

Skinning a Deformation Model

by
Jean-Philippe Egger

Supervisors: Anderson Maciel, Sofiane Sarni and Patrick Lemoine
Director: Prof. Daniel Thalmann

Semester Project winter 2003/2004

Virtual Reality Lab (VRLab)
School of Computer and Communication Sciences (I&C)
Swiss Federal Institute of Technology (EPFL)



Content

1	Introduction.....	3
1.1	Context.....	3
1.2	Project Goals.....	3
1.3	Organisation.....	4
2	Analyse and Conception.....	4
2.1	Introduction.....	4
2.2	Bezier Curves.....	5
2.2.1	Theory.....	5
2.2.2	Implementation.....	6
2.2.3	Measures.....	6
2.2.4	Conclusion.....	7
2.3	Analyse of the COME-BIO project	8
2.3.1	Structure of the implied COME-BIO classes.....	8
2.3.1.1	COME_Molecule.....	8
2.3.1.2	COME_MoleculesTissue.....	9
2.3.1.3	COME_BioStructure.....	10
2.3.1.4	COME_Mesh.....	11
2.3.2	Class hierarchy of the COME-BIO application.....	11
2.4	Usability of the Mesh-Molecule connection algorithm	12
2.4.1	The theory behind the algorithm.....	12
3	Implementation.....	15
3.1	Additional description of positioning surface vertices.....	15
3.2	Skinning.....	16
3.2.1	Mesh initialisation.....	16
3.2.2	Periodical Mesh updates.....	18
3.3	Collision Detection.....	18
3.3.1	Rapid library.....	18
3.3.2	Detection of immersed faces.....	18
3.3.3	Collision response.....	19
4	Conclusions.....	19
4.1	Skinning.....	19
4.2	Collision Detection.....	19
5.	Final Word.....	20
	References.....	20
	Appendix A: Summary of the modifications made to the original program.....	21

1 Introduction

1.1 Context

The NCCR-COME [1][2], more specifically the Project 10: *A generalized approach towards functional modelling of human articulations* [COME 02] aims to provide medical applications to aid on diagnosis of joint disease and planning of surgical interventions. Such applications will be based on a biomechanical model of the tissues present in the joint and a framework for visualization and interaction with this model, both being developed in the VRLab[8]. Such model and framework rely on the mechanical and physical properties of biomaterials, and then they depend of the force exchange between the different structures of joints anatomy to provide correct motion and deformation. In this context, the COME-BIO [3] application is a complex physical model designed by A. Maciel [4]. Enhancing this application is the context of the work presented here.

1.2 Project Goals

According to the situation described above, the Objective of this project is to enhance the existent deformation model conceived in the VRLab[8]. The COME-BIO [3] model describes the interactions and deformations of objects in a virtual world with gravity. The model is based on a sub division of objects into molecules represented by spheres. The connecting forces between the molecules are simulated by springs. Exerting external forces (Collisions, torsions, etc.) on the objects will cause deformations witch should simulate those of reality. This model has already been implemented and is working well. Thus, this project consists into adding two new features to the model.

- For the realistic visualisation of the simulated objects, a polygon mesh should cover the molecules in a way to hide the sphere model behind it. This is referred as “skinning” an object (see figure 1).
- Later, the collision detection should be moved from the actual molecule based detection onto the detection of colliding meshes. This should clearly increase the reality of the model and resolve a few problems due to normally impossible situations.

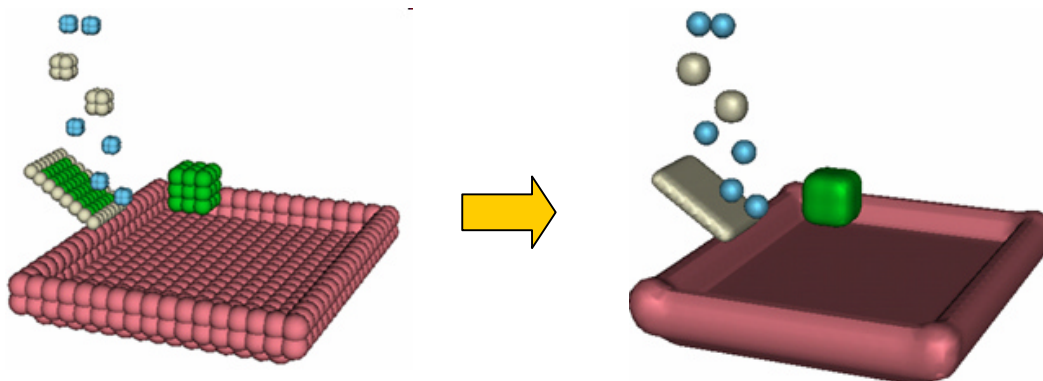


Figure 1: Skinning the deformation model

1.3 Organisation

This project was separated in four main phases as following:

- Analyse possibility of using Bezier Curves
- Analyse and understand the functioning of the COME-BIO [3] application in order to get familiar with the environment of the application
- Evaluate the feasibility of porting the algorithm used in the Ph.D. thesis “Anatomically-Based Human Body Deformations” by A. Aubel [5] to the COME-BIO [3] environment.
- Choose and implement a solution for skinning the objects, for the collision detection and for the collision response

2 Analyse and Conception

2.1 Introduction

The first phase of the analysis was to evaluate the feasibility of using Bezier Curves as a method for skinning our objects. In particular we were interested in performance issues as well as in the flexibility regarding the number of control points and the mesh size. For this purpose a separate test application had to be implemented.

The second phase consisted of finding why the implementation done by F. Benedetti [6] wasn't working properly. In her work, she very successfully implemented the generation of a smooth surfaces for any static object defined by molecule positions and radiuses. Unfortunately the connection of the correctly generated skin to the molecule model didn't work properly. We didn't know if the algorithm used wasn't appropriated to our context in general or if there had been implementation errors. Thus, this second phase implied a thorough analyse of the complete code of the COME_BIO [3] application.

