Respiration Simulator

Project Final Report
18th September 2009

Dr David Lilburn, MSc Computing Science
Imperial College London, CID 00527332

Project Supervisors: Dr Fernando Bello
Dr Pierre-Frédéric Villard

Contents

1. Introduction

1.1. **Aims of Project**

The aim of the project is to build on the work done on simulating the motion of several of the key organs during the respiration cycle from previous academic years at the Department of Biosurgery and Surgical Technology, Imperial College at St Mary's Hospital, London [1,2,3]. The previous work focused mainly on motion of the diaphragm and ribs during the respiration cycle as a surrogate for motion of the liver during the same period, to allow the creation of a realistic model for liver motion.

The main aim of this project is to create an anatomically and physiologically realistic simulation of the deformation of the major organs involved in respiration for medical student teaching. This will be created in X3D, Python and C++ under the H3D® open source development framework to be fully compatible with the department's haptics devices.

If successful, the simulation will hopefully form the basis of a future liver access procedure simulator for the training of clinical personnel in the technically difficult procedure of liver biopsies.

1.2. **Contributions**

i.e. what has this project added and how has it moved the field forward.

The main contribution of this project has been to implement a dynamic mass spring system under the H3D development framework and to apply it to the technically difficult area of anatomical and physiological modelling of organ movement and deformation during the respiratory cycle.

The project has drawn on my various backgrounds, notably medicine, physics and computing science to produce a realistic simulation of the above and has resulted in a new simulation of lung and costal cartilage movement and deformation during the respiratory cycle.

It has also built on and improved on previous years' work at the Department of Biosurgery and Surgical Technology and resulted in an improved realism of the rib anatomy and motion and a more medical student centred and useful package overall.

2. Problem Domain

2.1. Fundamentals of Medical Simulation

i.e. Fundamentals of what is to be achieved and why

Simulations are artificial models of real systems which have certain rules that define the system and how it behaves. Certain elements are removed or not modelled to simplify and focus on the important features. This not only aids understanding but reduces drain on resources, notably computing power.

Simulations are often used to visualise systems, perform tasks or model the effect of change which are simply not possible to do so on a regular basis due to expense, safety or feasibility. Simulations are also used to model events and so predict the future.

The ultimate aim of a simulation is to provide a greater understanding of the world and universe around us. To achieve this aim, simulations need to be:

1. **Realistic:** Closely represents reality now and in the future
2. **Validated:** verify that the simulation is realistic and gives consistent results
3. **Accessible:** affordable and easy to use by the target audience

Medicine utilises simulation for training of students and doctors at all levels. It is seen as a way to develop a greater understanding of complex systems, e.g. biochemical, physiological or pathological processes. Medical simulations can also allow practice to develop familiarity with instruments for technically difficult procedures and so simulate the likely outcome from a procedure performed in a certain way.

This project will be used primarily for teaching medical students the anatomy, physiology and biomechanics involved in the respiration cycle and how this is affected by factors such as respiration rate, diaphragm and rib motion and the effect this has on the surrounding organs. If successfully validated, it will then be used as a starting point on which to build a liver access procedure simulator.

2.2. Anatomy and Physiology

i.e. What the relevant anatomy and physiology of each organ is and how it moves during respiration

As mentioned above, it is necessary to simplify the anatomy to be represented in the simulation for several very good reasons. Firstly, it is only necessary to include that which will actually aid the student's understanding of the process. It is also necessary to remove organs that will only clutter the scene and impede the students' understanding. To this end, organs which have little involvement or are simply too complicated have been omitted.

From a point of view of computing power and time, very complicated structures such as the heart have had to be omitted as it would in fact be a project in its own right to model the cardiac cycle during respiration. Other organs have been left static such as the aorta and inferior vena cava which would obviously contract and expand during the cardiac cycle.

Those organs which have been included are:

- Muscular structures- diaphragm
- Bony structures- ribs, sternum, spine
- Costal cartilages
- Solid organs- lungs, trachea, liver
- Vessels- aorta, inferior vena cava

Respiration cycle:

The respiratory cycle is driven by the respiratory centre in the medulla oblongata in the brainstem. From this coordinating centre the skeletal muscles of the diaphragm and those involved in expanding the rib-cage are controlled via efferent neurones.

The respiratory drive is usually in response to arterial carbon dioxide concentration, P_aCO_2 (the partial pressure of carbon dioxide in the serum), but at higher levels of exertion or metabolic demand (e.g. exercise or infection) then other factors come into play such as hydrogen ion concentration, levels of arterial oxygen (P_aO_2) and other metabolites.

The normal respiratory rate is about 8-14 breaths/minute but this can increase to about 40 breaths/minute in times of maximal exertion or during illness, although this cannot be sustained form more than a few hours due to the significant effort required.

The respiratory cycle follows an approximately sinusoidal pattern as can be seen from figure 1. There is an inspiratory phase followed by an expiratory phase. Normal respiration occurs with a tidal volume (TV) of about 500ml of air in an average 70kg male, where the inspired air mixes with that left in the lungs at the end of expiration. The volume of air left in the lungs on normal expiration is called the functional residual capacity (FRC), a result of the spaces produced by the non-elastic or incompressible airways of the bronchi and bronchioles.

Maximal inspiration results in a vital capacity (VC) usually around 3.5 - 4 litres. At maximal expiration the volume left in the lungs is the residual volume (RV), the minimum physiological volume of the lungs.

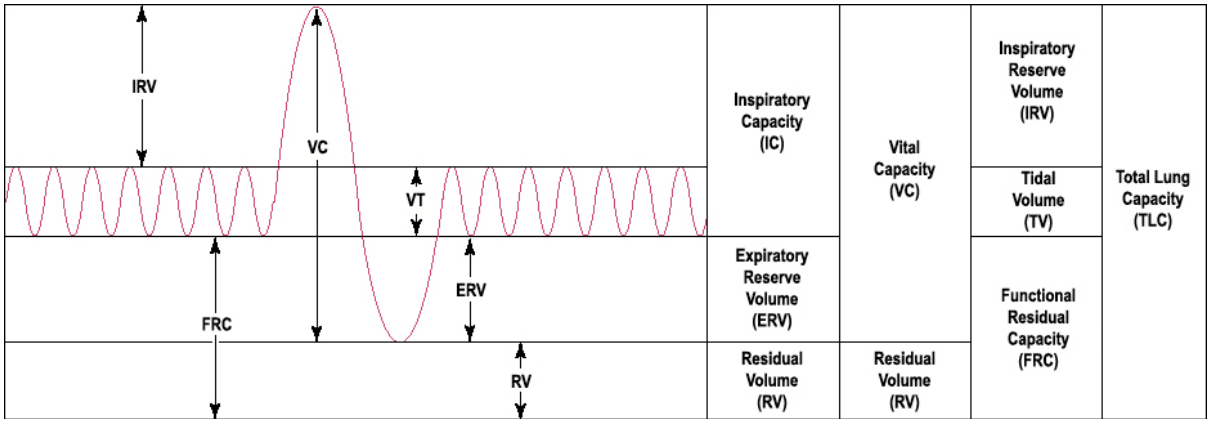


Figure 1. Respiratory volumes during normal breathing and at maximal inspiration and expiration[4]

2.2.1. Diaphragm

Accurate modeling of the diaphragm is central to the development of a realistic simulation as it is a surrogate for liver motion during respiration.

Anatomy:

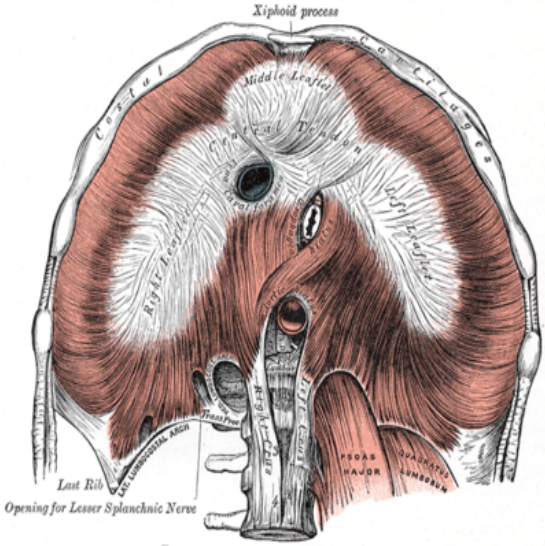


Figure 2 Inferior view diaphragm[5]

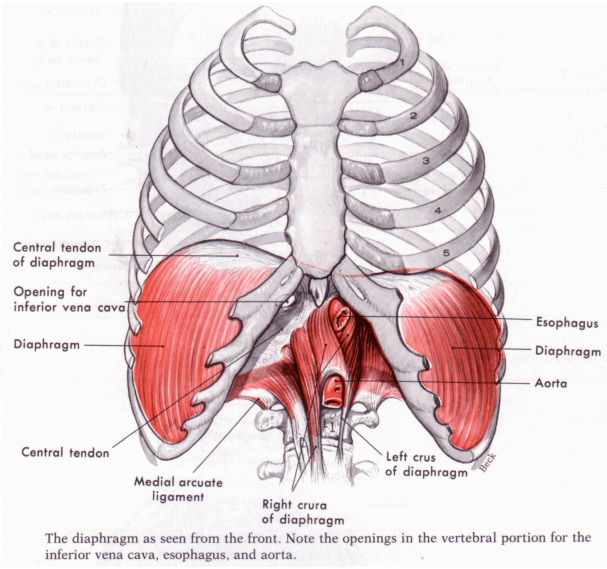


Figure 3 Anterior view diaphragm and attachments[6]

As can be seen from figure 2 and figure 3 the diaphragm is a thin layer or muscular and tendinous tissue. Its primary attachments are at the following positions [5,7,8]:

1. Sternal portion:
    - Anteriorly at the xiphoid process at the bottom of the sternum
2. Costal portion:
    - Anteriorly to the 5th-10th ribs and their costal cartilages
    - Laterally along the lower border of the ribcage
3. Lumbar portion:
    - Posteriorly to the thick aponeurotic medial and lateral arcuate ligaments which themselves are attached to the inferior tips of the 11th and 12th ribs and the spinous processes of the 1st Lumbar vertebra (these allow the lumbar psoas and quadrate (quadrates lumborum) muscles to pass behind the diaphragm).
    - Finally it forms the thick tendinous attachments to the 1st to 3rd Lumbar vertebrae called the cruae with the right crus being longer than the left, extending to the 3rd lumbar vertebra whilst the left extends only as far as the 2nd lumbar vertebra.
    - The aorta passes through the aortic hiatus inside the thick median arcuate ligament formed from the two cruae

- From these points of attachment originate the skeletal muscle fibres of the diaphragm, which form the dome shape attaching to the fibrous central tendon through which the inferior vena cava and esophagus pass.

