SOFA

Simulation
Open
Framework
Architecture

An open-source solution for collaborations, prototyping and innovation in simulation

Innovation catalyst in medicine and robotics

Overview

CCFR CENTRE DE COMPÉTENCE HPC.HPDA.IA

- Discover SOFA
- Community and project governance
- Main principles of the framework
- Simulation features
- How to get started / Take home message

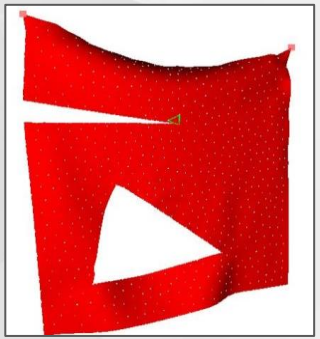
SOFA

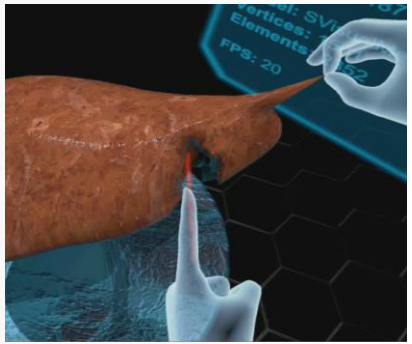
Round table

Speaker: Erik Pernod

15 years of experience in physical modeling

Dedicated to the improvement and use of SOFA as key element for innovations





Inria SOFA inventeurs du monde numérique — 2008-2011

belcurves — 2012-2016

2017

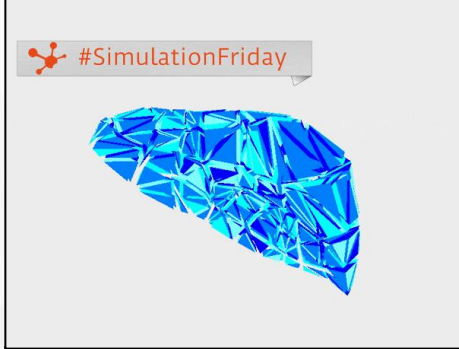
InfinyTech3D — 2018-2025

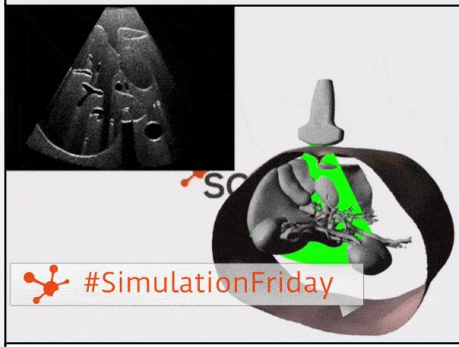
INCUBATEUR PACA-EST

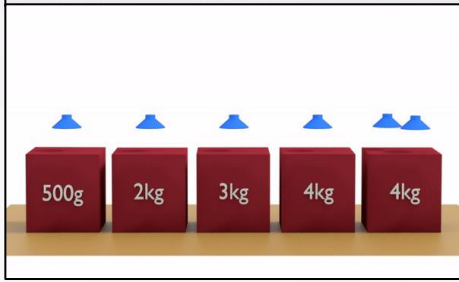
SOFA

What is SOFA?

- An interactive mechanical simulation tool
  - Implements algorithms for continuum mechanics
  - Model object deformations and their interactions
  - Suitable for creating digital twins
- Application fields
  - Medicine
    - surgical training
    - preoperative planning & intraoperative guidance
  - Robotics
    - test and create new control methods
    - design a new generation of robots
  - BioMechanics

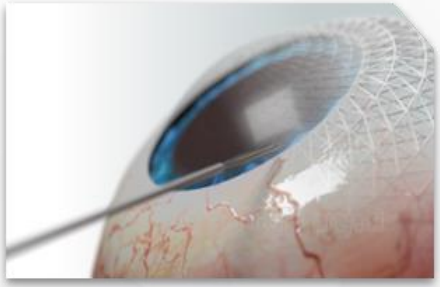


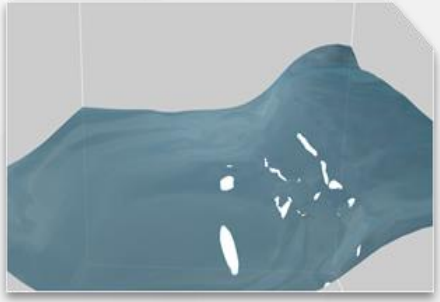




Multiphysics simulation

- Soft and rigid body dynamics
- Heat transfer
- Fluid mechanics
- Robotics & Control

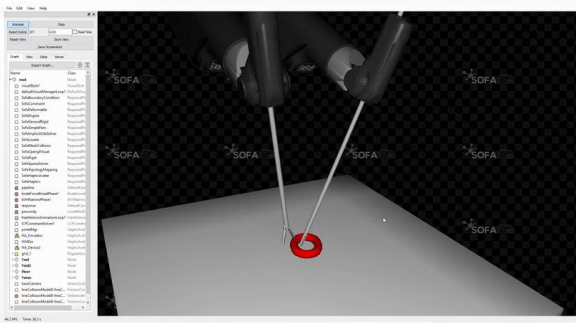


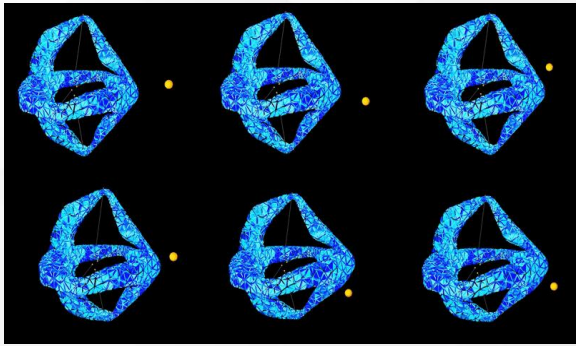


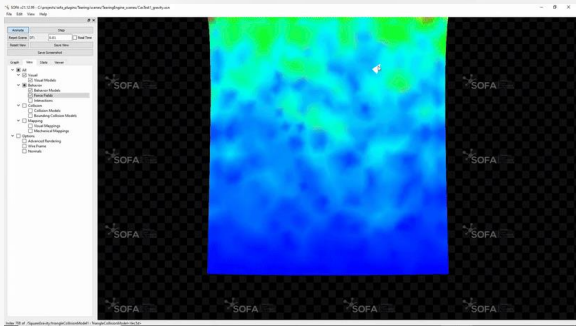
SOFA

Why choosing SOFA?

