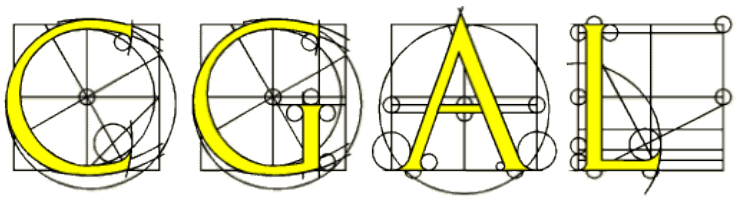
Modèle topologique générique pour la simulation physique

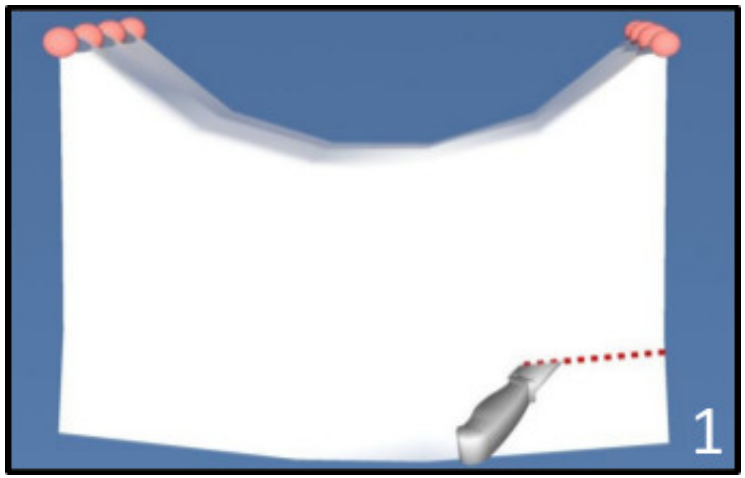
Guillaume Damiand, Elsa Fléchon, Fabrice Jaillet, Florence Zara

Université de Lyon, CNRS, LIRIS, UMR5205, F-69622 France
http://liris.cnrs.fr
http://www.cgal.org



8 Juillet 2013

Contexte



Simulation physique : besoins

- Décrire les objets subdivisés : cellules, incidences
- Traiter les changements topologiques
- Contrôler la validité des objets
- Mélanger différent type de cellules

Modèles existants

B-Rep pour la modélisation géométrique

Représentation des objets par leurs bords
- flexible : nombreuses opérations locales
- cellules non régulières

Structure de données B-Rep

- Une des plus utilisée : Halfedge data structure (HDS)
  - seulement 2D
- Généralisation dD : les **cartes combinatoires**

CGAL : Computational Geometry Algorithms Library

Bibliothèque C++ de géométrie algorithmique

Nombreux avantages :

- importante : nombreuses structures de données et algorithmes
- libre : LGPL3+/GPL3+
- reconnue : utilisateurs, communauté
- robuste : prédicats et/ou noyaux exacts

Principales structures de données pour représenter des objets

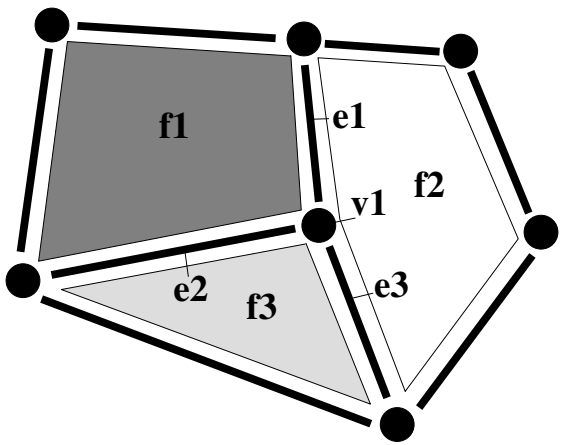
	2D	3D	*n*D
Simplicial	TDS_2 ✓	TDS_3 ✓	≈
Cellulaire	HDS & Polyhedron_3 ✓	✗	✗

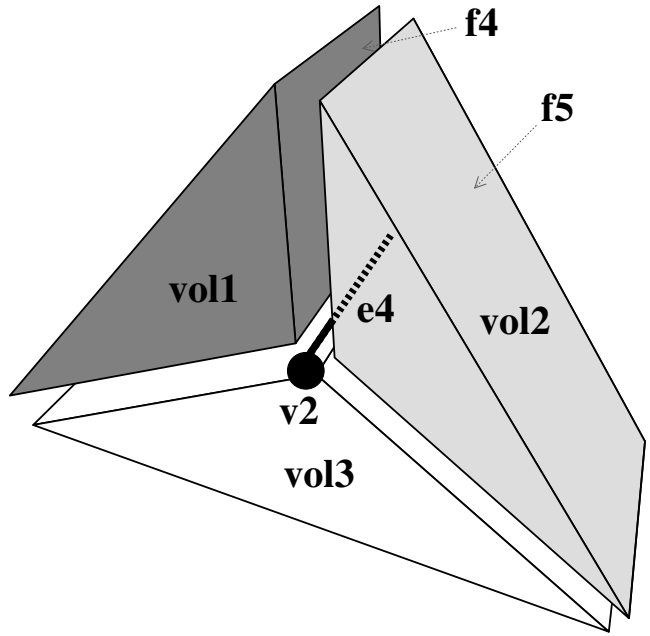
Plan

1. Les Cartes Combinatoires

Objectif

Représenter des objets dD orientables subdivisés en **cellules**

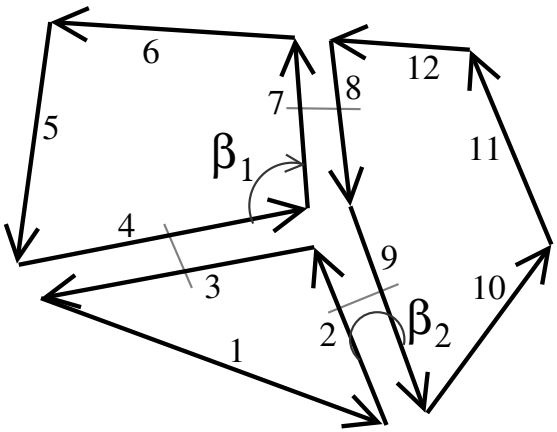


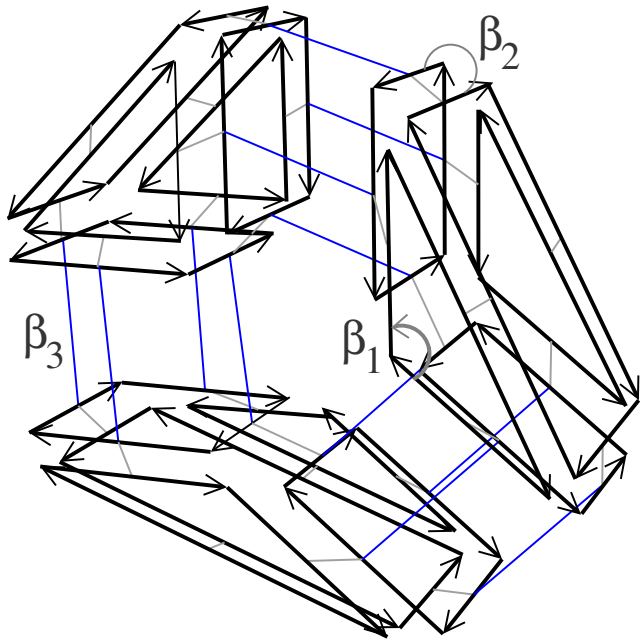


et relations “topologiques” entre les cellules :
incidence et **adjacence**

Solution

Représenter les objets par leur bords : **brins** et **relations**





Un brin : un morceau d'arête orientée et de chaque i-cellule incidente

Définition Générique

Definition (n-map)

