STC#5 Report

Date: 19-21/06/2018
Location: IHU Strasbourg
Please find also enclosed the slides of the all presentations.

Attendees

- Hugo Talbot, SOFA Consortium
- Guillaume Paran, SOFA Consortium
- Damien Marchal, Inria Lille
- Olivier Goury, Inria Lille
- Christian Duriez, Inria Lille (remote)
- Bruno Carrez, Inria Lille (remote)
- Stéphane Cotin, Inria Strasbourg
- Hadrien Courtecuisse, Inria Strasbourg
- Frédérick Roy, Inria Strasbourg
- Rémi Bessard, Inria Strasbourg
- Jean-Nicolas Brunet, Inria Strasbourg
- Antoine Petit, Inria Strasbourg
- Didier Weckmann, IHU/Ircad
- Erik Pernod, InfinyTech3D
- Christian Diaz, SimDesign EAFIT (remote)
- Jorg Peters, Uni. Florida (remote)
Synthetic agenda of STC#5

Day 1 - Tuesday 19th June (afternoon)

14:15 - SOFA report
Consortium activity
Memberships, communication, events
Tech work
SOFA activity
Major PRs, important discussions
v18.06 status: unfinished tasks (for the sprint) ?
v18.12 roadmap proposal (targeting next release)

16:00 - Demos, features, scientific results with SOFA

17:30 - Round the table: what's up (plugins, features, ...)

18:00 - Reminder v18.12 roadmap

Day 2 - Wednesday 20th June

ROOM: "Debrief 2", 2nd floor IHU Strasbourg

09:00 - NG report
Approach, work done, used tools (scripts)
Next steps, merge with v18.12

10:30 - Defining v18.12 roadmap
Expected technical evolutions, among the possible topics
Objective: determine what could be accomplished for v18.12 + DOC = define subtasks

16:30 - Coding sprint
Browse GitHub issues labeled “STC#5”

Day 3 - Thursday 21st June

09:00 - Coding sprint

15:00 - Validation of technical v18.12 roadmap
Reports and updates

SOFA report
You will find all information regarding the report on SOFA, its activity and the consortium activity within the STC#5 slides.

Demos, features, scientific results with SOFA

Sparse matrices - Christian Duriez
In SOFA, matrices are handled using different format not always for good reasons. Further to a work implying to construct a linear system from mapped matrices coming from ForceFields (see Olivier Goury’s work below). Non-linear mappings would also require some update on the matrix storage in SOFA.

The question of the storage of the stiffness matrix arose but storing it into the MechanicalObject does not go into the direction of a lighter / non-mechanical-only MechanicalObject.

SOFA in Unity (VR) - Erik Pernod
Erik Pernod brought his VR setup to show his latest application involving SOFA in Unity. This work has been carried out within Erik’s new company InfinyTech3D, proposing consulting and services around SOFA. This recent work is still in active development and ensures nice achievement between SOFA and Unity3D.

Check out his demo: https://www.youtube.com/watch?v=w89t7Z6sG6g

OpenCV plugin - Bruno Marques
OpenCV plugin aims at interfacing simulation in SOFA, computer vision algorithms and input images or video streams to achieve new applications like VR applications or animation. Developed in the Mimesis team, this plugin is a framework itself and can be extended with any new algorithms (image analysis, processing, shape recognition etc.), methods or external libraries regarding computer vision applications.

Model Reduction - Olivier Goury
Arising from a research project conducted in the Inria Defrost team, a new simulation method will soon be made available in SOFA : simulation based on reduced models. Using all the power of mappings in SOFA, Olivier Goury introduced the method of model hyperreduction allowing to significantly decrease the number of degrees of freedom considered for the physics simulation. Hyperreduction therefore dramatically reduces the computational time while preserving high numerical accuracy. The user can thus benefit from a reduced and interactive version of the simulation without significant loss of accuracy. Coming soon in SOFA!