The third phase was the analysis of the algorithm used in the Ph.D. thesis “Anatomically-Based Human Body Deformations” by A. Aubel [5]. In his work, the algorithm has successfully been applied for modelling the skin. The question was to evaluate if this concept would work as well in the COME-BIO [3] environment. This third phase is highly connected to the work done by F. Benedetti [6] as she tried to implement a modified version of this algorithm.

2.2 Bezier Curves

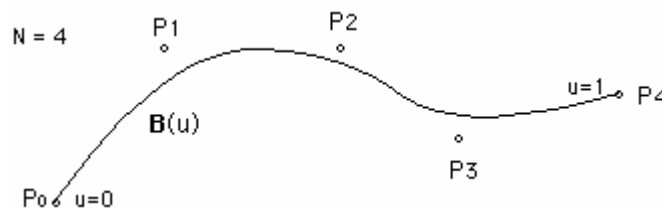
Bezier Curves are a very well known mathematical concept which is widely used in many graphical applications. The main advantage of Bezier curves is the smoothness of the Curves [7].

2.2.1 Theory

Consider $N+1$ control points P_k ($k=0$ to N) in a 3-dimension space. The Bezier parametric curve function is of the form

$$B(u) = \sum_{k=0}^N p_k \frac{N!}{k! (N-k)!} u^k (1-u)^{N-k} \text{ for } 0 \leq u \leq 1$$

$B(u)$ is a continuous function in 3 dimensional space defining the curve with N discrete control points P_k . The parameter u equals 0 at the first control point ($k=0$) and u equals 1 at the last control point ($k=N$).



The Bezier surface is formed as the Cartesian product of the blending functions of two orthogonal Bezier curves.

$$B(u,v) = \sum_{i=0}^{N_i} \sum_{j=0}^{N_j} p_{i,j} \frac{N_i!}{i! (N_i-i)!} u^i (1-u)^{N_i-i} \frac{N_j!}{j! (N_j-j)!} v^j (1-v)^{N_j-j}$$

$0 \leq u \leq 1$
 $0 \leq v \leq 1$

Where $P_{i,j}$ is the i,j th control point. There are N_i+1 and N_j+1 control points in the i and j directions respectively.

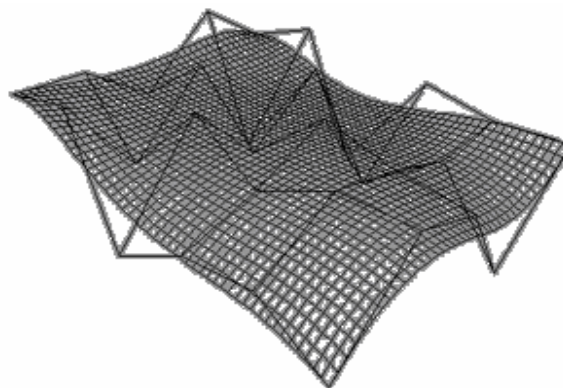


Figure 2: A Bezier surface with it's corresponding control points

2.2.2 Implementation

To evaluate the practical point of view of the theory, we implemented a test application. The number of control points could freely be chosen in the i and j direction independently. The Bezier curve was evaluated without any segmentation into smaller patches. We made the control points move periodically following a sinus wave while the Bezier surface was constantly reconstructed and displayed. To correctly measure the performance of this solutions the calculation of the vertices normal were also implemented.

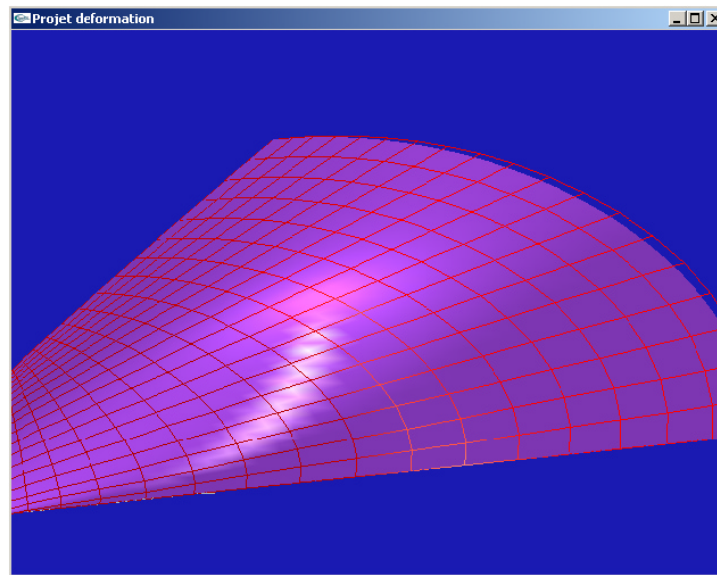


Figure 3: The Bezier surface (purple) clearly follows the control points (red lines)

2.2.3 Measures

The COME-BIO [3] application is designed for real time manipulation of deformable objects. Acceptable frame rates for such a real time application would be around 20 frames per second. The test application on the tested system (Win2000, MS Visual C++, AMD Athlon XP2600+, 524mb RAM) runs at 20 frames per second using 10×10 control points with the mesh divided into 40×40 intervals. Based on this acceptable performance, two kind of measures have been taken:

- Fixing the number of control points to 100 (10×10) and measuring the frame rates for changing numbers of intervals evaluated in the Bezier Curves.
- Fixing the number of evaluated intervals in the Bezier Curves 1600 (40×40 evaluated points implies $(39 \times 39 \times 2) = 3042$ polygons for the mesh) and measuring the frame rates for changing number of control points.