Soit $d \geq 0$. Une d-carte combinatoire, (ou d-carte) est une algèbre $C = (B, \beta_1, \ldots, \beta_d)$ où :

1. B est un ensemble fini de brins ;
2. β_1 est une *permutation partielle* sur B ;
3. $\forall i : 2 \leq i \leq d : \beta_i$ est une *involution partielle* sur B ;
4. $\forall i, j : 1 \leq i < i + 2 \leq j \leq d : \beta_i \circ \beta_j$ est une *involution partielle*.

- permutations/involutions **partielles** : objets à bords, brins liés à $\varnothing$
- contraintes de cohérence : *analogue combinatoire* des variétés topologiques

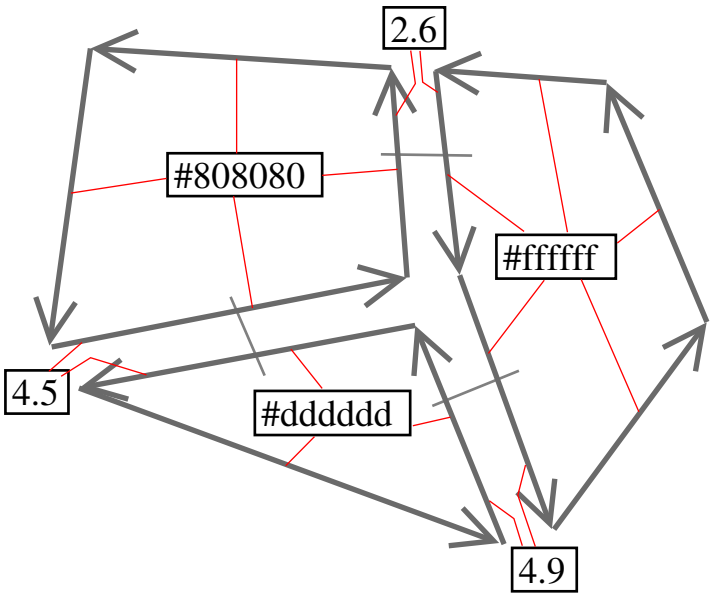
Cellules et Attributs

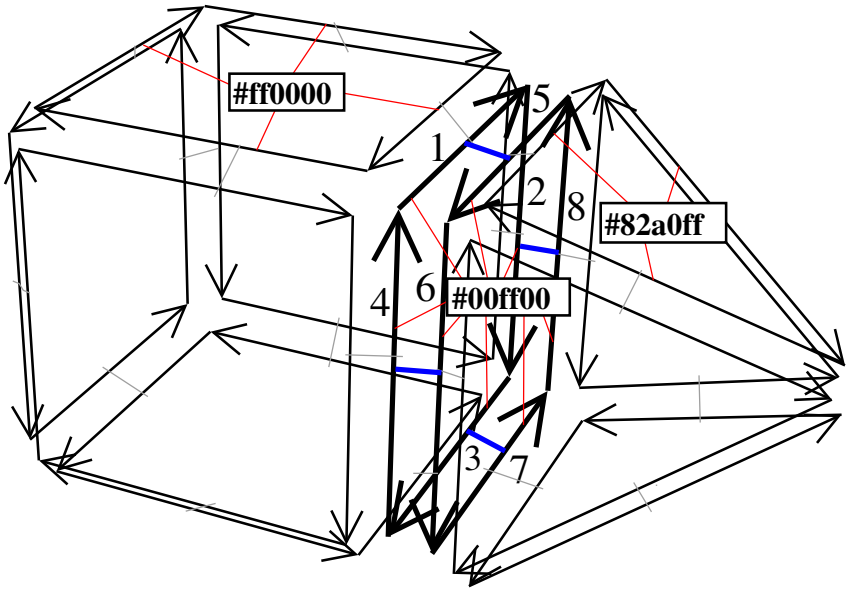
Chaque cellule : ensemble de brins (notion d'orbite)

ex. en 2D : 2-cellule = $\langle \beta_1 \rangle$, en 3D : 2-cellule = $\langle \beta_1, \beta_3 \rangle$

