High Order Elements in Sofa

Hervé Delingette
INRIA Méditerranée, Sophia Antipolis, France

2014

1 High Order Triangular Meshes

1.1 Introduction

We define high order triangular elements in their Bernstein / Bezier form rather than Hermite or Lagrange forms. A high order triangular mesh is defined by :

- An underlying triangular mesh $\mathcal{M}$ consisting of a set of *"underlying triangle vertices"* $\mathcal{V}$, edges $\mathcal{E}$ and triangles $\mathcal{T}R$. We write V, the number of "triangles vertices", E the number of edges, TR the number of triangles.
- A set of control points $\mathcal{C}$ defined on each triangle which is used to defined the Bézier triangle.
- A set of trivariate Bernstein polynomials allowing to describe the value of a node anywhere on the mesh $\mathcal{M}$

In the remainder, the word *vertex* refers to a vertex of the underlying triangulation whereas the word *point* or *control points* refer to a control point of the Bézier triangle. A vertex is also a control point.

In terms of topology, a high order triangular mesh has more control points than triangle vertices. Below are examples of high order tetrahedral elements of various degree.

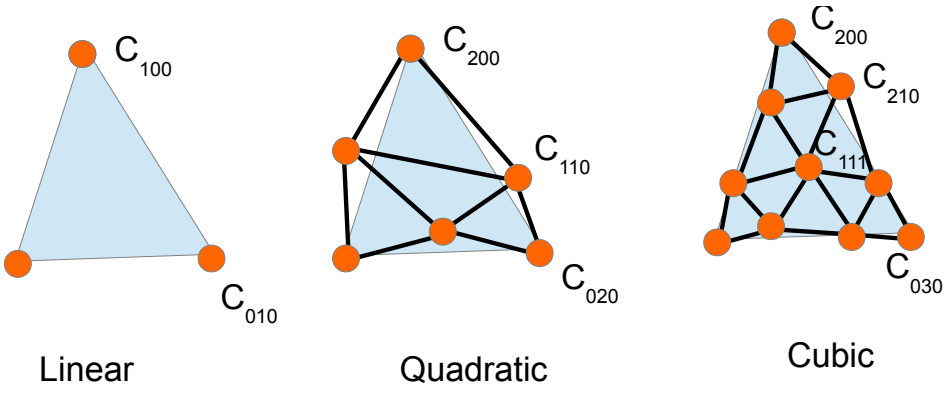

Figure 1: Linear ($d = 1$), Quadratic ($d = 2$) and Cubic ($d = 3$) Triangular Bezier triangles and notation of control points using Triangle Bezier indices

1.1.1 Number of Control Points

If $d > 0$ is the degree (or order) of a triangular Bézier element, then there are :

- V controls points that coincide with the *"underlying triangle vertices"*.
- $(d-1)E$ if $d > 1$ control points that are lying on edges.
- $\frac{(d-1)(d-2)TR}{2}$ if $d > 2$ control points that are lying inside triangles.

Thus the total number of control points are :

$$C = V + (d-1)E + \frac{(d-1)(d-2)TR}{2};$$

1.1.2 Triangle Bezier Indices

We use specific notations of control points inside a Bézier triangle which we call *Triangle Bezier Indices* (TRBI). A TRBI $p \in \mathbb{N}^+ \times \mathbb{N}^+ \times \mathbb{N}^+$ is a triplet of positive natural numbers that indicate their relative position in the Bézier triangle.

Thus for an element of degree d, $\mathbf{p} = (i, j, k)$ is such that $|\mathbf{p}| = i + j + k = d$. The three TRBI $(d, 0, 0)$, $(0, d, 0)$, $(0, 0, d)$ coincides with the 3 triangle vertices while $(0, i, j), i > 0, j > 0$ is lying on the edge opposite to the first vertex and $(i, j, k), i > 0, j > 0, k > 0$ is lying inside the triangle. We write $\mathbf{C_p}$ the control point associated with indices (i, j, k).

