ViZiR 4: Manage state

GAMMA Team, Inria

July 31, 2023

Motivation

What are state files ?

State files contain rendering options and information on the view, the clip plane, the mesh or the solution.

Goal: Quickly save or load a view with rendering options

The goal of this document is to present **how to manage state** in `ViZiR 4`, explain how each keyword works, present **the state sequences mode** and `pyviz`**: the state and sequence controller** for `ViZiR 4`.

Some interesting links for the following:

- `ViZiR 4` web site:
  `https://pyamg.saclay.inria.fr/vizir4.html`
- `ViZiR 4` user guide:
  `https://pyamg.saclay.inria.fr/download/vizir/vizir4_user_guide.pdf`
- A directory containing `pyviz` package and examples:
  `https://pyamg.saclay.inria.fr/download/vizir/pyviz`

State files

The keywords in state files can be decomposed in categories:

- View
- Clip plane
- Mesh
- Solution
- Other

In `ViZiR 4`, these parameters can be changed in the menus: State Manager, Modify View and Cut Plane Visualization.

State: view parameters

These parameters can be changed in the Modify View Menu:

- **Center** (3 reals): center of the view in the real physical world
- **Rotation** (4 reals): rotation of the view
- **Translation** (3 reals): translation of the view in the window world
- **Fovy** (1 real): zoom

It is not advise to manually change the parameters **Center**, **Rotation** and **Translation**. Otherwise, it is possible to pick an element and use V to center the view to it. Furthermore, the button *Set View* allow to modify the view alongside an axis.

State: clip plane parameters

These parameters can be changed in the Cut Plane and State Manager menus:

- **CutPlaneEquation** (4 reals): a, b, c, d from equation $ax + by + cz + d = 0$
- **UseCutPlane** (boolean): activate cut/capping plane mode
- **ShowCut** (boolean): display the edges of the cut plane
- **Capping** (boolean): activate capping

Manually, F1 is used to (de)activate cut plane mode. F2 is used to modify the cut plane and to update it. C is use to (de)activate capping. The boolean **UseCutPlane** allow to activate (i.e. use) the clip plane mode. The boolean **ShowCut** can be set to 0 to hide the edges of the cut plane.

State: mesh parameters

These parameters can be changed in the State Manager menu:

- **LineOn** (boolean): display the mesh edges
- **TessOn** (boolean): display the tessellation
- **TessLevel** (1 integer): tessellation level for high-order meshes
- **EdgeOn** (boolean): display the edges

The mesh can be shown with l while the tessellation is shown with L. The tessellation level can be increased with t and decreased with T. The edges can be shown with E.

State: solution parameters

These parameters can be changed in the State Manager menu:

- **SolOn** (1 integer): display solution: 0 off, 1 exact, 2 filled
- **IsoOn** (boolean): display isolines: 0 off, 1 colors, 2 black
- **PalOn** (boolean): display the palette
- **UsePalette** (boolean): use the given palette
- **Palette** (5 reals): values of the palette (used if UsePalette is true)
- **CurSol** (1 integer): solution type: 0 no sol., 1 sol. at vertices, 2 sol. at elements
- **ithSol** (1 integer): index of the field displayed (1 is the first)
- **IsoSiz** (1 real): size of the isolines

Three modes are defined for the keywords **SolOn** and**IsoOn**. **PalOn** is used to hide the palette which usually lies in the top of the window. **UsePalette** can be set to 1 to use the 5 values given by **Palette**. **CurSol** is useful to swap between solution at vertices and solution at elements and can be changed with n.

State: other parameters

- **WindowSize** (2 integer): size of the window
- **TextOn** (boolean): display the text
- **LineSurfOn** (boolean): show the mesh lines on surfaces
- **TransOn** (boolean): activate transparency
- **ShaMod** (1 integer): change the shading rendering: 0 usual, 1 Phong, 2 toon, 3 no
- **RenMod** (1 integer): change the rendering: 17 off, 19 back, 21 ref, 23 grey
- **WireSiz** (1 real): size of the lines
- **UseBoundingBox** (boolean): use the given bounding box
- **BoundingBox** (6 reals): xmin, xmax, ymin, ymax, zmin, zmax (used if UseBoundingBox is true)
- **NumberOfHiddenRefObjects** (1 integer + list): number of hidden surface + list of types and references

NumberOfHiddenRefObjects gives the list of type and number of reference elements that are hidden.

State sequences mode: automatically load state files and generate images

It can be launched from the menu or with

```
vizir4 -seq -in xxx.mesh[b] (-sol xxx.sol[b] -nameseq xxx)
```

- -in is required to define the input mesh files
- -sol is optional to define the input solution files
- -nameseq is optional to define the state sequences files (otherwise vizir.seq)

A .seq file is required to give the name of all the state files.
For each line, the name of a state file is given and optionally a name of a png file (if only a state file is given, the png file will take the name of the state file).

```
view1.state     top.png
view2.state     bottom.png
view3.state     zoom.png
```

Code Listing 1: An example `vizir.seq` file

It generates 3 png from 3 state files.

An example of use is shown in `pyviz` section.

pyviz: the state and sequence controller for ViZiR 4

Goal of pyviz: easy generation of state files from one and sequences files containing state files to quickly script and render view or generate pictures.