Pour associer des **informations** aux cellules

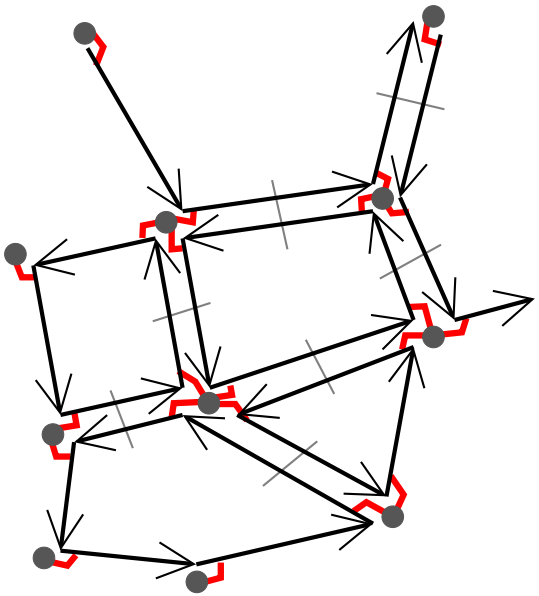
i-attributs associés aux i-cellules

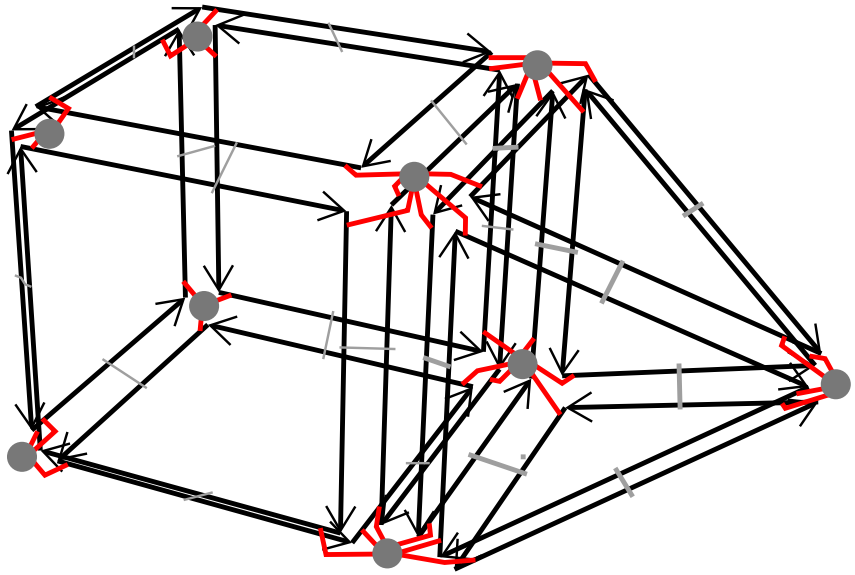




Plongement des Cartes

Pour associer de la “géométrie” aux cartes
Chaque sommet doit avoir un point associé (en dimension $d2$)
$\Rightarrow$ c’est “juste” un 0-attribut particulier contenant un point





Attention : d et $d2$ ne sont pas forcément liés
On parle de **complexe cellulaire linéaire**

2. Modules CGAL Développés

Structure des Modules

2 modules : `Combinatorial map` et `Linear cell complex`

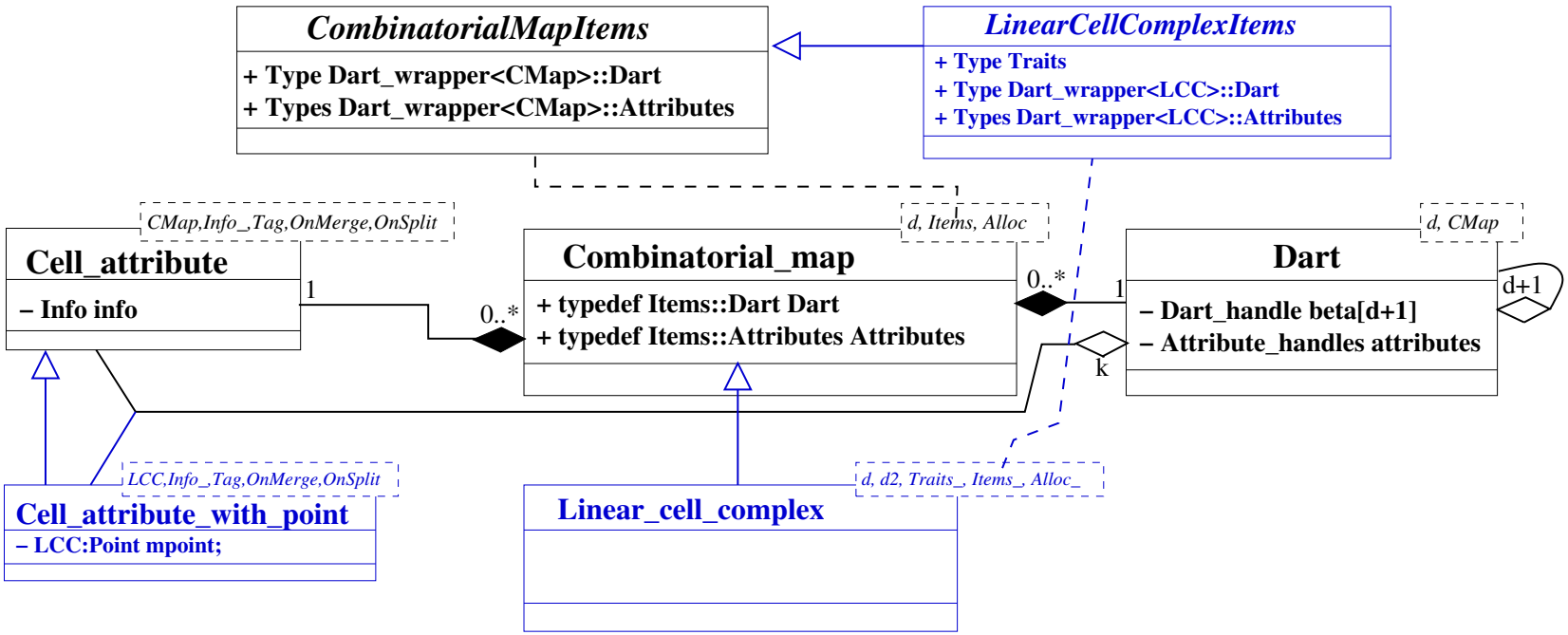


Diagramme UML des principales classe des modules `Combinatorial map` en noir et `Linear cell complex` en bleu. k est le nombre d'attributs *non void*.

Exemples de définition de complexes cellulaires linéaires

```
typedef CGAL::Linear_cell_complex<2> Plane_graph;
typedef CGAL::Linear_cell_complex<2,3> Polyhedron_3;
typedef CGAL::Linear_cell_complex<3> LCC_3;
```

```
struct  Myitem {
  template<class CMap>
   struct  Dart_wrapper {
     typedef CGAL::Dart<3, CMap> Dart;
     typedef CGAL::Cell_attribute_with_point< CMap, int> Myattr0;
     typedef CGAL::Cell_attribute< CMap, double> Myattr2;
     typedef CGAL::cpp0x::tuple<Myattr0,void,Myattr2> Attributes;
   };
};
typedef CGAL::Exact_predicates_inexact_constructions_kernel EPIC;
typedef CGAL::Linear_cell_complex_traits<3,EPIC> mytraits;
typedef CGAL::Linear_cell_complex<3,3,Mytraits,Myitem> My_lcc_3;
```

Exemples de définition de complexes cellulaires linéaires

```
typedef CGAL::Linear_cell_complex<2> Plane_graph;
typedef CGAL::Linear_cell_complex<2,3> Polyhedron_3;
typedef CGAL::Linear_cell_complex<3> LCC_3;
```

```
struct  Myitem {
  template<class CMap>
   struct  Dart_wrapper {
     typedef CGAL::Dart<3, CMap> Dart;
     typedef CGAL::Cell_attribute_with_point< CMap, int> Myattr0;
     typedef CGAL::Cell_attribute< CMap, double> Myattr2;
     typedef CGAL::cpp0x::tuple<Myattr0,void,Myattr2> Attributes;
   };
};
typedef CGAL::Exact_predicates_inexact_constructions_kernel EPIC;
typedef CGAL::Linear_cell_complex_traits<3,EPIC> mytraits;
typedef CGAL::Linear_cell_complex<3,3,Mytraits,Myitem> My_lcc_3;
```

Opérations de Base

- Créations d'objets de base

  `lcc.make_segment`, `lcc.make_triangle`,
  `lcc.make_quadrangle`, `lcc.make_tetrahedron`,
  `lcc.make_hexahedron`

