Real-time Control of Advanced Physical Models using the Sensel Morph

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

ABSTRACT

In this paper we introduce several physical models of bowed string instruments which use accurate numerical simulations using partial differential equations but are also efficient enough to run in real-time. We describe the mathematical formulation of these models, the real-time implementation in JUCE, the control using the Sensel Morph interface as well as results from qualitative evaluations from experts.

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. Several techniques ranging from numerical solutions of wave equations [1], mass-spring models [2], modal synthesis [3] and waveguide based models [4] have been developed over the years. More recently, the behaviour of musical instruments has been mathematically defined by partial differential equations (PDEs) [5].

Finite-difference schemes (FDSs) have been proposed to accurately simulate the behaviour of several musical instruments. Given the high computational cost of such simulations, most of them have been not implemented in real-time. We are interested in bridging the gap between accurate numerical solutions and sonic interaction design [6], to be able to control such simulations in real-time.

Specifically, we are interested in using the expressivity of Sensel Morph [7] to control physical models simulated in real-time; our ultimate goal is to create models that are both mathematically accurate but also efficient to be controlled in real-time. This goal is nowadays possible thanks to improvements in hardware and software technologies for sound synthesis, yet it has not been achieved yet.

This paper is structured as follows: Section 2 describes the physical models used in the implementation and Section 3 shows the FDSs used to digitally implement these models. Furthermore, Section 4 will show how to implement the FDSs, Section 5 shows several different configurations of the physical models presented inspired by real musical instruments and finally Section 7 and Section 8 will discuss and conclude upon the work shown in this paper.

Copyright: © 2019 Silvin Willemsen et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

2. MODELS

In this section, the PDEs for the damped stiff string and the plate will be presented. The notation used will be the one found in [8] (among others) 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 [8].

2.1 Stiff string

A basic model of the linear transverse motion of a string of circular cross section may be framed 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, to be described shortly. A partial differential equation model may be written as [8]

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

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 $x \in [0,1]$. $\gamma = \sqrt{T/\rho\pi r^2 L^2}$ and $\kappa = \sqrt{Er^2/4\rho L^4}$ are parameters with units s^{-1}. Subscripts represent partial differentiation with respect to time t and coordinate x.

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

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

A model of a bowed string [8] may be incorporated into (1) as

$$u_{tt} = ... - \delta(x - x_\text{B})F_\text{B}\phi(v_\text{rel}), \quad \text{with} \qquad (3)$$
$$v_\text{rel} = u_t(x_\text{B}) - v_\text{B} \qquad (4)$$

where $F_\text{B} = f_\text{B}/M_\text{s}$ is the excitation function (in m/s^2) with bowing force f_B (in N) and total string mass $M_\text{s} = \rho\pi r^2 L$ (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 bowing velocity v_B [m/s] and ϕ is a dimensionless friction characteristic, chosen here as [8]

$$\phi(v_\text{rel}) = \sqrt{2a}v_\text{rel}e^{-av_\text{rel}^2+1/2}. \qquad (5)$$

$\delta(x - x_\mathrm{B})$ 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 [8].

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

$$u_{tt} = ... + F_\mathrm{e} E_\mathrm{e}, \tag{6}$$

using an externally-supplied excitation function $F_\mathrm{e} = F_\mathrm{e}(t)$ and distribution function $E_\mathrm{e} = E_\mathrm{e}(x)$. 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 kg$\cdot$ 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, to be discussed shortly.

In terms of dimensionless spatial coordinates x and y scaled by $\sqrt{L_x L_y}$, the defining for a damped plate is [8]

$$u_{tt} = -\kappa^2 \Delta\Delta u - 2\sigma_0 u_t + 2\sigma_1 \Delta u_t, \tag{7}$$

Here, $u(x, y, t)$ is the transverse displacement of the plate as a function of dimensionless coordinates $x \in [0, \sqrt{\alpha}]$, $y \in [0, 1/\sqrt{\alpha}]$, where $\alpha = L_x/L_y$ is the plate aspect ratio, as well as time t. The subscript t represents partial differentiation wit respect to t, and Δ represents the 2D Laplacian [8]:

