Fast audio-haptic prototyping with mass-interaction physics

James Leonard, Jerome Villeneuve

Fast audio-haptic prototyping with mass-interaction physics

James Leonard*[1] and Jérôme Villeneuve†[1]

[1]*Univ. Grenoble Alpes, CNRS, Grenoble INP*, GIPSA-LAB, 38000 Grenoble, France,*
** Institute of Engineering Univ. Grenoble Alpes*

This paper presents ongoing work on the topic of physical modelling and force-feedback interaction. Specifically, it proposes a framework for rapidly prototyping virtual objects and scenes by means of mass-interaction models, and coupling the user and these objects via an affordable multi-DoF haptic device. The modelled objects can be computed at the rate of the haptic loop, but can also operate at a higher audio-rate, producing sound. The open-source design and overall simplicity of the proposed system makes it an interesting solution for introducing both physical simulations and force-feedback interaction, and also for applications in artistic creation. This first implementation prefigures current work conducted on the development of modular open-source mass-interaction physics tools for the design of haptic and multisensory applications.

Introduction

In recent years, the access to digital fabrication technologies coupled with the surge of *do-it-yourself* electronics and a drive from the Virtual Reality industry have given rise to a new interest in haptics: once considered only as costly lab equipment reserved to specific academic fields, a new generation of force-feedback and vibrotactile devices now offer affordable open designs, that can be built, customised or tampered with, and used by individuals for a vast range of purposes including musical & artistic creation [1, 12].

Prototyping haptic interactions generally involves creating and computing a physical scene that the user will interact with through the device. However, tools for designing the virtual scenes (such as CHAI3D[1]) are often difficult to apprehend. We argue that following the increased accessibility of haptic solutions, equally accessible tools should allow for simple and fast prototyping of modular virtual scenes, applicable to any number of purposes, including haptic-audio multi-modal VR.

This paper proposes such a framework using a new open-source mass-interaction physical modelling engine and an existing open-source / open-hardware haptic device. First, we briefly introduce the context of force-feedback interaction technologies and physical modelling concepts for designing multisensory virtual scenes. Our prototyping framework is then presented and illustrated through a series of examples. Finally, we will offer perspectives for future work.

Natural Interaction with Digital Realities

Throughout the history of information technologies, advances in computation methods and process have always evolved in tight relationship with technologies allowing humans to interact with the machine. In recent years, global trends in HCI have shifted from input peripherals (such as the keyboard and mouse), towards *natural interaction* through speech recognition, motion capture and gesture recognition, and virtual reality technologies.

Blending digital realities with the real physical world is now commonplace, and to this end technologies have evolved significantly both in performances and affordability, driven essentially by the video-game and entertainment industry. However, focus has primarily been on visual aspects (CGI animation, etc.), sometimes sound (through immersive 3D audio techniques), but rarely on the sense of touch or physical presence inside a virtual scene, using haptic (or force-feedback) technologies.

Indeed, the sense of touch relies on a coupled action-perception loop [11], distributed across the entire human body. Hence, when developing or using haptic systems capable of "rendering" this interaction with a virtual entity, one is faced with two main technological challenges:

- Simulating the immediacy of the action-perception loop: in a force-feedback chain the computer receives position data from a sensor, computes interaction forces based on a virtual scene, and sends these forces data back to mechanical transducers[2]. All of these operations take time, and hardware/software architectures must allow a sufficiently fast loop to convey the impression of immediate action-reaction, and also for stability reasons [14, 10].

- Sufficient degrees of freedom: very few human gestures can be reduced to a small number of degrees of freedom, and applied in only one or a few points. However, designing haptic systems with a large number of DoF and/or contact points poses many mechanical challenges. Various systems have been proposed, from pen-like interfaces [20], to exoskeletons [13] or haptic gloves [2], but in each case they are generally specifically tailored to a specific type of interaction.

Developments and applications of haptic devices for artistic creation can be traced back to the late 1970s [8]. Since then, a steadily growing scientific community has formed around the use of force-feedback and vibrotactile technologies applied to the fields of music and digital arts [22].