- Conversion à partir d'autres types CGAL

  `import_from_triangulation_3(lcc,atr)`,
  `import_from_polyhedron_3(lcc,ap)`

Parcours

Utilisation du concept de `Range` (remplace couple d'itérateurs begin/end)

3 intervalles de tous les brins :

- `Dart_range` tous les brins
- `Dart_of_orbit_range<Beta...>` tous les brins de l'orbite `<Beta...>(d0)` d'un brin `d0` donné
- `Dart_of_cell_range<i>` tous les brins de la i-cellule contenant un brin donné

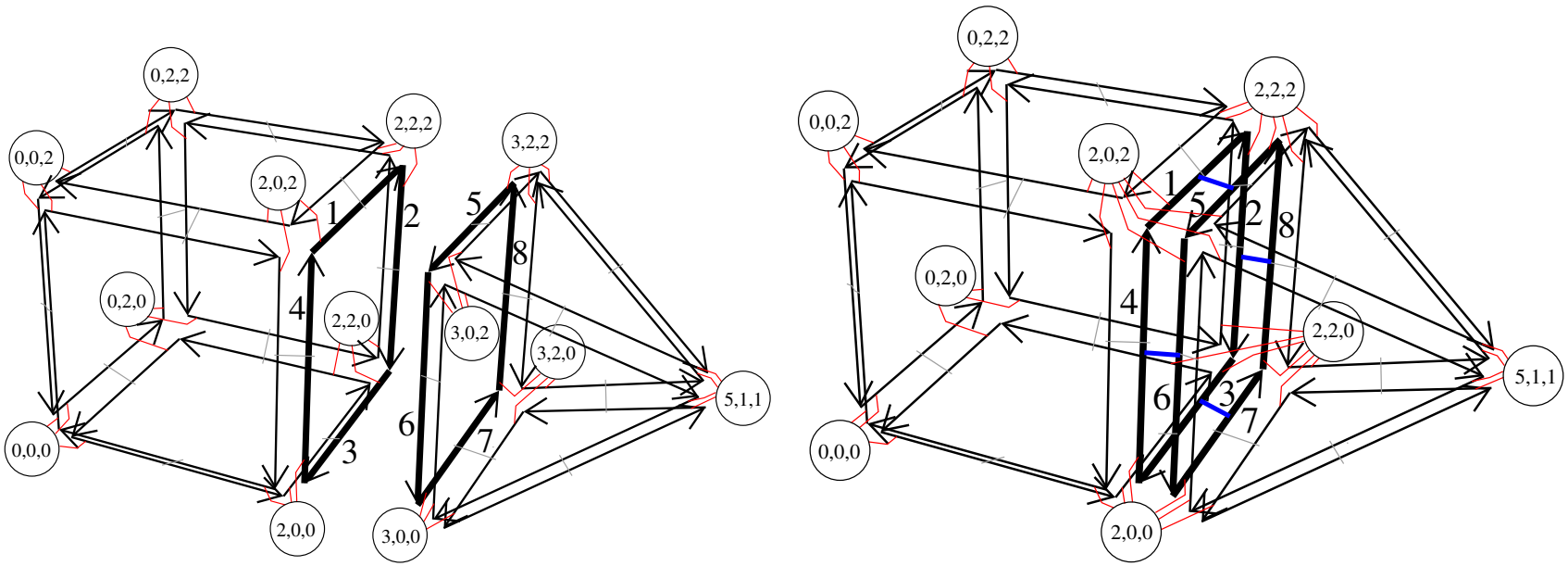
2 intervalles d'un brin par i-cellule :

- `One_dart_per_cell_range<i>` un brin de chaque i-cellule
- `One_dart_per_incident_cell_range<i,j>` un brin de chaque i-cellule incidente à une j-cellule donnée

Coutures/Décousures

Objectif : coller deux i-cellules par identification de $(i-1)$-cellules

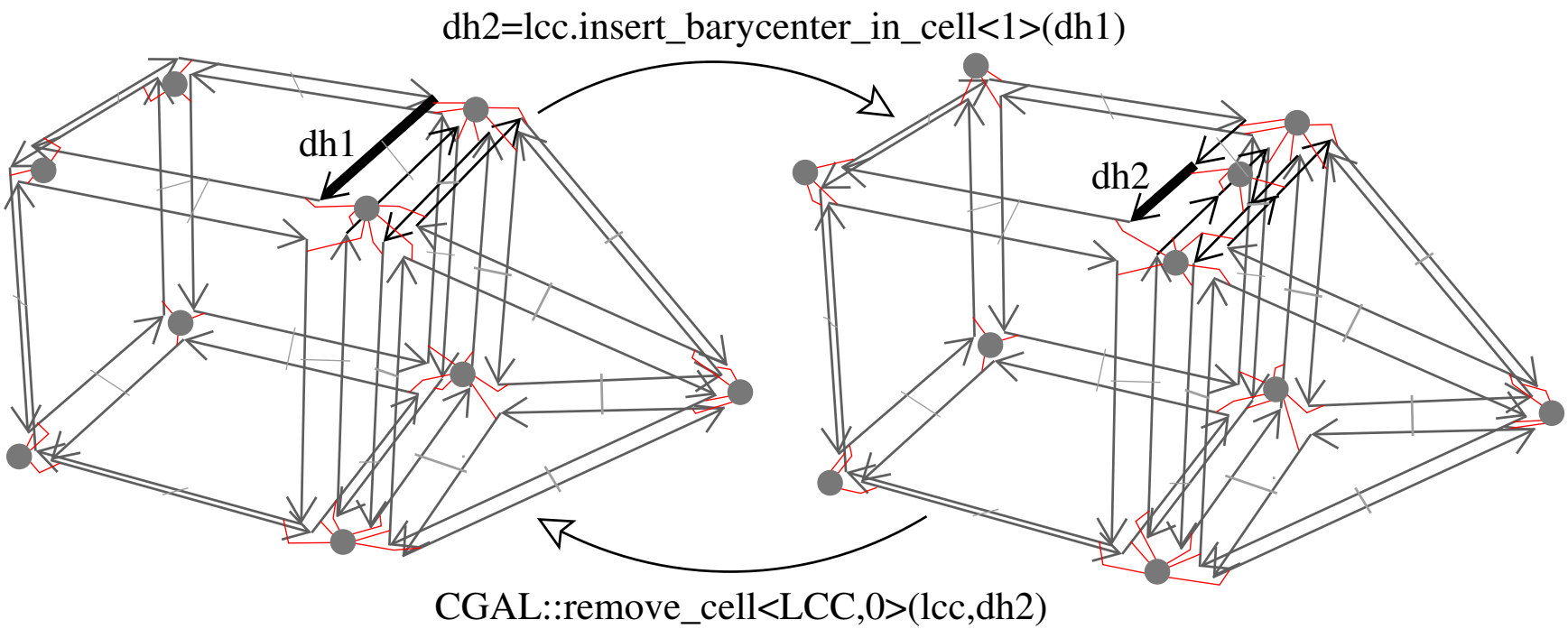
Sens $\rightarrow$: `sew<3>(1,5)` (ou `sew<3>(2,8)`, `sew<3>(3,7)`, `sew<3>(4,6)`)
Sens $\leftarrow$: `unsew<3>(1)` (ou `unsew<3>(5)`, `unsew<3>(3)...`)



