Real-Time Control of Large-Scale Modular Physical Models using the Sensel Morph

Willemsen, Silvin; Andersson, Nikolaj Schwab; Serafin, Stefania; Bilbao, Stefan

Published in:
Proceedings of the 16th Sound and Music Computing Conference

DOI (link to publication from Publisher):
10.5281/zenodo.3249295

Publication date:
2019

Document Version
Accepted author manuscript, peer reviewed version

REAL-TIME CONTROL OF LARGE-SCALE MODULAR PHYSICAL MODELS USING THE SENSEL MORPH

Silvin Willemsen, Nikolaj Andersson, Stefania Serafin
Multisensory Experience Lab, CREATE,
Aalborg University Copenhagen
Copenhagen, Denmark
{sil, nsa, sts}@create.aau.dk

Stefan Bilbao
Acoustics and Audio Group
University of Edinburgh
Edinburgh, UK
s.bilbao@ed.ac.uk

ABSTRACT

In this paper, implementation, instrument design and control issues surrounding a modular physical modelling synthesis environment are described. The environment is constructed as a network of stiff strings and a resonant plate, accompanied by user-defined connections and excitation models. The bow, in particular, is a novel feature in this setting. The system as a whole is simulated using finite difference (FD) methods. The mathematical formulation of these models is presented, alongside several new instrument designs, together with a real-time implementation in JUCE using FD methods. Control is through the Sensel Morph.

1. INTRODUCTION

Physical models for sound synthesis have been researched for several decades to mathematically simulate the sonic behaviour of musical instruments and everyday sounds. Various techniques and methodologies have developed, ranging from mass-spring models [1–3] to modal synthesis [4] and waveguide based models [5]. The latter two techniques may be viewed as numerical simulation techniques applied to the systems of partial differential equations (PDEs). These equations define the dynamics of a musical instrument, either real or imagined.

Mainstream time-domain simulation techniques, such as finite difference (FD) methods, were first applied to the case of string vibration by Ruiz [6] and Hiller and Ruiz [7, 8], and then later by other authors [9] including, most notably Chaigne [10] and Chaigne and Askenfelt [11]. The general use of finite-difference schemes (FDSs) in sound synthesis is described in [12]. Modularized physical modelling sound synthesis, whereby the user may construct a virtual instrument using basic canonical components dates back to the work of Cadoz and collaborators [1–3]. It has been also used as a design principle in the context of FD methods [13–15], where the canonical elements are strings and plates, with a non-linear connection mechanism. Though computational cost of such methods is high, standard computing power is now approaching a level suitable for real-time performance for simpler systems.

We are interested in bridging the gap between large-scale modular physical modelling synthesis and sonic interaction design [16], to be able to play with such simulations in real-time. Specifically, we are interested in using the expressivity of the Sensel Morph [17] to control our simulations, using both percussive and bowing excitations. Our ultimate goal is to create models that are both mathematically accurate and efficient. This goal is nowadays possible thanks to improvements in hardware and software technologies for sound synthesis, yet it has rarely been achieved. The ultimate goal is to provide a modular efficient synthesizer based on accurate simulations, where real-time expressivity can also be achieved. This synthesizer has already been informally evaluated by composers and sound designers, who appreciated the current sonic palette.

This paper is structured as follows: Section 2 describes the physical models used in the implementation and Section 3 shows a general description of the FD methods used to digitally implement these models. Furthermore, Section 4 elaborates on the real-time implementation, Section 5 shows several different configurations of the physical models inspired by real musical instruments, Section 6 will present the results on CPU usage and evaluation and discuss this and finally, in Section 7, some concluding remarks appear.

2. MODELS

In this section, the PDEs for the damped stiff string and plate will be presented. The notation used will be the one found in [12] where the subscript for state variable u denotes a single derivative with respect to time t or space x respectively. Furthermore, to simplify the presented physical models, non-dimensionalization (or scaling) will be used [12].

2.1 Stiff string

A basic model of the linear transverse motion of a string of circular cross section may be described in terms of several parameters: the total length L (in m), the material density ρ (in kg·m^{-3}), string radius r (in m), Young's modulus E (in Pa), tension T (in N), and two loss parameters σ_0 and σ_1.

The PDE for a damped stiff string may be written as [12]

$$u_{tt} = \gamma^2 u_{xx} - \kappa^2 u_{xxxx} - 2\sigma_0 u_t + 2\sigma_1 u_{txx}. \quad (1)$$

In this representation, spatial scaling has been employed using a length L, so the solution $u = u(x,t)$ is defined for $t \geq 0$ and for dimensionless coordinate $x \in [0,1]$. Furthermore, parameters $\gamma = \sqrt{T/\rho\pi r^2 L^2}$ and $\kappa = \sqrt{Er^2/4\rho L^4}$ and have units s^{-1}.

In this work, the string is assumed clamped at both ends, so that

$$u = u_x = 0 \quad \text{where} \quad x = \{0,1\}. \quad (2)$$

A model of a bowed string [12] may be incorporated into (1) as

$$u_{tt} = \ldots - \delta(x - x_\mathrm{B}) F_\mathrm{B} \phi(v_\mathrm{rel}), \quad \text{with} \quad (3a)$$
$$v_\mathrm{rel} = u_t|_{(x=x_\mathrm{B})} - v_\mathrm{B}, \quad (3b)$$

where $F_\mathrm{B} = f_\mathrm{B}/M_\mathrm{s}$ is the excitation function (in $\mathrm{m/s}^2$) with externally-supplied bowing force $f_\mathrm{B} = f_\mathrm{B}(t)$ (in N) and total string mass $M_\mathrm{s} = \rho\pi r^2 L$ (in kg). The relative velocity v_rel is defined as the difference between the velocity of the string at bowing point x_B and the externally-supplied bowing velocity $v_\mathrm{B} = v_\mathrm{B}(t)$ (in m/s) and ϕ is a dimensionless friction characteristic, chosen here as [12]

$$\phi(v_\mathrm{rel}) = \sqrt{2a} v_\mathrm{rel} e^{-a v_\mathrm{rel}^2 + 1/2}. \quad (4)$$

Furthermore, $\delta(x - x_\mathrm{B})$ is a spatial Dirac delta function selecting the bowing location $x = x_\mathrm{B}$. The single bowing point can be extended to a bowing area [12]. More detailed models of string dynamics, again in a bowed string context, have been proposed by Desvages [18].