Open-hardware haptic systems

Due to their mechanical complexity and often dedicated hardware/software, haptic systems are generally considered costly. Among others, we can cite the Phantom system [20], or the TGR from ACROE [9]. The gaming industry has led to some cheaper alternatives, such as the Novint Falcon [19], at the cost of reduced performance. What's more, the rapid evolution of computer systems and peripherals has rendered the software and hardware compatibility of proprietary/commercial solutions increasingly problematic as drivers are not always maintained for recent operating systems.

*james.leonard@gipsa-lab.fr
†jerome.villeneuve@gipsa-lab.fr

[1] http://www.chai3d.org

[2] In the case of impedance-based haptic devices.

Partially in response to these issues, several open-hardware systems have been proposed, such as the simple, low-tech haptic systems designed by Bill Verplank [25, 24]. More recently, the rise of digital fabrication technologies and open-electronics have given rise to new, affordable and open-source & hardware haptic devices, such as Edgar Berdahl & A. Kontogeorgokapopoulos' FireFader [1], or the Haply[3] system [7]. These systems are cheaper to build and repair, use simple communication protocols, and minimise or entirely circumvent the use of any proprietary software.

We decided to use the Haply system for our work. Similarly to the FireFader, it relies on an Arduino-based board (the Haply M0) to process sensor data and drive motors, and communicates over USB (through a virtual serial port) with a simulation context running on a host machine. However, the FireFader offers one or several 1 DoF sliders that are connected to one-dimensional sound synthesis simulations, whereas the Haply is a 2 DoF (cf. Figure 1) system that can be extended to 4, allowing exploration of more complex interaction modalities with multidimensional virtual environments. The open-source hAPI programming interface allows to easily configure the haptic device and interact with a virtual scene.

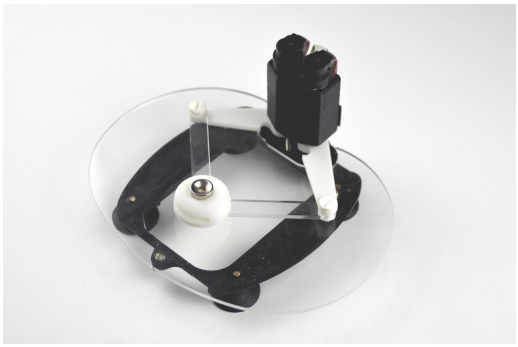

Figure 1: The 2-DoF Haply force-feedback device

Designing multimodal virtual scenes

A question of equal importance to the force-feedback peripheral itself concerns how a multimodal (visual, audio and haptic) virtual scene can be designed, computed and "rendered" for the user to see, hear and feel. We categorise two main approaches, shown in Figure 2:

Distributed Approach Complex virtual scenes are often distributed into separate computational processes for each modality [23]. Visual rendering is handled at a relatively low rate (50-100 Hz, latency of up to 40ms tolerated), audio is processed at a high-rate (44.1 kHz, with latency under 10ms), and physics are generally computed around 1kHz, with critical latency conditions for the haptic loop (1ms or less). This scheme is especially adapted in cases where audio or visual processes may rely on abstract (non-physical) algorithms. However, it does pose the problem of defining mapping and control relationships between the processes. If the correlation between different modalities is not sufficiently explicit, the sensation of believability and presence of the virtual objects may suffer.

Single Model Approach An alternative way of designing multimodal virtual scenes is to model them with a single formalism (cf. Figure 2b). Physical modelling allows creating scenes of deformable objects that exhibit visual, mechanical and acoustic behaviour. The object that we touch is the one we see, and that we hear. This approach is sometimes referred to as *multisensory*, and guarantees coherence between the different modalities. Works such as [16, 1] use this approach with mass-interaction physical modelling, a formalism presented below.

Mass-Interaction Physical Modelling

Representing mechanical systems by means of punctual masses linked together by elements such as springs or dampers, and submitted to various constraints, is one of the most common ways to describe and calculate their behaviour. From Newton's laws we know the equation of movement of a mass in a given referential; the action of springs, dampers and other elements can be mathematically described or approximated by well known formulas. By resolving the equation system composed of the equations of each element in a mechanical construction, we obtain the global behaviour.

