodes** list. For the time being leave all other options at their default values.



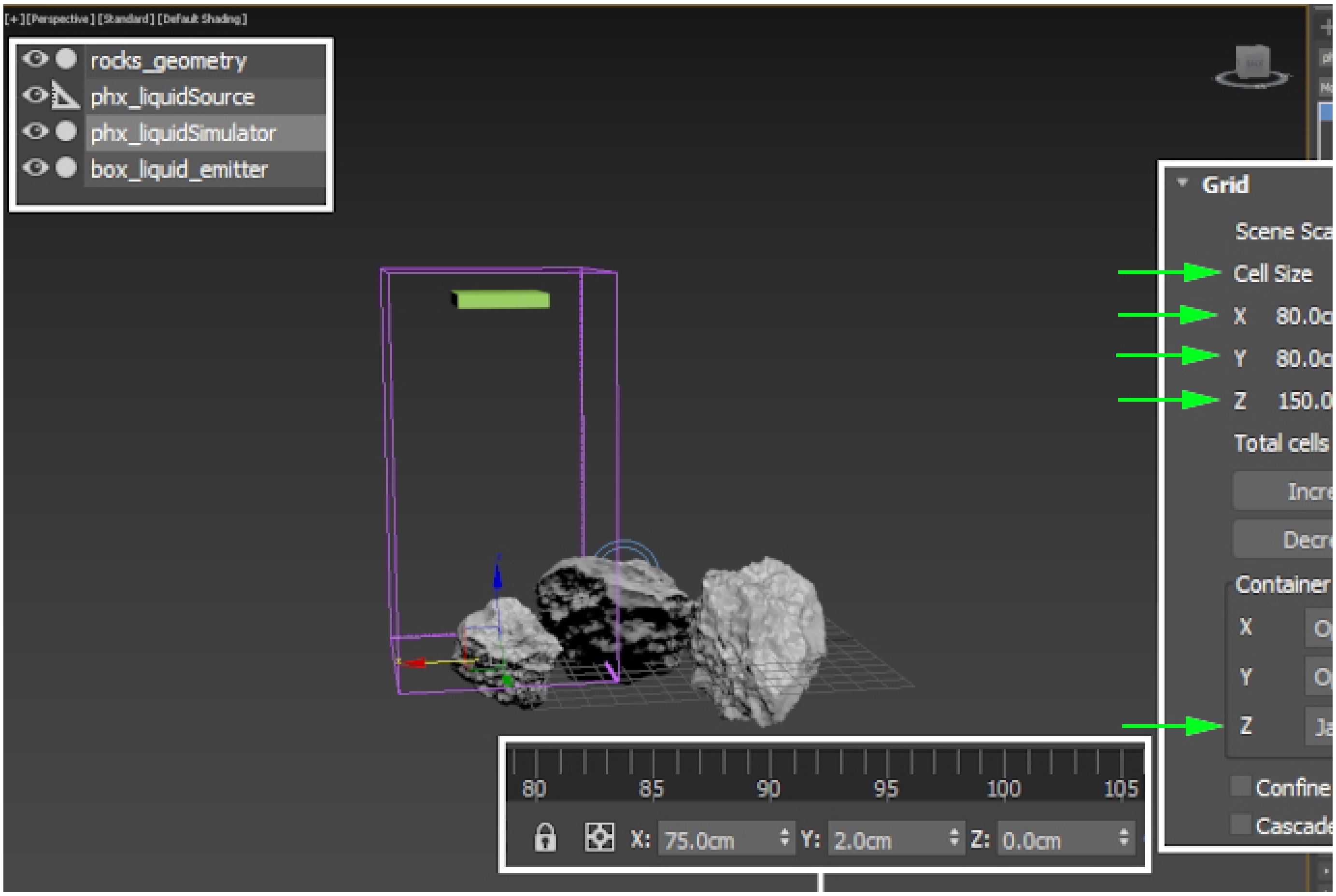
Create a **Geometry Phoenix FD Liquid Simulator**.

Set the **Grid -> Cell Size** to **1.0cm**. The lower the Cell Size is, the more detailed the simulation will be but the longer it will take to complete. We start off with a reasonably small value so we can iterate quickly over different parameter variations.

The **Size** of the Simulator is set to **X/Y/Z: [80/80/150]**.

Set the **Grid Container Walls Z: Jammed (-)**. When a Container Wall is Jammed, it acts as a solid wall against which the liquid will collide.

The exact **Position** of the Simulator in the scene is **X/Y/Z: [75/2/0]**.



To the right is a Viewport Preview showing the result of the simulation so far.

At the moment, **liquid** is being **emitted** from the **entire Box** geometry.

Instead, we would like to **emit only** from the **bottom faces** of the **Box**.

We take a look at one possible solution in the next step.

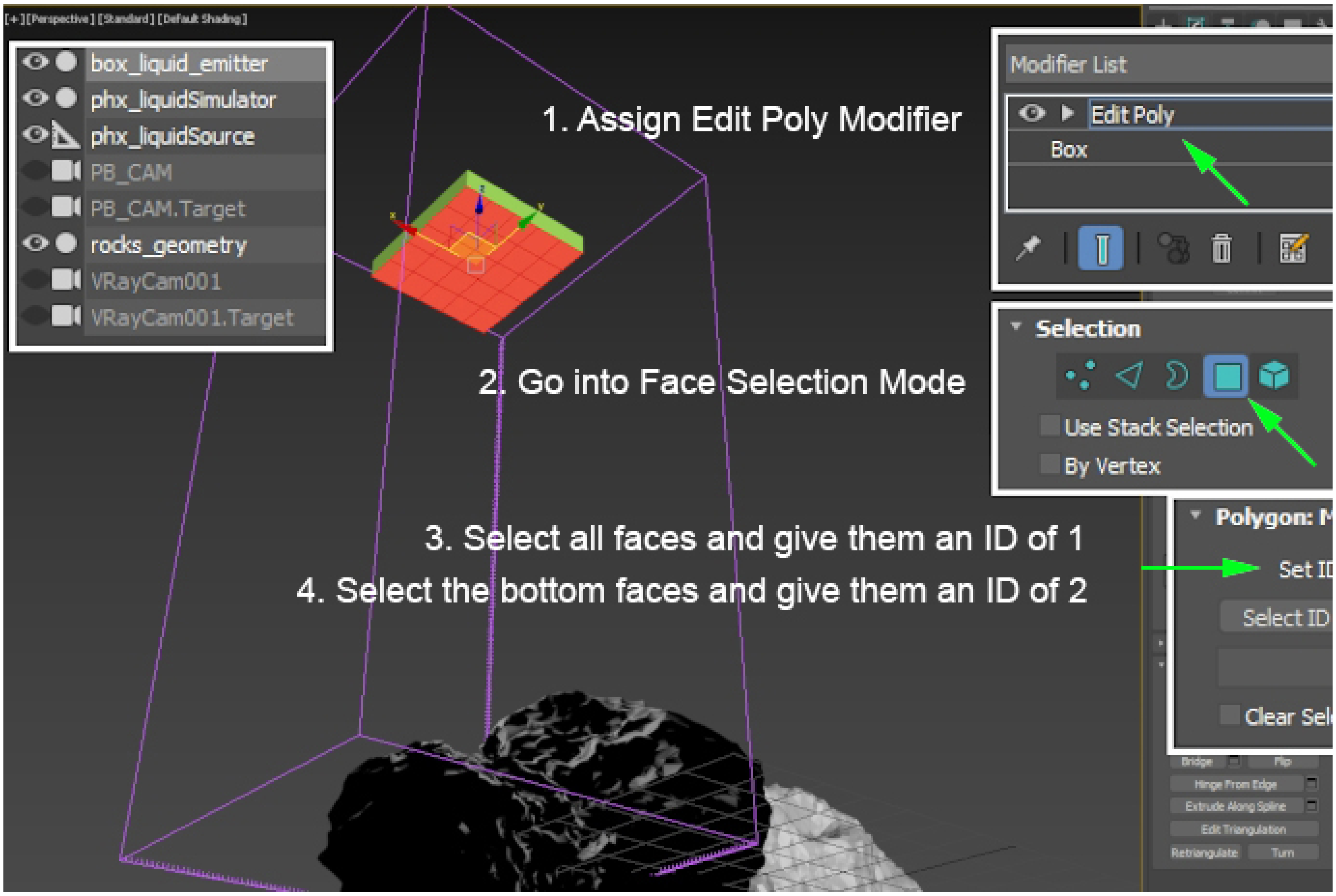
The **Phoenix Source** nodes can use the **Polygon IDs** of an object to limit the emission only to those faces which share a certain ID.

Assign an **Edit Poly** modifier to the **Box** geometry and go into **Face Selection mode**.

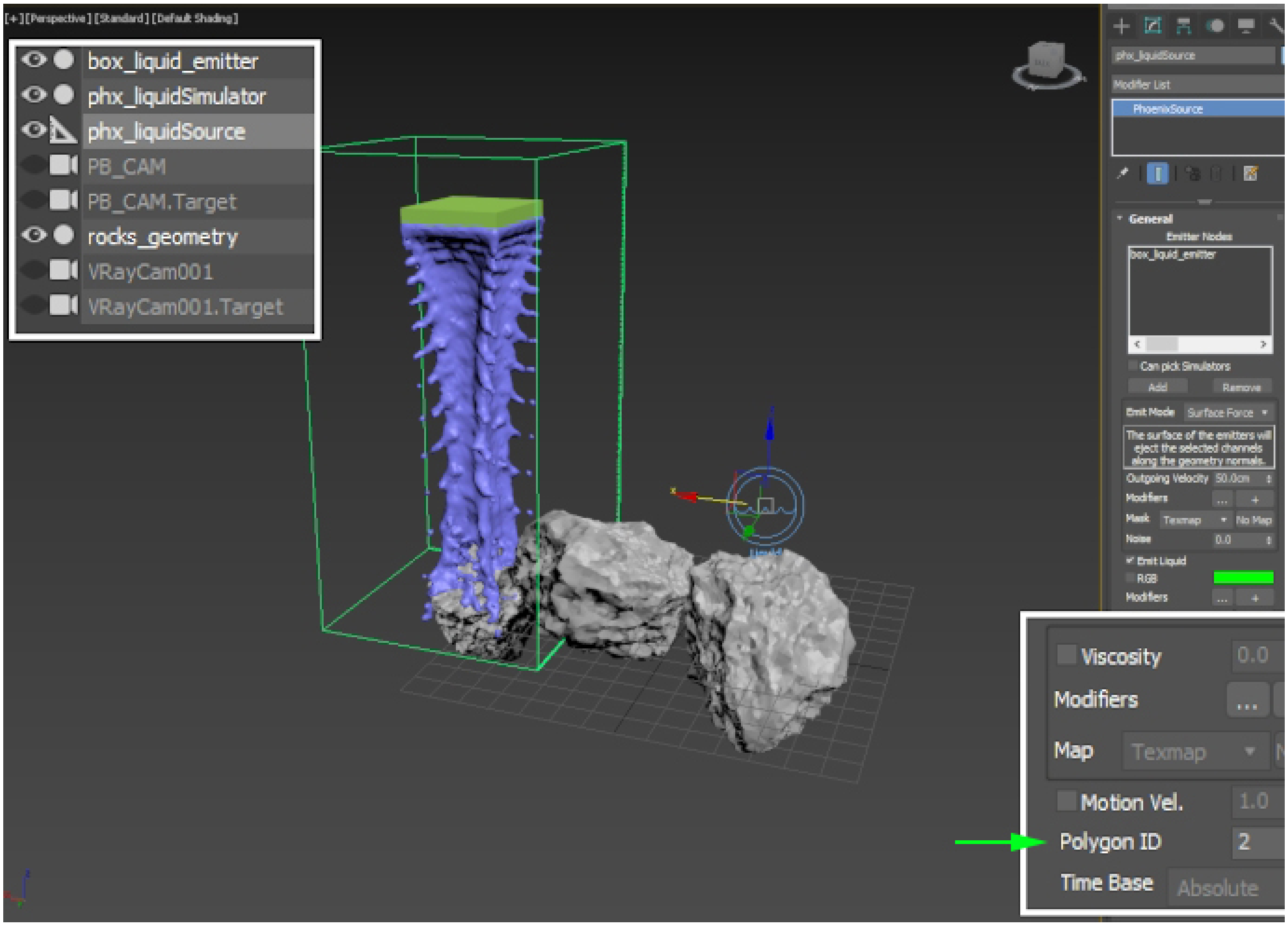
Select **all faces** and give them an **ID** of **1**.

Then, select **only the bottom faces** which will be used for emission of liquid and assign them an **ID** of **2**.

The cameras on the screenshot to the right have been added to the scene to generate the Viewport Previews. The final camera setup will be discussed in the Rendering section of this tutorial.



Set the **Polygon ID** parameter of the **Phoenix Liquid Source** to **2** so that only the faces with this ID are used for emission.