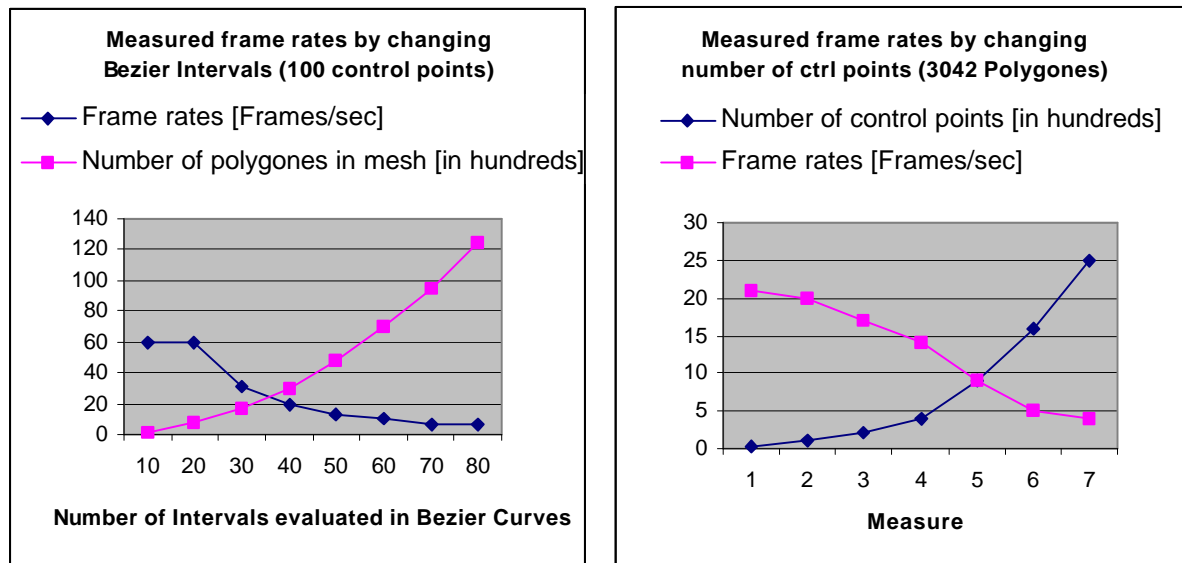


Figure 4: Measure of performance for creating Bezier surfaces

2.2.4 Conclusion

Bezier surfaces guarantee a very smooth surface what is very likable. Unfortunately the computation time is very high and thus the scalability limited. Further the surface doesn't pass exactly through the control points witch increases the incertitude about the geometrical correctness of the mesh. Finally an important question aroused to witch no definite answer could be found. How would we find the control points witch would create a Bezier surface corresponding to the objects given. We talked through a few possible solutions to compute these control points given the positions and radiuses of the molecules, but nobody could affirm that these control points would lead to a appropriate surface.

The heavy computation time and the incertitude about the final graphical result made the use of Bezier curves for skinning our deformation model less attractive.

2.3 Analyse of the COME-BIO [3] project

The COME-BIO [3] project is a set of many classes with complex interaction. The general hierarchy is the following: The virtual world of the simulation has patients who are composed of organs that are again composed of molecules. Throughout the rest of this document the context of a biological application will be extracted. Therefore the earlier mentioned organs will be referred to as objects in the simulation and treated as fully independent.

Describing the complete application is far beyond the scope of this project and is published elsewhere. I will therefore concentrate on describing the classes concerned by the skinning procedure.

2.3.1 Structure of the implied COME-BIO [3] classes

The following structures are implied when constructing the mesh of an object. The description of each structure is purposely focused on the skinning procedure and therefore is not complete.

2.3.1.1 COME_Molecule

Description

This class represents a generic molecular subdivision of an object. Each molecule in the model replaces a set of molecules of the real object. A COME_Molecule consists of a spherical volume with a constant radius that approaches and diverges of its neighbours without regard to geometrical penetration. A 4x4 Matrix defines the position and orientation of the molecules. Further each molecule is connected to others by means of a list of COME_MoleculeLinks. The orthogonal neighbours are used to create the actual orientation. The necessity of the orientation is described in paragraph 2.3.1 and 3.1.

Structure

COME_Molecule (*partial description)

Attributes:

The characteristics of a molecule

double	radius
COME_Matrix *	localFrame //position and orientation
vector<COME_Vector3D>	originalLocalFrame;
double	frictionConst

Conections between the different molecules

list< COME_MoleculeLink * > *	connectionList
vector< COME_Molecule * >	orthogonalNeighbors

Forces actually applied to the molecules

COME_Force	fGravity
COME_Force	fExternal //Total external force
vector< COME_TimeForce * >	externalForces //list of all separate external forces
COME_Force	collisionForce

Functions:

void	addCollisionForce (COME_Force forceN)
void	addExternalForce (COME_TimeForce *forceN)
void	initializeOrthogonalNeighbors ()
vector< COME_Molecule * > &	getOrthogonalNeighbors ()
void	makeLocalFrame ()
COME_Matrix *	getLocalFrame ()

2.3.1.2 COME_MoleculesTissue

Description

A tissue is basically a set of molecules. This class describes the complete characteristics of the molecular system of an object. For our purpose no further description is needed.

2.3.1.3 COME_BioStructure -> COME_Cartilage -> COME_MoleculesCartilage

Description

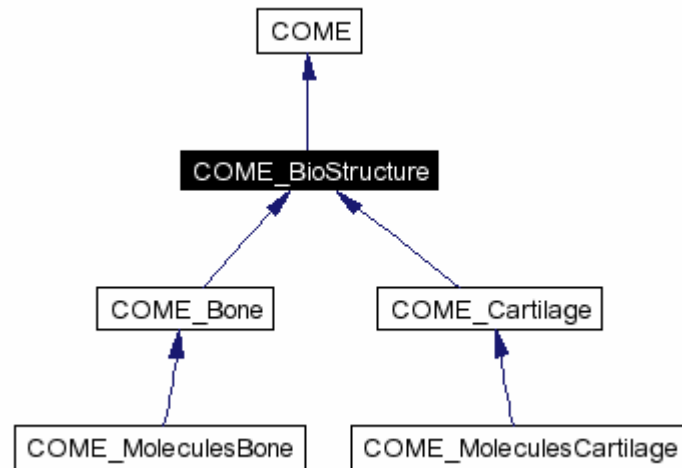


Figure 5: Inheritance diagram

The COME_BioStructure is an abstract class representing one organ (biological structure) of a patient. The COME_Cartilage and COME_MoleculesCartilage classes implement the abstract class by adding specific features and behaviours. I will refer to COME_MoleculesCartilage as being an individual object of the virtual world as foresaid in the introduction of chapter 2.2.

