Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2009)
H. Prautzsch, A. Schmitt, J. Bender, M. Teschner (Editors)

A Sub-world Coupling Scheme for Haptic Rendering of Physically-based Rigid Bodies Simulation

L. Glondu, M. Marchal and G. Dumont

INRIA Bunraku, IRISA, Campus de Beaulieu, Rennes, FRANCE

Abstract

In the virtual reality context, the use of haptic rendering as an additional sensory modality significantly improves the degree of realism of virtual worlds. The physical realism of the interaction between the user and the objects of the virtual world is particularly important when dealing with contact or collision between rigid objects as, for example, in assembly tasks. The high frequency rates required for smooth manipulations are often difficult to reach, in particular for rigid bodies simulations. Hence, we propose a new coupling scheme based on a dynamic subset of the virtual world, a localized Haptic Sub-World, running at a higher frequency than the rest of the virtual world. This Sub-World, located around the virtual object manipulated by the user, is synchronized with the virtual world through a dynamic analysis of the interface between the two subsets. Using this coupling scheme in our software environment, we are able to achieve high frequency haptic rendering using sophisticated simulation methods on virtual worlds with a large number of rigid bodies.

Categories and Subject Descriptors (according to ACM CCS): I.6.8 [Simulation and Modeling]: Types of Simulation—Parallel I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling I.3.7 [Computer Graphics]: Three Dimensional Graphics and Realism—Virtual Reality

1. Introduction

In virtual reality, letting a human user interact through the sense of touch with a virtual world greatly improves his sensation of immersion. If the simulation takes into account the physical properties of the objects of the virtual world, it allows to create realistic and interactive applications. Physical simulation methods for interactive applications have been widely studied during the last decades [MSJT08], and different software solutions resulting from research works are now available either as open source or commercial products.

Haptic rendering is the modality of interaction with a virtual world relative to the sense of touch. A user usually holds a haptic device with his hands, through which he can manipulate tools in a virtual environment. Haptic rendering is a bidirectional interaction: the user brings energy to the system by moving the device, while a force feedback is computed from the simulation in order to make the user feel the virtual world. As of today, the possibilities of interaction offered by haptic rendering lead to an increasing number of planification and learning applications, for example in industry for assembly tasks or in health domain for gesture planning. Among the different types of possible physical interactions, haptic rendering between rigid bodies is particularly challenging as it requires critical processing times to achieve a stable and realistic force feedback [CSB95]. To take this constraint into account, we often find in the literature *ad hoc* models or methods [AKO95, Bal99, OL05], enabling interaction with haptic rendering in specific cases. However, there is still a lack of coupling schemes between physical simulation of rigid bodies and haptic rendering that are independent of the underlying models or methods used to simulate the world. In this context, our paper proposes a new coupling scheme for haptic rendering independent of any specific method or model for physical simulation of rigid body interactions.

Organization of the document The paper is divided into five sections. After presenting related work in Section 2, the design of a generic environment for the integration and the evaluation of three dynamic engines allowing rigid body simulation is detailed in Section 3. Section 4 presents our new coupling scheme, and the results we obtained. Section

DOI: 10.2312/PE/vriphys/vriphys09/067-076

5 concludes this paper with a discussion on our contributions and some possible perspectives.

2. Related Work

Among the existing methods for physically simulate contacts between rigid bodies, three categories of method can generally be distinguished in the literature: penalty-based methods, impulse-based methods and constraint-based methods. Penalty-based methods [MW88] express a reactive force at each contact point between rigid bodies depending on an amount of penetration. They are simple and fast but may be difficult to work with due to parameter tuning as well as instabilities can appear when stiff contacts have to be simulated for large scale simulations. Impulse-based methods [Hah88] are based on momentum conservation laws to express immediate changes of velocities between rigid bodies, and to prevent inter-penetrations. However, this type of method treats contact points independently, and can then difficultly be scaled to large systems of bodies, despite recent advances [GBF03, Erl07]. Finally, constraint-based methods use the physical laws as constraints on the state of the bodies. Currently, the most common solution is Linear Complementarity Programming (LCP). More recently, different formulations have been proposed, expressing the physical simulation into convex quadratic optimizations under linear constraints [MS01, KEP05, KSJP08].

All these methods have been used to perform haptic rendering, and severals frameworks are available today [LBFD05, PK04]. Constraint-based methods offer the best physical realism for most simulated scenes. However, haptic rendering imposes serious constraints on the simulation processing time. Actually, it has been established [CSB95] that the stability of a haptic rendering loop and the stiffness of the force feedback increase with the frequency of the simulation. Therefore, to the best of our knowledge, constraint-based methods have been used only in simple cases to perform haptic rendering because of their higher processing time [RK00]. Impulse-based methods treat resting contacts with series of micro-impulsions, making them deprecated for resting contact haptic rendering [CC97]. However, they are useful to render impact between rigid bodies avoiding any inter-penetration, and to increase the stiffness of the haptic feedback [CSC05]. Finally, due to their simplicity and fast execution, penalty-based methods have been widely used to perform haptic rendering. When using penalty based methods, the bottleneck of the time stepping is the collision detection between bodies. In order to accelerate the collision detection, one can make the assumption that only the virtual tool is moving [GLGT99], leading to volume hierarchy models [RMB*08] or distance field techniques to get penetration distances. Collision detection between complex geometries can be accelerated by sampling the geometry in so-called point shell, or even point shell hierarchy [BJ07] to sample contact points and approach the response. The localized simplification of the geometries without loosing perceptual information [OL05, OL06] has also been studied for collision detection acceleration.