Another, and more simple way to excite the string is by extending Equation (1) to

$$u_{tt} = \ldots + E_\mathrm{e} F_\mathrm{e} \quad (5)$$

using an externally-supplied distribution function $E_\mathrm{e} = E_\mathrm{e}(x)$ and excitation function $F_\mathrm{e} = F_\mathrm{e}(t)$. In this case, the excitation region is allowed to be of finite width.

2.2 Plate

Under linear conditions, a rectangular plate of dimensions L_x and L_y may be parameterized in terms of density ρ (in $\mathrm{kg}\cdot\ \mathrm{m}^{-3}$), thickness H (in m), Young's modulus E (in Pa) and a dimensionless Poisson's ratio ν, as well as two loss parameters σ_0 and σ_1.

In terms of dimensionless spatial coordinates x and y scaled by $\sqrt{L_x L_y}$, the equation of motion of a damped plate is a variant of the Kirchhoff model [19]

$$u_{tt} = -\kappa^2 \Delta\Delta u - 2\sigma_0 u_t + 2\sigma_1 \Delta u_t. \quad (6)$$

Here, $u(x,y,t)$ is the transverse displacement of the plate as a function of dimensionless coordinates $x \in [0, \sqrt{a}\,]$, $y \in [0, 1/\sqrt{a}\,]$, where $a = L_x/L_y$ is the plate aspect ratio, as well as time t. Furthermore, Δ represents the 2D Laplacian [12]:

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}. \quad (7)$$

The stiffness parameter κ, with dimensions of s^{-1}, is defined by $\kappa = \sqrt{D/\rho H L_x^2 L_y^2}$ where $D = EH^3/12\left(1-\nu^2\right)$. As in the case of the stiff string, we chose to use clamped boundary conditions:

$$u = \mathbf{n} \cdot \nabla u = 0 \quad (8)$$

over any plate edge with outward normal direction $\mathbf{n}$ and where ∇u is the gradient of u.

2.3 Connections

Adding connections between different physical models, further referred to as elements, adds another term to Equation (3a), (5) or (6). Assuming that element α is a stiff string and β is a plate, the following terms are added to the aforementioned equations:

$$u_{tt} = \ldots + E_{\mathrm{c},\alpha} F_\alpha, \quad (9a)$$
$$u_{tt} = \ldots + E_{\mathrm{c},\beta} F_\beta, \quad (9b)$$

with force-functions $F_\alpha = F_\alpha(t)$ and $F_\beta = F_\beta(t)$ (in $\mathrm{m/s}^2$) and distribution functions $E_{\mathrm{c},\alpha}$ and $E_{\mathrm{c},\beta}$ which have chosen to be highly localised in our application and reduce to $\delta(x - x_{\mathrm{c},\alpha})$ and $\delta(x - x_{\mathrm{c},\beta}, y - y_{\mathrm{c},\beta})$ respectively, but can be extended to be connection areas [13]. We use the implementation as presented in [13] where the connection between two elements is a non-linear spring. The forces it imposes on the elements it connects are defined as

$$F_\alpha = -\omega_0^2 \eta - \omega_1^4 \eta^3 - 2\sigma_\times \dot{\eta}, \quad (10a)$$
$$F_\beta = -\mathcal{M} F_\alpha, \quad (10b)$$

where ω_0 and ω_1 are the linear (in s^{-1}) and non-linear (in $(\mathrm{m}\cdot\mathrm{s})^{-1/2}$) frequencies of oscillation respectively, $\sigma_\times$ is a damping factor (in s^{-1}), $\mathcal{M}$ is the mass ratio between the two elements and η is the relative displacement between the connected elements at the point of connection (in m). Lastly, the dot above η denotes a derivative with respect to time.

3. FINITE-DIFFERENCE SCHEMES

To be able to digitally implement the continuous equations shown in the previous section, they need to be approximated. In this section, a high-level review of a finite difference approximation to a connected system of strings and plates is presented. For more technical details, see [13].

In the case of the stiff string, state variable $u(x,t)$ can be discretised at times $t = nk$, where $n \in \mathbb{N}$ and $k = 1/f_\mathrm{s}$ is the time step (at sample-rate f_s) and locations $x = lh$, with $l \in [0, \ldots, N]$ where the total number of points is $N+1$ and grid spacing h. We can now write the discretised state variable as u_l^n, representing an approximation to $u(x,t)$.

In the case of the plate, $u(x,y,t)$ is discretised to $u_{(l,m)}^n$ using $x = lh$ where $l \in [0, \ldots, N_x]$ with $N_x + 1$ being the total horizontal number of points and $y = mh$ where $m \in [0, \ldots, N_y]$ with $N_y + 1$ being the total vertical number of points.

