STC#6 Report

Date: 07/11/2018 - 09/11/2018
Location: IHU Strasbourg

Please find also enclosed the slides of the all presentations.

Attendees

- Julien Abinahed, Hamad Medical Corporation
- Anthonin Bernardin, Inria Rennes
- Jean-Nicolas Brunet, Inria Strasbourg
- Stéphane Cotin, Inria Strasbourg
- Hadrien Courtecuisse, Inria Strasbourg
- Christian Duriez, Inria Lille
- Mike Filippov, MarionSurgical
- Thierry Gaugry, Inria Rennes
- Olivier Goury, Inria Lille
- Nick Hockings, CSIRO
- Camille Krewcun, University of Clermont-Ferrand
- Damien Marchal, Inria Lille
- Bruno Marques, Inria Strasbourg
- Guillaume Parant, SOFA Consortium
- Erik Pernod, InfinyTech3D
- Jorg Peters, University of Florida
- Frédérick Roy, Inria Strasbourg
- Federico Spadoni, InfinyTech3D
- Eleonora Tagliabue, University of Verona
- Hugo Talbot, SOFA Consortium
- Felix Vanneste, Inria Lille
Synthetic agenda of STC#6

Day 1 - Wednesday 7th November (afternoon only)

1. SOFA report
   - Consortium activity
   - SOFA activity

2. Completed/on-going work with SOFA
   - Round table

3. Specific project updates
   - Damien: Python bindings
   - Guillaume: SOFA NG
   - Bruno M.: Data updates in SOFA
   - Erik: assimp, topologies, Unity integration
   - Federico: multithreading

Day 2 - Thursday 8th November

1. From v18.12 to v19.06
   - Feedback on v18.12 definition and management
   - Potential tasks for v19.06

2. Defining v19.06 roadmap
   - Selecting tasks for v19.06
   - Detailing subtasks doable in 6 months

3. Validating v19.06 roadmap

4. Starting coding sprint

Day 3 - Friday 9th November

- Coding sprint all day -
Reports and updates

SOFA report
You will find all information regarding the report on SOFA, its activity and the consortium activity within the [STC#6 slides](#).

Roundtable
Here is a quick overview of each participant current work and needs with SOFA.

- **Erik**: loader (assimp), topologies.
- **Felix & Olivier**: MechanicalMatrixMapper (see previous report).
- **Camille**: stent procedures, BeamFEM involving plasticity.
- **Eleonora**: interested in SOFA validation.
- **Jean Nicolas**: Immerse Boundaries Method and meshless.
- **MarionSurgical**: using BeamAdapter and Skin Puncture plugin, interested in asynchronous haptics and Multithreading approaches.
- **Federico**: multithreading AnimationLoop (next step: work on constraint).
- **Christian**: multiple instances of SOFA (with batch mode), implementation of ForceFields/Constraints that involves derivative of constraints matrix H, homogeneous to forces, work on formal math equation.
- **Thierry**: work on HeadlessRecorder, SofaPython.
- **Nick**: running computation on clusters, CGAL work, implement a plugin for SPH using OpenCL, Clones with shared memory.
- **Antonin**: specific contact model, contact solver for suction cup: flexible AnimationLoop, script for bootstrapping SOFA classes.
- **Olivier**: Model order reduction work, reducing shell models, discussion with Oticon Medical patient-specific needle insertion in cochlea, meso-structure simulation.
- **Bruno**: open-source plugin wrapping OpenCV functionalities.
- **Damien**: refactoring SofaPython plugin, making implicit modeling into SOFA for parametric design of shapes and easy meshing, asynchronous scenes.
- **Jorg**: updating to Geomagic, interest in hexa-based MOR.
- **Julien**: interest in cutting.
- **Hadrien**: needs improvement on computation time for controlling robot, GPU computing, ready to open-source many plugins (cutting plugin from Cardiff, scene creation with data dependencies), addKToMatrix.
- **Youness**: run several instances of SOFA.
Specific project updates

All presentations and slides are publicly available:
https://drive.google.com/drive/folders/1u1i65NFs4w4oOUyIrAgM0oDhRRVZKEUI

Python bindings - Damien Marchal

Damien showed us the principal problems of the current SofaPython implementation:

- usability issues
- interpreter is embedded in the SOFA runtime
- cannot manipulate SOFA from external python script
- python 2.7 is too old and its support stops in 2019
- performance issue
- bindings are SLOW (500x slower than clean python code)

To solve these problems, a lot of test implementations has been done around bindings and python3 upgrade. Significant experience has been gathered:

- Ctypes API
- Cython: greatly improve the existing but implies to be expert in idiomatic python and C++
- PyBind11: a more high-level solution with bindings in pure C++ with still high performance gain (but less that others)

Chosen solution: PyBind11.