- Flexible
  - Ease design and prototyping of simulation
- Modular architecture
  - OpenCore with plugins: activate / deactivate features
- Interactive
  - User interactions to take into account
  - Enclose SOFA in optimization loops, learning
  - Dynamic simulation updates







Community and project governance

SOFA

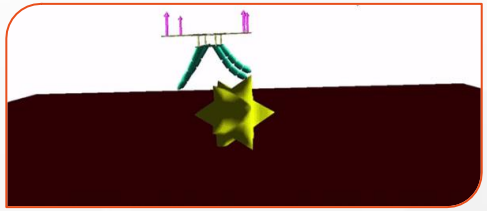
An open-source project

- Arose from research back in 2006 at Inria
- Need of a collaborative framework for physics simulation
- Project definition

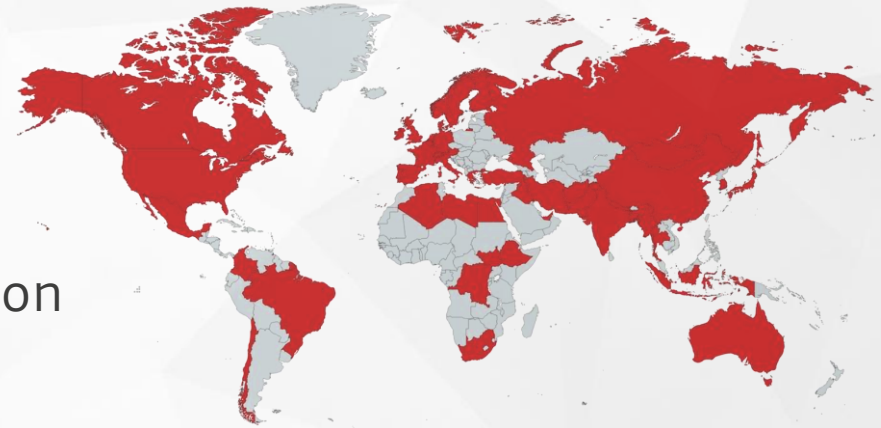
Being the open-source reference for interactive and real-time applications

SOFA

Our international community



- Thriving community
  - +140 daily SOFA-based developers
  - +500 regular users
  - +700 monthly downloads
- Driving innovation and collaboration
  - 12 startups created since 2009
  - 80% academic community
  - Industrial partners: LN Robotics, InSimo, Caranx Robotics, Dassault, Essilor
  - Academic partners: Harvard, Stanford, Johns Hopkins University, ETH Zurich, Luxembourg University, JAIST (Japan) MIT



SOFA

An open-source consortium

- Gathers partners willing to share the costs related to the missions of
  - Coordination of the developments
  - Industrialization and maintenance
  - Animation and support to the community
  - Structure the ecosystem
- Acting as a third-party and non-profit organization

SOFA

An open-source consortium

- **Strategic partners**, steering the strategy of the project

- **Technical members**, defining the roadmap based on contributions

- Donors

SOFA

Joint governance

Executive Committee (SEC)

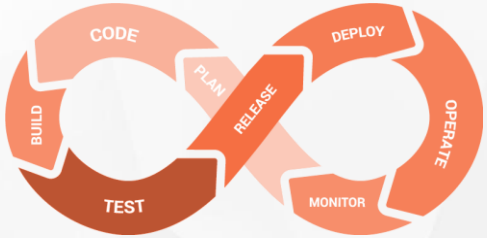
- Minimum once a year
- Decide on the Consortium engineers priority tasks
- Examine the Consortium's financial situation and take decisions accordingly
- Decide the communication policy

Technical Committee (STC)

- Twice a year : June & November
- Vote the technical roadmap
- Approve/modify SOFA development and contribution rules
- Approve who is granted write access to the master reference code

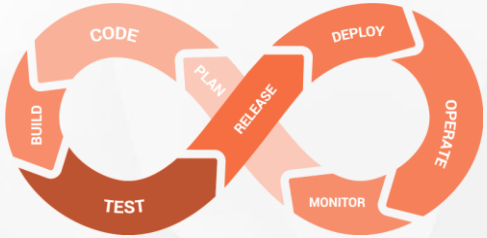
SOFA

Development process



- Robustness
  - Explicit contributing rules (GitHub)
  - Explicit review process (GitHub)
  - Continuous Integration (CI) : regression, unit and feature tests
  - Stable releases every 6 month → regular releases (v24.12)
- Foster contributions from external contributors

Development process



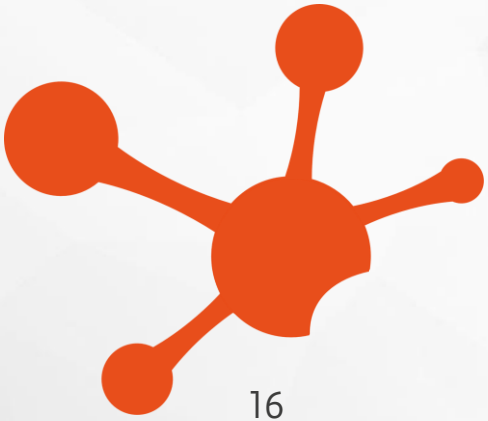
- Robustness
  - Explicit contributing rules (GitHub)
  - Explicit review process (GitHub)
  - Continuous Integration (CI) : regression, unit and feature tests
  - Stable releases every 6 month → regular releases (v24.12)
- Constant industrialization
  - Improve code stability and quality
  - Document
  - Bug tracking

Let’s discover the software project

SOFA

Software architecture

- Code structure: OpenCore
  - Stable core written in C++ under LGPL-v2.1 (regular releases)
  - Coming with Python bindings
  - Development of plugins (free-to-define licence)



SOFA

Software architecture

- Code structure: OpenCore
  - Stable core written in C++ under LGPL-v2.1 (regular releases)
  - Coming with Python bindings
  - Development of plugins (free-to-define licence)
- Multi-platform

- Documentation
  - User online documentation → sofa-framework.org/**doc**
  - API documentation → sofa-framework.org/**api**
  - Open forum → github.com/sofa-framework/sofa/**discussions**

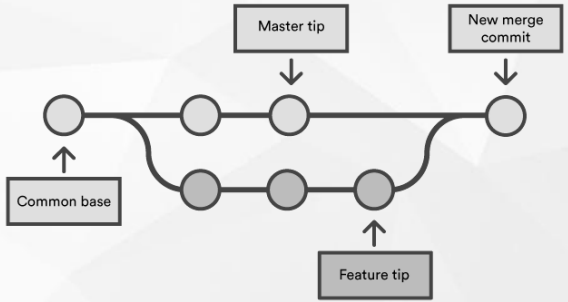
SOFA

Versioning of sources

- Relying on Git versioning system
- Hosted on GitHub

  https://github.com/sofa-framework/sofa/

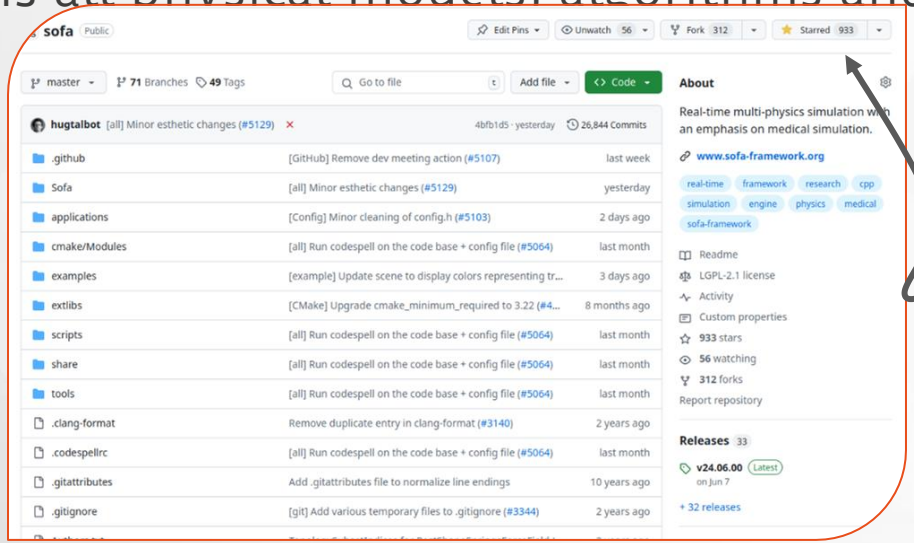
- Start with GitHub
  - Create your fork
  - For your development project, work in branches
  - Keep your master branch up-to-date regularly