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \tag{8}$$

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:

$$\begin{aligned} u = u_x = 0 \quad &\text{where} \quad l = \{0, 1\} \quad \text{and} \\ u = u_y = 0 \quad &\text{where} \quad m = \{0, 1\}. \end{aligned} \tag{9}$$

2.3 Connections

Adding connections between different physical models, further referred to as elements, adds another term to Equation (3) or (7)

$$\begin{aligned} u_{tt} &= ... + \delta(x - x_{\mathrm{c},\alpha}) F_\alpha, \\ u_{tt} &= ... + \delta(x - x_{\mathrm{c},\beta}) F_\beta, \end{aligned} \tag{10}$$

where F_α and F_β are the forces of the connection at connection points $x_{\mathrm{c},\alpha}$ and $x_{\mathrm{c},\beta}$ respectively. For simplicity, we chose to only work with a connection point, but it can be extended to a connection area [9]. We use the implementation as presented in [9] where the connection between two elements is a non-linear spring. The forces it imposes on the elements it connects - denoted by α and β - are defined as

$$F_\alpha = -\omega_0^2 \eta - \omega_1^4 \eta^3 - 2\sigma_\times \dot{\eta}, \tag{11a}$$

$$F_\beta = -\mathcal{M}_{\alpha/\beta} F_\alpha, \tag{11b}$$

where ω_0 and ω_1 are the linear and non-linear (angular) frequencies of oscillation respectively [rad/s], $\sigma_\times$ is a dimensionless damping factor, $\mathcal{M}_{\alpha/\beta}$ is the mass ratio between the two elements and η is the relative displacement between the connected elements at the point of connection. Lastly, he 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. 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 with sample-rate f_s and locations $x = lh$, with $l \in [0, \dots, N]$ where N is the total number of points and grid-spacing h. We can now write the discretised state variable as u_l^n which is u at the nth time step and the lth point on the string.

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

As our implementation uses already existing models, we will not go much into depth in this section, but will rather show visualisations of the finite-difference schemes used.

In a general sense, when discretising PDEs as presented in Equations (2.1) and (7), we will need to solve for u^{n+1}, i.e. state u at the next time step. For a PDE of the form $u_{tt} = f$ the FDS will be of the form

$$u^{n+1} = 2u^n - u^{n-1} + k^2 f^n, \tag{12}$$

where f^n depends on the model at hand. It can be useful to talk about the **complexity** of f^n in terms of a 'stencil'. A stencil describes the number of grid-points needed to calculate a single point at the next time step.

3.1 Stiff String

In the case of a stiff string, f^n will have a stencil of 5 grid points. In other words, two grid-points at either side of l (including l itself) are necessary to calculate u^{n+1} at grid-point l. See Figure 1 for a visualisation of this for the stiff string.

3.2 Plate

For the plate, the stencil will consist of 13 grid-points as can be seen in Figure 2.

For stability reasons, the grid-spacing needs to abide the following condition

$$h \geq \sqrt{\frac{\gamma^2 k^2 + 4\sigma_1 k + \sqrt{(\gamma^2 k^2 + 4\sigma_1 k)^2 + 16\kappa^2 k^2}}{2}}. \tag{13}$$

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

$$N = h^{-1}. \tag{14}$$

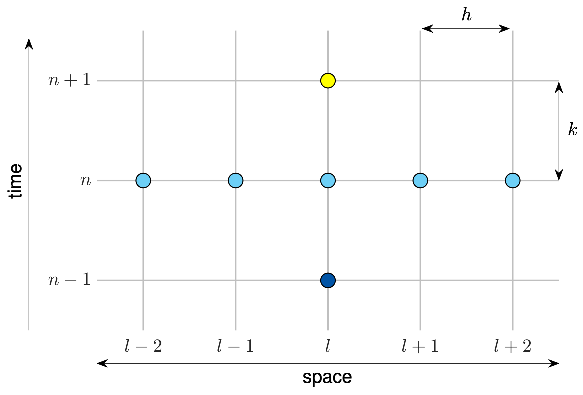

Figure 1. Stencil for a stiff string finite-difference scheme with grid-spacing h and time-step k. The point l at the next time-step (yellow) is calculated using five points at the current time-step (blue) and one at the previous time-step (dark blue).