In a general sense, when discretising PDEs as presented in Equations (1) and (6), we will need to solve for $\mathbf{u}^{n+1}$,

i.e., $\mathbf{u}$ at the next time step, where $\mathbf{u}$ is a vector of size $N-1$ containing values of $u_l\ \forall l$ for a string and $(N_x-1)(N_y-1)$ containing values of $u_{(l,m)}\ \forall(l,m)$ for a plate. Note that the vector sizes are smaller than the total number of grid points as we do not include the values at the boundaries (which are always 0). For a PDE expressed as a function of u_{tt}, its FDS will be of the form

$$\mathbf{u}^{n+1} = 2\mathbf{u}^n - \mathbf{u}^{n-1} + K\boldsymbol{\mathcal{F}}^n, \tag{11}$$

where

$$K = \frac{k^2}{1+\sigma_0 k}, \tag{12}$$

and $\boldsymbol{\mathcal{F}}^n$ is a combination of the discretised PDE (excluding terms containing u^{n+1}) together with connection and excitation terms.

3.1 Stiff String

In the case of the stiff string, $\boldsymbol{\mathcal{F}}^n$ in Equation (11) is a combination of the discretised PDE (1) $\mathbf{f}_\alpha^n$, connection term (9a) and bowing (3a)

$$\boldsymbol{\mathcal{F}}^n = \mathbf{f}_\alpha^n + \mathbf{E}_{\mathrm{c},\alpha}F_\alpha^n - \mathbf{J}(x_\mathrm{B}^n)F_\mathrm{B}^n\phi(v_\mathrm{rel}), \tag{13a}$$

or excitation (5) term

$$\boldsymbol{\mathcal{F}}^n = \mathbf{f}_\alpha^n + \mathbf{E}_{\mathrm{c},\alpha}F_\alpha^n + \mathbf{E}_\mathrm{e}F_\mathrm{e}^n, \tag{13b}$$

where $\mathbf{E}_{\mathrm{c},\alpha}$ contains the discretised distribution function for the connection ($1/h$ at connection index $l_{\mathrm{c},\alpha}$, rest 0's [12]), $\mathbf{E}_\mathrm{e}$ contains the discretised distribution function for the excitation (which will be presented in Equation (25) in the next section) and $\mathbf{J}(x_\mathrm{B}^n)$ is a spreading operator containing the discretised bowing distribution ($1/h$ at time-varying bowing position x_B). If x_B is between grid points, cubic interpolation is used to spread the bow-force over neighbouring grid points [12]. All vectors are columns of size $N-1$.

It can be useful to talk about the *region of operation* of a FDS in terms of a 'stencil'. A stencil describes the number of grid points needed to calculate a single point at the next time step. The stiff string FDS has a stencil of 5 grid points. In other words, two grid points at either side of l – and l itself – are necessary to calculate u_l^{n+1}. See Figure 1 for a visualisation of this.

In order for the scheme to be stable, the grid spacing needs to abide the following condition [12]

$$h \geq \sqrt{\frac{\gamma^2k^2 + 4\sigma_1 k + \sqrt{(\gamma^2k^2+4\sigma_1 k)^2 + 16\kappa^2k^2}}{2}}. \tag{14}$$

The closer h is to this limit, the higher the quality of the implementation. The number of points N can then be calculated using

$$N = \mathrm{floor}\left(\frac{1}{h}\right). \tag{15}$$

3.2 Plate

In the case of the plate, $\mathbf{u}$ is a column vector of concatenated vertical 'strips' of the plate state as in [13] of size

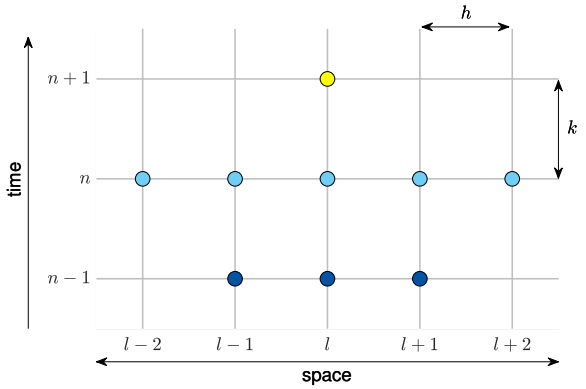

Figure 1. Stencil for a stiff string FDS with grid spacing h and time step k. The point l at the next time step (yellow) is calculated using 5 points at the current time step (blue) and 3 at the previous time step (dark blue).

$(N_x-1)(N_y-1)$ and $\boldsymbol{\mathcal{F}}^n$ in Equation (11) is a combination of the discretised PDE (6) $\mathbf{f}_\beta^n$ and connection term (9b)

$$\boldsymbol{\mathcal{F}}^n = \mathbf{f}_\beta^n + \mathbf{E}_{\mathrm{c},\beta}F_\beta^n. \tag{16}$$

Here, $\mathbf{E}_{\mathrm{c},\beta}$ contains the discretised distribution function for the connection ($1/h^2$ at connection index ($l_{\mathrm{c},\beta}$, $m_{\mathrm{c},\beta}$), rest 0's [13]) and is a column vector of size $(N_x-1)(N_y-1)$. For the plate, the stencil will consist of 13 grid points as can be seen in Figure 2.

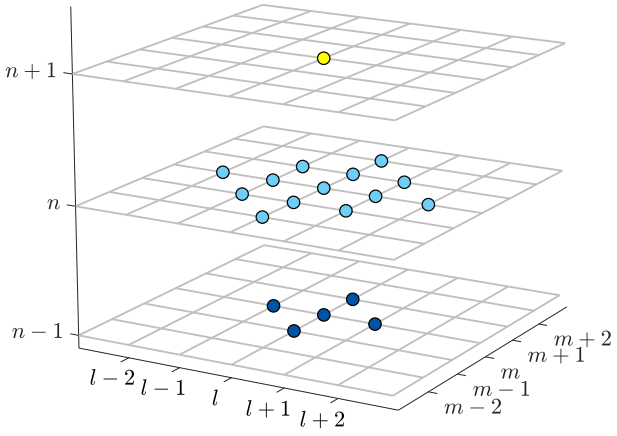

Figure 2. Stencil for a plate FDS. The point (l,m) at the next time step (yellow) is calculated using 13 points at the current time step (blue) and 5 at the previous time step (dark blue).