SOFA

SOFA architecture

- Programmation by component: one C++ classes = one task
  - Level of abstraction defining base classes : common ground
  - You can also add totally new types : must inherit from BaseObject
- Contains all physical models, algorithms and simulation mechanisms



https://github.com/sofa-framework/sofa

Star * the project to follow and support us

SOFA architecture

- Programmation by component: one C++ classes = one task
  - Level of abstraction defining base classes : common ground
  - You can also add totally new types : must inherit from BaseObject
- Contains all physical models, algorithms and simulation mechanisms
  - **Sofa/framework/** → *no implementation, only abstract classes = API*
  - **Sofa/Component/** → *implementation of models and algorithms*
  - examples/**Component**/
  - applications/plugins/

How to get SOFA?

sofa-framework.org/download

- Compile SOFA
  - Checkout the sources
  - Configure with Cmake
  - Build (using make, ninja or others)
  - Documentation
- Download the binary version (pre-compiled)

Note that stable binary version of SOFA is available: **v24.12**

SOFA

SOFA main principles

Solid mechanics

$$\frac{d\boldsymbol{p}}{dt} = f$$

$$\frac{d\rho v}{dt} = f_{\text{vol}} + f_{\text{surf}}$$

- Historical field of application in SOFA
- Classical (Newtonian) mechanics
- Motion of bodies under external & internal forces
  - Rigid = no internal force = no deformation
  - Elasticity = return to rest shape after stresses
  - Visco-elasticity = elastic material with damping
  - Plasticity = permanent deformation after stress
  - Thermoplasticity = coupling between mechanical and thermal laws

Solid mechanics

$$\sum F = m\ a = m\ \ddot{x}$$

- Historical field of application in SOFA
- Classical (Newtonian) mechanics
- Motion of bodies under external & internal forces
  - Rigid = no internal force = no deformation
  - Elasticity = return to rest shape after stresses
  - Visco-elasticity = elastic material with damping
  - Plasticity = permanent deformation after stress
  - Thermoplasticity = coupling between mechanical and thermal laws

Solid mechanics

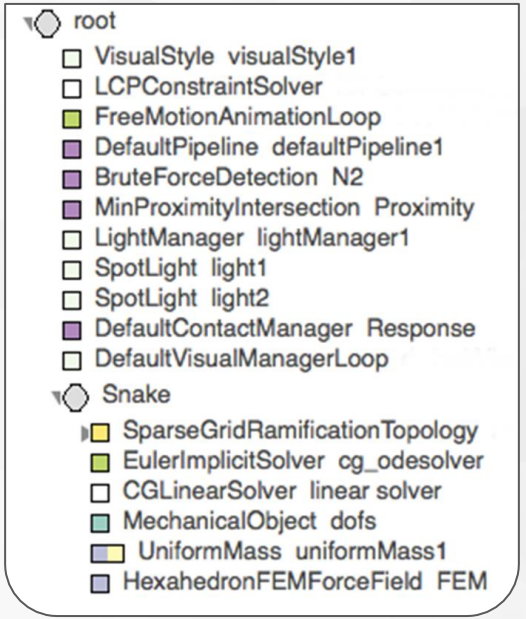
$$\sum F = m\ a = m\ \ddot{x}$$

- Historical field of application in SOFA
- Classical (Newtonian) mechanics
- Motion of bodies under external & internal forces
  - Rigid = no internal force = no deformation → Rigid particle
  - Elasticity = return to rest shape after stresses → Deformable liver
  - Visco-elasticity = elastic material with damping
  - Plasticity = permanent deformation after stress
  - Thermoplasticity = coupling between mechanical and thermal laws

Scene Graph

- Simulation described as a graph

  (Direct Acyclic Graph, i.e. generalized hierarchy)
  - Composed of nodes
  - Nodes contain **Components**
  - Components have parameters: **Data**



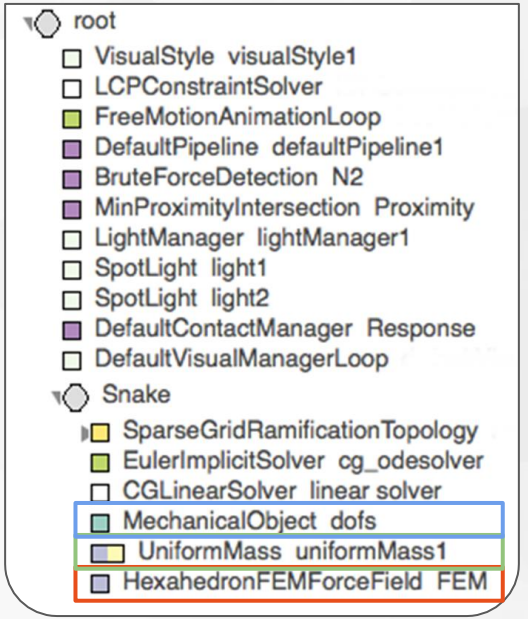
SOFA

Scene Graph

$$\sum F = m\,a = m\,\ddot{x}$$

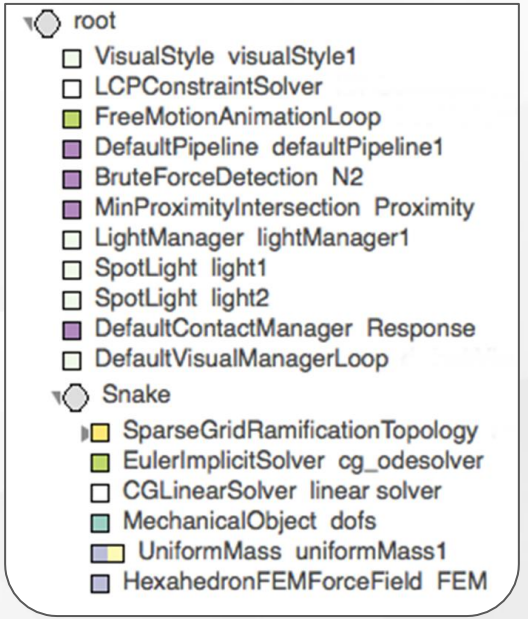
- Simulation described as a graph

  (Direct Acyclic Graph, i.e. generalized hierarchy)
  - Composed of nodes
  - Nodes contain **Components**
  - Components have parameters: **Data**



