**Proceedings of ASME 2010 World Conference on Innovative Virtual Reality**
**WINVR 2010**
**May 12-14, 2010, Ames, Iowa, USA**

**WINVR2010-3726**

# EVALUATION OF PHYSICAL SIMULATION LIBRARIES FOR HAPTIC RENDERING OF CONTACTS BETWEEN RIGID BODIES

**Loeiz GLONDU**
ENS Cachan/IRISA/INRIA Rennes
Campus de Beaulieu
35 042 Rennes, France
Email: loeiz.glondu@irisa.fr

**Maud MARCHAL**
INSA/IRISA/INRIA Rennes
Campus de Beaulieu
35 042 Rennes, France
Email: maud.marchal@irisa.fr

**Georges DUMONT**
ENS Cachan/IRISA/INRIA Rennes
Campus de Beaulieu
35 042 Rennes, France
Email: georges.dumont@irisa.fr

## ABSTRACT

*Haptic rendering has opened a new range of virtual reality applications, enabling a human user to interact with a virtual world using the sense of touch. This kind of interaction enables to enhance applications such as computer-assisted design, where 3D manipulations are part of the system. However, building an application with an accurate haptic feedback is still challenging, especially for interactions between rigid bodies, where stiff contacts can only be displayed with a high simulation frequency.*

*This paper presents the possibilities of implementation of a modular haptic display system that relies on two main components: a physical simulation part and a haptic rendering part. For that purpose, we define a generic coupling approach that enables to perform haptic rendering using admittance haptic devices, through a scaling interface that cleanly separates the physical simulation and the haptic rendering system of units. Four physical simulation libraries are evaluated with respect to haptic rendering quality criteria, based on their behavior in four discriminant test cases. We show that the proposed approach leads to a modular, generic and stable haptic application.*

## INTRODUCTION

Haptic rendering, defined as the interaction with a virtual world using the sense of touch, is becoming more and more spread, having natural applications in virtual surgery, motion planning or computer-aided conception for example. The 3D interactions proposed by haptic rendering are attractive in these applications, giving the possibility to place the human operator in conditions comparable to the one occurring in the real world. Figure 1 shows a concrete example of a virtual workstation that can be used for ergonomics studies: a user assembles bearings using the mallet and accessing the different pieces several times, highlighting the solicited articulations and the possibilities of traumas through the user gestures. In this scenario, haptic interactions are used to put the human users in the same conditions as if they were operating on a real workstation. Therefore, quality of haptic rendering between the objects, in majority rigid in this scene, is essential. In this paper, we restrict ourselves to haptic interactions between rigid bodies, for which specific physical simulation models and methods have been defined (we refer the

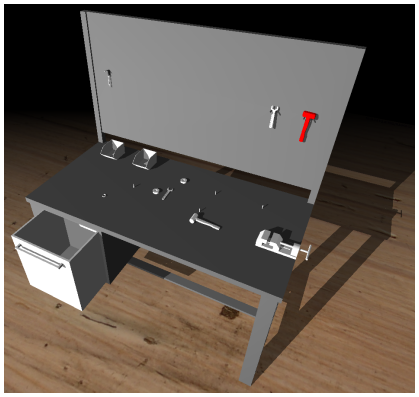

Figure 1. 3D SCENE OF A WORKSTATION FOR ERGONOMICS STUDIES. HAPTIC RENDERING ALLOWS TO DIRECTLY INTERACT WITH THE DIFFERENT TOOLS.

reader to *e.g.* [1] for an introduction to rigid body dynamics).

Haptic rendering knows a great development in research labs, and no real standard exists for haptic devices control and interfaces. For specific haptic devices and applications, the device control algorithms must be entirely defined, and can lead to fastidious tasks. However, most of the time, the controller is already defined by the haptic devices vendors, and can be accessed through an associated Application Programmable Interface (API).

Haptic applications are often implemented by plugging the user simulation directly to the haptic API. The haptic display is tuned through the API parameters, or user defined mechanisms (see Figure 2). Since haptic applications require physical simulation computations to reproduce interactions that occur in real life, frameworks embedding both physical simulation system and haptic rendering control have been developed. Some of these frameworks are presented as an extension of the haptic rendering API such as formerly **GHOST** by Sensable, now redesigned in **OpenHaptics** (`http://www.sensable.com/`). Equivalent frameworks that are not limited to Sensable haptic devices are **Reachin** (also known as **MAGMA**) (`http://www.reachin.se/`), or **Chai3d** (`http://www.chai3d.org/`). Other frameworks (see *e.g.* **I-TOUCH** [2], and **ImmersiveTouch** [3]) also embed the technical and material aspects of a haptic application, proposing working space, visual display system and sometimes a tracking system.

In this paper, we present an alternative way to build haptic applications using a physical simulation library (called dynamic engine) and a haptic device and associated API, coupled through an interface as illustrated on Figure 2. Our approach has the advantage to consider the two components as black boxes. The intermediate interface, which is responsible for the haptic display tunings, enables to link them and obtain a stable, modular and generic haptic rendering application. Dynamic engines are usually designed for real-time visual applications such as games or offline accurate simulations. Therefore, they may not be adapted to haptic rendering, where the frequency of the simulation should be over 1kHz and accurate at the same time. This observation motivated us to begin our work with an evaluation of the existing dynamic engines with respect to haptic rendering.