1.1.3 Trivariate Bernstein Polynomial

The control points are used to define a parametric surface in space. The parameters are the barycentric coordinates $(r, s, t$ such that $r+s+t = 1$ and $0 \leq r, s, t \leq 1$. The shape functions are the trivariate Bernstein polynomials $B^d_{i,j,k}(r, s, t)$ of degree d that are themselves parameterized by 3 indices (i, j, k) such that $i + j + k = d$ with the following expression:

$$B^d_{i,j,k}(r, s, t) = \frac{d!}{i!j!k!} r^i s^j t^k$$

For given degree d there are $N_d = 3 + 3 * (d-1) + (d-1) * (d-2)/2$ such polynomials. To simplify notations, we use the same Triangle Bezier Indices for the Bernstein polynomial as for the control points. Therefore $B^d_{\mathbf{p}}(r, s, t) = B^d_{i,j,k}(r, s, t)$.

1.1.4 Regular or Integral Bezier triangles

With this notation, the position of a point parameterized by (r, s, t) on a Bezier triangle is given by :

$$\mathbf{C}(r, s, t) = \sum_{|\mathbf{p}|=d} B^d_{\mathbf{p}}(r, s, t)\mathbf{C_p}$$

Any point on the Bezier triangle is a linear combination of its control points and lies within the convex hull of control points as all weights (Trivariate Bernstein Polynomials) are positive.

1.1.5 Rational Bezier triangles

A common extension of Bezier triangles is to consider weights $w_{\mathbf{p}}$ at each control points to change the importance of each control point in the definition of the surface. Each weight $w_{\mathbf{p}}$ of each control point is positive but may be greater than 1. The position of a point on a rational Bezier triangle is then :

$$\mathbf{C}(r,s,t) = \frac{\sum_{|\mathbf{p}|=d} B^d_{\mathbf{p}}(r,s,t) w_{\mathbf{p}} \mathbf{C}_{\mathbf{p}}}{\sum_{\|\mathbf{p}\|=d} B^d_{\mathbf{p}}(r,s,t) w_{\mathbf{p}}}$$

Rational Bezier triangles provide additional control over the shape and have the property to represent exactly any quadrics surface (rational cubics for cylinders and rational quartics for spheres).

1.2 SOFA Implementation of Bézier triangles

1.3 Layout of Degrees of Freedom in SOFA

In SOFA, the degrees of Freedom (DOF) are stored into a set of arrays inside objects called MechanicalState. We have chosen to store the $\mathcal{C}$ DOFs of a Bezier Triangle mesh inside a single MechanicalState. Furthermore, we provide a *default layout* for ordering those DOFs in a MechanicalState.

Figure 2 shows this ordering of control points. First of all, the control points associated with the underlying triangle vertices are stored, then those associated with edges, and triangles.

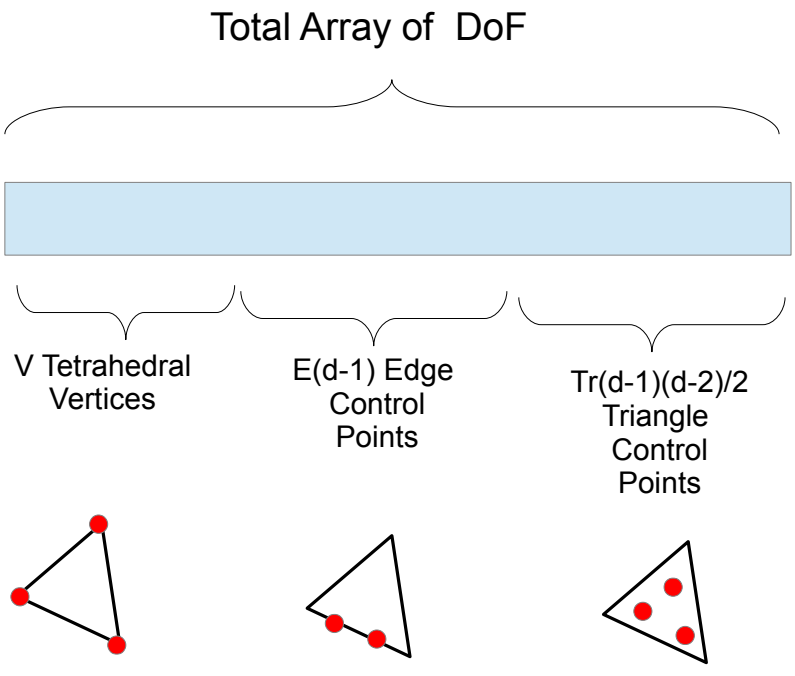

Figure 2: Layout of Degrees of Freedom of Bezier Triangle meshes inside a MechanicalState object