Physiology:

The action of the diaphragm is complex and depends on its shape, attachments and surrounding structures[9]. The diaphragm can essentially be separated down into zones depending on their different movements:

1. Determined points at the bottom of the lateral rib cage, medial and lateral arcuate ligaments and crura which move only in relation to those structures.
2. Central tendon which is almost rigid and so is pulled and pushed in relation to the tension in the muscular apposition zones
3. Apposition zones composed of the skeletal muscle fibres which do all of the movement

During the respiration cycle the domes of the diaphragm move cranially and caudally. Figure 4 shows reconstructed MRI images of the diaphragm showing the change in its shape during respiration.

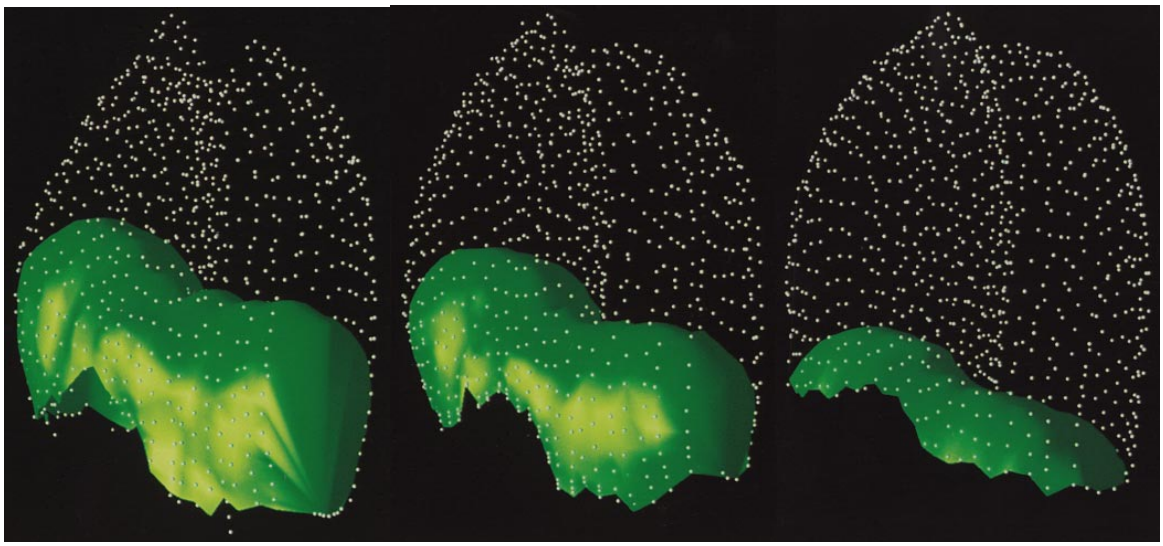


Figure 4. Reconstructed MRI images of the change in shape of the diaphragm between its relaxed state(a), at tidal volume(b) and total lung capacity(c) [5]

From Whitelaw et al.[10] it can be noted at relaxation the right dome is 2 cm higher than the left dome and 3 cm higher than the midline. At tidal volume the right dome is only 0.5 cm higher than the left dome and 1.5cm higher than the midline. This means that there is a rotation and slight flattening out of the central tendon during inspiration.

2.2.2. Thoracic Wall

Anatomy:

The major components of the thoracic wall are the bony structures of:

1. Sternum and the cartilaginous xiphoid process anteriorally
2. Thoracic spine posteriorally
3. Ribs between these 2 points connected by the flexible costal cartilages anteriorally and by fibrous ligaments across the synovial joints with the vertebrae posteriorally

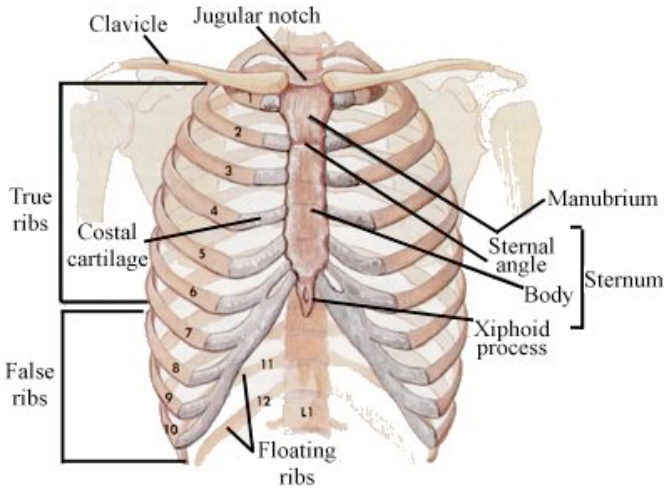


Figure 5. Bony structures of rib cage[11]

As can be seen from figure 5 the first six ribs are true ribs which have their own costal cartilages, ribs 7-10 being false ribs sharing a composite cartilage. The 11th and 12th ribs are floating ribs as they do not wrap round to fuse with the sternum.

The ribs as already mentioned are anchor points for the diaphragm but the ribs themselves are interconnected by a thin pair of muscles called the intercostals, composed of the internal and external intercostals muscles.

Accessory muscles also attach at various points including the scalene and the sternocleidomastoid muscles to the sternum and 1st ribs superiorally and the rectus abdomi muscles attach to the lower sternum and the 5th and 6th ribs and their costal cartilages inferiorly as shown in figure 6.

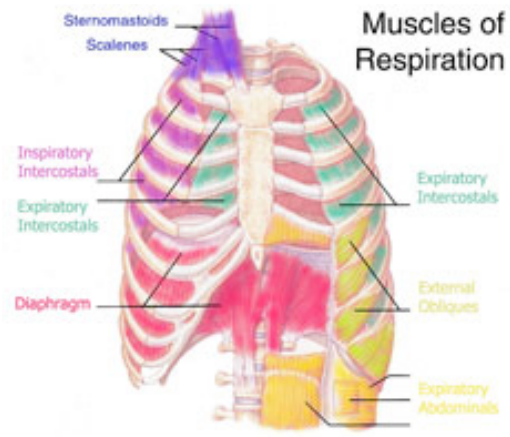


Figure 6. Accessory Muscles of Respiration[12]

Physiology:

The chest wall's motion depends on the complicated interaction of the structures mentioned above causing it to expand and contract during the respiratory cycle. The chest wall expands in-order to create a negative intra-thoracic pressure to draw air into the lungs, inflate the airways and allow gas transfer. When the chest wall contracts it pushes the expired air out of the lungs, to prepare for the next inspiration.

The main component which determines how much the chest wall expands, and so the degree of lung inflation, is rib movement. This has been studied over centuries and is tied intricately to the action of the intercostal muscles. It was classically thought that the internal intercostals pull the ribs closer together during inspiration and the external intercostals pull the ribs apart opening up the chest, increasing intra-thoracic volume, during inspiration. It has since been discovered that while this is largely correct the action is not as simple as this[9] with:

- Ventrally, the external intercostals contribute a significant expiratory effect from rib 6 onwards
- The internal intercostals switch to having a large inspiratory effect from rib 6 onwards

The main motions of the ribs have been described in terms of 2 actions:

- Bucket handle movement which increases the lateral excursion of the ribs
- Pump handle movement increases the anteroposterior diameter of the thorax

The net effect is essentially to increase the intrathoracic diameter in the anteroposterior and the transverse directions.

A study by Wilson, et al. [13] showed that each rib is contained in a plane that has a certain orientation relative with the original coordinate system (x, y,z) . They defined a new coordinate system (ξ,η,ζ), where ξ is the intersection of the plane of the rib and the sagittal midplane, η the intersection of the plane of the rib and the perpendicular of the sagittal midplane, and ζ is perpendicular to the plane of the rib (see figure 7.).

