

SOFA: a modular yet efficient physical simulation architecture

Francois.Faure@imag.fr



Outline

Motivation

Simple bodies

Layered bodies

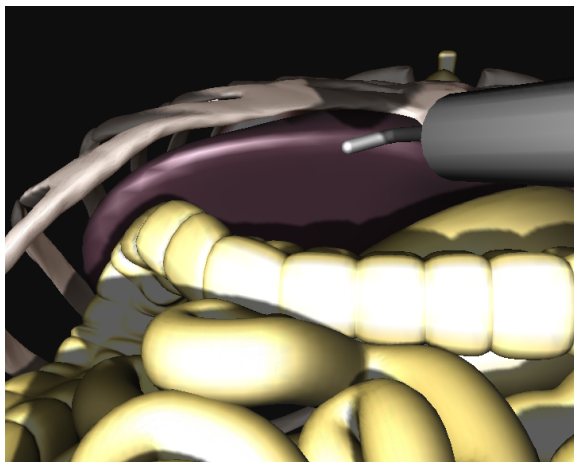
Mappings

Interacting bodies

Data processing

Conclusion

A complex physical simulation



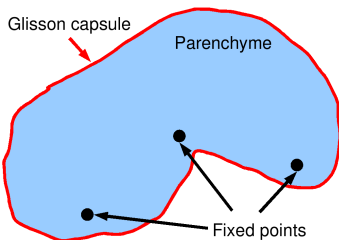
Material, internal forces, constraints, contact detection and modeling, ODE solution, visualization, interaction, etc.

Simulation platforms

- ▶ Current platforms (ODE, Havok, Novodex, 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
 - ▶ few (if any) parallelism
- ▶ We need much more !
 - ▶ models, algorithms, scheduling, visualization, etc.

Animation of a simple body

▶ a liver



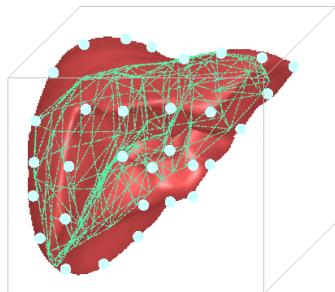
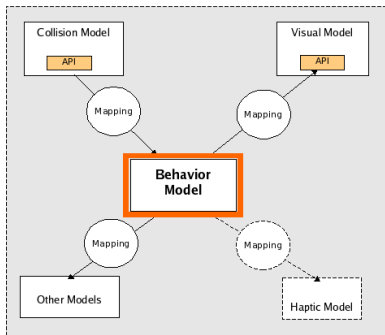
- ▶ inside : soft material
- ▶ surface : stiffer material