The grid spacing needs to abide the following condition [13]

$$h \geq 2\sqrt{k\left(\sigma_1^2 + \sqrt{\kappa^2+\sigma_1^2}\right)}, \tag{17}$$

(again, the closer h is to this limit the better) from which

N_x and N_y can be derived using

$$N_x = \text{floor}\left(\frac{\sqrt{a}}{h}\right) \quad \text{and} \quad N_y = \text{floor}\left(\frac{1}{h\sqrt{a}}\right). \quad (18)$$

3.3 Connections

In the following, we discretise the equations in (10) as shown in [13]. However, as these equations are not expressed as a function of u_{tt}, their FDS counterpart will be different. Moreover, instead of solving for $\mathbf{u}^{n+1}$, we need to solve for η^{n+1}, i.e., the relative displacement at the next time step, which will be in the form of

$$\eta^{n+1} = p^n F_\alpha^n + r^n \eta^{n-1}, \quad (19)$$

where $p^n = p(\eta^n)$ and $r^n = r(\eta^n)$ are functions of the relative displacement η if $\omega_1 \neq 0$ and constants if $\omega_1 = 0$. Again, assuming that element α is a stiff string and β is a plate, η can be calculated using

$$\eta^n = h_\alpha u^n_{\alpha,l_{\text{c},\alpha}} - h_\beta^2 u^n_{\beta,(l_{\text{c},\beta},m_{\text{c},\beta})}. \quad (20)$$

In other words, this is the difference between the state of element α at $l_{\text{c},\alpha}$ and the state of element β at $(l_{\text{c},\beta}, m_{\text{c},\beta})$ scaled by their respective (for plates, squared) grid spacings h_α and h_β. The next step is to obtain F_α^n, which can be used to easily calculate F_β^n. We first obtain values for $\mathbf{u}^{n+1}$ by solving (11) using (13a), (13b) or (16) (without the connection term!) for a string or plate respectively. As, at this point, no connection forces have been added yet, this state will be referred to as an intermediate state $\mathbf{u}^{\text{I}}$. This intermediate state can be used to obtain η^{n+1} using (20)

$$\eta^{n+1} = h_\alpha(u^{\text{I}}_{\alpha,l_{\text{c},\alpha}} + K_\alpha F_\alpha^n) - \left[h_\beta^2(u^{\text{I}}_{\beta,(l_{\text{c},\beta},m_{\text{c},\beta})} + K_\beta F_\beta^n)\right], \quad (21)$$

where K_α and K_β are as described in (12) using the damping coefficient σ_0 of their respective element. This can then be set equal to (19). Using Equation (10b), solving for F_α yields

$$F_\alpha^n = \frac{r^n \eta^{n-1} - (h_\alpha u^{\text{I}}_{\alpha,l_{\text{c},\alpha}} - h_\beta^2 u^{\text{I}}_{\beta,(l_{\text{c},\beta},m_{\text{c},\beta})})}{h_\alpha K_\alpha + \mathcal{M} h_\beta^2 K_\beta - p^n}. \quad (22)$$

4. IMPLEMENTATION

In this section, we elaborate more on the chosen values for the parameters described in the previous two sections and present the system architecture of the real-time application. The values for most parameters have been arbitrarily chosen and can – as long as they satisfy the conditions in Equations (14) and (17) – be changed. We used C++ along with the JUCE framework [20] for implementing the physical models and connections in real-time. The main hardware used was a MacBook Pro with a 2.2 GHz Intel Core i7 processor.

4.1 Stiff String

As many string properties stay constant, we chose to set the following parameters directly, rather than calculating them from their physical properties: $\kappa = 2$, $\sigma_0 = 1$, $\sigma_1 = 0.005$. An interesting parameter to make dynamic is the fundamental frequency f_0 (in s^{-1}) of the string. According to [12], the fundamental frequency can be approximately calculated using

$$f_0 \approx \frac{\gamma}{2}. \quad (23)$$

However, as the grid spacing h is dependent on the wave speed γ according to the condition found in (14), we must put a lower limit on the number of points N if we plan to dynamically increase γ.

Another way to change frequency is to add damping to the model at specific points acting as a (simplified) fretting finger. The advantage of this is that the condition (14) will never be violated. On top of this, a tapping sound will be introduced when fretting the string making it more realistic than changing the wave speed. If the string is fretted at single location $x_\text{f} \in [0, 1]$ and $l_\text{f} = \text{floor}(x_\text{f}/h)$ we use

$$u_l^n = \begin{cases} 0, & l = l_\text{f} - 1 \vee l = l_\text{f} \\ (1 - \alpha_\text{f}^\epsilon) u_l^n, & l = l_\text{f} + 1 \\ u_l^n, & \text{otherwise} \end{cases} \quad (24)$$

where $\alpha_\text{f} = x_\text{f}/h - l_\text{f}$ describes the fractional location of x_f between two grid points. Note that the grid point at the finger location and the grid point before are set to 0 to (recalling the stencil) prevent the states at either side of the finger to influence each other. The disadvantage of using this technique over regular linear interpolation, is that the effect of damping between grid points does not linearly scale to pitch. We thus added $\epsilon = 7$ as a heuristic value to more properly map finger position to pitch.