In order to comply with the frequency constraint brought by haptic rendering, some authors have defined the concept of intermediate representation [AKO95, Bal99], which requires a specific software architecture and allows to dissociate the simulation from the haptic interface control. The principle is to gather around the manipulated virtual tool called proxy some information that can be exploited by the haptic control loop to deliver orders at the required frequencies. In this way, an *ad hoc* control loop based on the intermediate model is defined. What we propose in this paper is a coupling scheme where the intermediate model potentially uses the same model and methods than the simulation loop, but that is working on less data to make it run faster. No *ad hoc* control loop has to be developed.

Our contributions can be summed up as:

- The design of a generic environment that can integrate any physical simulation software for rigid body simulation, abstracting ourselves from choosing specific models and methods for the physical simulation. Different haptic rendering criteria can then be evaluated.
- A new coupling scheme between simulation of rigid body dynamics and haptic rendering, independent from the physical simulation method. This coupling based on a concept of *haptic sub-world* allows to perform high frequency haptic rendering in a scene with a large number of contacts and rigid bodies, and thus offers promising perspectives.

3. A Generic Environment for the Simulation of Rigid Body Interactions

Our objective of defining new coupling schemes led us to design a generic environment to integrate dynamic engines, providing both an unified solution for physical simulation and the possibility to use several algorithms for rigid body simulation. We designed this environment to set up an evaluation of different dynamic engines of the literature with respect to haptic rendering quality criteria.

3.1. A Generic Environment

In order to allow the integration of different dynamic engines to our environment, we defined a common interface and data format that represented the description of a set of rigid bodies being simulated in a virtual world. This interface stores the configuration of the world in a common structure, and defines high level methods to access the dynamic engines capabilities in a transparent way. We have also implemented helper methods in this interface, such as statistical values on performances, stability measurements and world state history generation to export obtained values during the simulation, and use them in a spreadsheet software. A system of

locks enables thread safe access to the methods implemented without any influence on the simulation performances. In this way, it is easy for the user to define test cases and measure key values in order to evaluate the different dynamic engines.

This architecture has also the advantage of making possible the communication between different dynamic engines. Indeed, in order to communicate, the dynamic engine converts its body data structure into the common data structure, and the common format is then used to set the world configuration of an other dynamic engine. Moreover, the common data format can be exploited by any other part of our application independently from the dynamic engine that generated the data. The positions and states of the bodies are used by our 3D rendering system to draw bodies at the right position, using the corresponding meshes.

3.2. Evaluation of Three Dynamic Engines in our Environment

As a first evaluation, we chose to integrate in our environment three dynamic engines among the most used simulation softwares: Havok physics (`http://www.Havok.com`), NVidia PhysX (`http://developer.nvidia.com`) and as an open source solution, Bullet physics (`http://www.bulletphysics.com/`) .

As detailed further in this section, we defined some test procedures that allows the evaluation of the performance of each dynamic engine both in terms of computation time and accuracy. One should not forget that the tests presented concern only the rigid body simulation feature of each engine, although the engines propose also other features such as character and ragdoll simulation, cloth, deformable bodies or fluids.

Test cases We have selected three test cases to study more precisely the behavior of the different libraries for rigid body interactions. In order to check the accuracy and stability of the simulation algorithms, our first test consists in a stack of 50 cubes initially perfectly aligned (and without extra start margin between the cubes), as illustrated on Figure 1. This classic test is a challenging scenario due to its contact disposition: naive iterative solvers have a very slow convergence rate when propagating the non-penetration constraints [MS01]. Good solvers should be able to maintain the stack standing in a stable fashion. In order to evaluate the friction phenomena, we chose the seven-stages card house discussed in [KSJP08] and represented on Figure 2. This is a structure composed of 89 cards that fully relies on friction: if the friction phenomenon is well reproduced, the card house should not collapse, and the simulation should be stable. Finally, in order to check the scalability of the libraries, we propose a third test consisting in dropping 8000 cubes in a basin (Figure 3). This test puts a lot of objects in a high contact configuration, measuring the behavior of the solvers when many constraints must be respected.

Measured indicators For each of the presented test cases, we measured different indicators to evaluate the computation time of the simulation. In order to measure the dynamic engine performances, we measured the average processing time of the simulation of one time step. A time step includes the time needed for the library to perform one collision detection, the constraints resolution and the time integration. An engine can obtain good performances in average, but some special cases can take a lot of time to solve, leading to undesired lags. To detect such cases, we measured the maximum processing time for one time step. It indicates the longest time the library needs to process a time step during the simulation. Finally, we also observed the end state of the simulation after a fixed simulation time. This is an indirect indication on the accuracy and the stability of the engine for the three tests. To conclude on the stability of the scene, we also made some measurements on the kinetic energy of the bodies of the world, as discussed in Section 3.4.

Simulation parameters For a given world, we selected a set of parameters that influence the outcome of the simulation. We classified the parameters as belonging to the category of either physical parameters or numerical parameters. In the physical parameters category, we retained the *friction coefficient* and the *restitution coefficient*, since friction and bouncing are the two main behaviors expected from a rigid body simulation. In the numerical parameters category, we selected the *integration time step* of the simulation. Each library provides a specific way to define this time step, depending on the integration scheme used, with a significant impact on the performances. Thus, we retained the value of this parameter as an indicator for our simulation.