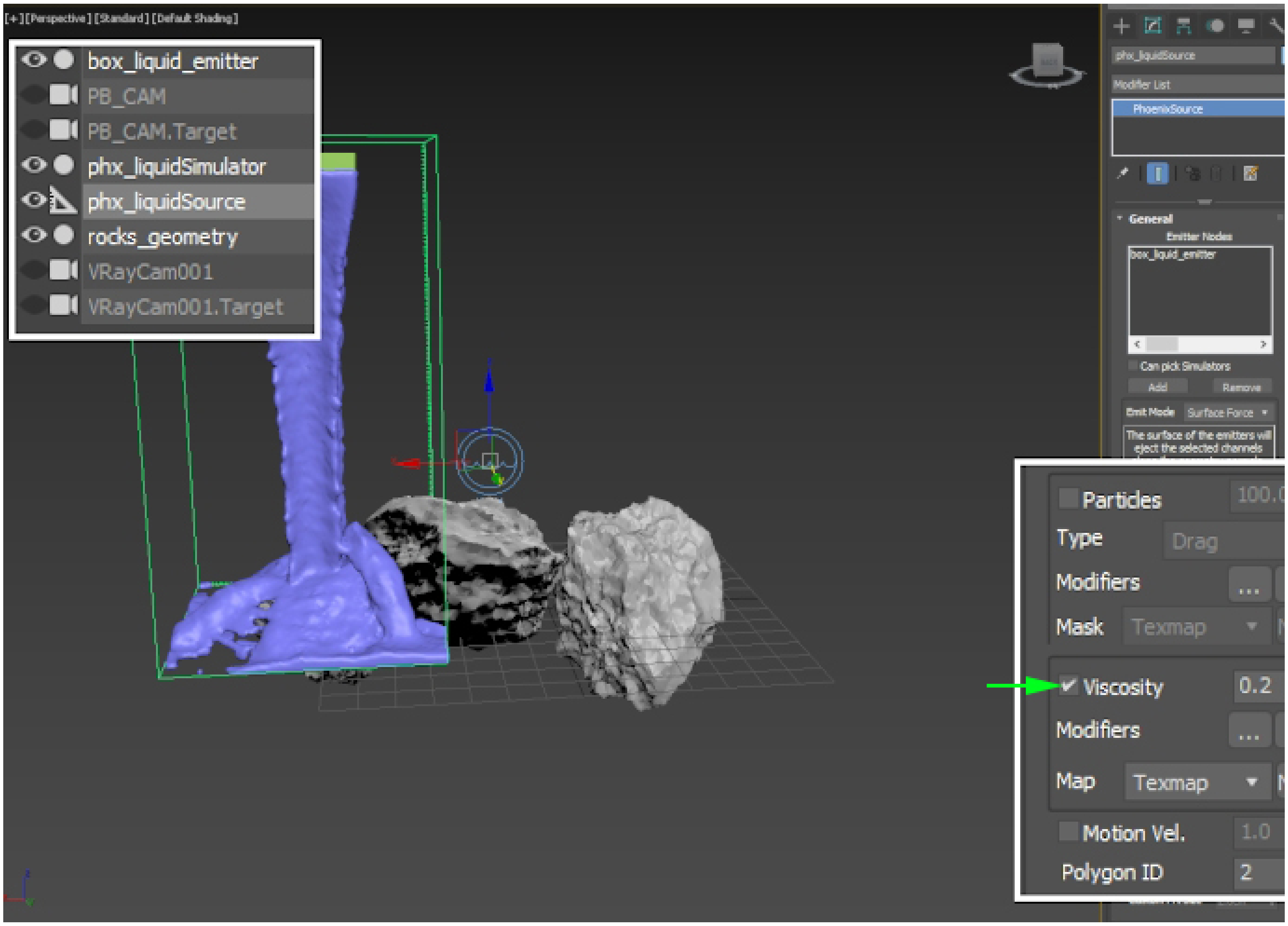


To the right is a Viewport Preview showing the result of the simulation so far.

The **liquid** is now correctly being **emitted** only from the **bottom faces** of the **Box** geometry.

Select the **Phoenix Liquid Source** and set **Viscosity** to **0.2**.

Viscosity emulates thickness - the higher this value is, the more the liquid will resemble thick mud, honey or in this case - lava.



Under the **Output** rollout of the **Phoenix Simulator**, **enable** the export of the **Viscosity Grid Channel**.

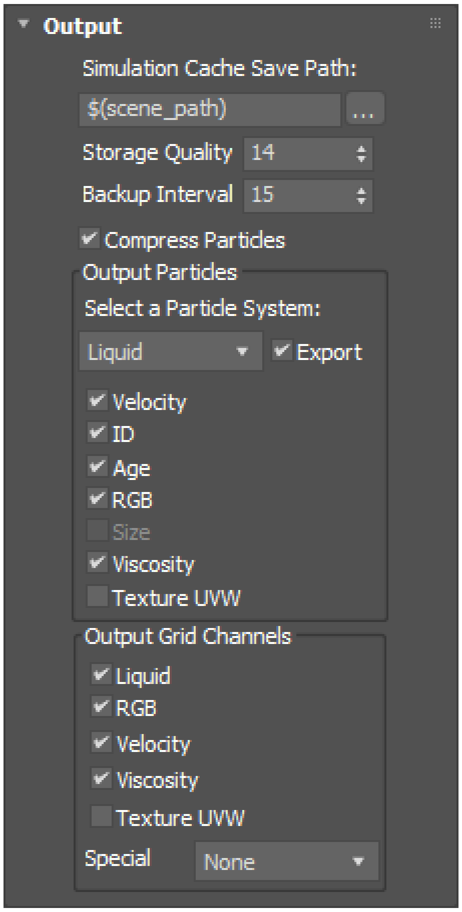
Also enable the export of the **Viscosity Liquid Particle Channel**.

Outputting those channels is required for the calculation of **Vi scosity** for the liquid to work.

Enable the **RGB Channel**. This is required for the simulation of the RGB channel to work.

Make sure to also **enable** the **Velocity Channel**. The Velocity Grid Channel is used when rendering with **Motion Blur**.

Enable the **Age Channel. This channel is used by the** Particle Tuner, so it can randomize the viscosity by the age of the liquid particles.



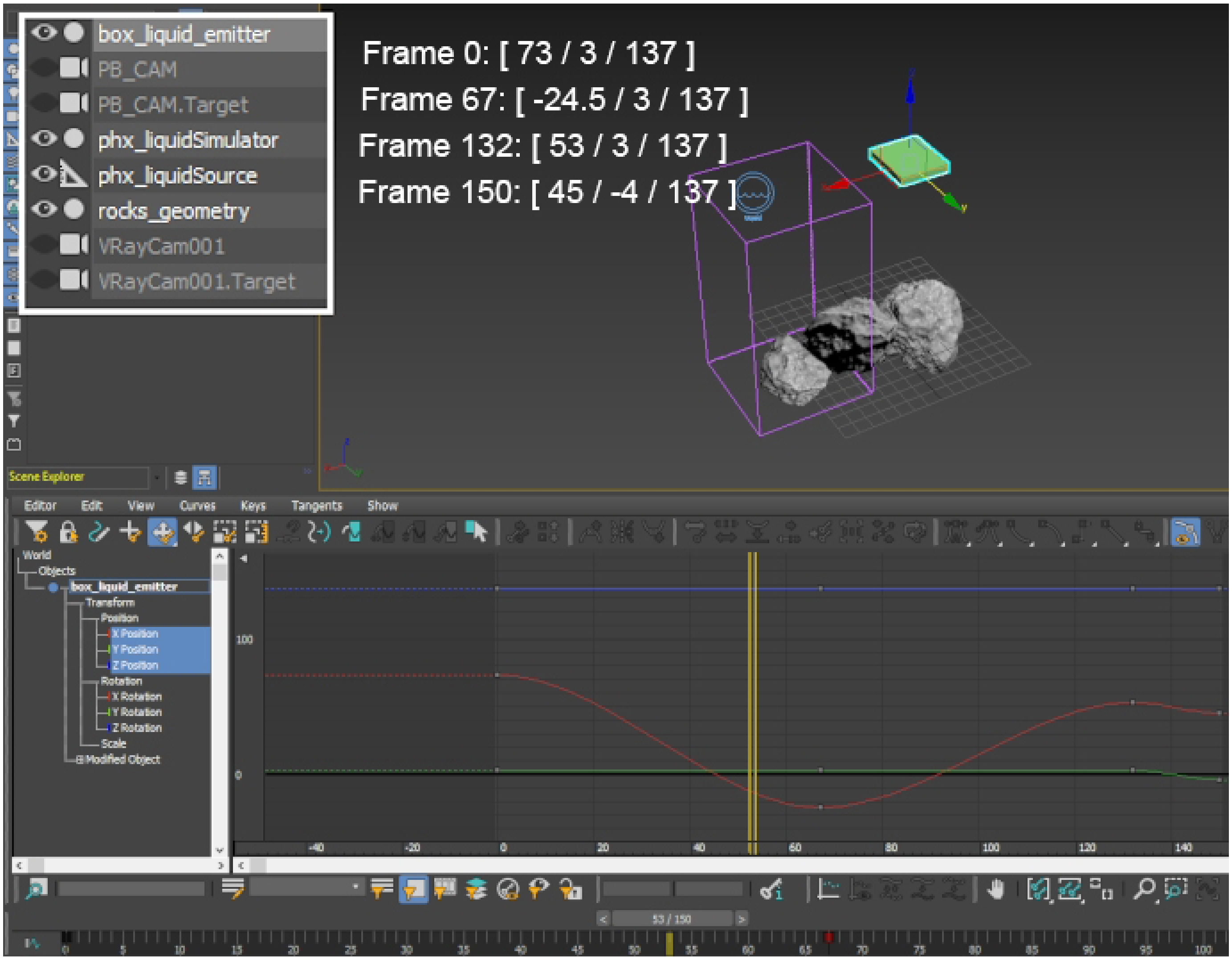
To the right is a Viewport Preview showing the result of the simulation so far.

The **liquid** feels much **thicker** now, and the **stepping artifacts** are much **l ess** pronounced.

In the next step we **animate** the **Box** geometry to move to the right so the lava pours all over the rocks.

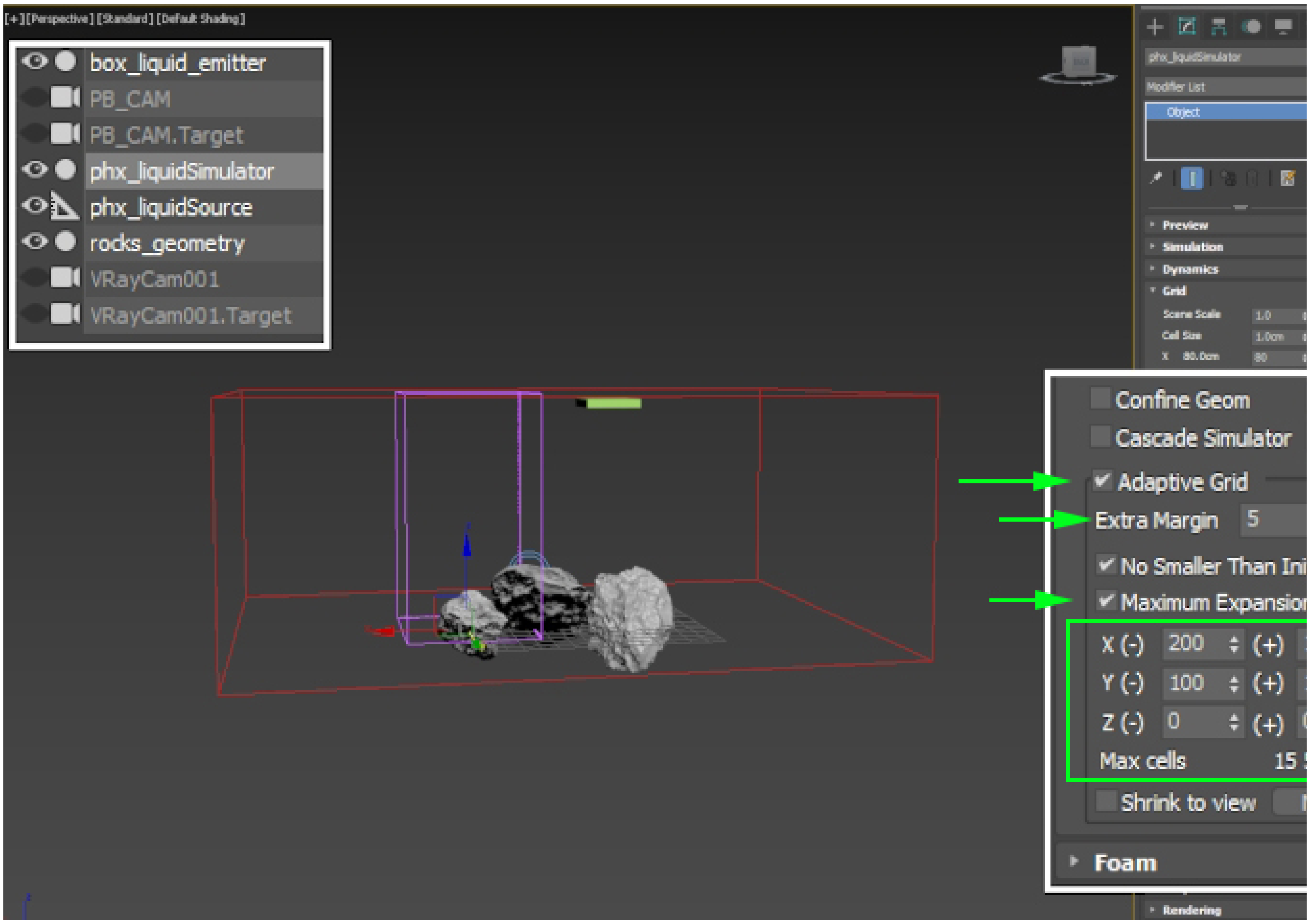
Animate the **Box** geometry with the following **keyframes**:

Frame 0: [X: 73 | Y: 3 | Z: 137]
Frame 67: [X: -24.5 | Y: 3 | Z: 137]
Frame 132: [X: 53 | Y: 3 | Z: 137]
Frame 150: [X: 45 | Y: -4 | Z: 137]



Enable **Simulator Grid Adaptive Grid** and **Maximum Expansion**. The red box in the screenshot to the right is a preview of the Maximum Bounds for the simulation. Adaptive Grid is a huge time saver - the initial grid is dynamically expanded to accommodate the movement of the liquid, cutting down on both processing time and memory. If you notice any clipping, increase the **Extra Margin** to a value of **5 - 10**. This will add a few extra voxels at the borders during adaptation.

The Adaptive Grid **Maximum Expansion** settings are **X: [200, 100]**, **Y: [100,100]** and **Z: [0, 0]**.