Insertion/Suppression de Cellules

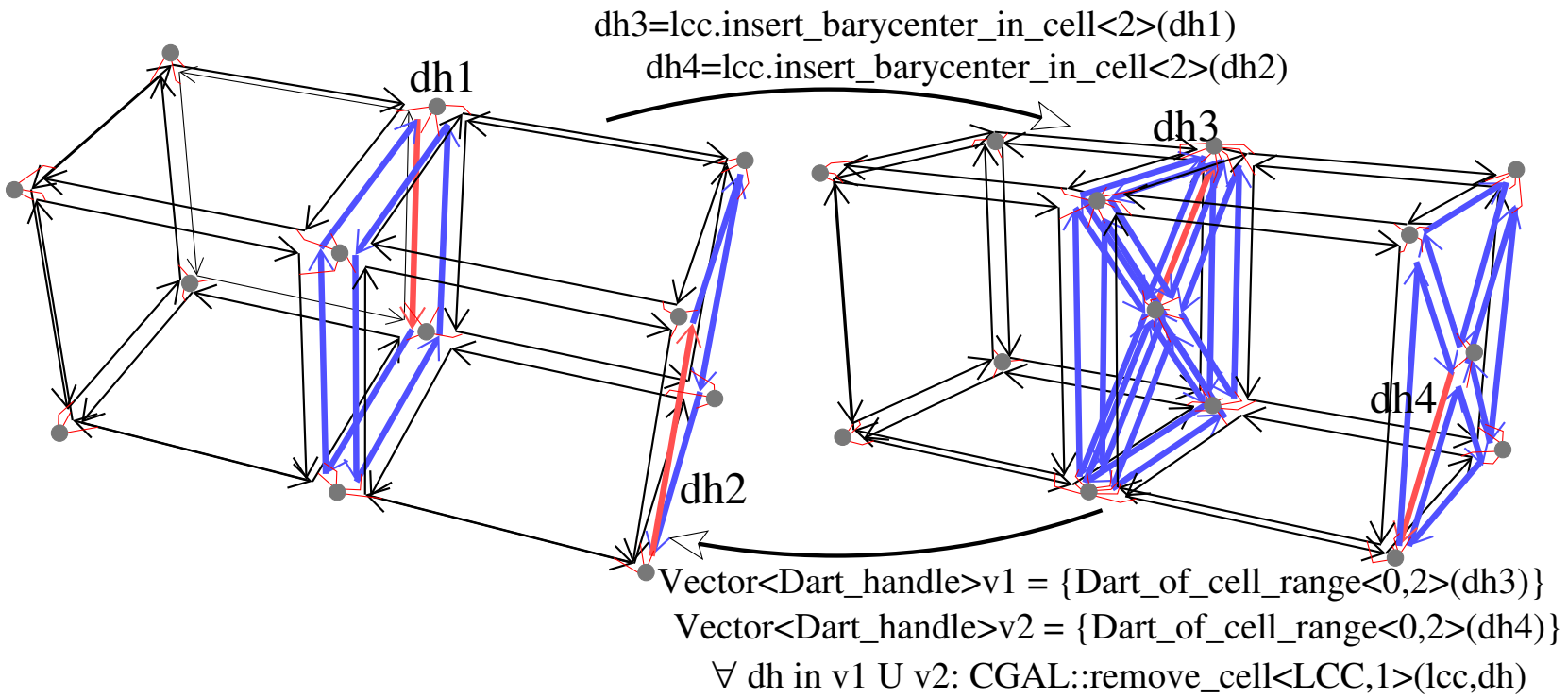
Suppression : 1 seule opération générique `remove_cell<i>`
Insertion : 1 opération par dimension (différents paramètres)

Insertion d'un sommet dans une arête (1-cellule)



Insertion/Suppression de Cellules

Insertion d'un sommet dans une face (2-cellule)



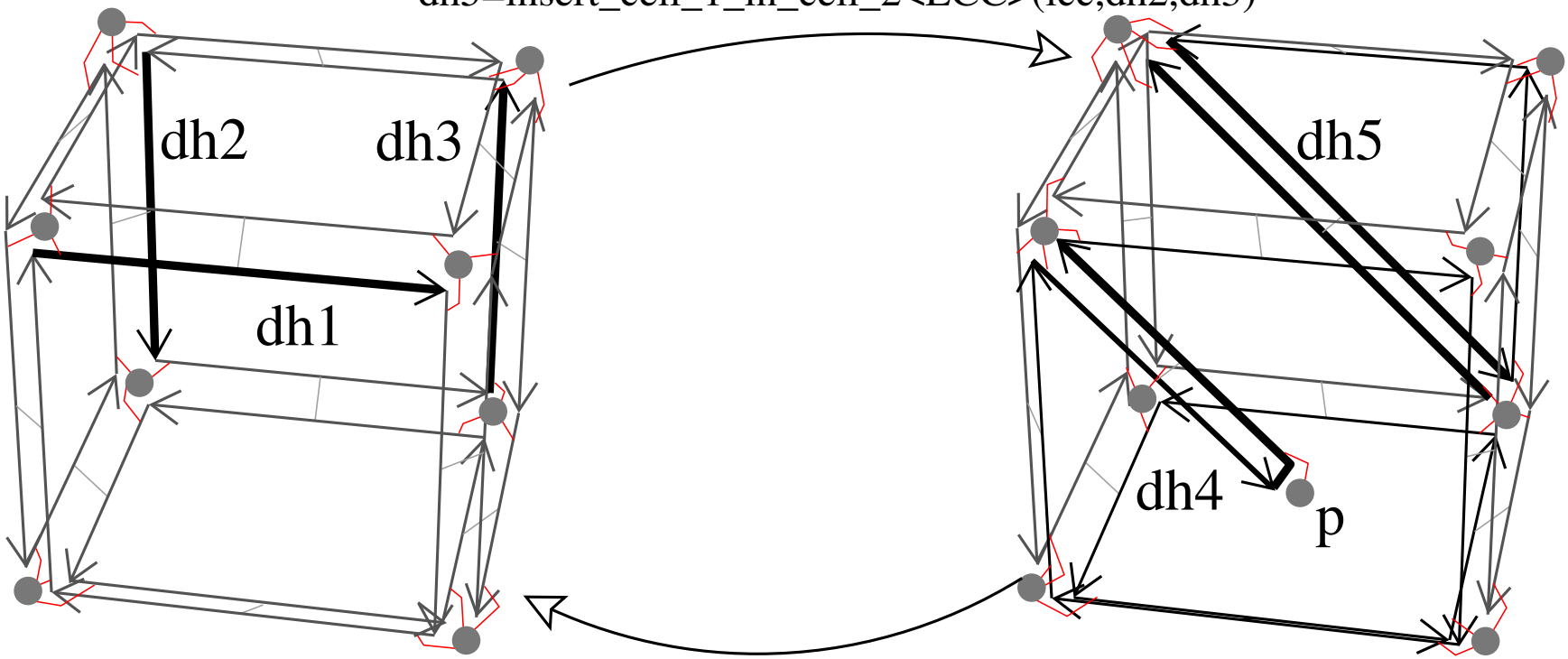
Insertion/Suppression de Cellules

Insertion d'un arête dans une face

entre deux sommets, ou une arête pendante

dh4=lcc.insert_dangling_cell_1_in_cell_2(dh1,p)