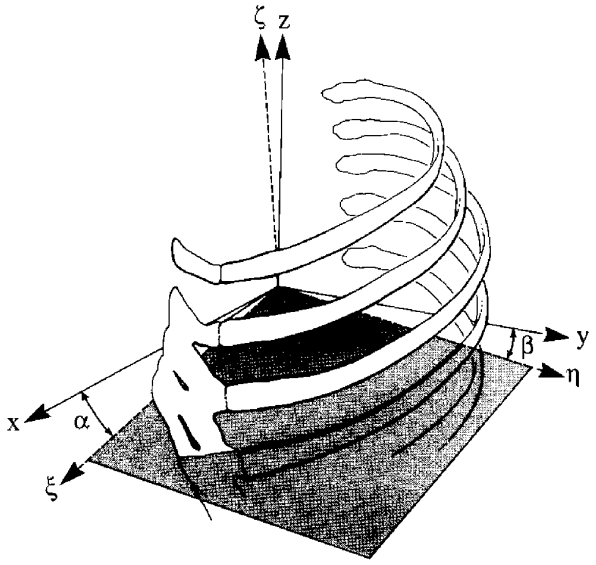


Figure 7- Wilson et al. definition α and β angles

Data was obtained at Total Lung Capacity (TLC) and Functional Residual Capacity (FRC) for the 2nd to the 9th rib. In particular, they obtained values for α and β, which are the angles between the new axis and the original ones. α is called the 'pump handle' angle, and β is called the 'bucket handle' angle.

All the ribs showed a decrease in both α and β with passive inflation to TLC. It seems logical that the plane of the rib has a tendency to coincide with the transverse plane. However, the changes in amplitude decrease. For instance $\alpha_{TLC} - \alpha_{FRC}$ went from 14.3 ° for the 2nd rib to 6° for the 9th rib, and $\beta_{TLC} - \beta_{FRC}$ went from 13.7° for the 2nd rib to 6.3 for the 9th rib.

Rib Number	$\alpha_{TLC} - \alpha_{FRC}$ *(degrees)*	$\alpha_{TLC} - \alpha_{FRC}$ *(radians)*	$\beta_{TLC} - \beta_{FRC}$ *(degrees)*	$\beta_{TLC} - \beta_{FRC}$ *(radians)*
2	14.3	0.2496	13.7	0.2391
3	11.4	0.1990	13.3	0.2321
4	10.7	0.1868	10.1	0.1763
5	9.6	0.1676	8.9	0.1553
6	9.4	0.1641	6.9	0.1204
7	7.9	0.1379	6.6	0.1152
8	7.9	0.1379	6.2	0.1082
9	6	0.1047	6.3	0.1100

Table 1. α and β angles ribs 2-9.

Movement of the lower 10th rib has not been presented but it should be noted that movement of the 7-9th do not vary much and as rib 10 is one of the false ribs its rotation angle will likely be similar to the previous 3 rotation angles.

The 11th and 12th ribs have minimum rotation as they are splinted by the quadratus lumborum muscles [14]. The net effect is to provide a fixed base for the diaphragm.

Another issue is the lack of a 1st rib rotation angle and after an extensive review of literature this cannot be found so again this will have to be estimated visually.

2.2.3. Lungs and Pleura

The lungs are surrounded by the pleura. The pleura itself is composed of 2 layers separated by the pleural cavity, a potential space (see figure 8). The thin visceral pleura lines the lungs and secretes fluid into the pleural cavity. The parietal pleura is a thick fibrous layer that closely adheres to the inside of the chest cavity.

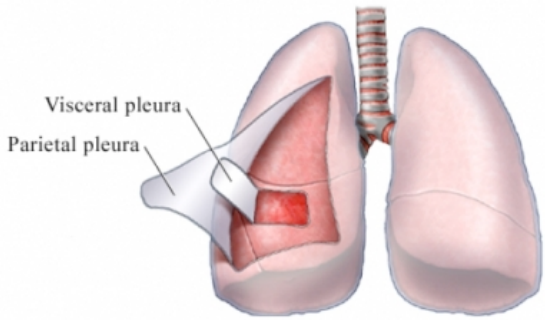


Figure 8. Lungs and surrounding pleura[15]

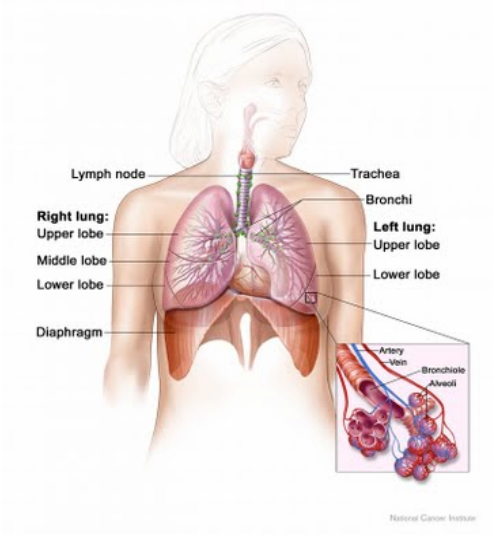


Figure 9. Trachea, bronchi and terminal airways[16]

The lungs are composed of three main components, the bronchial tubes along which inhaled and expired air passes, the pulmonary arteries and veins, and the pulmonary parenchyma, composed of the terminal bronchioles and alveoli which are responsible for gas exchange (see figure 9).

In terms of their behaviour the lungs are anchored at the hili (the roots) to the main bronchi and the pulmonary veins and arteries. During the respiration cycle the lungs expand and contracts, following the chest wall, expanding into the pleural space.

2.2.4. Liver

The liver is a fragile vascular structure surrounded by a tough membrane. It is attached to the diaphragm by the falciform ligament which splits the liver into left and right lobe. Its motion during breathing follows that of the diaphragm.

2.3. Physical Modelling

i.e. Basic overview of differences geometrically and physically based models and then talk about each of the physical models, what each type involves and pros and cons

2.3.1. Mass Spring Model

A Mass Spring model is a type of dynamic model where the position of the object is dependent on time. The object is represented as a particle cloud, where each particle represents a vertex defined in the topology of the original object data. The particles are connected together by damped springs which correspond to edges between vertices in the object.

Each particle has a resting position, a current position, a mass, a velocity and a net force applied. Calculation of the force occurs in real-time where it is dependent on the position of the neighbouring particles connected by springs.

This method of deformation is fast and easy and closely approximates the real-world physical behaviour of tissues. Along with its speed and capability means that it is the most commonly used method of tissue modelling.

Basic Mass Spring Model

Using Newton's second law of motion it is possible to derive the position of a particle in space as:

F=ma **w**here **F**= force, **m**=mass, **a**=acceleration
As, **a= dv/dt** where **v**= velocity, **t**= time
And **v=dx/dt**, therefore $\mathbf{a=dx^2/d^2t}$
Therefore $\mathbf{F= m * dx^2/d^2t}$

This means that to find the position is a case of solving a 2^{nd} order differential equation once the force has been found.

In a Mass Spring system the force is usually assumed to be linear and is found from Hooke's Law:

F= -k * Δx where **k** is the spring constant and determines the rigidity of the spring and **Δx** is the difference between the current position and the resting position of the end of the spring.

Fortunately this is fairly simple to calculate from the positions of the particles in 3D space, the complexity arises when there is a network of springs as the forces have to be accumulated through all the neighbouring springs and then passed through the network iteratively.

Damping has to be introduced into the system to increase stability and simulate energy loss due to friction and even air resistance.

The resulting position of the particle is then calculated using a 2^{nd} order differential equation solver using for example the explicit Euler or Runga Kunta methods.

The problems with this method are:

- The behaviour can become quite unrealistic for large deformations.
- There is also a limited volumetric behaviour because of the local structure of the mesh
- The Mass Spring system has a predisposition to oscillate because of its iterative structure.

Relaxation:

It has been recognised that the problem of springs being stretched unrealistically in relation to others is a major problem in Mass Spring systems. Relaxation is a heuristic that has no direct physical interpretation that has since been applied to remove this problem[17]. As it sounds it iteratively transforms the springs to their relaxed configuration.

It is calculated as follows:

$L_{relaxed} = L_{natural}$ + linear factor. $L_{natural}$ *if* ($L_{actual} > L_{natural}$ + *linear factor.* $L_{natural}$)
or: $L_{natural}$ - linear factor. $_{u}L_{natural}$ *if* ($L_{actual} < L_{natural}$ - *linear factor.* L_{natral})
or: $L_{natural}$ else

Essentially the relaxation algorithm displaces the nodes once the difference between the initial length ($L_{natural}$) plus the current length (L_{actual}) is greater than the linear factor.

The net effect is to allow deformation to propagate through the model quickly. The issue with this method is that form large deformation models it can result in a big computation cost.

Quasi-Static Mass Spring Model:

Implementing this model involves finding an approximation to the dynamic solution[17]. There is no mass, inertia and no damping is required as no energy introduced into the system. This should resolve the oscillation problem in the basic model, meaning that no waves or vibrations will pass through the system.

This is achieved by assuming that the internal and external forces are in equilibrium, i.e.

$$\sum \text{internal forces on particle} - \text{external forces on particle} = 0$$

An iterative algorithm is used to this end where an attempt at calculating the configuration where the forces balance during each frame interval is performed. To help this succeed each iteration is seeded with the solution of the previous as this might be close to the solution for the configuration of that frame.

If the frame interval has been reached and a no solution where equilibrium is achieved is reached, then the current (best) result is used.

2.3.2. Finite Element Models

A Finite Element Model (FEM) is based on the law of continuum mechanics and computes a solution closer to the continuous solution rather than a discrete solution acquired by the Mass Spring models. In a FEM the object is subdivided into a mesh of simple elements, such as tetrahedra or triangles. Physical properties, like elastic modulus, stress, and strain, are associated with each of these elements. The corresponding equilibrium equations are solved numerically by the approximation provided by a polynomial equation which is related to key control nodes.