Mass-interaction physical modelling and simulation [4] relies on exactly this principle: the inertial behaviours of material elements and interactions (springs, dampers, etc.) are described by simple discrete-time difference equations, following a given discretisation scheme (see [3, 15] for algorithms and implementation details). Positions and forces can be expressed as scalar values (for 1D systems) or as 2D or 3D vectors according to the spatial attributes of the scene.

Mechanical constructions are then built by assembling masses and interactions together in a network, setting physical parameters and initial conditions, and then computing the behaviour over time. Figure 3 shows a topological representation of a mass-interaction model.

Owing to their inherent simplicity and efficient computation, lumped methods such as mass-interaction physics have been widely used and studied in the field of haptics, for the design of virtual deformable matter and haptic interaction models [6, 18, 10], including for direct force-feedback interaction with virtual musical instruments [17, 1]. However, we notice a relative lack of open and accessible frameworks or tools allowing for modular and unified design of multisensory virtual objects and scenes, that can be considered for their visual, acoustical, haptic behaviour, or indeed any combination of the three.

In the following section, we present our contribution to such a framework in the form a mass-interaction physics engine that can be coupled with an open-hardware force-feedback device (in our case, the Haply) in order to create any kind of haptic or audio-haptic virtual scene.

A Physics-based Haptic Prototyping Framework

Our prototyping framework is based on Processing[4], an open-source Java based programming environment tailored for sketching 2D and 3D animated scenes. It is currently one of the most popular tools used to introduce and teach interactive and visual programming, while remaining capable of complex rendering, communication with hardware, and even physical modelling[5]. What's more, hAPI, the Haply programming interface, is written in Java and is directly fitted for running within Processing sketches. Although, it is not an *audio* programming environment as such, external libraries can be used to add real-time sound synthesis functionality.

A mass-interaction physics engine in Java

miPhysics is a new open-source modular mass-interaction physics engine prototype, recently developed by the authors. As opposed to existing physical modelling libraries within Processing, miPhysics allows designing 3D physical objects through modular networks of masses and interactions, easily integrating any type of user interaction (for manipulation, parameter modification, dynamic topology changes, etc.), and building any type of visualisation process on top of the created model.

[3] http://www.haply.co

[4] https://processing.org/

[5] https://github.com/diwi/PixelFlow

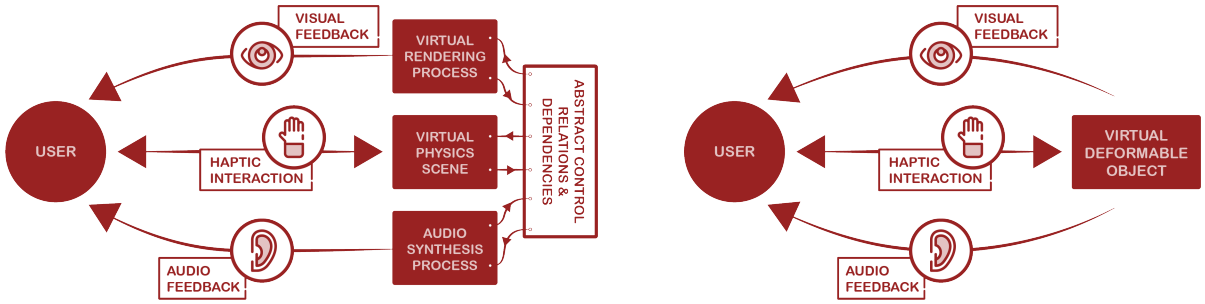

Figure 2: a) Distributed approach to multi-modal virtual scenes (left), b) single model approach (right).

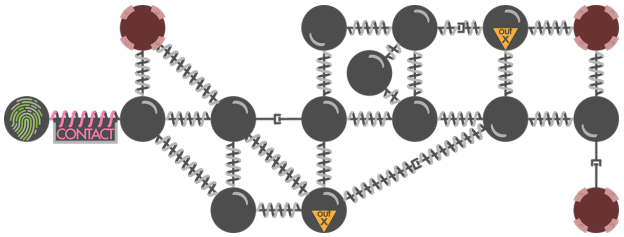

Figure 3: Representation of a mass-interaction model, composed of different types of mass and interaction elements.

