Visualization of large meshes and solutions from numerical simulations with ViZiR 4

Matthieu Maunoury, Adrien Loseille

Inria Saclay
GAMMA (Automatic Mesh Generation and Adaptation Methods) Team

Journée Visu 2021
June 08, 2021
**Introduction**

**Numerical simulations**: predict the behavior of physical phenomena without using prototypes or experimentations.

**Applications**: Computational Fluid Dynamic, acoustics, energy, electromagnetism or medical modeling...

Visualization tools are necessary:
- inspect the CAD model
- check the validity and quality of the meshes
- display the numerical solutions computed
- analyze the potential problems on meshes and solutions
- validate algorithms
Many visualization softwares (e.g. Gmsh, Medit, ParaView, Tecplot, VisIt, Vizir Legacy) to analyze numerical results:

- Based on **linear** primitives as imposed by the commonly-used baseline graphic pipeline.
- Many interesting **plugins** and **tools** to help the analyses.

Some limitations:

- **Interactivity** might be missing (time to open files and render meshes and solutions).
- Lack of tools to **manipulate efficiently** these meshes.
- High-order meshes are **generally not handled**.
- High-order solutions: **visualization error** due to **low order remeshing**.
### Interactivity bottleneck: CPU Times comparisons

<table>
<thead>
<tr>
<th># vertices</th>
<th># triangles</th>
<th># tetrahedra</th>
<th>ParaView (s)</th>
<th>ViZiR 4 (s)</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 342 310</td>
<td>446 158</td>
<td>7 370 829</td>
<td>14.9</td>
<td>0.76</td>
<td>19.6</td>
</tr>
<tr>
<td>2 699 131</td>
<td>802 316</td>
<td>14 968 807</td>
<td>32.8</td>
<td>1.20</td>
<td>27.3</td>
</tr>
<tr>
<td>5 415 482</td>
<td>1 285 472</td>
<td>30 541 700</td>
<td>71.4</td>
<td>1.80</td>
<td>39.7</td>
</tr>
<tr>
<td>10 784 310</td>
<td>2 080 672</td>
<td>61 563 158</td>
<td>155.1</td>
<td>3.12</td>
<td>49.7</td>
</tr>
<tr>
<td>21 695 268</td>
<td>3 614 018</td>
<td>124 736 423</td>
<td>333.6</td>
<td>7.78</td>
<td>42.9</td>
</tr>
<tr>
<td>43 380 172</td>
<td>6 275 672</td>
<td>250 898 971</td>
<td>980.2</td>
<td>16.05</td>
<td>61.1</td>
</tr>
<tr>
<td>3 084 324</td>
<td>6 166 689</td>
<td>0</td>
<td>11.2</td>
<td>2.25</td>
<td>5.0</td>
</tr>
</tbody>
</table>

**Table 1:** Comparison of total rendering CPU time (s) including files (mesh and solution) opening.

<table>
<thead>
<tr>
<th># vertices</th>
<th># triangles</th>
<th># tetrahedra</th>
<th>ParaView (s)</th>
<th>ViZiR 4 (s)</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 342 310</td>
<td>446 158</td>
<td>7 370 829</td>
<td>1.6</td>
<td>0.18</td>
<td>8.9</td>
</tr>
<tr>
<td>2 699 131</td>
<td>802 316</td>
<td>14 968 807</td>
<td>3.1</td>
<td>0.48</td>
<td>6.5</td>
</tr>
<tr>
<td>5 415 482</td>
<td>1 285 472</td>
<td>30 541 700</td>
<td>6.4</td>
<td>0.93</td>
<td>6.9</td>
</tr>
<tr>
<td>10 784 310</td>
<td>2 080 672</td>
<td>61 563 158</td>
<td>13.9</td>
<td>1.99</td>
<td>7.0</td>
</tr>
<tr>
<td>21 695 268</td>
<td>3 614 018</td>
<td>124 736 423</td>
<td>28.6</td>
<td>4.80</td>
<td>6.0</td>
</tr>
<tr>
<td>43 380 172</td>
<td>6 275 672</td>
<td>250 898 971</td>
<td>91.3</td>
<td>10.04</td>
<td>9.1</td>
</tr>
</tbody>
</table>