Structure

COME_MoleculesCartilage (*partial description)

Attributes:

```

//The molecular system of an object
COME_MoleculesTissue *  tissue
//The polygon surface of an object
COME_Mesh                surface
//The local and global orientation and position matrix
COME_Matrix              localInstanceMatrix
COME_Matrix              globalInstanceMatrix
  
```

Functions:

```

//Connects the generated skin to the molecule model
void  initializeSkinning ()
  
```

2.3.1.4 COME_Mesh

Description

This class represents a polygon surface, the skin of any object in the virtual world. It is composed of a list of vertices and of faces. The faces consist of a normal and the three indexes of their corresponding vertices in the list of vertices.

Structure

COME_Mesh (*partial description)

Attributes:

```
vector< COME_Vertex3D > vertices
vector< COME_Vertex3D > verticesGlobal
vector< COME_Face > faces
```

Functions:

```
vector< COME_Vertex3D > & getVertices ()
vector< COME_Vertex3D > getVerticesGlobal ()
void updateVerticesNormals ()
bool isInside (COME_Point3D currentPoint)
void anchorToMolecules (COME_MoleculesTissue *substract, int anchorNumber)
```

Constructor:

```
COME_Mesh (const vector< COME_Vertex3D > verticesN, const vector< COME_Face >
facesN, COME_BioStructure *parentN)
```

2.3.2 Class hierarchy of the COME-BIO [3] application

The environment of the COME-BIO [3] application is a three dimensional world exposed to gravity. For better understanding of the hierarchical structure the individual elements which fill the empty space are represented on Figure 6 in a highly simplified manner.

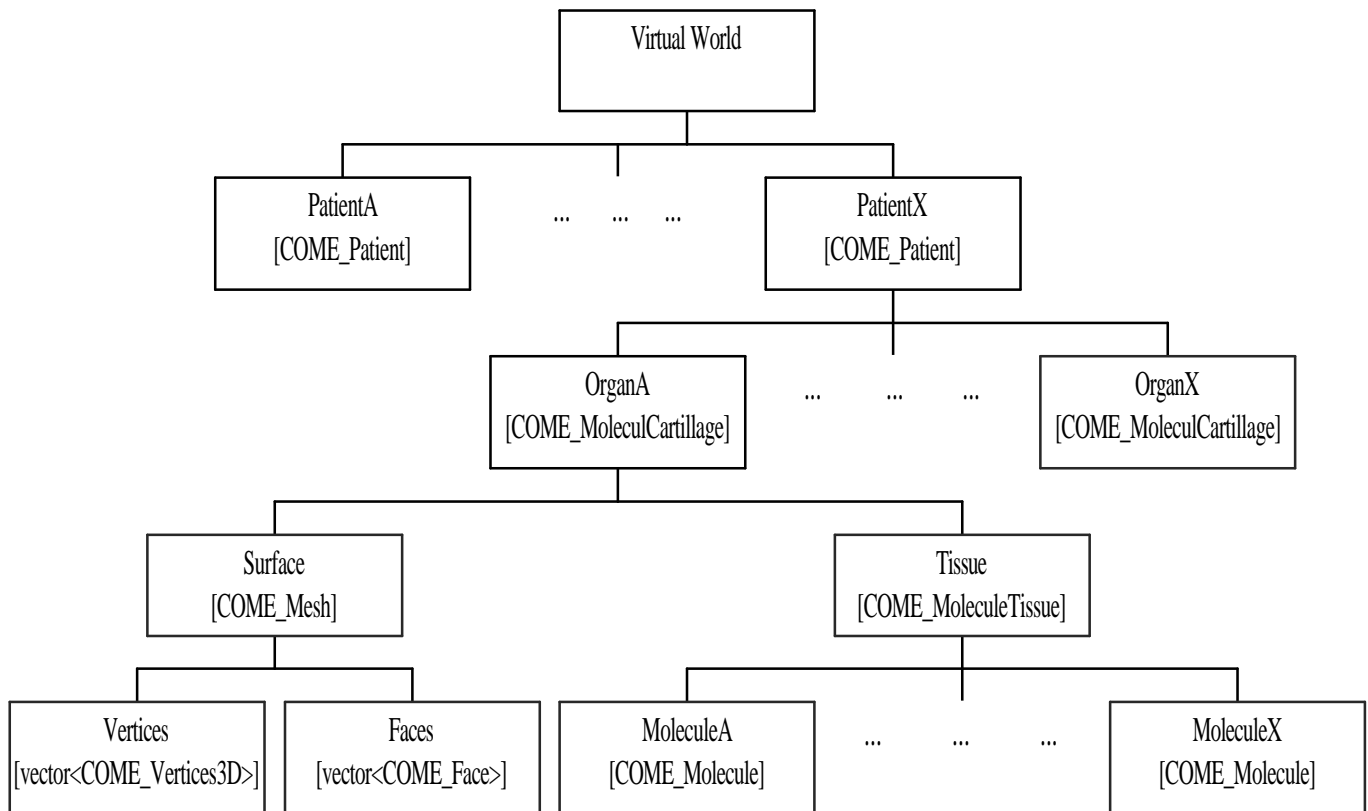


Figure 6: Simplified hierarchical structure of the COME-BIO [3] application

2.4 Usability of the Mesh-Molecule connection algorithm

The algorithm presented in this chapter was inspired by the Ph.D. thesis “Anatomically-Based Human Body Deformations” by A. Aubel [5]. In his work, the algorithm has successfully been applied for modelling the skin.

