Integrate systems into multi-modal display

Name of responsible	Aurélien Pocheville, Abderrahmane Kheddar
Date	March 2005
Del. Identifier	D6.8
Work Package	WP6: Haptic Illusion Based Systems
Partner(s)	LSC
Work Package Leader	LSC
Confidentiality Level	Public

Abstract:

We present here the description of the integration of different modalities into a multimodal framework, I-Touch.

IST-2001-38040

TOUCH-HapSys
Towards a Touching Presence:
High-Definition Haptic Systems
http://www.touch-hapsys.org

Table of Contents

1. Introduction

In the current virtual reality field, each of current display (audition, visual and touch) are often indepently studied. But a more interresting approach consist to blend them to create a multi-modal virtual reality. The fact is putting more than one modality together does not result in the sum of the modalities. The experience can be enriched by such cooperation, but user can also feel destructive interaction between the modalities. This creates new challenges for the virtual reality creator. For the most cases, the modalities are not to be handled in the same way, thus creating diffuclties for integration. In the following, we focus on such challenges.

In order to be able prototype multimodal integration schemes and models, the LSC developed a generic framework called I-Touch [POC04a][POC04b] that serves also the rigid bodies prototyping demonstrator of the deliverable D7.5 and D7.6. This part will describe the components of the I-Touch framework. The fact the framework is multimodal is of great importance, in order to be able to evaluate the importance of each feedback. We tried to give each of the modalities the same 'importance', from an integration point a view, precisely in order to be able, after that, to modulate the user experience.

The I-Touch framework

1.1 *Design of the I-Touch framework*

The I-Touch framework is designed to be modular from its core to its interfaces. Although it is still in the development process, it already allows plug-ins (static linking in program) of different behaviors models and different collision detection algorithms. The framework architecture is given in the Figure 1.

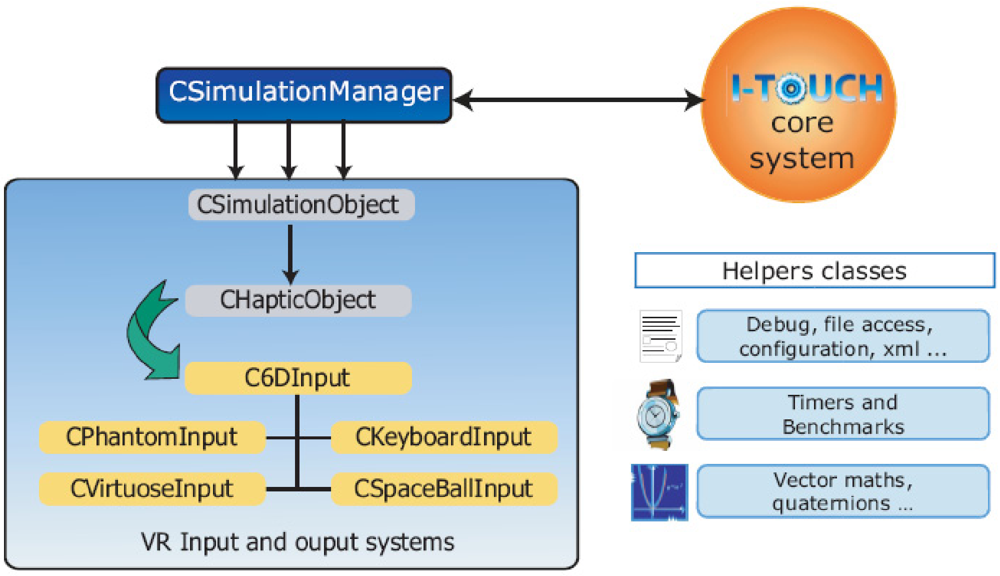


Figure 1: I-Touch framework architecture.

As it can be noticed, haptic interfacing, among other rendering capabilities, takes an important place. This particular focus meets the Touch-HapSys objectives in better understand haptics and its relation with the other modalities. However, this modality has the same 'rights' and 'duties' as other modalities. Precisely, the haptic modality can be as easily put in / removed from the simulation as any other.

From a pure technical point-of-view, the framework is divided in three main modules; each of them is further subdivided in as many sub-modules as needed. The I-Touch framework is completely object-oriented. This allows easy part replacement and improvements. It is implemented in a pure standard C++, and apart from the driver libraries, does not use platform dependent code. It can be easily ported to Linux or MacOsX, however it is designed to be best suited to MS Windows OS. Of course, this object-orientation makes also easy the sub classing of some part of the framework, in order to create completely new applications. The fact we can devise new application "on demand" is of considerable importance. Namely, we are also envisaging its use for dedicated psychophysical investigations.

1.2 The core system

The core system is responsible for handling the operating system (the platform specific code is inserted here), the configuration, and the basic functionalities of a physically-based simulation. It provides a basic scene graph for managing the various objects that composes the virtual scene. This core system can accept many simulation algorithms along with different input methods. Classical mathematical methods [1] [2] and structures are also provided, for the easy prototyping/evaluation of new/existing algorithms. The core system also provides a very flexible configuration in XML, which parametrizes any aspect of the simulation, thus allowing the creation of test-beds rather easily.

In addition to configuration tools, a file format for holding together "geometrical" object properties has been devised. This format is open and flexible; moreover, additional data can be included and be ignored if not necessary to the simulation, even if it is an unknown data. An importer and exporter have been written for 3DSMax, along with C++ and C# libraries for loading efficiently theses files. This exporter can, for now, export geometry, normal, bi-normal and texture information, however, it does not save the relationship between texture information and texture map (this link has to be recreated in configuration files, however, the result is of course exactly the same). This data format, exporter, importer and C++ libraries have been conjointly developed by A. Pocheville and M. Brunel. Details concerning the implementation can be found in [3].

