C 4 Using Instancing to Efficiently Render Super Carbon Nanotubes

Authors: Burger, Michael, Bischof, Christian

Using Instancing to efficiently render Super Carbon Nanotubes

Michael Burger · Christian Bischof

Abstract In this paper, we present an efficient way to visualize Super Carbon Nanotubes (SCNTs). SCNTs are complex, hierarchical structures and their models easily consist of more than 1 million atoms. Our SCNTs are modeled as graphs of uniform nodes and egdes. We show that OpenGL instancing is a very suitable technique for rendering such large graphs, because they only consist of two different types of geometry. Our visualizer software exploits this property and we demonstrate that it allows to render the tubes in a fashion that is time- and space-effective. We implemented auto-tuning of the model to the underlying graphics card through adaptive mesh-resolution-choices. We also designed and implemented our own shading programs in the OpenGL Shading Language (GLSL) to realize a sufficient but performant and configurable lighting computation. This allows us to render big models even on laptop GPUs and to cope with models that consist of 150 million triangles, which is still a challenging amount for most of today's graphics cards.

1 Introduction to Super Carbon Nanotubes and OpenGL Instancing

Carbon Nanotubes (CNTs) are an interesting field of research, because they can be used in a lot of applications. New synthesis approaches allow the composition of tubes with Y-shaped junctions to structures that possess again the

Michael Burger
FG Scientific Computing, TU Darmstadt
Mornewegstrasse 30, 64293 Darmstadt
Tel.: +49 6151 16-76991
E-mail: michael.burger@sc.tu-damrstadt.de

Christian Bischof
FG Scientific Computing, TU Darmstadt
Tel.: +49 6151 16-71001
E-mail: christian.bischof@sc.tu-damrstadt.de

shape of a tube. The resulting tubes are called Super Carbon Nanotubes (SCNTs) ([SW13]). This process can be repeated an arbitrary number of times. Figure 1 shows a CNT. A Y-shaped junction to construct tubes of level 1 is depicted in figure 2.

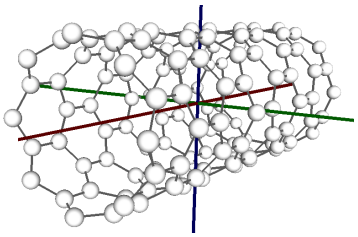

Fig. 1 SCNT of level 0

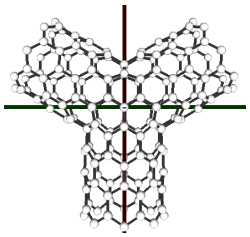

Fig. 2 Junction as basic building block of a SCNT of level 1.

The shape is defined by a quadruple of values (D_0, L_0, D_X, L_X) which determines the following properties:

- D_0: Diameter of the resulting SCNT, L_0: Length of the resulting SCNT
- D_X: Thickness of the Y-junction arms, L_X: Length of the junction arms

Each valid combination of these parameters, together with the tube level, results in a unique tube. Even for small CNTs this procedure leads very fast to structures with high atom-count. This makes it challenging to cope with such a high amount of data and to render these structures.

Our visualization tool uses OpenGL and instancing [SSKLK13, p. 128 - 139]. This allows to draw several elements with the same geometry with slight differences. The technique makes it possible to reduce the OpenGL API-calls by only using one instanced draw-call for n instances of an object. This drastically reduces the CPU load, as the host only needs to pass the mesh itself and additional information like the position of the models, their material, etc.

2 Features of our tool

Our program is able to zoom into and to arbitrarily rotate the scene. It can interactively visualize the constructed tubes within the main program by exploiting the graph properties that are suitable for instancing. Compared to a data export to a general purpose visualization tool, this saves time and memory.

2.1 Auto-performance-tuning by adaptive meshes

The graph is rendered in a two step approach. First, the connections between the nodes are drawn as GL_LINES, a primitive within OpenGL. GL_LINES are

determined by their end points and the graphics driver and card are responsible for rasterizing the line. The second step adds the atoms. These are represented as spheres, modeled by a triangle-mesh. To be able to run on different systems, the visualizer adapts the resolution of the triangle meshes, depending on the graphics card it is executing on. This change is transparent to the user. Our visualizer monitors the frame rate and automatically reduces the details if it is too slow for a predefined time interval. The current version includes five levels of detail for the spheres, plus a sixth level that draws no spheres at all.