Each library provides a large set of tunable values that impact the simulation performance (in term of processing time) and/or accuracy. All values relative to the performances were set to their default values. Most of the time, parameters that have an impact on accuracy have also been set to default, although we tried more costly and accurate constraints processing for each library as discussed later in the section. We used discrete collision detection type for all the results presented in this paper. For a given simulation, we kept the friction and restitution coefficients identical between all bodies.

Our tests were performed with an Intel Pentium D (3.40 GHz) using the CPU only, with 2 Go RAM on Windows XP. All measurements were realized with release and fully optimized version of the libraries.

3.3. Tests Results

The test procedure is as following. A combination of friction and restitution coefficient, and all other parameters is fixed. For each dynamic engine, the scene is created and all

counters are set to 0. The simulation is started, and 10 virtual seconds of simulation are performed. We implemented a tool integrated in our environment that can measure the kinetic energy of each body of the world. Thus, measuring the variation of the sum of the kinetic energy of all the bodies composing the virtual world gives an information on the stability of the global world, and allows us to limit our simulation time to 10 seconds. Statistics and visual inspection are also stored. For each test case, we tested at least 5 combinations of friction/restitution coefficient (each of them ranging from 0 to 1) as shown in Table 1. The measured times do not take into account anything else than library collision detection, constraints resolution and time integration. We used a microsecond accurate timer for all our experiments.

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

Table 1: *Combinations of parameters tested for each test case. We measured performance results for all combination of the three test cases for each dynamic engine.*

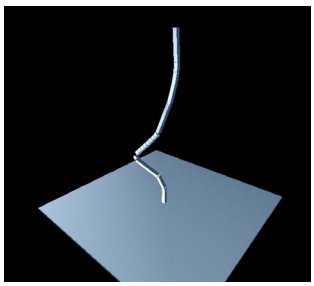

Figure 1: *First test case: stack of cubes.*

Stack of Cubes Figure 1 illustrates the stack of cubes being simulated with Havok physics. Regarding the average processing time, Havok physics always has an advantage over the other libraries, while Bullet physics is always the most time consuming. In most cases, reducing the time step leads to faster computations and a more stable simulation, except for PhysX where decreasing the time step in our range (from 1/100s to 1/60s) did not make the stack of cubes hold longer. Bullet physics obtains the lowest results compared to other dynamic engines, making the stack to hold at best for 4 seconds when using a time step of 1/100s. Note that for PhysX, even if the stack stands a few seconds, there is a visible penetration at the beginning followed by a noticeable reaction making the stack expand and leaving a visible distance between the cubes. On Havok physics, although the stack stands longer, we can notice a small accordion effect on the stack before it collapses (see Figure 1). When we tested a null friction simulation (with a time step of 1/60s). Havok physics achieved the best results making the stack stand for 4 seconds instead of 1.5 seconds and 1 second for PhysX and Bullet physics respectively.

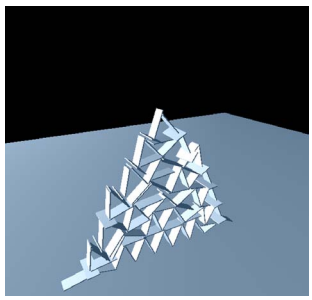

Figure 2: *Second test case: seven-stages card house.*

Card House Every test performed with Havok physics leads to a stable simulation, and gives the best results in terms of computation time and accuracy. The advantageous average simulation time for Havok is mainly due to the fact that since the simulation is quickly stabilized, the objects are frozen and the simulation time falls under 60 microseconds, making the average time decrease. When using full friction (friction coefficient set to 1) and a time step of 1/100s, the simulation is stabilized in less than 2 seconds of simulation. With PhysX, due to an initial inter-penetration and a quite strong expansion reaction, the card house collapses slowly. For the first simulation, 50 seconds are necessary to obtain a quasi-stable state. We can also notice that peak simulation times are greater than Havok's ones by one millisecond. Finally, Bullet physics does not seem to handle static or dry friction properly (see Figure 2). The cards immediately slide and the house is completely broken in less than 2 seconds, whatever the friction coefficient used. Moreover, both peak simulation time and average simulation time are significantly higher than the two other engines.

Cubes in Basin The 8000 cubes falling in a basin test has been performed in order to compare the behavior of the libraries with scenes containing many interacting bodies, as illustrated on Figure 3. When comparing the average times, PhysX and Havok are quite equivalent and provide about 5 time steps per second, while in the frictionless version of the test, Havok showed a peak simulation time of more than 800 milliseconds. Bullet physics is slower and averages 500 milliseconds per simulation step.

3.4. Discussion on the Tests and the Dynamic Engines

Multiple methods of contact/collision resolution Concerning collision detection and resolution, Havok and PhysX define around any body of the world a so-called "shell" that extends the shape of the body of a given distance. This enables to use fast algorithms to solve simple cases.