As in any mass-interaction system, the main types of physical elements proposed by MIPHYSICS are:

- **Mass-type modules**: Three-dimensional masses, oscillators and fixed-Points, as well as 2D masses (constrained on the *z* plane) or 1D masses (that only vibrate along the *z* plane).
- **Interaction-type modules**: Three-dimensional interactions such as springs, dampers, contacts, enclosing bubbles, etc. For specific use cases, 1D springs have also been implemented.

One of the main design goals for MIPHYSICS is simplicity of use and direct access to each element of a physical model, including during run-time. Each mass or interaction is labelled with a specific name or identifier that can be used to connect the element to others, to read its state or change its physical parameters during the simulation. Meta-categories of physical elements can be created to allow for grouped parameter modifications. In addition to specific physical parameters, global "air friction" and gravity direction/force of the entire virtual scene can be configured and modified. Figure 4 shows the general code structure when creating a physical model.

INTEGRATION OF THE HAPTIC DEVICE

From the point of view of the physical simulation, the haptic device is represented by an *avatar*: a mass-type module, called *HapticInput3D*. Instead of calculating a new position based on its previous states and the forces applied to it (as a regular mass would), this module sends its force signal out to the real haptic device's transducers and sets its position from the haptic device's sensor data.

We use the existing hAPI functions to transform the encoder values of the two motors into a 2D position by calculating the kinematics of the linkage arms, and reciprocally to translate forces applied in a 2D space to torque values applied to the motors. Therefore, the end-user has only to consider the device's end-effector in a 2D plane and connect it directly to the simulation.

```
/*  global variable */
PhysicalModel mdl;
simRate = 300;
dispRate = 60;

void setup() {

  /* ...general setup code... */

  mdl = new PhysicalModel(simRate, dispRate, paramSystem.ALGO_UNITS);

  /* Create a mass, connected to fixed points via Spring Dampers */
  mdl.addMass3D("mass", m, new Vect3D(0., 0., 0.), new Vect3D(0., 0., 1.));

  mdl.addGround3D("ground1", new Vect3D(dist, 0., 0.));
  mdl.addGround3D("ground2", new Vect3D(-dist, 0., 0.));

  /* ...add other material elements... */

  mdl.addSpringDamper3D("spring1", 1., 0.1, 0.01, "mass", "ground1");
  mdl.addSpringDamper3D("spring2", 1., 0.15, 0.01, "mass", "ground2");

  /* ...add other interactions... */

  mdl.init();
}

void draw() {

  /* Calculate Physics */
  mdl.draw_physics();

  /* Get the position of the "mass" module */
  PVector pos = mdl.getMatPVector("mass");

  /* ... Draw the scene, the mass and the springs ... */
}
```

Figure 4: Creating a mass-interaction physical model with MIPHYSICS.

The 8-bit resolution of the encoders can result in strong quantisation of the position data and fairly "jumpy" physical behaviour. We chose to feed the raw position value into an *Exponential Weighted Moving Average* filter with a high weighting decrease. As such the filter adds latency, theoretically impacting haptic stability. However, given the low resolution position data, the trade-off of smoother data versus the small increase in latency leads to no observable increase of instability.

Time has not yet allowed for complete real/virtual calibration of the haptic device. For now, the $real \rightarrow virtual$ position gain $alpha$ and the $virtual \rightarrow real$ force gain $beta$ (cf. Figure 5) are empirically defined by the user depending on the mechanical impedance of the virtual scene/objects.

END USER ENVIRONMENT: PROCESSING

Haptic Simulation Framework In the case of non-audio haptic simulations, the PROCESSING sketch is organised as shown in Figure 5: the haptic input/output communication with the Haply Board, the kinematics and the physical computation all run in a dedicated thread running at a fixed-rate, generally 1 kHz. The *draw()* method renders the model by observing its state at a lower display rate. The rendering process can be generic (visualise all mass-type and interaction-type elements according to pre-defined drawing routines) or custom-coded by the user.

