Lighting In-Depth QuickStart

This page provides a detailed QuickStart guide to using V-Ray lights in Nuke.

Overview

In this tutorial we are going take a look at the V-Ray lights in V-Ray for Nuke. In the Lights menu are are several options, which we explore in this tutorial:

VRayLightAmbient, which contributes a solid-colored light to scene
VRayLightDome, which creates a global illumination image based light
VRayLightIES uses common light IES files to describe light’s falloffs and intensities
VRayLightMesh turns any geometry into a light source including animated geometry
VRayLightRect, a simple light source and the most commonly used
VRayLightSpehere, a simple point light source

To follow this tutorial, you will need to have the V-Ray for Nuke plugin installed. This tutorial is a companion to go along with the QuickStart video posted on our YouTube channel.

Tutorial Assets

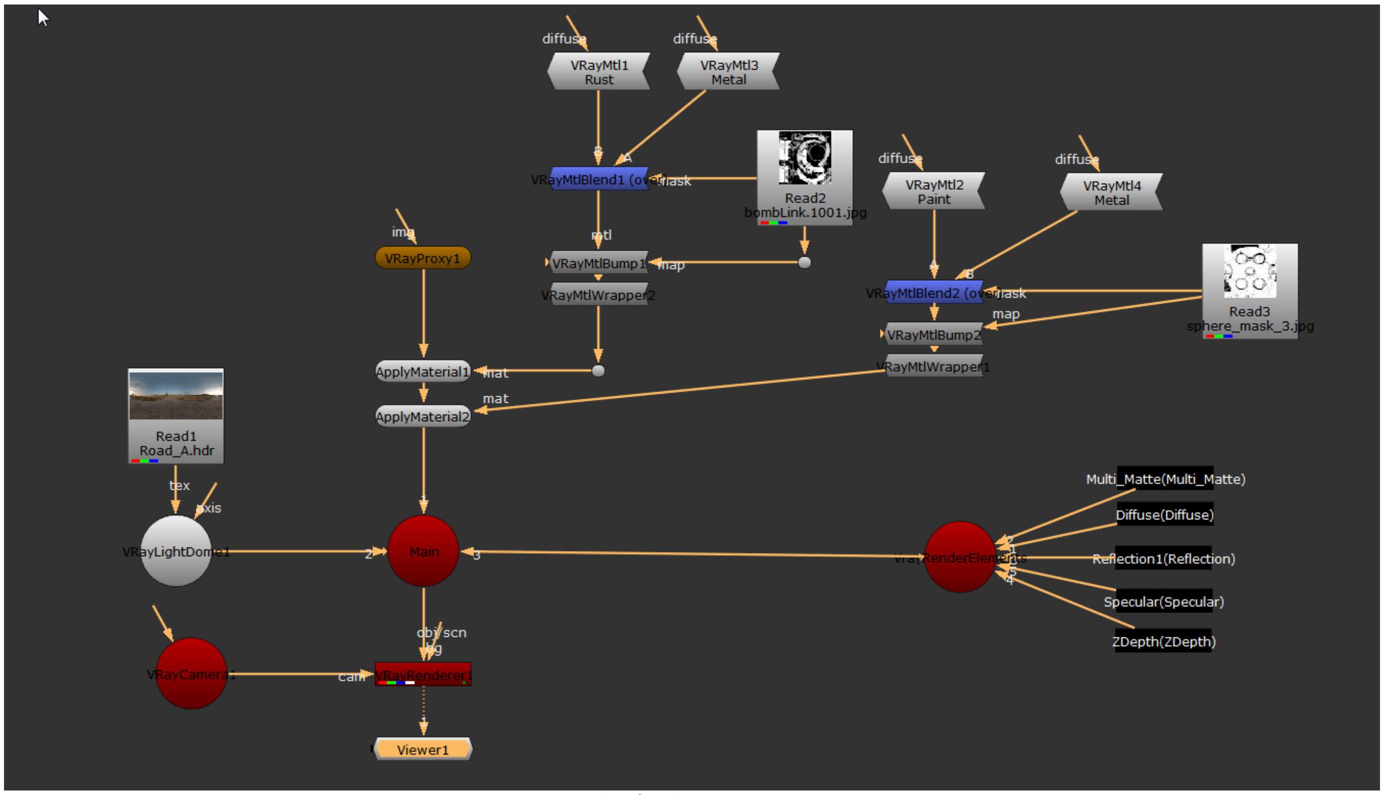
To download the files used in this tutorial, please click on the button below.

Download
Project Files

Due to licensing issues, the Road_A.hdr file used in the tutorial video is not included in the downloadable assets. Instead, please use one of the provided HDR images in its place.

Tutorial Steps

1) Adjusting the VRayLightDome

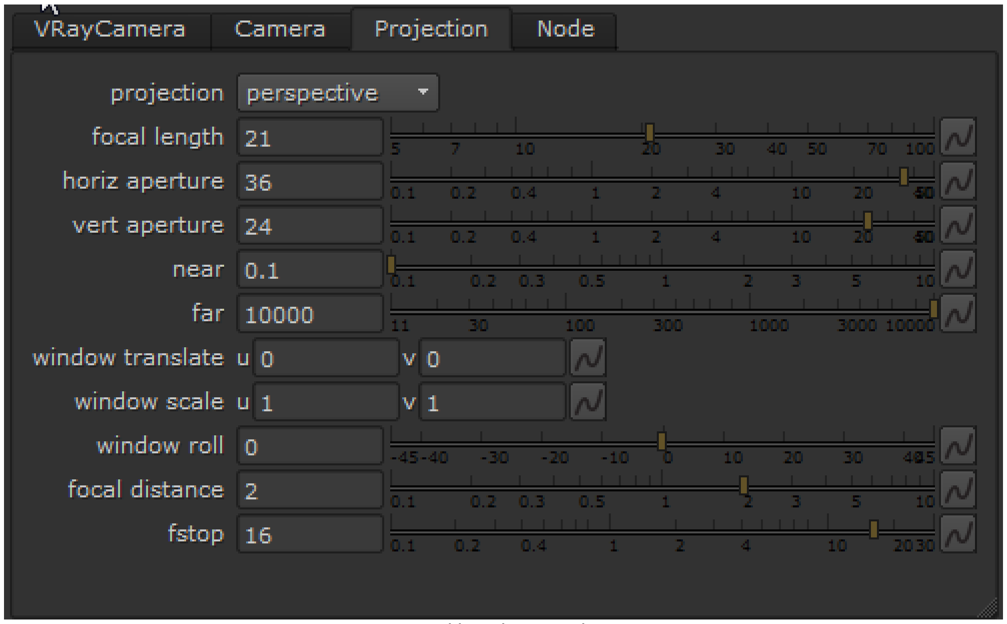


Starting Scene Layout

Open the finished scene from the tutorial **QuickStart 3** on texture mapping. Change the **Focal Length** of the **VRayCamera** to *21* so you can see more of the scene.

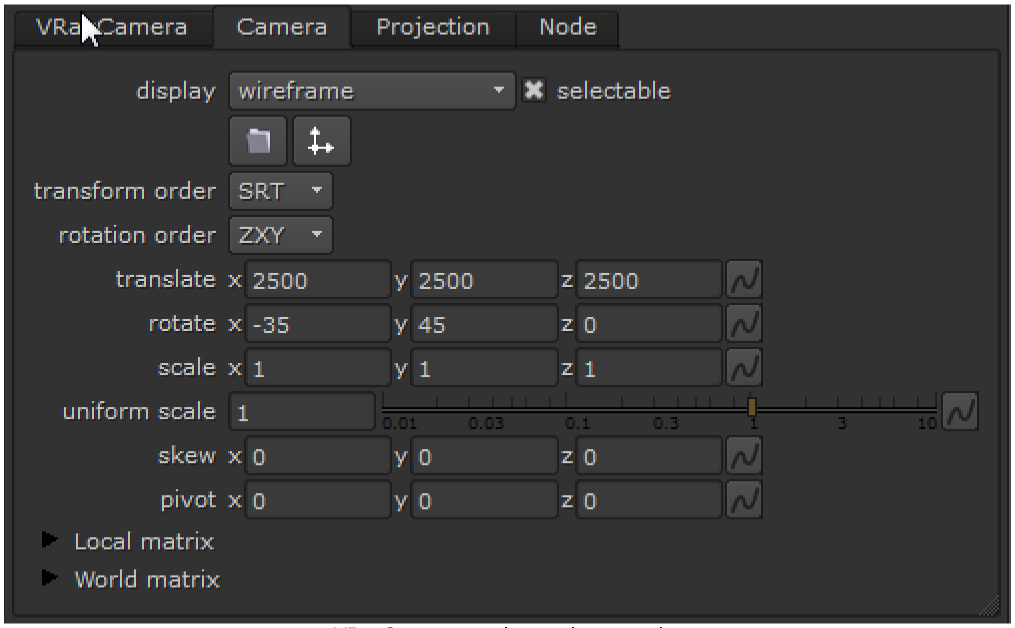
The scene is currently lit by a **VRayLightDome** with its properties set to use only the upper half of the hemisphere and its visibility set to **Invisible**. This gave us a darker lighting underneath.

Enable the **Dome Spherical checkbox** to make the **VRayLightDome** a sphere. You now have a full sphere with the HDR texture lighting the geometry and filling in these areas with light. Disable the **Invisible** checkbox so the HDR is visible as a texture on the **VRayLightDome**. The HDR is now both above and below the geometry, and wraps around it as a full sphere.