Reworking Data Update - Bruno Marques
During the last months, the need of a uniform process for data updates in SOFA components has been noticed. Data are more and more manipulated, modified on the fly (increasing use of Python). This topic is directly linked with the dependency graph, which is not yet visible and available in SOFA. Bruno Marques explained that the Data update API is heavy, a bit complex (debug) and not
flexible enough. Moreover, the process of data update is not well documented and not used in a uniform way across all components. We therefore need to bring a clear documentation as well as developer guidelines. A clean of direct links (getContext) should be avoided/removed as much as possible, thus avoiding implicit dependencies. A visualization of this graph would be of great help, maybe included into the SofaQtGui.

**Improving SofaPython - Jean-Nicolas Brunet**
Several approaches are possible to improve SofaPython architecture. The idea is to modularize and simplify the plugin. First version has been done to support an external python interpreter. It permits to create a python version of runSofa. Next version will propose a complete “sofa” package importable in any python interpreter.

**Round the table**
After a quick round the table, we noticed that the subjects with most interests were “Improving SofaPython”, “Reworking Data update” and “SofaQtQuick”.

**SOFA-NG report**
You will find all information regarding the SOFA-NG report within the STC#5 slides. Main information point for SOFA-NG task: [Roadmap GitHub issue](#)
Defining v18.12 roadmap

Reworking Data update

Context
There is few documentation about how Data updates are handled in SOFA. Two distinct graphs coexist: the Scene graph (visualized in runSofa) and the Data Dependency Graph (internal). The former specifies the architecture of a simulation from a Component point of view while the later specifies the architecture of a simulation from a Data point of view. The idea of this project is to simplify both graphs into something clearer, better documented and less error prone.

Discussions
The discussions went about “what to do?”, “how to do?”, “where to start?”. It was finally agreed that, as a first step, it would be nice to start by fixing all the bad links in SOFA (getContext way).

“Data update” v18.12 roadmap
1. Write documentation on update process and expected behavior (DataEngine)
2. Fix bad links (getContext way) one by one when you encounter them, use addLink instead
3. Create a DataGraph widget (see SofaQtQuick) for DDG visualization

More information
Main contact: Bruno Marques
Gitter chat room: https://gitter.im/sofa-framework/data-update

SofaQtQuick

Context
SofaQtQuick is a project started last year aiming to replace runSofa by a brand new QtQuick based (QML) GUI. The project is not yet integrated into SOFA but can already be used as runSofa2. New widgets can be added easily.

Discussions
A lot of things can be done in this project. The main issue during the discussion was to isolate things that should be done for v18.12. This includes things that have to be done (needed) and things that project actors want to do.
First, it has been agreed that SofaQtQuick must be used by the developers as default GUI. It is the best way to boost the project. A QuickStart guide must be written to explain how to set everything up.
Then, Damien wants to focus on new features like scene reorganization, field edition, python live coding, … Bruno and Stefan are interested in creating a DataGraph widget.
Erik suggested to get inspired from “Unreal Engine Blueprint” like Data graph: example1, example2.
“SofaQtQuick” v18.12 roadmap
1. Write a QuickStart guide
2. Use SofaQtQuick as much as possible
3. Create a DataGraph widget
4. Refactor all the ListModel to separate the filtering from the data source
5. runSofa PickHandler as specific scene component
6. Optimize code for QML - binding slows down the application
7. Implement rendering code using QtQuick or webGL

More information
Main contact: Damien Marchal
GitHub repository: http://github.com/sofa-framework/SofaQtQuick
Gitter chat room: https://gitter.im/sofa-framework/SofaQtQuick

Improving SofaPython

Context
As presented during demos, the goal of this project is to simplify SofaPython architecture and permit its usage as python package.

Discussions
No further discussion were led. Jean-Nicolas made it clear about what were the next steps: first support an external python interpreter, then propose a complete “sofa” package.

“Improving SofaPython” v18.12 roadmap
1. Keep it simple: create a PoC with python env external to SOFA

More information
Main contact: Jean-Nicolas Brunet
Coding sprint

As usual, a coding sprint was organised to finalize some pending tasks. The tasks were listed as GitHub issues and can still be accessed.