SOFA

Scene Graph

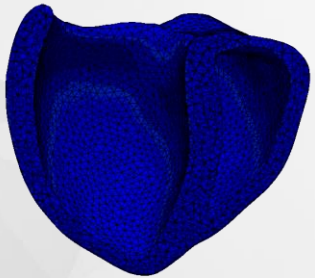


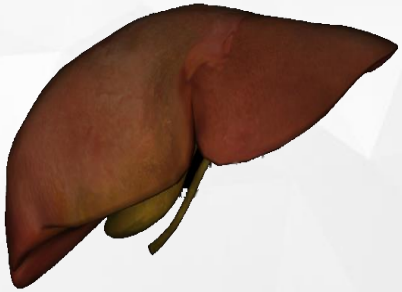
- Simulation described as a graph

  (Direct Acyclic Graph, i.e. generalized hierarchy)
  - Composed of nodes
  - Nodes contain **Components**
  - Components have parameters: **Data**
- Data can be linked together (*input=@dofs.value*)
- Described using XML or Python

SOFA

Scene Graph





Root

AnimationLoop

Heart

Liver

DOF

FF

M

Solvers

DOF

FF

M

Solvers
- integration scheme (ODE)
- linear solver

Scene Graph (XML format)

```
<Node name="root" dt="0.01" gravity="0 0 0">

        <Node name="Liver">
                <EulerImplicitSolver name="EulerImplicit_scheme" />
                <CGLinearSolver name="CG" iterations="200" tolerance="1e-09"/>
        <MechanicalObject template="Vec3d" name="myLiverDOF" position="..." />
                <MeshMatrixMass totalMass="1" />
                <TetrahedronFEMForceField name="LinearElasticity"
youngModulus="1e7" />
        </Node>


        <Node name="Heart">
                <EulerImplicitSolver name="EulerImplicit_scheme" />
                <CGLinearSolver name="CG" iterations="200" tolerance="1e-09"/>
        <MechanicalObject template="Vec3d" name="myHeartDOF" position="..." />
                <MeshMatrixMass totalMass="2" />
        </Node>

</Node>
```


SOFA

Scene Graph (XML format)

$$\sum F = m\,a = m\,\ddot{x}$$

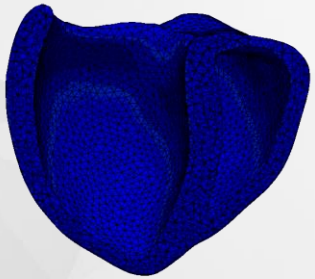
```
<Node name="root" dt="0.01" gravity="0 0 0">

        <Node name="Liver">
                <EulerImplicitSolver name="EulerImplicit_scheme" />
                <CGLinearSolver name="CG" iterations="200" tolerance="1e-09"/>
        <MechanicalObject template="Vec3d" name="myLiverDOF" position="..." />
                <MeshMatrixMass totalMass="1" />
                <TetrahedronFEMForceField name="LinearElasticity"
youngModulus="1e7" />
        </Node>


        <Node name="Heart">
                <EulerImplicitSolver name="EulerImplicit_scheme" />
                <CGLinearSolver name="CG" iterations="200" tolerance="1e-09"/>
        <MechanicalObject template="Vec3d" name="myHeartDOF" position="..." />
                <MeshMatrixMass totalMass="2" />
        </Node>

</Node>
```

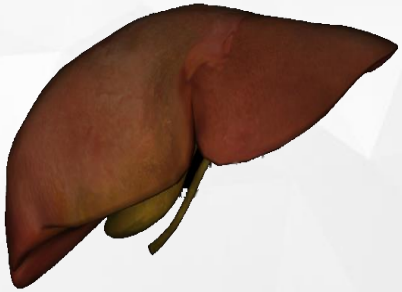

SOFA

1. Scene Graph (Python format)

```
import Sofa
import SofaRuntime

def createScene(rootNode):
          rootNode.dt=0.01
          rootNode.gravity=[0, 0, 0]

          liverNode = rootNode.addChild("Liver")
          liverNode.addObject(EulerImplicitSolver)
          liverNode.addObject('CGLinearSolver', iterations="200", tolerance="1e-09"...)
          liverNode.addObject('MechanicalObject',  template="Vec3d", name="myLiverDOF",
position="...")
          liverNode.addObject('MeshMatrixMass', name="mass", totalMass="1" )
          liverNode.addObject('TetrahedronFEMForceField', name="LinearElasticity",
youngModulus="1e7")

          heartNode = rootNode.addChild("Heart")
          heartNode.addObject(EulerImplicitSolver)
          heartNode.addObject('CGLinearSolver', iterations="200", tolerance="1e-09"...)
          heartNode.addObject('MechanicalObject',  template="Vec3d", name="myHeartDOF",
position="...")
          heartNode.addObject('MeshMatrixMass', name="mass", totalMass="2" )
```

Scene Graph





Root

AnimationLoop

Heart

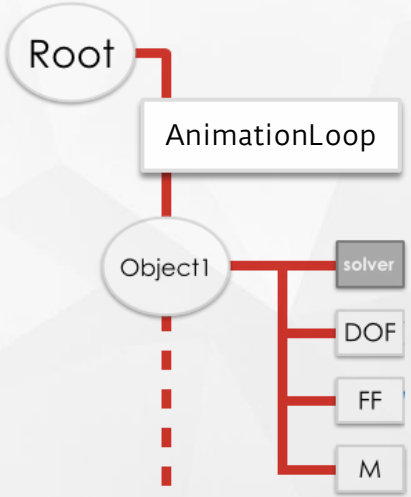
Liver

DOF FF M solver

DOF FF M solver

SOFA

AnimationLoop and Visitor



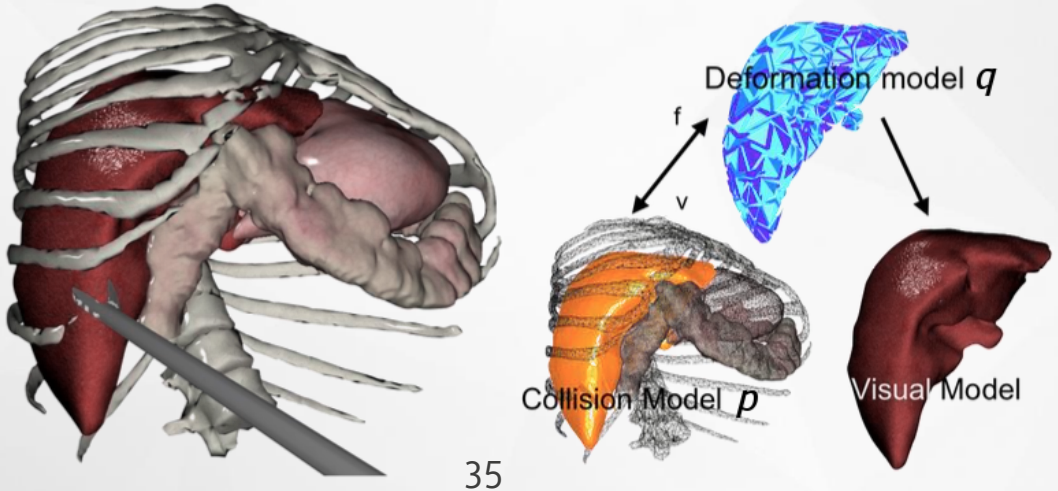
- Compulsory in a simulation
- Rules and orders the simulation into steps.