1.3 The input and output system

While the input system needs to be flexible and needs to manage many different inputs, the output system should ensure high fidelity rendering along with adequate refresh rates according to the addressed modality/output. Of course, the output system is tightly related to multi-modal integration. One of the most important facts here is that the output system should be very flexible and allow almost any information as its input, in order to be able to prototype new integration schemes. However, there are limits to flexibility, in the end, each modality still requires special handling.

1.4 The simulation system

The simulation is composed of the simulation manager, and a set of simulation virtual objects.

The simulation system uses the core system for standard interaction with the computer and the user, and input/output system for multi-modal interaction. Collision detection algorithms are part of the simulation system. The simulation system, like any other part of the framework, can be replaced or modified rather easily. The simulation system can be decomposed in two subsystems: the collision detection subsystem and the behavior model (the physic step). The fact is that collision detection step can be also benchmarked to certain extend: for example, a penalty or a constraint based physic resolution has known inputs, we can then select detection collision algorithms that provides such information. The following schema illustrates this interaction:

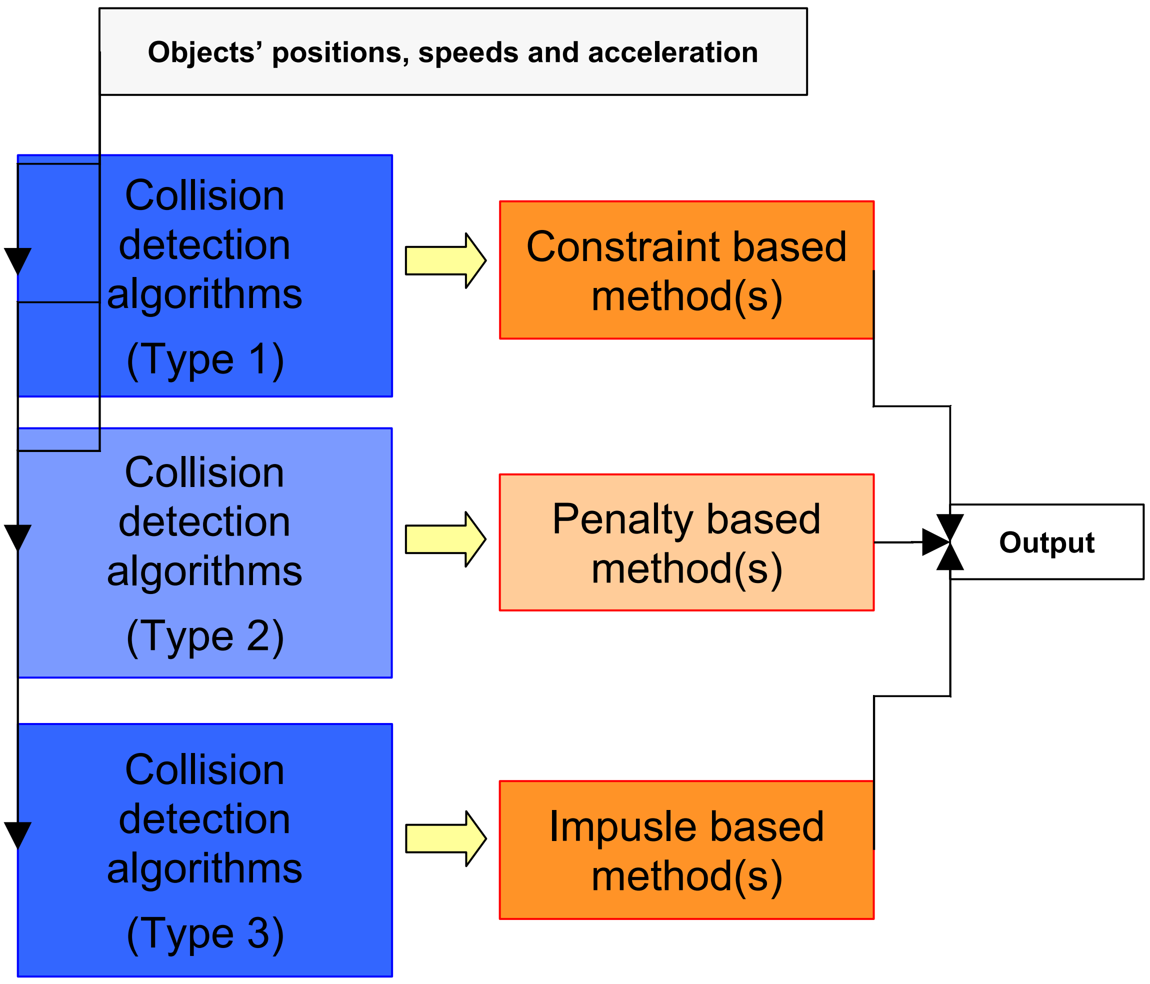

Objects' positions, speeds and acceleration
Collision detection algorithms (Type 1)
Constraint based method(s)
Collision detection algorithms (Type 2)
Penalty based method(s)
Output
Collision detection algorithms (Type 3)
Impulse based method(s)

Figure 2: I-Touch interaction between collision detection and response

2 Multimodal integration

One of the most challenging issues of I-Touch is multimodal integration. Visual, audition and haptic senses have different refresh rates: from as low as 30Hz for visual interaction, up to as high as 10kHz for the 3D sound one. Integrating each of these modalities is not a trivial task. Others tentatives tried through parallelization of the computation on different computers. Here, we decided to push the limits on focusing in using one computer, but this unveils some problems as exposed later.

1.5 Simulation engine flexibility

The fact that the simulation engine is completely flexible and modular allows the integration of different behaviors models with the same multimodal rendering. This is done through abstraction of the output methods. For example, haptic rendering can be easily derived to create new rendering. These rendering can then be easily switched. Moreover, since the simulation engine use the simulation objects as placeholders, alsmost any information can be provided to the output. For example, simulation objects are holding the whole normal map used in visual bump mapping and haptic bump mapping. This information can then be accessed at will.

