



# Shader Engine

---



**Difficulty:** Hard

**Required skills:** Java, OpenGL, GLSL

This proposal is an experimental work about drawing large networks using GPU Shaders technology to improve performances and scalability. The aim is to create a basic 3D engine using suggested technologies and use it for drawing as many nodes and edges as possible.

Existing Gephi 3D engine is intended to be used with old hardware and so don't profit from GPU revolution. Basically the engine draw only very basic shapes (sphere or cube for nodes and lines for edges), but a LOT of them. Drawing millions of objects, even if it is the same object is currently far from possible.

Shaders technologies have interesting features for our topic. For instance Geometry Shaders and (Pseudo-)Instancing techniques enhance performance when duplicating objects. Other techniques could be explored to reduce the amount of needed vertices or computing.

We suggest this draft roadmap:

- Getting started with loading and executing vertex and fragment shaders within a JOGL routine in a sample application.
- Try pseudo-instancing techniques for duplicating an existing shape, for instance a sphere.
- If available, use Geometry Shaders to implement an instancing test case.
- Try to create a Shader which will help to reduce sphere mesh size. For instance see if it is possible to create a sphere from only few triangles.
- Experimentation for increasing edge drawing performance using Shaders.
- Propose nice post-processing effects if you like!

## Resources:

- JOGL Website (<https://jogl.dev.java.net>)
- GLSL (<http://en.wikipedia.org/wiki/GLSL>)
- GLSL Specifications (<http://www.opengl.org/documentation/glsl>)

- GLSL Tutorial (<http://www.lighthouse3d.com/opengl/glsl>)
- JOGL Developer forum (<http://www.javagaming.org/index.php?board=25.0>)
- OpenGL Geometry Instancing (<http://www.ozone3d.net/blogs/lab/?p=87>)
- Instancing resources (<http://www.gamerendering.com/2008/10/21/instancing>)
- OpenGL resources from Nvidia (<http://developer.nvidia.com/page/opengl.html>)
- Molecular visualization PyMOL (<http://pymol.sourceforge.net>) and QuteMol (<http://qutemol.sourceforge.net>)