MODULAR PHYSICAL MODELS IN A REAL-TIME INTERACTIVE APPLICATION

Silvin Willemsen, Titas Lasickas, and Stefania Serafin
Multisensory Experience Lab, CREATE,
Aalborg University Copenhagen, Denmark
sil@create.aau.dk

ABSTRACT

Through recent advances in processing power, physical modelling using finite-difference time-domain (FDTD) methods has gained popularity. Many different musical instrument models based on these methods exist, and nearly all are based on the same underlying systems and interactions between them. This paper presents an application where individual resonator modules, such as strings, bars, membranes and plates, can be connected in a modular fashion and interacted with in real time. Various excitations, including the bow, hammer and pluck, are implemented as well, allowing for expressive control and a wide sonic palette. Existing and non-existing model configurations can easily be implemented, modified and experimented with, as well as the parameters describing them.

1. INTRODUCTION

Any acoustic musical instrument can be considered as the combination of a resonator and an exciter component [1]. Examples of resonator-exciter combinations are the violin and the bow, and the guitar and the pick. Many resonators, such as those mentioned here, can be further subdivided into basic components, i.e., a set of individual strings, a bridge and a wooden body. As many instruments consist of the same basic resonators – with different geometries or made from different materials – one can imagine some application that can implement many musical instruments based on the same fundamental resonator components in a modular fashion. Using physical modelling to implement these resonators allows for accurate implementation of the resonators, as well as the interactions between them.

Modularity in physical modelling sound synthesis is by no means a new concept. The earliest example of a modular system for sound synthesis was due to Cadoz *et al.* [2], with the CORDIS system. CORDIS allows complex instruments to be created using simple mass-spring interactions. Later, modular systems based on mass-spring systems appeared in various publications [3–5]. Morrison and Adrien created Mosaic [6], a modular environment using modal synthesis [7], more recently used by van Walstijn *et al.* in [8] and the Sound Design Toolkit [9]. Rabenstein *et al.* presented block-based modular physical models using wave digital filters [10] and digital waveguides [11].

Finite-difference time-domain (FDTD) methods, first used in a musical context by Ruiz [12], Hiller and Ruiz [13, 14] and later by Chaigne [15], also lend themselves to modularity. In [16], Bilbao presents a modular environment where bars and plates are connected by nonlinear springs, and in [17] Bilbao *et al.* propose a modular environment including higher dimensional systems.

Due to recent increase in computational power, FDTD methods have gained popularity in real-time applications [18]. A real-time modular environment using strings and lumped objects is presented in [19], and Südholt *et al.* present a real-time implementation of connected strings and bars using the FAUST programming language in [20].

The main goal of the aforementioned literature on FDTD-based modular environments is to create arbitrary sound-generators based on physical equations of motion, rather than modelling existing instruments. Furthermore, those able to run in real time only include systems spatially distributed over a maximum of one dimension. Many instruments require higher-dimensional structures to more accurately reproduce their sound. As done in e.g., [21, 22], the body of stringed instruments can be simplified to a thin plate, which is a system distributed over two dimensions.

This work presents an interactive modular environment in a real-time application where FDTD implementations of strings, bars, membranes and plates can be connected and played by the user. Although it is still possible to create arbitrary configurations to build non-existing instruments, the focus of this contribution is to allow for the possibility of modelling existing instruments. The current work aims to generalise previous work done on musical instruments modelled using FDTD methods (see e.g. [21, 22]), such that these can all be easily implemented as 'presets' of a modular application. Furthermore, some novel additions to the formulations of the hammer and pluck excitations are presented, required for their real-time interactions with the various resonators. The ultimate goal of this work is to act as a sound engine for a virtual reality application, potentially allowing for a more natural interaction.

This paper is structured as follows: Section 2 describes the physical models used in this work and Section 3 describes how to implement these. Section 4 presents the real-time application, Section 5 presents results and discusses these, and concluding remarks appear in Section 6.

2. MODELS

Consider a state variable $q(\boldsymbol{x}, t)$ with time $t \geq 0$ (in s) and spatial coordinate $\boldsymbol{x} \in \mathcal{D}$, where the dimensions and definition of domain $\mathcal{D}$ depend on the system at hand. The dynamics of a system can then be written using the following general form:

$$\mathcal{L}q = 0, \tag{1}$$

where linear partial differential operator $\mathcal{L}$ describes the dynamics of a model in isolation.

2.1 1D Systems

A commonly used 1D model (see e.g. [19, 21]) is the damped stiff string (or string for short). With reference to Eq. (1), consider a string of length L (in m), its transverse displacement described by state variable $q = u(\chi, t)$ (in m), and spatial coordinate χ is defined over domain $\mathcal{D} = [0, L]$. Furthermore, $\mathcal{L} = \mathcal{L}^{(1)}$ is defined as [23]

$$\mathcal{L}^{(1)} = \rho A \partial_t^2 - T\partial_\chi^2 + EI\partial_\chi^4 + 2\sigma_0 \rho A \partial_t - 2\sigma_1 \rho A \partial_t \partial_\chi^2, \tag{2}$$

where ∂_t and ∂_χ denote partial differentiation with respect to time and space respectively. The model is parameterised by material density ρ (in kg/m^3), cross-sectional area $A = \pi r^2$ (in m^2), radius r (in m), tension T (in N), Young's modulus E (in Pa), area moment of inertia $I = \pi r^4/4$ (in m^4) and loss coefficients $\sigma_0 \geq 0$ (in s^{-1}) and $\sigma_1 \geq 0$ (in m^2/s). If $T = 0$, the model reduces to a damped bar and the (damped) 1D wave equation if $E = 0$. Note that a circular cross-section is assumed here. Finally, in this work, the boundary conditions are chosen to be simply supported as