VRayCamera | Camera | Projection | Node

projection: perspective
focal length: 21
horiz aperture: 36
vert aperture: 24
near: 0.1
far: 10000
window translate: u 0, v 0
window scale: u 1, v 1
window roll: 0
focal distance: 2
fstop: 16

Focal length properties

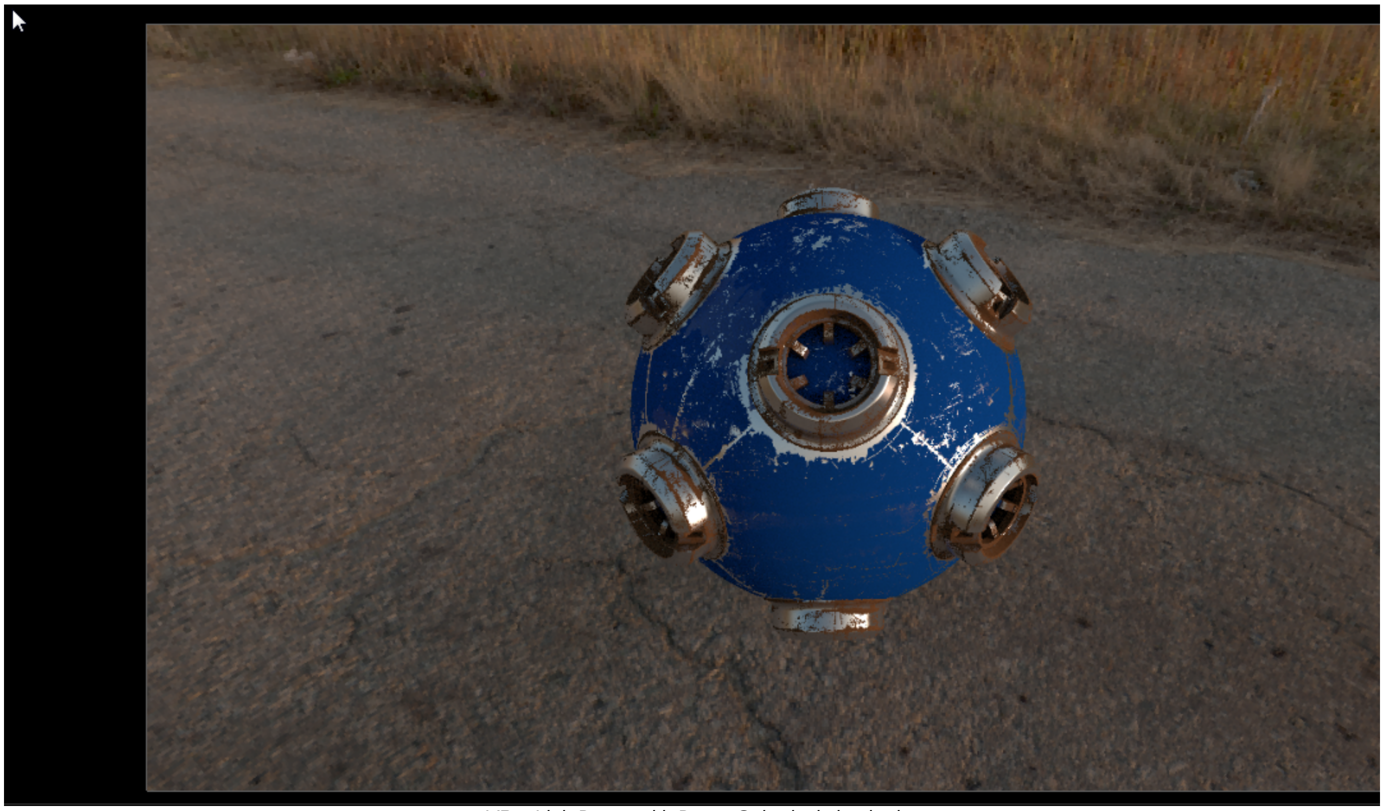

VRayCamera | Camera | Projection | Node

display: wireframe, selectable
transform order: SRT
rotation order: ZXY
translate: x 2500, y 2500, z 2500
rotate: x -35, y 45, z 0
scale: x 1, y 1, z 1
uniform scale: 1
skew: x 0, y 0, z 0
pivot: x 0, y 0, z 0
Local matrix
World matrix

eVRayCamera translate and rotate values

Change the camera angle by clicking the **VRayCamera** and changing the **x rotate value** under the Camera tab of the **VRayCamera properties** from *-35* to *-15*. Now you can see the horizon on the HDR. To see this in the render, disable the **Dome Spherical** checkbox so that the **VRayLightDome** is a hemisphere.

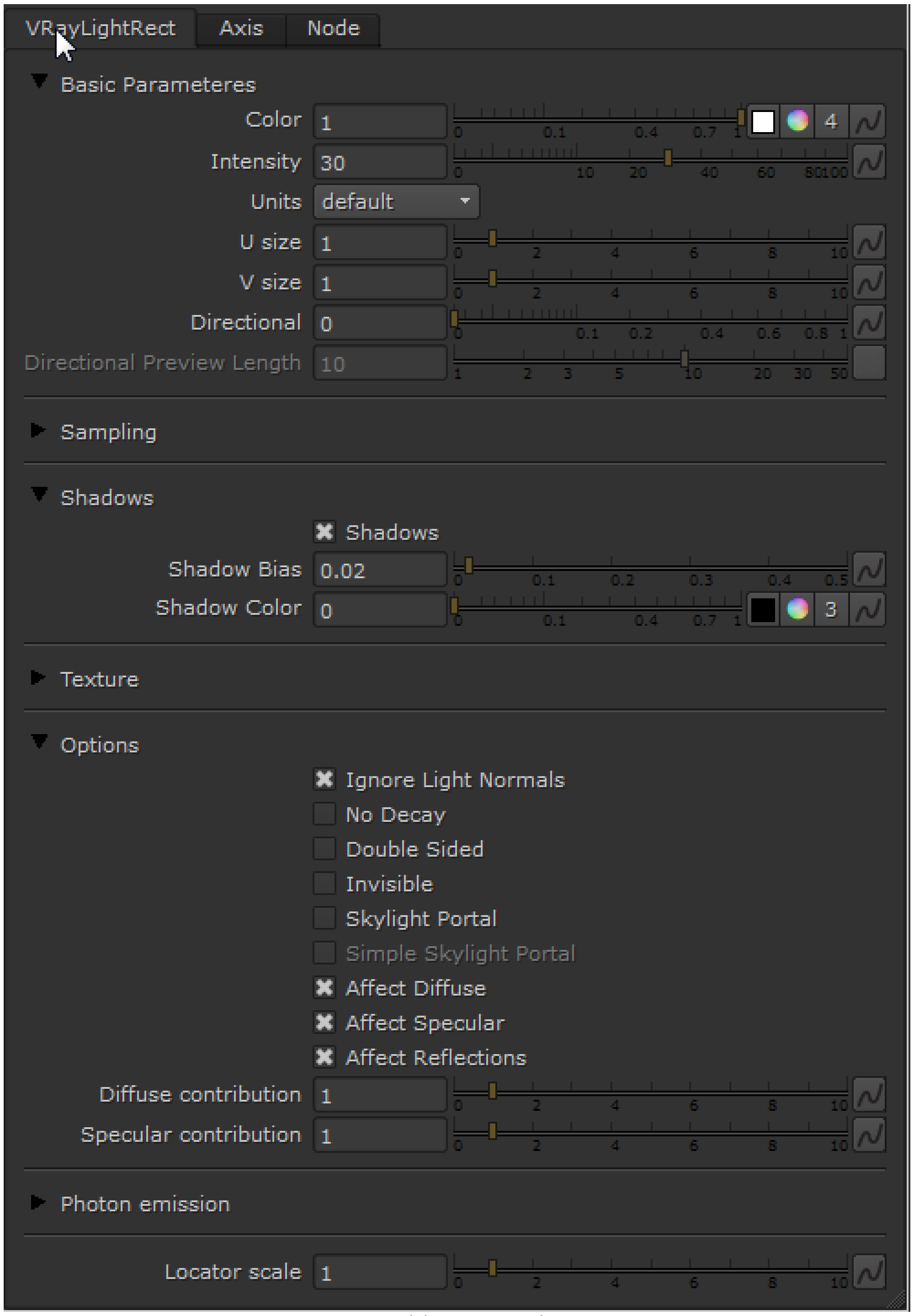
Enable the **Dome Spherical** checkbox again, and reposition the **x rotate** value back to *-35*. Disconnect this light from the scene for now and move it to one side so it doesn't get in the way.


VRayLightDome with Dome Spherical checked

2) Adding a VRayLightRect

Add a **VRayLightRect** to the Nuke script and create a new **scene** named **Lights**. Connect the **Lights** scene to the main scene and the **VRayLightRect** to the **Lights** scene so that the **VRayLightRect** is now the only light lighting the geometry. Switch to the 3D view and move this light up above the geometry. Enter *-90* for the **rotate x** value under the Axis tab.

You'll notice that the light isn't illuminating things at all at this stage. The brightness level can be changed by either increasing the light's **Intensity** value or by adjusting the scale for the **VRayLightRect**. In this case we have a very large scene, so it makes more sense to increase the **uniform scale** value under the Axis tab. Change the **uniform scale** value to *1000*. Now you should be able to see the **VRayLightRect** lighting the geometry from above.



VRayLightRect | Axis | Node

Basic Parameteres

Color	1
Intensity	30
Units	default
U size	1
V size	1
Directional	0
Directional Preview Length	10