dh5=insert_cell_1_in_cell_2<LCC>(lcc,dh2,dh3)



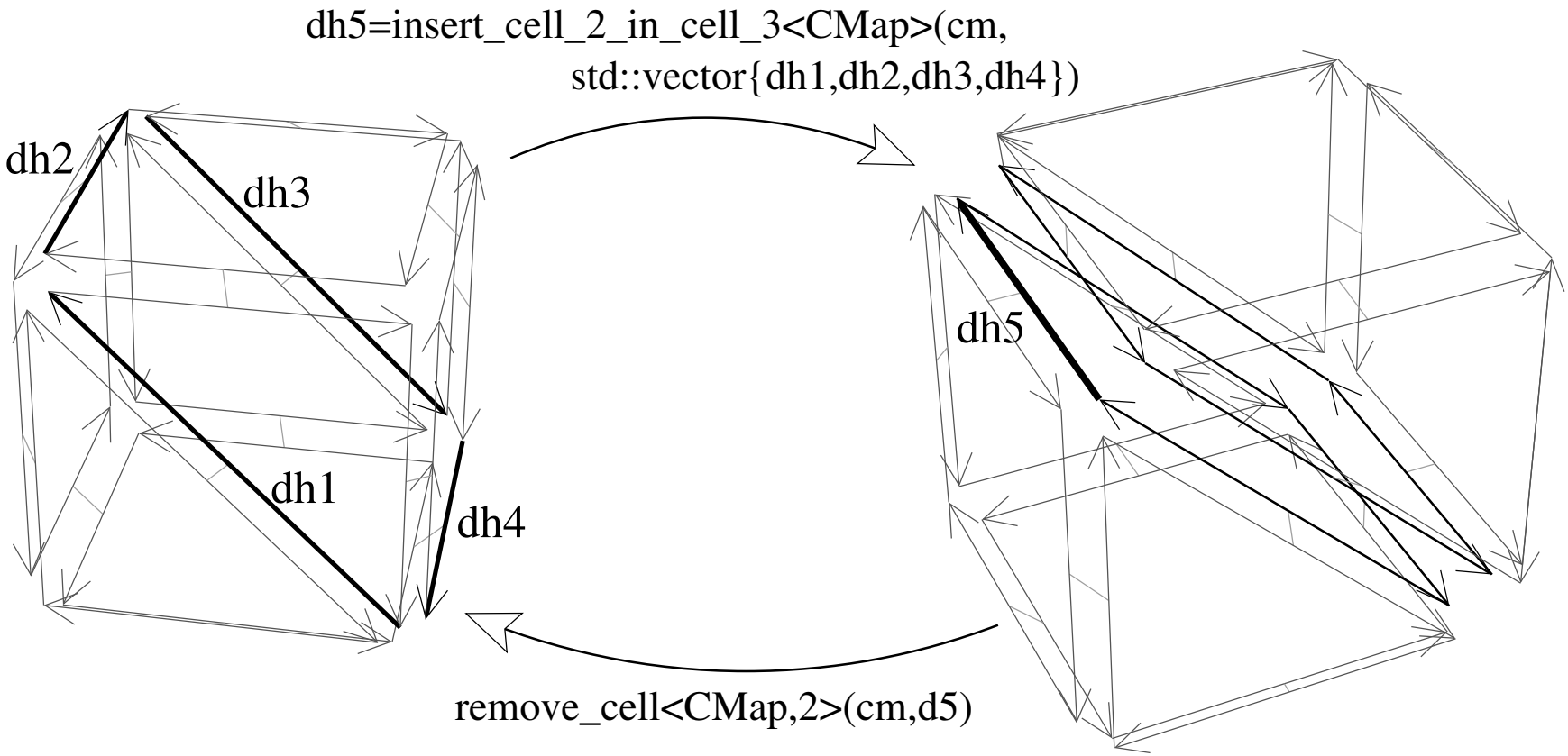
remove_cell<LCC,1>(lcc,dh4)

remove_cell<LCC,1>(lcc,dh5)