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)
```

A generic approach

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



Outline

Motivation

Simple bodies

Layered bodies

Mappings

Interacting bodies

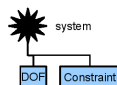
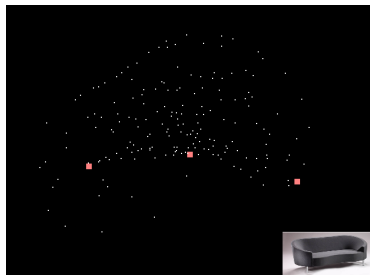
Data processing

Conclusion

Components

Data :

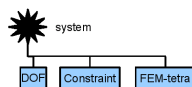
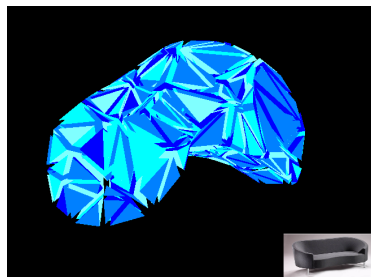
- ▶ sample points and associated values :
 x, v, a, f
- ▶ constraints : fixed points
other : oscillator, collision plane, etc.



Components

Data :

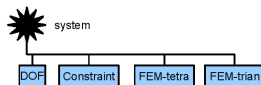
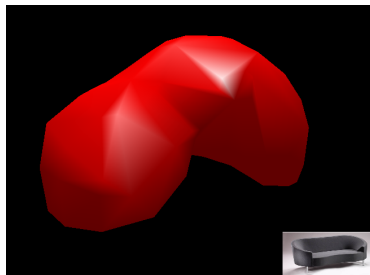
- ▶ sample points and associated values :
 x, v, a, f
- ▶ constraints : fixed points
- ▶ force field : tetrahedron FEM
other : triangle FEM, springs, Lennard-Jones, SPH, etc.



Components

Data :

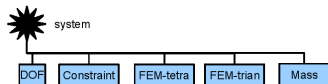
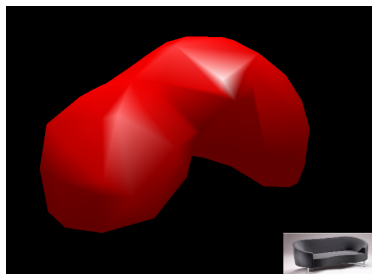
- ▶ sample points and associated values :
 x, v, a, f
- ▶ constraints : fixed points
- ▶ force field : tetrahedron FEM
- ▶ force field : triangle FEM



Components

Data :

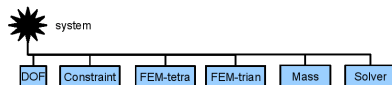
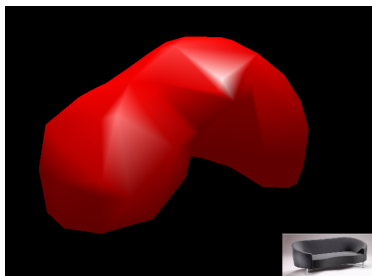
- ▶ sample points and associated values :
 x, v, a, f
- ▶ constraints : fixed points
- ▶ force field : tetrahedron FEM
- ▶ force field : triangle FEM
- ▶ mass : uniform
other : diagonal, sparse symmetric matrix



Components

Data :

- ▶ sample points and associated values :
 x, v, a, f
- ▶ constraints : fixed points
- ▶ force field : tetrahedron FEM
