An Environment for Physical Modeling of Articulated Brass Instruments

An Articulated Brass Instrument Synthesis Environment

Reginald Langford Harrison[a], Stefan Bilbao[b], James Perry[c], Trevor Wishart[d]

[a] Acoustics and Audio Group, University of Edinburgh, Rm 1606, James Clerk Maxwell Building, Edinburgh, EH9 3JZ UK, r.l.harrison-3@sms.ed.ac.uk

[b] Acoustics and Audio Group, University of Edinburgh, Rm 1602, James Clerk Maxwell Building, Edinburgh, EH9 3JZ UK, s.bilbao@ed.ac.uk

[c] Edinburgh Parallel Computing Centre, University of Edinburgh, James Clerk Maxwell Building, Edinburgh, EH9 3JZ UK, j.perry@epcc.ed.ac.uk

[d] Department of Music, University of Durham, Palace Green, Durham, DH1 3RL, trevor.wishart@durham.ac.uk

Abstract

This article presents a physical modeling synthesis environment for valved brass instrument sounds. Synthesis is performed using finite-difference time-domain methods which allow for flexible simulation of time varying systems. Users have control over the instrument configuration as well as player parameters such as mouth pressure, lip dynamics and valve depressions which can be varied over the duration of a gesture. This article introduces the model used in the environment, the development of code from prototyping in MATLAB and optimisation in C and then incorporation of the executable file in the Soundloom interface of the Composers Desktop project. Planned additions to the environment are then discussed. The environment binaries are available to download online along with example sounds and input files.

Introduction

Sound synthesis for brass instruments is a subject of longstanding interest, dating back at least as far as the early additive synthesis experiments of Risset (1966), and the application of FM methods, presented by Morrill (1977) in the very first issue of the Computer Music Journal. Such synthesis methods may be characterised as abstract—they are signal based and lacking in an underlying physical representation. Although they may constitute relatively efficient means of producing isolated tones which resemble those of brass instruments, progress towards a system capable of emulating the subtlety and full expressive range of acoustic brass instruments is difficult.

Physical modeling techniques address this difficulty, and various techniques have emerged. Perhaps the most successful have been digital waveguides (Smith 1986, 2004), which were first used in the context of brass instrument synthesis by Cook (1991). These efficient methods are based on the use of digital delay lines to model wave propagation in tubes of cylindrical or conical cross section. Waveguide methods developed to the point that they were incorporated into the Yamaha VL1 synthesizer in 1994 (Russ 1994); further work along these lines has included the finer modeling of bore profiles through the concatenation of multiple waveguide-like components (Berners 1999; van Walstijn and Campbell 2003; Mignot et al. 2010). To model valve gestures, a simple waveguide approach involves truncating the delay lines to change the instrument length but this does not include the effects from other interacting tube sections. Other modeling methods are based upon the use of modal representations for the instrument bore (Silva et al. 2014). In all cases, the basic breakdown of the model into source and resonator components follows the canonical description given by McIntyre et al. (1983). Extensions to the case of nonlinear wave propagation leading to shock formation and increased timbral brightness at high amplitudes have been incorporated into work on the BRASS project (Vergez and Tisserand 2006).

All the methods mentioned above rely on simplifying assumptions, in order to obtain high computational speeds. In particular, in the digital waveguide formalism, a possibly complex bore profile must be approximated by simplified cylindrical or conical segments, and in a modal formalism, the set of modes corresponding to a given bore profile must be computed beforehand. This can mean that complex time-variation in the instrument body itself, as in the case of valved gestures, can become awkward. Given that computational power is now abundant, one way of proceeding is to appeal to more general numerical procedures which allow for complete modeling, with minimal simplifying assumptions. The approach taken here will be to make use of direct time-space integration techniques such as the *finite-difference time-domain* method (FDTD). Such methods have seen use in synthesis for some time, dating back to early work in string synthesis by Ruiz (1969) and Hiller and Ruiz (1971); recent work on FDTD for the brass instrument bore appears in Bilbao and Chick (2013), and synthesis employing the valve mechanism in Bilbao (2011). Though computational cost is higher than for other methods, runtimes for most reasonable instrument geometries are real or near-real-time.

This article is concerned with a synthesis environment for brass instruments, where the user has complete control over the instrument design (i.e., the bore profile, as well as the lengths and positions of valves), as well as several streams of control input: one set for the mouth excitation, and another for the valve displacements. Some background material on the basic operation of an FDTD scheme for the entire system is provided, followed by a description of the development of the environment from protoyping using MATLAB (MathWorks 2014) and then optimising in C, accompanied by benchmarking results. Control and user interface details are then discussed, with particular reference to use in ongoing experimental electroacoustic work. Finally, planned extensions to the environment are introduced.