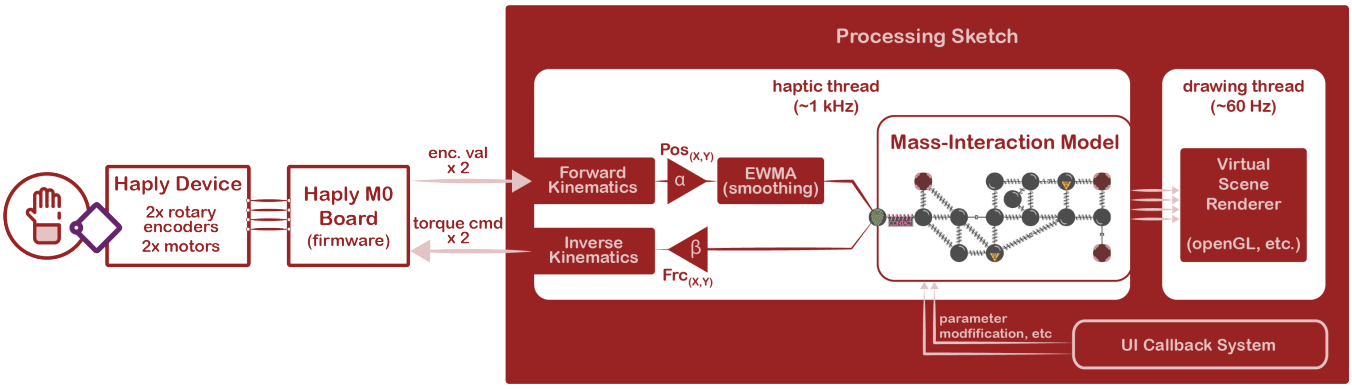

Figure 5: Full user-model interaction chain for non-audio haptic simulations. The kinematic calculations, physical computation and position/force I/O are all performed in a haptic thread, running at 1 kHz. The model is rendered at a lower rate in the draw() method.

Portability and ease of use is an important consideration. The sketches run on Mac, Linux and Windows, the only noticeable difference measured being the scheduling performance of the haptic thread: early tests show a higher dropout rate (missed steps in the haptic-loop) on a Windows system, correlated with system load; however, on a standard Linux OS the scheduling is more robust, and very few steps are missed. Performance in these conditions is satisfactory, meaning there is no need for a dedicated Real-Time OS, sometimes used in high-performance synchronous systems [17].

Real-time Audio-Haptic Simulation Framework Handling real-time audio streams in Processing can be done using the MINIM library[6], which implements the common *Unit Generator* paradigm, found in environments such as Super Collider [21]. A PHYUGEN class has been created, extending the MINIM UGEN template. It contains the mass interaction model (built and initialized in the class constructor), and computes steps whenever needed (by overriding the *uGenerate()* method).

The system is slightly more complex here:

- The haptic thread runs at 1 kHz, getting the haptic I/O and performing the kinematic transformations. At each step, it sets a new position value from incoming sensor data, and discharges the force from the model to the haptic device (sending it to the motors).
- The sound synthesis UGEN runs at the audio rate (44.1 kHz), providing new audio data by computing steps of the physical model whenever needed through a callback API. The haptic *avatar* module upsamples and smoothes raw position data from the haptic device, using by the *EWMA* filter. Symmetrically, force applied by the model to the *avatar* is accumulated into a buffer until it is consumed by the haptic thread (see [17] for details regarding multirate audio mass-interaction systems with haptics).
- The visualisation thread (*draw()* method) observes the model positions and creates the visual representation.

A lock synchronisation system ensures mutual exclusion in the critical sections of the haptic, sound synthesis and rendering threads, thus avoiding temporary glitches & errors that can be induced by concurrent memory access (e.g. reading the model state in the display thread while it is being updated by the physics thread).

[6] http://code.compartmental.net/tools/minim/

RESULTS & EXAMPLES

A number of Haply-specific examples can be found the in the example section of the MIPHYSICS github repository[7], as well as ready-to-use templates in which the user has only to add his specific model code and visualisation specificities. Below, we present four simple cases:

HAPTIC EXAMPLE 1: HEAVY MASS CHAIN

This simple example attaches a chain of 3D masses to the haptic device (which moves on the *x-y* plane). The springs are fairly loose, but the combined weight of the virtual masses is enough to add significant inertial behaviour to the system: when dragging the chain, one can feel the effort needed to set them in motion, and must work with/against their momentum when stopping or changing direction.