The advantage of the FEM is that it has a greater accuracy compared to Mass Spring models, and the results can be very realistic, but it has a very high computational cost and is far more difficult to implement than the mass spring system.

The high computational cost means that the FEM is rarely used for real-time simulations although methods such as the Boundary Element Method solves equations for unknown displacements and forces only on the boundary of the object, rather than the interior as with FEM. The movement is then governed by a large linear system of equations. Such a system could normally not be solved in real time, but with some pre-computations, it becomes possible to obtain at each step a modified system with only few updates compared with the previous system. This allows for fast and accurate deformations.

2.3.3. Tensegrity

Tensegrity is a blend of methods using both rigid and elastic links in order to form stable structures[18]. The elastic links make part of its function very similar to that of the Mass Spring systems but the main part of the model is constraining the simulatation by the rigid links.

2.3.4. Chainmail

The Chainmail model is another method developed by Gibson[19]. It is a geometrically based method that was developed to preserve the size and geometry of volumetric objects during deformation in surgery simulations. It was specifically designed to speed computation and so can be used with large datasets.

In this model, the object is defined as a set of point elements. The elements are interconnected as links in a chain. Each point can move freely without influencing its neighbours whilst within certain pre-defined limits. When one element of the object is moved and reached its maximum or minimum geometric size, the neighbours are then deformed accordingly, setting up a chain reaction that is governed by the stiffness of the links in the mesh. This has the net effect that any deformations are usually absorbed in small areas rather than as in the mass spring model.

In FEM, there are complex calculations on a small number of elements. The Chainmail model is doing simple calculations on a large amount of elements. As in the Mass Spring model there is also a relaxation step in which the energy of the configuration is minimized.

As with the Mass Spring model its' great advantage is its' simplicity[20]. The volumetric behaviour is guaranteed by the chainlike structure of the model. It is also fast deformable model but required some optimization of Gibson's initial algorithm to model inhomogenous data[21].

2.4. Previous work

As mentioned, this project builds on previous work. Below is a summary of previous projects that this project uses:

i. Initial project 2008: "Modelling Diaphragm Kinematics for 3D Simulation of Liver Access" [1]

   This focused on the movement of the diaphragm during respiration separating out key anatomical and physiological zones. Non-rigid registration techniques were applied to record tension and displacement vectors. The main contribution of this project is to show the areas of the diaphragm movement.

ii. Follow-up project 2008: "The Use of Tensegrity to Simulate Diaphragm Motion Through Muscle and Rib Kinematics" [2]

   This project used the Step forward as diaphragm motion and rib movement were simulated with a mass-spring based method. Good modelling of the diaphragm and close approximation to physiological results from patient CT images were achieved, but adding in rib kinematics slowed simulator considerably. Another issue was that the simulator was written in Java and not possible to include haptic APIs.

iii. Late 2008 project: "Computer-based multi-sensorial environment for anatomy teaching: Dynamic modelling of the rib cage anatomy during respiration" [3]

   Attempt to convert simulator to C++ using the H3D development framework to allow haptic interaction. Use of chainmail rather than mass-springs/tensegrity for deformation. Good attempt and key organs modelled, but not anatomically or physiologically realistic. Another problem was that the deformation based on the chainmail algorithm with haptics was too slow to be off any use for training. Comparisons with CT images showed result slightly worse than the mass-spring based method.

3. Materials and Methods

3.1. H3D- Adapted from reference 22

H3D is an open-source, cross-platform, scene-graph API[22]. It is written in C++ using the Standard Template Library (STL). H3D uses the Open Source Graphics Library (OpenGL) for graphics rendering and Haptics API (HAPI) for haptics interaction.

H3D is designed using the extensible 3D (X3D) graphics file format, an ISO standard file format based on the XML (Extensible Mark-up Language) standard. X3D is used for scene-graph design and allows rapid development adding new functionality in a modular way.

Programs can be written to run with the H3D API in 3 different languages- C++, Python and X3D. Each has their own advantages and disadvantages:

- **X3D:** Allows rapid modular development but programs written entirely in X3D, they will be slow and are only able to use the pre-defined X3D/H3D nodes and fields (see section 3.1.1), limiting adaptability.
- **Python:** Allows relatively fast development but again programs would have a reduced run-time speed as python is an interpreted language. However, user-defined H3D nodes can be developed allowing greater adaptability.
- **C++:** Slower and more intensive development as it is a compiled language but results in greatly improved run-time performance and maximum adaptability.

In practice a combination of these 3 languages is used to gain maximum benefit from each.

3.1.1. Fields and Routes

Fields are the most fundamental building blocks of X3D and the H3D API. Fields have several functions:

- **Data storage:** fields contain data of some type
- **Data dependency:** fields can be set up to be dependent on each other so that a change of value of one field triggers an update of another (see the update() function in section 3.1.3).
- **Functional behaviour:** fields can calculate values in any way they want, depending on the fields routed to them.

Fields are usually of type SFields or MFields, where the SField contains a single value of some type, e.g, SFVec3f contains a single Vec3f or SFInt32 contains a single 32 bit integer value. MFields contain multiple values, e.g. MFFloat contains multiple floats.

Dependencies between fields in H3D are specified by connecting them by routes. These are arranged in a directed graph, creating a field network where events are passed from field to field according to how the field network has been constructed.

A route between field A and field B means that if something changes in field A an event message is sent to field B to let it know that A has changed and B can take appropriate actions. By default, routes can only be set up between fields of the same type.

3.1.2. Nodes

Nodes are containers for fields, grouping together fields to create larger reusable entities. They are used to build the X3D scene-graph as they control all the behaviours of the composite fields, specifying the interface to itself, determining what fields the user of the node can access and then hides away all the internal functionality from the user.

Fields in a node can be one of only four types:

- **inputOnly –** Input fields to the node. Can only be set and routed to. It is not possible to get the value or route from it.
- **outputOnly -** Output fields from the node. Can only route from and get the value, not route to or set.
- **inititalizeOnly –** Value of these fields can only be initialized. After that the value can only be used as an outputOnly field, with no routing to or setting.
- **inputOuput -** No restrictions, can both get and set the value as well as route to and from the node.

3.1.3. Important functions and fields

Global field member functions

Important functions that are available for all fields are:

- route(Field) - sets up a route from a field to another. An event is generated.
- routeNoEvent(Field) - same as route, but no event is generated.
- unroute(Field) - remove a route.
- touch() - manually generate an event from this field.

For SFields and MFields:

- getValue() - get the value of the field.
- setValue(value) - set the value of the field.

Traversing the scene-graph

The two important functions that for traversing the scene-graph are:

- render() - makes all the OpenGL calls in order to graphically render the node.
- traverseSG(TraverseInfo &ti) - called upon traversal of the scene-graph. The TraverseInfo object contains information about the coordinate space, current surfaces, haptics devices that are active

Note that the scene-graph is traversed twice each loop, one time to do the graphics rendering and one time for other things like adding haptics primitives.

Update function

When the value of a field is called by e.g. the getValue() function or by a route the following happens:

- If the field is up to date, just use the current value.
- If not, an event has been received and we have to update the value.

This is called lazy-evaluation. The update() member function of the field takes care of updating of the value. The default case is simply to just copy the value of the incoming event, but it can be changed to do any arbitrary calculation by specialising the update function yourself. In C++ the default case would look like:

```
class SFFloat: public Field {
        virtual void update() {
                value = static_cast< SFFloat >(event.ptr)->getValue();
        }
}
```

This simply sets "value" to the value of the field that caused the event to happen e.g. from a route.

A more complicated conditional update to set the value to 0 or 1 would be:

```
class TrueOrFalse: public SFInt32 {
        virtual void update() {
                value = 0;
                if(static_cast< SFFloat >(event.ptr)->getValue() == 1)
                        value = 1;
                }
        }
```

AutoUpdate field

If lazy evaluation is not wanted, i.e. update is wanted every frame whether the value is called or not then this can be done by using the AutoUpdate field, which calls the update function every frame. In C++ this would be:

```
class PrintInt32: public AutoUpdate< SFInt32 > {
        virtual void update() {
                SFInt32::update();
                cerr << value << endl;
        }
}
```

TypedField

Routes can be set up between fields of different types by using the TypedField template modifier. The TypedField modifier allows specification of what type the route must have. In C++ you would specify the class as follows:

```
class MyField: public TypedField< SFFloat, Types< SFBool, SFFloat > >{
        virtual void update() {
                bool b = static_cast< SFBool * >( routes_in[0] )->getValue();
                H3DFloat f1 = static_cast< SFFloat * >( routes_in[1] )->getValue();
                if(b=1)
                        value= f1;
                else
                        value= 0;

        }
}
```

The first argument given to the TypedField template states that the base class is of SFFloat and the Types template specifies that it takes in an SFBool and an SFFloat. The Boolean determines whether it is set to the incoming route value or zero.

3.1.4. Creating User-defined Nodes

In any custom H3D program it is necessary to define and create new nodes. This is done in either C++ or Python.

When designing new nodes it is necessary to first of all determine what fields should be available in the interface to the node. After resolving this, dependencies between fields have to be determined to create an internal field network that allows the node to perform as required, using a combination of the above functions and fields.

For a complete description of how to write simple nodes in H3D please refer to reference [15].

After the new node has been written it is possible to use them in X3D files by either combing them with the executable X3D viewer (in the case of H3D, H3DLoad) or to compile the nodes into Dynamic Linked Library (DLL). This is preferred as you keep nodes created for specific purposes out of the X3D viewer but can simply include them as required by importing the DLL.