Sampling

Shadows

✓ Shadows

Shadow Bias	0.02
Shadow Color	0

Texture

Options

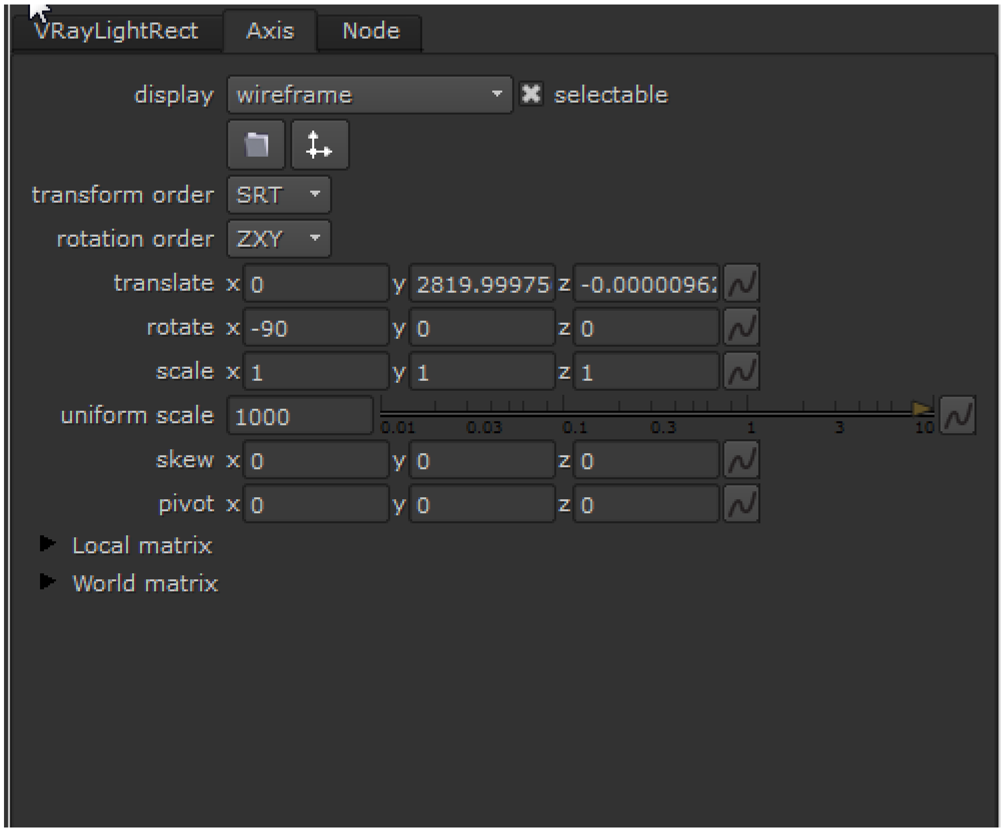
- ✓ Ignore Light Normals
- No Decay
- Double Sided
- Invisible
- Skylight Portal
- Simple Skylight Portal
- ✓ Affect Diffuse
- ✓ Affect Specular
- ✓ Affect Reflections

Diffuse contribution	1
Specular contribution	1

Photon emission

Locator scale	1

VRayLightRect properties


VRayLightRect | Axis | Node

display: wireframe ✖ selectable

transform order: SRT

rotation order: ZXY

translate x 0 y 2819.99975 z -0.0000096:

rotate x -90 y 0 z 0

scale x 1 y 1 z 1

uniform scale 1000

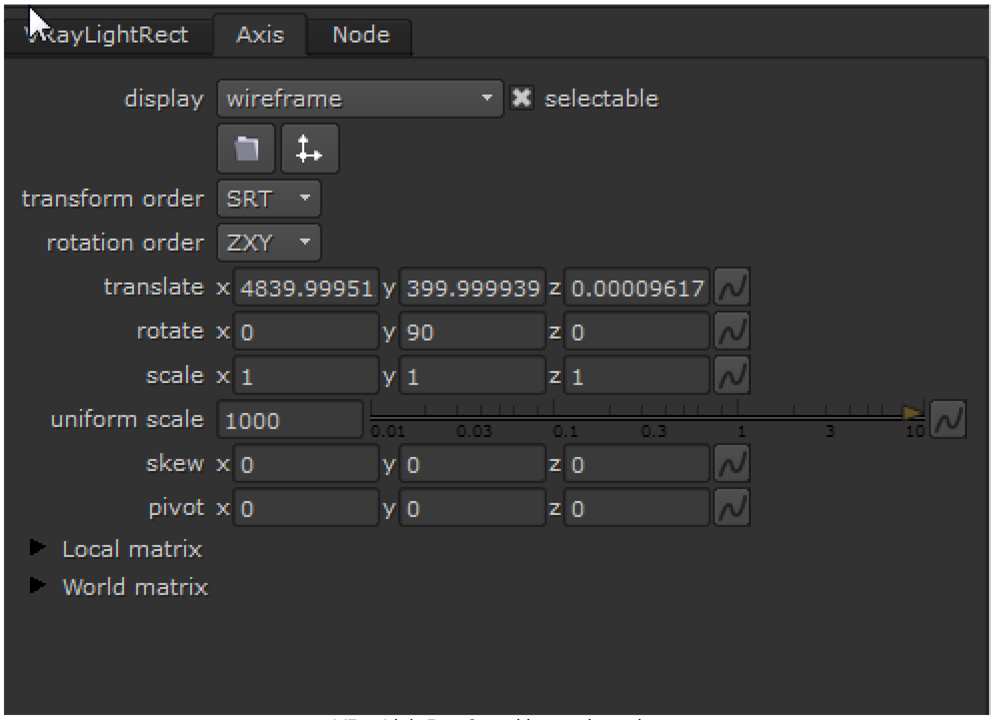
skew x 0 y 0 z 0

pivot x 0 y 0 z 0

▶ Local matrix

▶ World matrix

VRayLightRect position and rotate values


VRayLightRect | Axis | Node

display: wireframe ✖ selectable

transform order: SRT

rotation order: ZXY

translate x 4839.99951 y 399.999939 z 0.00009617

rotate x 0 y 90 z 0

scale x 1 y 1 z 1

uniform scale 1000

skew x 0 y 0 z 0

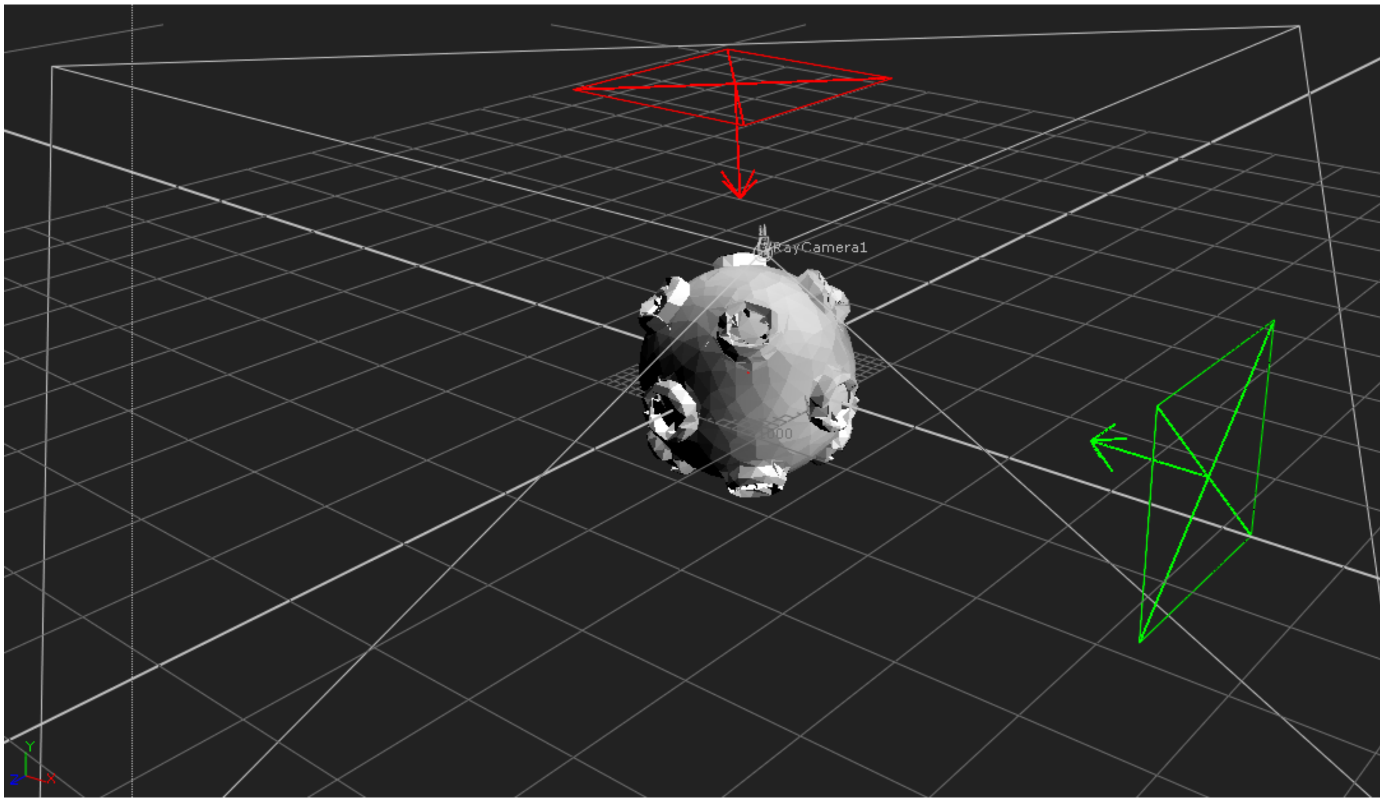
pivot x 0 y 0 z 0

▶ Local matrix

▶ World matrix

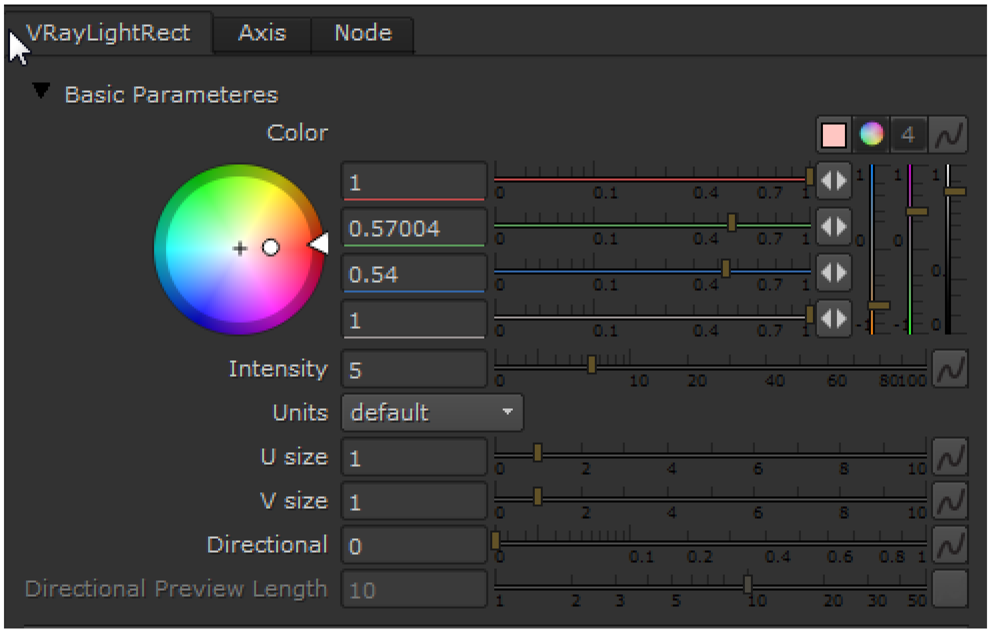
VRayLightRect2 position and rotation

Duplicate the **VRayLightRect** and connect this new light into the **Lights** scene. Right now this new light has exactly the same position and intensity as the version you copied it from. You are going to change this so that the new light illuminates the geometry from the side. First, press **Pause** on the interface to stop Nuke from updating the render in the viewer while you work on the **VRayLightRect** settings. Grab the new **VRayLightRect**, move it to the right in the 3D view, and pull it downwards in the Z axis so it’s level with the geometry as shown:



VRayLightRect2 position

Under the Axis tab, change the **rotate** values to *0*, *-90*, and *0* for **x, y,** and **z** respectively. To make this light less intense and not as strong as the first **VRay LightRect**, change its **Intensity** value to *5.* Let's change the color produced by this light by clicking on the color wheel button and changing the color to a warm red color. It's a good idea to disable the other **VRayLightRect** (by hovering over it and pressing the **D** key) so we can see exactly how the second light is affecting the scene while we make any tweaks or changes to its color. We are aiming for a warm color as shown, without it being too saturated.