However, the simulation engine has to provide some specific information to the output routines. For example, for the real-time 3D sound rendering, contact information (and changes in contact through time) are required. The immediate benefit of the ability to switch between renderings is that we can benchmark how well does a simulation engine behaves with multimodal rendering. For example, bounce models have difficulties in rendering contact information with sound, while they provide excellent rendering of bounce sounds. It can be observed that

1.6 Collision detection integration

The I-Touch framework has a certain amount of requirement from the collision detection module. It should be noted that due to the architecture of the framework, inter penetration never occurs. For I-Touch, objects can only be touching. Objects are considered to be touching when they are less than a certain predefined distance away from each other. We will refer to this distance as the tolerance of the collision system. When a contact occurs, the contact area should be represented by a group of point pairs, each of which is on the surface of one object. The points on each object should represent the contact region of this object. No point pairs should be duplicated in the list since each pair is later used to calculate the reaction force.

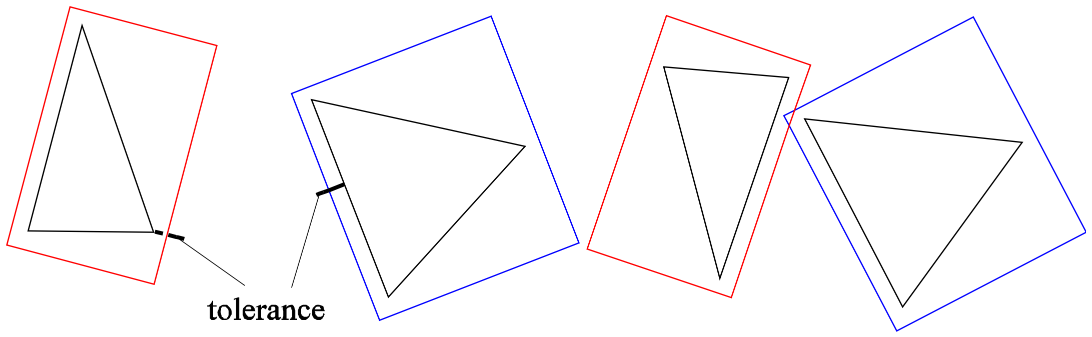
The collision detection module should be robust to handle polygon soups where duplicated vertices can be found and thus neighborhood information between triangles might be erroneous. The collision results should be calculated fast enough to leave the required time for the dynamical model to calculate the reaction forces at interactive rates.

Our proposed system provides the required information. It is composed of three layers:

The first is an acceleration layer that determines the triangle pairs having a high probability of colliding, thus needing further processing. The second layer performs our topology determination test on each of the pairs obtained from the first layer. This test obtains for each pair, the list of point pairs that defines the contact between these triangles. The last layer is where all the contact point pairs from the second layer are gathered, while eliminating duplicates. In the following, we will explain each of the three layers of our system.

1.6.1 Layer1

The first layer is an acceleration phase, based on object oriented bounding box hierarchies. Each hierarchy is constructed top down, starting by the first box at the object level, and proceeding by dividing the triangles into two groups, and constructing a bounding box for each group. The leaves of the trees of the hierarchies contain each, one triangle. This construction produces 2n boxes for each object, where n is the number of triangles for that object. In order to account for the tolerance, we have to enlarge each box of the tree by half the value of the tolerance in the direction of the three axes of the box. This action is crucial since we consider a pair of triangles to be in contact when they are within the tolerance from each other. An example is shown is the following figure.



When traversing the tree, if the current nodes are leaves we add the pair of triangles that they contain to the list of triangle pairs that we will process in layer 2. If the nodes are not both leaves, then we process their child boxes together. The output of this layer is the list of triangle pair that need further processing.

1.6.2 Layer2

The input of the second layer is the output of the first layer. We need to determine the contact type for each triangle pair in the list, which can be a point, edge or surface contact and for each case, find the points that delimit the contact zone for each triangle. We define the "margin" as the variation in the distance within which we consider the two vertices as parallel to a plan. For example, having v_{11}, v_{12} and v_{13} as vertices of the first triangle and p2 as a plan of the second triangle with the tolerance fixed to 10 and the margin fixed to 2, then if

$d(v_{11}, p_2) = 11$

$d(v_{12}, p_2) = 5$

$d(v_{13}, p_2) = 6$

We consider that we are potentially in a case of edge-plan contact because we consider that v_{12} and v_{13} are both lying in p_2 within the tolerance and the margin.

Our algorithm starts by calculating the distances from each vertex to the plan of the second. The distance is not a signed number, and do not depend on the direction of the normal. This is done to account for the case where the vertices of the triangles are given in the reverse order.

This first part of the algorithm has the following pseudo code

```
*$t_1$ and $t_2$ are the first and second triangle respectively*
```

```
v_11, v_12, v_13,, v_21, v_22, v_23 are the vertices of t1 and t2 respectively
p_1 and p_2 are the plans of t_1 and t_2 respectively
For each vertex of t_1
    Calculate the distance for this vertex to p_2
End for
For each vertex of t_2
    Calculate the distance for this vertex to p_1
End for
If at lease one vertex of t_1 is in t_2 or at least one vertex of t_2 is in t_1
    Determine the type of potential contact of t_1 using the calculated distances
    Determine the type of potential contact of t_2 using the calculated distances
    Call PlanContact(HighestContactOrder(t_1,t_2))
Else
    Call EdgeEdge()
End if
```

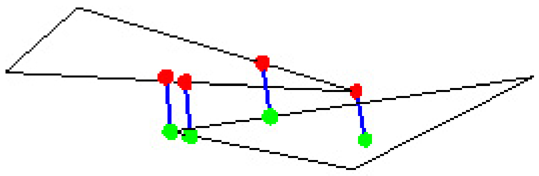
In order to determine the type of potential plan contact, we start by identifying the vertex with the smallest distance, and then verify if the other two vertices of his triangles are within the margin. If both are, then we are in a potential plan-plan contact. If only one is, it is a potential edge-plan contact for this triangle. If none is, then it is a potential vertex-plan contact. All vertices within the margin and the closest one are called active.