- ▶ force field : triangle FEM
- ▶ mass : uniform
- ▶ ODE solver : explicit Euler
other : Runge-Kutta, implicate Euler, static solution, etc.



Outline

Motivation

Simple bodies

Layered bodies

Mappings

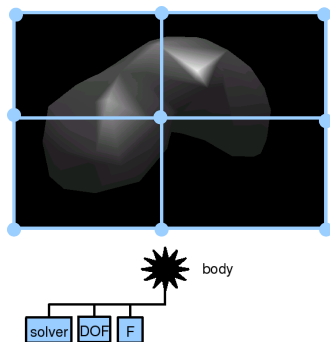
Interacting bodies

Data processing

Conclusion

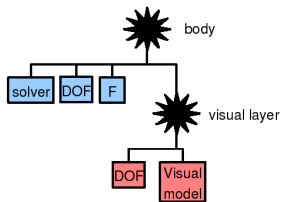
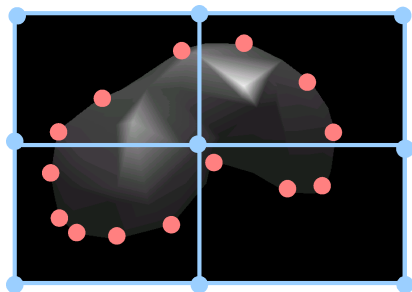
Layered body

- ▶ independent DOFs (blue)



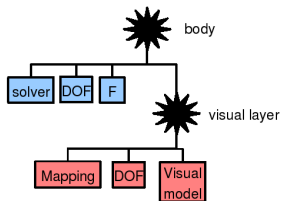
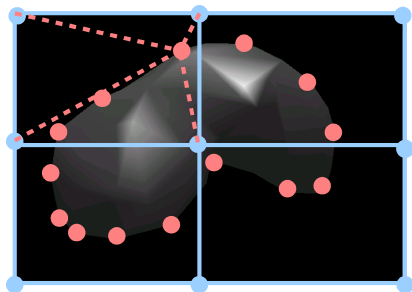
Layered body

- ▶ independent DOFs (blue)
- ▶ visual samples (salmon)



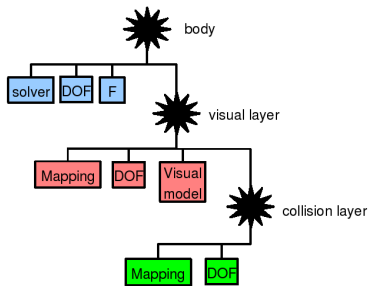
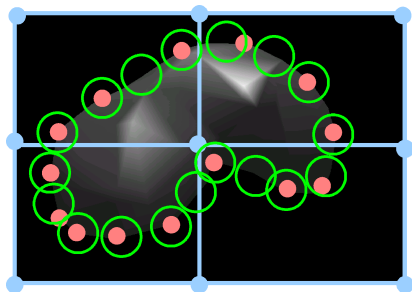
Layered body

- ▶ independent DOFs (blue)
- ▶ visual samples (salmon)
- ▶ visual mapping



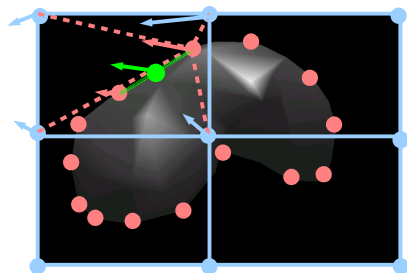
Layered body

- ▶ independent DOFs (blue)
- ▶ visual samples (salmon)
- ▶ visual mapping
- ▶ collision samples (green)
- ▶ collision mapping



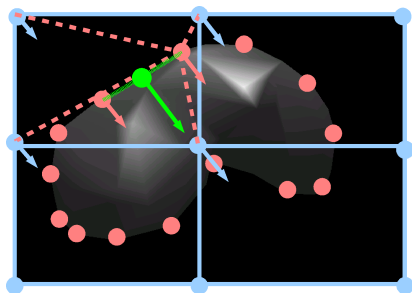
Layered body

- ▶ independent DOFs (blue)
- ▶ visual samples (salmon)
- ▶ visual mapping
- ▶ collision samples (green)
- ▶ collision mapping
- ▶ apply displacements
 1. $v_{visual} = J_{visual}v$
 2. $v_{collision} = J_{collision}v_{visual}$



Layered body

- ▶ independent DOFs (blue)
- ▶ visual samples (salmon)
- ▶ visual mapping
- ▶ collision samples (green)
- ▶ collision mapping
- ▶ apply displacements
 1. $v_{visual} = J_{visual} v$
 2. $v_{collision} = J_{collision} v_{visual}$
- ▶ apply forces
 1. $f_{visual} = J_{collision}^T f_{collision}$