$$u = \partial_\chi^2 u = 0, \quad \text{for} \quad \chi = 0, L. \tag{3}$$

2.2 2D Systems

Consider a rectangular stiff membrane (or membrane for short) with side lengths L_x and L_y (both in m). With reference to Eq. (1), its transverse displacement can be described by $q = w(x, y, t)$ (in m), which is defined for domain $\mathcal{D} = [0, L_x] \times [0, L_y]$, and $\mathcal{L} = \mathcal{L}^{(2)}$ is defined as [24]

$$\mathcal{L}^{(2)} = \rho H \partial_t^2 - T\Delta + D\Delta\Delta + 2\sigma_0 \rho H \partial_t - 2\sigma_1 \rho H \partial_t \Delta, \tag{4}$$

Here, $\Delta = \partial_x^2 + \partial_y^2$ is the Laplacian, and parameters are material density ρ (in kg/m^3), thickness H (in m), tension per unit length T (in N/m), stiffness coefficent $D = EH^3/12(1-\nu^2)$ (in kg $\cdot$ m$^2\cdot$s^{-2}), Young's modulus E (in Pa), dimensionless Poisson's ratio ν, and loss coefficients $\sigma_0 \geq 0$ (in s^{-1}) and $\sigma_1 \geq 0$ (in m^2/s). If $T = 0$, the model reduces to a thin plate, and if $D = 0$ it reduces to the (damped) 2D wave equation, which can be used to model a non-stiff membrane. For simplicity, boundary conditions are chosen to be clamped, such that

$$w = \mathbf{n} \cdot \nabla w = 0, \tag{5}$$

where ∇ denotes 'the gradient of', and $\mathbf{n}$ is a normal to the plate area at the boundary.

2.3 Connections

One can add connections between instances of the models presented above by extending the general form in Eq. (1). Also see Figure 1. Consider M models q_m indexed by $m \in \mathfrak{M}$, where $\mathfrak{M} = \{1, \ldots, M\}$ and C connections between them. A connection is indexed by $c \in \mathfrak{C}$ with $\mathfrak{C} = \{1, \ldots, C\}$, and is characterised by the indices of the models it connects – $r_c \in \mathfrak{M}$ and $s_c \in \mathfrak{M}$ – and the locations where these models are connected – $\boldsymbol{x}_{r,c} \in \mathcal{D}_{r_c}$ and $\boldsymbol{x}_{s,c} \in \mathcal{D}_{s_c}$. Here, domains $\mathcal{D}_{r_c}$ and $\mathcal{D}_{s_c}$ are the (spatial) domains that models q_{r_c} and q_{s_c} are defined for.

Model q_{r_c} will be placed 'below' q_{s_c} such that the connection force acts positively on the former and negatively on the latter. The general form in Eq. (1) can then be extended to include connections according to

$$\mathcal{L}_m q_m = \sum_{\substack{c \in \mathfrak{C} \\ r_c = m}} \delta(\boldsymbol{x}_m - \boldsymbol{x}_{r,c}) f_c - \sum_{\substack{c \in \mathfrak{C} \\ s_c = m}} \delta(\boldsymbol{x}_m - \boldsymbol{x}_{s,c}) f_c, \tag{6}$$

where $f_c = f_c(t)$ is the force (in N) of the c^{th} connection.

The simplest connection considered in this work is the rigid connection, which assumes that the connected systems have an identical displacement at their respective connection locations. Alternatively, as done in [16, 24], one can use a spring to connect two systems. A connection force due to a nonlinear damped spring is

$$f_{\mathrm{c}} = K_{1,c}\eta_c + K_{3,c}\eta_c^3 + R_c \partial_t \eta_c, \tag{7}$$

with is a linear spring coefficient $K_{1,c} \geq 0$ (in N/m), nonlinear spring coefficient $K_{3,c}$ (in N/m^3), and damping coefficient $R_c \geq 0$ (in s^{-1}) of the c^{th} connection. If $K_{3,c} = 0$, Eq. (7) reduces to a linear spring. Furthermore, $\eta_c = \eta_c(t) = q_{s_c}(\boldsymbol{x}_{s,c}, t) - q_{r_c}(\boldsymbol{x}_{r,c}, t)$ is the relative displacement of the two systems at their respective connection locations (in m). Section 3.2 will elaborate on how to calculate the connection forces for all cases in discrete time.

To illustrate, consider two models ($M = 2$), a string and a membrane, such that $q_1 = u(\chi, t)$ and $q_2 = w(x, y, t)$, and a single connection between them ($C = 1$) at unspecified locations $\chi_{\mathrm{c}} \in \mathcal{D}_{s_1}$ and $(x_{\mathrm{c}}, y_{\mathrm{c}}) \in \mathcal{D}_{r_1}$ respectively. See Figure 1. Placing the string above the membrane, we get that $s_1 = 1$ and $r_1 = 2$, i.e., the 'above' model of connection 1 has index 1, and the 'below' model of connection 1 has index 2. Substituting the partial differential operators for the string and membrane found in Eqs. (2) and (4) respectively, Eq. (6) becomes, for the string and the membrane respectively

$$\mathcal{L}^{(1)} u = -\delta(\chi - \chi_{\mathrm{c}}) f_1, \tag{8a}$$

$$\mathcal{L}^{(2)} w = \delta(x - x_{\mathrm{c}}, y - y_{\mathrm{c}}) f_1. \tag{8b}$$

One can apply a rigid connection to Eq. (8a) according to [24]:

$$u(\chi_{\mathrm{c}}, t) = w(x_{\mathrm{c}}, y_{\mathrm{c}}, t). \tag{9}$$

If instead a spring connection is chosen,

$$f_1 = K_{1,1}\eta_1 + K_{3,1}\eta_1^3 + R_1 \partial_t \eta_1, \tag{10}$$

with $\eta_1 = \eta_1(t) = u(\chi_c, t) - w(x_c, y_c, t)$.