**Table 2:** Comparison of CPU time (s) to generate cut plane (clip).
Interactivity bottleneck: CPU Times comparisons

<table>
<thead>
<tr>
<th>Case</th>
<th># vertices</th>
<th># triangles</th>
<th>ParaView (s)</th>
<th>ViZiR 4 (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>640K</td>
<td>217 000</td>
<td>433 653</td>
<td>7.2</td>
<td>0.01</td>
</tr>
<tr>
<td>1280K</td>
<td>392 257</td>
<td>783 947</td>
<td>13.9</td>
<td>0.01</td>
</tr>
<tr>
<td>2560K</td>
<td>628 223</td>
<td>1 255 604</td>
<td>24.6</td>
<td>0.01</td>
</tr>
<tr>
<td>5120K</td>
<td>1 018 135</td>
<td>2 035 092</td>
<td>39.2</td>
<td>0.01</td>
</tr>
<tr>
<td>10240K</td>
<td>1 772 712</td>
<td>3 543 955</td>
<td>63.1</td>
<td>0.01</td>
</tr>
<tr>
<td>20480K</td>
<td>3 084 324</td>
<td>6 166 689</td>
<td>109.3</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Table 3: Comparison of CPU time (s) to render isolines (contours) for different meshes composed only of triangles.

Some issues ViZiR 4 tries to answer:

- **Interactivity** (i.e. fast) to develop meshes algorithms.
- Display with **high fidelity** the computed numerical solution.
- Handle **high-order** meshes and solutions.
Main features of ViZiR 4:

- **Light, simple** and **interactive** visualization software.
- **Surface** and **volume** (tetrahedra, pyramids, prisms, hexahedra) meshes.
- **Pixel exact** rendering of **high-order** solutions on straight elements.
- **Almost pixel exact** rendering on curved elements (high-order meshes).
- **Post-processing tools**, such as picking, isolines, clipping, capping.
ViZiR 4 is based on OpenGL Shading Language (GLSL).

![Diagram](image-url)

**Figure 1:** Shaders used for the OpenGL graphic pipeline.
Fast I/O

Input and output handle by the libMeshb library (Loic Maréchal, Inria).

Mesh of Lucy:
- 14 millions vertices and 28 millions triangles (642 Mb).
- Mesh opened in less than 1.5 seconds.
- Rendered in 7.5 seconds (total time) on a laptop.
Pixel exact rendering on flat elements

- For each pixel, Fragment shader determines the appropriate color.
- It certifies a faithful and interactive depiction (up to degree 10 polynomial function).
- High order solutions are natively handled by ViZiR 4 on surface and volume (tetrahedra, pyramids, prisms, hexahedra) meshes.

Figure 2: High-order (degree $Q^6$) solution of a wave propagation problem. Right: zoom of the solution on 4 hexahedra.
For complex geometry, curved elements perform a better approximation of the geometry.

Tessellation shaders: creation of sub-elements on the fly by the GPU.

Figure 3: Rendering of high-order mesh (left) and its tessellation constructed by the GPU (right).
Tessellation on GPU for high-order elements

Figure 4: Comparison of meshes of degree 1 (left) and 2 (right) for the same number of elements.
Isolines (instant rendering) with possibly filled solution rendering

Figure 5: Examples of isolines rendering.
Clip Planes

- Difficult to navigate in 3 dimensional meshes.
- For this reason, interesting to use clip planes where all volume elements belonging to a plane are displayed.

Figure 6: Examples of cut planes.
Scripting tools to easily generate images or go over large set of meshes / solutions.
Numerical results

Figure 7: Friction Coefficient (Cf) solutions for different adapted meshes.
ViZiR 4 web site: https://pyamg.saclay.inria.fr/vizir4.html with executables (Mac, Linux, Windows), samples (meshes and solutions files) and user guide.
Thank you for your attention
Movie Mode: vizir4 -movie. An example of file vizir.movie

mesh1.meshb sol1.solb
mesh2.meshb sol2.solb
mesh3.meshb sol3.solb
mesh4.meshb sol4.solb
mesh5.meshb sol5.solb
mesh6.meshb sol6.solb
Figure 8: Example of jacobian rendering: exact (left) and minimal (right).
Figure 9: Use of filters: all elements in light blue appear as they belong to the range of the filter (for a given criterion).