VRayLightRect | Axis | Node

Basic Parameteres

Color: 1, 0.57004, 0.54, 1

Intensity: 5

Units: default

U size: 1

V size: 1

Directional: 0

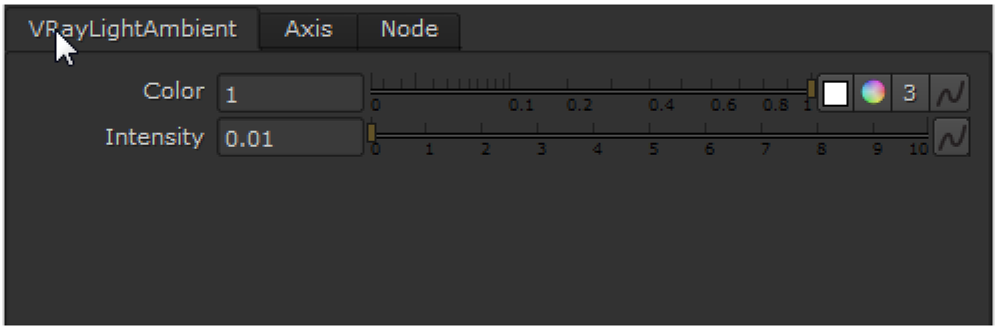
Directional Preview Length: 10

VRayLightRect2 light color settings

3) Adding a VRayLightAmbient

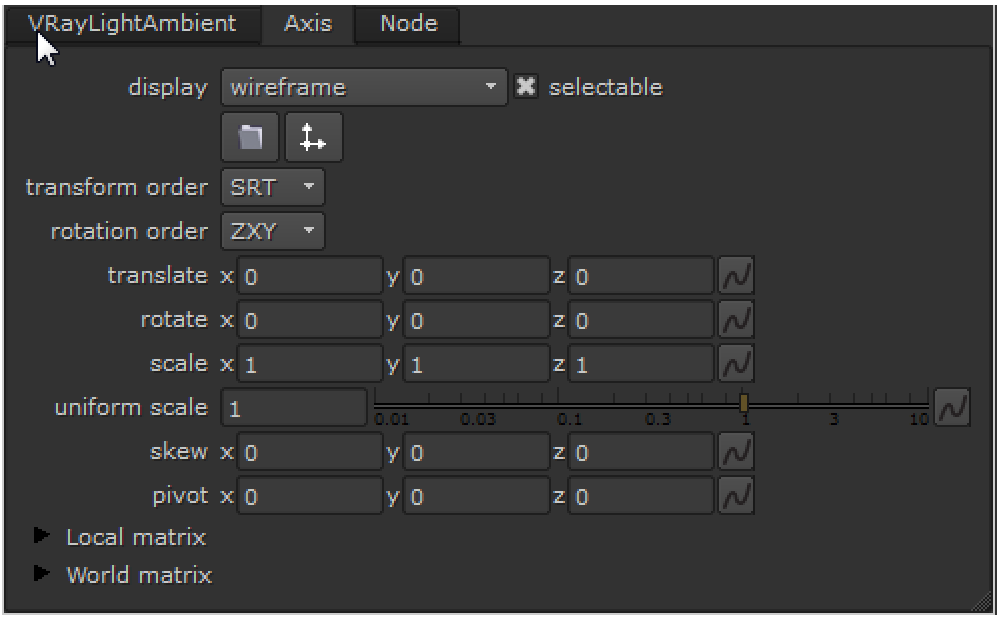
Add a **VRayLightAmbient** into the Nuke script and connect it to the **Lights** scene. This will act as the fill light. You could use the **VRayLightDome** as a fill light, but first let's look at how to use **VRayLightAmbient** as the fill.

You will notice that this light's default intensity is very strong and flattens out the lighting. Change its **Intensity** value to *0.01*. This lights the dark areas in the render so that you don't have total blackness in those parts of the image. At this low intensity, the effect is very subtle.



Color 1
Intensity 0.01

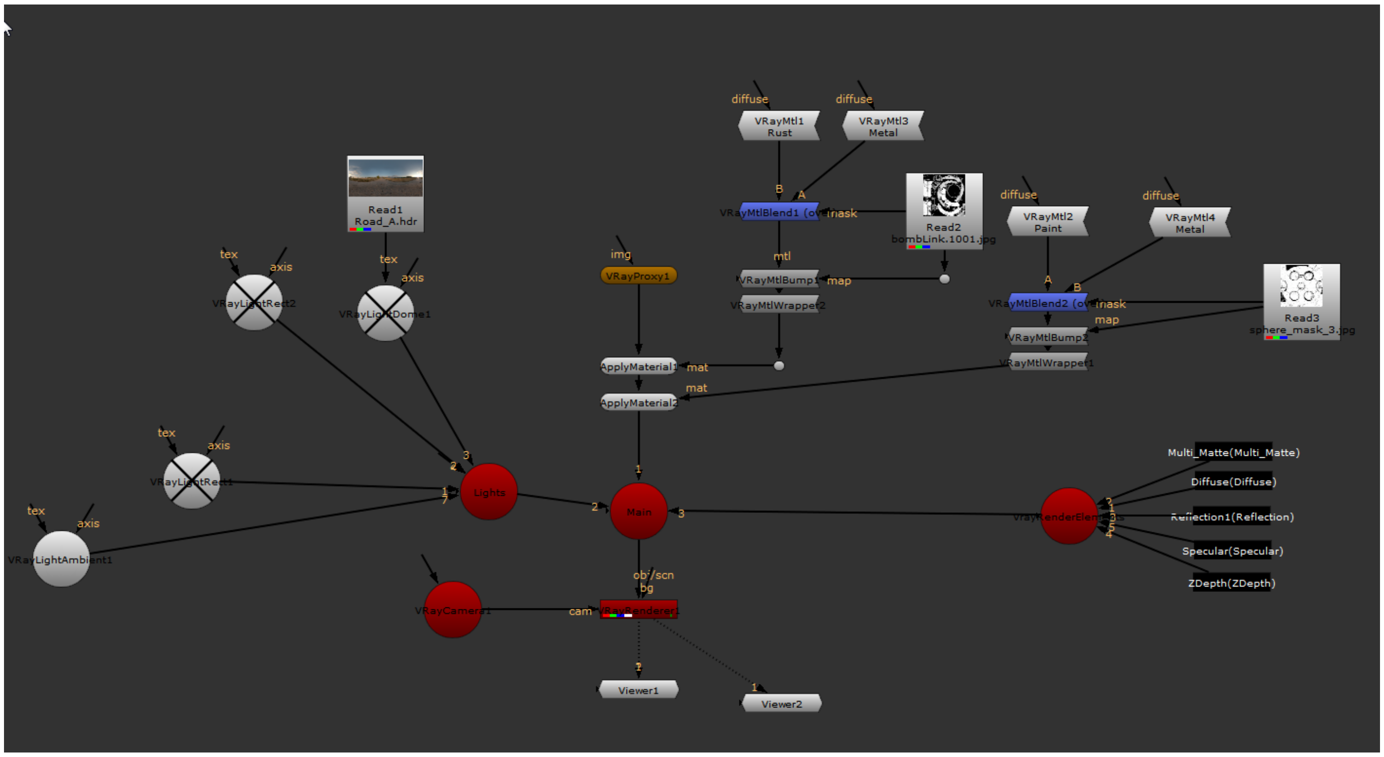
VRayLightAmbient properties



display wireframe
selectable
transform order SRT
rotation order ZXY
translate x 0 y 0 z 0
rotate x 0 y 0 z 0
scale x 1 y 1 z 1
uniform scale 1
skew x 0 y 0 z 0
pivot x 0 y 0 z 0