3.2. Analysis Previous Model

As mentioned earlier, the aim of this project was to build on the work done during previous years, especially on Mathieu Jacob's project[3] which was the first attempt within the Department of Biosurgery and Surgical Technology at Imperial College London to implement a respiration simulation under the H3D development framework.

A logical place to start is with a full analysis of the problems with the previous simulation that resulted in it being deemed unsuitable for use as an anatomical and physiological teaching model. After completion a list of improvements to be made will be compiled.

The viewing of the previous project was performed on a dual core 2.1 GHz machine with 3 GB of RAM and a 256 megabyte ATI Radeon HD 2600 graphics card. The operating system was the 32 bit Windows Vista Service Pack 1.

3.2.1. Simulation of Skin

The movement of the skin on top of the chest is unrealistic with poor deformation of the skin with the respiration cycle and no fixed contact points at the shoulders, resulting in gaps appearing in which the underlying organs can be seen.

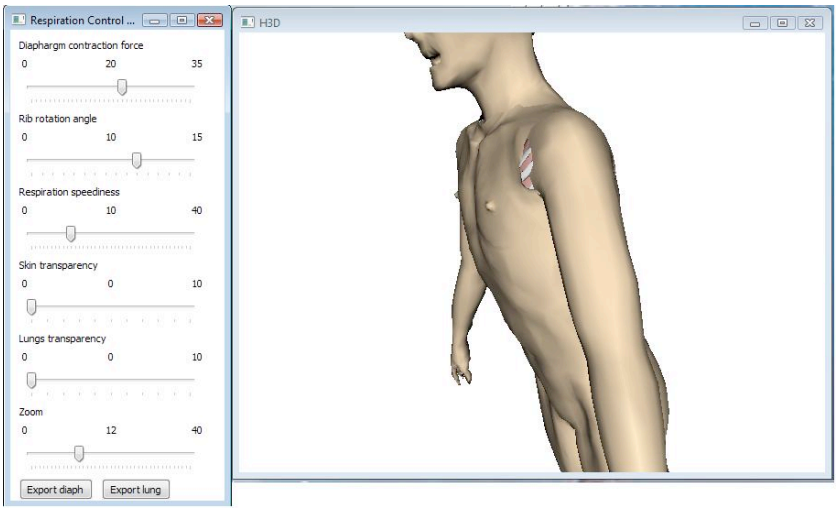


Figure 10. Screenshot of skin deformation

There is also no movement of accessory muscles in the neck, such as the sternocleidomastoids or scalene. There is also no contribution from the rectus abdomini muscle and no abdominal movement at higher respiration rates.

3.2.2. Rib and Sternal Motion

The rib motion looks some-what unrealistic with a swinging motion that is applied almost equally across all the 11 ribs shown, rather than the physiological tapering that occurs and has been documented. Also the 11th and 12th ribs (not shown) are static due to the aforementioned splinting by the quadrate muscles.

Sternal motion is difficult to determine with no previous figures available, so will be assumed to be satisfactory at the moment.

The absence of the costal cartilages seems to me, to exacerbate the swinging motion of the entire rib-cage and detracts from the correct motion of the upper-ribs.

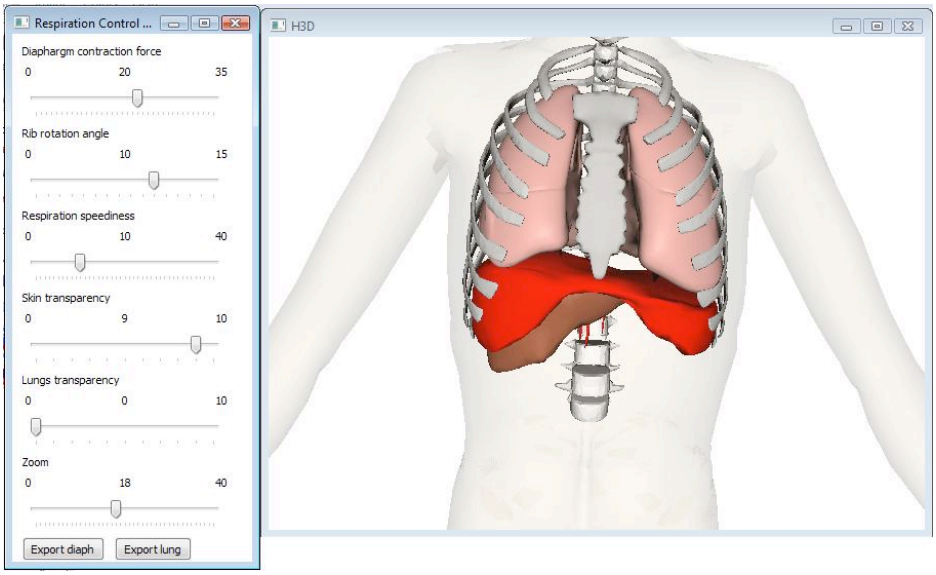


Figure 11. Screenshot of rib, lung and diaphragm

3.2.3. Diaphragm

The diaphragm has been modelled well at the bases with the rigid portions represented by the medial and lateral arcuate ligaments being appropriately modelled, attached to the transverse spinous processes of L1, although there are no rigid attachments to the fixed 11th and 12th ribs. The left and right crura are again well represented again being fixed to the underlying lumbar vertebrae. The chainmail does give a sense that it is a muscular flexible organ under tension.

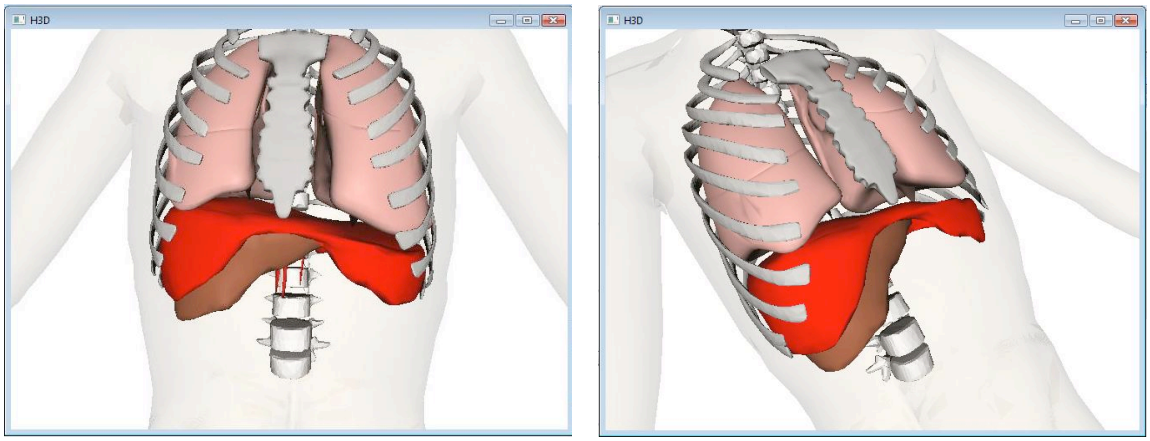
The motion of the diaphragm seems rather mechanical and seems slower than the rib-cage. It does not follow the motion of the rib-cage as it should do, expanding out at the lower rim with each breath. At greater inspiratory volumes there is collision with the underlying liver

and there is motion of the arcuate ligaments which is completely artificial. Even at lower inspiratory volumes there is collision with the lower ribs bilaterally.

3.2.4. Lungs

This is probably the most unrealistic part of the current model with the lungs not being fixed anywhere in space, rather floating and being pulled and pushed by the motion of the surrounding structures it is anchored to.

There is little deformation of the lungs with no appreciable change in volume during the respiration cycle. The touch points on the diaphragm and ribs look completely artificial and detract from the model.



Figures 12 and 13. Screenshot lungs moving during respiration

3.2.5. Controls

The original graphical user interface (GUI) is intuitive and easy to operate. Issues that require updating are:

- Ability to hide all bony tissue leaving the soft-tissue organs displayed
- Ability to hide the diaphragm to display the underlying liver
- Make a little more physiological rather than using common terminology

3.3. Implementation

The implementation of the project will build on the source code from Mathieu's model and so will be a step-wise improvement on his work and in many sections part of his code has been reused for simplicity and speed of development.

The early stages of the project were occupied with learning the basics and later the intricacies of X3D, H3D and the C++ implementation of H3D nodes. This has not been trivial, as having limited programming experience in C++ and Python and certainly none in X3D and H3D, work has had to start at a very basic level, learning and working up to deformation of the actual organs themselves.

3.3.1. Organ Modelling

i.e. What models have been applied to what and how

The model has been implemented using a dynamic mass-spring model rather than the preferred quasi-static mass spring model. The reason for this was that due to my inexperience and the enforced time constraints after the initial delay getting the required software to work on my machine and the time to familiarize myself with X3D and H3D, I decided to maximize the time spent working with the actual organ deformation, rather than optimizing the underlying deformation model.

Although not ideal, it offers the opportunity to study the application of a widely accepted, tested and documented model under H3D that will hopefully give some interesting results and it would not be a massive task to implement the quasi-static mass spring on the current system in the future.

3.3.2. Node Implementation (see appendix 1 and 2 for c++ source code)

DeformableShapeNew:

The basic ***DeformableShape*** node is a simple ***X3DShapeNode***, which has the added ability to have its' geometry deformed when touching it with a haptics device. As the ultimate aim is deform the geometries of the objects by stretching the vertices and to use haptics under H3D it is necessary to implement under this than a simpler ***Shape*** node.