 2. $f = J_{visual}^T f_{visual}$



More on mappings

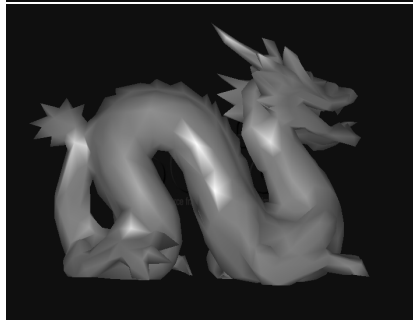
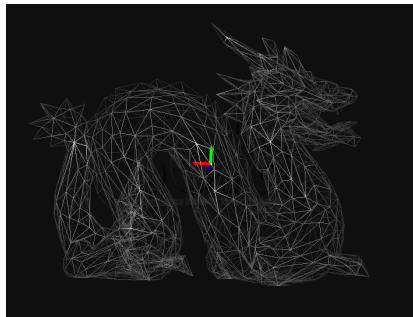
- ▶ Map a set of degrees of freedom (the parent) to another (the child).
- ▶ Typically used to attach a geometry to control points.
- ▶ Child degrees of freedom (DOF) are not independent : their positions are totally defined by their parent's.
- ▶ Displacements are propagated top-down (parent to child) :
$$v_{child} = Jv_{parent}$$
- ▶ Forces are propagated bottom-up

The physics of mappings

Example : line mapping

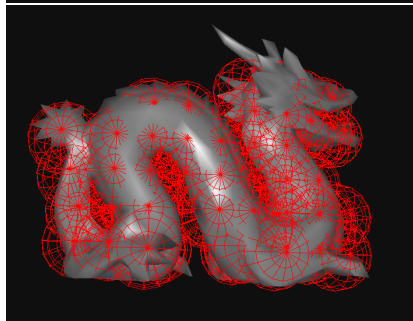
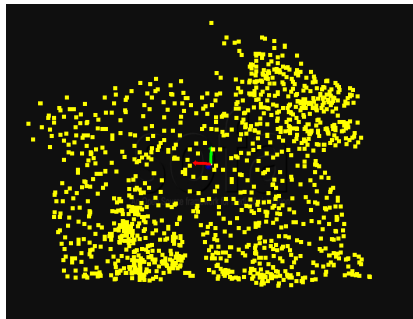
Examples of mappings

- ▶ RigidMapping can be used to attach points to a rigid body
 - ▶ to attach a visual model



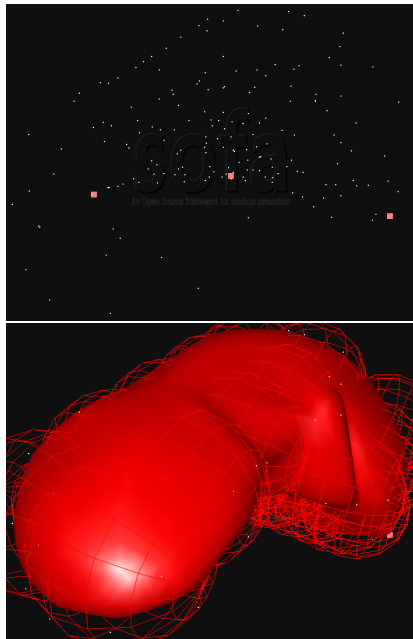
Examples of mappings

- ▶ RigidMapping can be used to attach points to a rigid body
 - ▶ to attach collision surfaces



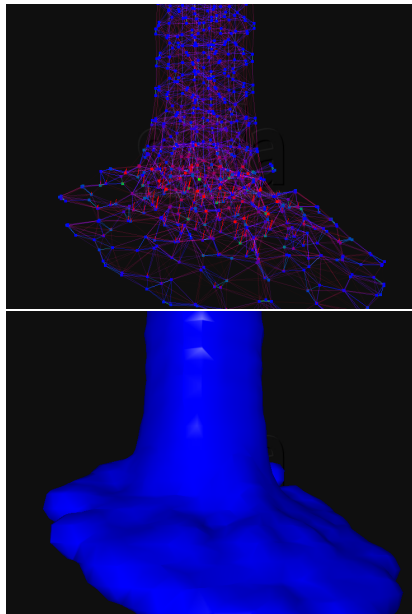
Examples of mappings

- ▶ RigidMapping can be used to attach points to a rigid body
- ▶ BarycentricMapping can be used to attach points to a deformable body
 - ▶ to attach collision surfaces



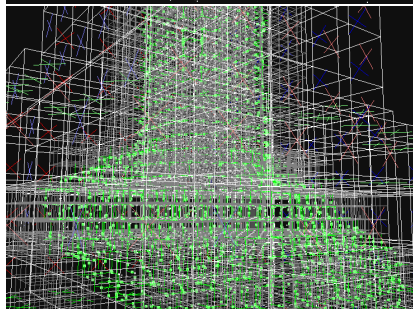
Examples of mappings

- ▶ RigidMapping can be used to attach points to a rigid body
- ▶ BarycentricMapping can be used to attach points to a deformable body
- ▶ More advanced mapping can be applied to fluids



Examples of mappings

- ▶ RigidMapping can be used to attach points to a rigid body
- ▶ BarycentricMapping can be used to attach points to a deformable body