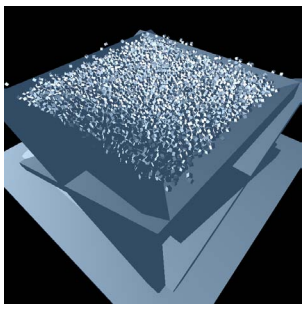

Figure 3: *Third test case: 8000 cubes falling into a basin.*

Actually (with Havok and PhysX), when two bodies collide, if the penetration distance is lower than the thickness of the shell, then a fast algorithm is used to solve for contacts. If the fast method is not sufficient to hold the bodies apart (*e.g.* in the stack of cube example, naive methods fail) then inter-penetration persists and goes through the "shell" of the bodies. Therefore, for those "deep" contact points, a more sophisticated algorithm is used. This is why in some cases, great inter-penetrations occur as highlighted in Figure 4. This manner of solving contact is not annoying on most cases (inter penetrations are brief), but on structure that need a great accuracy to be stable, it reduces the global stability. In our test, we put the shell width to a small value (0.01) so that drawbacks of using a fast resolution are reduced.

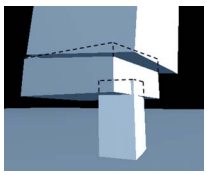

Figure 4: *PhysX inter penetration case with bodies colliding (the top bodies are falling on the bottom one with a small velocity) and restitution coefficient set to 0.*

Summary charts Charts on Figure 5 sums up global performances indicators in term of processing times, while Table 2 sums up global accuracy obtained for each test case.

	Havok physics	NVidia PhysX	Bullet physics
Stack of cubes	++	+	+
Card house	++	0	-

Table 2: *Global accuracy of the dynamic engines on our test cases. ++: Good stability, no visible penetration, the structure holds for a time step of* $1/100$*s. +: The structure breaks even with a small time step. 0: No stable state is reached after 10 seconds of simulation. -: The structure can not be simulated with realism*

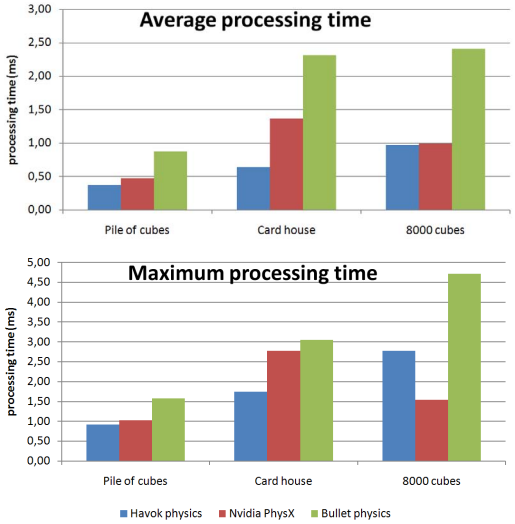

Figure 5: *Mean and maximum computation times of 5 combinations of friction coefficient, restitution coefficient and time step value, for each test case; Time values have been divided by a factor of 200 for the third test.*

Our environment allows us to unify the use of dynamic engines by defining a common interface, providing high level rigid body simulation with a dynamic choice of the simulation engine. We leveraged this environment to perform an evaluation of the leading dynamic engines, that has given us clear ideas on the potential use of those solutions for haptic rendering. However, although good, the performances of the current dynamic engines are not sufficient to perform haptic rendering in scenes containing hundreds of movable rigid body. Thus, we define in the next section our new coupling scheme whose purpose is to enable high frequency haptic rendering in large scenes.

4. A New Coupling Scheme Between Physical Simulation and Haptic Rendering

One of the most common coupling scheme to connect the results of a dynamic engine to control a haptic interface is called *admittance* control. Positions and velocities of the virtual object (or proxy) coupled with the haptic interface are retrieved from the simulation and used to set the state of the haptic interface. The effort brought by the human user on the haptic interface through the haptic controller is then send back to the simulation. This coupling is summed up on Figure 6.

Using this coupling scheme, we performed satisfying 1kHz haptic rendering in modest scenes composed of a small number of object in our environment. As soon as the number of bodies increases in the scene, the simulation frequency decreases, and the haptic feedback is less convincing with

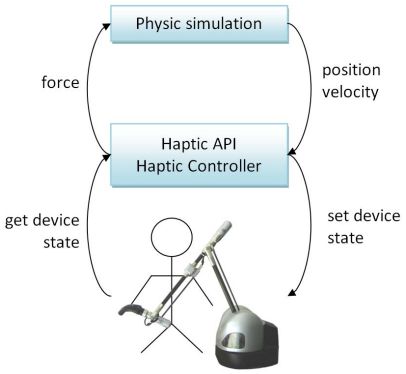

Figure 6: *Description of the admittance coupling.*

the introduction of vibrations. Therefore, we propose in this section a new coupling scheme in order to preserve the quality of the high frequency haptic rendering on complex scenes without relying on the physical simulation frequency. To free the strong dependence to the physical simulation in term of haptic frequency, conceptual dissociation of the software architecture is often considered. Two processes – the physical simulation and the haptic rendering – are living in parallel, and communicate. Figure 7 shows the dissociation we defined in our environment, and used to design our new coupling scheme.