Example screenshots for this procedure are shown in figure 3 and figure 4. Figure 3 shows a scene with the highest resolution, e.g. level 5, while figure 4 represents the same scene with detail-level 1.

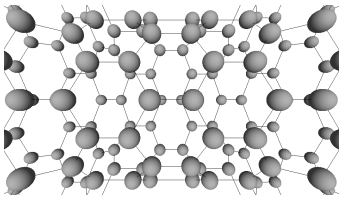

Fig. 3 Full detail

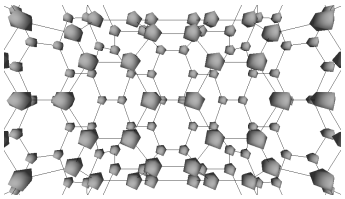

Fig. 4 Reduced detail

The visualizer is configured to target about 12 frames per second (fps). From experiments we noticed that this rate is sufficient for practical use.

2.2 GLSL shading programs

The model-transformations are applied within our GLSL-vertex-shader programs by the GPU. This moves work load from the CPU to the graphics card.

Lighting is also done by our shading programs. We developed two different shaders for the two distinct components of the model. The one for spheres is based on Phong-shading-model. The lines are drawn in a predefined color.

3 Test Systems and Results

The tests show that our visualization software is able to render all structures that are of relevance for us. For the measurements we used a desktop system equipped with an Intel Xeon 1230v2 processor, 16 GB of DDR3 RAM and a AMD R280x graphics card. For comparison, a laptop with an Intel i7-3520M, integrated HD4000 graphics and 8 GB of DDR3 main memory is employed.

The visualizer is able to render a $(16, 16, 1, 2)$ tube of level 2 on a R280x with 24 fps with detail level 1 and still 11 fps with detail level 2. The HD4000 is able to reach a frame rate > 12 when falling back to detail-level 0. In that case the scene is rendered with about 20 fps. This scene consists of 50 million triangles for the spheres in detail level 1. This value rises to about 150 million

triangles for the detail level 2. This tube is also one of the most complex structures that our software package for constructing SCNTs can cope with at the moment. Table 1 summarizes the results for two smaller tubes.

level 2 $(6,6,1,3)$ **tube**, 1179648 atoms						
	L5	**L4**	**L3**	**L2**	**L1**	**L0**
R280x	1.99	5.61	14.97	27.07	63.76	284.35
HD4000	0.45	1.33	3.51	5.62	11.19	23.04
level 1 $(10,8,1,11)$ **tube**, 81920 atoms						
	L5	**L4**	**L3**	**L2**	**L1**	**L0**
R280x	28.57	80.21	211.53	380.66	868.03	3336.66
HD4000	6.48	18.15	45.01	66.13	113.38	177.91

Table 1 Summary of the achieved frames per second for a two different tubes.

As can be seen in table 1, the R280x allows the use of detail-level 3 for the level 2 $(6,6,1,3)$ tube case. For HD4000, the visualizer chooses level 1. It can be noticed that the scaling between the distinct level of details of the R280x is much better than the one of the HD4000. This can be explained with the performance for rasterizing lines. Considering the change from level 1 to 0, the rendering speed of the R280x increases by the factor of 3 when completely disabling sphere-rendering, the frame rate of the HD4000 only doubles.

Another test is the $(10,8,1,11)$ level 1 tube. As figure 5 shows, it has the property that some parts of it are very dense (the „dark lines“), while the others contain no atoms (the rest). Figure 6 depicts such a „dark line“ in detail.

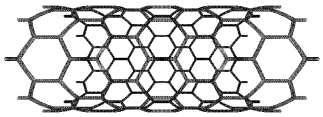

Fig. 5 Level 1 tube with longer arms at the junctions

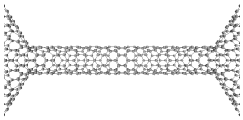

Fig. 6 Single line within the tube

The R280x has no problems in visualizing the tube at all. The fact of higher line-rendering speed of the R280x is confirmed. While the performance when neglecting spheres triples for the R280x, the HD4000 increases by a factor of 1.57. The overlaying lines in the dense parts seem to be the reason.

References

[SSKLK13] Shreiner, Dave ; Sellers, Graham ; Kessenich, John ; Licea-Kane, Bill: *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.3*. 8th. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 2013. – ISBN 978–0–321–77303–6

[SW13] Schroeppel, Christian ; Wackerfuss, Jens: Constructing meshes based on hierarchically symmetric graphs. In: *Internal Report, TU Darmstadt* 0 (2013)