In the ***DeformableShape*** node are the following unique non-inherited fields:

- ***deformer-*** contains an H3DCoordinateDeformerNode that determines how the deformation should be done

- ***origCoord-*** contains the original coordinates of the object before any deformation began
- ***deformedCoord***- contains the coordinate after the deformation and is used for the graphics rendering
- ***restingCoord-*** contains the coordinates that the geometry will go back to when there are no contacts to the geometry any longer, (if the deformation is non-plastic, this will be the same as origCoord)

New nodes have had to be created that are organ specific as the organs being modelled are inhomogeneous with widely differing structures and behaviours.

Figure 14. Inheritance diagram for DeformableShapeNew and derived nodes

The inheritance structure is seen in figure 14 where there is a general ***DeformableShapeNew*** which has all the required common functionality and the derived nodes for the diaphragm, lung and the cartilage.

To implement the mass spring system requires calculation of the neighbours, their indicies and some manner of finding them in real-time. The ***DeformableShapeNew*** node contains vectors that contain all the neighbours of every coordinate.

As is required in the mass spring system there are also fields that contain all velocities and forces for every particle, i.e. vertex. It also contains fields with the specified anchor nodes which are fixed and those with pre-defined movement. From these MFVec3f fields, arrays stating which point is anchored or moving is created so that the deformation on these vertices once they are passed onto the deformer node is either pre-defined or as in the case of anchored vertices non-existent.

The node also has routed to it the values for speed of respiration and magnitude of contraction and the degree of rib-rotation.

The derived nodes will be spoken about under the organ/structure being modelled.

MassSpringDeformerDynamic:

Implementation of the dynamic mass-spring model is in the ***MassSpringDynamicDeformer*** node in the important deformPointsNew function which takes in all the original coordinates, deformed coordinates of the current frame, the pre-accumulated velocities for every vertex. It then performs calculations using the user defined **Spring Coefficient** and **Damping Factor** and **Mass** of that organ/ structure.

Once solved using an explicit Euler solver, it then updates the velocity and the position of each of the vertices, before changing the deformed coordinates of the ***DeformableShape*** node to be graphically rendered.

The ***MassSpringDynamicDeformer*** node also contains a function called relaxation, which relaxes the mesh every frame, to prevent excessive lengthening of the springs.

RotTransform:

This is the same node used from Mathieu's project which has fields for the centre of the rotation and the composite 'pump handle' angle (α) and 'bucket handle' angle (β). This also creates and manages the touch points for motion of the chainmail objects.

3.3.2.1. Diaphragm

The work done on the diaphragm simulation[1] showed that the following:

1. Central tendon shape largely does not change but right side pulled down more than left side in relation to respiration (1.5 cm more) so will need to be shortened or rotated
2. Apposition zone dimensions change remarkably between relaxation and maximal at total lung capacity where the size of the apposition zone is essentially zero when the diaphragm is flattened [9]
3. Fixed points do not vary much except in relation to underlying attachment change

The diaphragm as mentioned earlier required separation out into several sub-structures with different properties and actions, allowing a more realistic motion can be more easily tailored.

The diaphragm was split into regions using Meshlab® (an open source 3D mesh editing and rendering package) to create lists of selected vertices that were to be either anchored, moving or rotated.

The regions correspond to the following:

1. The fixed points of the crura, aortic hiatus, arcuate ligaments, the lower margins of the rib-cage and costal cartilages.
2. Moving apposition zone where shape changes in relation to force contraction and the tension (force per unit length) assumed to move out radially from central tendon to the anchored points.
3. Central tendon where movement is generally sinusoidal in the cranio-caudal direction but if time allows required a transform to flatten and rotate it slightly

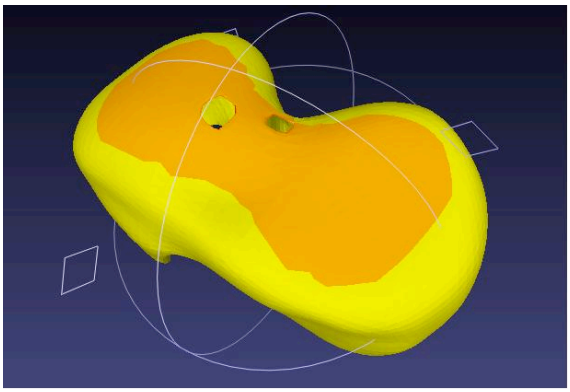
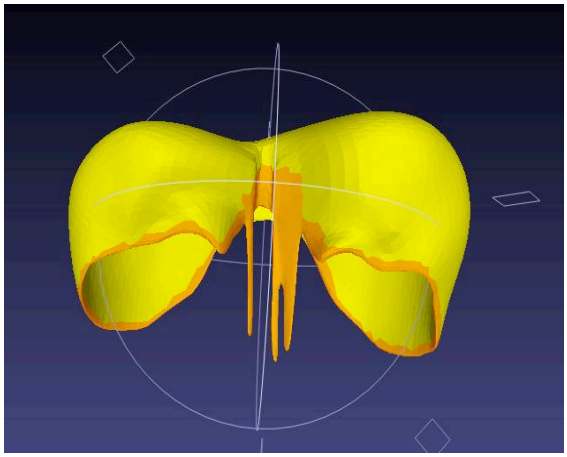
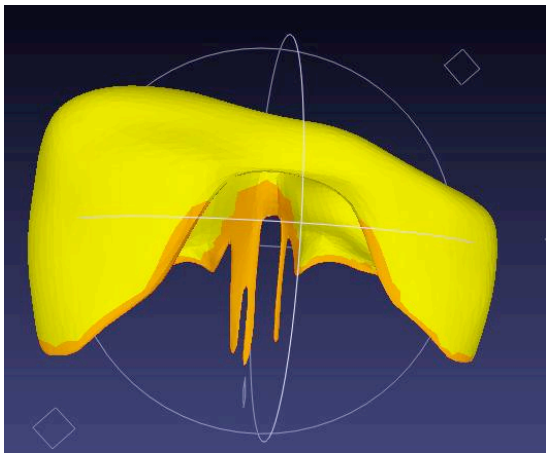


Figure 15. Central tendon region delimited in Meshlab





Figures 16 and 17. Anchor points at crura, and lower margin

The apposition zones were the region in which the mass-springs were allowed to act, with the forces determined by the movement of the central tendon region.

Some decisions had to be made as to what was feasible:
Attachments to the costal cartilages were going to be too difficult to model as this structure itself had the mass-spring model so would be in a constant state of flux giving unpredictable results, so a fourth region containing the fixed but rotating points of the lower margins of the rib-cage and costal cartilages was created.

DeformableShapeDiaphragm	Fields for: • fixed vertices

	• moving points central tendon
	Functions: • order the moving points in the same way as the original coordinates for graphics rendering

3.2.1.2. Thoracic Wall

The motion of the chest wall is complex and with so many structures involved, it obviously required simplification. The motion of the ribs is determined primarily by the intercostal muscles, but as they would obscure the view of the rib-cage and not being available they have not been modeled. Rather their net effect on the ribs themselves have been modeled, the pump and bucket-handle motion described earlier.

Ribs:

The ribs themselves are modelled as rigid structures which do not deform and rotate during the respiration cycle increasing and decreasing the intra-thoracic dimensions. The bulk of this work was done during the previous project and is defined in the RotTransform node which rotates by the α and β angles defined earlier.

The contribution here has been to apply the correct angles to the rotation of ribs 1-6 bilaterally and apply a more natural looking movement to the false ribs (ribs 7-10 bilaterally) which is essentially an approximation.

Costal Cartilages:

The costal cartilages are flexible cartilaginous structures that bend and flex during the respiration cycle, allowing the thoracic dimensions to change whilst maintaining the integrity and strength of the rib-cage itself. They are therefore modelled as rigid mass-spring systems.

The composite costal cartilage was broken down into individual cartilages to allow a greater modularity and reuse of code, rather than creating a clunky node for the whole cartilage. This did however require the ability to specify multiple attachments for the large composite cartilage anchoring to the 7-10th ribs.

To simplify the number of varying attachment positions it was assumed that the points at the sternal edge were anchored and so did not move. The points simply followed the

rotation of the sternum during the respiratory cycle. Rather, the deformation requires the vertices attached to the tip of the articulating rib to deform.

The anchored and moving vertices had to be delimited using Meshlab (see figure 18) The sternal edge is obviously anchored to the sternum. The ends which articulate with the ribs are altered in accordance to the updated position of the rib with which they are articulating, being stretched towards it during the respiration cycle.