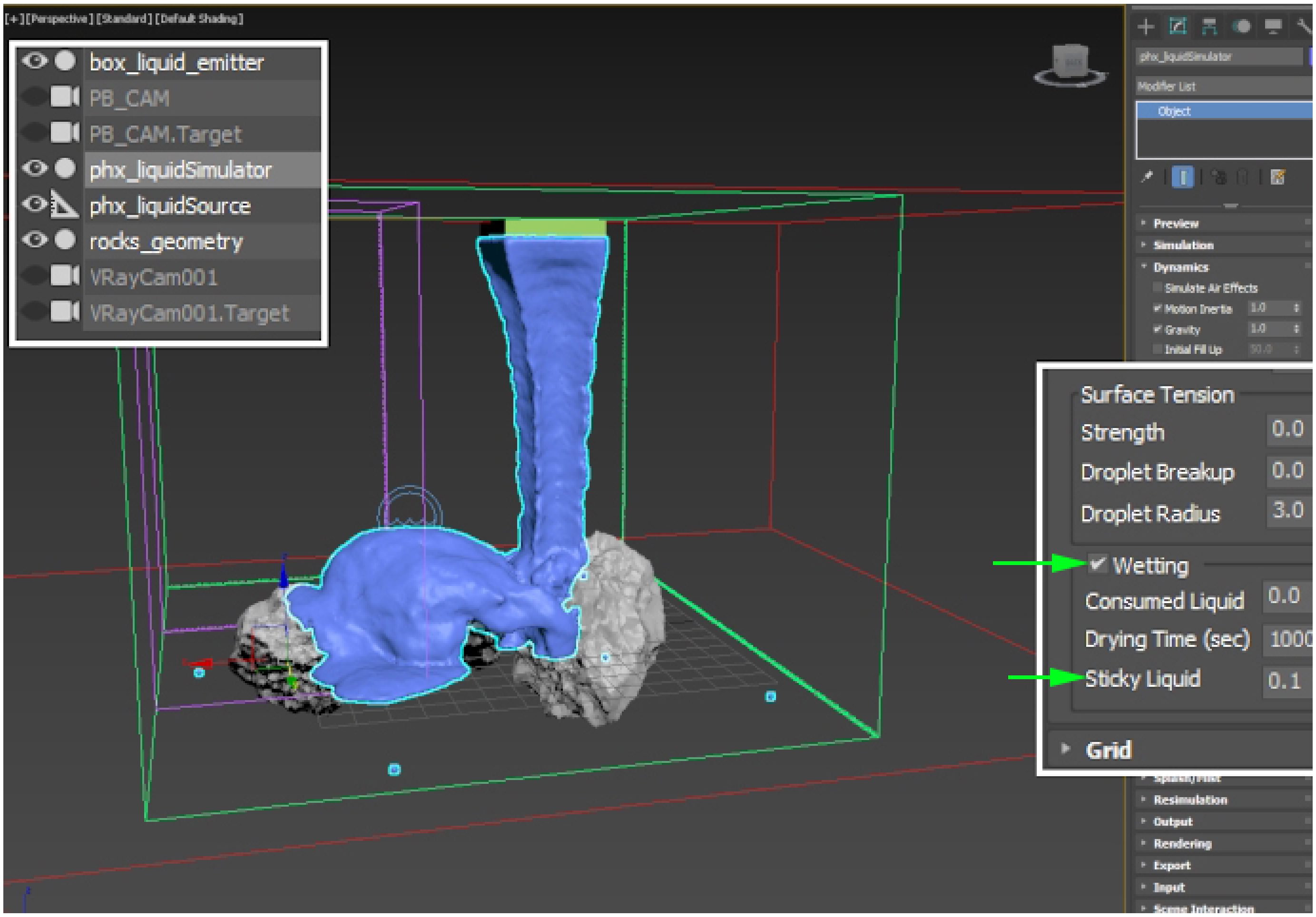
To the right is a Viewport Preview showing the result of the simulation so far.

Now that the **liquid** is pouring all over the **rocks** we can see that it's **not sticking** to them at all.

In the next step we **enable Wetting** so the **lava** can **stick** to the **rocks**.

Enable **Phoenix Simulator Dynamics Wetting**. When Wetting is enabled, Phoenix generates another set of particles called WetMap particles. Those particles are created at the point of contact between the liquid and the scene geometry and can be used to drive shaders (through a Phoenix FD Particle Texture) or specify where the liquid should try to adhere to. You can disable Wetting for a specific object in your scene from its Phoenix Per-Node Properties.

Set **Sticky Liquid** to **0.1**. The lava will now try to adhere to the rocks.



To the right is a Viewport Preview showing the result of the simulation so far.

The lava is now naturally flowing over the rocks.

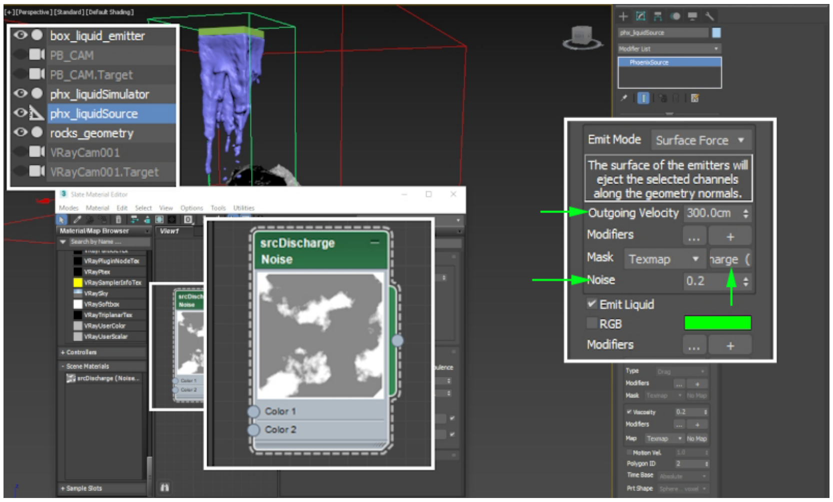
At the moment the entire bottom side of the box geometry is emitting lava giving the fluid an unnatural cubic appearance.

In the next few steps we **randomize the emission** for a more natural looking lava.

Select the **Liquid Source** and set the **Noise** parameter to **0.2**. This option works as a multiplier on the **Outgoing Velocity** - some areas of the emission geometry will emit with a higher Outgoing Velocity while other areas will be weaker.

Click on the "**No Map**" button to the left of **Mask** and plug a **N oise** texture.

Finally, **increase** the **Outgoing Velocity** to **300** - this value is completely arbitrary and can be modified based on the requirements of your setup.



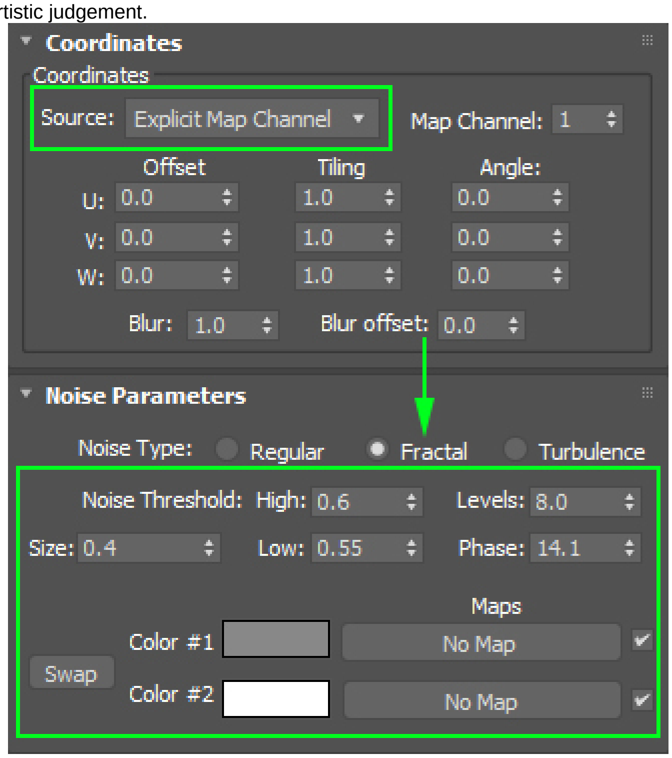
Here are the exact values for the **Noise** texture:

Source: Explicit Map Channel
Type: Fractal
Size: 0.4
Threshold High: 0.6
Threshold Low: 0.55
Levels: 8
Phase: animated with the following **keyframes**:

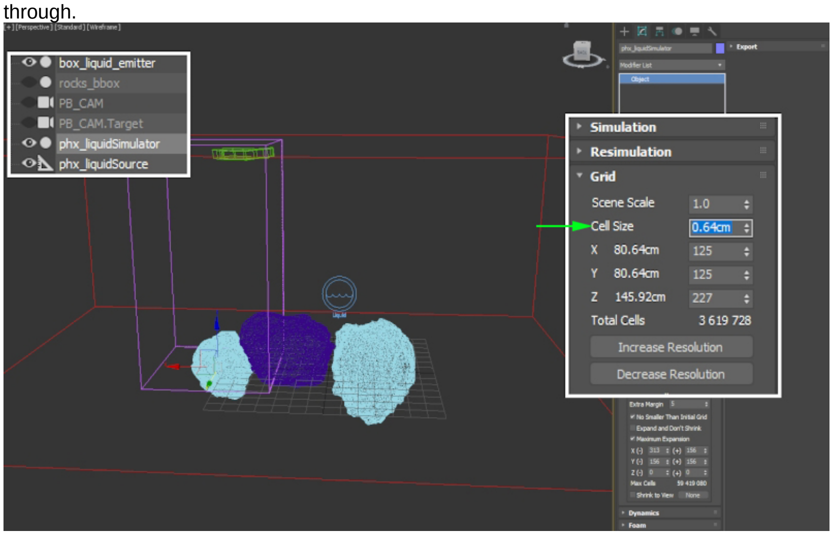
Frame 0: 14
Frame 150: 30

Color 1: RGB [64, 64, 64]
Color 2: RGB [255, 255, 255]

Feel free to tweak the settings based on your own artistic judgement.



Reduce the **Phoenix Simulator Grid Cell Size** to **0.64** so the extra detail provided by the Noise texture can come through.



To the right is a Viewport Preview showing the result of the simulation so far.

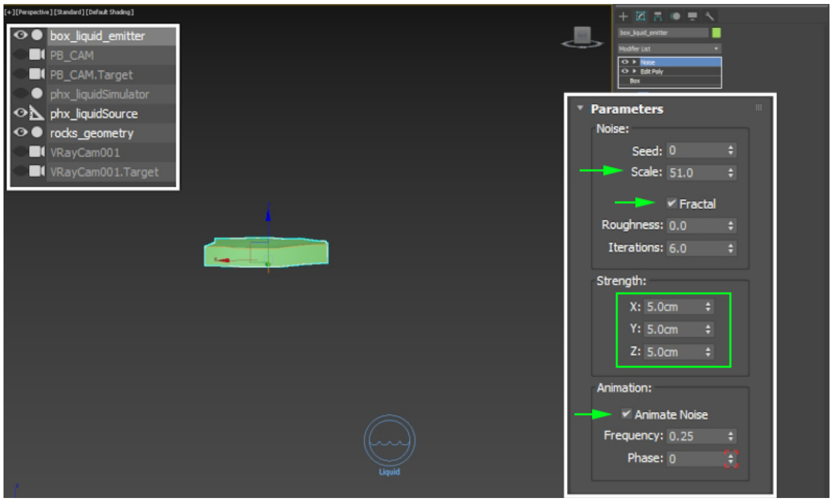
The emission pattern is now much more natural.

Furthermore, the **higher Outgoing Velocity** is causing the **lava** to **splash** as it hits the rocks.

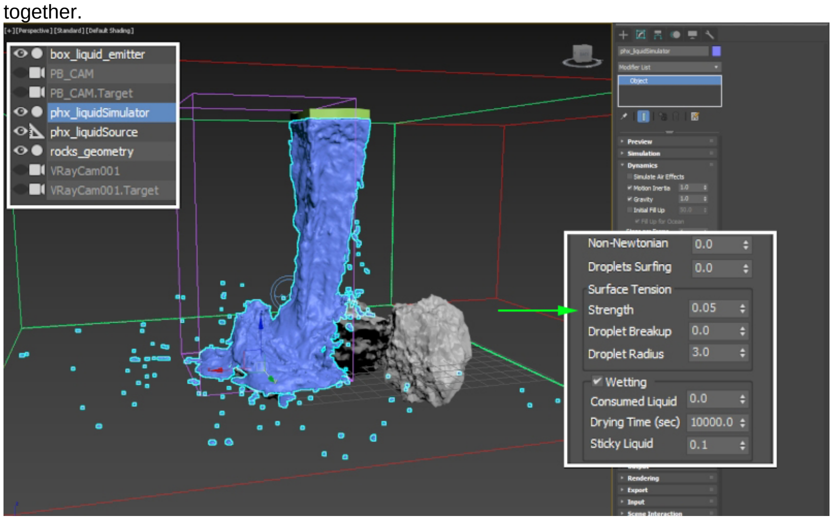
For even more randomness, add a **Noise** modifier to the **Box** emission geometry:

Here are the exact values used in this setup:

Scale: 51
Fractal: Enabled
Strength X/Y/Z: 5
Animate Noise: Enabled



Increase the **Phoenix Simulator Dynamics Surface Tension** to **0.05**. Surface Tension helps keep the liquid together.