The organization of the paper is as follows: we present in the first part an evaluation and a comparison of four real-time dynamic engines based on their rigid body simulation capabilities, with respect to haptic rendering quality criteria. In a second part, we show how it is possible to simply link the result of the physical simulation to the haptic API, introducing a useful scaling interface of physical quantities between the real world and the virtual world. We end the document with a discussion on our results.

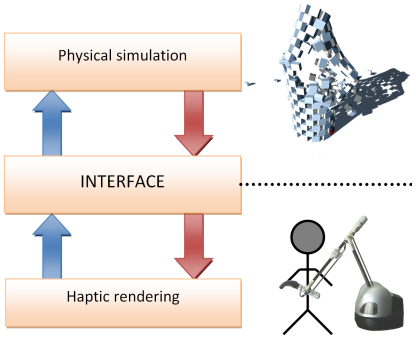


Figure 2. TWO MAIN COMPONENTS OF THE MODULAR HAPTIC APPLICATION (THE PHYSICAL SIMULATION MODULE AND THE HAPTIC RENDERING) COUPLED THROUGH A GENERIC INTERFACE.

## EVALUATION OF RIGID BODY DYNAMIC ENGINES

We present in this section an evaluation of dynamic engines, based on relevant criteria chosen with respect to haptic rendering of contact between rigid bodies. The objective of this evaluation is to determine whether popular real-time dynamic engines are well-suited for haptic rendering, and to highlight their limits. The list of dynamic engines evaluated below is not exhaustive, but we tried to choose the dynamic engines that seemed the most promising to us. Moreover, we defined an environment for our experiments that is modular enough to be able to integrate any physical simulation library.

- **Havok physics** (`http://www.havok.com`). Havok was created in 1998 in Dublin (Ireland), and is the world leader provider for rigid body dynamics solution. It also implements sophisticated collision detection algorithms, ragdoll animation, vehicles dynamics and character animation toolkit.
- **NVidia PhysX** (`http://www.nvidia.com/`, section PhysX in technology menu). PhysX (former NovodeX) has been created in 2002 by the semi-conductor company Ageia that was the first to propose on the market a hardware acceleration solution called PPU (Physic Process Unit) for physical simulation. NVidia PhysX implements collision detection algorithms, rigid, soft body and fluids simulation. PhysX has recently been acquired by NVidia (in February, 2008) to become NVidia PhysX (the hardware acceleration capabilities have been deported on the GPU).
- **Bullet physics** (`http://bulletphysics.org/`). Bullet physics is an open source physic engine founded in 2003 by Erwin Coumans, a former Havok employee and is supported by Sony Entertainment division. Like its competitors, Bullet physics provides collision detection and rigid body simulation. It also implements soft body simulation.

• **Open Tissue** (`http://www.opentissue.org/`). OpenTissue is born from a research project initiated by Kenny Erleben [4]. He presents in his papers inviting algorithms for rigid body simulation. OpenTissue integrates collision detection system and simulation of rigid and deformable bodies.

All these dynamic engines use Gauss-Seidel like iterative solvers to manage non-penetration and user-defined constraints, that are expressed at the velocity level, *i.e.* the solver will try to find impulses that prevent or correct the constraints violations. (OpenTissue performs a final shock-propagation stabilization step as presented by its author). They are all implemented at least on Windows, Linux, MacOSX, and, except for OpenTissue, Playstation 3, Xbox 360 and Nintendo Wii.

Our evaluation is focused on a comparison between the results obtained with the different libraries. We do not present technical comparisons of the methods used by the dynamic engines. Moreover, since neither Havok physics nor NVidia PhysX is open source, it remains difficult to have information on the underlying algorithms used for collision detection and constraints resolution.

## Performance Criteria

We designed our experiments in order to measure the quality of haptic rendering through the three following performance criteria:

• **Computation time**. A stable and realistic haptic rendering needs high refreshment updates (see *e.g.* [5]). It is commonly admitted that a haptic display that renders contacts and impacts between rigid bodies should be performed at about 1kHz, *i.e.* the computation time spent to compute the state of the world from time $t_i$ to time $t_{i+1}$ must be performed in less than one physical millisecond. In order to measure this criterion, we start a timer before the call to the simulation method of the dynamic engine (this call includes collision detection and constraints solving) and stop it after the end of the call.
• **Stability**. The stability of the simulation indirectly measures how laws of physics such as energy conservation are respected. Passivity of a virtual world [5] (*i.e.* the fact that the world only dissipates energy) is a sufficient condition for ensuring its stability. Therefore, we made measurements of the variations of the total world energy to conclude on its stability.
• **Accuracy** The accuracy indicates how well the physical phenomena (such as dry or sliding friction, bouncing, ...) are reproduced in the virtual world. To give a mark for spatial accuracy, we performed spatial measurements of penetrations distances (using Euclidean distance between bodies centers as metric). We also visually appreciated the results based on reference simulations of the real world.