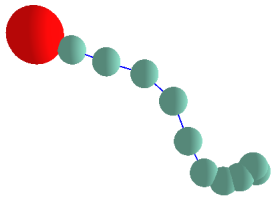

Figure 6: Chain of masses (green) attached to the Haply system (red) by springs.

HAPTIC EXAMPLE 2: ENCLOSED MARBLES

In this model, two-hundred 3D masses are enclosed within a sphere (using *bubble* interactions) and above a flat surface (using *planeContact* modules). Each mass has contact interactions set up with all other masses, as well as with the haptic input module. Gravity and air friction are configured so that the masses fall down onto the plane, and eventually stop moving. By "rummaging" around with the haptic device, the user interacts simultaneously with many masses, sending them bouncing up against the edges, giving a global "stirring" motion, and so forth. Dropping the haptic thread rate to 200 Hz allows to increase the number of masses to 600, at the cost of reduced bandwidth (less sharp dynamics) in the physics computation.

[7] https://github.com/mi-creative/miPhysics_Processing

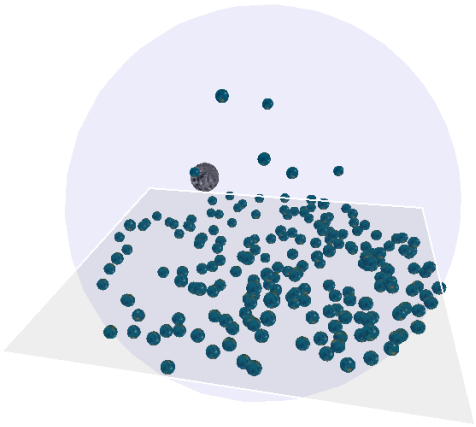

Figure 7: The "Enclosed Marbles" example sketch.

Audio-Haptic Example 1: Plucked String

Using the real-time audio architecture described in the previous section, this model illustrates the simple case of a plucked string. The string itself is composed of masses constrained to move along the *x-y* plane (keeping a constant *z* value), connected by damped 3D springs, and ended with two fixed points.

Contact interactions link the haptic device to each mass of the string, with a given sphere of action (analogous to the "width" of the plectrum), so that the user can pluck the string in any part (in the middle, near the bridge, etc.) and also excite and rapidly dampen the string in specific areas to obtain palm-mute or natural harmonic effects. A similar example combines multiple strings placed below each other, so that they can be played one at a time or strummed together.

Figure 8: 2D Plucked String Example. The haptic mass (red) is pressed against the string.

It can be noted that in the case of 2D strings, the resting length of the springs relative to the elongation imposed by the extremities is a key factor in the physical behaviour, in particular regarding non-linear tension effects [5] such as the "twang" or pitch glide at high excitation levels. This will be illustrated in the last example.

Audio-Haptic Example 2: The Recoil String

This model explores the non-linearities of 2D strings by pushing the situation to the extreme: the string is still attached to a fixed point at one end but the other end it a very heavy mass, set with an initial velocity. This mass slowly stretches the string until it recoils, then loosening up until it is completely limp, before stretching again. What's more, by plucking or exerting pressure on the string, the user can influence the course of the heavy mass, resulting in bizarre "worm-like" behaviour, the string sometimes wrapping almost entirely around the haptic device (as seen in Figure 9.

These four examples serve as a small introduction to what is possible with the platform. The modular design of the physics and the haptic interaction allow creating all kinds of virtual objects and interaction scenarios, the only real limit being the computational complexity that can fit inside a haptic or audio frame. Preliminary results (on a fairly worn-down laptop running Debian 7) show a maximal attainable model complexity of approximately ten thousand 3D elements at 1 kHz for haptics-only sketches, and four hundred elements for audio-rate sketches.