VRayLightAmbient position

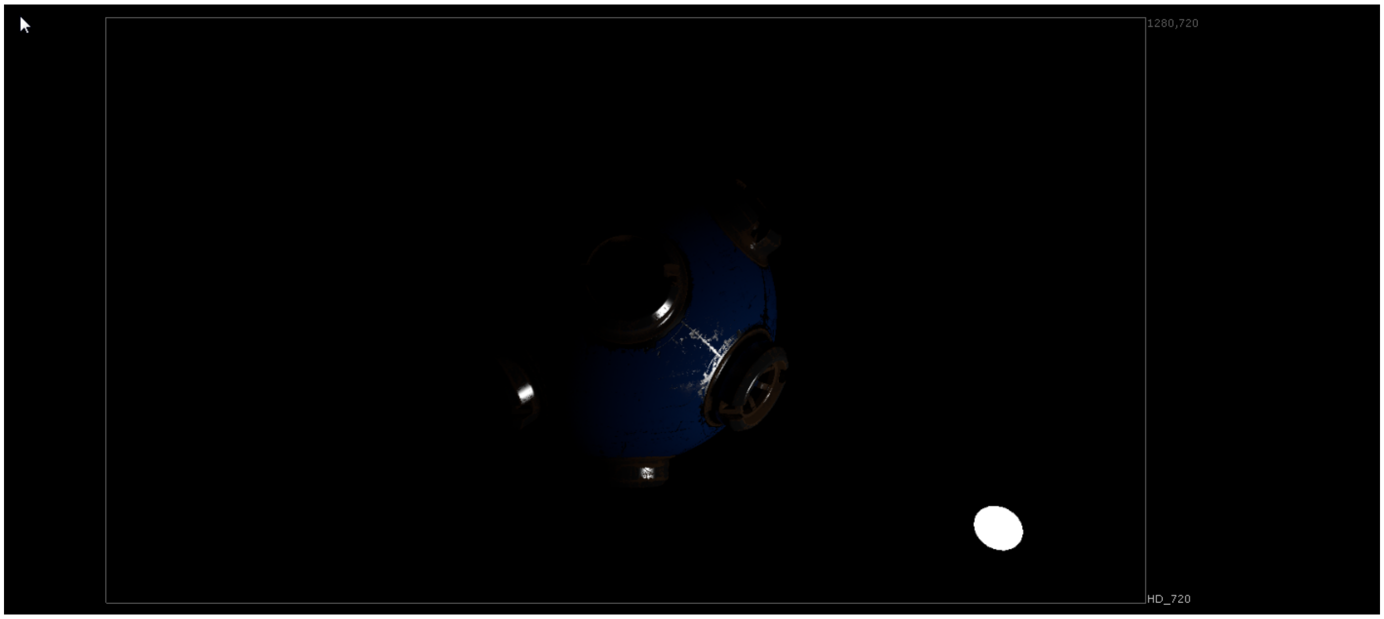
If you change the connection over from the **VRayLightAmbient** to the **VRayLightDome** (with the HDR connected to its **tex** slot) you can compare how the fill lighting compares. If you drag and select each **VRayLightRect** and enable and disable it, you can see how each of those lights affect the scene.



Scene Layout so far

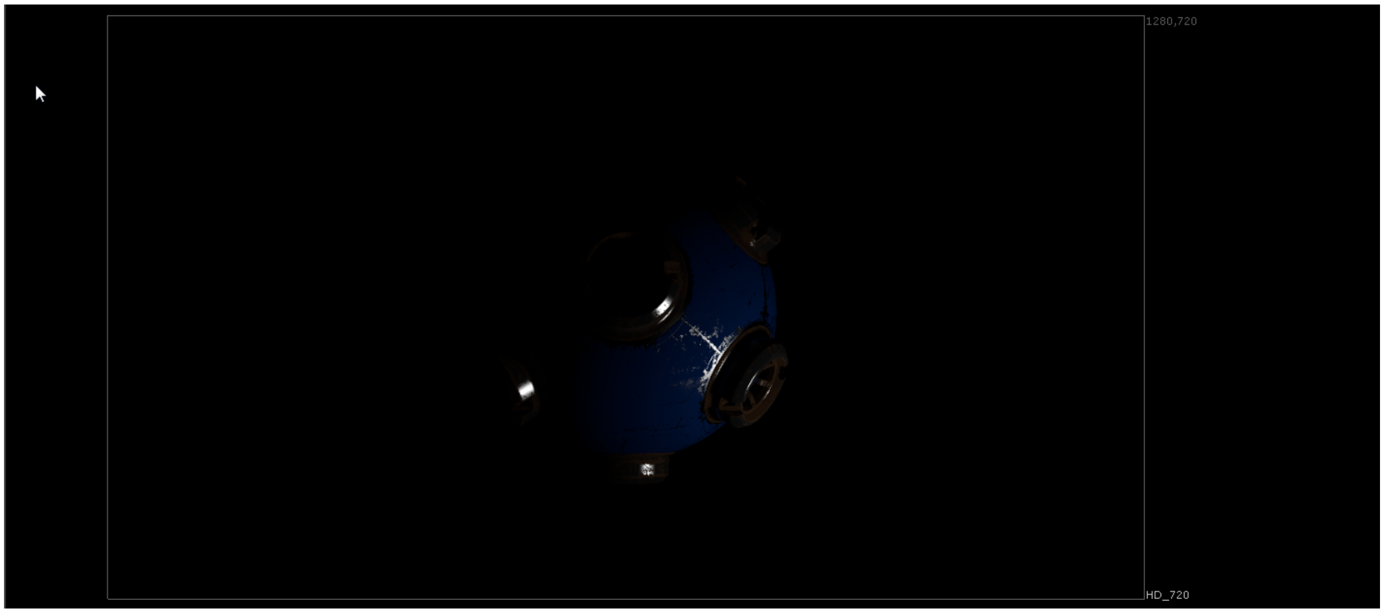
4) Adding a VRayLightSphere

Add a **VRayLightSphere** (a point source light) into the Nuke script and connect it to the **Lights** scene. Disable all the other lights in the Nuke script so you can work on setting up the **VRayLightSphere** on its own. In the 3D view, move this light over to the right a little. As this is a large scene, we will need to scale up the light by changing the **uniform scale** value to *100*.

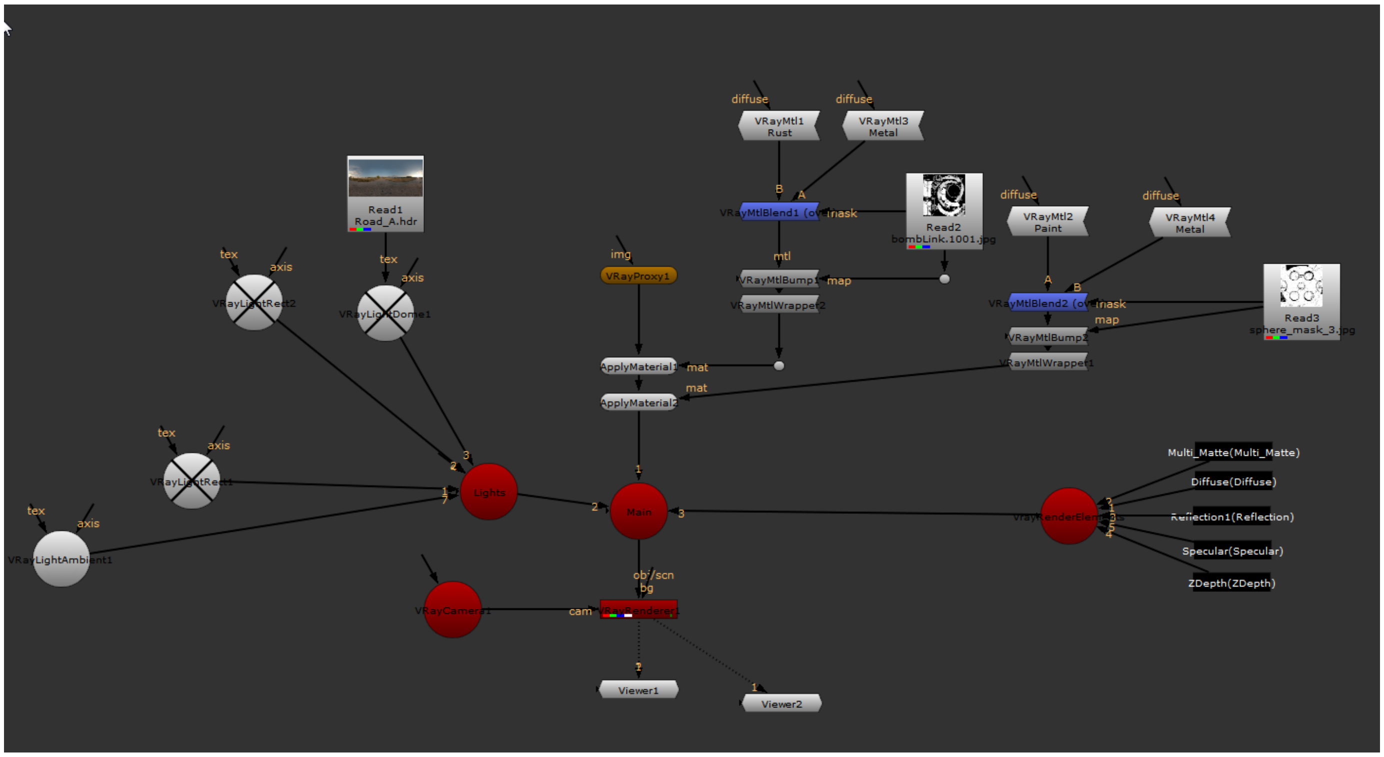


VRayLightSphere visible in the render

You will now be able to see the **VRayLightSphere** lighting the geometry, but you can also see the light itself in the render. Under the Options section for the **VRayLightSphere**, enable the **Invisible** checkbox. You will notice that although the light is still lighting the geometry, it is no longer visible in the render.