Thanks to the work done by F. Benedetti [6] the initial generation of a surface given any object described in the COME-BIO [3] application was already implemented. Thus the remaining problem to be solved was to connect the generated surface to the model. The goal was to avoid recalculating the surface at each time step. Therefore the Surface’s vertices should be connected to the model in such a way, that their new positions could be directly derived from the positions and orientations of the objects molecules.

2.4.1 The theory behind the algorithm

The initial situation is represented in figure 7. The global positions (relative to the origin of the virtual world) of all molecules and surface vertices are given. The algorithm is composed of the following steps:

1. Initialisation of the local coordinate systems of each molecule. The construction of this coordinate system is based on finding the 3 most perpendicular neighbours and using their relative positions for the direction of the coordinate system axis.
2. For each surface vertex store it's relative position to the n closest molecules expressed in the local coordinate system of each of those molecules (red arrows in figure 7). This position will remain constant throughout the whole simulation. We refer to this as “anchoring” the surface vertices to the molecules. Each vertex has now n anchoring molecules. From now on the position of a vertex relative to the origin of the world is expressed by the anchor-molecule's position and the relative coordinates of the vertex to it.
3. At each integration step the local coordinate systems of the molecules must be recalculated using the new positions of the neighbours molecules. Thus, the local coordinate systems of the molecules move and rotate. Because the relative coordinates of the surface vertices to the molecules stay constant the surface follows the rotational and positional movements as if the skin was effectively part of the model (see figure 7).

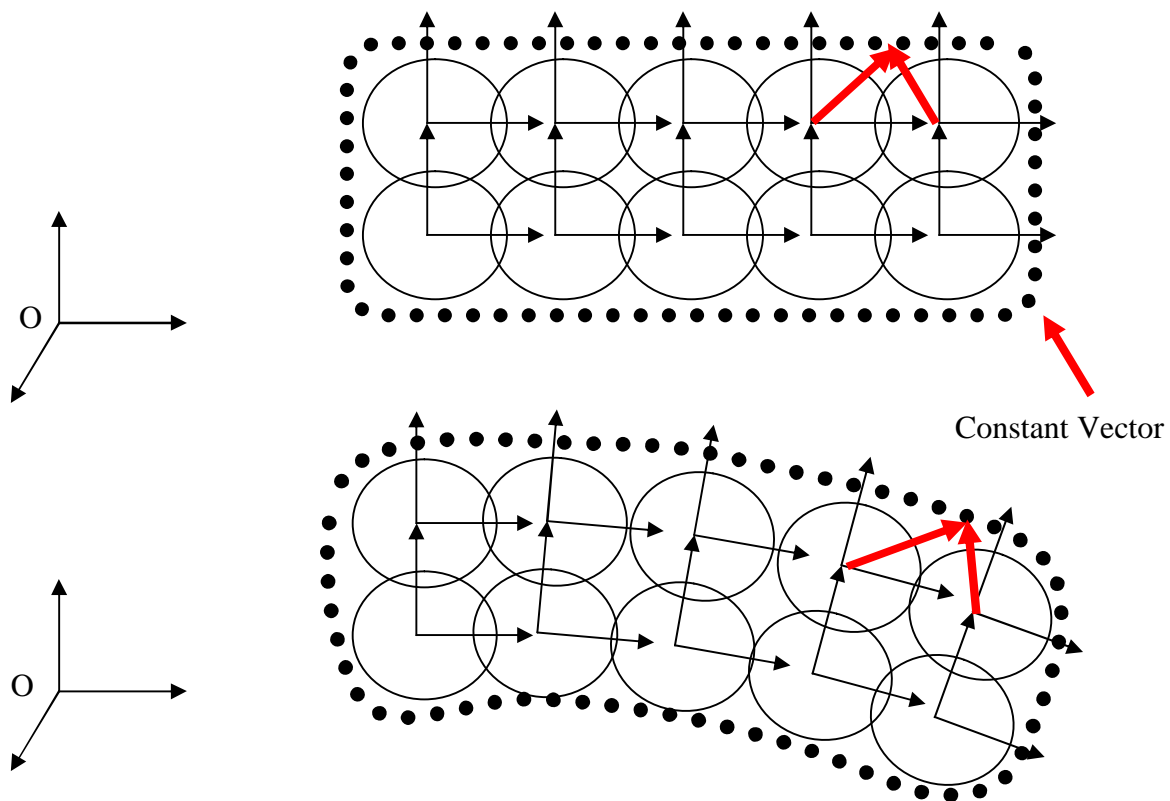


Figure 7: Illustration of the Algorithm: The relative coordinates of a surface vertex (red arrows) to its anchoring-molecules are constant while the local coordinate systems of the molecules move and rotate.

The reason for using n anchors is to prevent the formation of discrete changes in the surface. This is best understood by looking at figure 8. The upper two images represent the surface of a deformed object using only one anchor. Strong orientation differences between neighbouring molecules induce a surface jump at the border between vertices anchored to one molecule or the other. To counter this side effect we need to “blend” the transition from one anchor to another.

Because each anchor moves and rotates individually, the global positions of the surface vertices expressed through their relative positions to the molecules differs depending on which anchor is used. A blending function, which calculates a weighted average of all these positions, is needed to compute the correct final global position. The weights of the different positions are directly connected to the distance between the anchors and the corresponding surface vertex. The position of a vertex dictated by a closer molecule to it has a greater weight than the position dictated by a molecule further away.