1.6.3 PlanContact()

A plan contact can be a vertex-plan, edge-plan, or a plan-plan. In order to determine all the point pairs that define the contact, this function starts by checking if the active vertices of the two triangles are inside all the plans defined by the edges and the normal of the second triangle. If a vertex is, then we are sure it is a contact vertex since we already verified that it is within the tolerance, and we can add it and his projection on the second triangle are added to the result vertex pair list.

Then we check the edges of the two triangles, having both end points active, against each other. If their closest points are less then the tolerance away from each other then we add the closest points of the two triangles to the result pair list. Some conditions have to be verified in order to assure that no duplicate pair are added to the list, but will be omitted for simplicity.

```
For each active vertex v of t1
    If v inside the triangular prism of t2
        Add v and his projection to t2 to the result list
    End if
End for
For each active vertex v of t2
    If v inside the triangular prism of t1
        Add v and his projection to t1 to the result list
    End if
End for
For every edge e1 of t1 having active end points
    For every edge e2 of t2 having active end points
        If d(e1, e2)<tolerance
            Add the closest points on e1 and e2 to the result list
        End if
    End for
End for
```

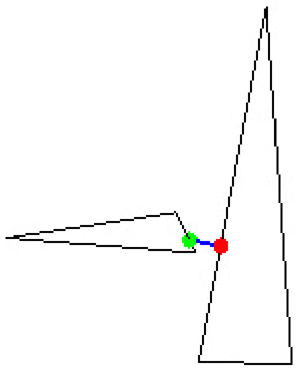


1.6.4 EdgeEdge()

If no vertex of the two triangles is in the plan of the other, then we must verify that we don't have edge-edge contacts before concluding with a no contact situation.

This test checks all pairs of edges again each other, and adds the closest point on two edges to the result list if the distance between their corresponding edges is less than the tolerance. We also have to check certain conditions to assure that no duplicates are added to the list.

The following is an edge-edge case



```
For every edge e1 of t
    For every edge e2 of t2
        If d(e1, e2)<tolerance
            Add the closest points on e1 and e2 to the result list
        End if
    End for
End for
```

An important remark is that the case of an edge penetrating the face of the second triangle, but having a distance to all three edges of the second triangle highest then the tolerance, is not detected.

This should not be a problem for I-Touch since this should not be allowed to happen.

When the end point of this edge reaches the plan of the triangle, the engine of I-Touch, prevents penetration, and thus the above mentioned situation never occurs.

1.6.5 Layer3

The last layer of the system deletes redundant pairs from the result list. This can be done by the brute force algorithm of order $O(n^2)$, where n is the number of result pairs. Knowing that the number of results pair is usually not very high, such complexity can be acceptable in most cases. However, we designed an optimized duplicate pair remover, which checks only the results of the neighboring triangles for duplicate pair. Such a strategy is possible, since duplicates can only exist among adjacent triangles.

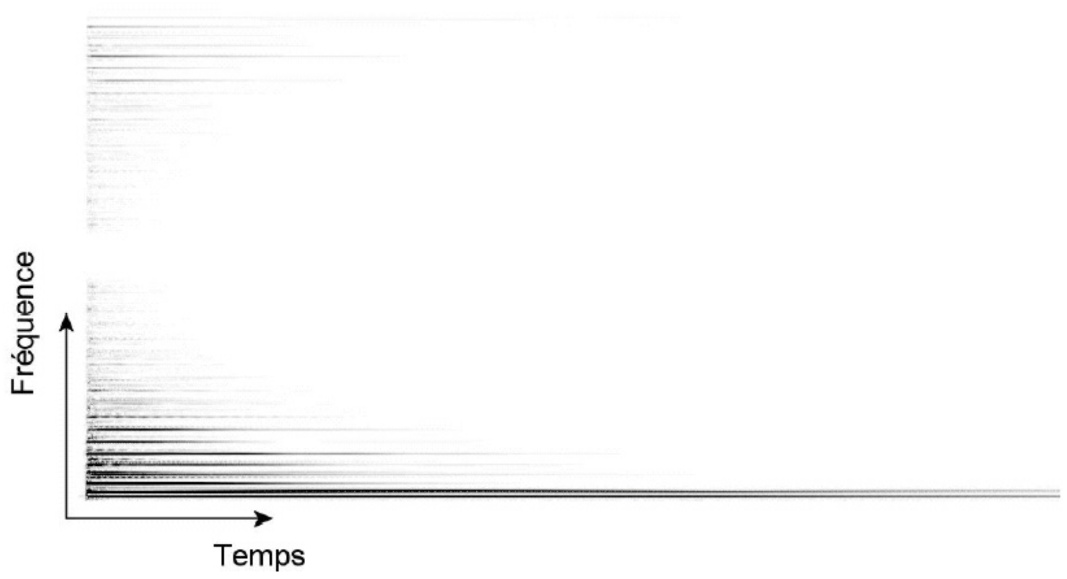
Such an optimized version should only be used when the models are well constructed, and neighborhood information for the triangles is correct. If it is not the case, it is better to use the brute force method.

The result of this layer is a list of point pairs, defining the contact between the two objects, and having no duplicates.