  One time-step performs:
  - Collision detection
  - Constraint resolution
  - Solving the mechanical matrix system
- At each time-step, **Visitors (C++ pattern)** browse the graph:
  - Graph traversal = traversing each component
  - Specific visitor triggers specific functions in components
    (ex: *accumulateForce* → *addForce()*)

SOFA

Mapping

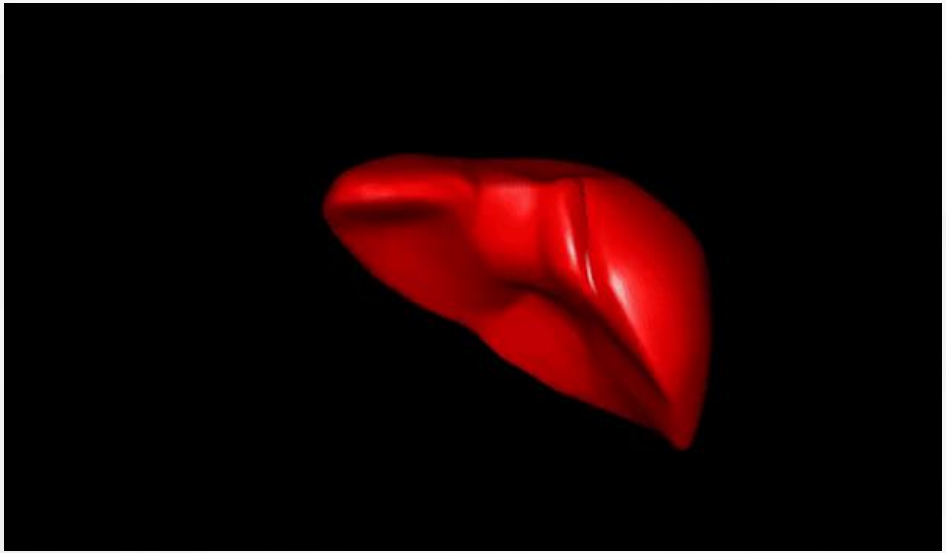
- Each object can have several models in SOFA
  - Mechanical model
  - Collision detection model
  - Visual model



Mapping

- Each object can have several models in SOFA
  - Mechanical model
  - Collision detection model
  - Visual model
- Each model can rely on a different topology
- Correspondence between these models is a Mapping!

Mapping

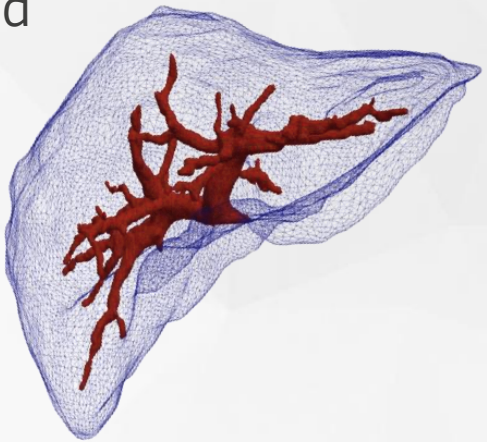


SOFA

SOFA features

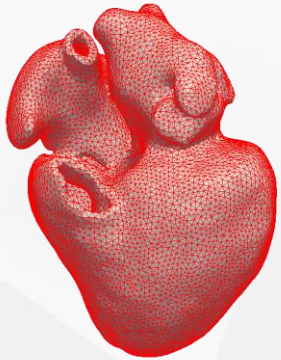
Loading geometries - MeshLoaders

- Discretization in space using the Finite Element Method
- Mesh are a required input: volumetric, surface and beam elements
- Using mappings, different topologies can be used for each representation (mechanics, visual, collision detection)

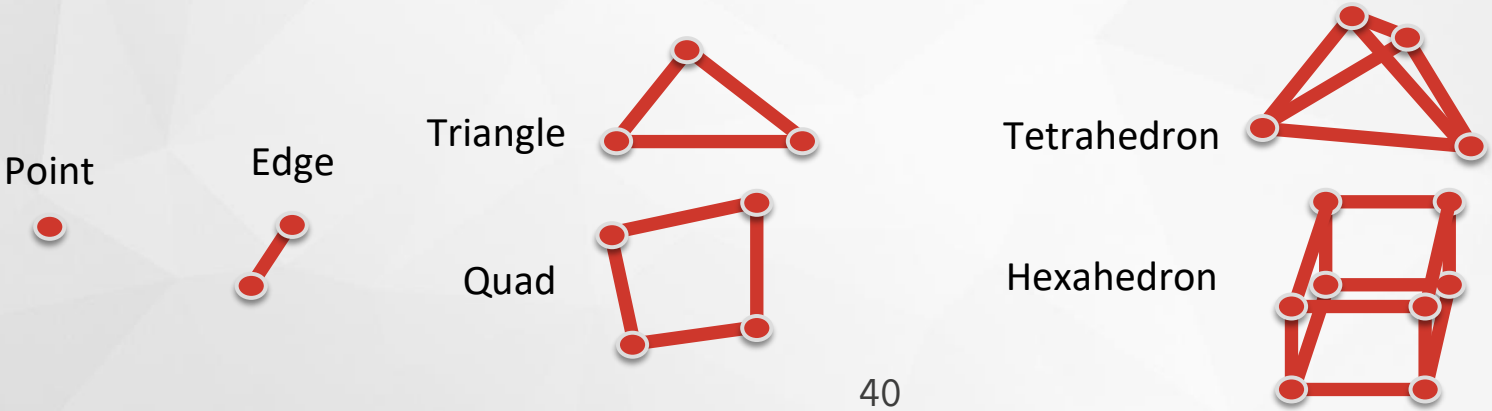


SOFA

Topology



- Describes the geometry
- Discretization of space in elements (DOF): definition of connectivity
- Load topologies in your simulation

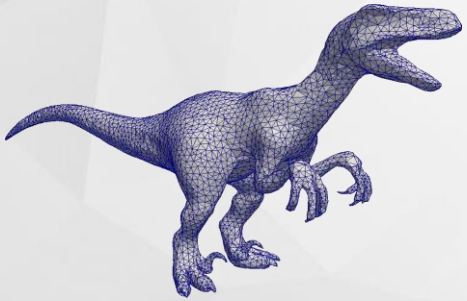


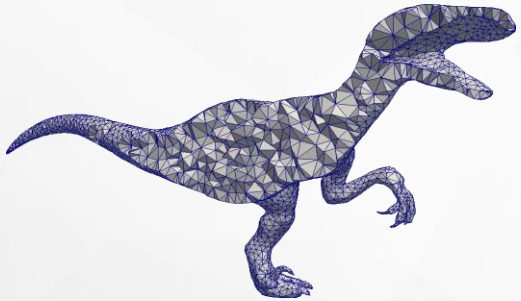
SOFA

Topology

- For each representation (mechanics, visual, collision detection), different topologies can be used