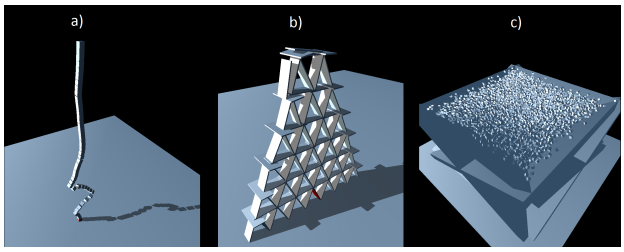


Figure 3. THREE TEST CASES IN IMAGE. (**a**) A PILE OF 50 CUBES. (**b**) A CARD HOUSE COMPOSED OF 89 CARDS. (**c**) 8000 CUBES IN A BASIN.

For each test case presented in the next section, we measure the average and maximum computation times, the sum of the total energy of all the bodies of the world, and we give a mark on the visually appreciated end state. If $\mathcal{B}$ is the set of bodies indexes of the world, $m_i$, $v_i$, $\omega_i$ and $J_i$ are respectively the mass, linear velocity, angular velocity and inertia matrix (related to the center of mass) of body $i$, the energy $e$ of the world is computed as:

$$e = \sum_{i \in \mathcal{B}} \frac{1}{2} \left( m_i v_i^2 + \omega_i^T J_i \omega_i \right) \quad (1)$$

## Tests Cases

We used four discriminant test cases for the measurements of our performance criteria (the words in italic facing the name of the tests indicate which main criterion is measured through the test):

1. **Pile of 50 cubes** – *stability*. The classical pile of cube test (Figure 3a) is a challenging structure because of its contact disposition: naive iterative solvers have a very slow convergence rate in order to propagate the non penetration constraints [6]. Also, this test measures the efficiency of the error correction due to interpenetration occurring.
2. **Seven-stages card house** – *friction accuracy*. The card house (composed of 89 cards, Figure 3b) is a structure that fully depends on an accurate simulation of friction phenomena [7]. If the friction is too much approximated or enforces penetrations, the card house is destabilized and collapses.
3. **8000 cubes in a basin** – *computation time*. In order to check the scalability of the libraries, the third test consists in dropping 8000 cubes in a basin (Figure 3c). A lot of objects are put in a high contact configuration (8 contacts per body in average), measuring the evolution of the timings of the solvers when the numbers of bodies and contact increase.

4. **Heavy block on a light block** – *efficiency of solvers*. This test puts Gauss-Seidel like solvers into slow convergence rate. If not enough iterations are used, or the correction methods are unappropriated, the upper heavy block penetrates the light one, and the system becomes unstable.

## Test Parameters

Since our tests are related to rigid body simulation, we retained two physical parameters:

• **The coefficient of friction**. According to the Coulomb model of friction, the coefficient of friction $\mu$ tells that the maximum tangential force that can be applied to oppose tangential motion at a contact point between two rigid bodies, is $\mu \times f$, $f$ being the magnitude of the normal force acting at the contact point to prevent interpenetration. As an example, the coefficient of friction between two dry pieces of steel is approximatively 0.15. Although friction is a very common phenomenon, it is far from simple to include accurately into rigid body dynamics [8]. Thus, we vary the coefficient of friction from 0 (no friction) to 1 (high friction) for each test case in order to see its influence on the obtained results, and how well the dynamic engines handle it.

• **The coefficient of restitution**. The coefficient of restitution indicates what percentage of energy is conserved and dissipated during an impact between two rigid bodies (0 means a total inelastic impact, while 1 means a perfect elastic and bouncy impact). In contact resolution between rigid bodies, resting contacts are often separated from collision events and are treated using different algorithms. To decide whenever a resting contact or a collision occurs, the relative velocity between colliding bodies at contact point is often considered. Since the restitution coefficient has an influence on how relative velocities are modified, we chose to study the influence of the restitution coefficient on the simulation results.

All our tests have also been performed by varying the time step. Depending on the integration scheme used, the time step has namely a great influence on the stability of the constraints solver. The different combinations of parameters are summed up on Table 1.

**Configuration of the dynamic engines** Each dynamic engine has its own parameters that may have impacts on the results. For example, it is possible to set the number of iterations used for constraint solving. Since the default configuration of the dynamic engines is set to optimize a trade-off between computation time and accuracy performances, we chose to let the default values. However, other aspects such as aggressive freezing strategies can alter the results. Another aspect is the collision detection system that can be turn into continuous or discrete mode, having great impacts on computation time and results. To make the comparison possible, we disabled freezing and continuous collision detection for all the results presented in this paper.

Table 1. DIFFERENT CONFIGURATIONS OF PARAMETERS (THE FRICTION COEFFICIENT, THE RESTITUTION COEFFICIENT AND THE TIME STEP) USED FOR EACH TEST CASE.

| Friction coefficient | Restitution coefficient | Time step (s) |
|---|---|---|
| 0.5 | 0.4 | 1/60 |
| 0.5 | 0.0 | 1/60 |
| 0.5 | 0.4 | 1/100 |
| 1.0 | 0.4 | 1/60 |
| 0.0 | 0.4 | 1/60 |

## Results

This section summarizes the results obtained for each of our performance criteria, for the four test cases previously defined. We believe that the collision detection system of OpenTissue slows down the simulation times drastically, and noticed that the stability of the simulation is lower than the other libraries. Therefore, we do not present the results obtained with OpenTissue since they are not comparable to the results obtained with the other dynamic engines.