Insertion/Suppression de Cellules

Insertion d'une face dans un volume

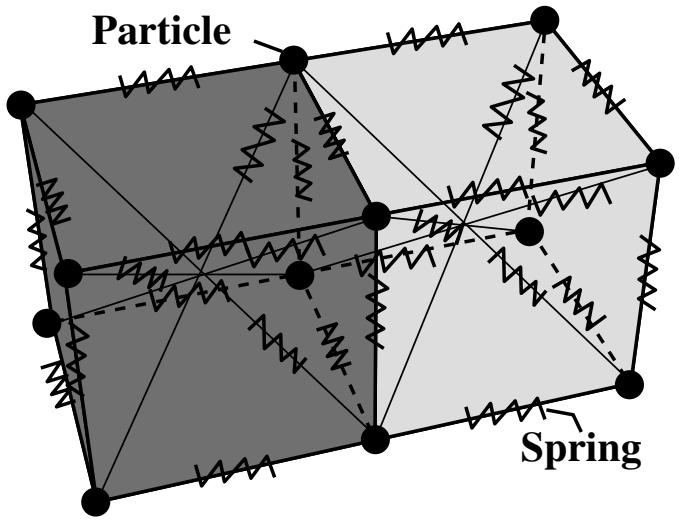
le long d'un chemin d'arêtes



3. Modèle Physique

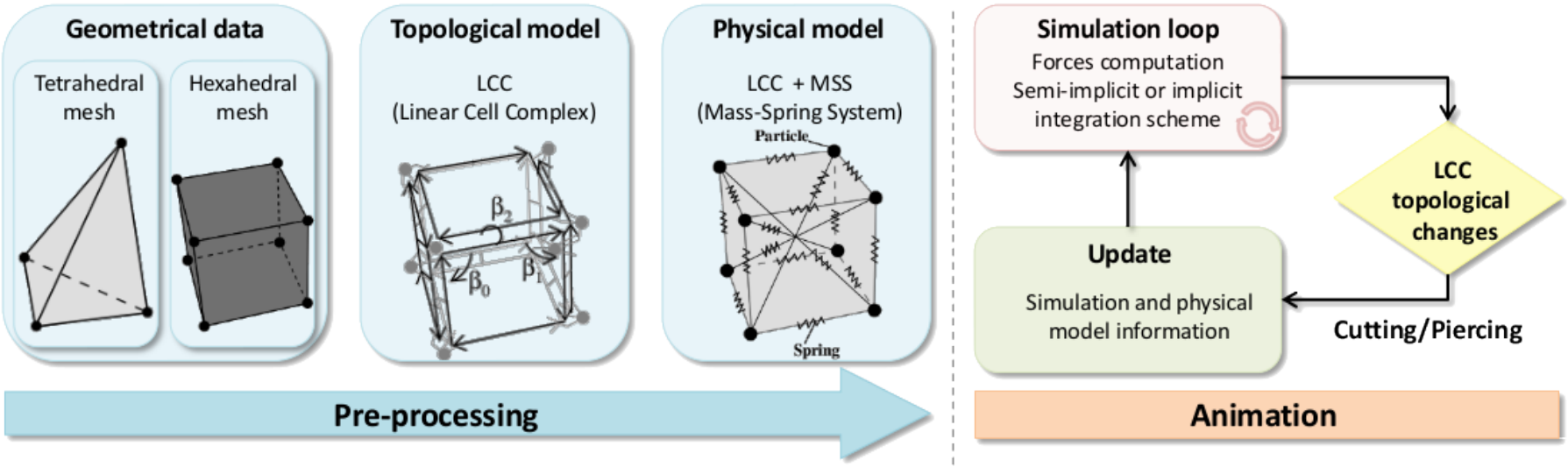
Mass-Spring System (MSS)

Physical model



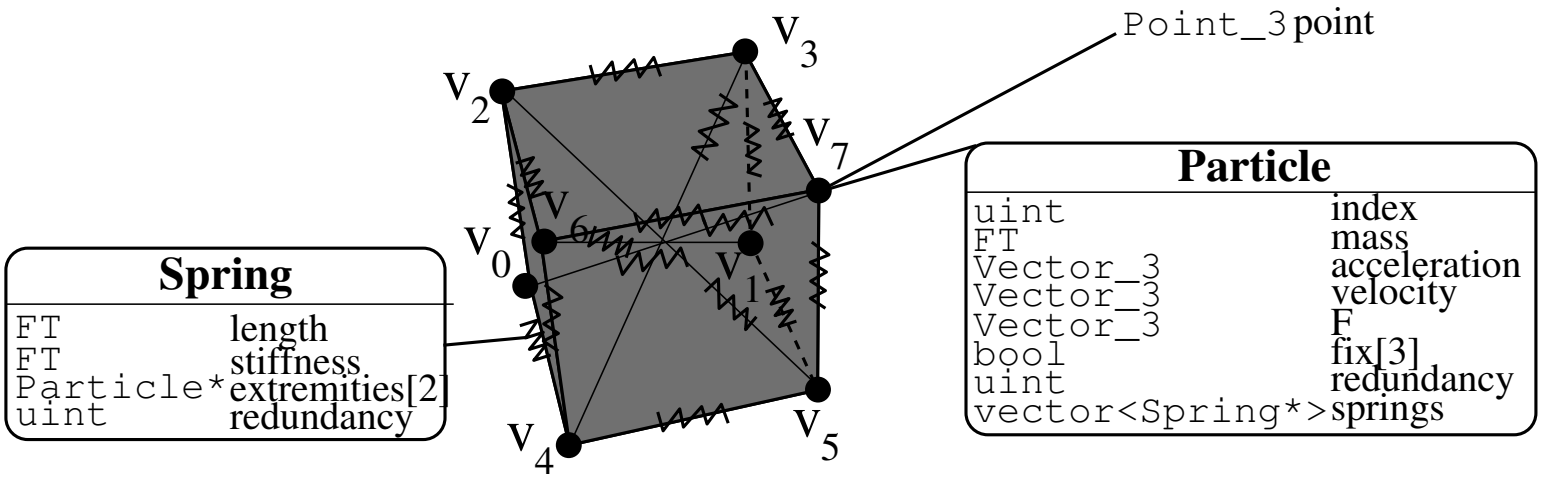
- Set of particles connected by springs
- Advantages
  - Computational efficiency
  - Algorithmic simplicity
  - Easily implemented on GPU
- Allow topological changes

Overview



- Input file describing geometrical information
- Create LCC
- Calculate and associate all physical information
- Simulation : accumulate forces and integration scheme
- Update information when topology change

Our hybrid model



- LCC + MSS
- Calculate and initialize information for simulation purposes
- Associate
  - $0-cell$ with Particle and 3D point
  - $1-cell$ with Spring

Simulation loop
Physical model

1. Computation and accumulation forces : external and internal

Spring forces

Force of spring between particle i and j

$$\vec{F}_{ij}(t) = \vec{F}^e_{ij}(t) + \vec{F}^v_{ij}(t)$$

Elasticity and viscosity force

$$\begin{cases} \vec{F}^e_{ij}(t) & = k_{ij}\left(\|\vec{P}_j(t) - \vec{P}_i(t)\| - l_{ij}\right)\vec{u}_{ij}(t) \\ \vec{F}^v_{ij}(t) & = \gamma_{ij}\left(\vec{V}_j(t) - \vec{V}_i(t)\right)\cdot\vec{u}_{ij}(t)\ \vec{u}_{ij}(t) \end{cases}$$

k_{ij} : stiffness, l_{ij} : initial length, $\gamma_{ij} = 2\sqrt{\frac{m_i+m_j}{2}\ k_{ij}}$: damping coefficient

$\vec{P}_i(t)$ and $\vec{V}_i(t)$: position and velocity of particle i ; $\vec{u}_{ij}(t) = \frac{\vec{P}_j(t)-\vec{P}_i(t)}{\|\vec{P}_j(t)-\vec{P}_i(t)\|}$

Simulation loop

Physical model

Euler semi-implicit

2. Calculate acceleration with Newton's law $m_i \frac{d^2}{dt^2}\mathbf{P}_i(t) = \mathbf{F}_i(t)$
3. Integrate to obtain velocity and position