1.7 Sound integration

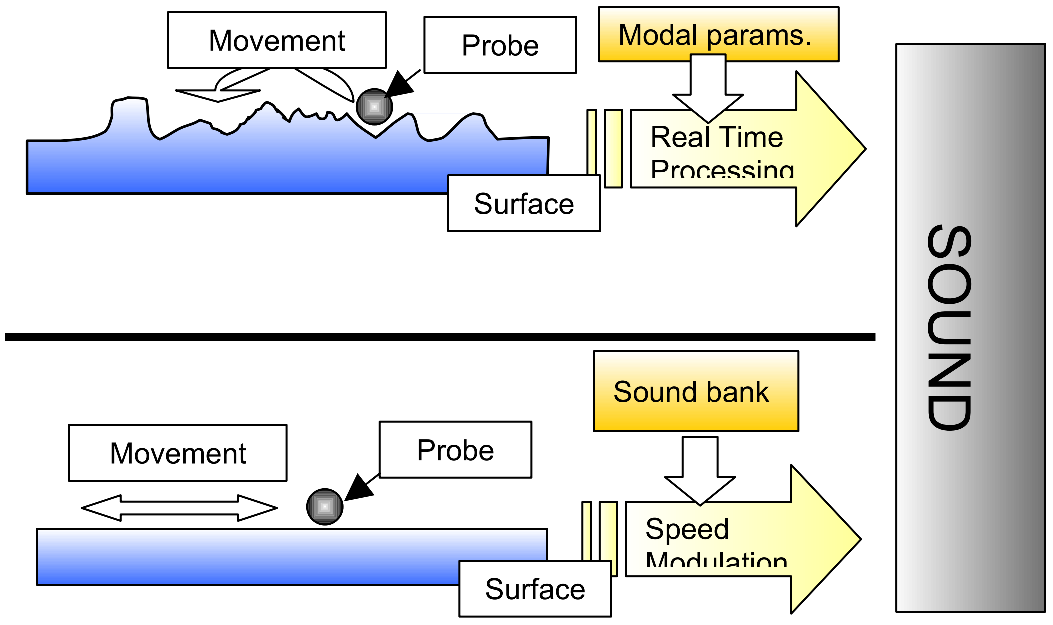
3D positional audio enhances the immersion of the operator in the simulation. We have two methods for rendering 3D sound: real-time rendering, and semi-real-time rendering. The real-time rendering uses information directly provided by the simulation, such as changes in the friction map to produce sound. It also uses object properties such as resonance frequencies to computes contact sounds [26] . While this is the correct method for producing friction and bounce sound, it suffers from several drawbacks. First of all, it is very computational-time consuming, and, in a system composed by only one processor, it can become the bottleneck of the simulation (and take the place of the collision detection!). Maybe relocating the sound computations could solve this problem. The other fact is that the sounds generated are, for now, less “realistic” than the ones produced by the second approach.

The following is an example of analysis of modal parameters of a material.



The semi-real-time sound rendering approach uses off-line recorded sounds of different materials in contact. These different sounds are stored in a database according to some material properties. They are used by the simulation as they are and the only amplitude and/or frequency modulation (pitch, volume...) are processed. This method can be seen as same as vertex transform followed by a texture pass in visual rendering.

The following figure stress out the difference between the two methods:



1.8 Visual integration

Relatively to the sound and the haptic rendering, the visual one is the easiest. We can use the same geometry as the one used for physics calculations, or a higher level, smoother one for better rendering. Objects are linked to rendering information, such as geometry, material and alpha information, and pixel and vertex shaders. This allows almost any rendering of the objects, from standard Gouraud-shaded plastic look, to advanced Phong-shaded semi-reflecting materials with bump mapping. Dynamic lighting is also supported. The visual rendering is completely configuration controlled, so there is a great flexibility in the rendering process. The fact that the visual rendering is well understood and has numerous techniques is very interesting here, because we can take inspiration from these techniques in others renderings.

1.9 Haptic integration

Our approach differs from the previous ones. Indeed we are conceptually considering that the haptic devices (interfaces) interact with the simulation and not the reverse way i.e. the haptic device does not drive the simulation. The point here is that haptic device can be removed at will. Even more, more than one haptic device should not pose any problem to this type of simulation, since haptic rendering is totally decorelated from simulation loop. Obviously, haptic devices can induce a change in the course of the simulation but they cannot compromise its integrity. To be clearer, the simulation does not take as granted what is needed from the input device and, in extreme cases, these particular inputs are ignored. In fact, this enhances considerably the stability of the interaction. For example, when the operator actions are toward violating a given non-penetration constraints, they are not considered integrally (as is the case for classical computer haptics API's). This is better for engine stbility, and allows for many new algorithms to be used. However, haptic feedback has to be consired from another point of view. The basic principle behind this is exmplained in the following figure:

