
Shader Instancing Performance

vs

Static Geometry & independant Entities

Summer of Code 2006

Student: Griffon Jean-Baptiste - [Crashy](#)
Mentor: Paul Cheyrou-Lagrèze – *Tuan Kuranes*

Introduction

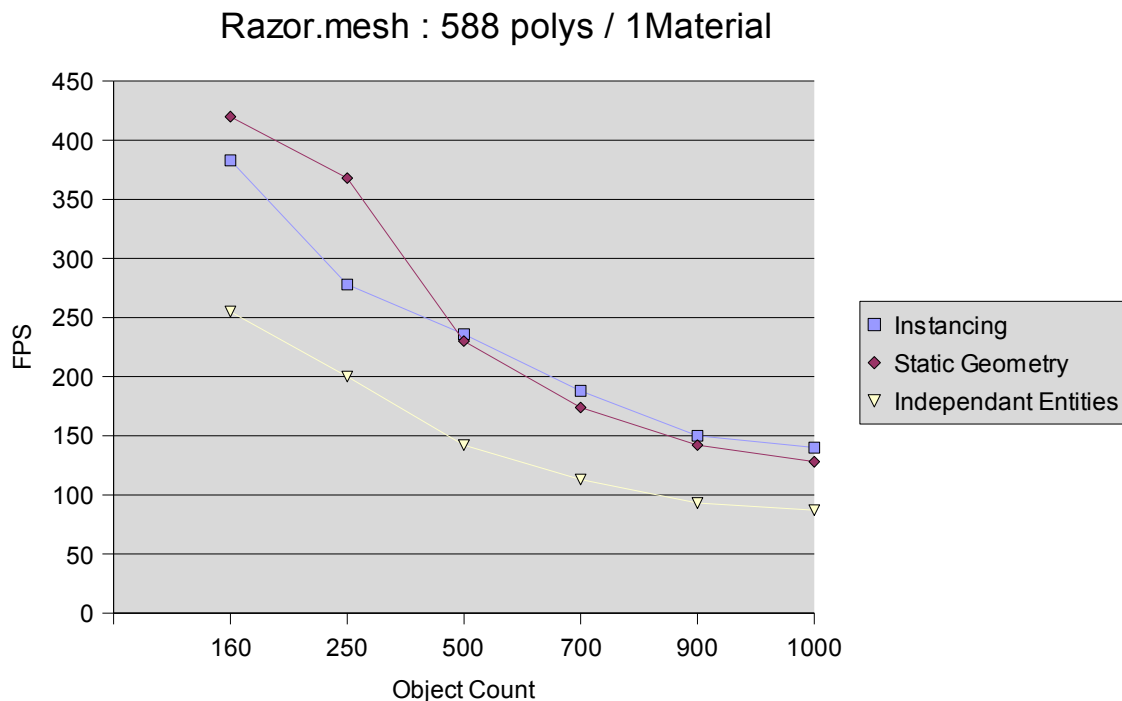
Basically, the objective of this document is to show how behaves instancing and to compare the performance with the classical rendering techniques that were provided by OGRE, and of course to analyze the benchmarks to help you to use the good technique according to your needs and your application.

The Benchmarks

- **First part: object rendering only**

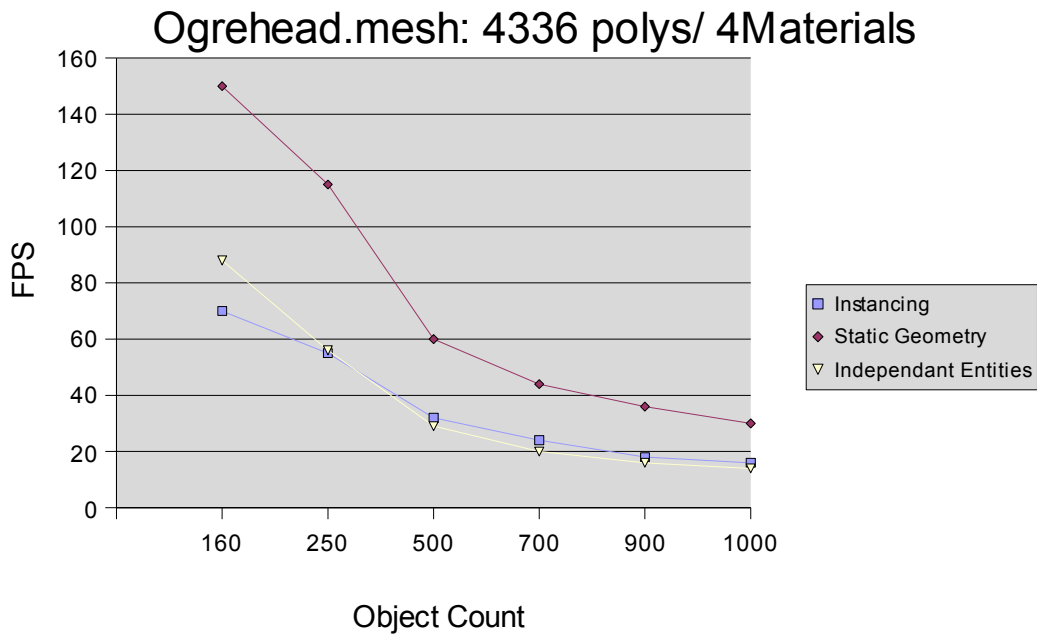
For the moment, we're only considering the performance when only the objects are used in the scene. It is just to see the pure speed of each technique.