In some cases, N is fixed to a certain value (as opposed to calculating it from Equations (14) and (15)) for multiple strings of different pitches. Even though some bandwidth will be lost (in the higher frequency range), this will allow the strings to be perfectly tuned to each other.

4.1.1 Bowed String

Parameters for the bowed strings abide the following conditions: $|v_\text{B}| \leq 1$ m/s and $0 \leq F_\text{B} \leq 100$ N. It was chosen to discretise Equation (3b) implicitly making it necessary to use an iterative root-finding method such as Newton-Raphson [21].

4.1.2 Excited string

If simply excited, we set the distribution function to a raised cosine with width w_e (in grid points)

$$E_\text{e}(l) = \begin{cases} \frac{1-\cos\left(\frac{2\pi(l-(l_\text{e}-w_\text{e}/2))}{w_\text{e}}\right)}{2}, & l_\text{e} - \frac{w_\text{e}}{2} < l < l_\text{e} + \frac{w_\text{e}}{2} \\ 0, & \text{otherwise} \end{cases} \quad (25)$$

scaled by the excitation function over time with excitation duration d_e (in samples)

$$F_\text{e}(n) = \begin{cases} \frac{1-\cos\left(\frac{\pi(n-n_\text{e})}{d_\text{e}}\right)}{2}, & n_\text{e} \leq n < n_\text{e} + d_\text{e} \\ 0, & \text{otherwise} \end{cases} \quad (26)$$

A visualisation of this can be found in Figure 3.

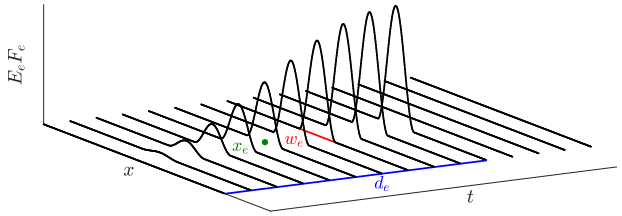

Figure 3. A visualisation of the excitation used in our implementation presented in Equation (5). The location of excitation x_e is shown in green, excitation width w_e in red and excitation duration d_e in blue (also see Equations (25) and (26)).

4.2 Plate

For the plate, the damping coefficients have been decided to be $\sigma_0 = 0.1$ and $\sigma_1 = 0.005$ and the aspect ratio is set to $a = 2$. The plate stiffness κ has been left as a user parameter to be changed dynamically and will be between the following bounds: $0.1 \leq \kappa \leq 50$ s^{-1}. In Equation (17), the grid spacing is calculated using the maximum value of κ to prevent stability issues. Using a sample rate of 44,100 Hz results in a plate with dimensions $N_x = 20$ and $N_y = 10$ (in grid points).

4.3 Connections

Increasing $\omega_1 \gtrsim 100,000$ (m·s)$^{-1/2}$ while keeping $0 < \omega_0 \lesssim 100$ s^{-1} will cause audible non-linear behaviour, such as pitch-glides and rattling sounds. These effects will be more dominant when the plate stiffness is higher. In our implementation we set $\omega_0 = 100$ s^{-1} and $\omega_1 = 100,000$ (m·s)$^{-1/2}$. The spring-damping $\sigma_\times = 1$ s^{-1} is kept to a minimum ($0 \leq \sigma_\times \leq 10$ s^{-1}).

4.4 System Architecture

The system architecture can be seen in Figure 4. The top box denotes the Sensel Morph (described in more detail in the next section) controlling the application, and the white boxes show the different classes or components of the application. The black arrows indicate instructions that one class can give to another and the hollow arrows show data flows between classes. All arrows are accompanied by a coloured box indicating which thread the instruction / dataflow is associated with and at what rate this thread runs.

The lowest priority thread, the graphics-thread, is shown by green boxes and runs at 15 Hz. This draws the states of the strings, connections and the plate on the screen.

Checking and retrieving the Sensel state happens at a rate of 150 Hz and is denoted by blue boxes. The parameters that the user controls by means of the Sensel, such as bowing position, force and velocity, will be updated in the models at this rate as well.

The highest priority thread is the audio-thread and runs at commonly-used sample rate 44,100 Hz. The main application gives an 'update' (u) instruction to the instrument,

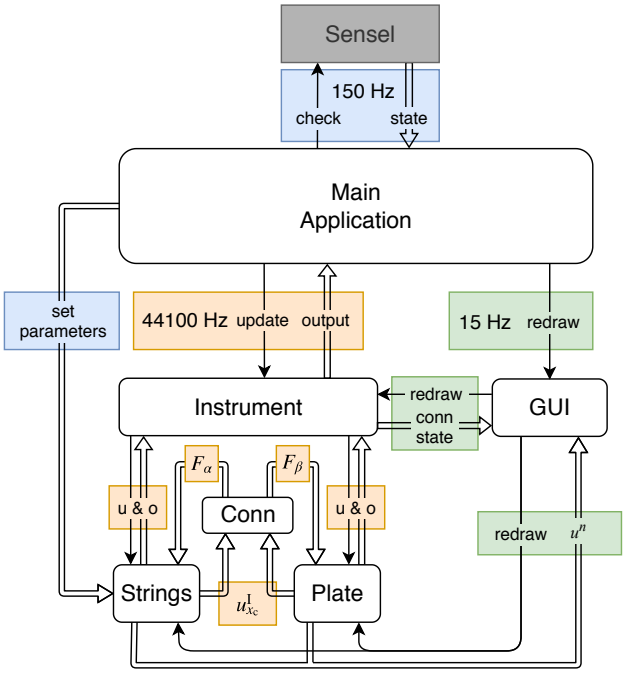

Figure 4. System architecture flowchart. See Section 4.4 for a thorough explanation.

which in turn updates the FDSs in its strings and plate. After the FDS update is done, the intermediate state at the connection points $u^{\mathrm{I}}_{x_c}$ (where $x_c = l_{c,\alpha}$ for the string or $x_c = (l_{c,\beta}, m_{c,\beta})$ for the plate) are sent to the connection (Conn) class which calculates the force-functions F_α and F_β. These values are then sent back to the string and plate classes and added to their respective states after which their outputs (o) (at arbitrary points) are sent back to the main application. See Algorithm 1 for this 'order of calculation'.