VRayLightSphere now invisible in the render

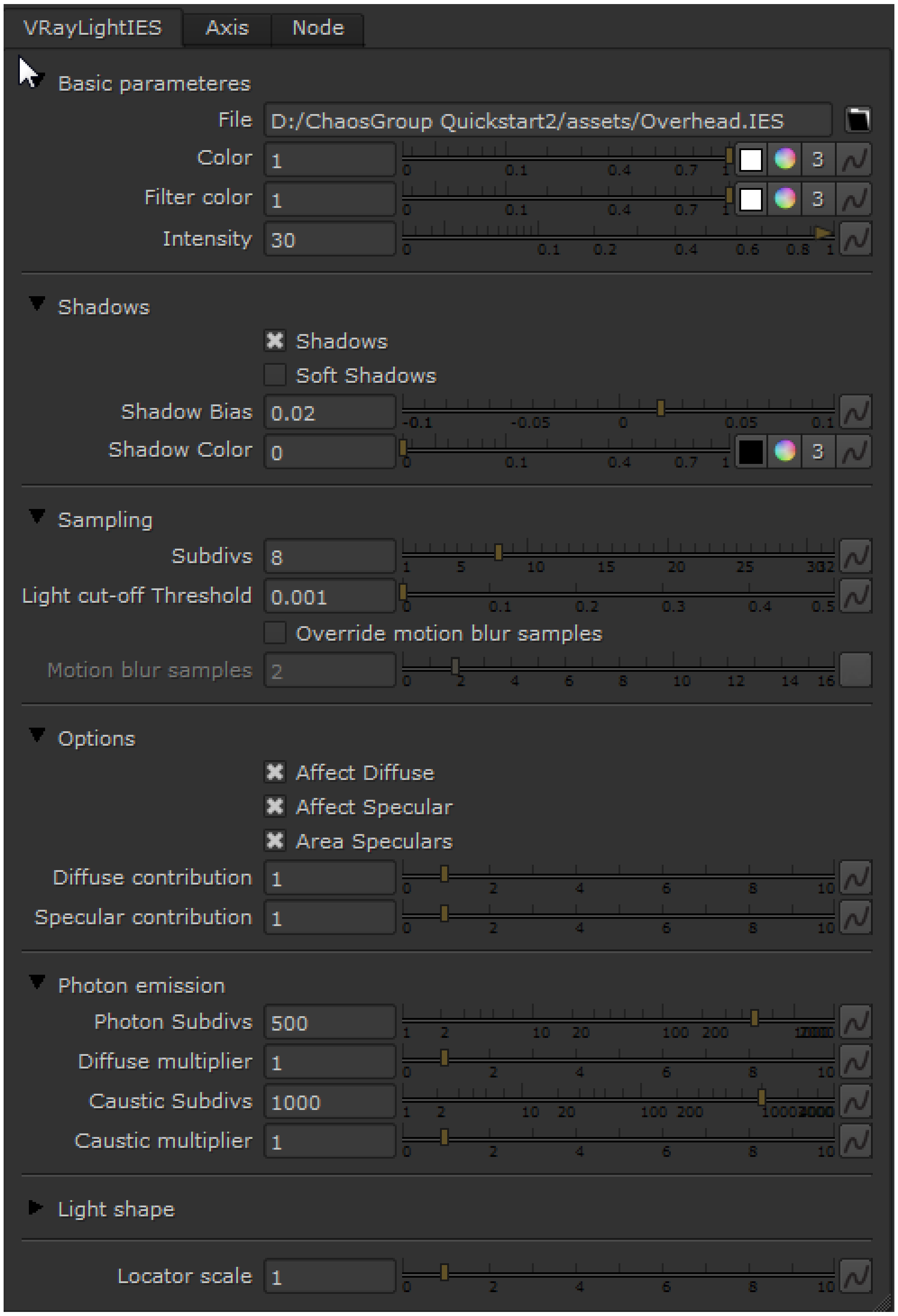


The layout of the Nuke script so far

5) Adding a VRayLightIES

Let’s next take a look at the **VRayLightIES** by adding it to the Nuke script and connecting it to the **Lights** scene. Ensure that for now the **VRayLightSphere** and all other lights are disabled so that you can only see the lighting contribution made by the **VRayLightIES**.

An IES file is a descriptor of a real world light. In the properties for this light, browse for the **Overhead.IES** file so it is used to illuminate the scene.


VRayLightIES | Axis | Node

Basic parameteres

File	D:/ChaosGroup Quickstart2/assets/Overhead.IES
Color	1
Filter color	1
Intensity	30

Shadows

- ✓ Shadows
- ☐ Soft Shadows

Shadow Bias	0.02
Shadow Color	0

Sampling

Subdivs	8
Light cut-off Threshold	0.001

- ☐ Override motion blur samples

Motion blur samples	2

Options

- ✓ Affect Diffuse
- ✓ Affect Specular
- ✓ Area Speculars

Diffuse contribution	1
Specular contribution	1

Photon emission

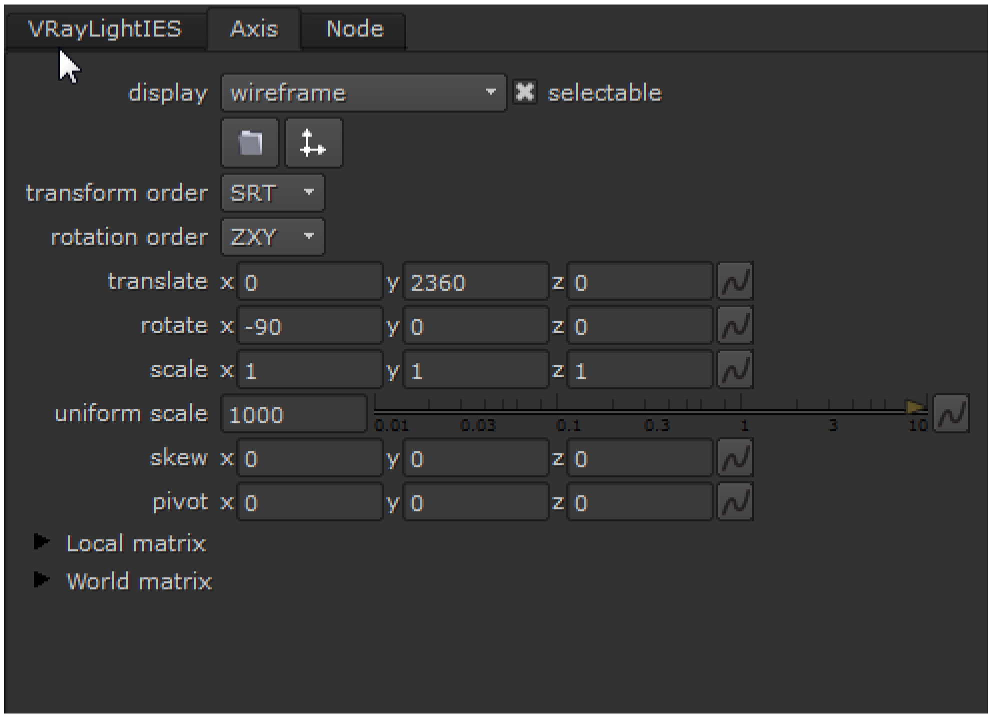
Photon Subdivs	500
Diffuse multiplier	1
Caustic Subdivs	1000
Caustic multiplier	1

Light shape

Locator scale	1

VRayLightIES properties

Move this light up above the geometry change its **uniform scale** value to *1000*. You can tell which way it is pointing by looking at the circle on the shape that represents the **VRayLightIES**. As you can see, it's currently not pointing at the geometry. Change its **rotate x** value under the Axis tab to *-90* to make the light point at the geometry.