The first test is made with the razor mesh.



As you can see, for this quite simple mesh, the Static Geometry start at the head of the bench. But as the object number increases, the relative performance of the Shader Instancing increases too. Rendering the objects as independant entities is by far the slower technique to choose.

The next test is made with the Ogrehead mesh. This mesh is really more complex than the Razor mesh, with more than 7times its polycount, and multiple vertex buffers.

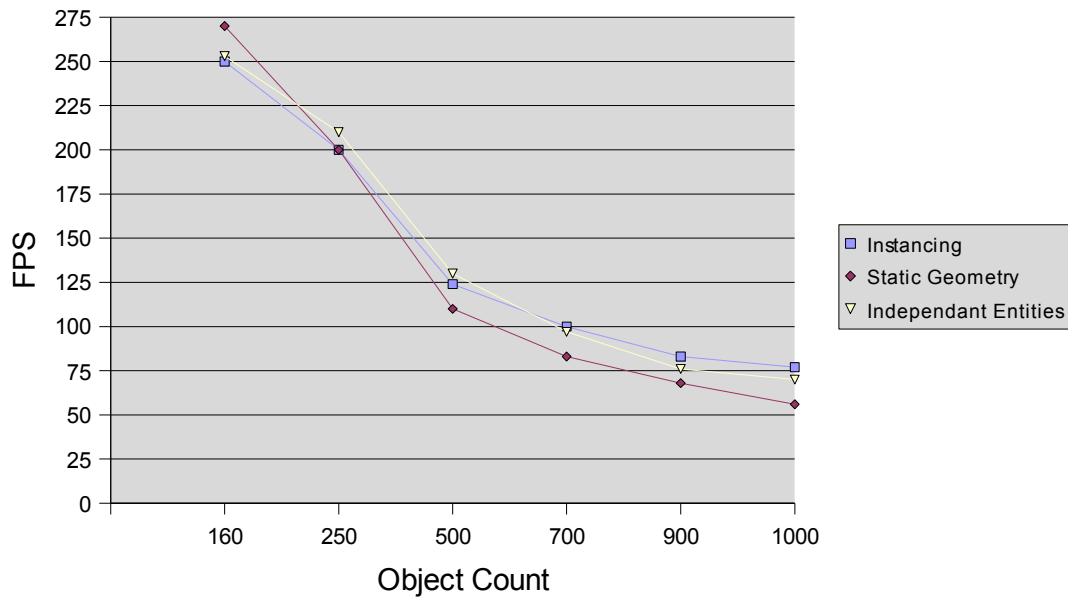


Here, the performance are really, really different. The instancing start as the slower technique, but after 300 objects it becomes to be a bit more effective than the independant entities. The Static geometry is the fastest technique. But as you can see in the graphics, its curve may intersect the two other curves with 1400 or more objects. Keep in mind that for the moment we're speaking of object rendering only.

Note: with an ATI X800 card, the texture of the tusks doesn't appear when we use Instancing. But with an nVidia 7600Gs there is no problem at all. As the current tests are made with the X800 card, I don't know if this bug has also a performance cost.

The final test of this part is made with the knot mesh. This mesh is between the two previous one on the complexity level, with 2880 polys and only 1 material.

Knot.mesh: 2880 polys/ 1Material



The results are interesting, as the three techniques have appreciably the same performance. As usual, the Static geometry starts as the faster one. But at the difference of the Ogrehead mesh, it finishes as the slower one. The Instancing follow a very different sheme, starting as the slower one, and finishing as the faster. The third technique is between the both except between 240 and 600 objects.