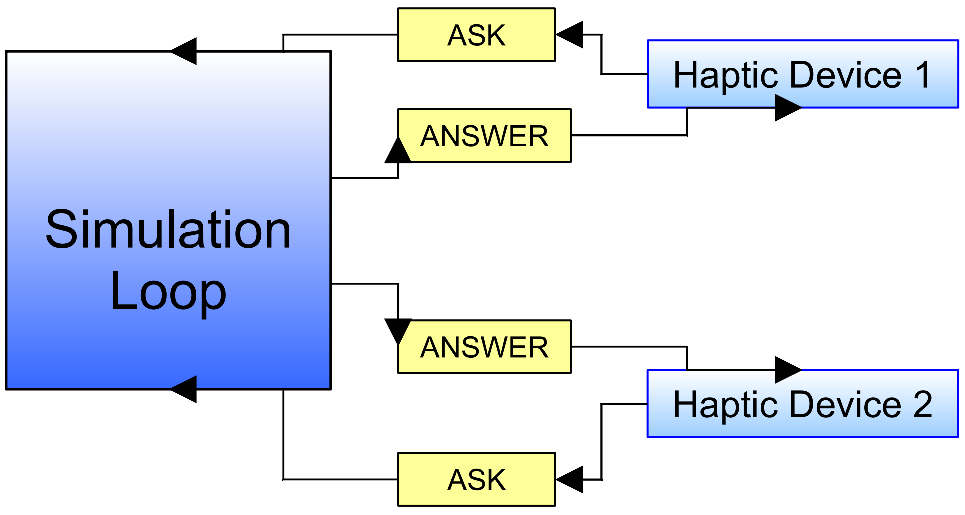
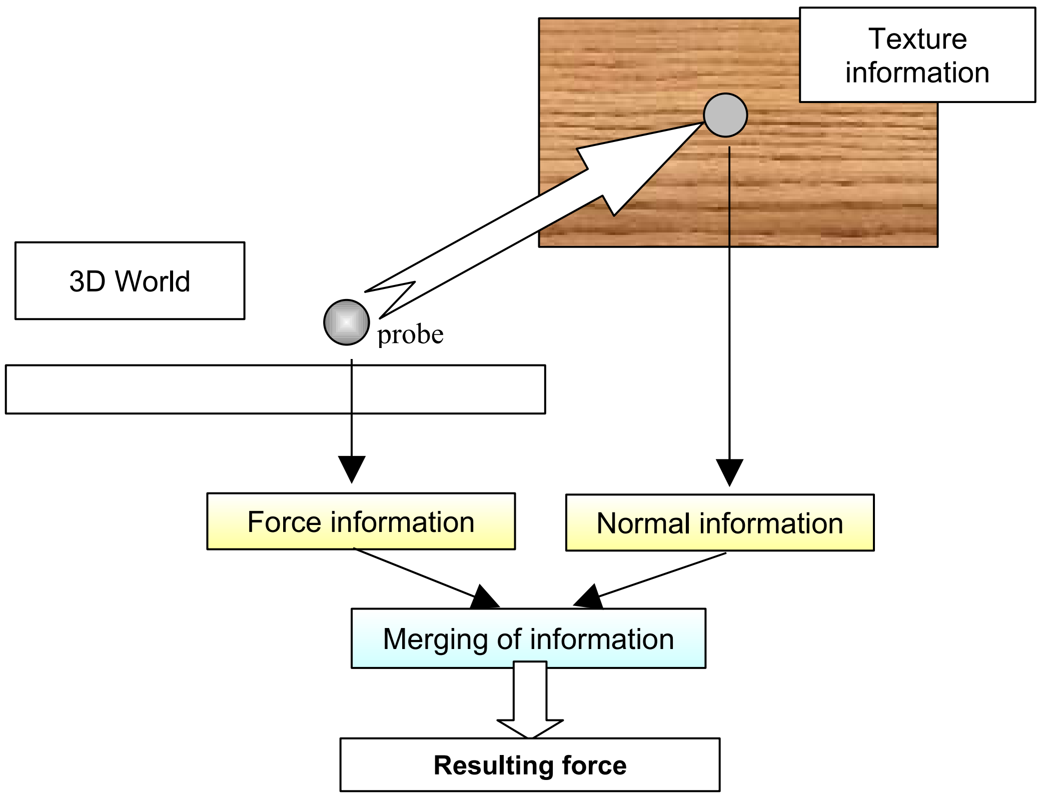


Figure : Principle of force handling.

The fact that haptic integration is not considered as a special rendering allows new synergies between rendering to be investigated. One example of this is the recently implemented haptic bump. As for visual bump mapping, we can simulate rough haptic surfaces through haptic bumps. We tried two different approaches: height based forces, and normal based forces. The basic principle is the same: the force computed by the simulation engine is slightly modulated by a term, which depends either on the height or the normal. In our actual implementation, haptic bump does only work with one contact point, but we are working on extension to multiple points.

The following figure explain the principle behind the haptic bump:



As far as "bump sensation" is concerned, the normal based force give superior results. The bump map used for haptic bump is exactly the same as the one used in the visual bump, thus the two modalities match perfectly and the rendering is coherent.

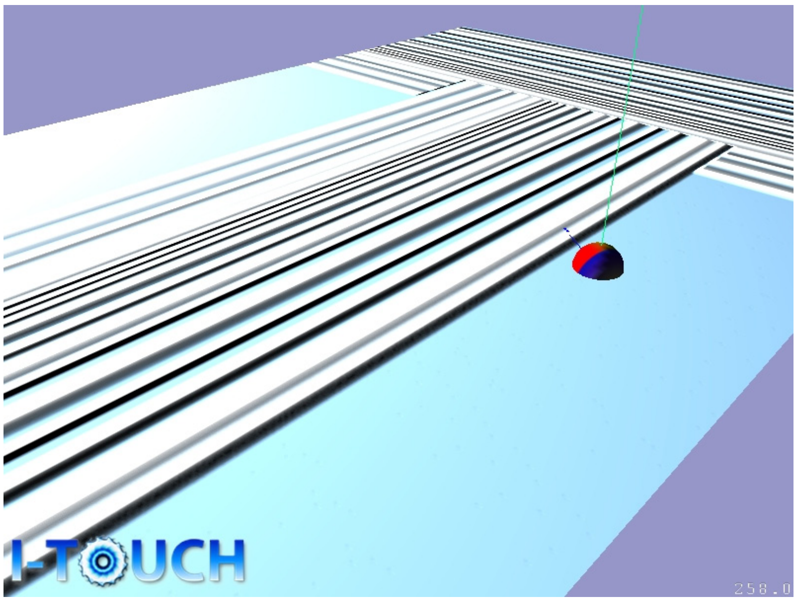
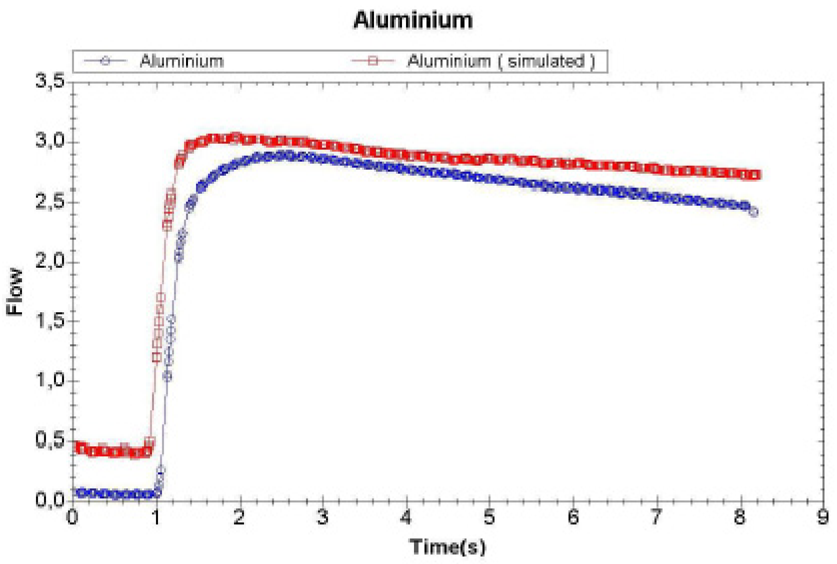


Figure : Example of haptic bump (combined with visual bump). The surface is in fact a plane.