- ▶ More advanced mapping can be applied to fluids



On the physical consistency of mappings

- ▶ Conservation of energy :

Necessary condition : $v_{child} = Jv_{parent} \Rightarrow f_{parent}^+ = J^T f_{child}$

- ▶ Conservation of momentum :

Mass is modeled at one level only. There is no transfer of momentum.

- ▶ Constraints on displacements (e.g. incompressibility, fixed points) are not easily applied *at the child level*

Outline

Motivation

Simple bodies

Layered bodies

Mappings

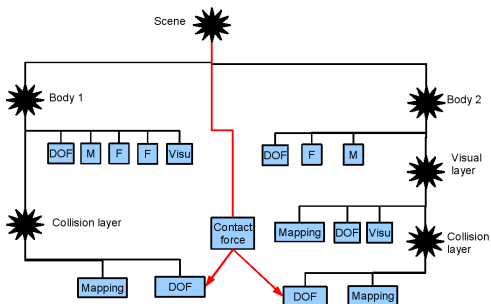
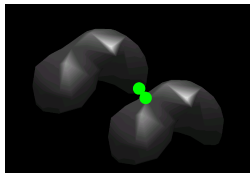
Interacting bodies

Data processing

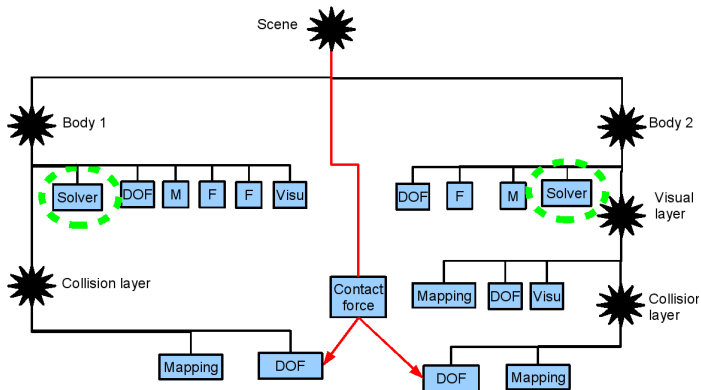
Conclusion

Two bodies in contact

Use extended trees (Directed Acyclic Graphs) to model trees with loops.

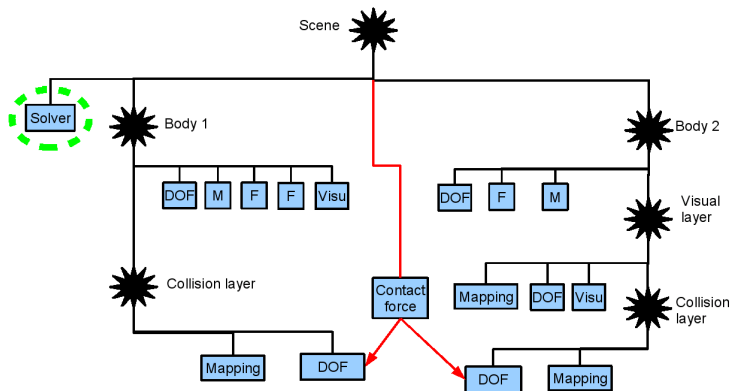


ODE solution of interacting bodies



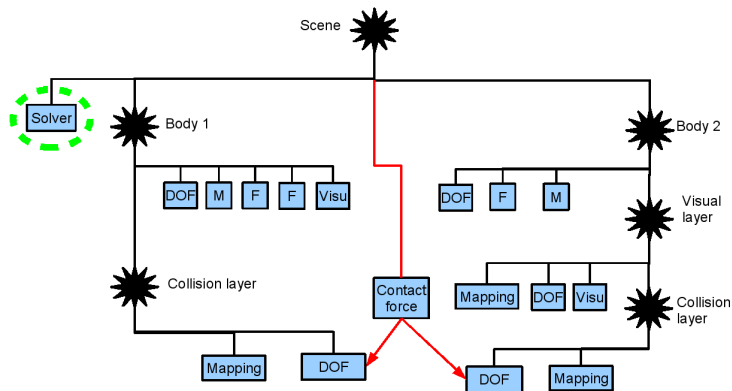
- ▶ Soft interactions : independent processing, no synchronization required

ODE solution of interacting bodies



- ▶ Soft interactions : independent processing, no synchronization required
- ▶ Stiff interactions : unified implicit solution, synchronized objects

ODE solution of interacting bodies



- ▶ Soft interactions : independent processing, no synchronization required
- ▶ Stiff interactions : unified implicit solution, synchronized objects
- ▶ Hard interaction constraints : work in progress (*Christian Duriez, ALCOVE*)

Outline

Motivation

Simple bodies

Layered bodies

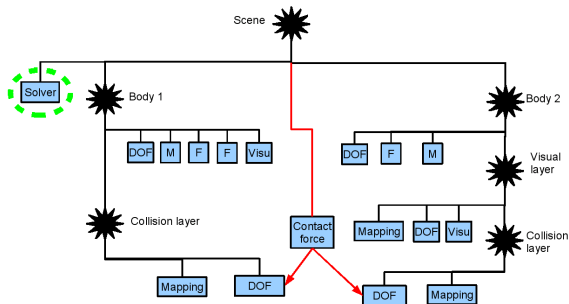