A directory **pyviz** contains the package (wheel) and an example and is available here: `https://pyamg.saclay.inria.fr/download/vizir/pyviz`

python3 is needed to install the package (wheel):

```
python3 -m pip install pyviz-4-py3-none-any.whl
```

The keywords of pyviz

There are 3 types of keywords:

- Boolean keywords which can be modified with **enable** and **disable**: ['UseCutPlane', 'Capping', 'ShowCut', 'PalOn', 'TextOn', 'LineOn', 'TessOn', 'LineSurfOn', 'TransOn', 'UsePalette', 'EdgeOn']
- Integers and reals keywords can be changed with **set**. The number of parameters depends of the keywords. Here are the size: ['WindowSize': 2, 'Center': 3, 'Rotation': 4, 'Translation': 3, 'CutPlaneEquation': 4, 'Fovy': 1, 'SolOn' : 1, 'IsoOn' : 1, 'CurSol': 1, 'ithSol': 1, 'IsoSiz': 1, 'TessLevel': 1, 'ShaMod': 1, 'RenMod': 1, 'WireSiz': 1, 'Palette': 5]
- The keyword **NumberOfHiddenRefObjects** is specific and actually cannot be modified with pyviz

pyviz: an example

example.zip contains an example to show how it works:
`https://pyamg.saclay.inria.fr/download/vizir/pyviz/example.zip`

Input: a mesh file, a solution file, a state file and the script gendata.py

To launch the script in the example directory:

```
python3 gendata.py
```

Output: It generates two state files vizir-solon.state and vizir-isoon.state and a sequence file vizir.seq containing the state files to load and the name of the images to generate.

```
vizir.state test1.png
vizir-solon.state test2.png
vizir-isoon.state test3.png
```

Code Listing 2: The file `vizir.seq` obtained

The file gendata.py:

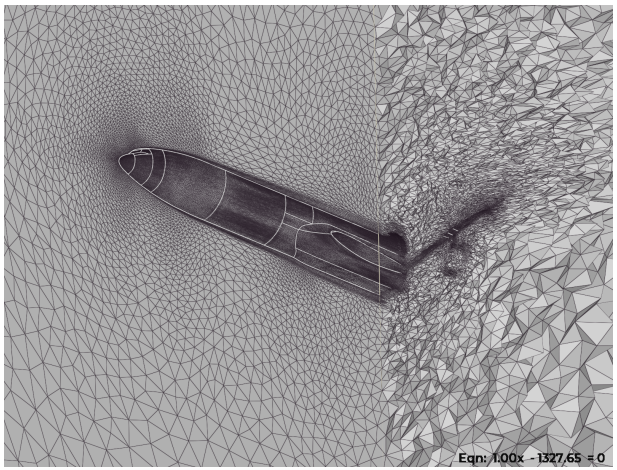
```
import pyviz

# declare a sequence list and state
s = pyviz.Sequence()
d = pyviz.State()

# read existing state file
d.read("vizir.state")

# personalize you .state here
d.set('IsoOn', 1)
d.disable('TextOn')
d.disable('LineOn')
d.set('SolOn', 1)
# write it
d.write('vizir-solon.state')

# add another state
d.set('IsoOn', 1)
d.enable('Capping')
d.write('vizir-isoon.state')

# add in the sequence file
s.add('vizir.state', 'test1.png')
s.add('vizir-solon.state', 'test2.png'
                      )
s.add('vizir-isoon.state', 'test3.png'
                      )
```

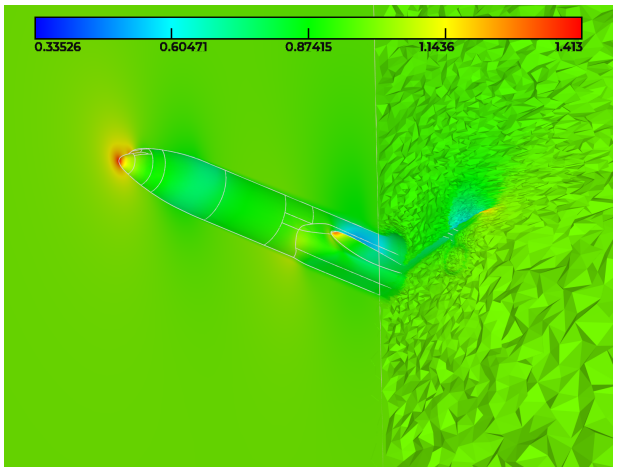
Images obtained with pyviz and ViZiR 4

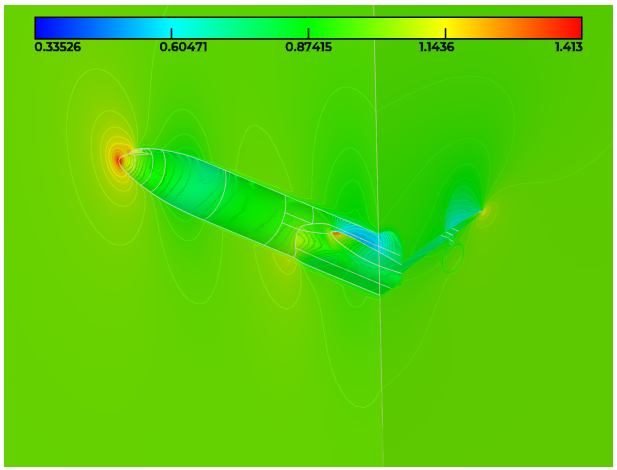
Finally, ViZiR 4 in the sequence mode can be used to generate the images:

```
vizir4 -seq  -in file.meshb
```

It creates the 3 images from the 3 state files:







Thank you for your attention

Contacts:
Adrien Loseille, Adrien.Loseille@inria.fr
Matthieu Maunoury, Matthieu.Maunoury@inria.fr