VRayLightIES | Axis | Node

display: wireframe ✖ selectable

transform order: SRT

rotation order: ZXY

translate x 0 y 2360 z 0

rotate x -90 y 0 z 0

scale x 1 y 1 z 1

uniform scale 1000

skew x 0 y 0 z 0

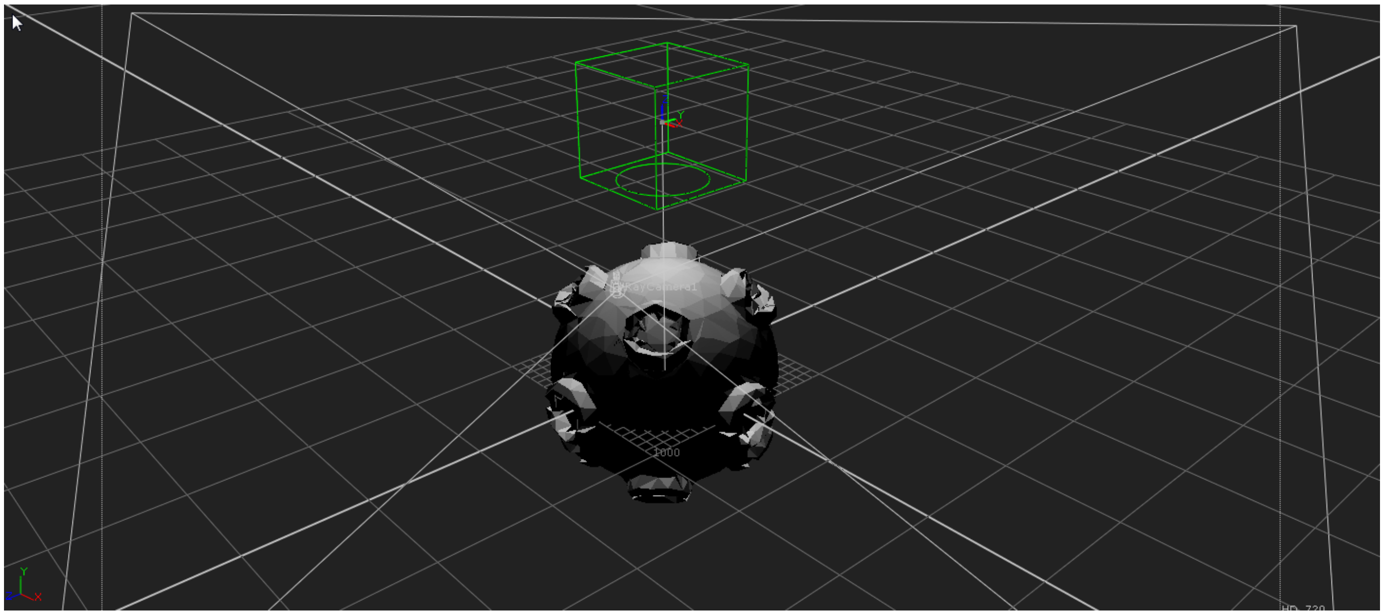
pivot x 0 y 0 z 0

▶ Local matrix

▶ World matrix

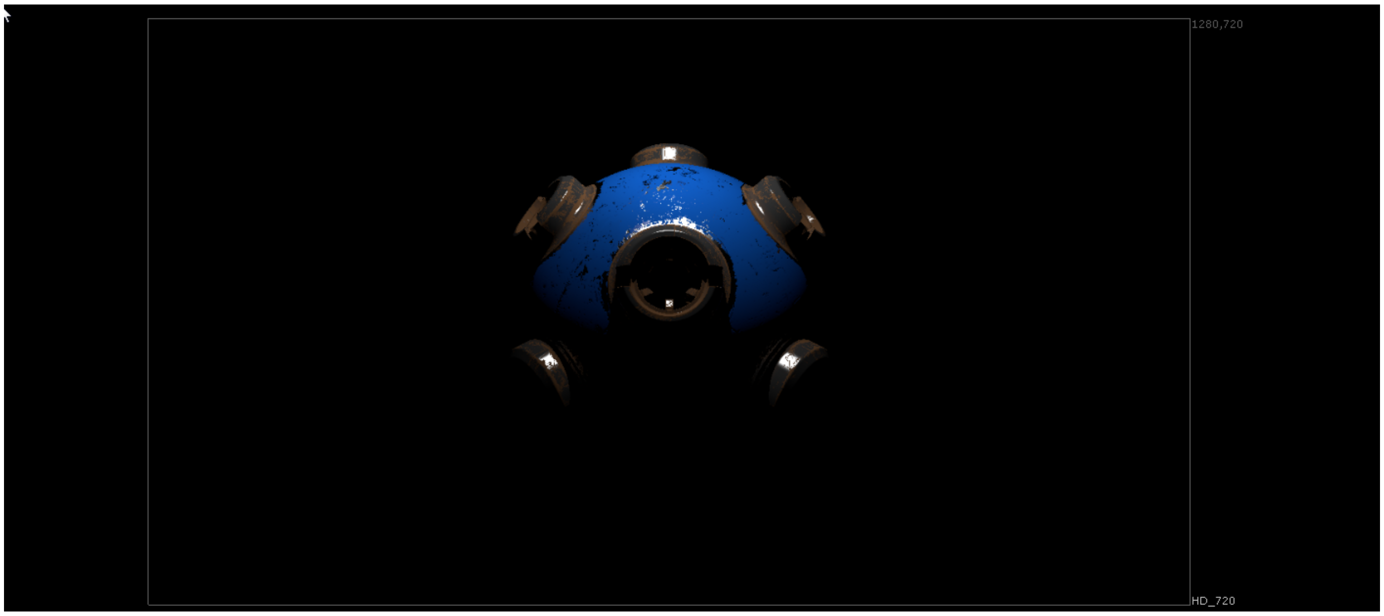
VRayLightIES position

Once again you should now have the light pointing at the geometry from the top, which you will now be able to see in the render.



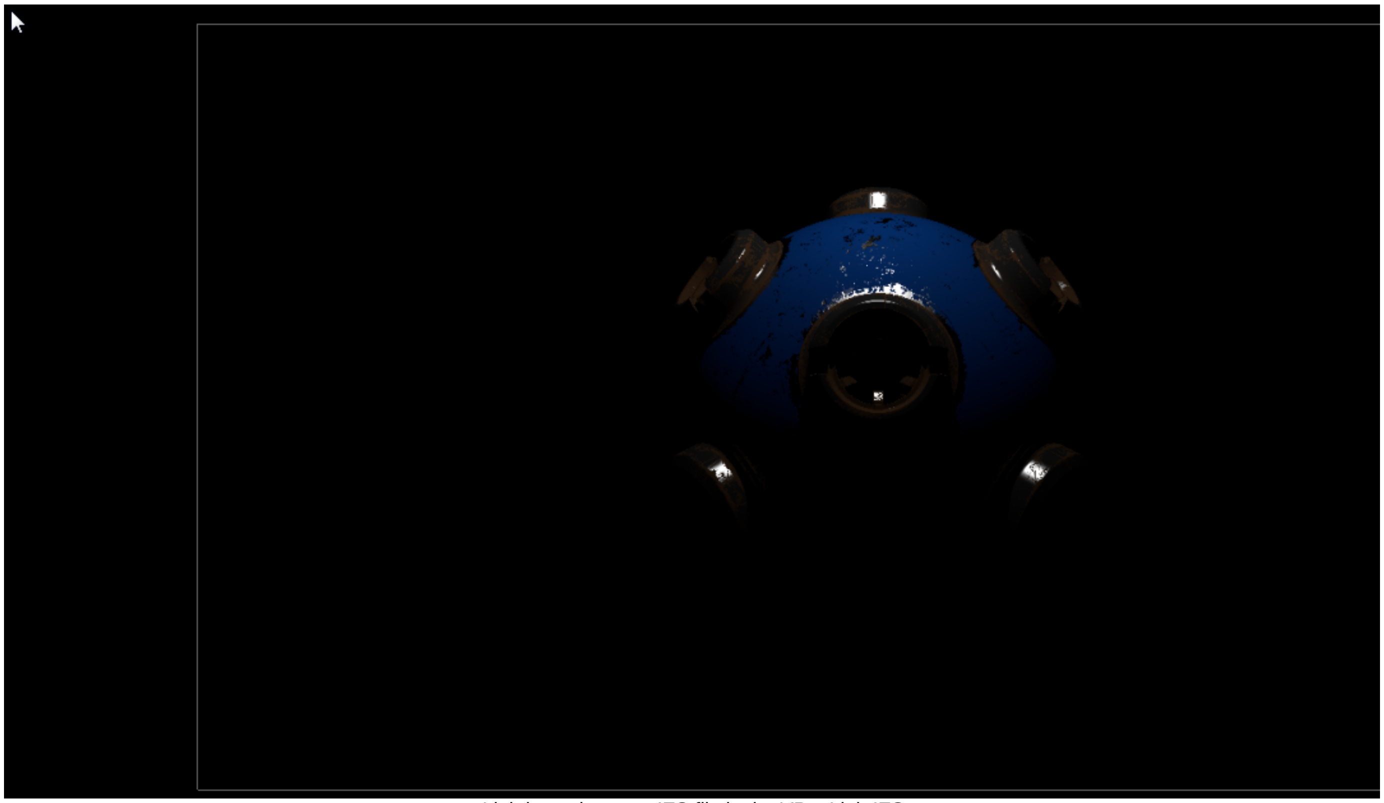
The VRayLightIES's position in the 3D viewport

Now that you can see the **VRayLightIES**'s lighting contribution to the render, we can turn its **Intensity** value up to *30*. You'll now notice that its lighting contribution is considerably brighter.



The lighting using the Overhead.IES file in the VRayLightIES

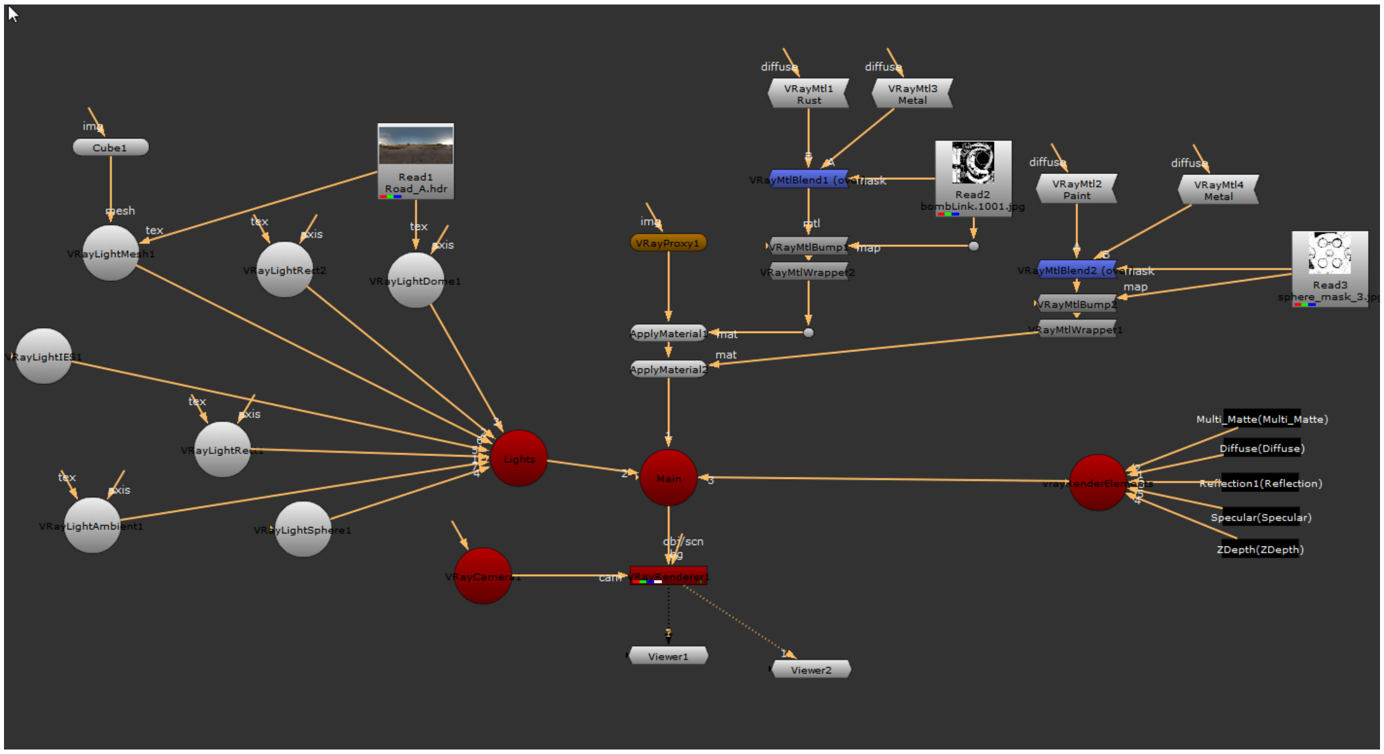
Swap the **Overhead.IES** file out for a pear-shaped IES light by clicking on the **VRayLightIES** node and then browsing for **pear.IES**. You should be able to see the difference in the light shape in the render.