There are however some issues. In a triangle mesh, edges are not oriented (*e.g.* an edge is common to 2 triangles and is ordered differently among each triangle) and therefore the order

in the DoF array may not reflect the order in each triangle. It is the role of the BezierTriangleSetTopologyContainer class to provide a proper ordering.

Second, since there are several control points for a given edge, and triangle, it is important to specify the ordering inside each element. Figure3 show in a simplified way how the control points are numbered for an edge, and triangle. In a nutshell, the element are stored in increasing lexicographic order of its barycentric coordinates. Thus on a quartic Bézier triangle (order 4), the points of barycentric coordinates $(1/4, 3/4), (1/2, 1/2), (3/4, 1/4)$ are stored in this order.

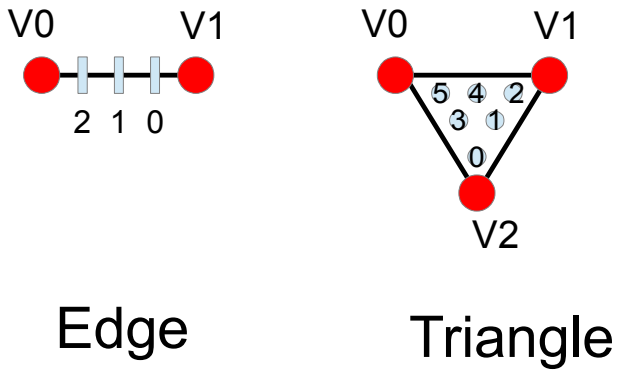

Figure 3: Layout of Degrees of Freedom of Bezier Triangle meshes inside a MechanicalState object

If this default layout is not respected, then a specific lookup table should be provided in BezierTriangleSetTopologyContainer in order to specify the location of each DOF.

Note that for rational Bézier triangles the weights are not considered as degrees of freedom but are specified at the initialization stage.

1.4 BezierTriangleSetTopologyContainer Class

This class provides an interface to the array of DoFs stored in the MechanicalState object. More precisely, a control point can specified in 3 different ways :

- A *global index* which is its rank in the array of size C.
- A triangle index and a TRBI, Triangle Bezier Index (like $(2, 0, 2)$ on a triangle of order 4).
- A triangle index and its *local index*. The local index is an integer between 0 and N_d which gives the rank of the control points in this local (tetrahedron specific) list of control points. For a quadratic triangle $N_d = 6$ and $N_d = 3 + 6 + 1 = 10$ for a cubic triangle.

The class thus provides functions to move from one indexing representation of a control point to another.

If the default layout is not used, then the 2 protected maps locationToGlobalIndexMap and globalIndexToLocationMap provide the mapping between global DOFs index and its location in the Bezier triangulations. globalIndexToLocationMap is a multimap as the same control point may belong to several triangles.

The function *void getGlobalIndexArrayOfBezierPointsInTriangle(const size_t triangleIndex, VecPointID & indexArray)* is particularly useful as it outputs an array of N_d global indices of

the control points located inside a given triangle. The triangle Bezier indices of those control points are always the same for a given degree d and is given by the array returned by the function *sofa::helper::vector<TriangleBezierIndex> getTriangleBezierIndexArray() const;*. Note that this function works if the default layout is not used.

Obviously the class can return the degree of Bezier triangles (they are all of the same degree). Note that it provides both the total number C of control points *getNbPoints*() and the number V of triangle points as *getNumberOfTriangularPoints()* .

Also the class provides the weights associated with each control point (getWeight(int i)) and a boolean indicating if a given triangle is rational or integral (isRationalSpline(int i)).