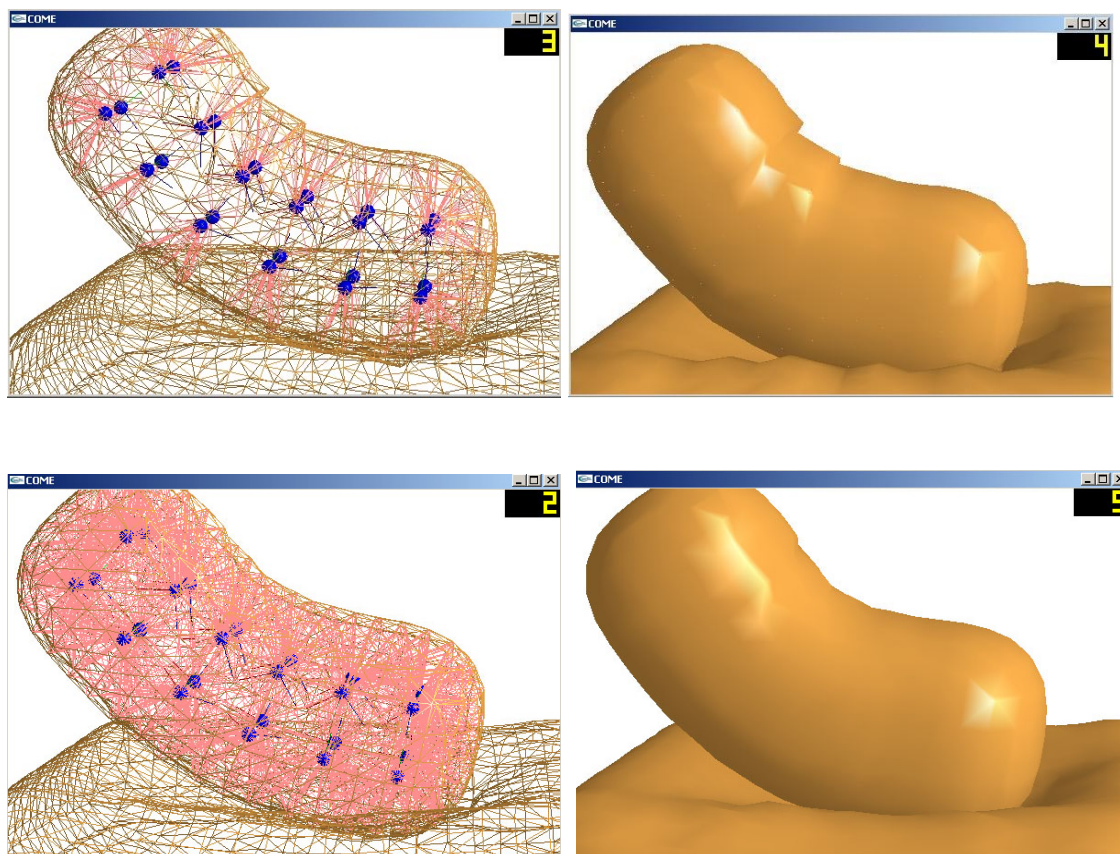


Figure 8: Skin deformation using only one (Upper row) anchor and using four (Lower row) anchors per skin vertex. With only one anchor the border where the anchors of the vertices change can clearly be seen (upper right). The connection between the anchors and the skin vertices are visualized in red on the transparent view (left column).

3 Implementation

During the analysis of the work done by F. Benedetti [6] a few serious implementation errors were found. This hardened the general impression that the chosen algorithm could function as well in the COME-BIO [3] environment, as it did for the application designed by A. Aubel [5]. Also the less attractive results obtain with the Bezier curves encouraged the continuation in the direction taken by F. Benedetti [6].

With this hope in mind the correcting and modifying of the source code began. A few days later the skinning of the deformation model was working well so that remained only the collision detection to finish.

3.1 Additional description of positioning surface vertices

Before describing the implementation the positioning system of the vertices relative to the molecules must be clarified. To compute the global position \mathbf{v}_G (relative to the origin of the virtual world) of a vertex knowing its relative position to a molecule we use Matrix multiplication. The relative position of the vertex \mathbf{v}_R is multiplied by the molecules position/rotation Matrix \mathbf{M}_m .

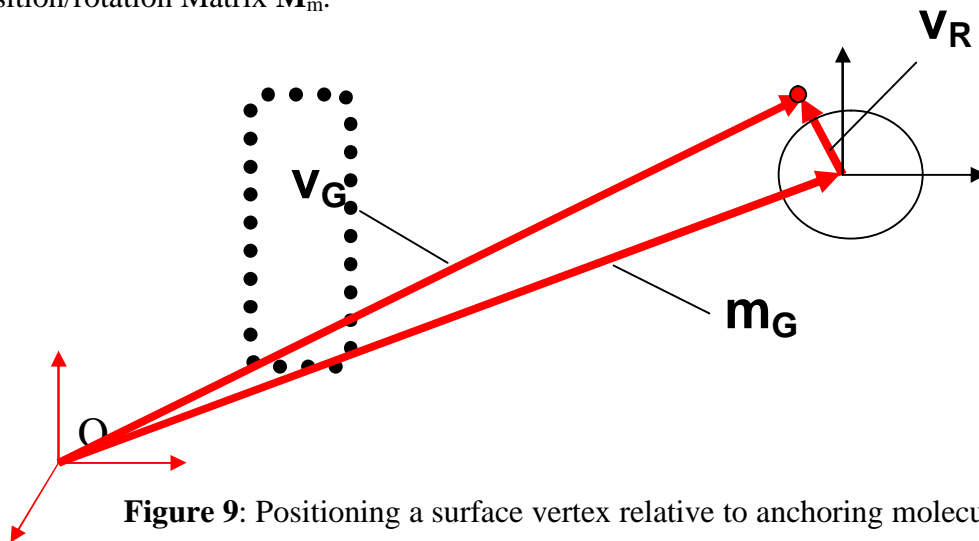


Figure 9: Positioning a surface vertex relative to anchoring molecule

Equation 3.1
$$\mathbf{v}_G = \mathbf{M}_m * \mathbf{v}_R$$

The 4x4 position/rotation Matrix is composed of a typical 3x3 axis rotation matrix \mathbf{R} , the global position of the molecule \mathbf{m}_G and a constant vector.