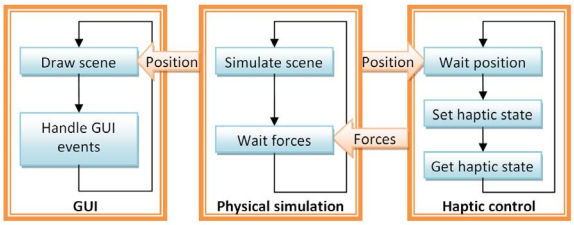

Figure 7: *Three main processes of our dissociated haptic rendering scheme. Each of the three blocks has its own independence.*

The *GUI* process is responsible for handling user events on the Graphic User Interface (GUI), and the display of the scene which is being simulated. The *Physical simulation* process stores the scene and performs collision detection and integration of the simulation over a given time step. Finally, the *Haptic control* process is responsible for giving orders to the haptic API at a constant frequency. The three processes are infinite loops that are free to iterate at different frequencies. Although the three processes are independent, they must communicate and they may synchronize. The GUI process must read the positions of all the objects in the scene in order to display them at the right positions. Between the physical simulation and the haptic control process, the bidirectional exchange of information is more constraining. The physical simulation must know the forces (indirectly or not) returned by the user via the haptic interface in order to integrate the simulation forward. In its side, the haptic control wants to use the position and the velocity of the proxy (the coupled object) to give relevant positions to order the haptic interface.

The main objective is to preserve the quality of the haptic rendering even when the physical simulation process can no longer provide data at the desired frequency. In this case, the haptic control process must create some intermediate data which are consistent with the quality criteria of the haptic rendering.

4.1. The Haptic Physical Sub-world

The main idea of our coupling scheme is to leverage the dissociation presented on Figure 7 creating two linked simulations, with two virtual worlds. The simulation process is responsible for the entire virtual world, while the haptic control process simulates a subset of this world around the proxy, that we named the *haptic sub-world*. Since the haptic sub-world is composed of a reduced number of bodies compared to the whole virtual world, the haptic control process can perform the simulation at a higher frequency. We call the ratio between the simulation world frequency and the haptic sub-world frequency the *simulation ratio*, noted r_{sim}.

In practice, the sub-world is currently defined as the subset of bodies intersecting an axis-aligned bounding box centered on the proxy, called the boundary box. Defining the sub haptic world in this manner enables us to use the fast broadphase detection collision system of the simulation process to generate a list of bodies entering or leaving the haptic sub-world. The size of the side of the boundary box is an important parameter discussed in paragraph 4.5.

The immediate advantage of the haptic sub-world comes from the fact that we can define constraints coming directly from the simulation process at a high frequency controlled by the haptic loop. However, two problematics emerge from this coupling scheme. First, the interface between the two worlds, *i.e.* how the constraints and the energy of the external world are sent into the haptic sub world, has to be precisely defined. The transmission of the energy brought by the haptic sub-world via the proxy in the whole world has also to be well-performed in order to keep a good physical realism. Figures 8.a and 8.b highlight two cases where the information exchange between the two worlds is crucial. Furthermore, since we have two linked worlds running at different frequencies, we have also to define a synchronization process.

4.2. Interfacing the Two Worlds

To avoid unfavorable extreme cases as described on Figure 8.b, we need to add new constraints at the boundaries of the

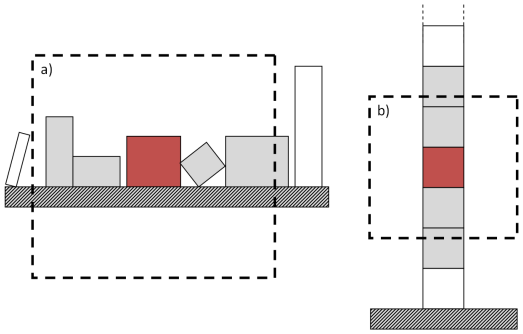

Figure 8: *The sub-world extraction. The red body is the proxy. The thick dashed line centered around the proxy is the boundaries of the haptic sub-world. The grays bodies are in the haptic sub-world, while the white one are not.* ***a****: The bodies of the sub-world are not connected by any contact to bodies of the simulation world.* ***b****: The proxy is at the middle of stacking cubes. If we consider only the sub-world composed of colored bodies, nothing prevents the bodies to fall down due to gravity. This creates a drift between bodies position and velocity of the sub-world, and their equivalent body in the big simulation world.*

haptic world. Let us define more precisely the boundaries of the haptic world. We call a *border contact* a contact point detected between a body that is in the haptic sub-world and a body which is not. We call a *border body* a body which has a border contact (see Figure 9). Note that the sub-world may have no border contacts or bodies.

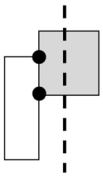

Figure 9: *Definition of the two border contacts (black dots) and their associated border body (in gray). The dashed line represents the sub-world boundaries.*

At each border contact, different parameters values taken from the dynamic engine of the simulation world are stored : the position, the contact normal, the tangent velocity vector, the tangential and normal impulse magnitudes. In order to transmit into the haptic sub-world information about its surrounding simulation world, we apply at each border contact the previously stored normal and tangential impulse on the border bodies (in our implementation, we use impulses but the use of forces instead is straight forward. This approximation of the surrounding world enables to circumvent the problem presented on Figure 8.b, and even to retrieve a mass information of the surrounding simulation world. This last information is very useful when, for example, the mass of a structure involves great friction at its base. Thus, the sub-world can gather only the objects of the base of the structure, as illustrated on Figure 10.