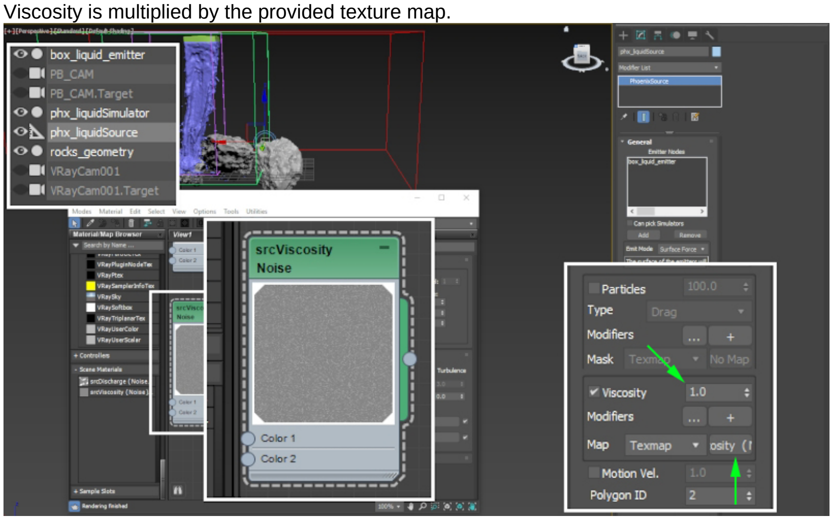
To the right is a Viewport Preview showing the result of the simulation so far.

At the moment the **viscosity** of the lava **is completely uniform**. Real lava has different viscosity depending on its temperature. Hot lava resembles liquid and stiffens as it cools down.

In the next step we **randomize** the **Viscosity** with a **Noise** texture.

Click the **No Map** under **Phoenix Liquid Source Viscosity Map**. Add a **Noise** texture from the pop-up menu.

Increase the **Viscosity** parameter to **1.0**. Usually a value of 1.0 is way too high but since the Noise texture has high-frequency detail, with most of the values in the texture much below 1.0, the final result will be significantly lower after the Viscosity is multiplied by the provided texture map.



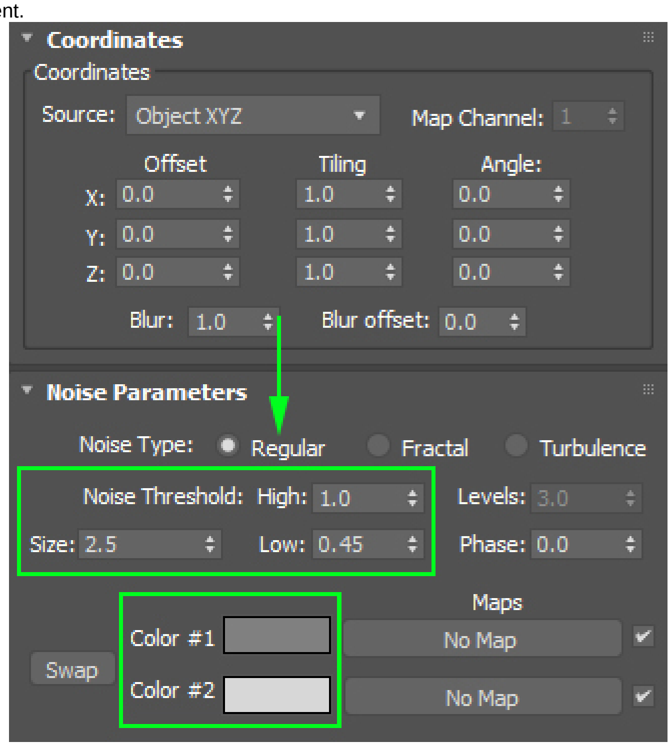
Here are the exact values for the **Noise** texture:

Noise Type: Regular
Size: 2.5
Threshold High: 1.0
Threshold Low: 0.45
Phase: animated with the following **keyframes**:

Frame 0: 0
Frame 150: 6

Color 1: RGB [56, 56, 56]
Color 2: RGB [175, 175, 175]

Feel free to tweak the settings based on your own artistic judgement.



To the right is a Viewport Preview showing the result of the simulation so far.

The difference after modifying the **Viscosity** with a **texture** is not immediately obvious but once the Cell Size is reduced for the final simulation, individual chunks of lava will start forming based on the different viscosity values.

Solidifying with a Particle Tuner

In this section of the tutorial, we look into the process of using a Particle Tuner to increase and randomize the viscosity of the lava over time. Real-world lava solidifies as it cools down and we want to replicate this behavior.

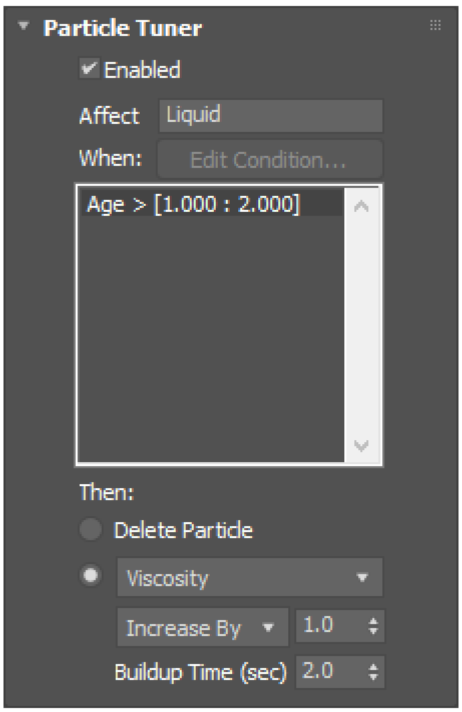
The Particle Tuner assesses all particles in the simulation and changes their values if they pass a certain condition.

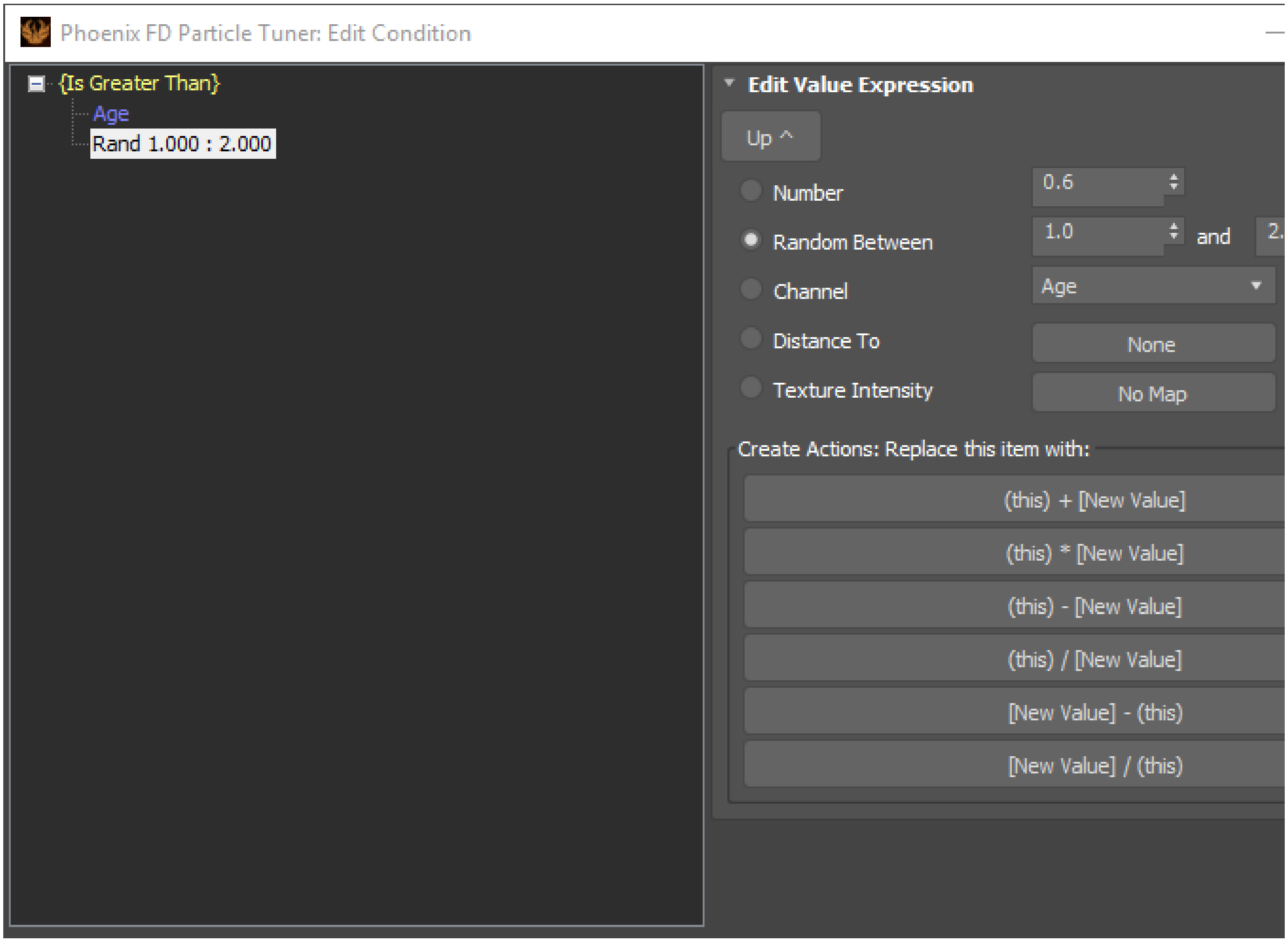
In this example we will raise the viscosity of particles as their age increases.

The conditions can be very simple, but you can also build more complex conditions with the Particle Tuner's expression operators.

In this case the **Particle Tuner expression tree** is created as follows:

- If the **Age** of a particle is **Greater Than** a **Random** v alue **Between 1.0** and **2.0** seconds
- Then the **Viscosity** is set to **Increase By 1.0**
- Over the **Buildup Time** which is set to **2.0** seconds.

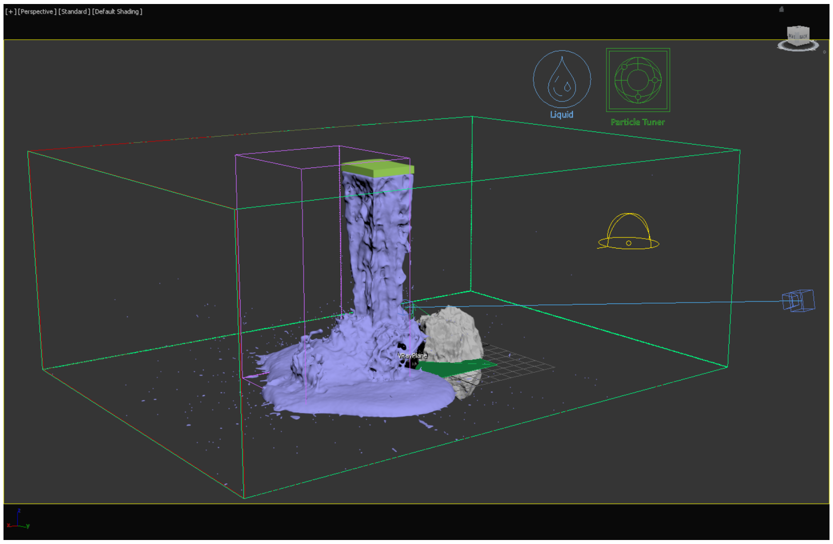




To the right is a preview showing the result of the simulation so far.

The lava now splashes as it hits the rocks but calms down and solidifies as it settles on the ground.

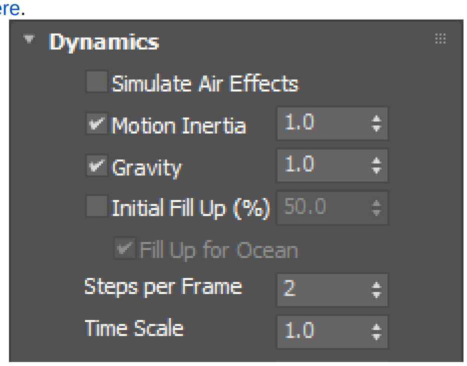
Note the small droplets of lava flying away from the rocks geometry - in the next couple of steps we will fix this problem by increasing the **Steps Per Frame** and the **Scene Scale**, and reducing the **Cell Size** of the simulation.



There are many stray particles flying all over the place as the lava collides with the rocks. Not only is this effect unnatural but it also drastically increases the simulation time by expanding the Adaptive Grid to its maximum extents.

Increase the **Phoenix Simulator Dynamics Steps Per Frame** to **2** to remedy this.

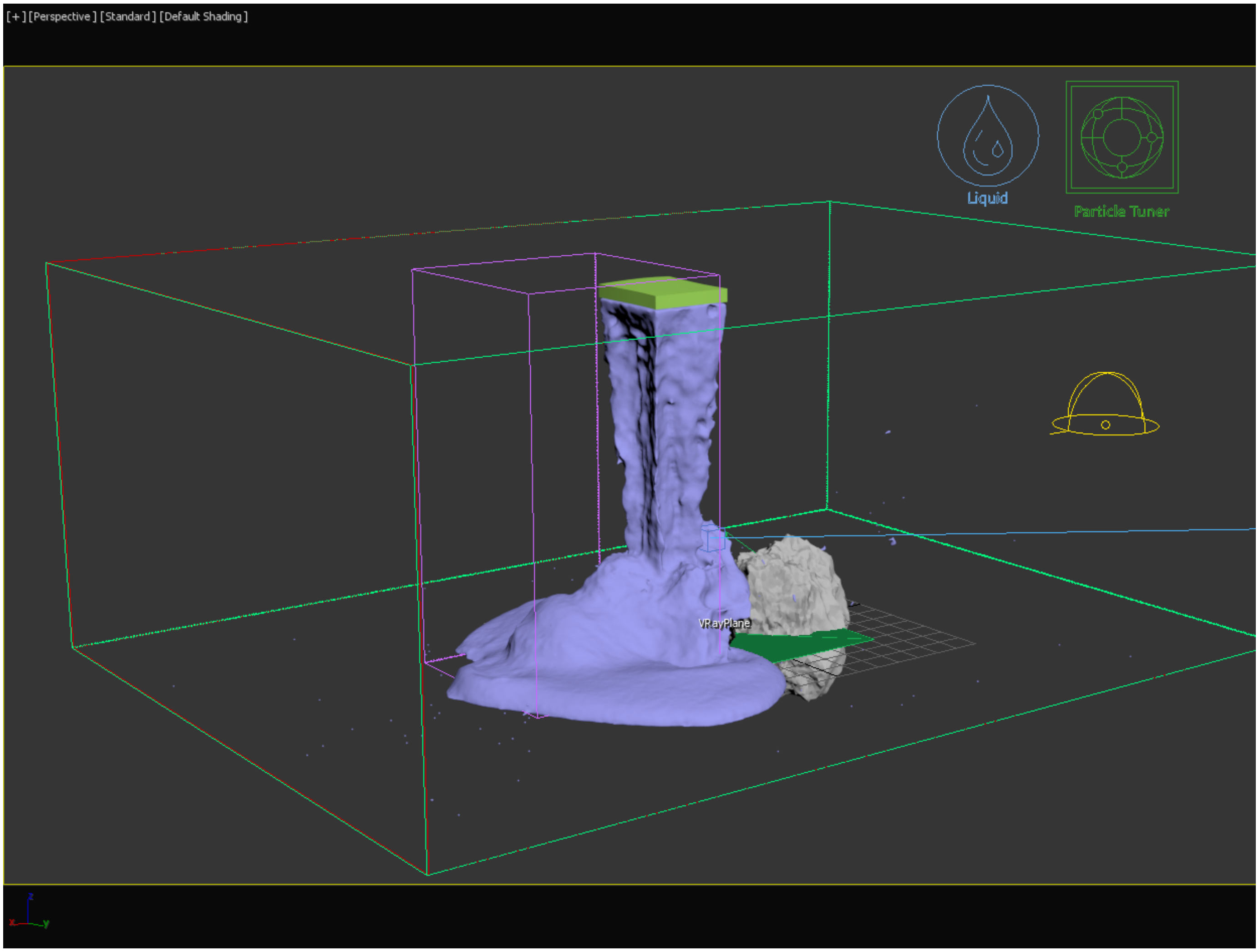
Increasing the Steps Per Frame will usually produce calmer fluids but will increase the simulation time. Each simulation step kills fine details, and thus for maximum detail it's best to use the lowest possible SPF that runs without any of the issues mentioned above. You can find more information on Steps Per Frame here.