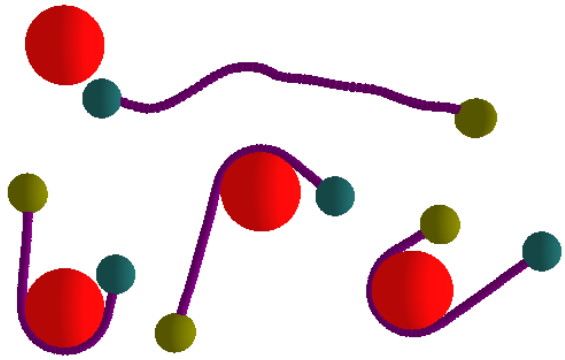

Figure 9: Various screen shots of the "recoil" string. The green point is fixed, whereas the yellow one is a very heavy mobile mass, causing the string to tighten/loosen, wrap around the Haply device, etc.

Perspectives & Future Work

This work offers several possible axes for improvement:

1. Performance, stability and robustness of the haptic device: the Haply's 3D printed plastic parts lack rigidity and present some play between mechanical elements (especially concerning the linkage arm pivots). This could be addressed by making sturdier aluminium parts based on the provided mechanical schematics. The motor performance also presents some limits for small amounts of force feedback (it takes a certain threshold of command torque for the motor to actually move). For these reasons, implementing stiff spring interactions directly between the device and a virtual point is problematic, as unstable oscillations tend to occur.

2. Calibration of the real/virtual chain: through mechanical characterisation of the device and dedicated calibration models, it should be possible to set exactly how a mass of 1 gram in the virtual model is projected into the real world, and reciprocally, helping to standardise the parametrisation of virtual models and make them easier to port to & from other haptic devices.

3. Optimisation of the simulation engine: this work is a prototype developed entirely in Java and has prioritised object-oriented concepts and global intelligibility over lean, speed-oriented processing. Switching the core engine to a more powerful language such as C++ while keeping user-friendly model building functions is an interesting perspective for future work.

The authors feel that the presented system constitutes an interesting base for teaching students about modular physical modelling and force-feedback interaction : the overall framework is very straightforward and concise, placing emphasis on the concepts of mass-interaction physical modelling, the specific time constraints of haptic feedback loops and real-time audio computation. The entire system is open-source, multi-platform and very affordable. As with any newly released open-source tool, the next step is to measure its ease-of-use and robustness in a variety of applications and contexts, hopefully developing a community of users and contributors over time.

Conclusion

In this paper, we have presented a new open-source & open-hardware framework for prototyping audio-haptic interaction with mass-interaction models. Based on the Haply device, a new physics engine prototype, and the Processing sketching environment, users can easily create scenes composed of virtual objects and interact with them. Two scenarios allow the physical computation to either be calculated in sync with the haptic data acquisition & computation loop, or in a higher rate thread in the form of a physics-based Unit Generator for real time audio.

A first batch of examples shows promising results. The 2D haptic device offers new interactions with virtual acoustical objects; augmenting the current Haply device to 3 or 4 DoF should open further perspectives. The simplicity and affordability of the proposed system makes it an ideal candidate for teaching students about force-feedback interaction and mass-interaction physical modelling. More generally, is serves as a general framework for modular, open haptic interaction design. We aim to host a series of introductory workshops in the coming months.

References

[1] E. Berdahl and A. Kontogeorgakopoulos. The firefader: Simple, open-source, and reconfigurable haptic force feedback for musicians. *Computer Music Journal*, 37(1):23–34, 2013.

[2] J. Blake and H. B. Gurocak. Haptic glove with mr brakes for virtual reality. *IEEE/ASME Transactions On Mechatronics*, 14(5):606–615, 2009.

[3] C. Cadoz and J.-L. Florens. The physical model : Modeling and simulating the instrumental universe. In *Representations of Musical Signals*, pages 227–268. MIT Press, 1991.

[4] C. Cadoz, A. Luciani, and J. L. Florens. Cordis-anima: a modeling and simulation system for sound and image synthesis: the general formalism. *Computer music journal*, 17(1):19–29, 1993.

[5] N. Castagné and C. Cadoz. Physical modeling synthesis: balance between realism and computing speed. In *Proceedings of the COST G-6 conference on the digital audio effects (DAFX-00)*, page 6, 2000.