Model

Sound production within a brass instrument can be broken down into three sections: resonator, generator, and radiator (Fletcher and Rossing 1998). The resonator consists of the acoustic bore and valve sections, and determines, grossly speaking, the natural frequencies of the instrument. The generator consists of the player's lips, driven into oscillation by the pressure difference between the mouth and the entrance of the instrument. The lips are therefore coupled to the instrument and oscillate near one of its natural resonances. The radiator of a brass instrument is the flaring section at the end of the instrument. This flaring section determines the efficiency of sound transfer from the instrument to the acoustic space and also alters the frequencies and bandwidths of the instrument's resonances.

Wave Propagation

Linear, lossless wave propagation within an acoustic tube can be described using Webster's equation (Webster 1919) and are valid when the tube cross section varies slowly and when the diameter of the tube is small relative to the wavelengths of interest. The first order conservation equations leading to Webster's equation can be written as:

$$\frac{S}{\rho c^2}\partial_t p = -\partial_x \left(Sv\right), \quad \rho \partial_t v = -\partial_x p \tag{1}$$

where $p(x,t)$ and $v(x,t)$ are the acoustic pressure and particle velocity as functions of time $t \in \mathbb{R}^+$, and axial coordinate $x \in [0, L]$, where L is the main bore length. $S(x)$ is the tube cross sectional area, ρ and c are the air density and speed of sound in air, and ∂_t and ∂_x represent first order differentiation with respect to time and the spatial coordinate. See Figure 1 for a representation of a generalized bore profile.

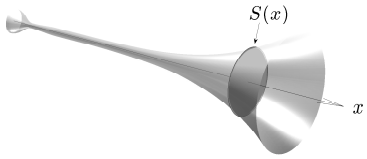

Figure 1. A generalized brass instrument bore with a variable cross sectional area, $S(x)$, that varies with spatial coordinate x.

To model energy losses associated with viscosity confined to a thin boundary layer inside the tube, a modified frequency domain model of Webster's equation is often used and which can be transformed into the time domain. In this work the model presented by Keefe (1984) for large tube radii is used as a starting point for further simplifications which lead to equations using half derivatives with respect to time (Bilbao and Chick 2013). Further simplifications of this type lead to the Webster-Lokshin formulation which is commonly modelled using digital waveguides (Hélie 2003).

Although brass instruments have complicated three dimensional geometries, this one dimensional description is valid for instruments like the trumpet and trombone and has shown good agreement with experimental measurements (Bilbao and Chick 2013; Eveno et al. 2012).

Modeling Valves

A simplified model of a valve consists of a main tube that joins two pieces of tubing, the default and bypass tubes, as shown in Figure 2. Under valve depression, the tubes are shifted relative to the main tube, thus partitioning the volume flow between the default and bypass sections. The default tube is the path the airflow can take when a valve is not depressed; the bypass tube is engaged when the valve is fully depressed. Usually the bypass tube is longer than the default path and therefore lowers the resonant

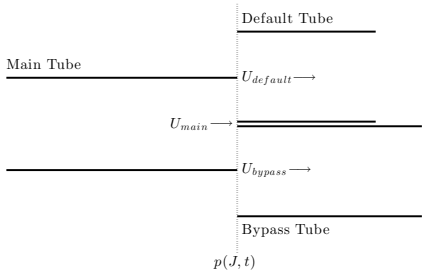

Figure 2. Schematic of a brass instrument valve. The dotted line denotes a junction connecting the main tube to default and bypass tubes at $x = J$. The pressure is the same at the junction in all three tubes and the signed volume velocities, $U(x,t) = S(x)v(x,t)$, at the junction sum to zero.

frequencies of the instrument when fully engaged. When the valve is partially open, two paths are available which, in general, creates a set of inharmonicaly spaced resonances, leading to the production of multiphonic tonal timbres.

The valve is modeled as a new boundary condition within the instrument linking pressure/velocity pairs in the three pieces of tubing. It is assumed that the pressure in all three tubes at the valve junction is the same and that the signed volume velocities of air sum to zero (Bilbao 2011), similarly to the case of junctions in the digital waveguide formulation (Smith 2004).

$$p_{main}\left(J,t\right) = p_{default}\left(0,t\right) = p_{bypass}\left(0,t\right), \quad U_{main}\left(J,t\right) = U_{default}\left(0,t\right) + U_{bypass}\left(0,t\right) \quad (2)$$

where the subscripts denote the tube, U is a volume velocity and J denotes the length along the main tube that the valve is positioned.