To the right is a preview, showing the result of the simulation so far.

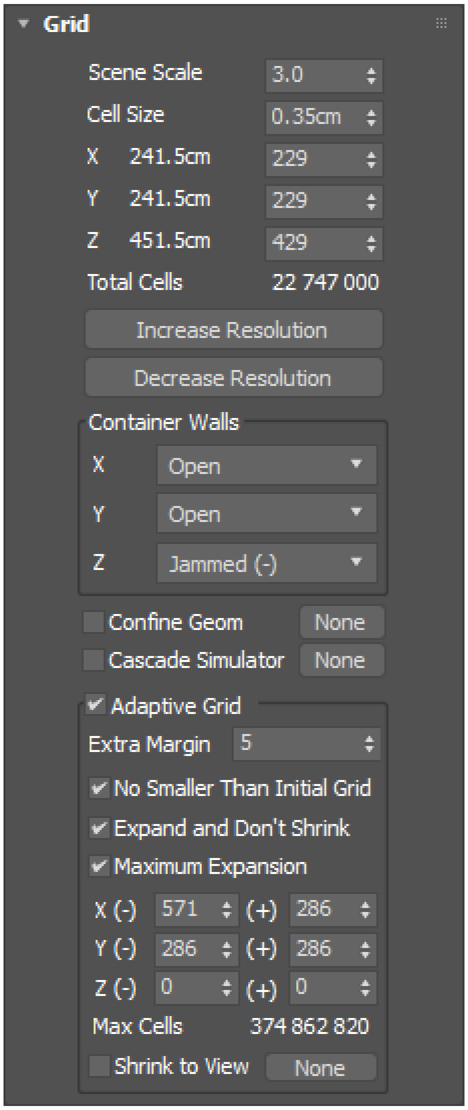
Increasing the **Steps Per Frame** substantially reduced the detail in the simulation but also helped to resolve the problem with individual droplets flying all over the place.

We can further reduce the **Cell Size** and thus increase the grid resolution to gain back some of that detail.



Increase the **Scene Scale** to **3.0**. Bigger scale would make the fluid move more slowly.

Reduce the **Cell Size** to **0.35**.



To the right is a Viewport Preview showing the result of the simulation so far.

This is the **final** shape of the **Lava simulation**.

We increased the **Steps Per Frame** to **eliminate** the **stray droplets**.

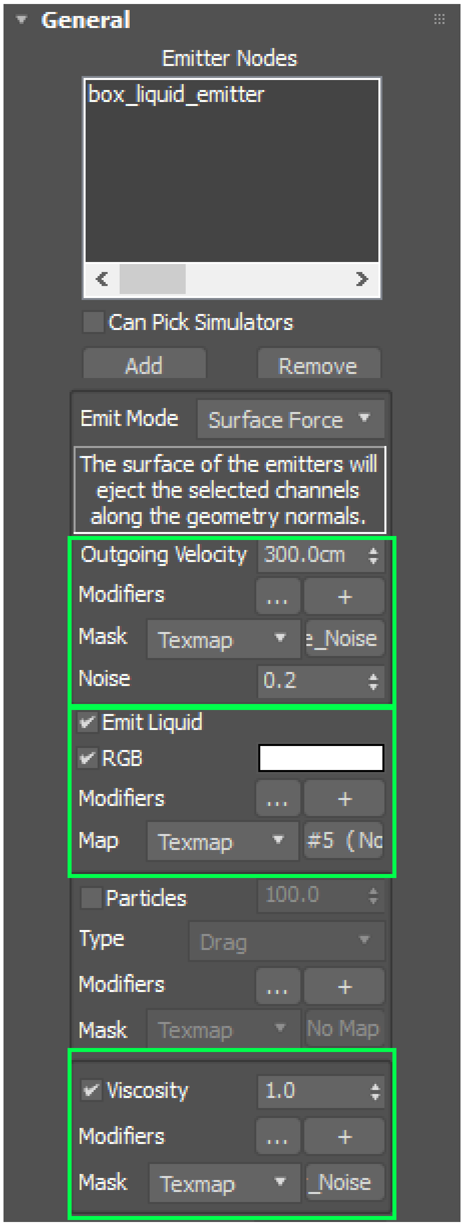
Increasing the **Scene Scale** made the **fluid** move **more slowly**.

Reducing the **Cell Size** gave us **more detail** in the fluid.

In the **next step** we will **set up** the **RGB channel** for the **Lava shader**.

Click the **No Map** under **Phoenix Liquid Source RGB Map**.
Add a **Noise** texture from the pop-up menu.

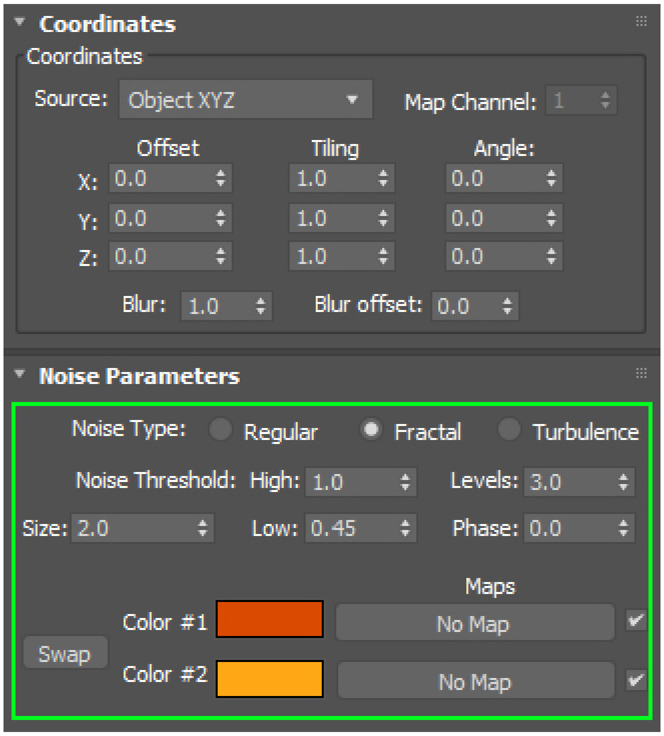
At this point you should have three different Noise textures, added in the Texmaps' slots for Outgoing Velocity, RGB and Viscosity.



Here are the exact settings for the RGB **Noise** texture:

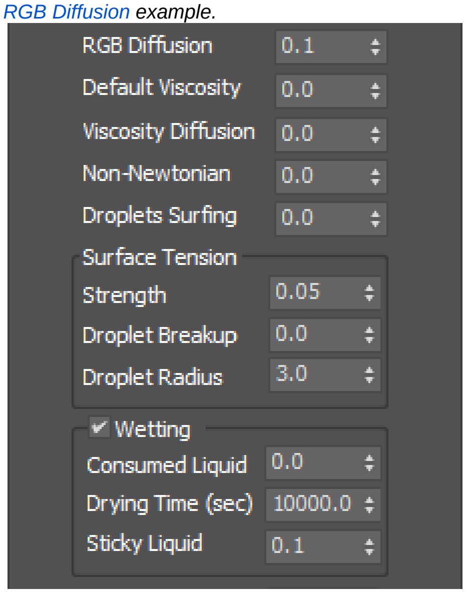
Type: Fractal
Size: 2
Threshold High: 1
Threshold Low: 0.45
Levels: 3
Phase: 0
Color 1: RGB [180, 13, 0]
Color 2: RGB [254, 102, 1]

Feel free to tweak the settings based on your own artistic judgement.



Set **Simulator Dynamics rollout RGB Diffusion** to **0.1**.

The RGB Diffusion parameter controls how quickly the colors of particles are mixed over time during the simulation. When set to 0, each liquid particle carries its own color, and the color of each individual particle does not change when liquids are mixed. This means that if red and green liquids are mixed, a dotted red-green liquid will be produced instead of a yellow liquid. This parameter allows the colors of particles to change when the particles are in contact, thus achieving uniform color in the resulting mixed liquid. *For more information, see the RGB Diffusion example.*



Camera & Light

Add a Command Panel **Cameras V-Ray VRayPhysicalCamera**.

The exact **position** of the **Camera** is **X/Y/Z: [2/318/108]**.

The exact **position** of the **Camera Target** is **X/Y/Z: [16/11 /44]**.

The **Film Gate** is set to **36.0**.

The **Focal Length** is set to **40.0**.

The Aperture **Film speed (ISO)** is set to **100**.

The Aperture **F-Number** is set to **0.8**.

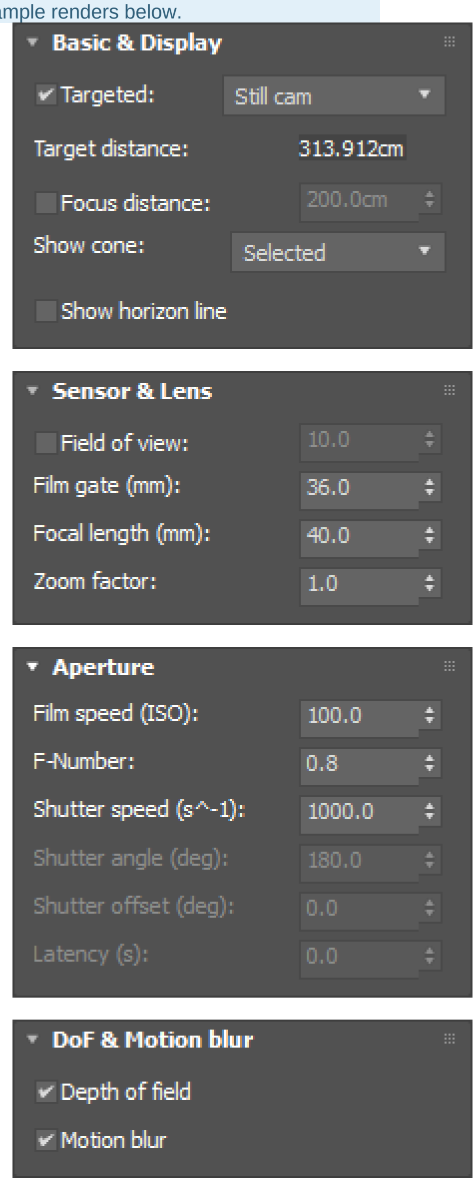
The Aperture **Shutter Speed** is set to **1000**.

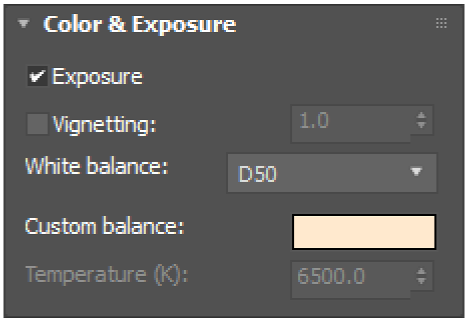
Both **Depth of Field** and **Motion Blur** are **Enabled**.

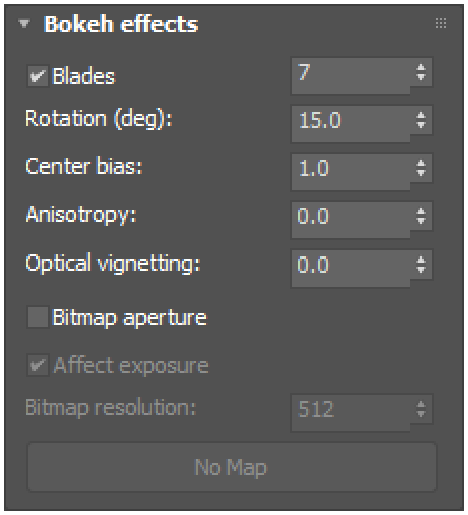
Color & Exposure **White Balance** is set to **D50**.

Bokeh Effects **Blades** is **Enabled**, with a value of **7**. **Rotation** is set to **15**, with a **Center Bias** of **1**.

The V-Ray Settings Rendering Output Width/Height is set to 960/540 for the example renders below.