Given a global index of a control point, it is possible to know in which triangle it is associated and where it is located within the triangle. The function *void getLocationFromGlobalIndex(const size_t globalIndex, BezierTrianglePointLocation &location, size_t &elementIndex, size_t &elementOffset) ;* returns the location (point, edge or triangle) of a control point, the index of the triangle (elementIndex) and the rank within its location.

Efficient internal data structures are built to move from one representation to another.

1.5 BezierTriangleSetGeometryAlgorithms Class

This class provides basic geometric functions :

- *Coord computeNodalValue(const size_t triangleIndex,const Vec3 barycentricCoordinate);* which computes the position / value of a node given the index of a triangle and its barycentric coordinate.
- *Real computeBernsteinPolynomial(const TriangleBezierIndex tbi, const Vec3 barycentricCoordinate);* which computes $B^d_{\mathbf{p}}(r,s,t)$.

The Bernstein coefficients of a given degree are precomputed and basic visualization of control points are provided if *drawControlPointsEdges*, *drawSmoothEdges* or *drawControlPoints* are set to true.

1.6 File Format

There does not exist any standard file format for Bezier triangle. Thanks to standard layout of DOfs, it is possible to use classical file format for triangulation where the number of vertices is equal to the number of control points C which is greater than the number V of vertices.

This is used for instance with the MSH format with the file stored in bezierTriangle.msh. Note that not file format has been used so far for rational Bezier triangles.

1.7 Mesh2BezierTopologicalMapping Class

This class provide a way to create a Bezier triangle mesh of any order given a (linear) triangle mesh. The control points are simply linearly interpolated from the triangle vertices : a control point of index $\mathbf{p} = (i,j,k)$ is computed with the barycentric coordinates $(\frac{i}{d}, \frac{j}{d}, \frac{k}{d})$. The mapping automatically sets the degree and the number of tetrahedral vertices in the object BezierTriangleSetTopologyContainer.

Note that the position of the mapping in the scene has to be after BezierTriangleSetTopologyContainer (see the example in BezierTriangleTopologicalMapping.scn)

1.8 Bezier2MeshTopologicalMapping and Bezier2MeshMechanicalMapping Class

Those 2 components creates a triangular mesh from a Bezier triangle by performing a regular tessellation of the triangle. The level of tessellation is controlled by the *tesselationTriangleDegree* data. For instance one single Bezier triangle of degree d may be tessellated into $(l+1)^2$ subtriangles if l is the tessellation degree.

The Bezier2MeshTopologicalMapping updates directly the TriangleSetTopologyContainer whereas the Bezier2MeshMechanicalMapping controls its associated MechanicalState.

This is the main method to visualize a Bezier triangulations as the created triangles may be visualized for instance with a regular OglModel.

This mapping also maps forces applied on the tessellated triangulation back to the Bezier triangle and therefore allows to use regular triangulation forcefield on Bezier triangles.

An example is provided in Bezier2MeshMechanicalMapping.scn

2 High Order Tetrahedral Meshes

2.1 Introduction

We define high order tetrahedral elements in their Bernstein / Bezier form rather than Hermite form. A high order tetrahedral mesh is defined by :

- An underlying tetrahedral mesh $\mathcal{M}$ consisting of a set of *"tetrahedron vertices"* $\mathcal{V}$, edges $\mathcal{E}$, triangles $\mathcal{T}R$ and tetrahedra $\mathcal{T}$. We write V, the number of "tetrahedron vertices", E the number of edges, TR the number of triangles and TE the number of tetrahedra.
- A set of control points $\mathcal{C}$
- A set of quadrivariate Bernstein polynomials allowing to describe the value of a node anywhere on the mesh $\mathcal{M}$

In terms of topology, a high order tetrahedral mesh has more control points than tetrahedron vertices. Below are examples of high order tetrahedral elements of various degree.