- Mappings links them

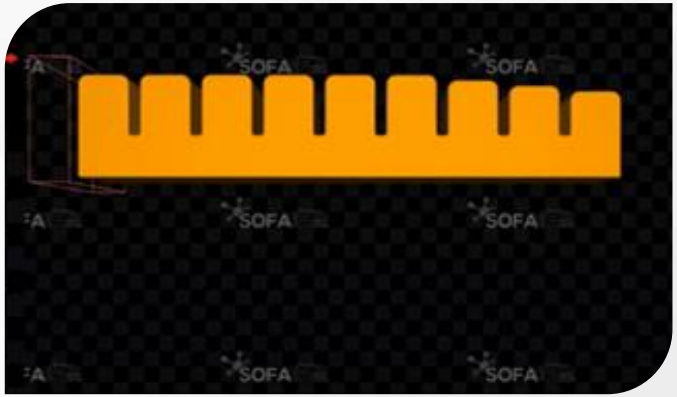




SOFA

Solid mechanics

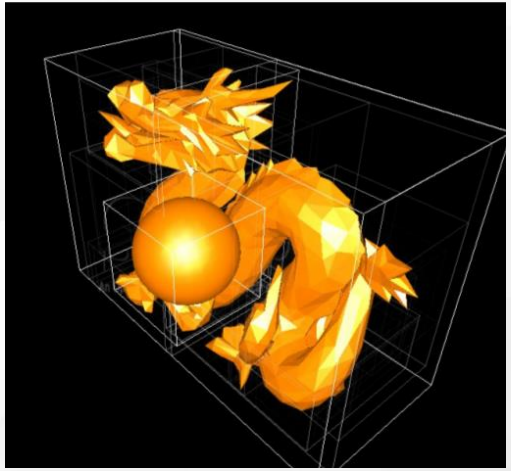
- Material laws
  - Linear elasticity
  - Plasticity
  - Hyperelasticity
  - Damping : numerical or physical
- Collision pipeline
  - Detection
  - Response
- Integration schemes and linear solvers

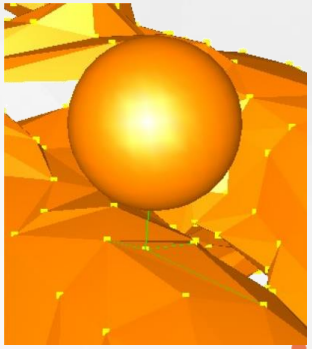


Collision pipeline

- Detection
  - Broad phase
    checks to quickly exit if objects are far
  - Narrow phase
    detects contacts between primitives

    using proximity intersection
- Response

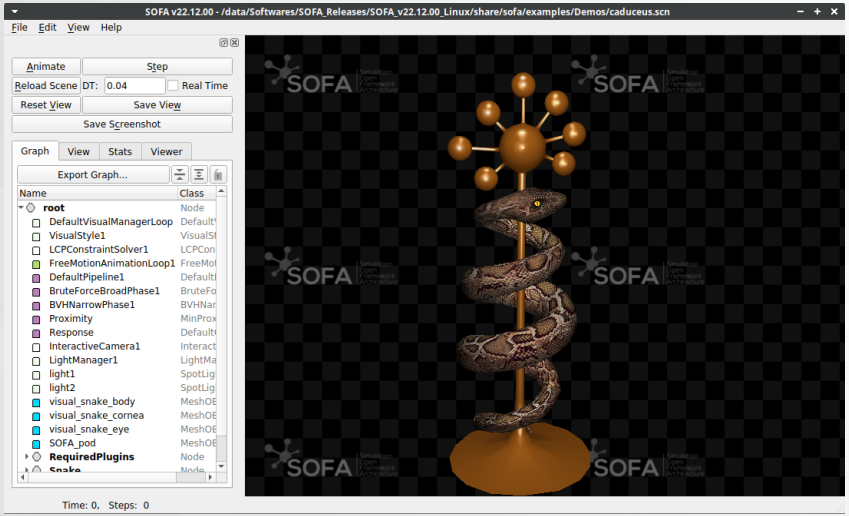


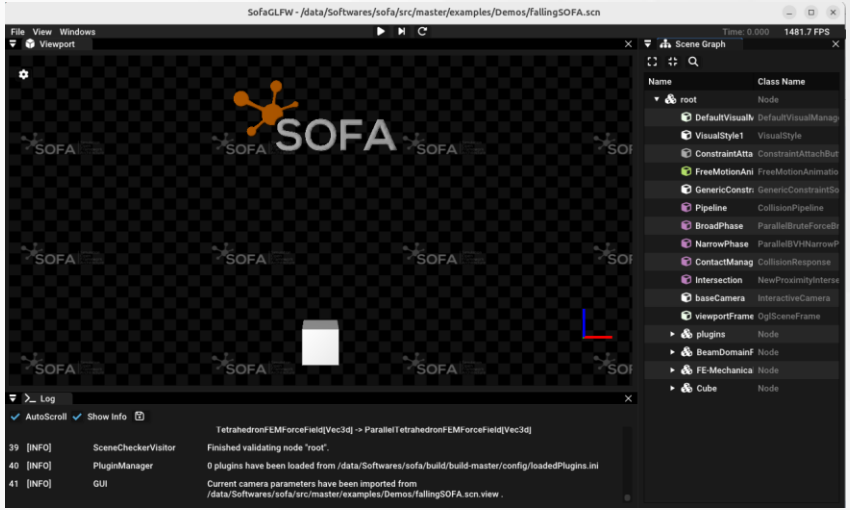


SOFA

GUI runSofa

- Build-in Graphical User Interface (GUI) is available
- When running the SOFA executable *./runSofa*

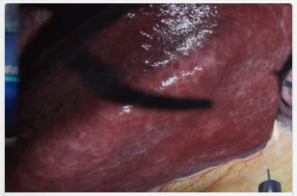




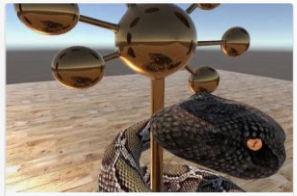
Plugins

- Officially-maintained plugins available in SOFA releases
- License of plugins is up to the plugin authors

SOFA & UnrealEngine 5 – Basic Renderer

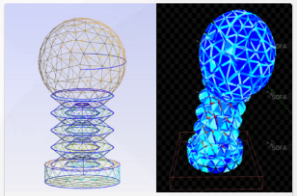


SOFA & Unity3D – Full Integration

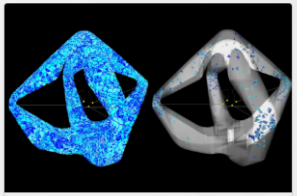


SOFA & Unity3D – Basic Renderer

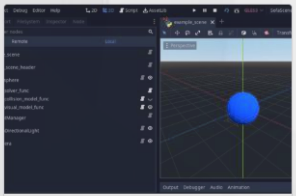
Reinforcement Learning Framework
SofaGym



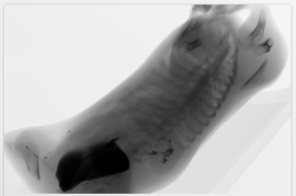
Design Optimization
This software toolkit implem...