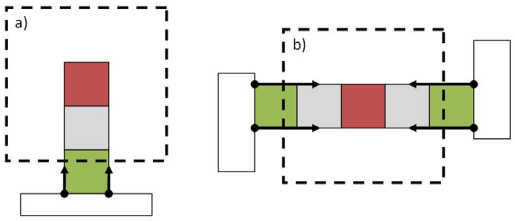

Figure 10: *By applying appropriate retrieved impulses at border contacts, we can have helpful information on the surrounding world* ***(a)****, or conserve compression information of the world, that allows to simulate the same friction or resistance behavior as in the simulation world* ***(b)****. The green bodies (at which border contacts are present) are border bodies.*

4.3. Synchronizing the Two Worlds

We now have a solution to treat boundaries conditions of our haptic sub-world. Thus, the haptic sub-world can run independently at high frequencies. However, a drift with the reference simulation world can be observed, due to the potential change of impulses magnitude at border contact, and most of all, the motion of the proxy. Indeed, the proxy will be impacted by the action of the user, and will add energy to the system. Since the proxy is part of the haptic sub-world, only this world will be impacted. During the synchronization between the two worlds, we must reflect in the simulation world the impacts of the energy added by the user. In the other side, the simulation world must correct the drift taken by the haptic sub-world during its independent time steps. If we impose the state of the haptic world to the simulation world, then we impose the drift of the sub-world to the whole world. This can lead to instability since the drift can enlarge itself, even if the boundaries constraints are correctly updated. On the other side, if we impose the state of the simulation world to the haptic world, we cancel the effects of the proxy, which is not possible for haptic rendering. Therefore, we adopted an intermediate solution which is a trade-off between the two previously presented solutions. For this trade-off, we selected several parameters:

- The ratio between the sum of impulses magnitudes of the contacts applied on proxy with respect to the sum of all magnitudes of all contacts of the sub-world. This ratio (going from 0 to 1) gives an indication on how much the proxy participates to the contact configuration of the sub-world.
- The sum of impulses magnitudes returned by the haptic

device. This value is used to determine whether some energy has been added to the sub-world via the haptic interface. In practice, if this value is under a threshold, we consider that no energy has been added during the haptic cycle.

- The number of border bodies. If there is no border body in the sub-world, then we are free to impose the sub-world state to the simulation state.
- The number of contact points between the proxy and the sub-world bodies. If this number is zero, we are free to impose the simulation world state to the haptic world, and by the way to make any potential drift vanish.

During the synchronization process, we update the state of all bodies of the sub-world, and the state of all their equivalents in the simulation world. After the synchronization, all pair of equivalent bodies must have the same state (same position, orientation, same linear and angular velocities). We introduced a coefficient s of interpolation, that defines on which part of each world will impact the new state of the pair of equivalent bodies. By convention, a value of 0 for s means that the state of the simulation world is prefered. An heuristic algorithm enables to find an appropriate value for s from a combinationo of the four previous listed parameters. Figure 11 sums up our new coupling scheme on a diagram sequence.

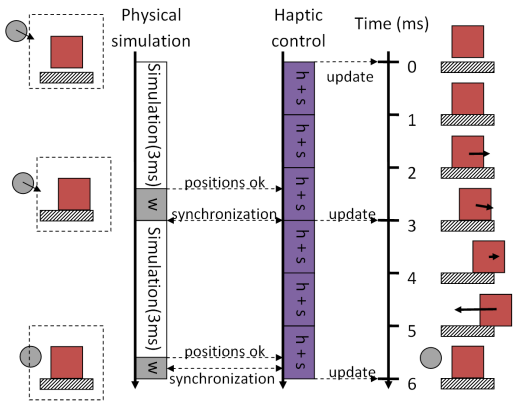

Figure 11: *Sequence of our coupling method, with $r_{sim} = 3$. "h + s" blocks represent haptic sub world simulation and synchronization on physical time. "w" blocks represent waiting times. The simulation world is represented on the left part and the haptic sub-world on the right part. The synchronization is performed in both directions, so that the position of all body common to the two worlds have the same position. The forces given by the haptic device (represented by black arrows in the haptic world) are integrated in the haptic world. At t=6ms, a new ball is entering in the boundary box of the sub-world, and added to the haptic world during the synchronization process.*

4.4. Haptic Sub-World Simulation Processing Time

The haptic frequency is directly linked to the haptic sub-world simulation time, since the proxy state is updated after each simulation step. The simulation time, including processing time for collision detection, constraints resolution and time integration can be linked to the number of bodies composing the world, the complexity of their geometric shape, and the number of contact (representing constraints) between the bodies. We performed some measurements to know the influence of the number of bodies and the number of contact point on the collision detection time and constraint solving time. Those experiments are summed up in Figure 12. We can see that detection collision time and constraint solving time are nearly linear to both the number of bodies and the number of contacts. We deduced from a linear identification that it takes in average 0.0016 ms to solve for one body, in a spaced configuration of cubes, against about 0.011 ms per body in a dense configuration, with a high number of contacts. The latest number enables us to determine a maximum number of bodies allowed to live in the haptic sub-world to be sure to respect the haptic frequency imposed.