In the case of the plate, calculated the grid spacing using [9]

$$h = 2\sqrt{k\left(\sigma_1^2 + \sqrt{\kappa^2 + \sigma_1^2}\right)} \tag{15}$$

after which we calculate N_x and N_y using

$$N_x = \text{floor}\left(\frac{\sqrt{a}}{h}\right) \tag{16}$$

$$N_y = \text{floor}\left(\frac{1}{h\sqrt{a}}\right) \tag{17}$$

where $a = N_x/N_y$ is a user-defined length-width ratio.

3.3 Connections

The equations in (11) can be approximated using [9]

$$F_\alpha = -\omega_0^2\mu_{t\cdot}\eta - \omega_1^4\eta^2\mu_{t\cdot}\eta - 2\sigma_\times\delta_{t\cdot}\eta, \tag{18a}$$

$$F_\beta = -\mathcal{M}_{\alpha/\beta}F_\alpha. \tag{18b}$$

The relative displacement η between α and β can be calculated as

$$\eta^n = \sigma_\alpha u^n_{\alpha,x_{\text{c},\alpha}} - \sigma_\beta u^n_{\beta,x_{\text{c},\beta}}, \tag{19}$$

where σ_α and σ_β are equal to h in the case of the string and h^2 for the plate. In other words, this is the difference between the state of element α at connection point $x_{\text{c},\alpha}$ and the state of element β at connection point $x_{\text{c},\beta}$ scaled by their respective (squared) grid-spacings σ_α and σ_β. Solving (18a) for η^{n+1} yields

$$\eta^{n+1} = p^nF_\alpha + r^n\eta^{n-1}, \tag{20}$$

where

$$p^n = \frac{-2}{2\sigma_\times/k + \omega_0^2 + \omega_1^4(\eta^n)^2}, \tag{21a}$$

$$r^n = \frac{2\sigma_\times/k - \omega_0^2 - \omega_1^4(\eta^n)^2}{2\sigma_\times/k + \omega_0^2 + \omega_1^4(\eta^n)^2}. \tag{21b}$$

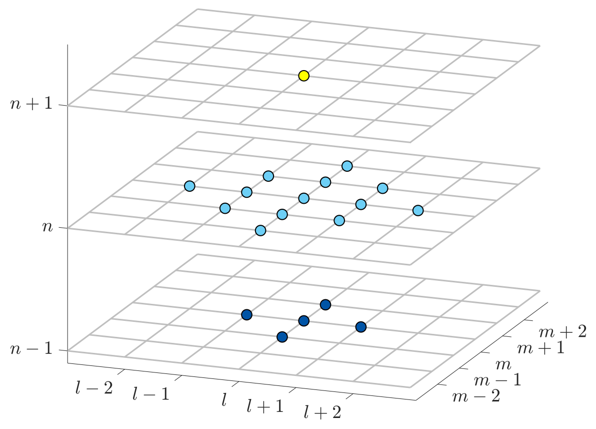

Figure 2. Stencil for a plate finite-difference scheme. The point (l, m) at the next time-step (yellow) is calculated using 13 points at the current time-step (blue) and five at the previous time-step (dark blue).

The next step is to obtain F_α which can be used to easily calculate F_β. We first obtain values for u by solving (**??**) or (**??**) for u^{n+1} for a string or plate respectively. As, at this point, no connection forces have been added yet, this state is referred to as an intermediate state or u^{I}. This intermediate state can be used to obtain η^{n+1} using (19):

$$\eta^{n+1} = \sigma_\alpha u^{\text{I}}_{\alpha,x_{\text{c},\alpha}} - \sigma_\beta u^{\text{I}}_{\beta,x_{\text{c},\beta}} \tag{22}$$

which can be set equal to (20). Solving for connection force F_α yields:

$$F_\alpha = \frac{r^n\eta^{n-1} - (\sigma_\alpha u^{\text{I}}_{\alpha,x_{\text{c},\alpha}} - \sigma_\beta u^{\text{I}}_{\beta,x_{\text{c},\beta}})}{\sigma_\alpha J_\alpha u^{\text{I}}_{\alpha,x_{\text{c},\alpha}} - \sigma_\beta J_\beta u^{\text{I}}_{\beta,x_{\text{c},\beta}} - p^n}, \tag{23}$$

where scaling factor J for element E is defined as

$$J_{\text{E}} = \frac{k^2}{1 + \sigma_{0,\text{E}}k}. \tag{24}$$

4. IMPLEMENTATION

In this section, we elaborate more on parameter choices 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 - be changed. We used C++ along with the JUCE framework [10] for implementing the physical models and connections in real-time. The main hardware used for testing was a MacBook Pro with a 2.2 GHz Intel Core i7 processor. The most important thing is to optimise the FDSs as these will be ran at sampling rate. We make sure to only have one multiplication per location of u per time-step (u_l^n, u_{l+1}^{n-1} etc) (**SEE APPENDIX DAFX PAPER**)

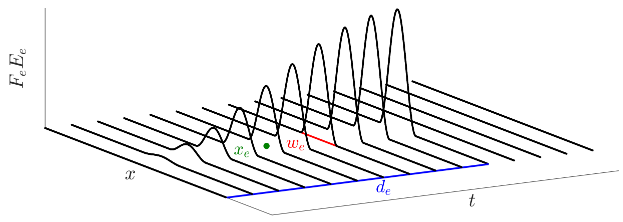

Figure 3. A visualisation of the excitation used in our implementation presented in Equation (6). The location of excitation x_e is shown in green, excitation width w_e in red and excitation duration d_e in blue (also see Equations (25) and (26)).

4.1 String

As can be seen from Equation (**??**) the solution for v_rel is implicit. It is thus necessary to use an iterative root-finding method such as Newton-Raphson [source].

If simply excited, or plucked in the case of a string, we set the distribution function is to a raised cosine with width w_e [m]

$$E(x) = \begin{cases} \frac{1-\cos\left(\frac{2\pi(x-x_\text{e})}{w_\text{e}}\right)}{2}, & x_\text{e} - \frac{w_\text{e}}{2} \le x \le x_\text{e} + \frac{w_\text{e}}{2} \\ 0, & \text{otherwise} \end{cases} \tag{25}$$

scaled by the excitation function over time with excitation duration d_e [s]

$$F(t) = \begin{cases} f_\text{e}\frac{1-\cos\left(\frac{\pi(t-t_\text{e})}{d_\text{e}}\right)}{2}, & t_\text{e} \le t \le t_\text{e} + d_\text{e} \\ 0, & \text{otherwise} \end{cases} \tag{26}$$

with excitation force f_e [N]. A visualisation of this can be found in Figure 3.

4.1.1 Frequency

There are several ways to change the frequency of the string According to [8], the fundamental frequency can be approximately calculated using

$$f_0 \approx \frac{\gamma}{2}. \tag{27}$$

However, as the grid spacing h is dependent on the wave speed γ according to the condition found in (13), we must put a lower limit on the number of points N if we plan to dynamically increase γ.

Another way to generate different pitches is to add damping to the model at specific points that acts as a fretting finger. The advantage of this is that the condition (13) 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 st single location $x_\text{f} \in [0, 1]$ and $l_\text{f} = \text{floor}(x_\text{f}/h)$ we use

$$u_l^n = \begin{cases} 0, & l = l_\text{f} - 1 \vee l = l_\text{f} \\ (1-\alpha_\text{f}^\epsilon)u_l^n, & l = l_\text{f} + 1 \\ u_l^n, & \text{otherwise} \end{cases} \tag{28}$$

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 prevent the states at either side of the finger to influence each other. The disadvantage of using this technique over 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.

4.2 Plate

An interesting parameter to change is the plate stiffness κ as this..

Length-width ratio $a = 2$.