Next steps for this project:

- Switch to Python 3
- Document bindings with docstring
- Implement bindings

SOFA NG - Guillaume Paran

Guillaume presented briefly the news about NG project. The objective of the project did not change: make SOFA 100% modular by refactoring current modules that are very monolithic at the moment.

Unfortunately, due to very low manpower on this project, there were no much changes:

- sofa2ng (refactoring scripts) has been improved
- Sofa.Helper is still being refactored

We hope that NG will benefit from more contributors in the next months.

Data updates - Bruno Marques

Bruno worked a lot on Data update mechanism lately with Damien and Erik. He took some time to understand and locate the unclear/wrong/badly-written parts.

Here are the major problems he pointed out:

- Default behavior of components is to NOT automatically update its datafields and DataEngines need to be used to correctly propagate a data change.
- Error-prone implementations of DataEngines (fixed in future update).
• Very inconsistent implementation of init / reinit methods.
To simplify the DataEngines, he implemented an easy-to-derive API that hides the code related to
updating datafields and calling cleanDirty() in an attempt to reduce human errors.

Consequently, every old DataEngine-derived should:
• replace its update() method with the new delegate doUpdate()
• remove the calls to data.isDirty() and (smartly) replacing them with
  m_dataTracker.isDirty(data)
• remove the cleanDirty() calls while making sure to not break the engine's behavior

There is still a lot of work concerning Data updates, this project will stay in v19.06 roadmap.

MeshLoader - Erik Pernod
Actual implementation of loaders (one Component per input type) leads to obvious limitations:
• Duplicated code
• Several way to define loaders
• Obligation to define specific format
• Not able to handle multimeshes
• Handle a lot of format support ourselves

In order to solve these, Erik started to refactor MeshLoader code in PR #661.
The objective is to get a generic MeshLoader component usable in scenes with any of the
supported formats (currently SOFA can load obj, trian, stl, gmsh, xsp).

This project will not be part of v19.06 roadmap but still will continue during next months.

Topologies - Erik Pernod
As a reminder, there are currently two pipelines for topological changes in SOFA:
• the old one: TopologicalChange events propagation using Visitors through the scene graph
  (with handleTopologyChange)
• the “new” (2012) one: TopologicalData inside components that are linked to topology
  buffers (using Data Graph
  → not finished: a lot of components are crashing while performing topological changes.

Erik defined some tasks to rework and clean the topological changes:
• Finish the TopologicalData change pipeline
• Remove the old pipeline: topology events visitor / handleTopologyChange methods
• Create a unique MeshTopology component with an option to set it as dynamic or not
• Add tests and doc

Work has been started with #788, #855, #866. Any help will be appreciated.

Multithreading - Federico Spadoni
Federico updated us about his work on multithreading. Here are his achievements:
• TaskScheduler moved to sofa::simulation::core to allow different scheduling strategies
• Default implementation: DefaultTaskScheduler
  ○ in C++11
  ○ uses a stack of tasks (LIFO)
• FreeMotionAnimationLoop is parallelized (not completely)

Future work
• implement concurrent memory pool allocator
• work on constraint

Remarks
• OpenCL integration (for SPL) should be in our long term goals
• issue with data updates? behavior of hidden dependencies in components?
Defining v19.06 roadmap

Data updates

Context
Two distinct graphs coexist in SOFA: 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. Moreover, the Scene graph is clearly visible in SOFA while the Data Dependency Graph is not clearly available and sometimes hidden in the code. The idea of this project is to simplify both graphs into something clearer, better documented and less error prone.

Discussions
In the past few months, an analysis was conducted on the Data API. It appeared that a first step would be to get a visual representation of the Data Dependency Graph. This could be used as an assistance tool to fix Dependency issues between components (getContext). On a more focus issue, the update API in engines must be simplified. Then, we could target a clean and robust initialization for all components, not necessarily engines. To do so, a value tool would be to implement a State information for each component (existing beginning of a State API). With this state and using a approach based on DataTracker (similar to engines), the clean of init() and reinit() could be tackled.

"Data updates" v19.06 roadmap

1. Create documentation from slides (from two previous presentation)
2. Use widely the API of SimpleDataEngine
   - Set it as DataEngine (default)
   - List all components inheriting DataEngine
   - Compare with BelCurves
   - Remove confusion between the 2 different isDirty() methods:
     data.isDirty() != m_datatracker.isDirty(data)
   - Change in all Engine components: update() → doUpdate()
   - Update doc
3. Rename DataTrackerDDGNode::isDirty() to hasChanged()
4. Set private access modifier to BaseData::isDirty() & DataEngine::isDirty()
5. Data<int> “d_state” to all objects
6. Display of data graph
7. Discussion : all components should be DataEngine !?
   - mass, forcefield, solver, constraints
8. Fix bad links (getContext way) one by one when you encounter them, use addLink instead
9. Create a DataGraph widget (in SofaQtQuick) for DDG visualization

More information
Main contact: Bruno Marques
Multithreading

Context
The task on Multithreading restarted less than a year ago and is a wide challenge. The idea is to propose a multithreaded version of SOFA. It requires to focus on thread safety in the code and to build new tools and algorithms.