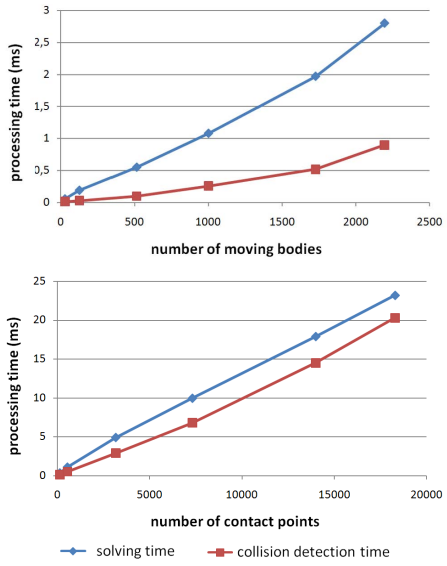

Figure 12: *Influence of number of bodies and number of contact points on the detection collision time and constraint solving time. The first chart shows measurements with spaced cubes without contact, while the second chart is made from a scene composed of cubes in high contact configuration. The tests were performed with Havok physics.*

4.5. Size of the Boundary Box and Sub-World Definition

The size of the boundary box has a direct impact on the size of the sub-world and on the potential number of bodies com-

posing it. The greater the size of the box, the greater the number of bodies potentially intersecting with it. Computation time needed to simulate the sub-world and to synchronize the two worlds depends also on the boundary box size.

In our tests, we determined a size value greater than the possible displacement of the proxy during the haptic cycle. This value can be obtained through the maximum velocity of the proxy. Actually, we experimented and deduced from our algorithm that the size of the boundary box should be big enough to avoid border bodies directly in contact with the proxy. Even if it will not compromise the haptic frequency or stability, there will be inconsistencies in the efforts needed to move border bodies. Bodies at the extremities have namely constraints coming from the simulation world, that persist until the next synchronization, reducing the reactivity when efforts are applied on them. However, if all the bodies that have comparable sizes have also comparable masses, it is easily possible to find good trade-offs for the size of the sub-world, so that processing time and synchronization time are small enough. Simulation results are illustrated on Figures 13 and 14).

4.6. Simulation Ratio Value

An other important aspect of our method concerns the determination of our simulation ratio r_{sim}. As shown on Figure 11, the simulation process uses a time step which is r_{sim} times greater than the time step used in the haptic control process.

Increasing the simulation ratio reduces the frequency of the synchronizations, and thus increases the independence of the haptic process. It also unfortunately increases the drift taken in the haptic sub-world. We also experienced another limiting constraint. Indeed, since a higher time step is used in the simulation world, the accuracy of its simulation can be reduced. This can be annoying in extreme cases. Consider the example where the time step used in the simulation process allows small penetration between stacking objects, the reduced time step of the haptic sub-world simulation will not allow it, producing very small vibrations along the synchronizations between the two worlds. Those small vibrations can move the bodies making stacking structures to fall down. This phenomenon is due to small penetrations allowed by the simulation softwares. A solution to avoid it is the use of more strict constraint solvers that forbid inter-penetrations.

A value of 4 for r_{sim} was reasonable in most cases where stacking structures are present, with a lot of border bodies in contact. However, where the virtual world is sparser and does not contain challenging structures such as a stack of cubes, a simulation ratio up to 10 can be used (if there is no border body, the limit of the simulation ratio is given by the larger time step that can be used for the simulation and ensures stability).

4.7. Results

Using the double simulation method, we performed satisfying 1 kHz haptic rendering on scenes containing about 250 moving cubes in high contact configuration. The compression information indirectly coming from impulses applied at border bodies enables to successfully take into account the contact constraints and compression information as shown on Figure 13.

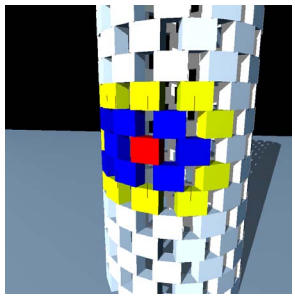

Figure 13: *Illustration of the double simulation method. The red body is the proxy. The blue and yellow bodies are belonging to the haptic sub-world. The yellow bodies are border bodies, while the red line drawn from their center represents the linear impulses applied on them during the haptic cycle. The simulation ratio value is 4.*

We also successfully performed tests that exploit the drift taken by the haptic sub-world, such as pushing a row of aligned cubes resting on a floor with friction, with only a part of the row belonging to the haptic sub-world (see Figure 14). We felt the mass of all the cubes of the row, and the entire row begins to slide if enough energy is added. We validated the dynamic update and propagation of the energy of the haptic sub-world on a heap of 250 cubes, all in contact, and on other scenes such as the stack of cubes or the card house presented in Section 3.

Finally, we were able to perform 2 kHz haptic rendering using a simulation ratio of 10, on a scene containing 150 cubes and approximatively 620 contact points. In this case, we measured that the bottleneck of the process is the synchronization time. Indeed, the synchronization is not an instantaneous operation. In practice, our synchronization method is linear in the number of bodies composing the haptic sub-world. To maintain the haptic frequency, the addition of the synchronization time and the simulation time of the haptic sub-world has to be lower than the inverse of the fixed frequency.

5. Conclusion and Perspectives

In this paper, we presented an environment integrating the leading physical simulation softwares, allowing their evaluation with respect to different haptic rendering criteria. The environment provides both an abstraction from potentially

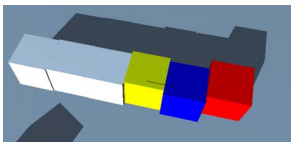

Figure 14: *Pushing a row of aligned cubes: our method can make the user feel the sum of friction of all the aligned cubes in the row, although only 3 of them belong to the haptic sub-world. The simulation ratio is 4.*

complex underlying physical simulation methods and the flexibility to use different physical engines according to the physical environment requirements. Inside our environment, we have implemented a new coupling scheme between the physical simulation and the haptic rendering. The scheme is independent from the rigid body dynamic simulation methods and allows to increase the haptic simulation frequency by a factor of 4 in very dense scenes, and a factor of at least 10 in sparse scenes, while keeping nearly the same haptic rendering quality. The haptic sub-world principle consists in extracting a subset of the bodies composing the virtual world, and in simulating the constantly updated sub-world at a higher frequency, and potentially with the same methods of simulation than the whole virtual world.