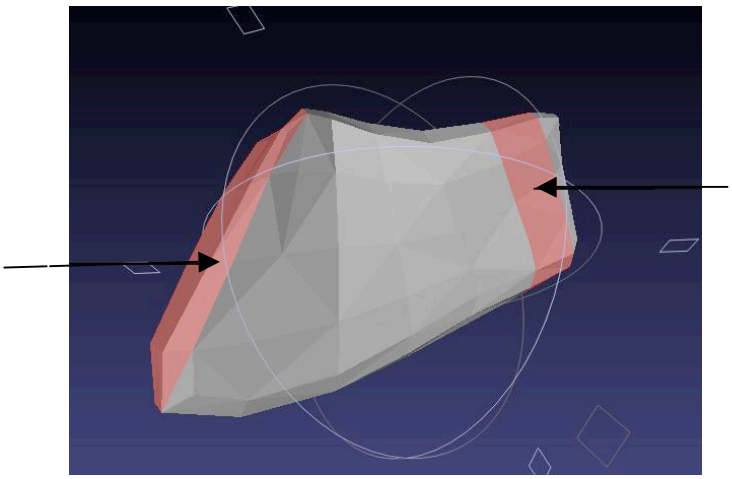


Figure 18. Cartilage with sternal anchor region and moving vertices delimited

To achieve this, the following nodes had to be created (see appendix :

DeformableShapeCartilage Sternal anchor	Fields for: Moving vertices • anchored vertices • moving vertices • routed rib tip vertices to follow once rotated
	Functions: • order the moving points in the same way as the original coordinates for graphics rendering • displace the moving vertices towards the rib tip point using a vector calculated from the subtraction of the routed rib tip point and the centre point of the cartilage
RibRotTransform	Simplified version of RotTransform that takes in the SFVec3f or MFVec3f and rotates it before routing it back
RibShape	Extra output field for the rib tip vertices, otherwise

	same as basic ***Shape*** node

Initially a basic algorithm that simply outputted a single point at the end of each rib was used with fairly poor results. This has been improved by using a matching algorithm where all the rib tip vertices are utilised. The algorithm works by initially finding the closest rib tip vertex to each of the moving cartilage vertices, storing the index of the rib tip vertex for future use in an array, so that for every frame, the cartilage vertex is moved to the rib tip vertex's position.

3.2.1.3. Lungs

The lungs as mentioned earlier were the part of the model that required the greatest improvement. The essential movement of the lungs which needed to be accurately reproduced was that of expansion and contraction during the respiration cycle.

To achieve this firstly the lung was separated out into zones which have different motions during the cycle with the mass spring model implemented between these regions. The regions correspond to the following:

1. The fixed region which moves very little during the respiration cycle, centred around the hili.
2. Next there is the base which largely mimics the motion of the diaphragm, following it closely to a point, having a sinusoidal motion applied to it.
3. The surfaces which oppose the ribs which again largely mimic the motion of the rib-cage, following them closely during the respiration cycle as their volume is dependent on this motion. This was achieved using the RibRotTransform node to rotate the lung coordinates. 3 zones on the concave surface of the lung were created.
    i. Upper zone following rib 1 rotation
    ii. Middle zone following rib 3 rotation
    iii. Lower zone following rib 5 rotation

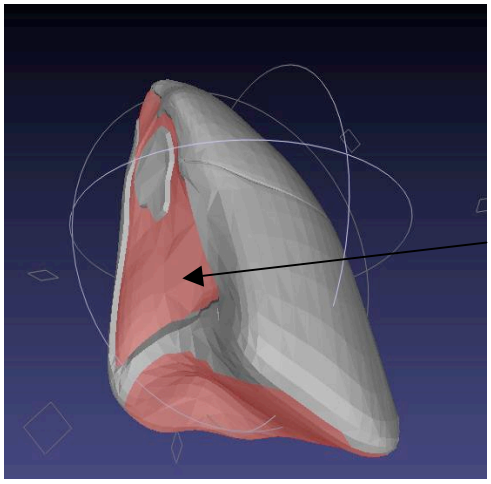


Figure 19. Fixed region around the left hilium and the moving base

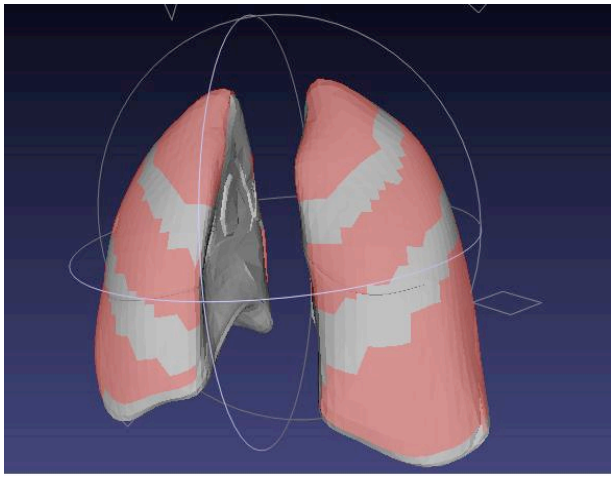


Figure 20. The moving upper, middle and lower zones on both lungs

Again a new ***DeformableShape*** node had to be created. Its fields and functions are as follows:

DeformableShapeLung	Fields for: • fixed vertices • base vertices • upper zone vertices for routing • middle zone vertices for routing • lower zone vertices for routing • upper zone vertices once rotated • middle zone vertices once rotated • lower zone vertices once rotated
	Functions: • order the moving points in the same way as the original coordinates for graphics rendering

3.2.1.4. Liver

The liver being a viscous organ with a rigid capsule was left as a chainmail as a mass-spring model would offer little improvement over the chainmail.

3.2.1.5. Skin

The skin has had the mass spring model applied to it. Again regions to be rotated have been delimited.

4. Results

In this section only those improvements which have been implemented will be discussed. The results have been obtained as above on a dual core 2.1 GHz machine with 3 GB of RAM and a 256 megabyte ATI Radeon HD 2600 graphics card. The operating system was the 32 bit Windows Vista Service Pack 1.

4.1. Mass-Spring Implementation

The actual dynamic mass-spring implementation has worked out well with initial testing producing a very realistic spring behaviour on single triangles and composite structures. The relaxation algorithm seemed to offer little benefit for the computational cost that was going to be required for all the vertices in the final model.

The dynamic mass spring system is tuneable with 3 variables that can be altered on each organ, namely the spring coefficient, damping constant and its' mass. The most aesthetically and realistic values have been selected at present but these offer a research opportunity to determine optimum values.

4.2. Diaphragm

The diaphragm has been created as essentially a central tendon being moved in the required manner, with the mass springs between the anchored points and the moving central tendon.

To maintain an acceptable speed of animation whilst maintaining a satisfactory visual representation, the diaphragm mesh was simplified using the quadratic edge collapse decimation algorithm in Meshlab. This reduced the number of vertices to 50% of the original.

Once implemented, simple trial and error were used to select the best values for the mass spring variables:

Variable	Spring Coefficient	Damping Constant	Mass
Value	0.9	0.9	6

Table 2 Spring coefficient, damping constant and mass of diaphragm

At present there has been no rotation of the central tendon with flattening out during the respiratory cycle, this could be done by updating the moving positions during the respiration

cycle but time simply would not allow this. Time was simply not available to implement this improvement.

One criticism is that there is marked collision between the diaphragm and the lung bases and between the diaphragm and the costal cartilages anteriorally.

Another criticism is that the diaphragm does not rotate with the rib-cage expansion and so is floating unattached laterally and anteriorally as there is no sternal attachment anteriorally that would have added increased realism and possibly reduced the collisions. This being said, the movement is relatively realistic at low respiratory rates and the softer movement is definitely an improvement. The perceived level of tension provided by the chainmail does lend itself more to a feeling of a muscle than the rather cloth-like mass-spring motion.

Other issues are that the polygonization of the mass-spring does detract from the smoothness of the model and the diaphragm does tend to reach its minimum energy configuration with it losing shape at the edges faster, hence resulting in collisions with the liver as the animation progresses.

4.3. Thoracic Wall

Ribs:

As mentioned this work was largely complete during the previous project but required some alteration and correction of the rotation angles which seemed rather unrealistic and especially with the movement of the 11^{th} rib and absence of the 12^{th} ribs bilaterally. This has since been corrected and looks more natural, added to by the moving costal cartilages.

Costal Cartilages:

The values of the mass spring parameter for the costal cartilages are:

Variable	Spring Coefficient	Damping Constant	Mass
Value	0.9	10	15

Table 3. Spring coefficient, damping constant and mass of costal cartilages (if different)

The movement of the costal cartilages seems rather realistic at low respiration rates when the displacement of the rib tips is less but when there is greater rotation, it seems rather sluggish and difficult to update smoothly. This is likely the result of the way X3D and indeed H3D routes the values between the different shape nodes, as they will be one frame behind and so an error will always be there- minimising this is the only way to improve this.

The goal of representing them as flexible cartilaginous structures has however been achieved and adds to the understanding of how the rib-cage dimensions change. There are however collisions between the rib tips that are unfortunately unavoidable without a collision detection algorithm in place.

Initially a basic algorithm that simply outputted a single point at the end of each rib was used with fairly poor results. This has been improved by using a matching algorithm where all the rib tip vertices are utilised. The frame needs improvement to output the whole rib tip and ensure that each point on the cartilage is updated to attach to the same point every frame.

Again the collisions with the diaphragm could be minimized by rotating the diaphragm anchor points with the costal cartilages

4.4. Lungs

The values of the mass spring parameter for the lungs are:

Variable	Spring Coefficient	Damping Constant	Mass
Value	0.01	5	8

Table 4 Spring coefficient, damping constant and mass of lungs

The goal of representing the lungs as flexible, fragile structures has been achieved and I believe this adds to understanding how the lungs contract and expand.

The movement of the lungs again seems rather realistic at low respiration rates with low rib rotation angles. Above a rate of 20-25 breaths/ minute there are marked collisions with the ribs over the mass spring regions due to the regions further away from the determined anchor points having to wait for updating in the next frame. At higher respiration rates the updating issue again affects the visualisation with more frequent collisions between the ribs and the lung margins.

The base as has been noted earlier collides with the diaphragm and this can only be totally removed by collision detection.

4.5. Controls

The updated controls have been implemented with new sliders for manipulating the transparency of the bone and cartilaginous structures and another for the transparency of the diaphragm.