$$\mathbf{M}_m = \begin{pmatrix} R_{1,1} & R_{1,2} & R_{1,3} & 0 \\ R_{2,1} & R_{2,2} & R_{2,3} & 0 \\ R_{3,1} & R_{3,2} & R_{3,3} & 0 \\ \mathbf{m}_G & \mathbf{m}_G & \mathbf{m}_G & 1 \end{pmatrix}$$

By inverting equation 3.1 we can calculate the relative position of the vertex to a molecule knowing its absolute position. This is done once at the beginning to store all relative positions (At the beginning, the mesh's position is given relative to the global coordinate system)

Equation 3.2
$$\text{inv}(\mathbf{M}_m) * \mathbf{v}_G = \mathbf{v}_R$$

To create the rotation Matrix \mathbf{R} of a molecule, additional information is needed. At initialisation the original local frame of a molecule must be calculated and stored. The local frame consists of three orthogonal vectors. At every integration step the new local frame is constructed using the position of the neighbour molecules. The rotation Matrix \mathbf{R} describes the rotation from the original local frame to the actual. With the help of quaternions we can easily create this rotation matrix using the original and the actual frame. Once the actual rotation matrix recalculated the position/rotation Matrix \mathbf{M}_m is updated without forgetting to add the molecule's actual position to the last row of the matrix. After this all the vertices anchored to this molecule can be positioned in the global system by using the equation 3.1.

3.2 Skinning

During the modifications of the COME-BIO [3] application the global structures of the project didn't change at all. Roughly a few methods were modified and a few added to different classes. The only class added to the project is the Quaternion class for rotation calculations.

The structure of the implied COME-BIO [3] classes has been presented in chapter 2.2.1. Please refer to these descriptions and to the graphical illustrations for a better comprehension of the following chapters.

3.2.1 Mesh initialisation

On start up the COME-BIO [3] is an empty space with no objects. Complete scenes can be loaded from files using a specific XML syntax. Once a scene successfully loaded and the molecule models are created, the initial surface of each object is calculated and stored in the respective COME_Mesh (see figure 6 for hierarchical information). Once this done, the connections between the molecules and the mesh's vertices have to be created. This is done by calling the `initialiseSkinning()` function on every organ in the scene.

`InitialiseSkinning()` performs the following operations:

1. For every vertex find and store the n nearest molecules (the closest is stored separately):

This is implemented in the COME_Mesh function `anchorToMolecules()`. Experience showed that more than four anchors for each vertex didn't increase the realistic aspect of the surface any more so $n=4$.

2. For every molecule initialise the original local frame (see figure 7):

The algorithm is implemented in the COME_Molecule function named `initializeOrthogonalNeighbors()` and works as following.

- First the three most orthogonal neighbours have to be found. The implemented algorithm chooses randomly the first neighbour out of the set of connected molecules. Next the neighbour forming the closest angle to 90 degrees with the previous neighbour is held. Finally the neighbour with the smallest angle to the normal formed by the two first neighbours is held. These three neighbours are stored in the COME_Molecule variable called `orthogonalNeighbors`.

- Once the neighbours defined, the three vectors connecting the actual molecule to the three orthogonal neighbours are calculated and stored in the COME_Molecule variable `originalLocalFrame` by calling the function `makeOriginalFrame()`.

Note: If the object has only one layer of molecules, the coordinate system will be far from orthogonal. However experiences have shown that this didn't matter much. In general though the Orthogonal neighbours algorithm finds almost always a perfectly orthogonal coordinate system.

3. For every molecule calculate the actual rotation/position Matrix

This is done by calling the COM_Molecule function `MakeLocalFrame()`. There, the rotation/position Matrix called `localFrame` is calculated as described in paragraph 3.1 using the position of the molecule, the actual local frame and the original local frame.

4. For every vertex the relative positions to their 4 anchors (red arrows in figure 7) must be initialised and stored:

This is directly done in the `InitialiseSkinning()` function. For each anchor of a vertex the equation 3.2 is used to compute the relative position of a vertex knowing it's global position. The relative positions to the 4 anchors are stored for each vertex in the COME_Vertex3d variable called `localPositions`.

5. For every Face of the surface find and store all neighbouring faces. This is done by calling the COME_Mesh function called `initFacesNeighbours()`.

6. For every vertex of the surface find and store all neighbouring vertices:

This is done by calling the COME_Mesh function called `initVerticesNeighbours()`. The use of storing the indices of neighbouring vertices is the gain of speed for the vertices normal calculation. Later the vertices normal are calculated by taking the average normal of all neighbouring faces.

3.2.1 Periodical Mesh updates

Once the anchoring of all surface vertices terminated the model is all ready to be used. At each integration step, the physical model calculates the new positions of each molecule, based on all the internal/external forces, the actual speed, the acceleration etc. Once the new molecules positions are calculated, the rotation/position matrix of each molecule is updated by calling the COME_Molecule function called makeLocalFrame() .

Finally, for displaying the mesh, the global positions of the vertices are precalculated by calling the COME_Meesh function updateGlobalPositions(). This function implements the blending function described at the end of paragraph 2.3.1. The final position is the weighted average of the four global positions obtained by using the four different rotation/position Matrixes of the anchoring molecules and the corresponding relative positions of the vertex to its anchors. Depending on witch display mode has been chosen (one normal per face or one normal per vertex) the normal of the faces/vertices must be update before displaying the mesh. This is either done by calling the COME_Mesh function updatFacesNormalsGlobalPosition() or respectively updateVerticesNormalsGlobalPosition().