5. INSTRUMENTS AND USER INTERACTION

In this section, the Sensel Morph (or simply Sensel) and user interface will be described in more detail. Furthermore, several configurations of strings, plates and connections that are inspired by real-life instruments will be presented. A demonstration of one of the instruments can be found in [22].

5.1 Sensel Morph

The Sensel Morph is a highly accurate touch controller that senses position and force of objects [17] (see Figure 5). We use the Sensel as an expressive interface for interacting with the instrument configurations. Right above the touch-sensitive area, the Sensel contains an array of 24 LEDs that can be controlled from the application.

5.2 User interface

Strings are shown as coloured paths (see Figure 6 for a descriptive visualisation). The state $\mathbf{u}^n$ of the string is visualised using the vertical displacement of the paths. Bowed strings are shown in cyan on the top left. The bow is shown

while *application runs* **do**
  for *all elements* **do**
    calculate intermediate state $\mathbf{u}^{\mathrm{I}}$ using previous state values (as in Equation (11))
    $\mathbf{u}^{\mathrm{I}}_{\mathrm{s}} = 2\mathbf{u}^{n} - \mathbf{u}^{n-1} + K\boldsymbol{\mathcal{F}}$
  end
  if *element is excited/bowed* **then**
    calculate excitation term $\mathbf{E}$ and add to intermediate state of the element
    $\mathbf{u}^{\mathrm{I}}_{\mathrm{s+e}} = \mathbf{u}^{\mathrm{I}}_{\mathrm{s}} + \mathbf{E}$
  end
  for *all connections* **do**
    calculate connection forces and add connection term $\mathbf{C}$ to elements to obtain the state at the next time step
    $\mathbf{u}^{n+1}_{\mathrm{s+e+c}} = \mathbf{u}^{\mathrm{I}}_{\mathrm{s+e}} + \mathbf{C}$
  end
  update state vectors
    $\mathbf{u}^{n-1} = \mathbf{u}^{n}$
    $\mathbf{u}^{n} = \mathbf{u}^{n+1}_{\mathrm{s+e+c}}$
  increment time step
    n++
end

Algorithm 1: Pseudocode showing the correct order of calculation. The subscripts for state vector $\mathbf{u}$ shows what it consists of ('s' for previous state, 'e' for excitation and 'c' for connection).

as a yellow rectangle and moves on interaction. The fretting position is shown as a yellow circle. Plucked strings are shown in purple in the top right, underneath which the sympathetic strings are shown in light green. The plate is shown in the bottom using a grid of rectangles (clamped grid points are not shown). Its state is visualised using a grey-scale. Furthermore, connections are shown using orange circles/squares for the points of connection and dotted lines between these points. Lastly, all parameters that are controlled by the mouse such as output-level and plate-stiffness are located in a column on the right side of the application.

5.3 Instruments

We subdivide string-elements into three types: bowed, plucked and sympathetic strings. All strings will be connected to one plate acting as an instrument body of which the user can control the plate-stiffness. Furthermore, the user can change the output-level of each element type. Apart from these parameters, which are controlled by the mouse, the instruments are fully controlled by two Sensels. The instruments we have chosen as our inspiration are the sitar, the hammered dulcimer and the hurdy gurdy.

5.3.1 Bowed Sitar

The sitar is originally an Indian string instrument that has both fretted strings and sympathetic strings. Instead of plucking the fretted strings, we extended the model to bow them. Our implementation consists of 2 bowed strings

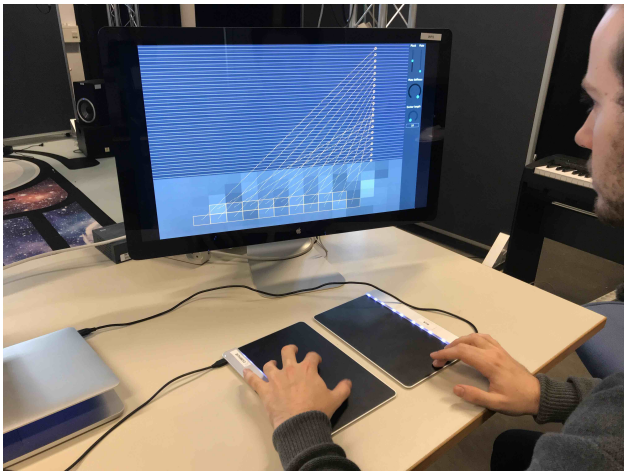

Figure 5. Player using the Sensel Morphs to interact with one of the instruments.

(tuned to A3 and E4), 13 sympathetic strings (tuned according to [23]) and 5 plucked strings (tuned A3-E4 following an A-major scale) as it is also possible to strum the sympathetic strings. See Figure 6 for a visual of the implementation. One Sensel is vertically subdivided into two sections; one for each bowed string. The first finger registered by the Sensel is mapped to a bow and the second is mapped to a fretting finger controlling pitch. The horizontal position of both fingers is visualised using the Sensel's LED array. The frets are not implemented as such (the pitch is continuous), but they are visualised for reference. The horizontal position of the first finger is mapped to the bowing position on the string, the vertical velocity to the bow velocity v_{B} and the finger force is linked to the excitation function F_{B} (both in Equation (3a)). The other Sensel is subdivided into 5 sections mapped to the plucked strings. These sections are visualised by the LED array for reference.