2.4 Excitations

In this work, as excitations will only be applied to 1D systems, the state variable of the stiff string, $u(\chi, t)$ will be used for the presentation of the various excitations. For the string, the general form in Eq. (1) can thus be extended to (ignoring connections for now)

$$\mathcal{L}^{(1)} u = e_\text{e} f_\text{e}, \tag{11}$$

where $e_\text{e} = e_\text{e}(\chi)$ is an excitation distribution and $f_\text{e} = f_\text{e}(t)$ is the externally supplied excitation force (in N).

2.4.1 The Bow

As done in previous work, see e.g. [21], one can include a bowing interaction by introducing a static friction model. With reference to Eq. (11), the excitation force can be defined using the following friction model [24]

$$f_\text{e} = -f_\text{B} \sqrt{2a_\text{B}} v_\text{rel} e^{-a_\text{B} v_\text{rel}^2 + 1/2}, \tag{12}$$

with externally supplied bow force $f_\text{B} = f_\text{B}(t)$ (in N), dimensionless free parameter a_B and

$$v_\text{rel} = \partial_t u(\chi_\text{B}, t) - v_\text{B} \tag{13}$$

is the relative velocity (in m/s) between the string at externally supplied bowing location $\chi_\text{B} = \chi_\text{B}(t)$ (in m) and the externally supplied bow velocity $v_\text{B} = v_\text{B}(t)$ (in m/s). Furthermore, the excitation distribution is set to be a single point along the string:

$$e_\text{e} = \delta(\chi - \chi_\text{B}). \tag{14}$$

2.4.2 Hammer

Another way of exciting a system is to use a hammer, which can be modelled as a simple mass-spring-damper system with a few proposed additions to allow for real-time interaction. In this work, state variable $z = z(t)$ is used to describe the displacement of the hammer from its equilibrium position (in m). The interaction between the hammer and the string is then modelled as a collision, using the following collision potential [25]:

$$\phi(\eta_\text{e}) = \frac{K_\text{e}}{\alpha_\text{e} + 1} [\eta_\text{e}]_+^{\alpha_\text{e}+1}, \tag{15}$$

with collision stiffness $K_\text{e} \geq 0$ (in N/m$^{\alpha_\text{e}}$) and nonlinear collision coefficient $\alpha_\text{e} \geq 1$. Furthermore, $[\eta_\text{e}]_+ = 0.5(\eta_\text{e} + |\eta_\text{e}|)$ describes the 'positive part of η_e' and

$$\eta_\text{e} = \eta_\text{e}(t) = \theta\big(u(\chi_\text{e}, t) - z(t)\big) \tag{16}$$

is the relative displacement between the string at collision location χ_e (in m) and the hammer (in m). Here, we propose $\theta = \tau$ if the string should be excited from above, and $\theta = -\tau$ if it should be excited from below, where $\tau = 1$ if the hammer interaction is triggered by the user and $\tau = 0$ if not (see Section 3.3.2).

Using a change of variables based on energy quadratisation proposed by Lopes *et al.* in [26] and used for collisions in a musical acoustics context in [27], one can rewrite the collision potential as

$$f_\text{e} = -\theta \psi \psi', \tag{17}$$

where $\psi = \psi(\eta) = \sqrt{2\phi}$, and using dots to denote a temporal derivative, $\psi' = \dot{\psi}/\dot{\eta}$. This change of variables ultimately allows for an explicit implementation of the nonlinear collision.

The dynamics of the hammer, including the collision with the string, can be described by the following PDE:

$$M_\text{z} \partial_t^2 z = -K_\text{z}\Big(z - (1-\tau) z_\text{off}\Big) - R_\text{z} \partial_t z + \theta \psi \psi', \tag{18}$$

with mass M_z (in kg), spring constant K_z (in N/m), loss coefficient R_z (in kg/s) and $z_\text{off} = z_\text{off}(t)$ is an externally supplied offset (in m) (also see [22]). The offset is used to keep the mass away from the equilibrium (and thus the system it excites) when controlling the application without wanting to excite the system. If the hammer excitation is triggered, and thus $\tau = 1$, the offset will no longer have an effect on the mass. The spring force then pulls the hammer towards the system, colliding with it in the process. After the collision, $\tau = 0$, and the offset will affect the mass again to avoid continuous collision between the hammer and the system.

Finally, with reference to Eq. (11), the excitation distribution is set to be a raised cosine with centre location χ_e and excitation width e_w (in m):

$$e_\text{e}(\chi) = \begin{cases} \frac{1-\cos\left(\frac{2\pi(\chi-\chi_\text{e})}{e_\text{w}} + \pi\right)}{2}, & \text{if } \chi_\text{e} - \frac{e_\text{w}}{2} \leq \chi \leq \chi_\text{e} + \frac{e_\text{w}}{2}, \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

Note that χ_e and e_w must be chosen such that $\chi_\text{e} - \frac{e_\text{w}}{2} \in \mathcal{D}$ and $\chi_\text{e} + \frac{e_\text{w}}{2} \in \mathcal{D}$, where $\mathcal{D}$ is the domain of the excited system.