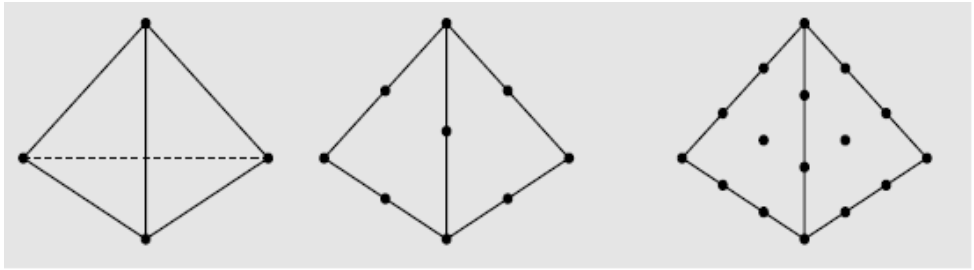

Figure 4: Linear ($d = 1$), Quadratic ($d = 2$) and Cubic ($d = 3$) Tetrahedral Elements

2.1.1 Number of Control Points

If we write $d > 0$ the degree (or order) of a tetrahedral element, then there are :

- V controls points that coincide with the *"tetrahedron vertices"*.

- $(d-1)E$ if $d > 1$ control points that are lying on edges.
- $\frac{(d-1)(d-2)TR}{2}$ if $d > 2$ control points that are lying on triangles.
- $\frac{(d-1)(d-2)(d-3)TE}{6}$ if $d > 3$ control points that are lying on tetrahedra.

Thus the total number of control points are :

$$C = V + (d-1)E + \frac{(d-1)(d-2)TR}{2} + \frac{(d-1)(d-2)(d-3)TE}{6}$$

2.1.2 Tetrahedron Bezier Indices

We use specific notations of control points inside a tetrahedron which we call *Tetrahedron Bezier Indices* (TBI). A TBI $p \in \mathbb{N}^+ \times \mathbb{N}^+ \times \mathbb{N}^+ \times \mathbb{N}^+$ is a 4-plet of positive natural numbers that indicate their relative position in the high order element.

Thus for an element of degree d, $\mathbf{p} = (i, j, k, l)$ is such that $|\mathbf{p}| = i + j + k + l = d$. The four TBI $(d, 0, 0, 0)$, $(0, d, 0, 0)$, $(0, 0, d, 0)$, $(0, 0, 0, d)$ coincides with the 4 tetrahedron vertices while $(0, i, 0, j), i > 0, j > 0$ is lying on the edge linking the second and fourth vertex and $(0, i, j, k), i > 0, j > 0, k > 0$ is lying on the triangle opposite to the first vertex. We write $\mathbf{C_p}$ the control point associated with indices (i, j, k, l).

2.1.3 Tetravariate Bernstein Polynomial

The control points are used to define a parametric volume in space. The parameters are the barycentric coordinates (r, s, t, u) such that $r + s + t + u = 1$ and $0 \leq r, s, t, u \leq 1$. The shape functions are the tetravariate Bernstein polynomials $B^d_{i,j,k,l}(r, s, t, u)$ of degree d that are themselves parameterized by four indices (i, j, k, l) such that $i + j + k + l = d$ with the following expression:

$$B^d_{i,j,k,l}(r, s, t, u) = \frac{d!}{i!j!k!l!} r^i s^j t^k u^l$$

For given degree d there are $N_d = 4+6*(d-1)+2*(d-1)*(d-2)+(d-1)*(d-2)*(d-3)/6$ such polynomials. To simplify notation, we use the same Tetrahedron Bezier Indices for the Bernstein polynomial as for the control points. Therefore $B^d_{\mathbf{p}}(r, s, t, u) = B^d_{i,j,k,l}(r, s, t, u)$.

2.1.4 Regular or Integral Bezier triangles

With this notation, the position of a point parameterized by (r, s, t, u) on a Bezier Tetrahedron is given by :

$$\mathbf{C}(r, s, t, u) = \sum_{|\mathbf{p}|=d} B^d_{\mathbf{p}}(r, s, t, u)\mathbf{C_p}$$

Any point on the Bezier tetrahedron is a linear combination of its control points and lies within the convex hull of control points as all weights (tetravariate Bernstein Polynomials) are positive.