[6] M. C. Cavusoglu and F. Tendick. Multirate simulation for high fidelity haptic interaction with deformable objects in virtual environments. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 3, pages 2458–2465. IEEE, 2000.

[7] S. Ding and C. Gallacher. The haply development platform: A modular and open-sourced entry level haptic toolset. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, page D309. ACM, 2018.

[8] J.-L. Florens. *Coupleur Gestuel Retroactif pour la Commande et le Controle de Sons Synthetises en Temps-Reel*. PhD thesis, Institut National Polytechnique de Grenoble, 1978.

[9] J.-L. Florens, A. Luciani, C. Cadoz, and N. Castagné. Ergos: Multi-degrees of freedom and versatile force-feedback panoply. In *EuroHaptics 2004*, 2004.

[10] J.-L. Florens, A. Voda, and D. Urma. Dynamical issues in interactive representation of physical objects. In *EuroHaptics 2006 conference*, pages 213–219, 2006.

[11] R. B. Gillespie and S. O'Modhrain. Embodied cognition as a motivating perspective for haptic interaction design: A position paper. In *World Haptics Conference (WHC), 2011 IEEE*, pages 481–486. IEEE, 2011.

[12] M. Giordano and M. M. Wanderley. Perceptual and technological issues in the design of vibrotactile-augmented interfaces for music technology and media. In *International Workshop on Haptic and Audio Interaction Design*, pages 89–98. Springer, 2013.

[13] A. Gupta and M. K. O'Malley. Design of a haptic arm exoskeleton for training and rehabilitation. *IEEE/ASME Transactions on mechatronics*, 11(3):280–289, 2006.

[14] V. Hayward, O. R. Astley, M. Cruz-Hernandez, D. Grant, and G. Robles-De-La-Torre. Haptic interfaces and devices. *Sensor Review*, 24(1):16–29, 2004.

[15] A. Kontogeorgakopoulos and C. Cadoz. Cordis anima physical modeling and simulation system analysis. In *4th Sound and Music Computing Conference 2007*, pages 275–282. National and Kapodistrian University of Athens, 2007.

[16] J. Leonard and C. Cadoz. Physical modelling concepts for a collection of multisensory virtual musical instruments. In *New Interfaces for Musical Expression 2015*, pages 150–155, 2015.

[17] J. Leonard, N. Castagné, C. Cadoz, and A. Luciani. The msci platform: A framework for the design and simulation of multisensory virtual musical instruments. In *Musical Haptics*, pages 151–169. Springer, 2018.

[18] S. Marlière, F. Marchi, J.-L. Florens, A. Luciani, and J. Chevrier. An augmented reality nanomanipulator for learning nanophysics: The. In *International Conference on Cyberworlds 2008*, pages 94–101. IEEE, 2008.

[19] S. Martin and N. Hillier. Characterisation of the novint falcon haptic device for application as a robot manipulator. In *Australasian Conference on Robotics and Automation (ACRA)*, pages 291–292. Citeseer, 2009.

[20] T. H. Massie, J. K. Salisbury, et al. The phantom haptic interface: A device for probing virtual objects. In *Proceedings of the ASME winter annual meeting, symposium on haptic interfaces for virtual environment and teleoperator systems*, volume 55, pages 295–300. Citeseer, 1994.

[21] J. McCartney. Rethinking the computer music language: Supercollider. *Computer Music Journal*, 26(4):61–68, 2002.

[22] S. Papetti and C. Saitis. Musical haptics: Introduction. In *Musical Haptics*, pages 1–7. Springer, 2018.

[23] S. Sinclair and M. M. Wanderley. A run-time programmable simulator to enable multi-modal interaction with rigid-body systems. *Interacting with Computers*, 21(1-2):54–63, 2008.

[24] W. Verplank. Haptic music exercises. In *Proceedings of the 2005 conference on New interfaces for musical expression*, pages 256–257. National University of Singapore, 2005.

[25] W. Verplank, M. Gurevich, and M. Mathews. The plank: designing a simple haptic controller. In *Proceedings of the 2002 conference on New interfaces for musical expression*, pages 1–4. National University of Singapore, 2002.