2.4.3 Pluck

The pluck is modelled nearly the same way as the hammer. The main difference is that $\tau = 1$ until the collision force is larger than a certain value, after which it will be set to 0. This will be elaborated on in Section 3.3. See Figure 1 for an example of the plucking interaction with a (connected) string.

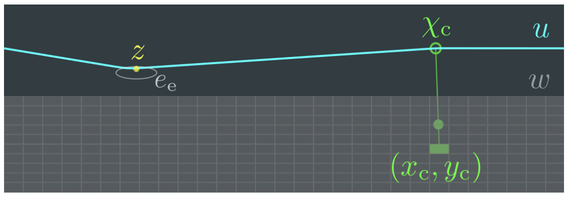

Figure 1. Snapshot of the application including a string and a thin plate with a rigid connection between them. The string is excited using a pluck.

3. DISCRETE TIME

In order to implement the models described in Section 2 using FDTD methods, a spatio-temporal grid needs to be defined [24]. For all models, time is discretised to $t = nk$ with time index $n = 0, 1, 2\dots$ and time step $k = 1/f_\text{s}$ (in s) where f_s is the sample rate (in Hz). For the 1D systems, space is subdivided into N equal intervals of length h_s (in m) according to $\chi = ph$ with spatial index $p \in \{0, \dots, N\}$. In the case of the 2D systems, the spatial coordinate is discretised as $(x, y) = (lh, mh)$ where spatial indices $l \in \{0, \dots, N_x\}$ and $m \in \{0, \dots, N_y\}$. Here, N_x and N_y are the number of intervals in the x and y direction respectively. Notice that the same value for grid spacing h is used for both the x and y directions.

Using these definitions, the general state variable $q(\boldsymbol{x}, t)$ can be approximated to grid function $q_{\boldsymbol{l}}^n$, where for the 1D systems $\boldsymbol{l} = p$ yielding grid function u_p^n and for the 2D systems, $\boldsymbol{l} = (l, m)$ yielding grid function $z_{(l,m)}^n$.

One of the main concerns when working with FDTD methods is the stability of the scheme. The stability condition for the discrete models can be described in terms of the grid spacing h as [18]

$$h \geq \sqrt{\beta\left(c^2k^2 + 4\sigma_1 k + \sqrt{(c^2k^2 + 4\sigma_1 k)^2 + 16\kappa^2k^2}\right)} \tag{20}$$

where for the string $c^2 = T/\rho A$, $\kappa^2 = EI/\rho A$ and $\beta = 0.5$, and for the membrane $c^2 = T/\rho H$, $\kappa^2 = D/\rho H$ and $\beta = 1$. The number of intervals between grid points can then be calculated as $N = \lfloor L/h \rfloor$ for 1D systems and $N_x = \lfloor L_x/h \rfloor$ and $N_y = \lfloor L_y/h \rfloor$ for 2D systems, where $\lfloor \cdot \rfloor$ denotes the flooring operation. Including the boundaries, discrete systems will have $N + 1$ and $(N_x + 1)(N_y + 1)$ grid points for the 1D and 2D case respectively.

3.1 FDTD schemes

The general form in Eq. (1) can be discretised to the following FDTD scheme:

$$\ell q_{\boldsymbol{l}}^n = 0 \tag{21}$$

where ℓ is the discretised version of linear partial differential operator $\mathcal{L}$. The discrete-time definitions of Eqs. (2) and (4) will not be given here for brevity, but can be found in the literature (e.g. [24], [18]). However, for any explicit FDTD scheme (which are used in this work), one can expand Eq. (21) to yield an update equation of the form

$$a q_{\boldsymbol{l}}^{n+1} = b q_{\boldsymbol{l}}^n + c q_{\boldsymbol{l}}^{n-1} \tag{22}$$

where a, b and c depend on the system at hand. In the following, we assume that $a = (1 + \sigma_0 k)$ for any discretised system.

3.2 Connections

To apply the effect of connections to FDTD schemes, interpolation and spreading operators must be introduced. As in Section 2.3, consider M models where the grid function of the mth model is $q_{m,\boldsymbol{l}_m}^n$. One can define a (zeroth-order) interpolation operator as

$$I_{m,\boldsymbol{l}_m}(\boldsymbol{x}_m) = \begin{cases} 1, & \text{if } \boldsymbol{l}_m = \lfloor \boldsymbol{x}_m / h_m \rfloor, \\ 0, & \text{otherwise,} \end{cases} \tag{23}$$

which can be applied to grid function $q_{m,\boldsymbol{l}_m}^n$ to to obtain the state of one grid point of the system. A spreading operator can then be defined as

$$J_{m,\boldsymbol{l}_m}(\boldsymbol{x}_m) = \frac{1}{(h_m)^\varepsilon} I_{m,\boldsymbol{l}_m}(\boldsymbol{x}_m), \tag{24}$$

where ε is the number of spatial dimensions the system is defined over ($\varepsilon = 1$ for 1D systems, $\varepsilon = 2$ for 2D systems). Equation (24) can then be used to localise a (connection) force onto a specific location along a system. The general form in Eq. (6) can be discretised to

$$\ell_m q_{m,\boldsymbol{l}_m}^n = \sum_{\substack{c\in\mathfrak{C} \\ r_c = m}} J_{m,\boldsymbol{l}_m}(\boldsymbol{x}_{r,c}) f_c^n - \sum_{\substack{c\in\mathfrak{C} \\ s_c = m}} J_{m,\boldsymbol{l}_m}(\boldsymbol{x}_{s,c}) f_c^n, \tag{25}$$

Assuming that none of the connections overlap one can solve for each force f_c^n individually.