2.1.5 Rational Bezier tetrahedra

An extension of Bezier tetrahedron is to consider weights $w_{\mathbf{p}}$ at each control points to change the importance of each control point in the definition of the volume. Each weight $w_{\mathbf{p}}$ of each control point is positive but may be greater than 1. The position of a point on a rational Bezier tetrahedron is then :

$$\mathbf{C}(r,s,t,u) = \frac{\sum_{|\mathbf{p}|=d} B^d_{\mathbf{p}}(r,s,t,u) w_{\mathbf{p}} \mathbf{C}_{\mathbf{p}}}{\sum_{\|\mathbf{p}\|=d} B^d_{\mathbf{p}}(r,s,t,u) w_{\mathbf{p}}}$$

Rational Bezier tetrahedra allow to represent exactly any quadrics volume (rational cubics for cylinders and rational quartics for spheres).

2.2 SOFA Implementation

2.3 Layout of Degrees of Freedom in SOFA

In SOFA, the degrees of Freedom (DOF) are stored into a set of arrays inside objects called MechanicalState. We have chosen to store the $\mathcal{C}$ DOFs of a Bezier Tetrahedral mesh inside a single MechanicalState.

Furthermore, we provide a default layout for ordering those DOFs in a MechanicalState. Figure 5 shows this ordering of control points. First of all, the control points associated with the tetrahedron vertices are stored, then those associated with edges, triangle and tetrahedra.

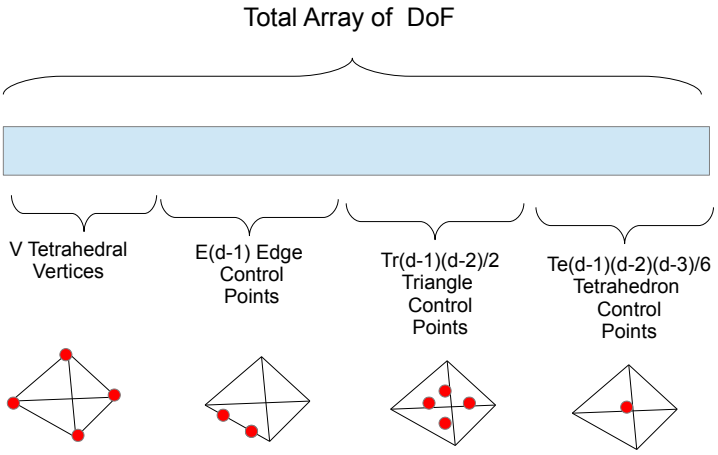

Figure 5: Layout of Degrees of Freedom of Bezier Tetrahedral meshes inside a MechanicalState object

There are however some issues. In a tetrahedral mesh, edges and triangles are not oriented (*e.g.* a triangle is common to 2 tetrahedra and is ordered differently among each tetrahedron) and therefore the order in the DoF array may not reflect the order in each tetrahedron. It is the role of the BezierTetrahedronSetTopologyContainer class to provide a proper ordering.

Second, since there are several control points for a given edge, triangle and tetrahedron, it is important to specify the ordering inside each element. Figure6 show in a simplified way how the control points are numbered for an edge, triangle and tetrahedron. In a nutshell, the element are stored in increasing lexicographic order of its barycentric coordinates. Thus on a quartic Tetrahedron element, the points of barycentric coordinates $(1/4, 3/4), (1/2, 1/2), (3/4, 1/4)$ are stored in this order. However, this order may not the one suitable for a given Bezier tetrahedron as the edge is shared by several tetrahedra.

If this default layout is not respected, then a specific lookup table should be provided in BezierTetrahedronSetTopologyContainer in order to specify the location of each DOF.

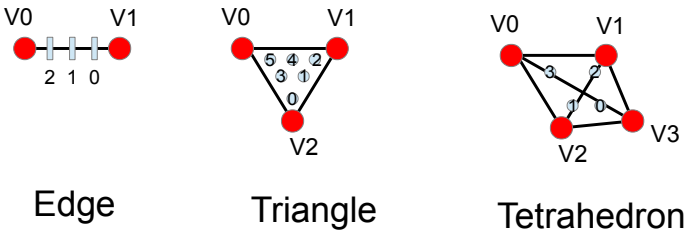

Figure 6: Layout of Degrees of Freedom of Bezier Tetrahedral meshes inside a MechanicalState object

2.4 BezierTetrahedronSetTopologyContainer Class

This class provides an interface to the array of DoFs store in the MechanicalState object. More precisely, a control point can specified in 3 different ways :

- A *global index* which is its rank in the array of size C.
- A tetrahedron index and a Tetrahedron Bezier Index (like $(2, 0, 2, 0)$ on a tetrahedron of order 4).
- A tetrahedron index and its *local index*. The local index is an integer between 0 and N_d which give the rank of the control points in this local (tetrahedron specific) list of control points.

