Tutorial 10: Pure OpenGL coding in OSG

Franclin Foping

franclin@netcourrier.com

May 11, 2008

Abstract

We already know that OSG is built upon OpenGL. However, sometimes it may be useful to be able to embed an OpenGL based code in OSG. As we will see through some examples, this way can let you define a very nice Bezier curve and patch.

1 Introduction

Sometimes, you may feel nostalgic about some outstanding OpenGL features that you are still struggling to implement. For example, let us assume that you have just implement a parametric surface in OpenGL and that you are yet to port in to OSG. Well, the only possibility will be to hard code the same thing in OSG. The way around this issue is to make use of a **custom shape drawable and add it to a Geode Node**. We will how throughout this tutorial. I will assume that you are already comfortable with the rationale of Bezier curves and patches, if not please look at the internet, there are loads of valuable resources to help you out!

2 The Geode Node revisited

Recall from the first tutorial that the **Geode** class is a special type of node class which allows us to add a geometry object. This could be a mesh as

well. The Geode class accepts a **Drawable** object as child in order to be rendered. The trick to create our custom shapes as **Drawable** objects and bind them to a Geode object!
We will be using it this hint to render a Bezier patch and a Bezier curve made up of respectively 16 and 4 control points.
The actual drawing code is handled within the **drawImplementation** of the Drawable class.

3 Scene graph of our scene

The simple scene graph is illustrated in the next sketch.

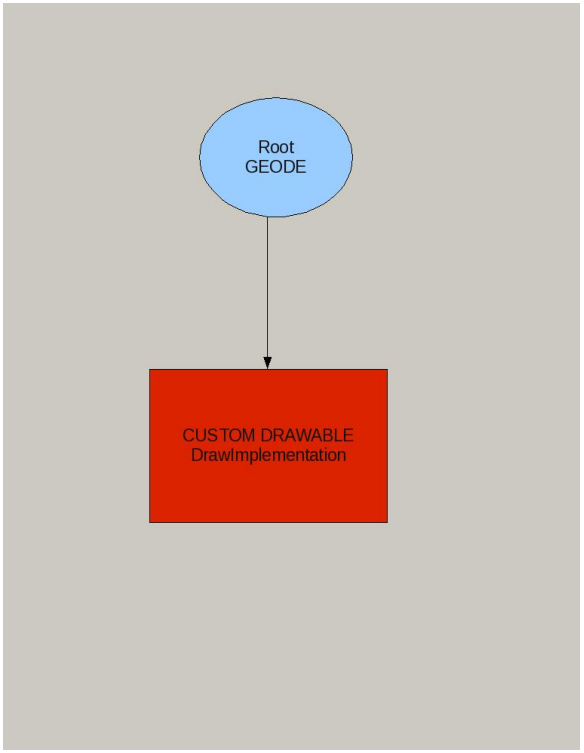

Figure 1: A scene graph

4 Results

Our final scene will look like the following picture.

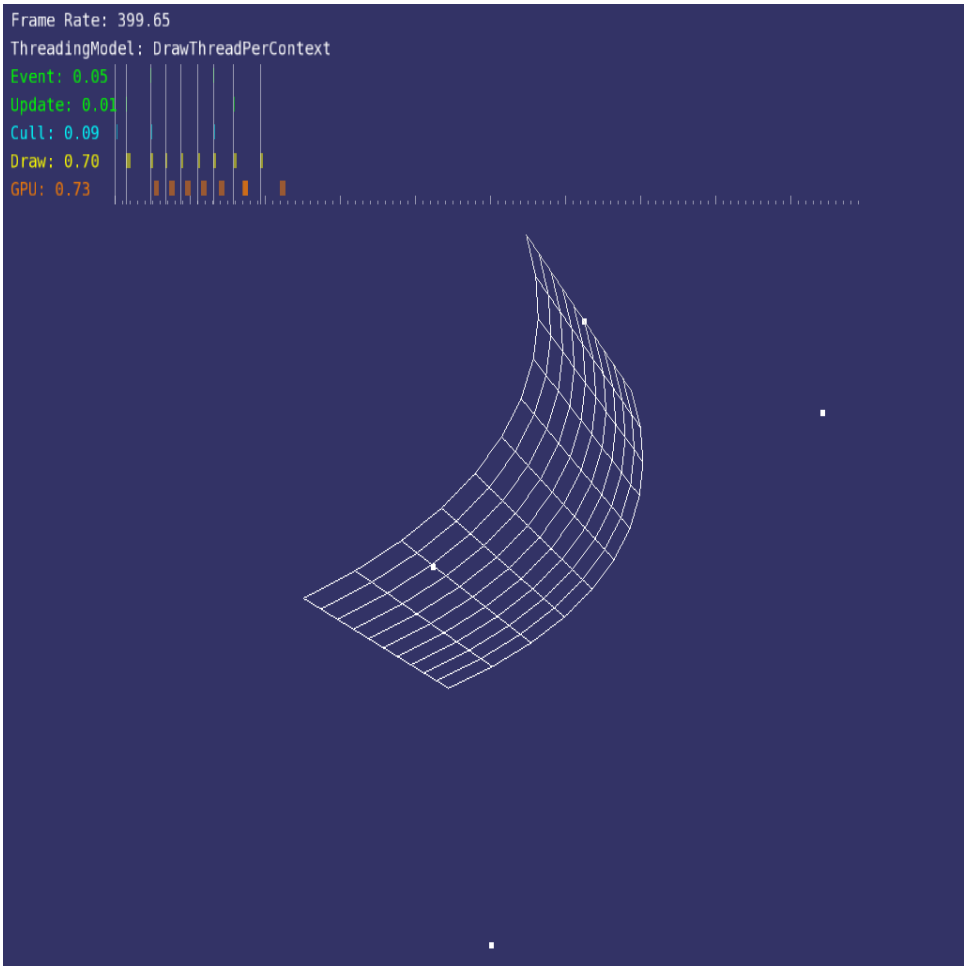

Figure 2: A Bezier patch with its control points

5 Do-it-yourself

Well, there are several further tweaking you should try.

1. If you look closely at the source code, I have added a separate snippet to render to Bezier curve. Comment the line 315 and uncomment the

previous line, that is the line 314. Recompile the code and run it, you should now see a Bezier curve.

2. Uncomment the line 317 and comment the line 315, you should now see a texture quad.

3. Look at the *drawImplementation* in the source code, you should see some hints to define materials to your objects, have a go at them!

4. By default, OSG makes use of display lists to improve the performance of our application, by uncomment the line 259, you are actually tell OSG not to use display lists but to send your geometry to the GPU exactly the same way you have defined. Your task is to understand what will happen to your application and to be able to explain why a significant frame rate loss was encountered? Do you think it was the best way to handle it?