 Mappings

Interacting bodies

Data processing

Conclusion

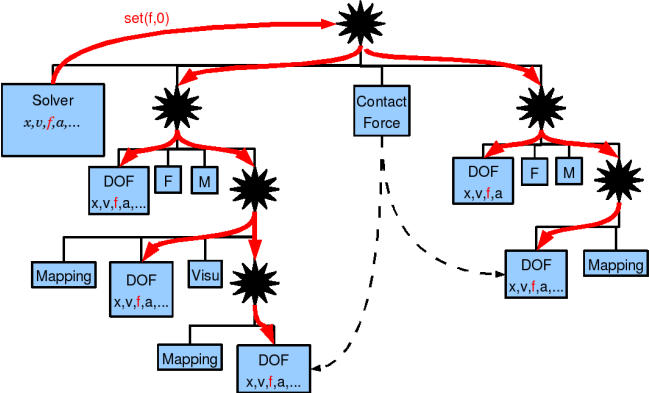
Actions



- ▶ No global state vector
- ▶ Action = graph traversal + global vector ids + call of abstract top-down and bottom up methods

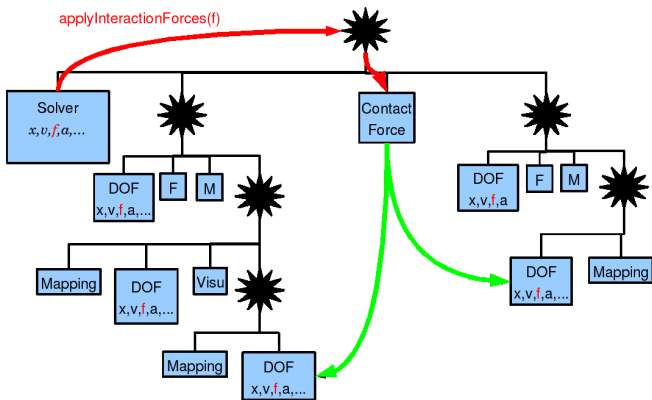
Example : clearing a global vector

- ▶ The solver triggers an action starting from its parent system and carrying the necessary symbolic information
- ▶ the action is propagated through the graph and calls the appropriate methods at each DOF node



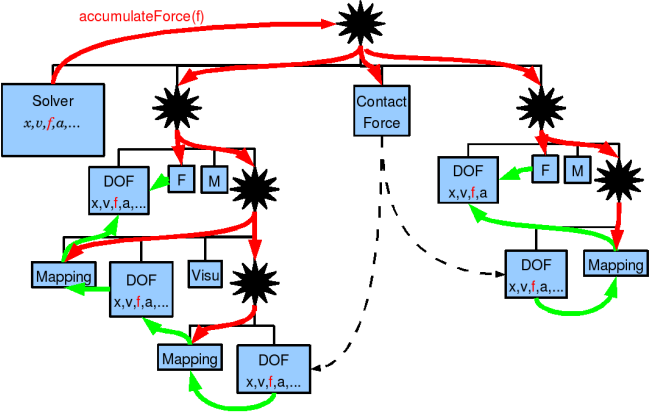
Example : applying interaction forces

- ▶ The solver triggers the appropriate action
- ▶ the action is propagated through the graph and calls the appropriate methods at each Contact node



Example : accumulating the forces

- ▶ The solver triggers the appropriate action
- ▶ the action is propagated through the graph and calls the appropriate (bottom-up) methods at each Force and Mapping node



Efficient implicit integration

- ▶ Large time steps for stiff internal forces and interactions
- ▶ solve $(\alpha M + \beta h^2 K)\Delta v = h(f + hKv)$ Iteratively using a conjugate gradient solution

Actions :

- ▶ propagateDx
- ▶ computeDf
- ▶ vector operations
- ▶ dot product (only global value directly accessed by the solver)

Ongoing work : building global vectors and matrices also

Efficiency

- ▶ No global state vector
 - ▶ they are scattered over the DOF components
 - ▶ each DOF component can be based on its own types (e.g. Vec3, Frame, etc.)