Computation times comprise both the collision detection, constraint solving and integration time. We performed all our tests on an Intel Pentium D (3.40 GHz) with 2.0 GB RAM on Windows XP. Except the third test for which we measured the results after 10 seconds of simulation (an arbitrary chosen representative value), we stop the simulation and the measurements whenever the simulation has reached a stable state.

**Pile of cubes** Havok physics brought the best results for the simulation of the pile of cubes. We obtained the best average computation times, and the best stability. It is possible to make the pile to stand up for time step going up to $1/60$s. Figure 4a shows the dissipation of the total energy that led to a stable state for the pile of cube using Havok physics and a time step of $1/100$s.

With NVidia PhysX, we notice a visible penetration between cubes at the beginning of the simulation, followed by a counter reaction that destabilizes the pile which breaks before 5 seconds of simulation if a time step over $1/100$s is used. Using tiny time steps (less than $1/800$s), it is possible to make the pile to hold, but it never reaches a stable state, and small oscillations can be observed.

Bullet physics brought not as good computation time results

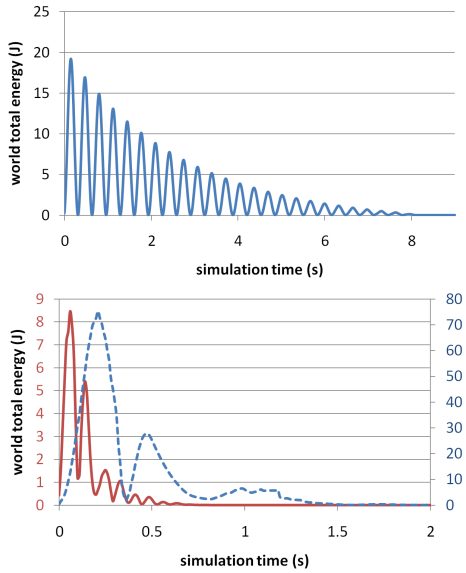


Figure 4. TOTAL ENERGY MEASURED OVER TIME (**a**) ON THE PILE OF CUBE TEST WITH HAVOK PHYSICS (**b**) ON THE CARD HOUSE TEST WITH HAVOK PHYSICS (RED SOLID LINE) AND NVIDIA PHYSX (BLUE DASHED LINE). TIME STEP = $1/100$s.

as its competitors, and had similar results than NVidia PhysX on the stability plan (we did not manage to make the pile hold more than 4 seconds of simulation for time steps over $1/100$s).

Using a null coefficient of friction, none of the engine enables to make the pile hold. However, in this case, we obtained the best results with Havok physics that maintains the pile for more than 4 seconds of simulation against 1.5 seconds for NVidia PhysX, and less than 1 second for Bullet physics.

**Card house** Havok physics enables to simulate the card house in a visually realistic manner, and with the best computation times. With a time step of $1/100$s, and a friction coefficient of 1, even the top most stage of the card house remains in place. NVidia PhysX shows the same phenomenon as for the pile of cubes: after a penetration between the cards, a counter reaction occurs (see Figure 4b at time 0.5s) and the card house is destabilized. However, although the card house is maintained for a long time, it never reaches a stable state before the card house is almost completely broken (after more than 50 seconds of simulation). Bullet physics shows a weakness on this test case: its default solver can not handle properly dry friction. Indeed, whatever the coefficient of friction used, the cards slide and the whole house is broken at the beginning of the simulation.

**8000 cubes in a basin** On this test case, we mainly measured the average and maximum computation times. For this simulation, we saw that Havok physics and NVidia PhysX brought very

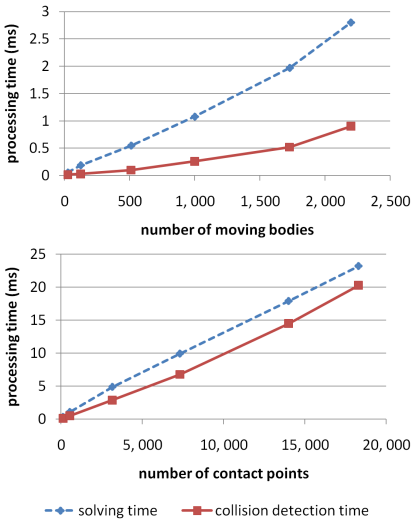


Figure 5. LINEAR RELATION BETWEEN (**a**) THE NUMBER OF BODIES OR (**b**) THE NUMBER OF CONTACTS AND COMPUTATION TIMES FOR HAVOK PHYSICS.

similar results. We noted however a slightly greater maximum computation time for Havok physics when the coefficient of friction is zero. Bullet physics is on the third place, with the highest computation times.