We have determined the maximum number of bodies composing the haptic sub-world so that the haptic frequency requirements are respected. Although the parameters of our method are currently set before the simulation,they could be adapted during the simulation. Depending on the measured refreshment rates, it is possible to dynamically change the maximum number of bodies composing the haptic sub-world. An alternative solution to the currently used boundary box could also be tested for haptic sub-world selection, such as the use of the contact graph of the world starting from the proxy and stopping at a maximum depth. In very large scenes, bodies are usually spaced, and we aim at dynamically adapting the simulation ratio so that our method can leverage sparse scenes. Finally, future work will deal with the use of our new coupling scheme in order to define potentially collaborative interactions in large worlds in which we could define *n* haptic sub-worlds to perform *n* haptic rendering in parallel.

References

[AKO95] ADACHI Y., KUMANO T., OGINO K.: Intermediate representation for stiff virtual objects. In *Proceedings of the Virtual Reality Annual International Symposium* (1995), pp. 203–210.

[Bal99] BALANIUK R.: Using fast local modelling to buffer haptic data. In *Proceedings of Fourth PHANTOM Users Group Workshop-PUG99* (1999).

[BJ07] BARBIČ J., JAMES D. L.: Time-critical distributed contact for 6-dof haptic rendering of adaptively sampled reduced deformable models. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer animation* (2007), vol. 2, pp. 171–180.

[CC97] CHANG B., COLGATE J.: Real-time impulse-based simulation of rigid body systems for haptic display. In *Proceedings of the ASME Interational Mechanical Engineering Congress and Exhibition* (1997), pp. 1–8.

[CSB95] COLGATE J. E., STANLEY M., BROWN J. M.: Issues in the haptic display of tool use. *IEEE/RSJ International Conference on Intelligent Robots and Systems 3* (1995), 3140.

[CSC05] CONSTANTINESCU D., SALCUDEAN S., CROFT E.: Haptic rendering of rigid contacts using impulsive and penalty forces. *IEEE Transactions on Robotics 21*, 3 (2005), 309–323.

[Erl07] ERLEBEN K.: Velocity-based shock propagation for multibody dynamics animation. *ACM Transactions on Graphics 26*, 2 (2007), 1–20.

[GBF03] GUENDELMAN E., BRIDSON R., FEDKIW R.: Nonconvex rigid bodies with stacking. In *Proceedings of SIGGRAPH* (2003), pp. 871–878.

[GLGT99] GREGORY A., LIN M. C., GOTTSCHALK S., TAYLOR R.: A framework for fast and accurate collision detection for haptic interaction. In *Proceedings of IEEE Virtual Reality Conference* (1999), pp. 38–45.

[Hah88] HAHN J. K.: Realistic animation of rigid bodies. In *Proceedings of SIGGRAPH* (1988), pp. 299–308.

[KEP05] KAUFMAN D. M., EDMUNDS T., PAI D. K.: Fast frictional dynamics for rigid bodies. In *Proceedings of SIGGRAPH* (2005), pp. 946–956.

[KSJP08] KAUFMAN D. M., SUEDA S., JAMES D. L., PAI D. K.: Staggered projections for frictional contact in multibody systems. In *Proceedings of SIGGRAPH Asia* (2008), pp. 164:1–164:11.

[LBFD05] LUCIANO C., BANERJEE P., FLOREA L., DAWE G.: Design of the immersivetouch: a high-performance haptic augmented virtual reality system. *Human Computer International Proceedings* (2005).

[MS01] MILENKOVIC V. J., SCHMIDL H.: Optimization-based animation. In *Proceedings of SIGGRAPH* (2001), pp. 37–46.

[MSJT08] MÜLLER M., STAM J., JAMES D., THÜREY N.: *Real Time Physics*. ACM Siggraph, 2008, ch. Courses.

[MW88] MOORE M., WILHELMS J.: Collision detection and response for computer animation. *Proceedings of SIGGRAPH* (1988), 289–298.

[OL05] OTADUY M., LIN M.: Sensation preserving simplification for haptic rendering. In *Proceedings of SIGGRAPH* (2005), pp. 543–553.

[OL06] OTADUY M., LIN M.: A modular haptic rendering algorithm for stable and transparent 6-dof manipulation. *IEEE Transactions on Robotics 22*, 4 (2006), 751–762.

[PK04] POCHEVILLE A., KHEDDAR A.: I-touch: a framework for computer haptics. In *Proceedings of the International Conference on Intelligent Robots and System* (2004).

[RK00] RUSPINI D., KHATIB O.: A framework for multi-contact multi-body dynamic simulation and haptic display. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems* (2000), vol. 2.

[RMB*08] RUFFALDI E., MORRIS D., BARBAGLI F., SALISBURY K., BERGAMASCO M.: Voxel-based haptic rendering using implicit sphere trees. In *Proceedings of Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems* (2008), pp. 319–325.