SOFA: a modular yet efficient physical simulation architecture

François Faure, INRIA

October 2012

Outline

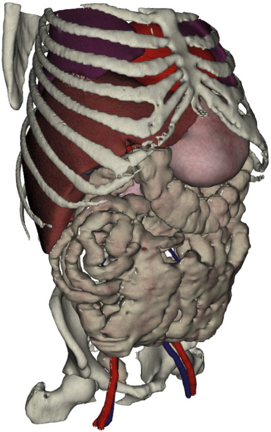
Motivation

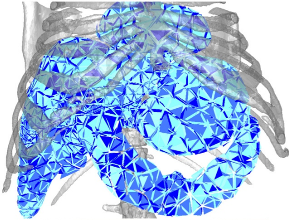
Modularity

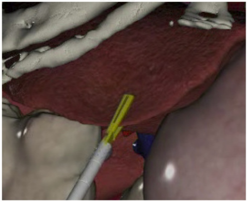
Software development

Conclusion

The complexity of physical simulations







Material, internal forces, contraints, contact detection and modeling, ODE solution, visualization, interaction, etc.

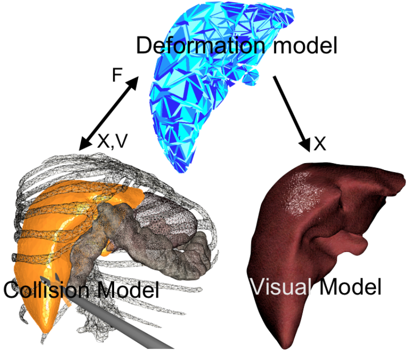
Simulation platforms

- Current platforms (ODE, Havok, PhysX, etc.) provide :
  - limited number of material types
  - limited number of geometry types
  - no control on collision detection algorithms
  - no control on interaction modeling
  - few (if any) control of the numerical models and methods.
  - no control on the main loop
- We need much more !
  - models, algorithms, scheduling, visualization, etc.

Outline

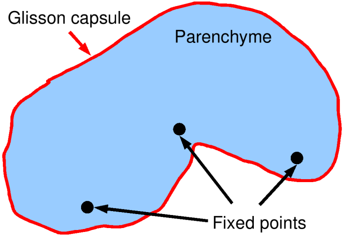
A generic approach

- Behavior model : all internal laws
- Others : interaction with the world
- Mappings : relations between the models (uni- or bi-directional)



Animation of a simple body

- a liver



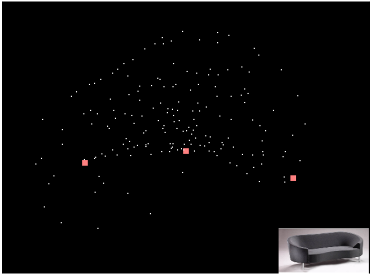
- inside : soft material
- surface : stiffer material

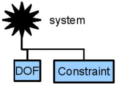
A specialized program :

```
f  =  M*g
f  += F1(x,v)
f  += F2(x,v)
a  =  f/M
a  =  C(a)
v  += a * dt
x  += v * dt
display(x)
```

Component-based simulation object

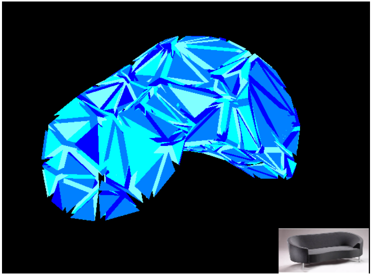
- state vectors (DOF) :
  x, v, a, f
- constraints : fixed points
  other : oscillator, collision plane, etc.

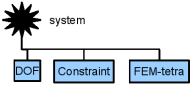




Component-based simulation object

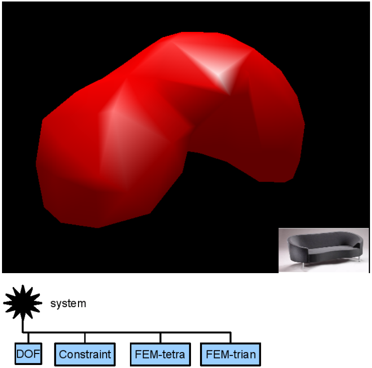
- state vectors (DOF) :
  x, v, a, f
- constraints : fixed points
- force field : tetrahedron FEM
  other : triangle FEM, springs, Lennard-Jones, SPH, etc.