Figure 5 shows the computation time evolution with respect to the number of bodies, and the number of contacts in the virtual world. On this chart, we separated the call to the collision detection module and the constraints solving, enabling to show the time spent for collision detection (solid line) and the time used for constraint solving (dashed line).

**Heavy block on a light block** None of the dynamic engines has been able to simulate visually plausible results for this test whenever the mass ratio between the two bodies is above 500 (see Figure 6, white body is heavy while the dark one is light). Actually, using a time step of $1/1000$s and masses equal to 1 and 500, it is possible to reach a stable state with Havok physics where the heavy cube is resting on the light one, after several unnatural bounces. We did not achieve to have the same state using NVidia PhysX or Bullet physics with this mass factor. NVidia PhysX has a different behavior, allowing the heavy cube to penetrate the light one (see Figure 6b, middle). This makes the light cube in red becoming unstable, being randomly shook until it is ejected away from the heavy cube that does not move. With Havok or Bullet physics, the interpenetration is corrected in such a manner that the heavy cube (and the light one for Havok) bounces abnormally over the light one, until both cubes are separated by tangential forces. Using a bigger time step (1/60s), the heavy cube penetrates the light one, and the latter is pushed away horizontally.

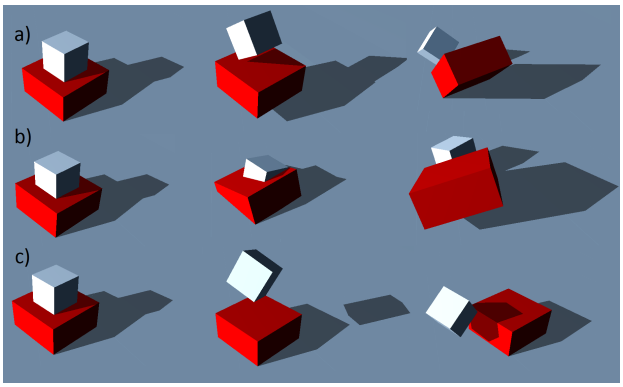


Figure 6. FOURTH TEST CASE. A HEAVY CUBE (WHITE) IS DROPPED ON A LIGHT ONE (DARK RED), WITH A SCALE FACTOR OF 1000 ON MASSES. (**a**) RESULTS WITH HAVOK PHYSICS. (**b**) RESULTS WITH NVIDIA PHYSX. (**c**) RESULTS WITH BULLET PHYSICS. TIME STEP = 1/800s, FRICTION COEFFICIENT = 0.5, RESTITUTION COEFFICIENT = 0.4.

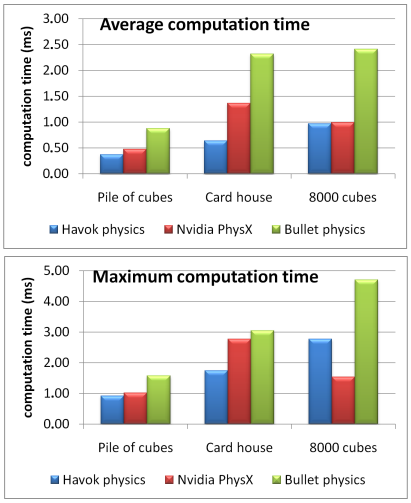


Figure 7. AVERAGE AND MAXIMUM PROCESSING TIMES FOR THE THREE FIRST TEST CASES.

### Summary of the Evaluation

We performed four discriminant tests, each of them was designed to measure one of the performance criterion. Figure 7 sums up the average timings (over all the configurations of parameters of Table 1) for each of the three first test cases (we do not show timings for the fourth test, since the scene composed of two bodies is not relevant for computation time comparisons).

We noticed that Havok physics has a small advance on computation time performance on NVidia PhysX, while Bullet seems to be less optimized. In average, we measured that it takes about 0.012 ms to solve for one cube in a contact configuration of 8 contacts per cube for Havok physics or NVidia PhysX, with our hardware configuration.

Table 2. SUMMARY OF THE STABILITY AND ACCURACY APPRECIATIONS. LEGEND ++: GOOD ACCURACY, STABLE SIMULATION. +: VISUALLY REALISTIC RESULTS, STABILITY REDUCED. 0: SIMULATION NOT SUCCESSFUL.

| | Pile of cubes | Card house | Heavy/light block |
|---|---|---|---|
| Havok physics | ++ | ++ | 0 |
| NVidia PhysX | + | + | 0 |
| Bullet physics | + | 0 | 0 |

Table 2 sums up the accuracy conclusion. We noticed that Havok leads to the more stable and accurate simulation, allowing to simulate the pile of cubes and the card house successfully. NVidia PhysX shows some weaknesses on those challenging structures, allowing an initial interpenetration that destabilizes the simulation further. Bullet physics does not seem to handle properly dry friction, making structure such as the card house impossible to be correctly simulated.

## COUPLING DYNAMIC ENGINE AND HAPTIC RENDERING

The first section presented an evaluation of the dynamic engines abilities for haptic rendering applications. In this section, we focus on how it is possible to couple the two main components of our haptic application, *i.e* how to couple the dynamic engine with the haptic rendering API, in order to obtain a modular and generic system. For that purpose, we introduce a scaling interface between the two components that in order to tune the haptic display without altering the overall stability of the system.