3.3 Collision Detection

3.3.1 Rapid library

The rapid library is used for the collision detection. First a general enclosing “box test” is done to find potentially colliding pairs of organs. Each organ pair is then reconstructed in the rapid environment and the collision detection executed. The rapid library computes and returns a list of colliding faces. Unfortunately, the rapid library does not reveal faces that are completely inside the other mesh. Thus an algorithm for finding these faces had to be elaborated.

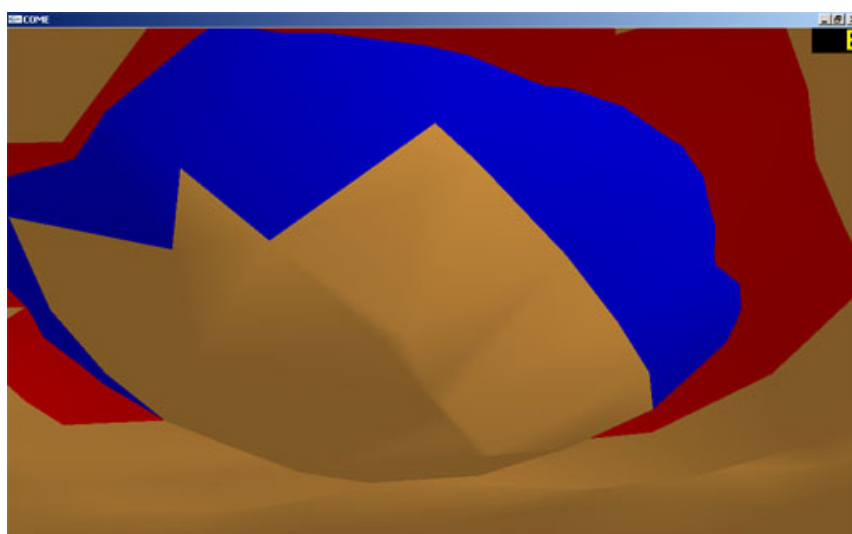


Figure 10: Rapid detects only colliding Faces (blue/red) but not immersed faces

3.3.2 Detection of immersed faces

Unfortunately, the rapid library does not reveal faces that are completely inside the other mesh. This means that for collision with high velocity, a major part of the repulsive forces are missing. The side effects are longer reaction times in the repulsion process which implies greater penetrations. To avoid this, the following recursive algorithm was elaborated.

Initial Situation: Rapid has detected all the directly colliding faces (totalFaces)

1. All neighbours of the totalFaces are potentially totally immersed in the other mesh and therefore not detected by rapid. Thus, compute all neighbouring faces and store them in PotentialFaces.
2. For each potential face test if it is inside the other mesh. If true and not in totalFaces yet, add face to immersedFaces.
3. Add the immersedFaces to totalFaces. Then, for each immersedFace compute neighbours that are not in immersedFace neither in totalFaces. These are the new potentialFaces for going back to point 2.
4. Stop when no new potentialNeighbours
5. TotalFaces is composed of all the faces that need to be taken into account for repulsion calculation.

3.3.3 Collision response

For every face that is colliding with another face, the penetration distance is calculated by using the COME_Face function distanceFaceFaceGlobalPosition(). Once the penetration distance computed, a temporary spring is applied between the two faces, which induces repulsive forces directly proportional to the penetration distance. The directions of these colliding forces are always parallel to the faces normal. Once all the forces calculated, they are separated into groups corresponding to the same closest molecule. The directions of all these forces are blended into one single vector. The final force is in direction of that blended vector and has the intensity of the greatest Force in the group. This force is then applied to the molecule in each organ. This “passes” the mesh collision detection results back to the molecule model.

4 Conclusions

4.1 Skinning

The implemented method inspired by the Ph.D. thesis “Anatomically-Based Human Body Deformations” of A. Aubel [5] works very well for this deformation model. The skin looks very realistic even for extreme deformations. This is a very satisfying solution in a graphical point of view. The computation time is still high but is beginning to be acceptable, even though thorough performance optimisation has not yet been applied.

4.2 Collision Detection

Collision detection using the rapid library is a very performing and reliable solution. The implemented algorithm in the COME-BIO [3] application works well and the reactions to the collisions seem realistic. For confirming this impression many real life measures should be compared to the simulation, what is far beyond the scope of this project and will probably be done later by A. Maciel [4].

5 Final Word

The COME_BIO project is a very ambitious and promising work. I was surprised by the accuracy of the model and by the well structured design. It was very motivating to work on a project involving many different areas of engineering. I believe this project has contributed to improve the graphical presentation of the models. The new skin gives the viewer a very realistic visual experience of deformable objects.

This contribution has brought the deformation model a step closer to the simulation of real life deformation of body tissues. I wish to the complete COME-BIO [3] project team good luck for the continuation of their effort and I thank especially Anderson Maciel for his help and support.

References

- [1] NCCR: National Centre of Competence in Research
- [2] COME: Computer Aided and Image Guided Medical Interventions <http://come.ch>
- [3] COME-BIO: Library developed in the context of the COME [2] project by A. Maciel [4]
- [4] Anderson Maciel, PhD Candidate and Research Assistant at the Virtual Reality Lab of Lausanne in 2004.
- [5] Amaury Aubel, Senior Researcher and Ph.D. at the Virtual Reality Lab of Lausanne in 2002. Publication: Ph.D. thesis “Anatomically-Based Human Body Deformations”
- [6] Fabiana Benedetti followed a postgraduate course on Graphics Visualization and Communication at the Virtual Reality Lab of Lausanne in 2002. Publication: “Physical Response to Collision between Deformable Objects”
- [7] Theory based on web site <http://astronomy.swin.edu.au/~pbourke/curves/bezier/> written by Paul Bourke
- [8] VRLab: Virtual Reality Laboratory at the EPFL [9]
- [9] École Polytechnique Fédérale de Lausanne