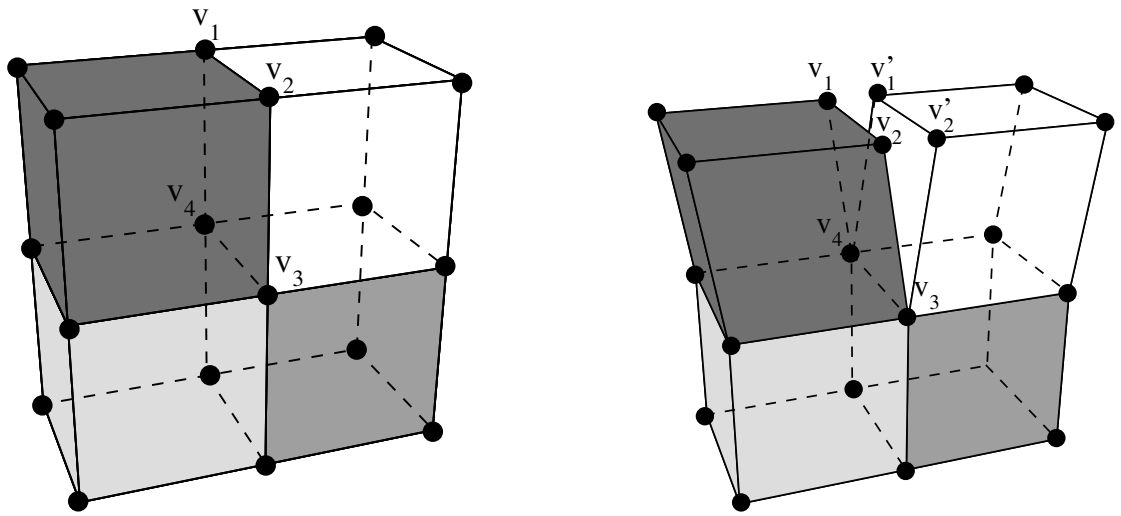
$$\begin{cases} \frac{d}{dt}\vec{P}_i(t+h) & = & \frac{d}{dt}\vec{P}_i(t) + h\frac{d^2}{dt^2}\vec{P}_i(t) \\ \vec{P}_i(t+h) & = & \vec{P}_i(t) + h\frac{d}{dt}\vec{P}_i(t+h) \end{cases}$$

Euler implicit

2. For each spring, fill stiffness matrix : $\partial F/\partial P$
   - Conjugate gradient method
3. Calculate velocity and integrate to obtain position

$$\left(M - h\frac{\partial F(t))}{\partial V(t)} - h^2\frac{\partial F(t)}{\partial P(t)}\right)\Delta V = h\,F(t) + h^2\frac{\partial F(t)}{\partial P(t)}V(t)$$

Topological change
Cutting

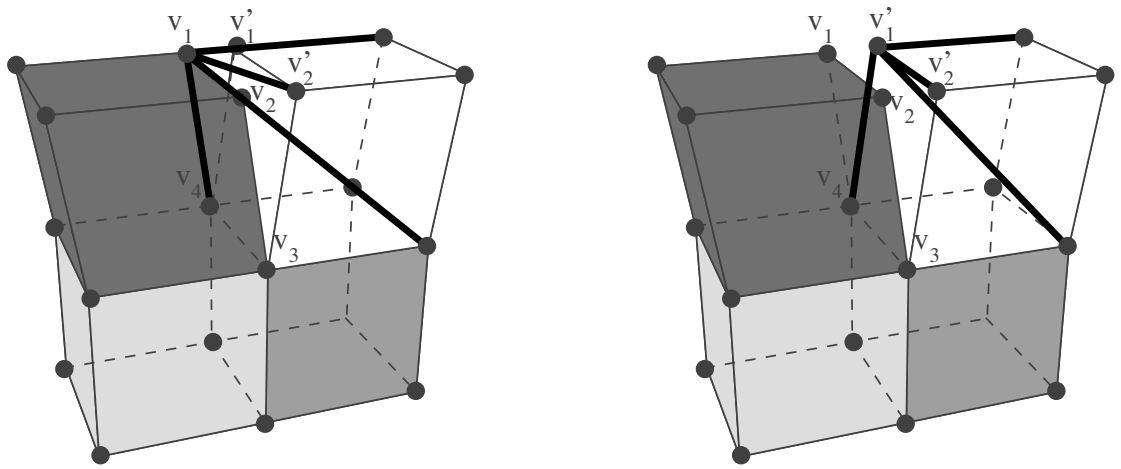


Cutting along faces of mesh

Unglue two $3-cells$ along $2-cell$

- Unlink β_3 pointers
- Duplicate
  - $0-cell$: v_1 and v_2
  - $1-cell$: (v_4, v_1), (v_1, v_2) and (v_2, v_3)

Topological change
Cutting



- Create v_1' : information duplicated from v_1
- For each pair of splitted cells
  - $0-cell$: initialize index, update masses, diagonal springs...
  - $1-cell$: extremities of spring...
- Re-size data structure as global stiffness matrix

4. Conclusion / Travaux Futurs

Conclusion (CGAL)

Résultats

- Deux modules pour manipuler des d-cartes
  avec ou sans plongement
  `http://www.cgal.org/Pkg/CombinatorialMaps`
  `http://www.cgal.org/Pkg/LinearCellComplex`
- Générique (dimension) et paramétrables (brins/attributs)
- Itérateurs, opérations de base, modifications

Intérêts

- Permettre la factorisation de plusieurs codes existants
- Peut servir de brique de base pour des applications graphiques
  modeleur de bâtiment, segmentation d'images 2D et 3D...

Conclusion (Toposim)

Résultats

- Modèle hybride : MSS + LCC pour la simulation physique
- Boucle de simulation totalement intégrée
- Découpe topologique avec mise à jour locale des information

Intérêts

- Généricité : type de cellules et attributs
- Utilisation des relations d'incidences et d'adjacence
- Contrôle de la validité topologique des objets

Travaux Futurs (CGAL)

Enrichir les deux modules

- Ajouter de nouvelles opérations
- Étendre certaines fonctionnalités

  `save/load` générique, attributs dynamiques, utilisation d'index...

Porter nos logiciels

- Moka, CarteTopo2D et 3D

Nouveaux projets

- Analyse de vidéo...

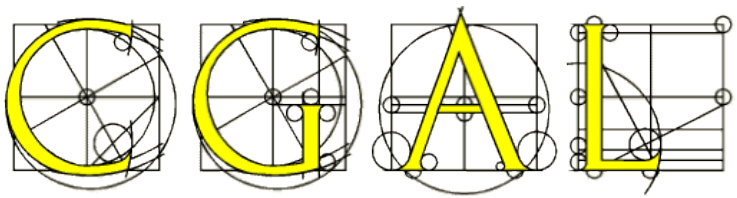
Travaux Futurs (Toposim)

- D'autres types de découpes (par ex. déchirure)
- Optimisations
- Parallélisation
- Gestion des cellules de topologie quelconques
- Modèles multi-échelle
- Raffinement/dé-raffinement automatique durant la simulation

Modèle topologique générique pour la simulation physique

Guillaume Damiand, Elsa Fléchon, Fabrice Jaillet, Florence Zara

Université de Lyon, CNRS, LIRIS, UMR5205, F-69622 France
http://liris.cnrs.fr
http://www.cgal.org



8 Juillet 2013