Haptic devices can be controlled in two modes: the *admittance* control mode and the *impedance* control mode (see *e.g.* [9] for a discussion on these control modes). In the impedance (or direct) control mode, the haptic device is waiting for forces to be applied on the user, and returns the current position of its tip to the simulation. In admittance control mode, the device waits for position that the tip must reach, and returns forces to the simulation. This control mode is well-suited for a simple coupling between a dynamic engine and a haptic rendering API. Indeed, the dynamic engines provide successive positions of the proxy (the body in the virtual world which is coupled for haptic interaction) over time, that can be used to pilot the haptic device. The forces returned by the haptic device can be applied on the proxy and integrated into the dynamic engine. Figure 8 shows the flow of data when using admittance mode. It is however too restrictive to plug directly the simulation results to the haptic API as shown on Figure 8. Indeed, values returned by the dynamic engine may

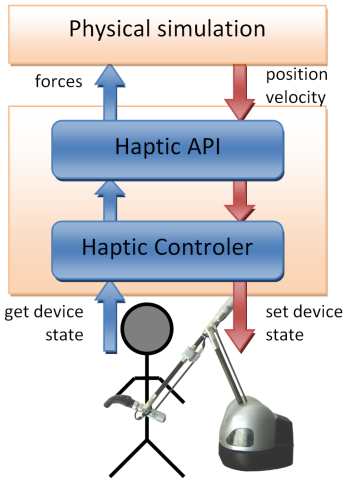


Figure 8. ADMITTANCE CONTROL OF AN HAPTIC DEVICE. HAPTIC RENDERING INCLUDES HAPTIC API, HAPTIC CONTROLLER AND HAPTIC DEVICE.

have a numerical representation different from the one used by the haptic API, the frame coordinates can be different, and the order of magnitude of the values used are likely to be incompatible. Moreover, haptic display must be tunable: changing the range of efforts allowed and calibrating the interaction so that most of the user movements stay in the haptic device working space are two manipulations that should offer haptic applications. These observations motivated us to define a scaling interface between the two modules as presented in the next section.

## Scaling Factors between Virtual and Real World

Each haptic device has its own mechanical constraints and working space. The haptic device used during our experiments (a 6 DOF Virtuose$^{TM}$6D35-45, from Haption, Soulge sur Ouette, France) supports less than 40 Newtons of effort, and has a workspace which is approximatively a 30cm side cube, while a PHANTOM Omni haptic device from Sensable (Woburn, MA 01801, USA) has a $16 \times 12 \times 7$cm working space, supporting around 3.3 Newtons of effort. Of course, the two devices are not designed for the same applications, but their differences highlight the fact that the physical simulations values can not be directly used for haptic rendering. Moreover, we want to be able to manipulate objects of a mass of several tons in huge environments as well as interacting with microscopic objects.

It is possible to amplify the movements of the user, and to give him more force so that the interactions fit into the working space of the haptic device and its range of allowed efforts. However, altering the coupling has consequences in both the simulation and the control of the admittance haptic device and the physical simulation. In this section, we propose an interface between the physical simulation and the haptic rendering that considers the two modules as black boxes, and that enables to freely give values for scaling factors, without altering the stability of the system to obtain a generic coupling.

In the following, the subscript $x_{simu}$ denotes a value $x$ coming from the physical simulation module, while the subscript $x_{hapt}$ denotes a value $x$ coming from the haptic rendering module.

**The scaling factor** We define the scaling factor $z_{fact}$ as the factor linking the virtual world unit of length (denoted $v_l$) to the real world unit of length, the meter. For example, a factor of 10 means that 1 virtual unit of length (*i.e.* the value 1 in the numerical representation of the length in the dynamic engine) is equivalent to 10 meters in the real world. This factor is useful for example to choose a value for the gravity of the virtual world that has the same representation than on the earth (*i.e.* around $98 v_l.s^{-2}$ for an earth equivalent gravity if $z_{fact} = 10$). Also, this factor can be helpful when creating the virtual world, and computing the mass of the objects from their density in $kg.m^{-3}$ to the virtual world density in $kg.v_l^{-3}$. The scaling factor has only an influence on the physical simulation (and the conception of the world) and thus does not belong to the scaling interface between the physical simulation and the haptic rendering. However, it is really helpful to cleanly separate the scaling interface, and to avoid mixing independent and linked variables into the coupling.

Since the physical quantities used for haptic rendering are related to space, mass and time dimensions, we define additional length and mass factors as presented in the following.