One can express the relative distance between two models (q_{s_c} and q_{r_c}) connected by connection c as

$$\eta_c^n = I_{s_c,\boldsymbol{l}_{s_c}}(\boldsymbol{x}_{s,c}) q_{s_c,\boldsymbol{l}_{s_c}}^n - I_{r_c,\boldsymbol{l}_{r_c}}(\boldsymbol{x}_{r,c}) q_{r_c,\boldsymbol{l}_{r_c}}^n. \tag{26}$$

If a rigid connection is chosen, the force of connection c can be solved for according to [24]

$$f_c^n = \frac{\eta_c^\star}{\frac{k^2}{a_{r_c}\mathcal{M}_{r_c}} + \frac{k^2}{a_{s_c}\mathcal{M}_{s_c}}}, \tag{27}$$

where $\mathcal{M}_m$ is the effective mass of a single grid point (in kg) of model q_m, i.e., $\mathcal{M} = \rho A h$ for 1D systems, and $\mathcal{M} = \rho H h^2$ for 2D systems and $a_m = (1 + \sigma_{0,m} k)$. Furthermore,

$$q_{m,\boldsymbol{l}_m}^\star = \frac{b_m q_{m,\boldsymbol{l}_m}^n + c_m q_{m,\boldsymbol{l}_m}^{n-1}}{a_m} \tag{28}$$

is the state of model q_m at the next time step in isolation, i.e., without the connection forces (obtained by rewriting Eq. (22)), and can be appropriately used in Eq. (26) to obtain $\eta_c^\star$.

Introducing the centred difference and centred averaging operator

$$\delta_{t\cdot}\eta^n = \frac{1}{2k}\left(\eta^{n+1} - \eta^{n-1}\right) \cong \eta, \tag{29}$$

$$\mu_{t\cdot}\eta^n = \frac{1}{2}\left(\eta^{n+1} + \eta^{n-1}\right) \cong \eta, \tag{30}$$

the spring in Eq. (7) can be discretised to

$$f_\text{c}^n = K_{1,\text{c}}\mu_{t\cdot}\eta_\text{c}^n + K_{3,\text{c}}(\eta_\text{c}^n)^2\mu_{t\cdot}\eta_\text{c}^n + R_\text{c}\delta_{t\cdot}\eta_\text{c}^n, \tag{31}$$

which can be shown to be inherently stable [16, 24]. The connection force can then be solved for according to [24]

$$f_c^n = \frac{\eta_c^\star + \frac{\gamma_{c,-}}{\gamma_{c,+}}\eta_c^{n-1}}{\frac{1}{\gamma_{c,+}} + \frac{k^2}{a_{r_c}\mathcal{M}_{r_c}} + \frac{k^2}{a_{s_c}\mathcal{M}_{s_c}}}, \tag{32}$$

where

$$\gamma_{c,\pm} = \frac{K_{1,c}}{2} + \frac{K_{3,c}(\eta_c^n)^2}{2} \pm \frac{R_c}{2k}. \tag{33}$$

Notice that the rigid connection force in Eq. (27) is a special case of Eq. (32) where $\eta_c^{n-1} = 0$ and $\gamma_{c,+} \to \infty$. See [18, Ch. 11] for a derivation and more details.

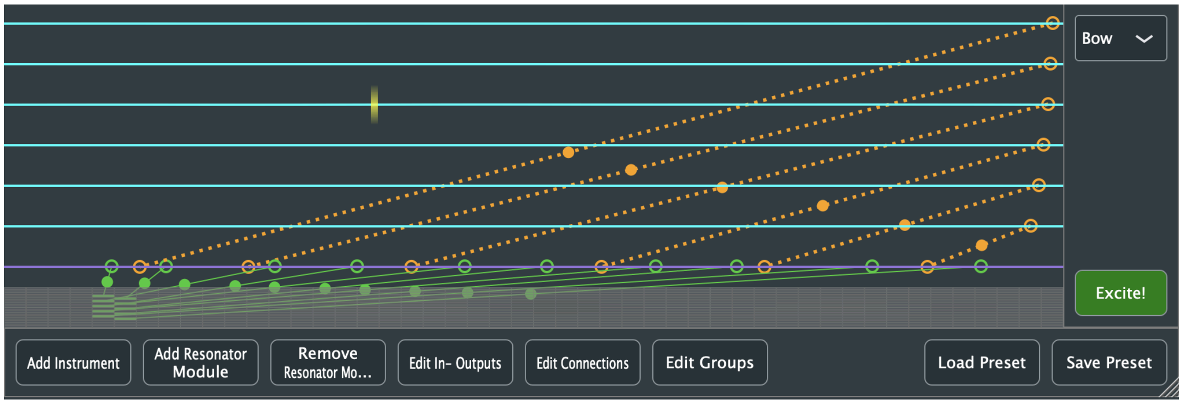

Figure 2. The graphical user interface (GUI) of the application presented in Section 4. Here, the guitar preset is loaded with 6 strings, 1 bar, 1 thin plate and several linear (orange) and rigid (green) connections between them, excited using a bow.

3.3 Excitations

3.3.1 The Bow

The discretisation of the friction model in Eq. (12) as well as its implementation using an iterative solver – such as Newton-Raphson – is well-covered in the literature (see e.g. [18, Ch. 8]). The spatial Dirac delta function in Eq. (14) is discretised using cubic spreading operator $J_{p,3}(\chi_{\rm i})$, rather than Eq. (24), for more accurate control. Its definition is also excluded here for brevity, but can be found in the literature [24, Sec. 5.2.4].