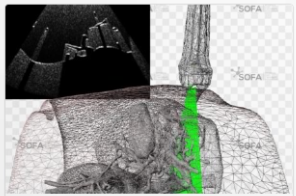
Model Order Reduction
Available under the GPL open...



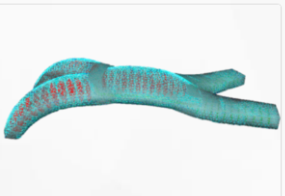
SOFA & Godot – Basic Renderer



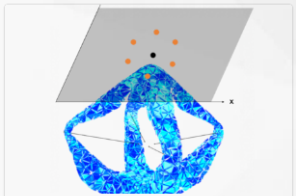
X-Ray Rendering
For many applications in the...



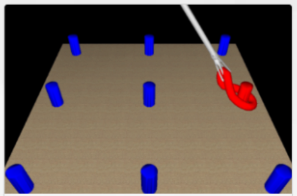
Ultra-Sound Rendering
This work, recently started ...



Reinforcement Learning with Domain Randomization



Condensed FEM Model Learning
This plugin for the open-sou...



LapGym
This repository introduces L...

SOFA

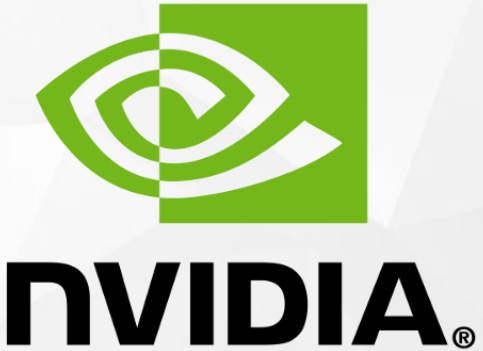
SofaPython3: scripting in python™

- As in XML, it allows for:
  - Fast prototyping
  - Create a scene without C++ skills
- Dynamic modifications (data, component creation), interactions
- Run your simulation for runSofa or python environment
- Create new components as python class!
- Official repository: https://github.com/sofa-framework/sofapython3/

SOFA

SofaCUDA: GPU computing

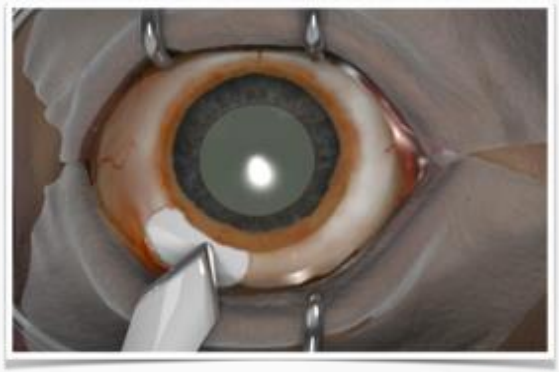
- Compatibility with CUDA GPUs
- Activate GPU version of the C++ code
- Only templates need to be changed (CudaVec3d)
- Accelerate your own code



SOFA

Interactive/haptic interfaces

- Compatible with many devices
  - Geomagic
  - Falcon
  - Entact
  - Leap Motion
  - Follou laparoscopic devices
  - Haply Robotics
  - ForceDimensions



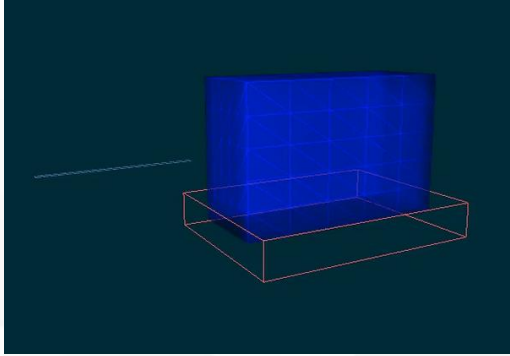
SOFA

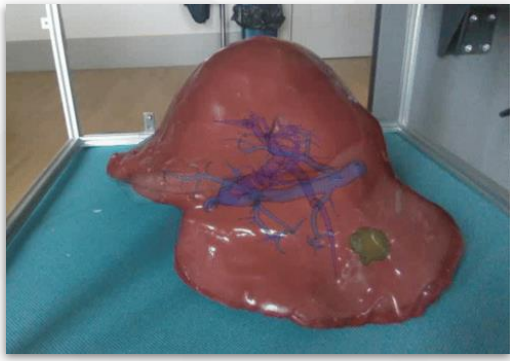
Getting started

SOFA

Why choosing SOFA?

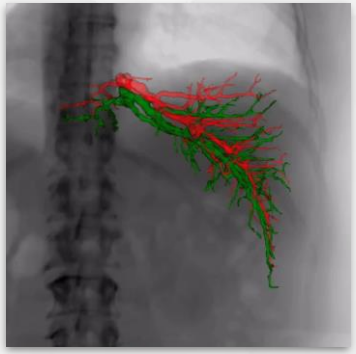
- Flexible
  - Ease design and prototyping of simulation
- Modular architecture
  - OpenCore with plugins: activate / deactivate features
- Interactive
  - User interactions to take into account
  - Enclose SOFA in optimization loops, learning
  - Dynamic simulation updates

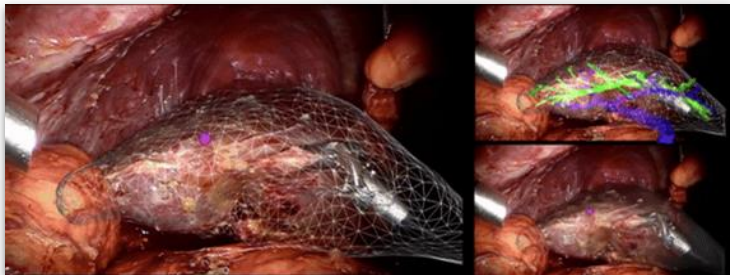




Applications in medicine

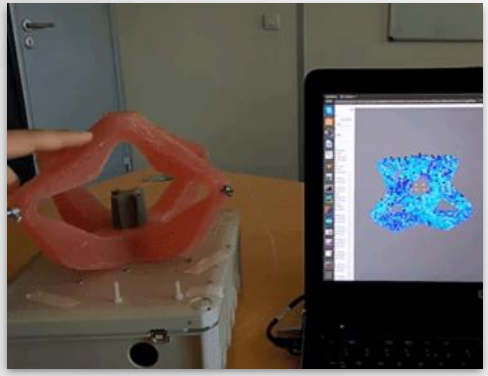
- For medical training
  - Surgical gesture training, avoiding the 1st gesture on a patient
  - Force feedback
- Predict and propose optimal surgical strategies
  - Optimize strategies (trajectory, entry point, device)
  - Optimize medical devices → patient-specific
- Guide during surgery
  - Visualize sensitive structures
  - Help navigating, keeping a target in sight