Lighting using pear.IES file in the VRayLightIES

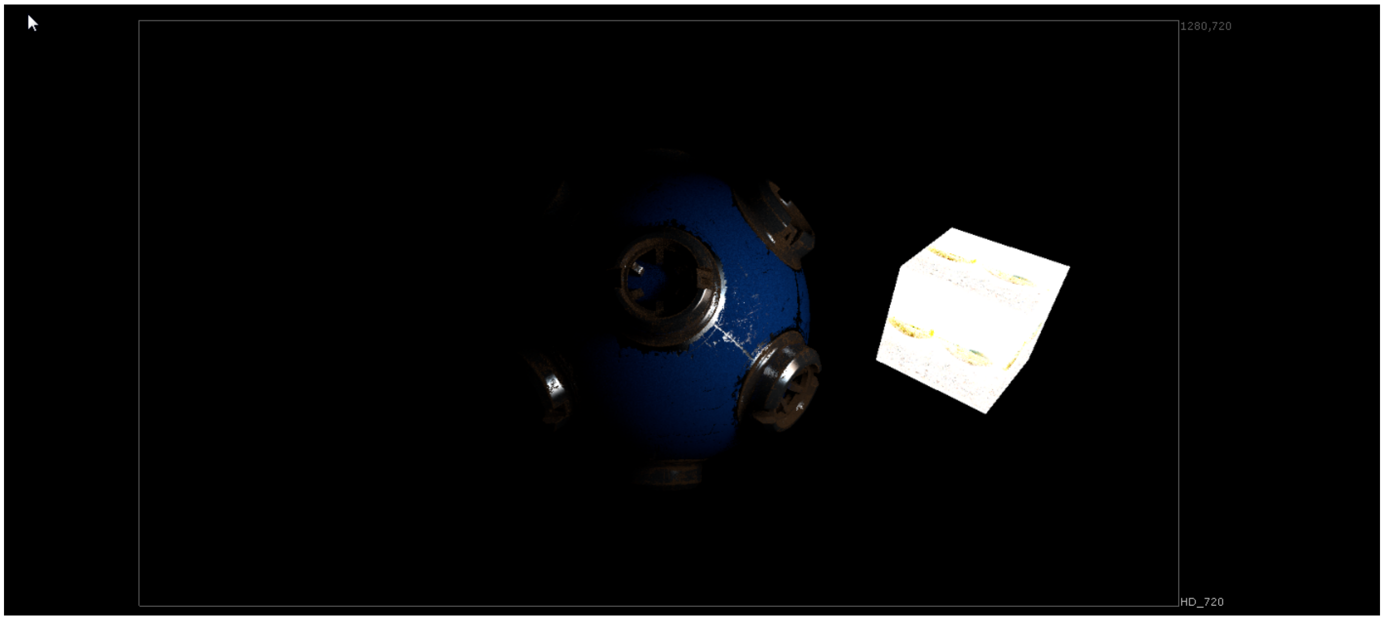
6) Adding a VRayLightMesh

Let's check out the **VRayLightMesh** next. This allows any mesh to be used as a light source within the Nuke script. Add a **VRayLightMesh** node into the Nuke script and connect it to the **Lights** scene. Add a **Cube** node (which we will use as geometry for the VRayLightMesh) and connect this to the mesh input of the **VRayLightMesh** node.



Final Scene layout of the Nuke script

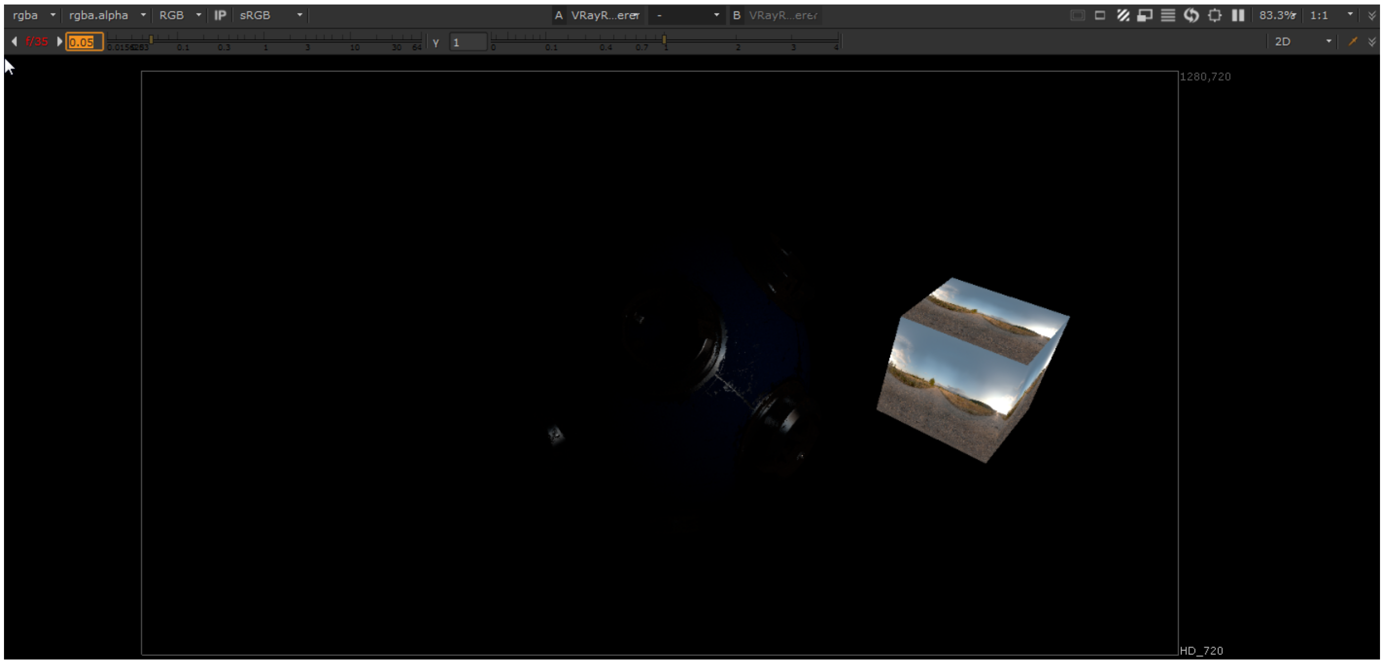
Move the **VRayLightMesh** up on the Y axis in the 3D view, and move it to one side on the X axis. Now once again the scale is too small for the large scene scale, so change the **uniform scale** under the Axis tab to *500*. **Connect** the **tex** input of the **VRayLightMesh** to the **Road_A.HDR** that we have been using so far in all the QuickStart videos. You can now see that the **VRayLightMesh** is both illuminating the render and also is visible in the render. Go to the main properties for the **VRayLightMesh** and enable the **Invisible** option. You will no longer be able to see the **VRayLightMesh** within the renders, even though it is still lighting the scene.



The VRayLightMesh visible in the render prior to setting it to Invisible

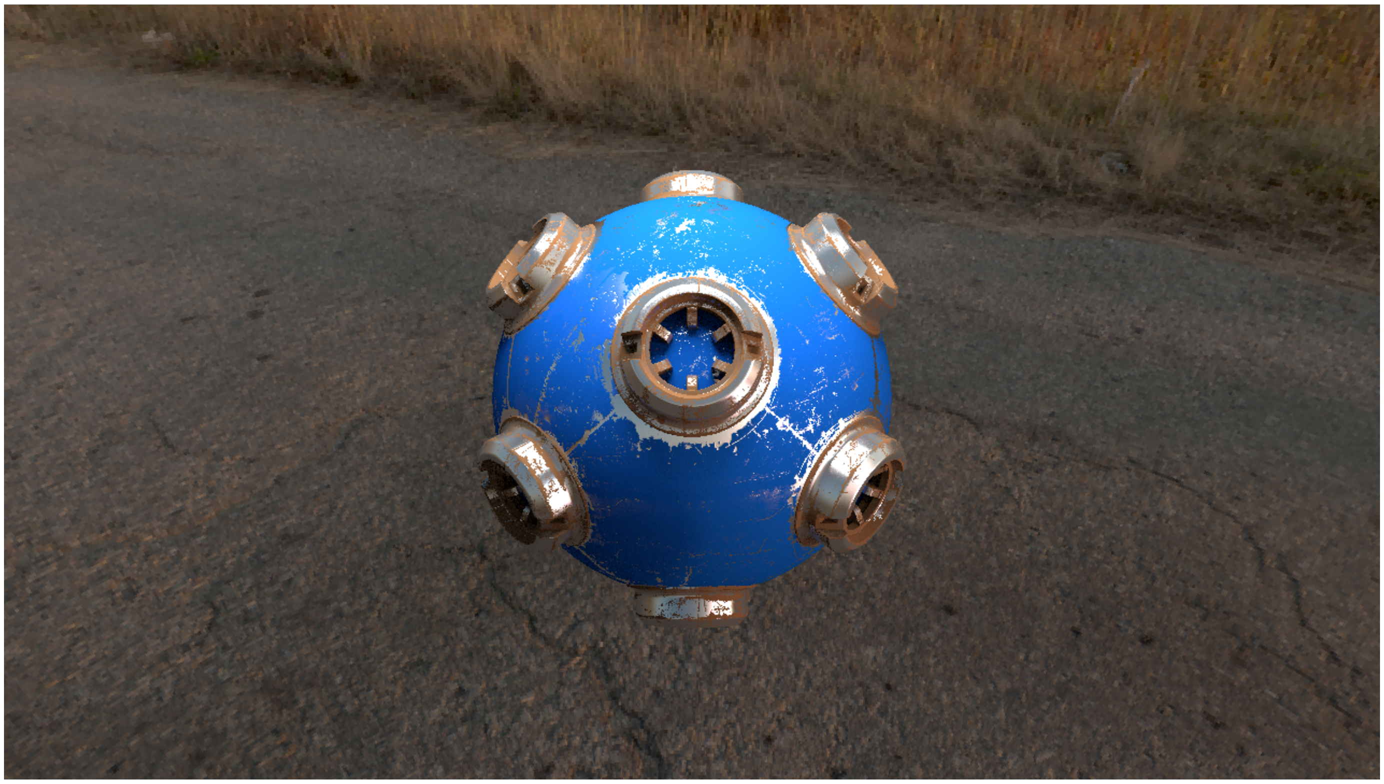
Disable the **Invisible** checkbox and drop the light's **Intensity** down to *1*. You will now be able to see the **Road_A.HDR** texture on the **Cube** that we are using as the mesh and how the colors in that HDR are contributing to the lighting from the **VRayLightMesh**. If you change its **Intensity** to *30* once more and drop the **exposure** in the viewer down to *0.05*, you will see that the texture is still being used even when the light is turned up considerably.

Return the viewer to normal once again and enable the **Invisible** checkbox.



With viewport exposure turned down, light from texture on the mesh is visible

The **Cube** we are using as the mesh input for this light could be moved around or even animated. If you have an animated mesh object in your scene, you could have that object produce light within the render.


Final render with all lights active

Turn on all the lights in the Nuke script back on by selecting each light, then hovering over them pressing the **D** key. With all the lights active, you can see the final lighting scheme we have produced.

In this tutorial, we have covered all the different light types that come with V-Ray for Nuke and how you can use them to light a scene.