3.3.2 Hammer and Pluck

The discrete-time definition of Eq. (18) will not be given here. The discretisation of a mass-spring-damper system can be found in e.g. [16, 22], and a derivation of the collision between the mass-spring and a 1D system using the non-iterative collision method used here (from [27]) is given in [18, Sec. 10.2]. Changes in θ and τ (introduced in this work) are considered instantaneous and do not affect the derivations.

For the hammer, τ will be set to 0 if the user triggers this with the mouse (see Section 4.2.7). For the pluck, τ will be set to 0 if $|f_{\rm e}/h| > \varphi$, where φ is a threshold, which simulates a plucking interaction. Note that the force is scaled by the grid spacing of the respective resonator module so that the plucking interaction will be similar for strings with different parameters (and thus different values of h).

4. REAL-TIME APPLICATION

A real-time application implementing the models above has been created in C++ using the JUCE framework [1]. Figure 2 shows the graphical user interface of the application using the guitar preset for illustration. The application is controlled using the mouse, and is divided into three main parts: the control panel (bottom), the excitation panel (right), and the instrument area. The latter is vertically and equally subdivided over the amount of instruments in the application, which in turn are subdivided vertically and equally into the amount of resonators they contain. The instrument area has a refresh rate of 15 Hz and visualises the states of the various resonator and exciter modules.

After describing the system architecture, this section will go into detail of the functionality of the application. An extensive demo of the application, going through all of the functionality described in this section, can be found via [28].

4.1 System Architecture

Figure 3 illustrates the architecture of the audio-generating part of the application. The application can contain several independent instruments, each of which can contain several resonators. Furthermore, instruments contain information about the connections between various resonators. Every 1D resonator has three exciter modules, one for each of the excitations described in this paper. Only one of the exciter modules is activated at a time, controlled by the excitation panel (see Section 4.2.7).

4.2 Functionality

This section goes through functionality of the application following the order of the various buttons shown in the control panel (bottom area in Figure 2).

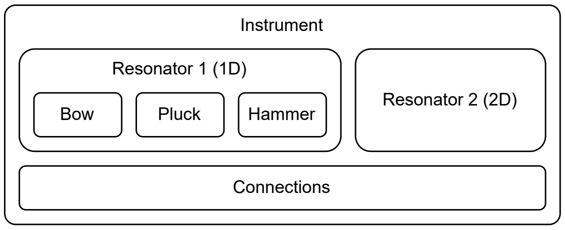

Figure 3. The system architecture. An instrument contains resonators as well as information about the connections between them. 1D resonators contain three exciter modules.

[1] `https://juce.com`

4.2.1 *Instruments*

The *Add Instrument* button adds an (additional) instrument to the application. Clicking on an instrument makes it the 'currently active instrument' to which resonator modules can be added.

4.2.2 *Resonator Modules*

The available resonators are: the stiff string, bar, membrane, thin plate and stiff membrane and are implemented as in Section 3.1. The states of the 1D models are visualised as cyan (string) and purple (bar) lines and the states of 2D models as grey-scale squares (as done in e.g. [21]).

The *Add Resonator Module* button opens a window where a user can select a resonator type as well as its parameters. For the stiff string, one can choose an advanced (all parameters) and non-advanced (only the fundamental frequency and radius) list of parameters. For 2D models, an extra parameter 'maxPoints' is given which alters the grid spacing such that the number of moving grid points, i.e., those that are not fixed at the boundaries, does not surpass this. This is to not overload the CPU and keep the application running in real time (also see Section 5). A button *Add Module*, adds the resonator module to the currently selected instrument.

The *Remove Resonator Modules* button changes the buttons in the control panel to *Remove* and *Done*. Clicking on a resonator module adds a red overlay, and clicking the *Remove* button removes the resonator from the instrument.

It must be noted that if any button is pressed, the states of all resonator modules will be set to 0 to prevent audible artefacts when editing the settings.

4.2.3 *Outputs*

The output of any model can be obtained by listening to $q^n_{\boldsymbol{l}_\text{o}}$ for an output location $\boldsymbol{l}_\text{o}$. In the application it is possible to change the output locations by clicking the *Edit In-Outputs* button. If the button is clicked, the control panel will show instructions on the chosen option as well as a *Done* button. The user can now add output locations to the various resonators.

For 1D systems, outputs will show as downwards pointing arrows from the system state at the respective output locations. For 2D systems, rectangles around the output grid point are used. Left, right and stereo channels are shown in white, red and yellow respectively. Functionality to add inputs has been left for future work (see Section 6).

4.2.4 *Connections*

Connections between resonators (implemented as in Section 3.2) are visualised using coloured (dotted) lines (also see Figure 2). Rigid connections are shown in solid green, linear springs in dotted orange and non-linear springs in dotted magenta. A connection location on a 1D system is accentuated using a circle of the same colour, and the same is done for a 2D system using a rectangle. A solid circle along the connection line indicates the mass ratio between the grid points of the two components: $\mathcal{M}_{s_c}/\mathcal{M}_{r_c}$ (see Eq. (27)). The closer this is to one component, the heavier a grid point of that component is with respect to the other.

If the *Edit Connections* button is clicked, the control panel will change in the same way as for the *Edit In- Outputs* button. Now, the currently active connection will be indicated by a yellow 'halo' around the connection locations. If a user tries to overlap two connections, the currently active connection will be removed from the application.

4.2.5 *Groups*

If the *Edit Groups* button is pressed, the control panel changes in the same way as for the previous two buttons. The user can click on 1D resonator modules to add them to groups (see [28] for more details). Grouped models can be excited simultaneously (see Section 4.2.7).

4.2.6 *Presets*

Presets are saved in '.xml' files and have the structure shown in Figure 4. Each preset contains several elements (instruments, resonators, connections, etc.), each containing one or more attributes.