1.10 Thermal integration

While there have been advances in thermal integration, there remains difficulty in some areas:

- First of all, we have to be able to reproduce any material from its thermal properties. For now, we can only approximate the law of the thermal interaction between a human finger and a material through a Peltier pump, but many progress are still to be made, in order to be able to deliver almost exact replication of thermal sensations in the real worl.
- Secondly, while the integration of the thermal sense is not very difficult in the simulation (mainly from its poor refresh rate, the human has not a high refresh rate thermal sense), it poses great difficulty from a practical point of view, where we should combine thermal with kinaesthetic feedback. An efficient device is yet to be devised.



1.11 Putting them together

Many synchronizing algorithms do exist, but for now, we did not experimented them, because they are time consuming when added to the simulation engine. Computation

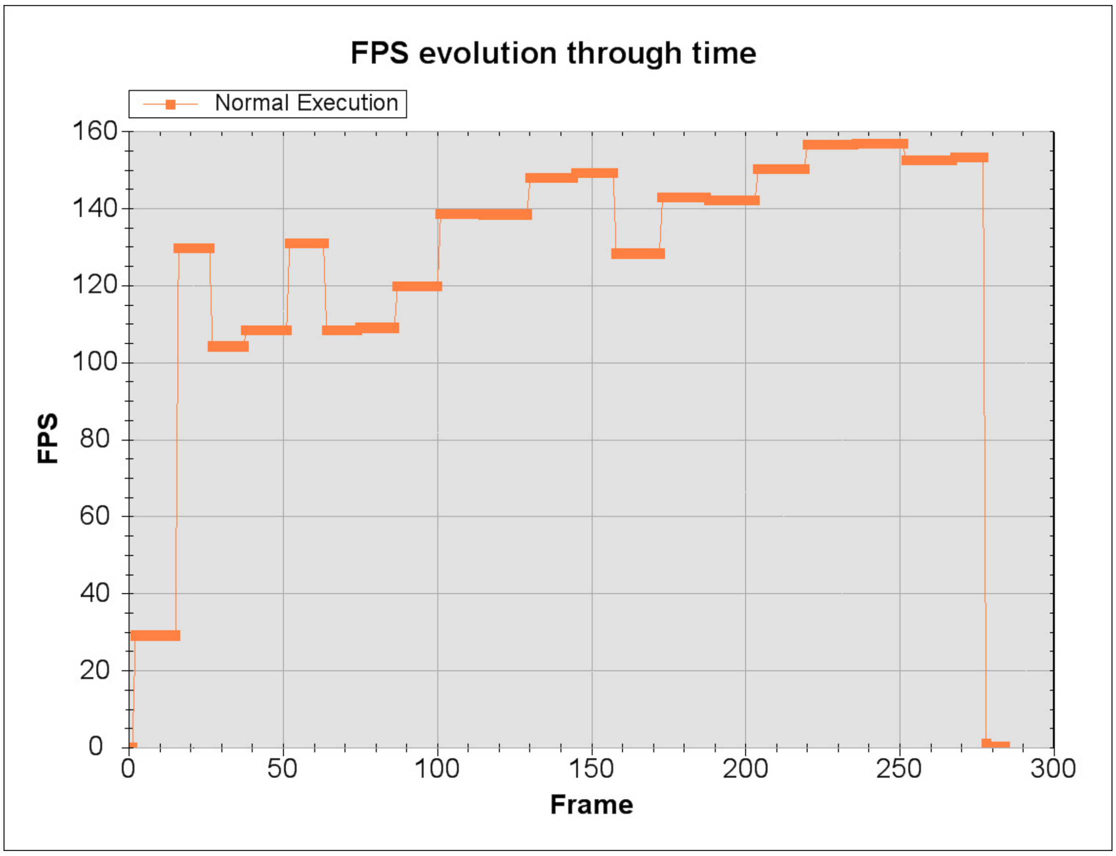
of different contacts, and then contact forces, takes most of the CPU time. As theses computation are the main bottleneck of the simulation, the rendering is sufficently fast. Moreover, the fact of multithreading does not harm this process, since the refresh rates are too high for an human to notice little differences.

The data is given to render in this order (immediately one after the other): haptics, sound and vision. In the future it would interesting to have scalable algorithms that would be totally disconnected from the physical simulation, we will then need synchronization algorithms. This would prevent that the new bottleneck (let’s say, sound rendering) to harm the other displays. However, it will also mean that one of the modalities will have degraded performance, we will need to investigate how far we go in degradation in one of the modalities without ruining interaction.

1.12 Evaluation tools

The testing of research projects is made easy with I-Touch, however such a testing requires to analyze data from the simulation. In I-Touch, every simulation variable can be “tagged” from recording; this allows after-run simulation analysis through tools. For example, FPS data, or time taken to compute a frame (much more speaking than FPS in regard to performance) evolution can be viewed easily. The ‘tagging’ is done by providing the pointer to the variable. Then, each frame (or at a given time), the variable is recorded. At the end of the simulation, a file that can be viewed in a special viewer is written. The actual viewer is written in .Net and is provided with the I-Touch framework.

Also, the debugging facilities and text functions in I-Touch make it easy to dump data. However, we want to go further, and new real-time tools are in development. Such tools will render in real time evolution of variables, in numerous manners (time graphs, bars, standard text, etc.). Specific analysis tools to simulation should also be included, such as automatic reporting of number of contacts, physics calculation time, time spent in rendering or in other tasks – this step is almost done, but we have to link it with the visual rendering. This will create a complete and easy evaluation tool, in order to accelerate further the development of test cases. This step is very important in the relationship that this project can have with psycho-physical and others experiments.