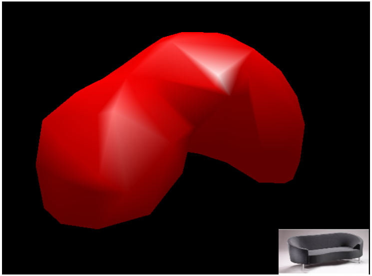
Component-based simulation object

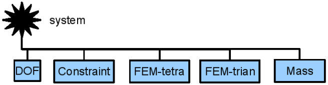
- state vectors (DOF) : x, v, a, f
- constraints : fixed points
- force field : tetrahedron FEM
- force field : triangle FEM



Component-based simulation object

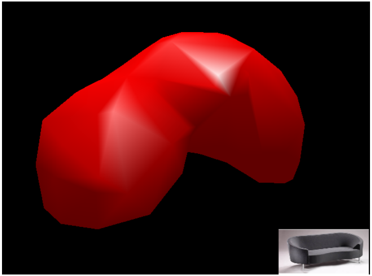
- state vectors (DOF) : x, v, a, f
- constraints : fixed points
- force field : tetrahedron FEM
- force field : triangle FEM
- mass : uniform
  other : diagonal, sparse symmetric matrix

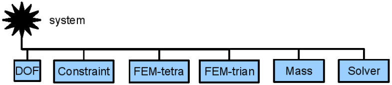




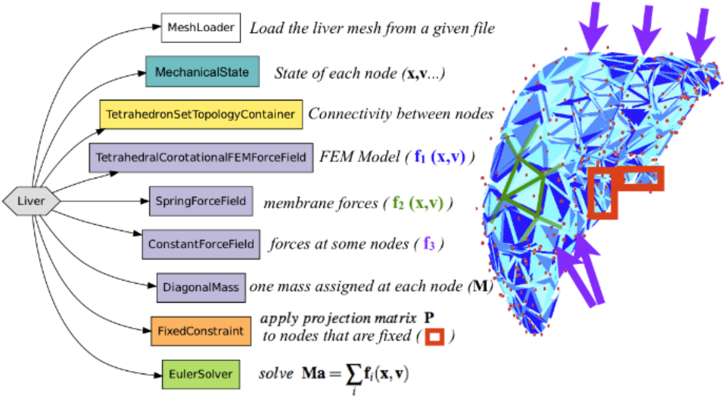
Component-based simulation object

- state vectors (DOF) : x, v, a, f
- constraints : fixed points
- force field : tetrahedron FEM
- force field : triangle FEM
- mass : uniform
- ODE solver : explicit Euler
  other : Runge-Kutta, implicite Euler, static solution, etc.



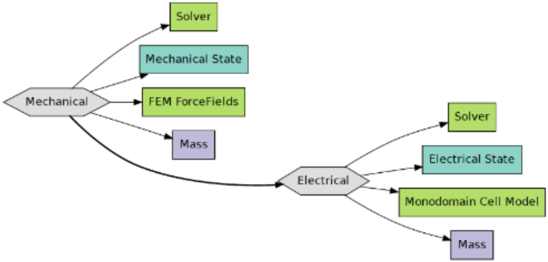


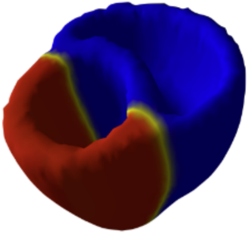
Actual decomposition



Multiphysics

Non-mechanical solvers can be coupled with the simulation





Operations

- The ODE solver sends visitors to apply operations
- The visitors traverse the scene and apply virtual methods to the components
- The methods read and write state vectors (identified by symbolic constants) in the DOF component
- Example : accumulate force
  - A `ResetForceVisitor` recursively traverses the nodes of the scene (only one node here)
    - All the `DOF` objects apply their `resetForce()` method
  - An `AccumulateForceVisitor` recursively traverses the nodes of the scene
    - All the `ForceField` objects apply their `addForce( Forces, const Positions, const Velocities )` method
  - the final value of f is weight + tetra fem force + trian fem force

Outline

Implementation

- C++ libraries
- Linux, Windows, Mac
- 800,000 lines of code
- 138,000 downloads

Contributors

80 contributors to the code. Three full-time engineers.

- Cimit (Boston)/Graphix (Lille)
- Evasion/Imagine (Grenoble)
- Epidaure/Asclepios (Sophia-Antipolis)
- Shacra (Strasbourg)
- MOAIS (Grenoble), Sed Sophia, ...
- Companies : Digital Trainers, Bellcurves, InSimo, ...

Features

- deformable solids : FEM, cables, frame-based,...
- mass-springs, rigid, SPH
- soft, stiff, hard constraints
- iterative and direct solvers
- collision detection and response
- multiphysics

Outline

Motivation

Modularity

Software development

Conclusion

Conclusion - Features

High modularity :

- Abstract components : DOF, Force, Constraint, Solver, Topology, Mass, CollisionModel, VisualModel, etc.
- Multimodel simulations using mappings
- Explicit and implicit solvers, Lagrange multipliers

Efficiency :

- global vectors and matrices are avoided
- parallel implementations

Implementation :

- currently $450,000$ C++ lines
- Linux, MacOs, Windows

Ongoing work

- models and algorithms : better numerical solvers, cutting, haptics, Eulerian fluids...
- asynchronous simulation/rendering/haptic feedback
- multiphysics (electrical/mechanical)
- parallelism for everyone
- more documentation

www.sofa-framework.org