<table>
<thead>
<tr>
<th>Proposed task</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change all ifndef NDEBUG in topoloy to define macro #698</td>
<td>Done</td>
</tr>
<tr>
<td>Remove deprecated define HAVE_NEW_HEXAXA #697</td>
<td>Done</td>
</tr>
<tr>
<td>Remove XBOX specific code #695</td>
<td>To do</td>
</tr>
<tr>
<td>Remove PS3 macro #694</td>
<td>Done</td>
</tr>
<tr>
<td>Improve packaging methods #690</td>
<td>To do</td>
</tr>
<tr>
<td>Update deprecated components list #689</td>
<td>To do</td>
</tr>
<tr>
<td>Remove NO_EXTERN_TEMPLATE #682</td>
<td>Done</td>
</tr>
<tr>
<td>Remove commented code #679</td>
<td>Done</td>
</tr>
<tr>
<td>Warn about Alias usage #677</td>
<td>Done</td>
</tr>
<tr>
<td>Homogeneize the use of drawTools() in SOFA #653</td>
<td>Done</td>
</tr>
<tr>
<td>Fix Tutorial applications #445</td>
<td>To do</td>
</tr>
<tr>
<td>Merge FixedConstraint and PartialFixedConstraint #440</td>
<td>To do</td>
</tr>
<tr>
<td>Remove duplicated code for data initialization in component #274</td>
<td>To do</td>
</tr>
<tr>
<td>Ensure coherency in ForceFields #232</td>
<td>To do</td>
</tr>
<tr>
<td>Creating tests on Topological operations #43</td>
<td>To do</td>
</tr>
</tbody>
</table>
Defining v18.12 roadmap - Verbatim

Data and update process

- documentation on update process and expected behavior (DataEngine)
- fix bad links (getContext way) one by one when you encounter them, explicit addLink
- get a first visualization (SofaQtGui) of the data graph

inherit from DDGNode ? no..
merge dataHandler (call back for each data regarding topo change) / dataTracker

relationship with multithreading

what do we do to clean update of data in core physical class?

initialize(FromContext)
update
state initialization from the leaves: always consistent?

graph representation should be at the will of the user
but the key graph, remains the dependency graph (eg python scripts) → constraint to go through a graph to get your components, no way to create free-from-graph components

TopologyMappings → DataEngine

Events in SOFA : some (BeginAnimationStep) are events whereas they look like visitors?!
How to handle static data? → readOnly (to improve)

SofaQtQuick

http://github.com/sofa-framework/SofaQtQuick

should it be shipped in SOFA ?
not yet, but we should all agree that all of us will use it as default GUI

The “what I want” list (please fill stuff):

- Damien: an editor to make sofa scene (add node, component, reorganize the scene, edit field, save, investigate problem in the simulation, live coding feature in python, glsl, qml, controlling the amount of data field showed, per component data view, per scene, support for project(directory) views and UI plugins, export scene as executable, plugin store)
- Bruno & Stefan: Graph/Node base vue to see the relationship/dependency graph between Data and the other is the relationship between component through the simulation point of view (the mapping).

Stuff to do:

- add the agreed functionality
- refactor all the ListModel to separate the filtering from the data source
- runSofa pickHandler features should be implemented directly as specific scene component
- optimize code for QML binding slows down the application
- implement rendering code using QtQuick or webGL (to remove rendering in the browser)

→ for v18.12: focus on features

Stuff that could be done in parallel
- QuickStart guide (doc)
- Do we use the same API on SofaPython and SofaJavaScript to ease transition?
- QML optimization (QtCreator debugger/profiler)
- reproducing the Modeler behaviour
- stay in a “messy” mode - no guidelines yet - not enough expertise yet

- python?
- create a DataGraph widget
- “Unreal Engine Blueprint” like Data graph: example1 example2?

SofaPython
< this afternoon >
1st step keep it simple: compatibility poc with python env external to SOFA
keep existing binding
with NG → update bindings to make it package specific?