The class thus provides function to move from one representation of a control point to another.

If the default layout is not used, then the 2 protected maps locationToGlobalIndexMap and globalIndexToLocationMap provide the mapping between global DOFs index and its location in the Bezier tetrahedra. globalIndexToLocationMap is a multimap as the same control point may belong to several tetrahedra.

The function *void getGlobalIndexArrayOfBezierPointsInTetrahedron(const size_t tetrahedronIndex, VecPointID & indexArray)* is particularly useful as it outputs an array of N_d global indices of the control points located inside a given tetrahedron. The tetrahedron Bezier indices of those control points are always the same for a given degree d and is given by the array returned by the function *sofa::helper::vector<TetrahedronBezierIndex> getTetrahedronBezierIndexArray() const;*.

Obviously the class can return the degree of Bezier tetrahedra (they are all of the same degree). Note that it provides both the total number C of control points *getNbPoint*() and the number V of tetrahedral points.

Also the class provides the weights associated with each control point (getWeight(int i)) and a boolean indicating if a given triangle is rational or integral (isRationalSpline(int i)).

Given a global index of a control point, it is possible to know in which tetrahedron it is associated and where it is located within the tetrahedron. The function void getLocationFromGlobalIndex(const size_t globalIndex, BezierTetrahedronPointLocation &location, size_t &elementIndex, size_t &ele- mentOffset) ; returns the location (point, edge, triangle or tetrahedra) of a control point, the index of the tetrahedron (elementIndex) and the rank within its location.

2.5 BezierTetrahedronSetGeometryAlgorithms Class

This class provides basic geometric functions :

- *Coord computeNodalValue(const size_ t tetrahedronIndex,const Vec4 barycentricCoordinate);* which computes the position / value of a node given the index of a tetrahedron and its barycentric coordinate.

- *Real computeBernsteinPolynomial(const TetrahedronBezierIndex tbi, const Vec4 barycentricCoordinate);* which computes $B^d_{\mathbf{p}}(r,s,t,u)$.

To optimize things it precomputes the Bernstein coefficients of a given degree. If *drawControlPointsEdges*, *drawSmoothEdges* or *drawControlPoints* are set to true then it draws the edges of the control point net, control points and Bezier edges.

2.6 Mesh2BezierTopologicalMapping Class

This class provide a way to create a Bezier tetrahedral mesh of any order given a tetrahedral mesh. The control points are simply linearly interpolated from the tetrahedral vertices : a control point of index $\mathbf{p} = (i,j,k,l)$ is computed with the barycentric coordinates $(\frac{i}{d}, \frac{j}{d}, \frac{k}{d}, \frac{l}{d})$. The mapping automatically sets the degree and the number of tetrahedral vertices in the object BezierTetrahedronSetTopologyContainer.

Note that the position of the mapping in the scene has be after BezierTetrahedronSetTopologyContainer (see the example in BezierTetrahedronTopologicalMapping.scn)

2.7 BezierTetra2BezierTriangleTopologicalMapping Class

This topological mapping creates a Bezier Triangulation from the surface of a Bezier Tetrahedral mesh. In another words, it is the Bezier equivalent of the Tetra2TriangleTopological mapping. For this to work, it is necessary that the following components be present in the same node : TriangleSetTopologyModifier, BezierTriangleSetTopologyContainer.

Note that the created Bezier triangulation share the DOFs as the Bezier tetrahedral mesh and therefore it does not rely on the default layout of DOfs for Bezier Triangles (see the example in BezierTetra2BezierTriangleTopologicalMapping.scn).