**The length factor** We define a length factor $s_{fact}$ that modifies the unit of length of the virtual world $v_l$ so that the simulated scene fits into the desired working space. To ensure the good separation of the systems of units of the physical simulation and the haptic rendering, we use this factor to modify the linear position $x_{simu}$ in meter, the linear velocity $v_{simu}$ in $m.s^{-1}$ and the linear forces $f_{simu}$ in $kg.m.s^{-2}$ linearly, while the torque $\tau_{simu}$ in $kg.m^2.s^{-2}$ and the inertia matrix $I_{simu}$ using the square of the factor:

$$x_{hapt} = \frac{x_{simu}}{s_{fact}} \; ; \; v_{hapt} = \frac{v_{simu}}{s_{fact}} \tag{2}$$

Concerning the orientation values $q_{simu}$ and angular velocity $\omega_{simu}$, it would be possible to similarly define a rotation factor, that could amplify or decrease the angular motion. We however kept the original values for orientation and angular velocities (the angular quantities have no unit, and do not scale with length or mass):

$$q_{hapt} = q_{simu} \; ; \; \omega_{hapt} = \omega_{simu} \tag{3}$$

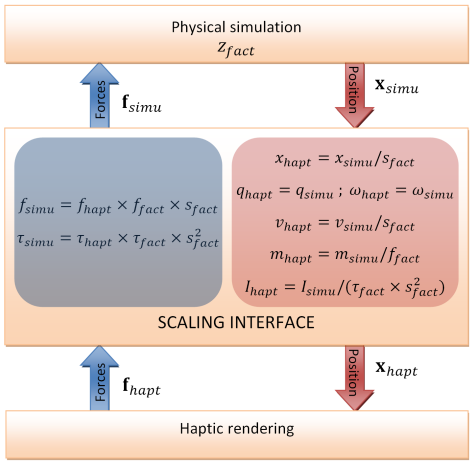


Figure 9. SCALING INTERFACE BETWEEN THE PHYSICAL SIMULATION AND THE ADMITTANCE HAPTIC RENDERING. BOLD VALUES DEPICT GLOBAL STATE VECTOR OR GLOBAL EFFORT VECTOR.

**The mass factors** We finally define the force factor $f_{fact}$ and torque factor $\tau_{fact}$ that increase or decrease mass values. The force factor modifies linearly the force returned by the haptic device $f_{hapt}$ and the mass $m_{simu}$. The torque factor modifies linearly the torque $\tau_{simu}$ and the inertia matrix $I_{simu}$:

$$f_{simu} = f_{hapt} \times f_{fact} \times s_{fact} \; ; \; m_{hapt} = \frac{m_{simu}}{f_{fact}} \tag{4}$$

$$\tau_{simu} = \tau_{hapt} \times \tau_{fact} \times s_{fact}^2 \; ; \; I_{simu} = \frac{I_{hapt}}{\tau_{fact} \times s_{fact}^2} \tag{5}$$

Figure 9 sums up the scaling interface between the physical simulation module and the admittance haptic rendering module.

## Synchronization with Physical Time

Although length and mass factors can be easily interpreted, time factor is more confusing. We want the virtual unit of time to coincide with the real second. Therefore, we perform an active wait at the end of each simulation step using the system clock, enabling to synchronize the simulation time with physical time (see Figure 10). To be feasible, this solution implies that the computation time needed to move the world $n$ seconds forward is smaller than $n$ physical seconds. Otherwise, the simulation takes some latency. To avoid this latency, the number of bodies in the virtual world can be limited. Indeed, if we master the time needed to simulate one body in the worst case (see the conclusion of the evaluation section), then we can estimate the maximum number of bodies that it is possible to simulate within a given time step in order to avoid the simulation to take some latency.

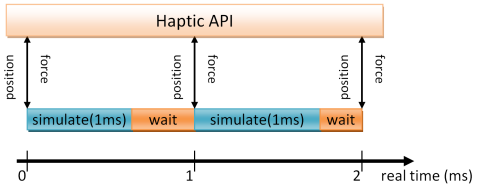


Figure 10. SYNCHRONIZATION WITH PHYSICAL TIME USING ACTIVE WAIT.

## Results

We implemented the coupling of Figure 2 using an admittance capable haptic device (a 6 DOF Virtuose$^{TM}$6D35-45, from Haption), and our scaling interface and time synchronization. In order to avoid the visual display to slow down the haptic frequency, we performed virtual world image synthesis into a separated thread that performs read-only operation on the virtual world (and is not synchronized with the simulation thread).

We experienced stable haptic interactions using Havok physics, NVidia PhysX and Bullet physics. The more pleasant and accurate interactions have been obtained with Havok physics, where we could feel dry and sliding frictions performing up to 1kHz haptic rendering for the pile of cube, and nearly 800Hz haptic rendering for the card house test.

Comparable interactions have been obtained with NVidia PhysX. In simple cases, fine interactions with dry and sliding frictions can be performed. However, interactions involving stacking structure such as the pile of cube and the card house, where visual oscillations appear, do not spare the haptic display that suffers from the same artifact.

Finally, we also performed satisfying interactions using Bullet physics. The main difference between the two previous haptic displays is the friction phenomena. Indeed, the tangential motion is felt viscous, and no clear dry friction is haptically reproduced.

## Discussion and Summary of the Coupling

The scaling interface enables us to easily tune the haptic display using few factors, but can also manage changes in frame coordinate or numerical representation interface. We presented simple linear or quadratic scaling functions for physical quantities, based on unit analysis. These functions have the advantage of isolating two systems of units, and thus conserve the original stability. The definition of other scaling functions must be carefully done in order to avoid loss of stability leading to unstable situations in haptic rendering.