The mass ratio for the bowed/plucked string to plate connections has been set to $\mathcal{M} = 2$ and ratio for the sympathetic string to plate connections has been set to $\mathcal{M} = 0.5$ to increase the effect that the playable strings have on the sympathetic strings.

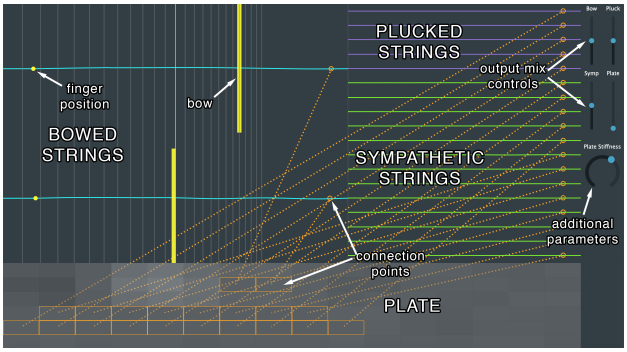

Figure 6. The bowed sitar application. The descriptions of the different elements and other objects are shown in the image, but will (naturally) not be visible in the application.

5.3.2 *Hammered Dulcimer*

The hammered dulcimer is an instrument that can be seen as an 'open piano' where the musician has the hammers in their hand. Just like the piano, the strings are grouped in pairs or triplets that are played simultaneously. In our implementation, we have 20 pairs of plucked strings. Even though most hammered dulcimers have more strings, we decided that this configuration has the highest number of strings while maintaining playability. One of each pair is connected to the plate which slightly detunes it, creating a desired 'chorusing' effect. See Figure 7 for a visual of the implementation. In order for the excitation to more resemble a strike of a hammer than a pluck, the contents of the cosine in (26) will be multiplied by 2 for the excitation to have a less abrupt ending, something desired for a hammered interaction. Moreover, the excitation-length can be changed to simulate short and long hammer-times.

The Sensels are placed vertically next to each other (see Figure 5). The pair with the lowest frequency will then be located in the bottom right and the highest in the top left, as in the real instrument. As with the plucked strings of the bowed sitar, the LED array is used to visualise the way that the Sensel is subdivided, which is especially useful here as one Sensel controls 10 string-pairs.

The mass ratio is set relatively high ($\mathcal{M} = 100$) to amplify the non-linear interaction between the strings and the detuning of the strings connected to the plate.

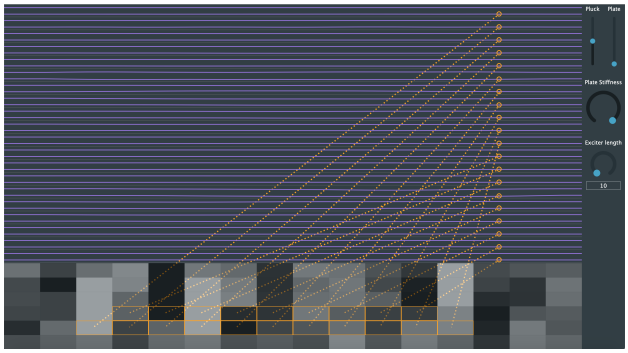

Figure 7. The hammered dulcimer application.

5.3.3 *Hurdy Gurdy*

The hurdy gurdy is an instrument that consists of bowed and sympathetic strings. The bowing happens through a rosined wheel attached to a crank and bows these strings as the crank is turned. It is possible to change the pitch of a few bowed strings - the melody strings - using buttons that press tangent pins on the strings at different positions. The other strings, referred to as drone strings, are mostly tuned lower than the melody strings and provide the bass frequencies of the instrument. The musician can place the bowed strings on rests that keep the wheel from interacting with it.

Our implementation consists of 5 bowed strings subdivided into 2 drone strings tuned to A2, E3 and 3 melody strings tuned to A3, E4 and A4 and 13 sympathetic strings tuned the same way as the sympathetic strings in bowed sitar. Furthermore, the mass ratios have been set the same as in the bowed sitar application. See Figure 8 for a visual of the implementation.

The Sensel is vertically subdivided into 5 rows that control whether the strings are placed on the wheel. The bowing velocity is mapped to the average pressure of the fingers. The fundamental frequency (in the model $\gamma/2$) of the melody-strings is changed by a Sensel with a piano-overlay acting as a midi controller. A demonstration of this instrument can be found in [22]. It is interesting to note here that the sympathetic strings that are in tune with the harmonics of the bowed strings resonate most, which is expected to happen in the real world as well.

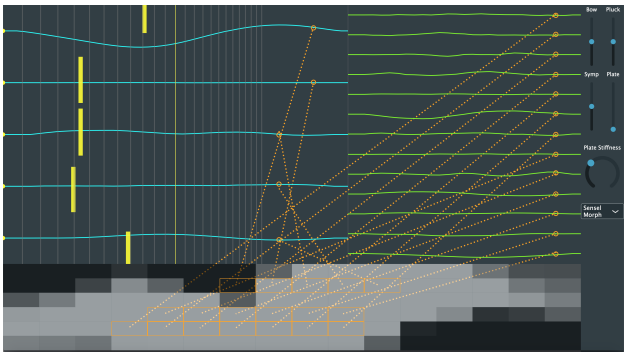

Figure 8. The hurdy gurdy application.

6. RESULTS AND DISCUSSION

Table 1 shows the CPU usage (on the same MacBook Pro 2.2 GHz i7 as described before) for the three instruments presented in the previous section. As the Sensel thread contributes a negligible amount to the CPU usage, this is not shown in the table.