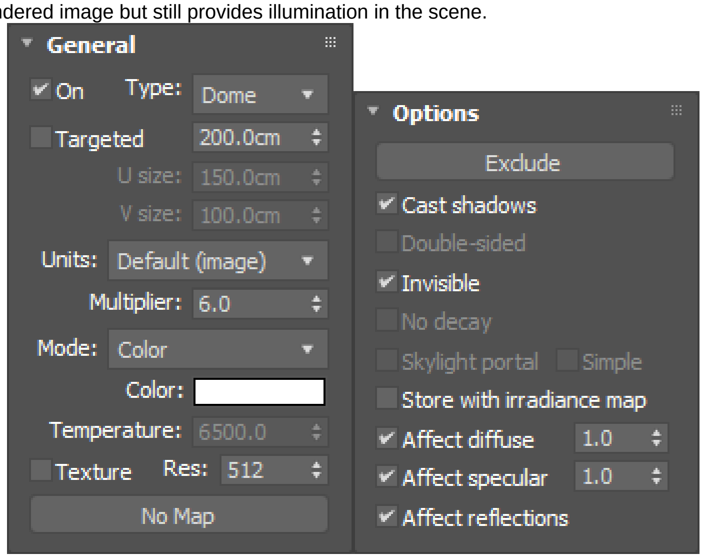


Add a Command Panel **Lights** **V-Ray** **V-Ray Light**.

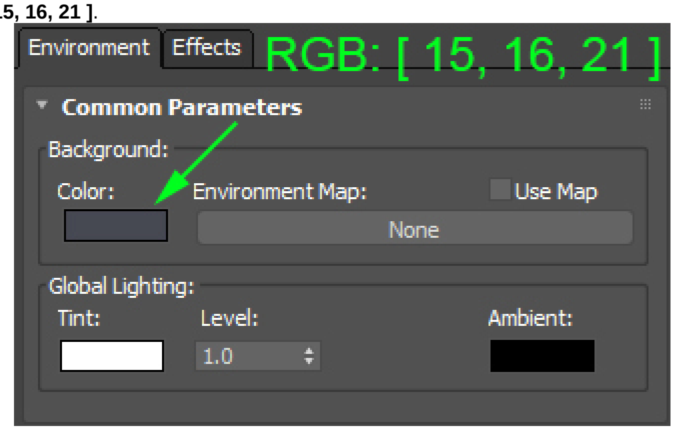
Set the Light **Type** to **Dome**.

Set the **Multiplier** to **6**.

Enable **Options** **Invisible** so the Dome Light is not visible in the rendered image but still provides illumination in the scene.

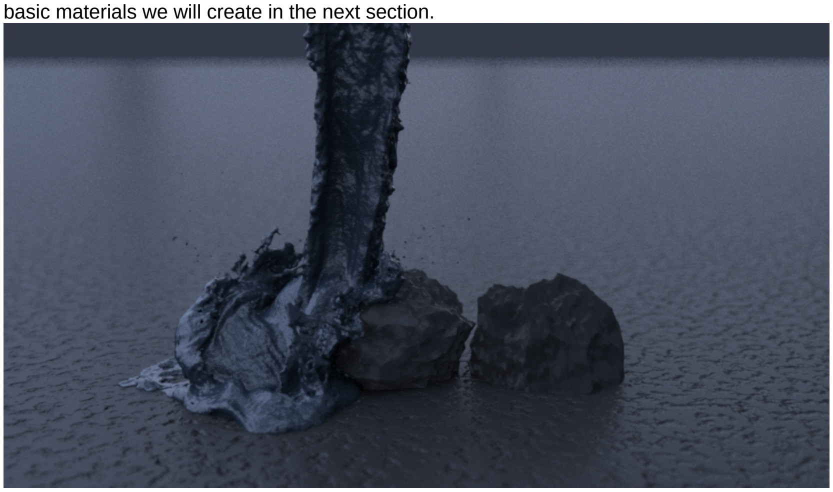


Set the **Rendering Environment Background Color** to **RGB [15, 16, 21]**.



Add a **V-Ray Infinite Plane** by going to **Geometry Standard Primitives V-Ray Infinite Plane**.

To the right is a rendered image of the current setup plus the basic materials we will create in the next section.



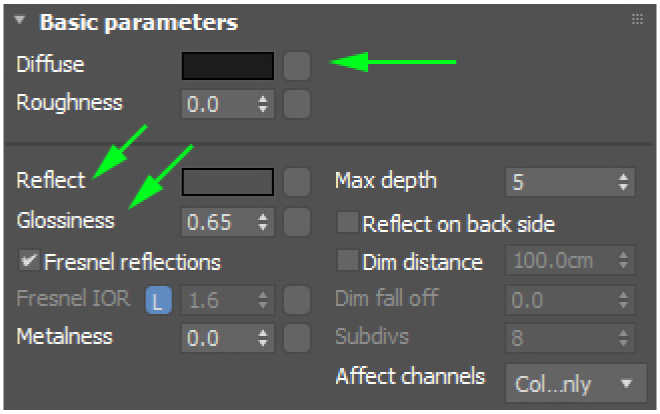
Materials for the Rocks and Ground

Assign a **V-Ray Material** to the **V-Ray Plane** and rename it to *mtl_Ground*.

Set the **Diffuse** to **RGB [2, 2, 2]**.

Set the **Reflect** to **RGB [21, 21, 21]**.

Set the **Reflection Glossiness** to **0.65**.



Add a new **Noise** texture and plug it into the **Bump** Map slot of *mtl_Ground*.

Source: Explicit Map Channel

Tiling: 10

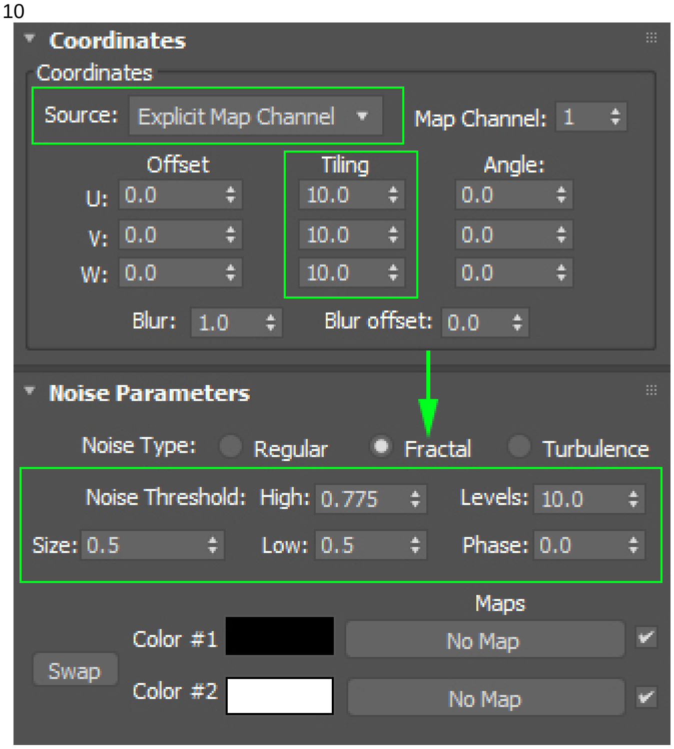
Noise Type: Fractal

Threshold High: 0.775

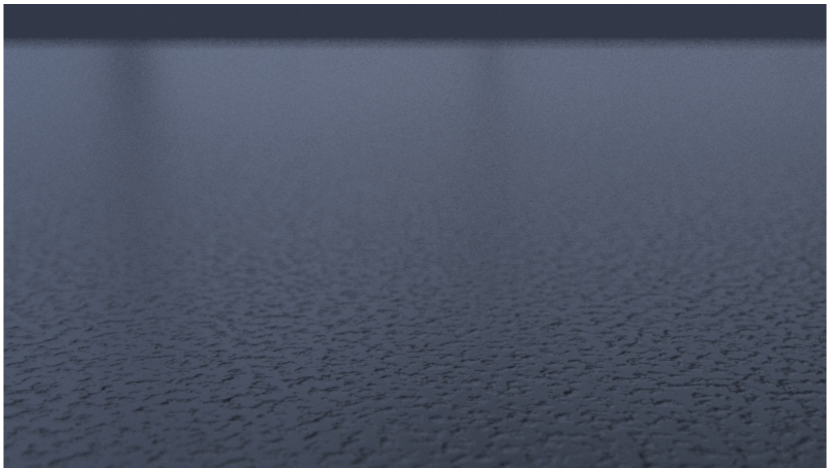
Threshold Low: 0.5

Size: 0.5

Levels: 10



Here's a rendered image of the V-Ray ground plane with mtl_Ground applied.

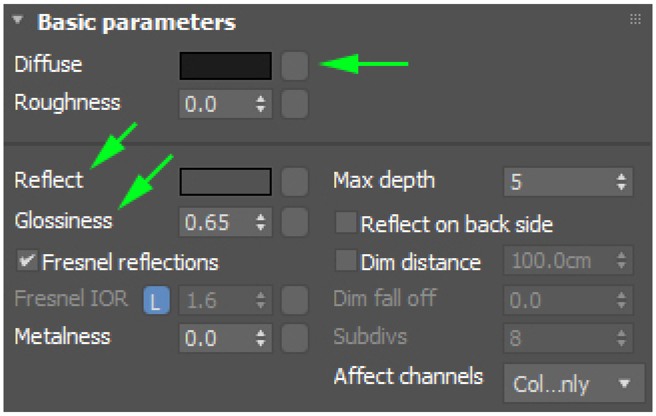


For the rocks, assign a **V-Ray Material** and rename it to *mtl_ Rocks*.

Set the **Diffuse** to **RGB [2, 2, 2]**.

Set the **Reflect** to **RGB [21, 21, 21]**.

Set the **Reflection Glossiness** to **0.65**.



Add a new **Noise** texture and plug it into the **Bump** Map slot of *mtl_Rocks*.

Source: Explicit Map Channel

Tiling: 1

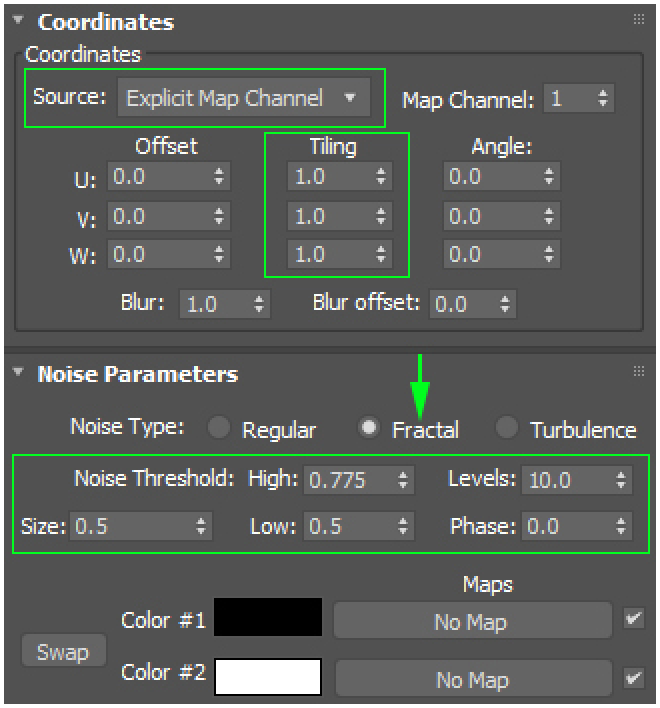
Noise Type: Fractal

Threshold High: 0.775

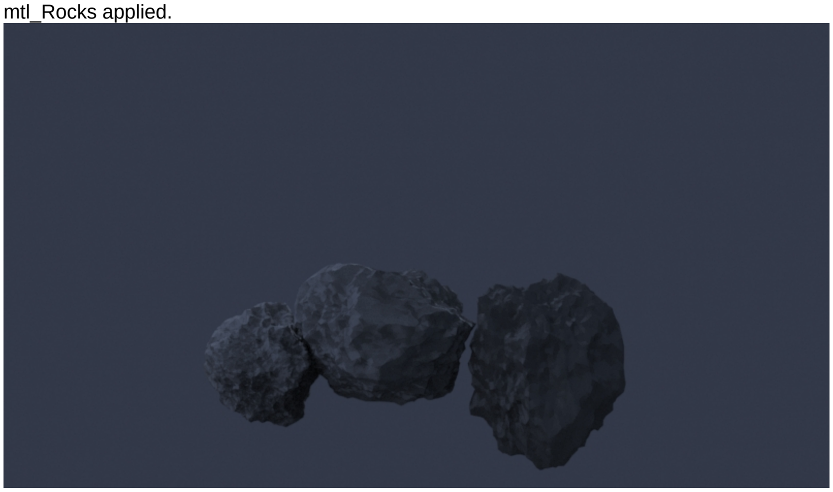
Threshold Low: 0.5

Size: 0.5

Levels: 10



Here's a rendered image of the Rocks geometry with mtl_Rocks applied.



Material for the Solid Lava

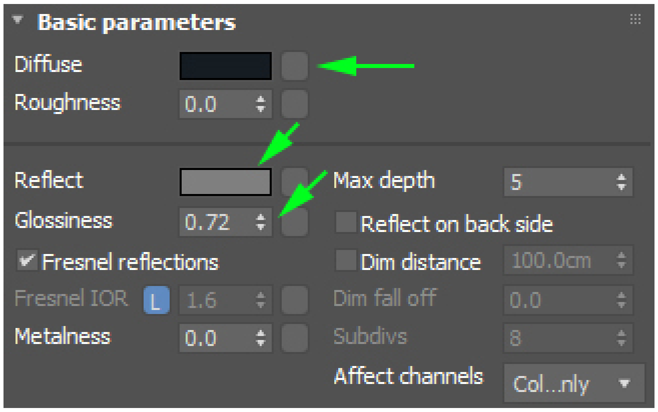
For the solid lava (the one with high **viscosity**), assign a **V-Ray Material** to the **Phoenix Simulator** and rename it to *cold SolidLava*. This material is the base for the final Complex Lava material we create in the next section.

Set the **Diffuse** to **RGB [1, 2, 3]**.

Set the **Reflect** to **RGB [55, 55, 55]**.