Pressing the *Load Preset* button causes a 'File Chooser' window to pop up, where the user can select a local '.xml' file containing a preset. The *Save Preset* button makes a window pop up with a text box into which the desired name of the current application configuration can be saved. If a file with that name already exists a warning window will pop up asking whether the existing file may be overwritten.

As different sample rates yield different values for h (see Eq. (20)) and thus the number of grid points for each resonator, the locations of the outputs and connections are saved as ratios of the total number of grid points of the respective resonator. This allows for the same relative locations of outputs and connections for one preset between different sample rates.

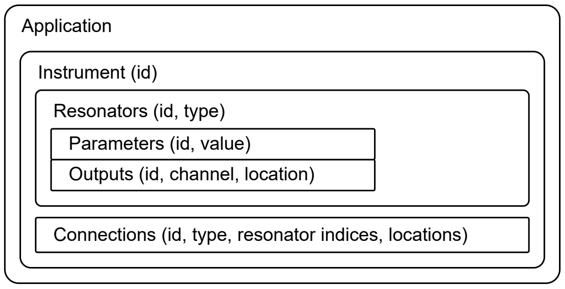
Application
Instrument (id)
Resonators (id, type)
Parameters (id, value)
Outputs (id, channel, location)
Connections (id, type, resonator indices, locations)

Figure 4. Structure of the '.xml' files containing presets. Attributes for several elements are given in parentheses.

4.2.7 *Excitations*

One can interact with the various 1D resonator modules in the instrument area. The excitation panel (right area in Figure 2) contains a dropdown menu with the various excitations: *Bow*, *Hammer* and *Pluck*. Selecting one of these activates the corresponding exciter module in all 1D resonator modules. Furthermore, an *Excite!* button toggles the currently active exciter module for all 1D resonator modules. Now a user can hover the mouse over the resonator modules in the instrument area (or click for the

Mouse action	Bow	Hammer & Pluck
x-position	χ_B	χ_e
y-position	-	z_off
Scroll-wheel	$-0.2 \leq v_\mathrm{B} \leq 0.2$	$h \leq e_\mathrm{w} \leq 10h$
Click	-	τ (hammer only)

Table 1. Mouse interactions related to the various excitations described in Section 4.2.7.

Name	Symbol (unit)	Value
Bow		
Free parameter	a_B (-)	100
Bow force	f_B (N)	$40 \cdot \rho A$
Hammer / pluck		
Mass	M_z (kg)	0.01
Spring coefficient	K_z (N/m)	1000
Loss coefficient	R_z (kg/s)	1
Collision stiffness	K_e (N/m$_\mathrm{e}^\alpha$)	10^6
Nonlin. coll. coeff.	α_e (-)	1.3
Pluck force threshold	φ (N/m)	500
Connections		
Linear spring coeff.	K_1 (N/m)	10^8
Nonlin. spring coeff.	K_3 (N/m^3)	10^{10}
Damping coeff	R (s^{-1})	0.01

Table 2. List of fixed parameter values (see Section 4.3).

hammer) to excite them. If the excitation is deactivated, all resonators (including 2D ones) can be excited using a raised cosine (although this is only used for testing purposes). Table 1 shows how the mouse is related to various excitation parameters.

The bow is visualised with a yellow rectangle with a moving gradient, the speed and direction of which depends on the value of the bow velocity (v_B in Eq. (13)). The mass used for the hammer and pluck (z in Eq. (18)), is visualised using a yellow circle and follows the mouse / cursor. The excitation distribution (e_e in Eq. (19)) is visualised using a grey ellipse around the cursor; its width is determined by the value of e_e. For the hammer, a mouse click sets $\tau = 0$ in Eq. (18).

4.3 Fixed parameters

As described in Section 4.2.2, the parameters of the resonator modules can easily be altered by the user when adding one to the application. The parameters found in Table 2, however, can not be altered by the user, and have been set to result in pleasing sounds for many different model configurations.

5. RESULTS AND DISCUSSION

Table 3 shows the CPU usage of the application with various settings and graphics turned off and on. Tests were carried out on a MacBook Pro with a 2,3 GHz Intel i9 processor. Results show that for many different configurations, the application is still able to run in real time (usage $< 100\%$).

Strings ($N = 118$)				
No. of strings	1	5	10	20
No graphics	1.7	7.4	12.3	26.8
Graphics	8.3	14.0	19.4	35.0
Plate				
Moving points	100	400	1600	2500
No graphics	4.8	15.4	47.0	63.3
Graphics	12.0	23.9	59.2	77.5
Two connected strings ($N = 118$, $C = 117$)				
Type	None	Rigid	Linear	Nonlinear
No graphics	3.8	8.3	13.1	13.3
Graphics	12.7	28.4	42.3	43.1

Table 3. CPU usage (in %) of the application with various configurations, and graphics turned on and off.

One can observe that the CPU increases with the number of grid points that need to be calculated per sample, which is an expected result. As more calculations need to be performed per sample for a 2D system the CPU usage is higher per grid point than for a 1D system. For two strings with all their moving grid points connected ($N = 118$, $C = 117$), the CPU usage increases significantly when compared to unconnected strings, and more so for non-rigid connections. Moreover, the graphics increase the CPU usage substantially, especially when connections need to be drawn. As the speed of the graphics thread has not been the focus during the development of the application, this could be optimised in the future.