5. Discussion

It should not be overlooked that this has been a rather complicated project drawing on a lot of areas, including anatomy, physiology, physical modelling and interactive 3D graphics. It has thus been rather challenging for a novice with little programming experience. The main

question to ask is whether I have achieved what I set out to at the beginning- a step-wise improvement in the original simulation using a more suitable physical model.

5.1. H3D Node Implementation

The creation of organ specific deformable shapes is necessary because they are very different with widely different behaviours that could not easily be modelled using a "one-size fits all" approach. By having them inherit from the ***DeformableShapeNew*** this has obviously reduced the need for changing code in multiple places. This will allow further organs to be implemented in a similar manner.

The ***RibRotTransform*** is a simpler version of the ***RotTransform*** node as it performs simpler operations on routed "touch" points, reducing the number of operations and to be honest resulting in a more realistic deformation.

One problem that has been encountered is that X3D and the Meshlab software does not allow delimitation of the actual coordinate indicies of the vertices to be moved or anchored in an easy manner. This has required a point matching algorithm to find the position of a vertex in a list and then save this for future use. This is rather crude and could have been open to error if several vertices initially started at the position.

A better software program would have been one in which all the indicies could have been delimited before therefore saving computational time and memory usage as there would have been no need to create multiple reference arrays for all the coordinates.

5.2. Mass Spring Model

The mass spring model has been correctly implemented. Relaxation did not seem to add much so has been omitted.

As mentioned earlier, a dynamic rather than a quasi-static mass spring model has been implemented. This has implications for the final model, in terms of oscillation and stability but as can be seen from the model these have not proved too troublesome and certainly not affected the workings of the model significantly. Also it is not a massive task to implement the quasi-static mass spring on the current system, although its behaviours will obviously be different

5.3. Applicability of the dynamic mass spring model to the various organs:

5.3.1. Lungs:

The mass spring model looks pretty realistic once the pre-determined points have been inserted and the mass-springs are there to model the lung segments between the fixed rotating regions.

The mass spring model however has certainly added to the realism of the lung model. It has also taken away the ugly anchor points which could only have worked if there were far more of them in the chainmail model.

The fixed rotating lung segments which have no physical deformation applied to them are obviously not ideal. An improvement could be made by adding a small deformation to the rotated segments but this would introduce a far greater level of complexity.

Another manner in which this could be improved is to use more but smaller segments to rotate with possibly all of the first 6 ribs, resulting in smaller regions of contiguous springs which would have less issue updating than the larger mass spring regions at present.

Collision detection with the ribs would and the diaphragm would also prevent many of the problems noted earlier when the lung bases overlap the diaphragm and the concave lung surfaces collide with the ribs, especially at the higher respiratory rates.

5.3.2. Costal cartilages:

Again the mass spring model looks quite realistic. The costal cartilages have been appropriately modelled as flexible, elastic structures and so the mass spring model seems a perfect fit.

Gaps do unfortunately open up between the rib tips and the moving ends of the costal cartilages especially at the lower ribs but these are pretty much unavoidable due to the way X3D updates with the cartilages receiving the previous frame's value.

Another issue with the costal cartilages is the waveform motion at the ends articulating with the ribs. This is because the articulating regions had to be selected away from the ends of the cartilage. Changing this to the ends themselves results in increasing gaps appearing which are less visually satisfactory than the waveform motion.

There are again collisions between the cartilages and the rib tips and between the right composite cartilage and the diaphragm.

5.3.3. Diaphragm:

Possibly the least satisfactory organ modelled by the mass springs. It looks a little cloth like at the edges with the mass springs updating. The best visualised values for spring coefficient, damping constant and mass has been selected but this could probably be optimised further.

The dynamic mass spring model seems to oscillate more in the diaphragm than anywhere else due to the energy involved with large areas of springs. This could be improved by simplifying the number of springs in the system, possibly having a spring only every 2^{nd} or 3^{rd} vertex.

Another issue is the collisions with the lung bases, the costal cartilages and the liver, again collision detection would assist.

6. Future Work

6.1. Testing and Medical Student Appraisal

As mentioned earlier in section 2.1 any simulation requires validation and this is the next logical step for the implemented respiration simulation model. It is a shame that time frame did not allow this with the early delays and problems getting the dynamic mass spring model implemented as it could have provided valuable information about improvements and which of the following areas should be tackled first.

6.2. Organ Improvements

Initial work could be to improve the motion of the central tendon region of the diaphragm with a flattening and rotation as has been explained earlier. Also a rotation of the anterior anchor points with the sternum would reduce collisions with the costal cartilages and the ribs. The liver motion would be improved if this was to be implemented by applying this same motion and flattening to its upper surface.

As mentioned above, the lungs could be altered with more moving segments with smaller regions of contiguous mass springs, possibly resulting in fewer collisions.

6.3. Collision Detection

This seems necessary to implement both on my simulation and on Mathieu's previous model. This would however require more computing power as complex bounding boxes would have to be created with an implementation of how any collisions should be passed onto the colliding objects. This would be a significant step up and would likely require optimization as H3D and the simulation already seem to be pushing the limits of what is feasible on the laptop I have at present.

6.4. Quasi-Static Mass-Spring Model Implementation

It would obviously be wise to implement a quasi-static mass spring model and see if this were to introduce any improvement in the motion of the diaphragm, lungs and costal cartilages. The pre-determined points would still have to be kept as this would look for the configuration where the internal and external forces cancel out, so movement would only be in terms of what has been discussed above.

7. Conclusions

The project has had to have its aims altered from the initial specification of modelling liver motion due to respiration and investigating the various boundary to producing an anatomically and physiologically realistic simulation of the key organs during the respiratory cycle.

It has obviously drawn on a lot of very diverse areas and has been a difficult project to implement, not made easier by the initial delay due to software and hardware problems and the time it took to familiarise myself with X3D and H3D.

I believe it has been a useful project which builds on and extends the work done at the Department of Biosugery and Surgical Technology at Imperial College by implementing a dynamic mass spring system under H3D that is suitable for representing several of the key organ involved once pre-determined motion has been applied to regions at both ends of the springs.

The project has shown that it is possible to produce an increasingly realistic simulation under H3D that will be useful for medical student teaching and may in future be developed further with increasing benefits.

8. References

1. M.N. Acharya. **"Modelling of Diaphragm Motion for Simulation of Liver Access"**, Surgery and Anaesthesia BSc Project, Imperial College London, May 2008, Pages 13-14, 33-35.

2. Bourne, Wesley. **"The Use of Tensegrity to Simulate Diaphragm Motion Through Muscle and Rib Kinematics."** MSc. Thesis, Department of Computing, Imperial College London, 2008.

3. Jacob, Mathieu. **"Computer-based multi-sensorial environment for anatomy teaching: Dynamic modelling of the rib cage anatomy during respiration."** MSc. Thesis, Department of Computing, Imperial College London, Sept 2008.

4. Image courtesy of knowledgerush.com

5. Philippe Cluzel, Thomas Similowski, Carl Chartrand-Lefebvre, Marc Zelter, Jean-Philippe Derenne, and Philippe A. Grenier. **"Diaphragm and Chest Wall: Assessment of the Inspiratory Pump with MR Imaging-Preliminary Observations."** Radiology 2000; 215: 574.

6. **Reference for diagram of rib cage and muscles**

7. ***"Gray's Anatomy: The Anatomical Basis of Clinical Practice."*** 40th edition (2008), Churchill-Livingstone. Pg 350-352

8. Keith L. Moore**,** Anne M.R. Agur**. "Essential Clinical Anatomy: Dynamic Human Anatomy."** 4th Edition (2004), Lippcott Williams and Wilkins.

9. Boriek, Aladin M., Joseph R. Rodarte**. "Inferences on passive diaphragm mechanics from gross anatomy."** J. A&. Physiol. 1994. 77(5): 2065-2070.

10. Whitelaw WA. **"Shape and size of the human diaphragm in vivo."** J Appl Physiol (1987) 62(1): 180-186

11. **Image courtesy of** daviddarling.info

12. **Image courtesy of** ironman.com

13. Wilson, Theodore A., Alexandre Legrand, Pierre-Alain Gevenois, and Andrãe De Troyer. **"Respiratory effects of the external and internal intercostal muscles in humans."** Journal of Physiology 530, no. 2 (2001): 319-330.

14. Jean Oliver, Alison Middleditch. **"Functional Anatomy of the Spine."** 2nd Edition (2005), pg 200, Elservier Butterworth Publications.

15. **Image courtesy of** ebscosafe.smartimagebase.com

16. **Image courtesy of** forcleverpeople.com

17. Mosegaard, Joseph. **"Real-time Cardiac Surgery Simulation",** MSc Disertation, Aarhaus University, June 2003.

18. D.E. Ingber, **"Opposing views on tensegrity as a structural framework for understanding cell mechanics",** *Journal of Applied Physiology 89: 1663-1678, 2000.*

19. Gibson, Sarah F. F. **"3D ChainMail: a Fast Algorithm for Deforming Volumetric Objects."** *Symposium on interactive 3D Graphics.* 1997. 149 - 154.

20. Luciana, Porcher, Nedel, Daniel, Thalmann. **"Real Time Muscle Deformations Using Mass-Spring Systems."** Computer Graphics International Proceedings. 1998 ;156-167

21. Li, Ying, and Ken Brodlie. "Soft Object Modelling with Generalised ChainMail - Extending the boundaries of Web-based Graphics." *Computer Graphics* 22, no. 4 (2003): 717-727.

22. H3DAPI Manual For Version 2.0. SenseGraphics AB - October 31, 2008