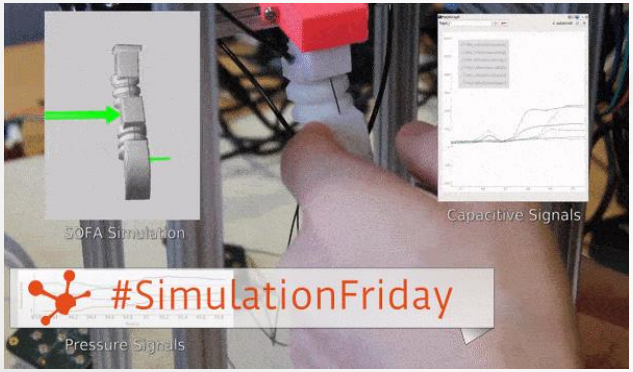


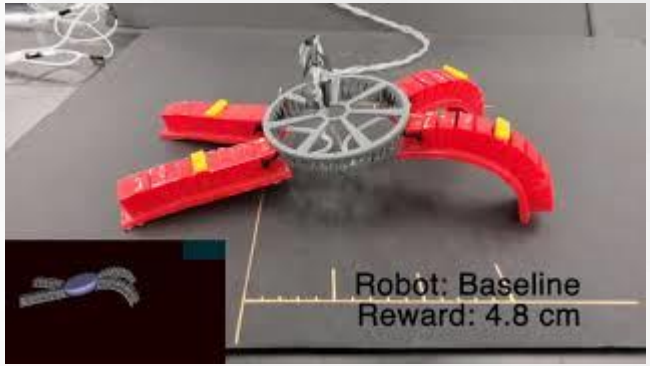


Applications in (soft)robotics

- Modeling a deformable robot or its deformable environment
- Design a new generation of soft-robots, closing the Sim2Real gap
- Assess new control laws, possibly learning/AI-based

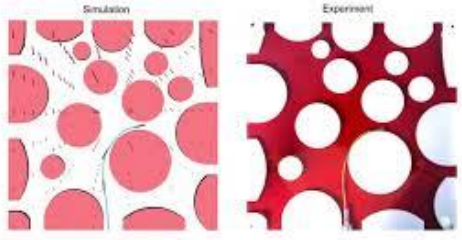






Applications in (soft)robotics

- Actuators
  - Pneumatic where the pressure of the chamber can be controlled
  - Fluidic where the volume of the chamber can be controlled
  - Tendon or a cable
  - Electromagnetic (proof of concept by ETH)
- Performance: model-order reduction, MT, GPU
- Learning from simulations
  - SofaGym: Reinforcement Learning - Open AI
  - DeepPhysX: Deep Neural Network - pytorch



Go further with SOFA

- Learn SOFA
  - Courses through the InriaAcademy program → inria-academy.fr
  - Free online introduction course on YouTube (doc)
- Get support
  - Online documentation → sofa-framework.org/doc
  - Forum : GitHub Discussions → github.com/sofa-framework/sofa/discussions
  - Chat on Discord with developers → discord.gg/G63t3a8Ra6

All information → sofa-framework.org/community/get-involved

Let’s collaborate

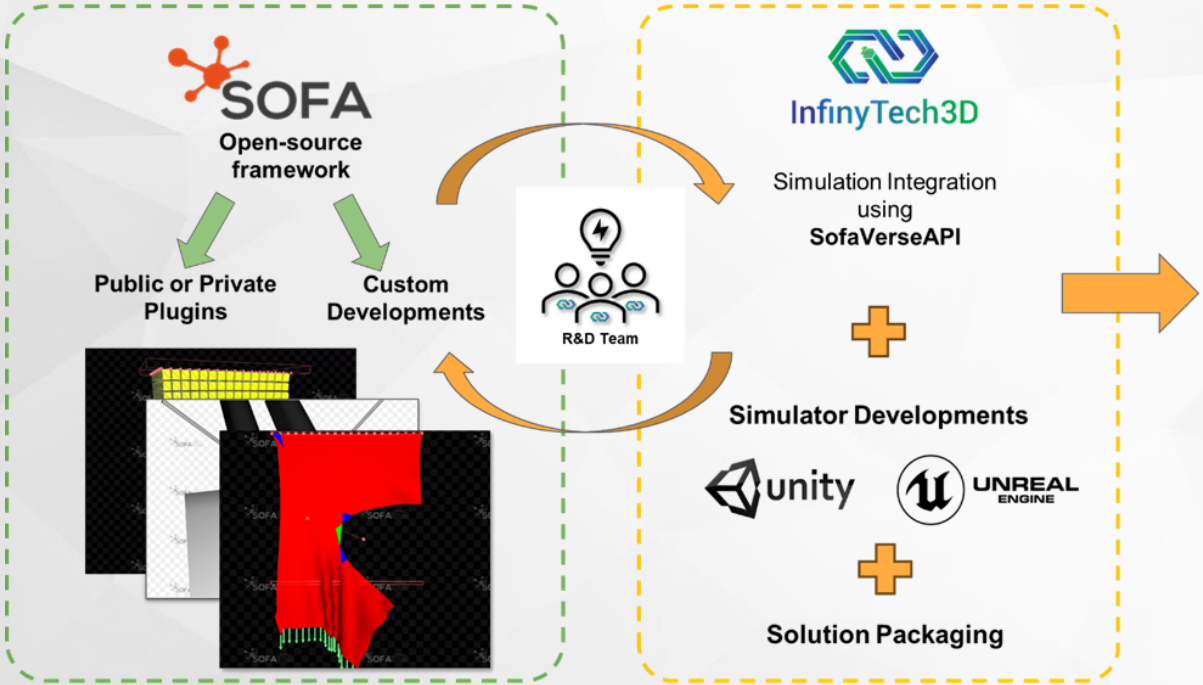
- Worldwide academic community
  - CIFRE
  - Bilateral partnership
  - EU project proposals
- Bilateral partnership with companies
  - Consulting
  - Product co-development
- Become a SOFA Consortium member

InfinyTech3D COMPLIANCE ROBOTICS

SOFA

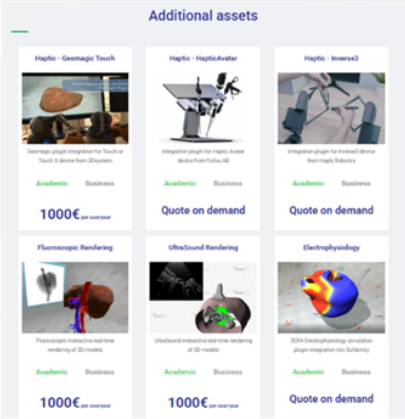
InfinyTech3D Model

Delivering Innovation consulting through SOFA at every step of the simulation



Freemium Model

Free	Academic	Enterprise	Commercial
access	SDK	SDK	License
The basics for learning for your projects	Advanced collaboration for research	Give you all rights to use the asset for commercial product development	Give you all rights to use the asset for commercial product deployment
0€ per user/year	500€* per user/year	2000€* per user/year	4000€* per site
Get from GitHub	Request License	Request License	Request License



All info here: https://infinytech3d.com/licenses/

Let’s meet: SOFA Week 2025

- In France (Lille) on November 24-28th
  - Discover the latest technological advances
  - Meet and networking within the community
- Program
  - Training session - 24th Nov
  - **SOFA Conference - 25th Nov**
  - Technical Committee - 26-28th Nov

Thank you!

- Discover SOFA, join the community and benefit from it!
- Meet us again at our 2025 events
- Any question? Web www.sofa-framework.org

Contact