4.3 Connections

As mentioned before, the connections need to be calculated using

In order to calculate the connection forces F_α and F_β, the relative displacement of the connection points E_α and E_β is needed. We must thus have knowledge of the states of connected elements α and β.

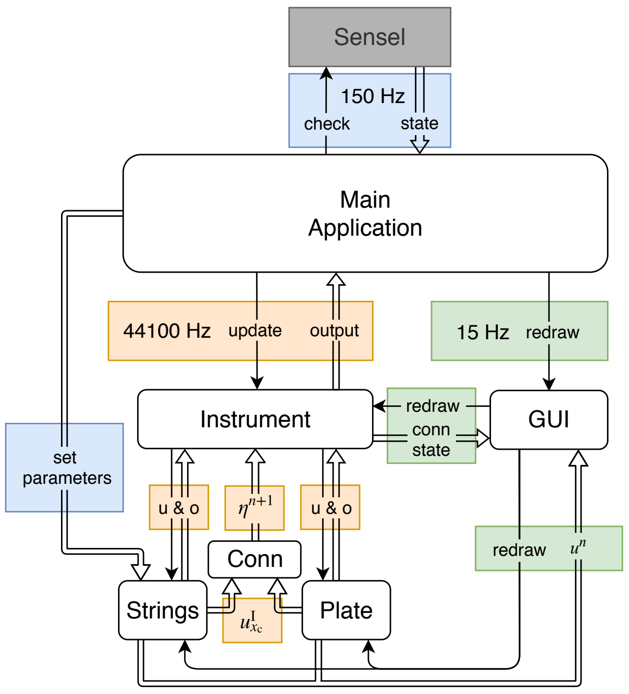

Figure 4. System architecture flowchart.

4.4 System Architecture

The system architecture can be seen in Figure 4. The white boxes show the different classes, or components of the application. The hierarchy we adopted can be seen in Figure 5.

The black arrows in Figure 4 indicate instructions that one class can give to another. The hollow arrows show data flows between classes. All arrows are accompanied by a coloured box indicating on which thread is associated this instruction / dataflow.

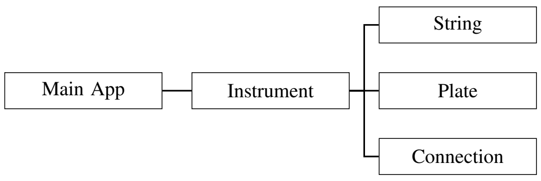

Figure 5. System hierarchy.

The lowest priority thread, the graphics-thread is shown by green boxes and runs at 15 Hz. This draws the states of the instrument 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 the same rate.

The highest priority thread is the audio-thread and runs at 44100 Hz. The main application gives an 'update' (u) instruction to the instrument which in turn updates the FDSs in its strings and plate. In order to update the states for connected elements, the

The connection class retrieves the intermediate state u^{I} at the connection point x_{c} for the two connected elements

The handling of the data from the sensel is 'converted' into usable parameters and sent to the string models (the plate is not controlled in the applications). As can be seen in Figure 4, In order to draw the system, the state needs to be retrieved. This happens on the GUI thread

The most important thing in real-time implementation of FDSs, is to optimise the FDSs themselves as these will be ran at sampling rate. We make sure to only have one multiplication per location of u per time-step (u_l^n, u_{l+1}^{n-1} etc) (**SEE APPENDIX DAFX PAPER**)

Furthermore, it is important to take note of is the order of calculation. First the non-extended schemes (Equations (1) and (7)) must be calculated before adding the effects of excitations (including bowing), which - in turn - needs to be calculated before adding the effects of connections (see Algorithm 1).

4.5 Sensel Morph

The Sensel Morph (or further referred to as Sensel) is an expressive touch controller that senses position and force of objects.

