This is not intended as a full manual, but rather a “good to know” document.
Most of the nodes have tooltips that should provide you with directions on how to use them.
But if you have any other questions please contact me at **nuke@higx.net**

Overview:

Installation:

- Place the /higx/ folder with all of it's content into your ~/.nuke folder.
- Open your ~/.nuke/init.py in a text editor
- Add a new line to the end of the file with the following text:

nuke.pluginAddPath('./higx/PointRender')

- Save and close the init.py file
- When you open nuke you should now see a new Higx icon in your tool row containing the tools

First Usage:

The first time you launch the point render it will take a bit of time before the node is ready as BlinkScript is compiling and caching the node into the system. When this is done once the node will load instantly next time you add it.

General:

Point Render

Inputs:

The following inputs are required:
Point Data Add your point data stream into the
Camera Add your camera into the camera input.
**Please note that the point render does not automatically source the Focal Length and Hapeture of the camera and as such you must link this manually!*
**You can also link the focal depth to the dof slider.*
**Also, due to a bug in Nuke, if you add a time offset or other time node after your camera it will not be picked up by the point render.*

The two other inputs are optional:
Occlusion let you make a holdout from a depth map.
The **red channel** should be the depth map, and the **alpha channel** should be white where the holdout should be applied.
Aux makes it possible to add a 4th render direction along a vector that you define. For example adding a constant with 0,-1,0 will make a wire that goes down.

Render Modes:

Point Render

This is the default rendering method and will by default render out 1 pixel (4x4 with filtering) additive points. If you enable “Depth Buffer” in the facet rendering options it will try to depth sort the points but it won’t be correct.

Wire Render
The wire render mode will draw lines between each point and it's neighboring points in the point position map of the point data.
This means that if you have point data that is randomly distributed, the wires will also connect randomly.
There are 2 connection modes. **Normal** and **Nearest**.
Normal will connect like mentioned above, while **Nearest** will try to find the nearest point in 3d space and draw a line to that.
This makes nearest really good for point data that is randomly distributed.
At this point **Nearest** is limited to 100x100 points and its generally recommended to only use this mode for point data with as few points as possible.

Facet Render
This is an experimental rendering mode that is not fully supported by all features and can cause artifacting.
Unlike all the other rendering modes, you can use the Depth Buffer option to only render the front most layer of geo.

FOG:

The fog is a simple way to darken points based on the distance from the camera.
Fog Near defines where the points should start to become darker and **Fog Far** is the point where the points are completely gone.

Depth Of Field:

Recommended settings:
Low - Millions of points
Medium - Hundred Thousands of points.
High - Ten Thousand points
Extreme - Thousand Points

Motion Blur:

Motion Blur is set to **Fast** by default. Fast will only do 2 temporal samples and interpolate between the two. That means that if the nodes you have upstream are computationally taxing it will make them faster. This however also means that it will not include sub-frame animations.
So if you need to have textures or other detail that have sub-frame detail that cannot be interpolated between then turn this feature off.

Generators:

Point Plane

The point plane is your basic flat card, like the Card geo node.
In the general section you get the basic controls such as number of points

The offset settings is some creative controls that let you offset the points or create gabs
This is best used in conjunction with the wire-render mode as it gives you some weave like patterns

---

Point Sphere

The point sphere is your basic sphere that mimics the look of the sphere geo node.
The points at the poles appear brighter because the points there are packed together, but you can use the “Reduce Pole” option to avoid this effect.

If you use wire and facet rendering you will get a the edges of the sphere won’t connect by default. Go into Render Settings of the point render node and hit “Close X” to make the sphere complete.

Point Cylinder

The point cylinder is your basic cylinder that mimics the look of the cylinder geo node.

If you use wire and facet rendering you will get a the edges of the cylinder won’t connect by default. Go into Render Settings of the point render node and hit “Close X” to make the cylinder complete.

Point Torus

The Point Torus is your basic cylinder that mimics the look of the cylinder geo node.

If you use wire and facet rendering you will get a the edges of the Torus won’t connect by default. Go into Render Settings of the point render node and hit “Close X” and “Close Y” to make the Torus complete.

The following generators are defined as “NU” or “Non Euclidean”
Since they are not defined as a Euclidean plane normal wire render mode and facet rendering might not give the results that you expect.
If you want to use wire rendering its recommended that you use the aux channel and specify your own wires, or that you use the nearest

---

Point Grid

The point grid is a bit like a bunch of stacked cards.

Point UnitSphere

The unit sphere works much like the regular sphere except for the fact that the points are Randomly distributed and as such don’t have any inherent connections.

Point UnitCube

The unit cube works much like the unit sphere except for the fact that this is just a cube

Point Particle (beta)

This is a simple particle system that will let you create small particle flows.
One thing to note is that at this point in time it's not possible to animate the attributes, as each frame is isolated from the surrounding frames.

Point Geo Source

The geo source is a basic example node that let you use your own geometry as a source.
One thing to be aware of is the fact that this uses the particle system in order to convert the geo data to image data, as such you will need NukeX to fully control the system.

Point Geo Source UV

Just like the Point Geo Source node, this node will convert incoming geometry to point data. This node will however convert them using UV rendering. So if your geo have a somewhat even UV layout, this should be the optimal solution.

Modifiers:

Point Expression

The point expression node is just a convenience node that let you control the point data directly. In this node pos.x, pos.y and pos.z is replaced by r, g and b.

Point Fractal

This is the standard fractal node that let you modulate the positions of the points using a fractal noise. The noise is split up into low and high frequency and as such you can control the frequencies individually.

Point Fractal Evolve

This is a more experimental node that let you evolve the basic fractal in a way that reminds a bit like a particle system. Basically a good way to "explode" your objects.

Point Radial Force

This node let you limit the point positions radially or in the x, y or z axis.

Point Transform

This is the basic 3d transform node that acts just like the transform geo node

Point Twist

The point twist node let you twist the points around a defined axis.

Point Merge

This node will let you merge two point streams into one.

Point Dublicator

This creates any number of unique instances of a point stream.
This node will create a new channel called id.r where each instance will get a unique id.

Shaders:

Point Light

This is a basic point light that starts of generating normals for the points, and then applies a basic phong light. This means that the input point data must be Euclidian.

Point Distance

This node calculates the distance from each point to a defined point in space.

Point Fractal Mask

The fractal mask is a easy way to break up the input data

Point Motion Shader

The motion shader calculates motion vectors for the points and output them as either a 3d vector or an1d magnitude.

Point Normal

This node calculates the normals of the incoming points based on euclidian space

Point Proximity

This node calculates the proximity to nearby points in euclidian space, one thing to note is that at this point this node does not calculate proximity across the edges of the map.

Point Reflection Shader

This node creates a reflection based on normals that are generated in euclidian space.

Point Texture

This node applies a texture to the points based euclidian coordinates.

Point Fractal Texture

This node applies a fractal texture to the points based worldspace coordinates.