Conclusion of this part

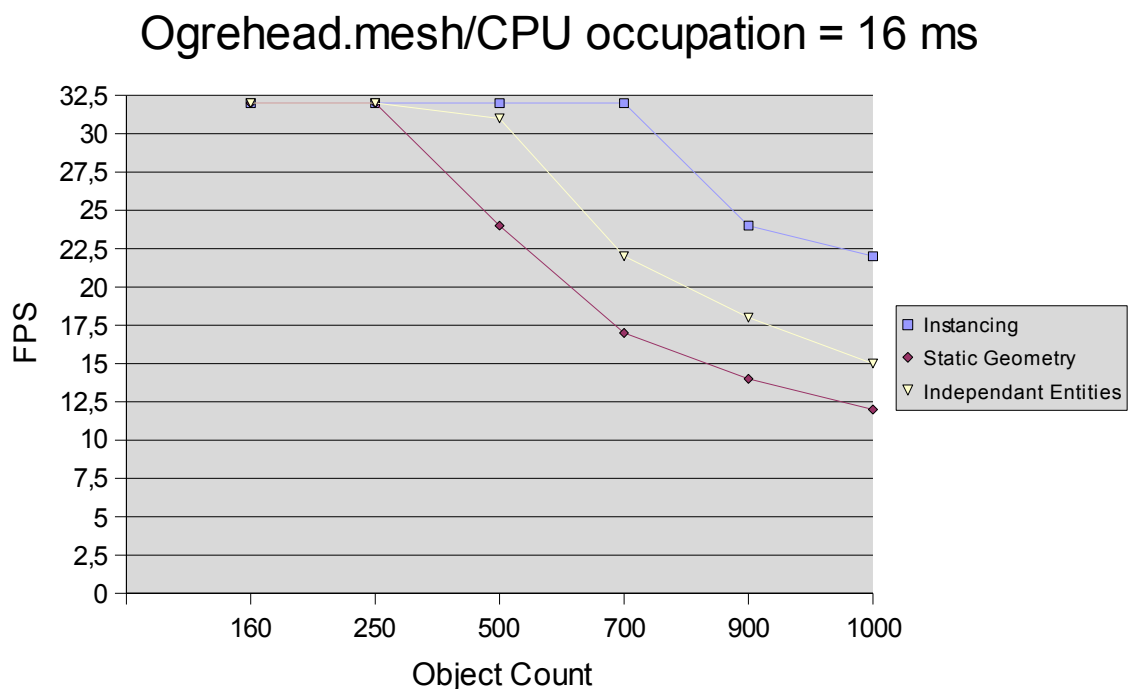
The first thing that we can say is that we cannot really separate the techniques. Except of the independant entities, which appears as the slower technique in the most of the cases, the Static Geometry and the Instancing have a varying behaviour, according to the object displayed.

Although, there are good points for the instancing and Independant Entities: the object can be moved. That's not really usefull if you want to render trees or rocks, but could be really usefull to represent, for instance, an asteroid field, with each asteroid moving independently. You can also add little differencies to each object when using this two techniques.

- **Second Part : adding some CPU occupation**

The objective of this part is to see which technique behaves the best when the CPU is heavily requested, making a lot of computations in addition of the work made to handle the objects. To occupy the CPU, we only use a while loop that loop during a given time. For this test, the loop is during 16 ms.

As the Ogrehead mesh was the one with which performances were the worst for the Instancing, we choosed this one to see how Instancing behaves in a more «real» situation. Let see the Benchmarks:



Now it becomes really interesting. We can see which technique is the most CPU-bound, and then determine which is the most GPU-bound.

With less than 250 objects, all techniques have the same performance, reaching a maximum of 32 FPS. But as we increase the object count, the static geometry becomes less and less effective, as, to a lesser extent, the Independent Entities.

The instancing, which was really slower than the Static Geometry technique with no CPU occupation, is now the fastest! It is not so unexpected, because when using instancing, most of the computation to handle the objects is made by the GPU.

Global conclusion

As a general conclusion, we'll just explain when to use instancing, and when to not use it.

When to use Instancing

As the last test highlighted, the instancing is really advantageous when you have a lot of stuff computed by the CPU, as it moves the main part of object rendering to the GPU. The other advantage of the instancing is to give you the ability to make slightly differences for each object that are in your batches. For instance, if you are using something like atlas texture, you could use different texcoord according to the objects index.

So instancing is really usefull to render multiple times the same geometry, as with the Static Geometry, but also to have little differencies on each object, as if you were rendering objects as independant entities.

When to not use Instancing

It is not a so good idea to use instancing if you really don't need to render more than 100 idencal objects. You may have a performance enhancement, but it may not be necessary. The other case is when your application is GPU bound. As instancing moves most of the computations to the GPU, if the GPU is already overloaded, it won't be a good idea to use it.

Same thing if your application is not CPU bound. In this case, you won't really see a performance improvement.