Conclusion and future work

We have shown that we have made a prototype of software framework that allows the creation of many test cases. They all have in common experiments with operation, in a multimodal way. The haptic rendering is not considered with special concerns, thus facilitating the evaluation of haptic impact in a multimodal context. Integration of all these modailities in one framework in not an easy task. We have chosen the modularity path, which allows testing of and imaging new scenarios. However, there is an attempt to provide each of the modularity a coherent information, in order to allow a real symbiose between the different senses.

We have succeeded in creating sample applications that use the flexibility of the I-Touch framework. However, integration with real psycho-physical tests is yet to be done. Further work is oriented toward refining I-Touch through its multimodal component to serve also as a psychophysics evaluation tool. Progressively our aim is to evolve it to a complete piece of software that can serve haptic research. Future improvements will include more intimate collaboration of the different modalities, and new ways of defining what a modility is from a software point of view.

References

[1] L. Dorst and S. Mann, “Geometric algebra: A computational framework for geometrical aplications,” IEEE Computer Graphics and Applications, 2002.

[2] J. C. Hart, G. K. Francis, and L. H. Kauffman, “Visualizing quaternion rotation,” ACM Transactions on Graphics, vol. 13, no. 3, pp. 256–276, 1994.

[3] M. Brunel, “Rendu multimodal haute fidélité, rapport de fin d’étude,” 2004.

[4] A. Drif, J. Cit´erin, and A. Kheddar, “A multilevel haptic display design,” in IEEE/RSJ International Conference on Intelligent Robotic Systems (IROS), 2004.

[5] Y. Chenu, “Algorithmes de détection du contact, topologie du contact, rapport de dea,” 2004.

[6] C. Lennerz, E. Schömer, and T. Warken, “A framework for collision detection and response”, in 11th European Simulation Symposium, ESS’99, 1999, pp. 309–314.

[7] J. Sauer and E. Schömer, “A constraint-based approach to rigid body dynamics for virtual reality applications,” Proc. ACM Symposium on Virtual Reality Software and Technology, 1998.

[8] K. T. McDonnell, “Dynamic subdivision-based solid modeling,” 2000.

[9] D. E. Stewart, Time-stepping methods and the mathematics of rigid body dynamics. Birkhuser, 2000, ch. 9.

[10] K. G. Murty, Linear Complementary Linear And Nonlinear Programming. Internet Edition, 1997.

[11] A. Lécuyer, “Contribution l’étude des retours haptique et pseudo-haptique et de leur impact sur les simulations d’opérations de montage/démontage en aéronautique,” Ph.D. dissertation, Université Paris XI, 2001.

[12] S. Redon, “Algorithmes de simulation dynamique interactive d’objets rigides,” Ph.D. dissertation, Université d’Evry, 2002.

[13] E. Guendelman, R. Bridson, and R. Fedkiw, “Nonconvex rigid bodies with stacking,” ACM SIGGRAPH, 2001.

[14] P. J. Berkelman, R. L. Hollis, and D. Baraff, “Interaction with a realtime dynamic environment simulation using a magnetic levitation haptic interface device,” IEEE International Conference on Robotics and Automation, pp. 3261 – 3266, 1999.

[15] D. Baraff, “Fast contact force computation for nonpenetrating rigid bodies, siggraph,” ACM SIGGRAPH, 1994.

[16] R. Barzel and A. H. Barr, “A modeling system based on dynamic constraints,” Computer Graphics, vol. 22, 1988.

[17] A. Gregory, A. Mascarenhas, S. Ehmann, M. Lin, and D. Manocha, Six Degree-of-Freedom Haptic Display of Polygonal Models. T. Ertl and B. Hamann and A. Varshney, 2000, pp. 139–146.

[18] D. Baraff and A. Witkin, “Dynamic simulation of non-penetrating flexible bodies,” Computer Graphics, vol. 26, no. 2, pp. 303–308, 1992.

[19] J. Schmidt and H. Niemann, “Using Quaternions for Parametrizing 3–D Rotations in Unconstrained Nonlinear Optimization,” in Vision, Modeling, and Visualization 2001, T. Ertl, B. Girod, G. Greiner, H. Niemann, and H.-P. Seidel, Eds. Stuttgart, Germany: AKA/IOS Press, Berlin, Amsterdam, 2001, pp. 399–406.

[20] A. Lécuyer, S. Coquillart, A. Kheddar, P. Richard, and P. Coiffet, “Pseudo-haptic feedback: can isometric input devices simulate force feedback?” in IEEE International Conference on Virtual Reality, New Brunswick, 2000, pp. 83–90.

[21] G. V. Popescu, “The Rutgers haptic library,” in IEEE International Conference on Virtual Reality, Haptics Workshop, New Brunswick, 2000.

[22] G. Bouyer, “Rendu haptique et sonore 3d en prototypage virtuel,” 2002.

[23] P. Meseure, A. Kheddar, and F. Faure, “Détection des collisions et calcul de la réponse,” Action Spécifique DdC du CNRS, Tech. Rep., 2003.

[24] M. C. Lin and S. Gottschalk, “Collision detection between geometric models: a survey,” in IMA Conference on Mathematics of Surfaces, vol. 1, San Diego (CA), May 1998, pp. 602–608.

[25] D. A. Lawrence, “Stability and transparency in bilateral teleoperation,” IEEE Transactions on Robotics and Automation, vol. 9, no. 5, pp. 624–637, 1993.

[26] K. van den Doel, P. G. Kry, and D. K. Pai. - FOLEYAUTOMATIC : Physicallybased Sound Effects for Interactive Simulation and Animation. In Proceedings of SIGGRAPH 2001 Conference, pages 537-544 , 2001.