Application	No Sound	No Graphics	Total
Bowed Sitar	32	63	85
Dulcimer	30	66	85
Hurdy Gurdy	28	58	78

Table 1. CPU usage (in %) for the instruments found in Section 5. Values show usage of one (virtual) thread and have been taken as an average (with a margin of ~5%) over a short period of time. The two middle columns show usage when the sound or graphics thread has been turned off.

As can be seen from the table, all instruments use about the same amount of CPU and none of them have audible dropouts (CPU $< 100\%$). It can be observed that the graphics use about 20% of the CPU, indicating that there is still much room to increase the complexity of the instrument-configurations before dropouts will occur. On the other hand, should the instruments be used in parallel with other audio applications or plug-ins, the CPU usage has to be greatly reduced. The first step towards this would be to vectorise the FDSs using AVX instructions.

While our instruments have been not formally evaluated yet, we have performed some qualitative evaluations with

sound and music computing experts. The goals of the evaluations were to explore the playability of the instrument, sonic quality and intuitiveness of control. These evaluations showed that especially the bowing interaction feels intuitive and creates a natural sound. The overall sound of the instruments was generally judged to be interesting, but not "sounding like a real-life instrument". This makes sense, as we did not seek to perfectly model each instrument, but rather used them as an inspiration for the configurations of the physical models. The next step for sound quality would be to replace the thin plate with a more realistic element, such as a wooden instrument body.

7. CONCLUSION

In this paper, a real-time modular physical modelling synthesis environment structured as a network of connected strings and plates has been presented. Several instruments have been created in the context of this environment which can be played by a pair of Sensel Morphs allowing for highly expressive control of these instruments. Informal evaluations with professional musicians have confirmed that the interaction is found natural and the output sound interesting. Further steps to improve this project are to optimise the algorithm and to replace the plate with a more realistic instrument body.

Acknowledgments

This work is supported by NordForsk's Nordic University Hub Nordic Sound and Music Computing Network NordicSMC, project number 86892.

8. REFERENCES

[1] C. Cadoz, "Synthèse sonore par simulation de mécanismes vibratoires," 1979, thèse de Docteur Ingénieur, I.N.P.G. Grenoble, France.

[2] C. Cadoz, A. Luciani, and J.-L. Florens, "Responsive input devices and sound synthesis by simulation of instrumental mechanisms," *Computer Music Journal*, vol. 8, no. 3, pp. 60–73, 1983.

[3] 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*, vol. 17, no. 1, pp. 19–29, 1993.

[4] J. D. Morrison and J.-M. Adrien, "Mosaic: A framework for modal synthesis," *Computer Music Journal*, vol. 17, no. 1, pp. 45–56, 1993.

[5] J. O. Smith, "Physical modeling using digital waveguides," *Computer music journal*, vol. 16, no. 4, pp. 74–91, 1992.

[6] P. Ruiz, "A technique for simulating the vibrations of strings with a digital computer," Master's thesis, University of Illinois, 1969.

[7] L. Hiller and P. Ruiz, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part I," *Journal of the Audio Engineering Society*, vol. 19, no. 6, pp. 462–470, 1971.

[8] ——, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part II," *Journal of the Audio Engineering Society*, vol. 19, no. 7, pp. 542–550, 1971.

[9] R. Bacon and J. Bowsher, "A discrete model of a struck string," *Acustica*, vol. 41, pp. 21–27, 1978.

[10] A. Chaigne, "On the use of finite differences for musical synthesis. Application to plucked stringed instruments," *Journal d'Acoustique*, vol. 5, no. 2, pp. 181–211, 1992.

[11] A. Chaigne and A. Askenfelt, "Numerical simulations of struck strings. I. A physical model for a struck string using finite difference methods," *Journal of Acoustical Society of America*, vol. 95, no. 2, pp. 1112–1118, 1994.

[12] S. Bilbao, *Numerical Sound Synthesis, Finite Difference Schemes and Simulation in Musical Acoustics*. John Wiley and Sons, Ltd, 2009.

[13] ——, "A modular percussion synthesis environment," *Proc. of the 12th Int. Conf. on Digital Audio Effects (DAFx-09)*, 2009.

[14] S. Bilbao, A. Torin, P. Graham, J. Perry, and G. Delap, "Modular physical modeling synthesis on gpu," in *Proceedings of the 40th International Computer Music Conference and the 11th Sound and Music Computing Conference*, 2013.

[15] C. Webb and S. Bilbao, "On the limits of real-time physical modelling synthesis with a modular environment," *Proc. of the 18th Int. Conference on Digital Audio Effects (DAFx-15)*, 2015.

[16] K. Franinović and S. Serafin, *Sonic interaction design*. Mit Press, 2013.

[17] Sensel Inc. (2018) Sensel morph. [Online]. Available: https://sensel.com/

[18] C. Desvages and S. Bilbao, "Two-polarisation physical model of bowed strings with nonlinear contact and friction forces, and application to gesture-based sound synthesis," *Applied Sciences*, 2016.

[19] K. Graff, *Wave Motion in Elastic Solids*. New York, New York: Dover, 1975.

[20] JUCE ROLI. (2019) JUCE. [Online]. Available: https://juce.com/

[21] J. Wallis, *A treatise of algebra, both historical and practical*. London, 1685.

[22] S. Willemsen. (2019) Hurdy gurdy demo. [Online]. Available: https://www.youtube.com/watch?v=BkxLji2ap1w

[23] Joe Rizzo. (2015) How to tune a sitar. [Online]. Available: http://www.joerizzo.com/sitar/