Set the **Reflection Glossiness** to **0.72**.

Reduce the **Bump** to **15** under the **Maps** rollout of the **V-Ray Material**.



Add a new **Noise** texture and plug it into the **Bump** Map slot of the *coldSolidLava* material.

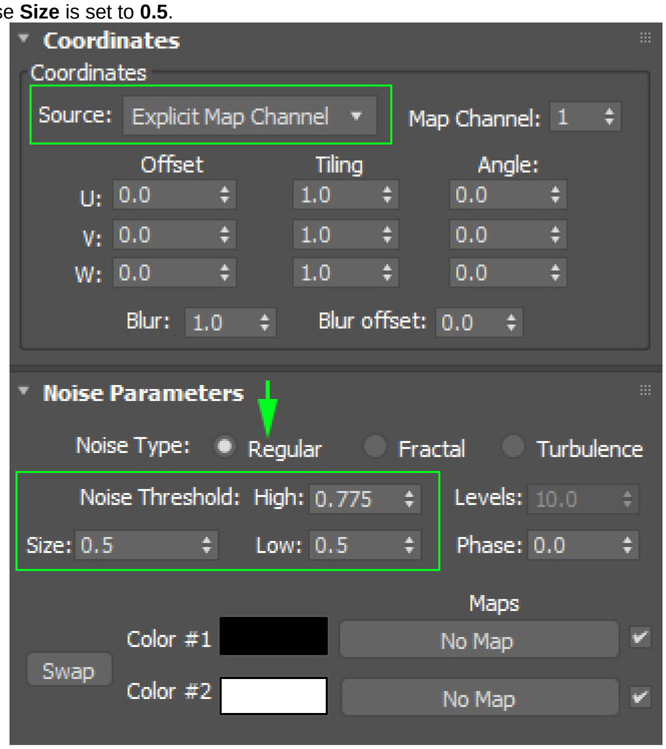
The **Source** is set to **Explicit Map Channel**.

The **Noise Type** is set to **Regular**.

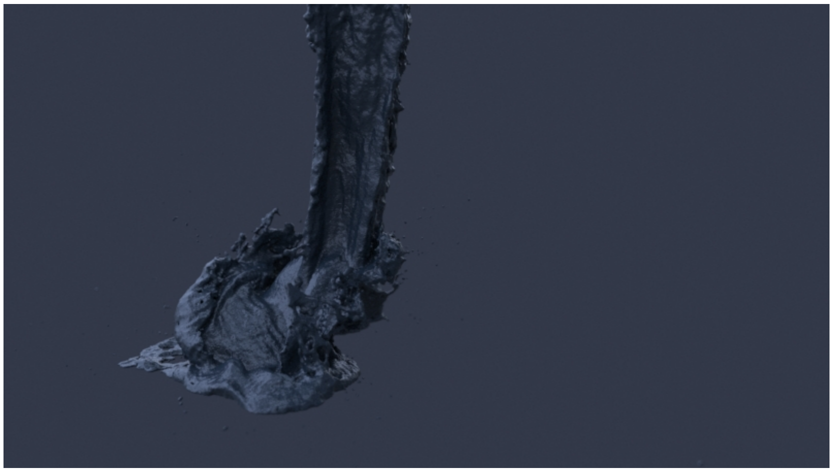
Threshold High is set to **0.775**.

Threshold Low is set to **0.5**.

The Noise **Size** is set to **0.5**.



Here's a rendered image of the Lava with the coldSolidLava material applied to the Simulator.



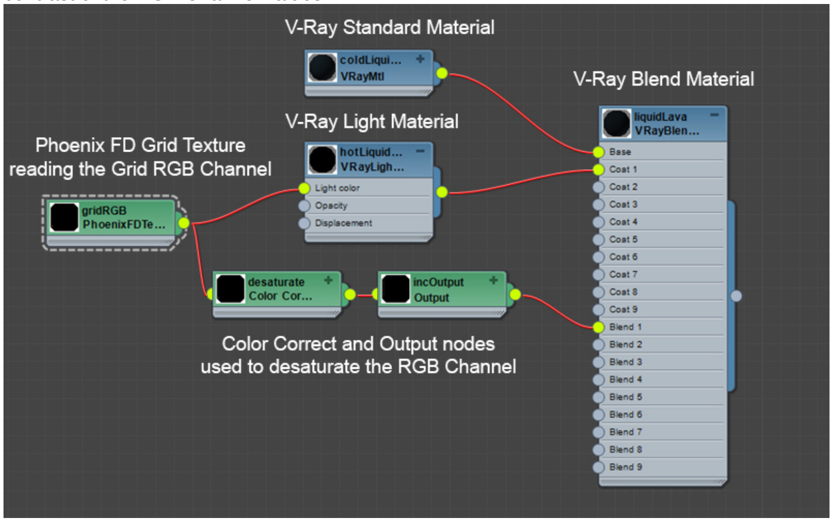
Material for the Liquid Lava

The material for the liquid lava consists of a regular V-Ray Material (*coldLiquidLava*) and a V-Ray Light Material (*hotLiquidLava*) blended together based on the output of a **Phoenix FD Grid Texture** reading the simulation's **RGB Channel**.

The *hotLiquidLava* **Light Material** is the core of the lava shader - the emitted light's color is based on the RGB Channel read through a Grid Texture.

The *coldLiquidLava* **V-Ray Material** is added into the mix for additional variation. Even very hot lava flows in real life has chunks of rocks that do not emit light. Having only a single Light Material would look unrealistic.

The **Blend** input of the *liquidLava* **Blend Material** is driven by the same RGB Channel - a **Color Correct** and **Output** nodes are used in-between to desaturate and increase the contrast of the RGB Channel values.



To start off, create a **V-Ray Blend Material** and **assign** it to the **Phoenix Simulator**. Rename it to *liquidLava*.

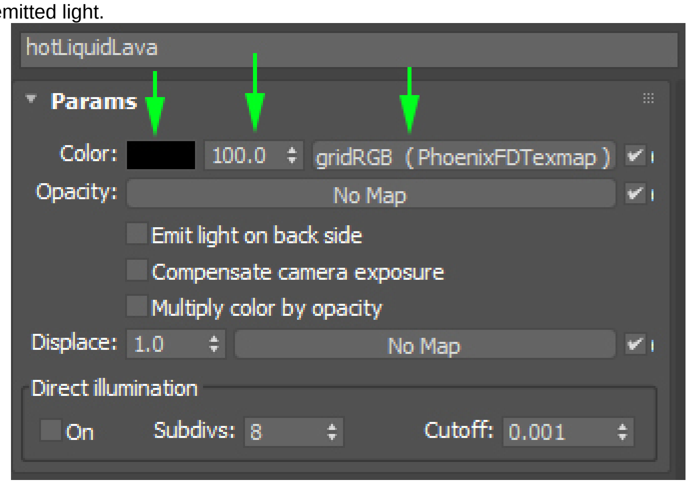
Create a **V-Ray Material** and plug it into the **Base** input of the *liquidLava* **Blend Material**. Rename it to *coldLiquidLava*.

Set the **Diffuse Color** to **RGB [1, 2, 3]**.

Create a **V-Ray Light Material** and plug it into the **Coat 1** input of the *liquidLava* **Blend Material**. Rename it to *hotLiquid Lava*.

Set the Color to **RGB [0, 0, 0]** - we use a Grid Texture reading the simulation RGB Channel to drive the color of the emitted light.

Set the **Color Multiplier** to **100** - this will affect the intensity of the emitted light.



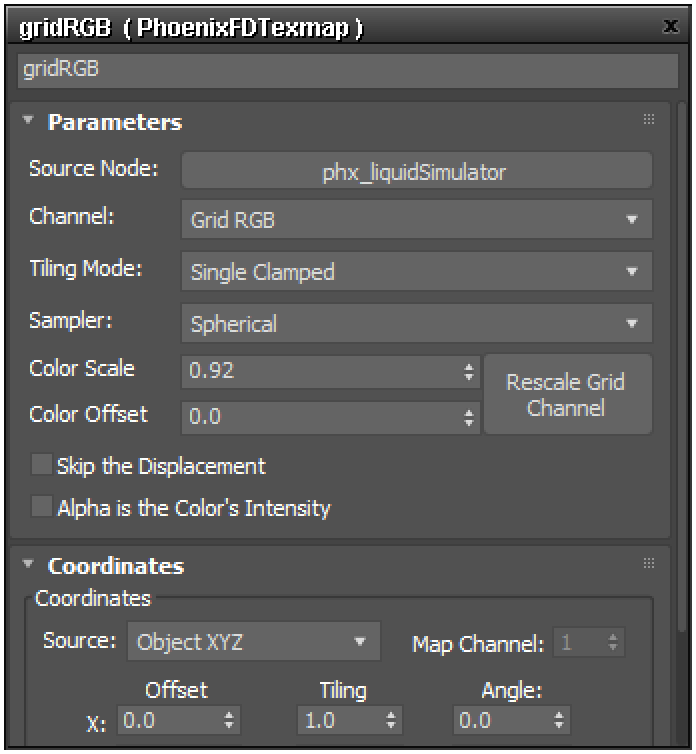
Create a **Phoenix FD Grid Texture** and plug it into the *hotLiq uidLava* Light Material's **Light Color input**. Rename it to *grid RGB*.

Click the **Pick a Phoenix Simulator** button next to the **Sourc e Node** parameter and select your **Phoenix Simulator** from the pop-up menu. The name of the button should change according to the name of the simulator once you hit the OK button.

Set the **Channel** to **Grid RGB** - this is the channel the texture will read from the cache files.

Set the **Sampler** to **Spherical**. You can think of the sampling process as Anti-Aliasing - the Box sampler will give you a rough texture, the Linear will try to smooth the colors and the Spherical will produce the smoothest result. Note that Spherical is 20-30% slower than Linear so make sure to check if the additional sampling at the expense of render time is worth with your setup.

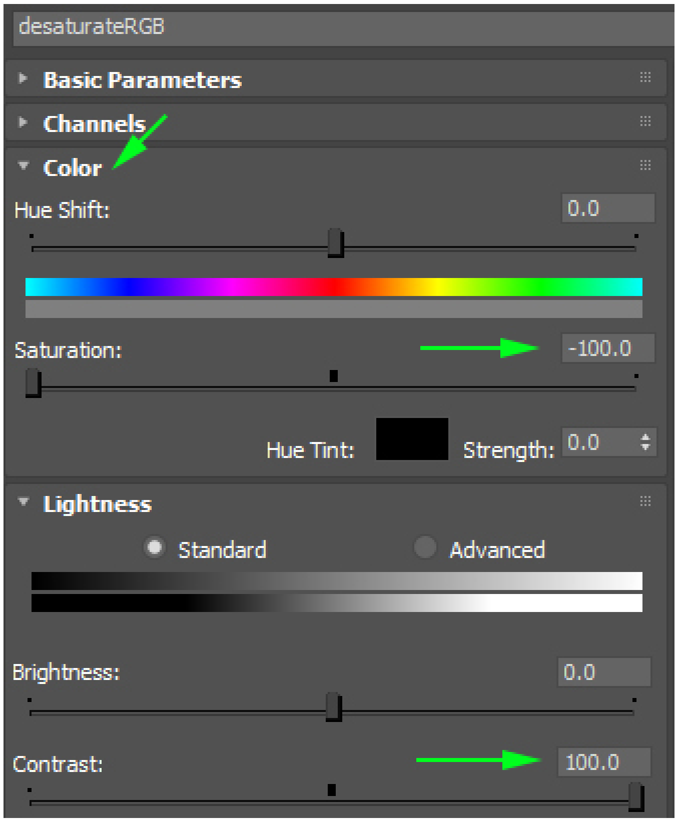
Set the **Color Scale** to **0.92**. This value multiplies the color output values of the Grid Texture. In our case we use it to achieve more complex shader of the lava.



Create a **Color Correction** node and plug the *gridRGB* **texture** into the **Map** slot. Rename the Color Correction node to *desaturateRGB*.

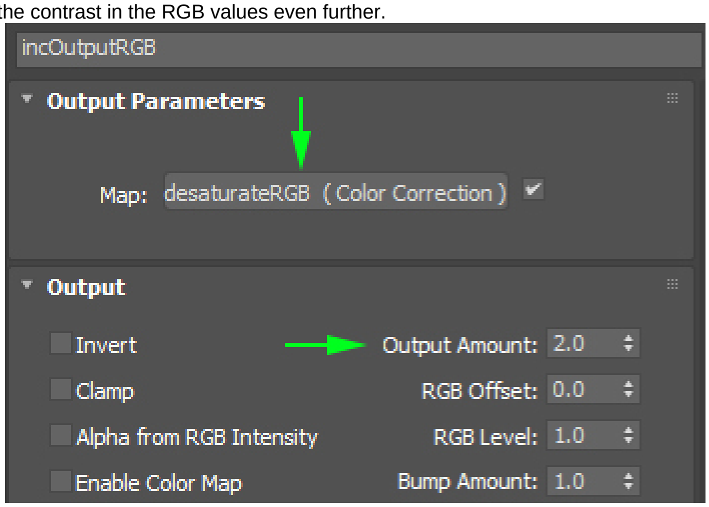
Set the Color **Saturation** to **-100**.

Set the Lightness **Contrast** to **100**.

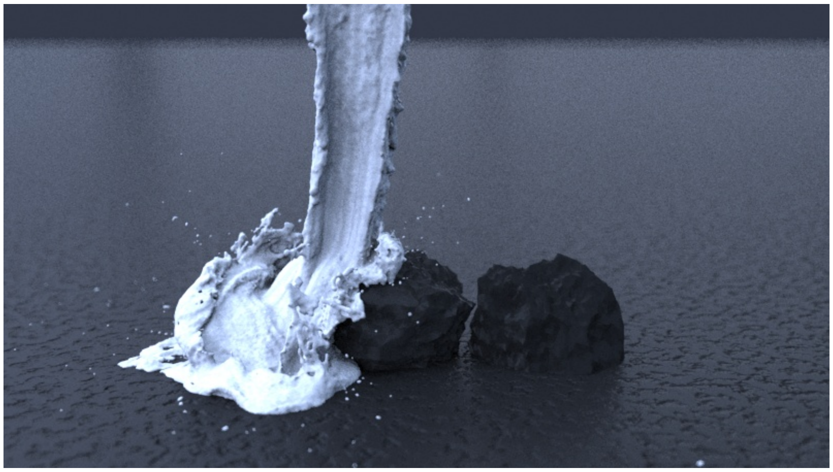


Create an **Output** node and plug the **Color Correction** texture into the **Map** slot. Rename the **Output** node to *incOutputRGB*.