4.5.1 Mapping strategies

Something about the different prototype mappings, and the "final" mapping Lights to indicate bowing position or string position

5. INSTRUMENTS AND USER INTERACTION

In this section, several configurations of strings, plates and connections that are inspired by real-life instruments will be presented. We subdivide string-elements into three types: bowed, plucked and sympathetic strings. All strings will be connected to one plate which simulates an instrument body. The user can control the plate stiffness and increase it to generate more interaction between the strings. 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. Videos and sound examples can be found through the following link:...

```
while application runs do
    calculate intermediate state $u^{\mathrm{I}}$ for all elements using previous state values
        $u_{\mathrm{s}}^{\mathrm{I}} = Bu^n + Cu^{n-1}$
    if element is excited/bowed then
        calculate excitation term and add to intermediate state of the element
            $u_{\mathrm{s+e}}^{\mathrm{I}} = u_{\mathrm{s}}^{\mathrm{I}} + E$
    end
    for all connections do
        calculate connection forces and add to elements to obtain the state at the next time step
            $u_{\mathrm{s+e+c}}^{n+1} = u_{\mathrm{s+e}}^{\mathrm{I}} + C$
    end
    update statevectors
        $u^{n-1} = u^n$
        $u^n = u_{\mathrm{s+e+c}}^{n+1}$
    increment time step
        $n$++
end
```

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

5.1 User interface

Strings are shown as coloured paths (also see figures in this section). The state u of the model is visualised using the vertical displacement of the paths. Bowed strings are shown in cyan on the top left. The bow is shown as a yellow rectangle and moves on interaction. The fretting position is shown as a yellow circle. Plucked strings are shown in purple i the top right, underneath which the sympathetic strings are shown in light green. The plate is shown in the bottom using a 18x8 grid of rectangles (clamped gridpoints 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 connected elements.

Lastly, all parameters that are controlled by the mouse - output-mix, plate stiffness, etc - are located in a column on the right side of the UI.

5.2 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 these strings. Our implementation consists of 2 bowed strings, 13 sympathetic strings and 5 plucked strings (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 the bow and the second is mapped to a fretting finger on the string controlling pitch. 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 with a maximum of $v_{\rm B} = 0.2$ m/s and the finger force is linked to the excitation function with a maximum of $F_{\rm B} = 100$ m/s^2. The other Sensel is subdivided into 5 sections mapped to the plucked strings.

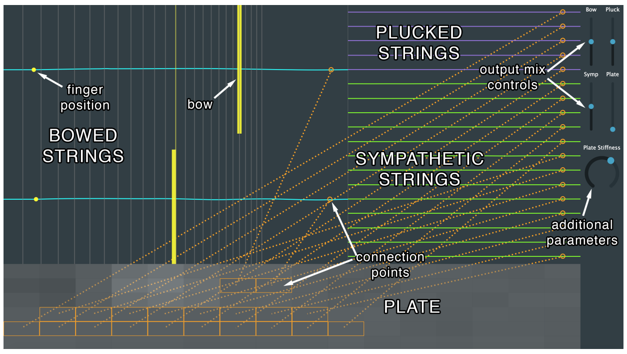

Figure 6. The bowed sitar application. The descriptions of the different elements and other objects are shown in the image, but will (naturally) not be visible in the application.

5.3 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. The excitation (which are plucked strings with a longer excitation length) One of each pair is connected to the plate which slightly detunes it.

5.4 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 base notes 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.

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 pitch (in the model this is the wave-speed c) of the melody-strings are changed by a midi controller.

6. RESULTS

Informal evaluation CPU usage

7. DISCUSSION

Body should not be plate. While our instruments have been not formally evaluated yet, noneless we 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. We did an informal test with a lecturer of the Rytmisk Center Kbenhavn The interaction with the bow has been found very natural. Bow force and velocity...

8. CONCLUSION AND FUTURE WORK

Speed up more: AVX vector such as [11] Model instrument bodies for more realistic sounds

Acknowledgments

We would like to thank Stefan Bilbao for his guidance during the course of this project.

9. REFERENCES

[1] L. Hiller and P. Ruiz, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part 1," *Journal of the Audio Engineering Society*, vol. 19, no. 6, pp. 462–470, 1971.

[2] 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.

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

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

[5] S. Bilbao, B. Hamilton, R. Harrison, and A. Torin, "Finite-difference schemes in musical acoustics: A tutorial." *Springer handbook of systematic musicology*, 2018.

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

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

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

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

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

[11] 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.