To ensure that the application runs in real time, several things need to be considered. One example is the 'maxPoints' variable, limiting the number of grid points for 2D systems mentioned in Section 4.2.2. Using fewer grid points reduces the simulation quality and the frequency bandwidth of the model at hand [24]. Furthermore, it has been chosen to only use explicit models, which, although more computationally cheap, cause detuning of modes in models with high stiffness values such as bars and plates [24]. These effects, however, have been found to be less important for stringed instrument models, due to the large amount of damping of bars and plates and the inharmonic nature of models with high stiffness.

6. CONCLUSION

This paper presented an application implementing various resonator modules using FDTD schemes which can be connected in a modular fashion to create (non)existing musical instruments. The instruments can be played in real time using bowing, hammering and plucking excitations.

Future work could include to embed this contribution in a virtual reality application where users will be able to build and play instruments in a virtual environment.

Acknowledgments

This work has been funded in part by the European Art-Science-Technology Network for Digital Creativity (EASTN-DC), project number 883023.

7. REFERENCES

[1] G. Borin, G. De Poli, and A. Sarti, "A modular approach to excitator-resonator interaction in physical models syntheses," *Proc. Int. Comp. Music Conf.*, 1989.

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

[3] N. Castagné and C. Cadoz, "GENESIS: A friendly musician-oriented environment for mass-interaction physical modeling," in *Proc. Int. Comp. Music Conf.*, 2002, pp. 330–337.

[4] E. Berdahl and J. Smith, "An introduction to the synth-a-modeler compiler: Modular and opensource sound synthesis using physical models," in *Proc. 10th Int. Linux Audio Conf. (LAC-12)*, 2012, pp. 223–228.

[5] J. Leonard and J. Villeneuve, "MI-GEN~: An efficient and accessible mass interaction sound synthesis toolbox," in *Proc. 13th Int. Conf. Sound Music Comp. (SMC)*, 2019.

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

[7] J.-M. Adrien, "The missing link: Modal synthesis," in *Representations of Musical Signals*, G. De Poli, A. Picalli, and C. Roads, Eds. MIT Press, 1991, pp. 269–298.

[8] M. van Walstijn and S. Mehes, "An explorative string-bridge-plate model with tunable parameters," in *Proc. 20th Int. Conf. on Digital Audio Effects (DAFx)*, 2017.

[9] S. Baldan, S. Delle Monache, and D. Rocchesso, "The sound design toolkit," *SoftwareX*, vol. 6, pp. 255–260, 2017.

[10] R. Rabenstein, S. Petrausch, A. Sarti, G. D. Sanctis, C. Erkut, and M. Karjalainen, "Block-based physical modeling for digital sound synthesis," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 42–54, 2007.

[11] J. O. Smith, "Physical modeling using digital waveguides," *Comp. Music J.*, vol. 16, no. 4, pp. 74–91, 1992.

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

[13] L. Hiller and P. Ruiz, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part I," *J. Acoust. Soc. Am.*, vol. 19, no. 6, pp. 462–470, 1971.

[14] ——, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part II," *J. Acoust. Soc. Am.*, vol. 19, no. 7, pp. 542–550, 1971.

[15] 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.

[16] S. Bilbao, "A modular percussion synthesis environment," in *Proc. 12th Int. Conf. on Digital Audio Effects (DAFx)*, 2009.

[17] S. Bilbao, A. Torin, P. Graham, J. Perry, and G. Delap, "Modular physical modeling synthesis environments on GPU," in *Proc. ICMC|SMC|2014*, 2014, pp. 1396–1403.

[18] S. Willemsen, "The emulated ensemble: Real-time simulation of musical instruments using finite-difference time-domain methods," Ph.D. dissertation, Aalborg University Copenhagen, Jul. 2021.

[19] S. Bilbao, M. Ducceschi, and C. Webb, "Large-scale real-time modular physical modeling sound synthesis," in *Proc. 22th Int. Conf. on Digital Audio Effects (DAFx)*, 2019.

[20] D. Südholt, R. Russo, and S. Serafin, "A faust implementation of coupled finite difference schemes," in *Proc. 2nd Nordic Sound and Music Comp. (Nordic-SMC) Conf.*, 2021.

[21] S. Willemsen, N. Andersson, S. Serafin, and S. Bilbao, "Real-time control of large-scale modular physical models using the sensel morph," *Proc. 16th Int. Conf. Sound Music Comp. (SMC)*, pp. 275–280, 2019.

[22] S. Willemsen, S. Serafin, S. Bilbao, and M. Ducceschi, "Real-time implementation of a physical model of the tromba marina," in *Proc. 17th Int. Conf. Sound Music Comp. (SMC)*, 2020, pp. 161–168.

[23] J. Bensa, S. Bilbao, R. Kronland-Martinet, and J. O. Smith, "The simulation of piano string vibration: From physical models to finite difference schemes and digital waveguides," *J. Acoust. Soc. Am.*, vol. 114, no. 2, pp. 1095–1107, 2003.

[24] S. Bilbao, *Numerical Sound Synthesis*. John Wiley & Sons, 2009.

[25] H. Hertz, "Über die berührung fester elastischer körper," *Journal für die Reine und Angewandte Mathematik*, vol. 92, pp. 156–171, 1881.

[26] N. Lopes, T. Hèlie, and A. Falaize, "Explicit second-order accurate method for the passive guaranteed simulation of port-hamiltonian systems," in *Proc. 5th IFAC*, 2015, pp. 223–228.

[27] M. Ducceschi, S. Bilbao, S. Willemsen, and S. Serafin, "Linearly-implicit schemes for collisions in musical acoustics based on energy quadratisation," *J. Acoust. Soc. Am.*, vol. 149, pp. 3502–3516, 2021.

[28] S. Willemsen, "Modular instrument demo," 2021. [Online]. Available: https://youtu.be/o6I2DlFIbsc