Set Output **Output Amount** to **2.0**. Doing so will essentially boost the contrast in the RGB values even further.

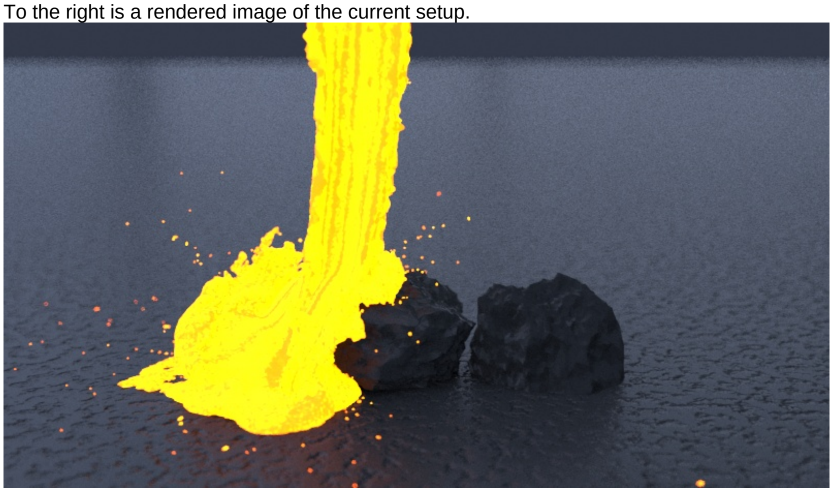


To the right is a rendered image of the incOutputRGB Output texture applied to the Diffuse Color of a default V-Ray Material.



Plug the *incOutputRGB* **Output texture** into the **Blend 1** slot of the *liquidLava* **Blend Material**.

To the right is a rendered image of the current setup.

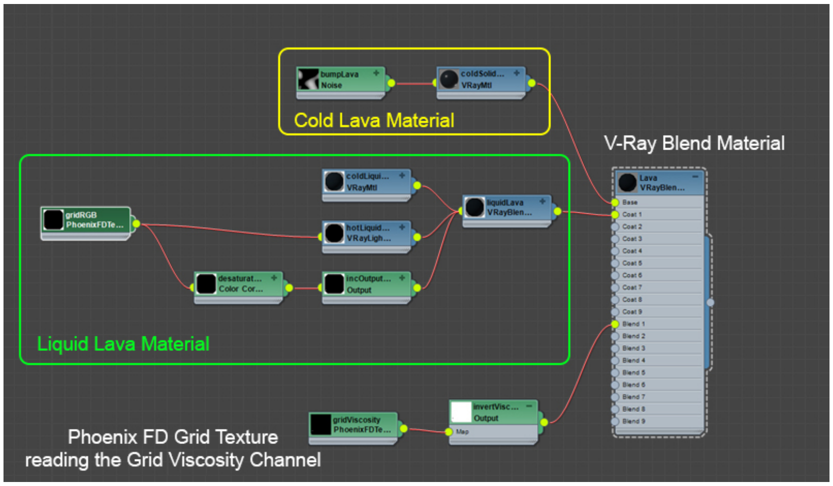


Complete the Lava Shader

The final material for the lava consists of the *coldSolidLava* and *liquidLava* materials we prepared in the previous two sections of this tutorial.

A V-Ray Blend Material is used at the final wrapper. The **Blen d Material** is driven by a **Phoenix FD Grid Texture** reading the **Viscosity Grid Channel**. Recall that the Viscosity in the simulation increases as the simulation progresses in time. Thus, higher viscosity values should correspond to solid lava with the *coldSolidLava* material applied.

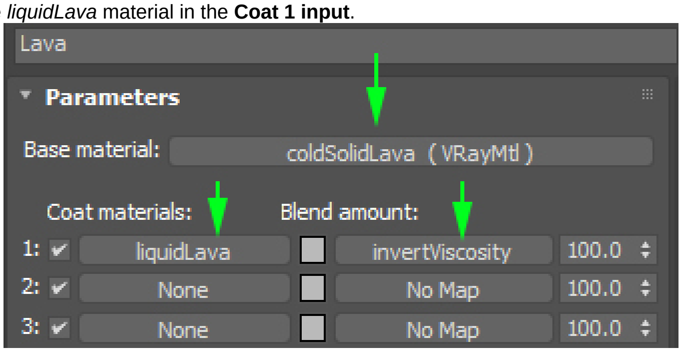
The purpose of the **Output** node is to **invert** the values coming in from the **Grid Texture**. Note that the Coat 1 input of the Blend material is the liquidLava material. The Coat 1 is applied where the Blend texture is white. Therefore, if the Grid Texture output is not inverted, the high-viscosity areas of the simulation will receive the liquidLava material which would be incorrect.



Create a **V-Ray Blend Material** and **rename** it to *Lava*.

Plug the *coldSolidLava* material in the **Base input**.

Plug the *liquidLava* material in the **Coat 1 input**.

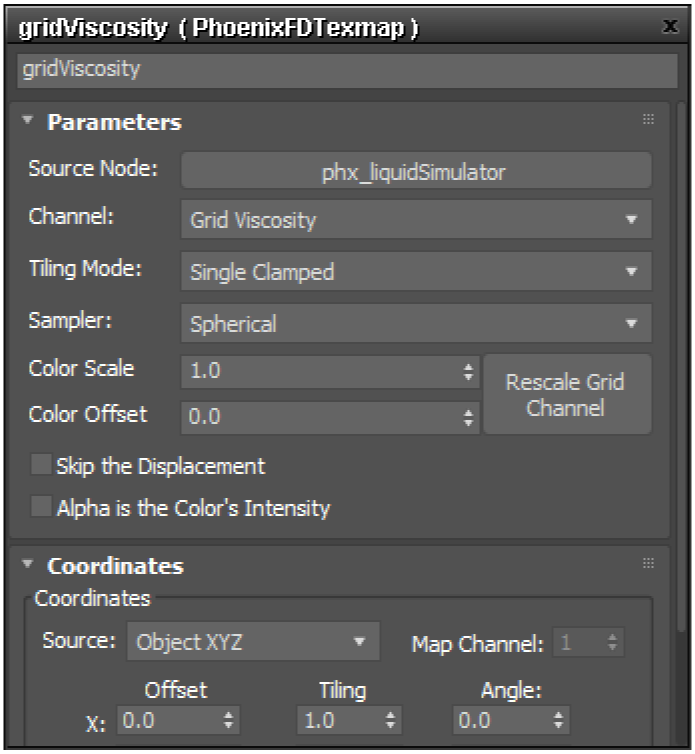


Create a **Phoenix FD Grid Texture** and rename it to *gridVisc osity*.

Click the **Pick a Phoenix Simulator** button next to the **Sourc e Node** parameter and select your **Phoenix Simulator** from the pop-up menu. The name of the button should change according to the name of the simulator once you hit the OK button.

Set the **Channel** to **Grid Viscosity** - this is the channel the texture will read from the cache files.

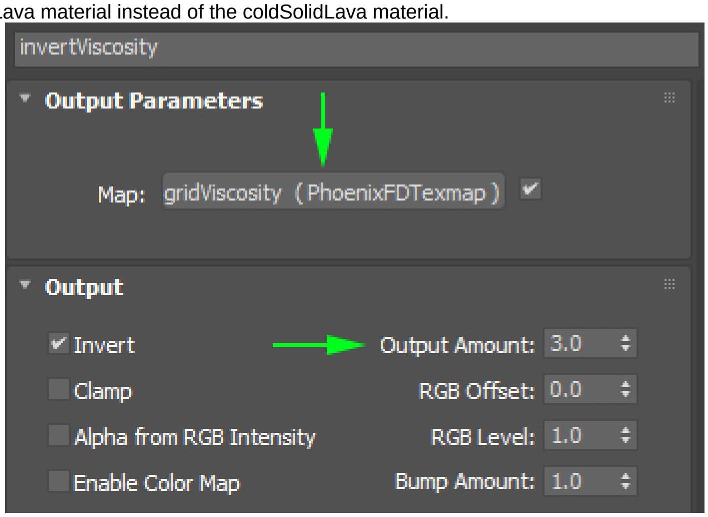
Set the **Sampler** to **Spherical**. You can think of the sampling process as Anti-Aliasing - the Box sampler will give you a rough texture, the Linear will try to smooth the colors and the Spherical will produce the smoothest result. Note that Spherical is 20-30% slower than Linear so make sure to check if the additional sampling at the expense of render time is worth with your setup.



Create an **Output** texture and plug the *gridViscosity* **Grid Texture**.

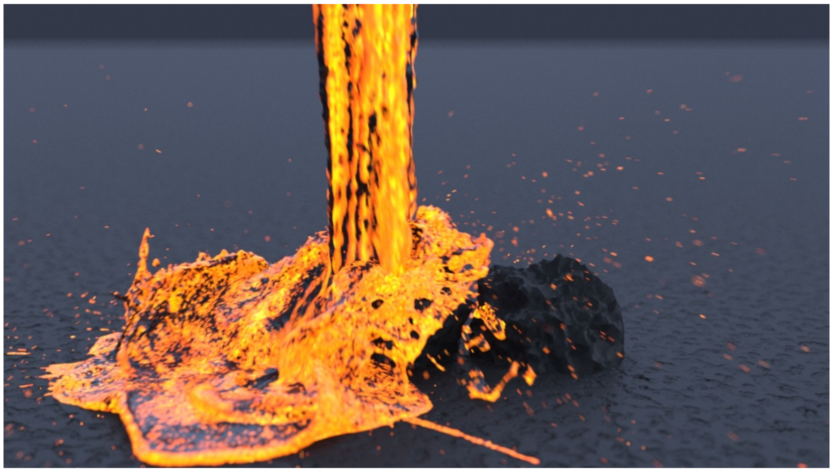
Set the **Output Amount** to **3**. Doing so will essentially boost the contrast in the Viscosity values.

Enable **Invert**. If the Grid Texture output is not inverted, the high-viscosity areas of the simulation will receive the liquidLava material instead of the coldSolidLava material.



Plug the **Output texture** into the **Blend 1** slot of the *Lava* **Blend Material**.

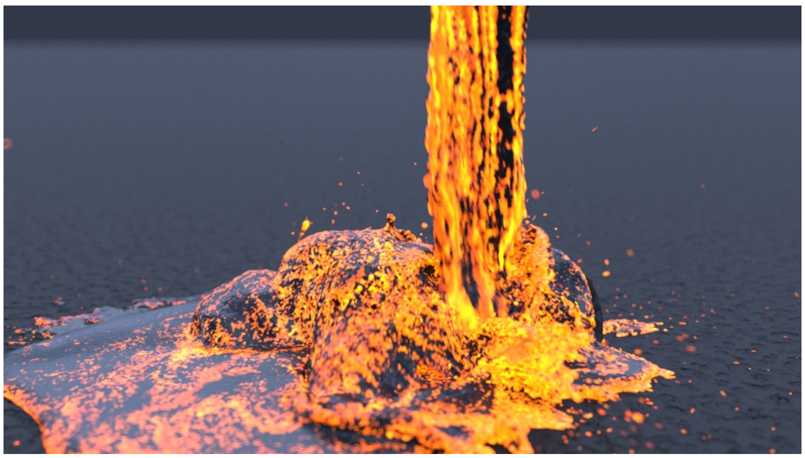
To the right is a rendered image of the final Lava material.



Mesh Smoothing for the Lava Mesh

Optionally, you may choose to enable some **Simulator Rendering Mesh Smoothing** options if your lava simulation looks too sharp/jagged.

The settings for the smoothed image to the right are **Simulator Rendering Use Liquid Particles enabled**, and a **Smoothness** of **5**.



Lava No Smoothing

Lava With Smoothing

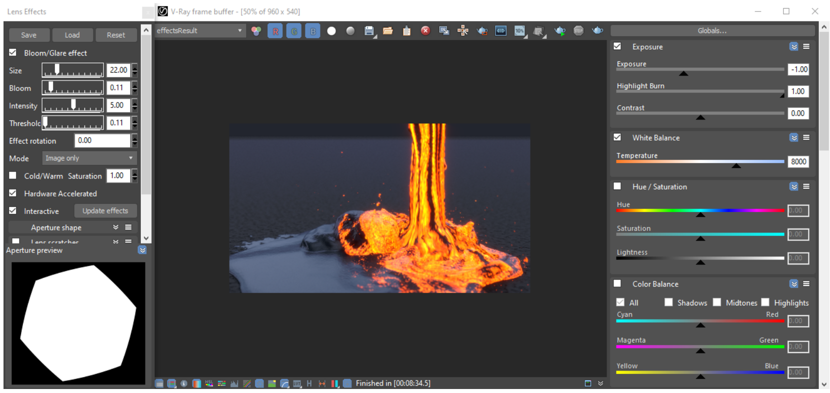
V-Ray Frame Buffer

For the final image, some corrections are made in the V-Ray Frame Buffer.

Bloom/Glare Effect is enabled from the **Lens Effects** panel. The **Size** is set to **22**, with a **Bloom** of **0.11**. The **Intensity** is set to **5.00** and the **Threshold** to **0.11**.

Exposure is enabled from the **Corrections** panel and reduced to **-1** to counter the brightening effect of the Bloom effect.

White Balance is also **enabled** and **Temperature** set to **8000** to pull the image away from the blue overall tint created by the Bloom effect.



Final Results