 - ▶ symbolic values are used to represent global state vectors
- ▶ Action = graph traversal + global vector ids + call of abstract top-down and bottom up methods
 - ▶ Displacements are propagated top-down
 - ▶ Interactions forces are evaluated after displacement propagation
 - ▶ Forces are accumulated bottom-up
 - ▶ Branches can be processed in parallel
 - ▶ virtual functions applied to components

Outline

Motivation

Simple bodies

Layered bodies

 Mappings

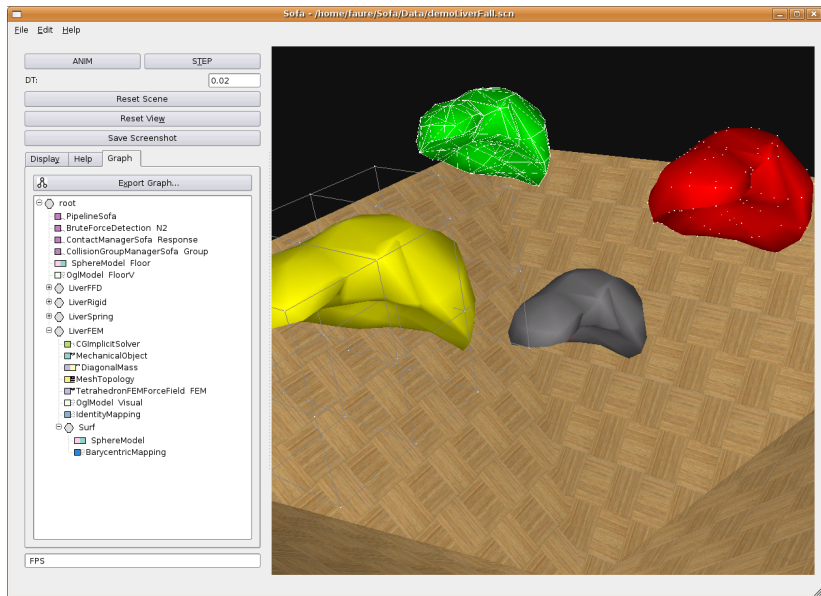
Interacting bodies

Data processing

Conclusion

Current stage

About 60 man.month, 20000 C++ code lines



Efficient coupling

High modularity :

- ▶ Abstract components : DOF, Force, Constraint, Solver, Topology, Mass, CollisionModel, VisualModel, etc.
- ▶ Arbitrary DOF types can coexist in the same scene

Efficiency :

- ▶ global vectors and matrices are avoided
- ▶ parallel processing is allowed

Implementation :

- ▶ currently 20000 C++ lines
- ▶ Windows, Linux
- ▶ Qt or FLTK user interfaces
- ▶ XML file format

Ongoing work

- ▶ More people : ETHZ, ...
- ▶ More algorithms : cutting (*Hervé Delingette, ASCLEPIOS*), interfaces (*François Faure, EVASION*),...
- ▶ More schedulers : asynchronous simulation/rendering/haptic feedback (*Jeremie Allard, Cimit*)
- ▶ More brute force : parallelization on PC cluster (*Everton Hermann, LIG/LJK*)
- ▶ More visual performance : coupling to a good render engine (*Pierre-Jean Bensoussan, ALCOVE*)
- ▶ More documentation (*everybody...*)
- ▶ Licensing

www.sofa-framework.org