The multithreading plugin has thus been developed to parallelize computationally intensive tasks without modifying the architecture of SOFA. To exploit the CPU parallelism, task and scheduler design is used. That is an efficient way to scale the computation to all the CPU cores available on a machine without directly manipulating threads.

Discussions
The work started since the previous STC. A plugin is now available. This Multithreading plugin provides a task scheduler implementation and a few SOFA component design to create tasks that can be executed concurrently.

The parallelization can be performed at component level when two or more independent components can be executed in parallel using a specific multithreaded AnimationLoop. The next step would be to allow for constraint solving in parallel. A parallel version of the ConstraintSolver is therefore the first objective.

“Multithreading” v19.06 roadmap

1. Constraint Solver
   a. Generic Constraint Solver
   b. LCPConstraintSolver
2. Tasks pool allocator:
   a. Implement both a global allocator or worker thread local allocator
   b. Implement unit tests
   c. Compare both allocators performance
3. Run the collision detection and deformable object solver concurrently
4. Think how to fit and support a fast haptic thread in the TaskScheduler

More information
Main contact: Federico Spadoni
SofaPython

Context
Improving SofaPython has been in SOFA devs objectives for a long time. Multiple ways have been explored around Python3 upgrade (see the dedicated branch) and bindings improvement. The ending of Python2 support being very close (2019), we have to move forward on this project.

Discussions
From Damien’s works and presentation, it seems obvious that we have to work on a Python3 implementation of SOFA bindings. The idea is to create an independent SofaPython3 plugin giving us complete freedom on tools and architecture. PyBind11 had been collectively chosen for bindings implementation. The objective for v19.06 would be to propose a transition state by permitting to switch between Python2 and Python3. Both version being independent, the bindings will differ.

“SofaPython” v19.06 roadmap
1. CMake: cannot have sofapython- legacy and sofapython3
2. CMake: make it an external package
3. Scene creation API + loader + external python3 (easy)
   a. memory leak on PythonScriptController (current sofapython)
   b. add, create, getdata, setdata
   c. add tests and docs
   d. integrate gtest
4. Data access API (1 remaining hard question)
5. Overridable component (1 remaining hard question)
   a. potentials problems
      i. override getattr/setattr
      ii. object not yet in scene graphs
   b. explore metaclass idea

More information
Main contact: Damien Marchal
Gitter chat room: https://gitter.im/sofa-framework/SofaPython3
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>Listing all type of files loaded in SOFA [811]</td>
<td>To do</td>
</tr>
<tr>
<td>check which component is using : handleTopologicalChange() function or TopologicalData [810]</td>
<td>To do</td>
</tr>
<tr>
<td>Propagate the doUpdate() function for all engine instead of update() [809]</td>
<td>Done</td>
</tr>
<tr>
<td>Remove useless template instantiation that are not needed [806]</td>
<td>Done</td>
</tr>
<tr>
<td>Memory leak MeshSpringForceField [804]</td>
<td>Done</td>
</tr>
<tr>
<td>Remove the duplicated code between equivalent template specialization (eg Rigid3dType vs Rigid3fType) [802]</td>
<td>To do</td>
</tr>
<tr>
<td>Reactivate regression tests in CI [801]</td>
<td>Done</td>
</tr>
<tr>
<td>Usage of eigen for vector &amp; matrices (eg instead of MultiMatrix) [800]</td>
<td>To do</td>
</tr>
<tr>
<td>Uniform all threshold, epsilon or Pi usage [799]</td>
<td>To do</td>
</tr>
<tr>
<td>AffineMovementConstraint: orientation issue [795]</td>
<td>Done</td>
</tr>
<tr>
<td>BaseObject::init &quot;call super&quot; code smell [785]</td>
<td>To do</td>
</tr>
<tr>
<td>[DrawToolGL] Some draw functions still miss the calls to the internal draw functions [757]</td>
<td>To do</td>
</tr>
<tr>
<td>Use type size_t everywhere when working with size [709]</td>
<td>To do</td>
</tr>
<tr>
<td>Remove XBOX specific code [695]</td>
<td>To do</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>Fix Tutorial applications [445]</td>
<td>To do</td>
</tr>
<tr>
<td>Ensure coherency in ForceFields [232]</td>
<td>To do</td>
</tr>
</tbody>
</table>