For the purpose of our evaluation, we defined a multithreaded test environment and a physic abstraction layer in order to have a full control on the parameters and the measurements. However, the abstraction layer and test environment are steps that can be skipped for general haptic rendering. Using a multithreaded environment is common in haptic rendering, to avoid

loss of time for external tasks such as visual rendering. Although we do not detail this aspect in this paper, it does not invalidate the scaling interface, which is proper to haptic rendering.

The list below sums up the steps that quickly and simply lead to a stable and modular haptic application:

1. Choose a dynamic engine, or an abstract layer that includes several dynamic engines such as PAL (`http://www.adrianboeing.com/pal/`) which embeds about ten dynamic engines in a unified interface.
2. Choose a haptic device that is well-suited for the application and that can be controlled in admittance mode (an abstraction layer for haptic device control would be even more elegant).
3. Define the scaling interface presented in the second part of the paper (and optionally the synchronization time procedure), and plug it to the dynamic engine and the haptic API as shown on Figure 9.

## CONCLUSION AND PERSPECTIVES

In this paper, we presented an evaluation of four dynamic engines with respect to haptic rendering quality criteria. We observed that all of engines present computation time and accuracy performances compatible with haptic rendering. Using admittance controlled haptic device, we showed how it is possible to couple the dynamic engines results with the haptic rendering API, defining a scaling interface that enables to modify the magnitudes of the physical quantities exchanged, without altering the stability of the overall system. We tested the coupling and the scaling interface on several scenarios (such as the test cases, or scenarios of Figure 11), maintaining the 1kHz haptic frequency constraint on scenes containing up to 50 movable rigid cubes on our configuration. Our results demonstrate that some real-time dynamic engines are adapted for accurate and stable haptic displays between rigid bodies, and that our scaling interface allows to quickly and simply obtain a modular and generic haptic application.

Our results motivated us to leverage this simple but modular architecture to build an application that manages multiple haptic displays (using potentially different haptic devices) and several dynamic engines. We aim at dynamically and adaptively choosing the dynamic engines and haptic display that fit the best with the current context of interaction. Such an application would help us to concretize ergonomics studies using an estimation of muscle forces involved in real interactions [10]. We think modular and generic haptic application may be useful in any context of interaction that involves haptic displays of different natures in complex physical environments (*e.g.* in virtual surgery, where several tools can be used for the medical gestures on complex anatomical structures).

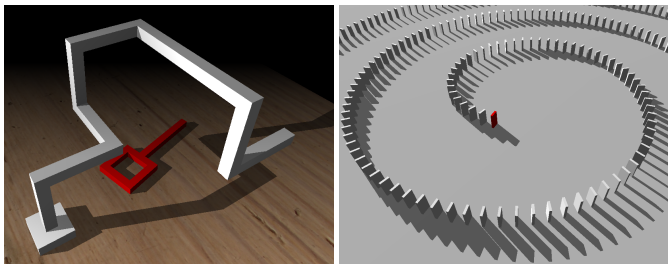

Figure 11. TWO HAPTIC RENDERING SCENARIOS.

## REFERENCES


[1] Baraff, D., 2001. "Rigid body simulation". In Physically Based Modeling, SIGGRAPH Course Notes.

[2] Pocheville, A., and Kheddar, A., 2004. "I-TOUCH: a framework for computer haptics". In Workshop on Touch and Haptics, IEEE/RSJ IROS.

[3] Luciano, C., Banerjee, P., Florea, L., and Dawe, G., 2005. "Design of the IMMERSIVETOUCH: a high-performance haptic augmented virtual reality system". In Salvendy G: Human Computer International Proceedings.

[4] Erleben, K., 2007. "Velocity-based shock propagation for multibody dynamics animation". *ACM Transactions on Graphics,* **26**(2), pp. 1–20.

[5] Colgate, J. E., Stanley, M. C., and Brown, J. M., 1995. "Issues in the haptic display of tool use". *Intelligent Robots and Systems, IEEE/RSJ International Conference on,* **3**, pp. 140–144.

[6] Milenkovic, V. J., and Schmidl, H., 2001. "Optimization-based animation". In Proceedings of the 28rd annual conference on Computer graphics and interactive techniques, pp. 37–46.

[7] Kaufman, D. M., Sueda, S., James, D. L., and Pai, D. K., 2008. "Staggered projections for frictional contact in multibody systems". *ACM Transactions on Graphics,* **27**(5), pp. 164:1–164:11.

[8] Baraff, D., 1991. "Coping with friction for non-penetrating rigid body simulation". In Proceedings of the 18th annual conference on Computer graphics and interactive techniques, pp. 31–41.

[9] Meseure, P., and Kheddar, A., 2007. *Le traité de la Ralité Virtuelle - Tome 3*. Fush F. Moreau G., ch. Modèles pour le rendu haptique, pp. 141–154.

[10] Pontonnier, C., and Dumont, G., 2009. "Inverse dynamics method using optimisation techniques for the estimation of muscle forces involved in the elbow motion". *International Journal on Interactive Design and Manufacturing (IJIDeM),* **3**, pp. 227–235.