Lip Dynamics

The lips of a brass player can be considered as an outward striking reed. This can be modelled as a damped mass-spring system that is driven by the pressure difference between the player's mouth and the mouthpiece of the instrument (Adachi and Sato 1995), under assumptions of Bernoulli-type nonlinear flow. The lip dynamics are given by

$$\ddot{y} + \sigma\dot{y} + 4\pi^2 f_{lip}^2 y = S_r \Delta p/\mu \tag{3}$$

where y is the lip displacement form its equilibrium position H, the single and double dots denote first and second order derivatives with respect to time, σ and f_{lip} are the lips damping and natural resonance frequency, μ is the lip mass and S_r is the surface area of the lip which is acted on by the pressure difference, $\Delta p = p_m\left(t\right) - p\left(0,t\right)$, between the mouth and the entrance of the instrument. Air is injected into the instrument as a result of the pressure difference across the entrance to the mouthpiece when the lips are open and also due to the motion of the lips so that

$$S\left(0\right)v\left(0,t\right) = u_m + u_r, \qquad u_m = \max\left(y + H, 0\right)\operatorname{sign}\left(\Delta p\right) w\sqrt{2|\Delta p|/\rho}, \qquad u_r = S_r\dot{y} \tag{4}$$

where w is the channel width. These properties can all be used to couple the lip model to the instrument in a by-now standard manner (McIntyre et al. 1983).

Radiation

Radiation properties of a flared acoustic tube can be suitably described using a model of an unflanged cylindrical pipe that was presented by Levine and Schwinger (1948). Rational approximations can be applied to this model to ease the transformation to a discrete time domain setting (Silva et al. 2009). The radiation model is expressed by

the impedance

$$Z_{rad} = \frac{L_r \left(R_1 + R_2\right) j\omega + L_r R_1 R_2 C \left(j\omega\right)^2}{R_1 + R_2 + \left(L_r + R_1 R_2 C\right) j\omega + L_r R_2 C \left(j\omega\right)^2} \tag{5}$$

$$R_1 = \rho c, R_2 = 0.505\rho c, \quad L_r = 0.613a, \qquad C = 1.111a/\rho c^2 \tag{6}$$

where a is the radius of the instrument at the radiating end, $j = \sqrt{-1}$, and ω is the angular frequency. R_1, R_2, L_r and C are the respective component values for an equivalent electrical network that approximates this impedance model with two resistors, an inductor and a capacitor. The network formulation of this impedance is found in Bilbao and Chick (2013). This time domain approximation displays similar loss and internal reflection characteristics to measured responses over the playing range of interest.

FDTD Scheme Construction

FDTD methods allow for flexible simulation of time varying systems. The process involves discretising the time and spatial fields and using approximations to differential operators that lead to recurrence relations.

Grids and Field Sampling

The first step in the construction of a FDTD scheme is the definition of grids in terms of a time step, k, and a grid spacing h. The time step, defined as the inverse of the sample rate is not independent of the grid spacing, and must satisfy $h \geq ck$ for numerical stability. This condition can be arrived at through energy methods (Bilbao 2009) and is the same as the Courant-Friedrichs-Lewy stability condition (Courant et al. 1928). Using this spatial step, the domain over which the instrument is defined over can be divided evenly into $N = \text{floor}(L/h)$ segments.

In analogy with work in electromagnetism, an interleaved pair of grids is used (Yee 1966). The pressure field is sampled at integer multiples of k and h and the velocity field is sampled at "half integer" multiples i.e. points half way between neighbouring integer multiples. This can be notated as

$$p_l^n \cong p(lh, nk), \quad v_{l+\frac{1}{2}}^{n+\frac{1}{2}} \cong v\left((l+1/2)\,h, (n+1/2)\,k\right)$$

The use of two grids requires distinct samplings of the cross sectional area $S(x)$ of the instrument bore. The cross sectional area grid function $S_{l+1/2}$ on the half-integer grid locations may be set directly from the bore profile $S(x)$ as $S_{l+1/2} = S(x = \left(l + \frac{1}{2}\right) h)$. For the maintenance of numerical stability, a useful choice of the dicrete cross sectional area over the integer-valued grid $\bar{S}_l$ is $\bar{S}_l = 0.5\left(S_{l+1/2} + S_{l-1/2}\right)$ (Bilbao and Chick 2013). See Figure 3, showing the arrangement of the discrete pressure and velocity fields as well as the sampling of the instrument bore.

The lip and radiation models are also sampled on the interleaved time grid; the element values required in the radiation model lie on the integer multiple time grid and the excitation values lie on the half integer multiple time grid. Other arrangements are of course possible.

Finite-Difference Approximations

There are several approximations used in the construction of this FDTD scheme. Time and space finite-difference operators are used to replace their continuous counterparts to generate a scheme which is implemented as a recursive update in a loop operating at the audio rate. Taking the simplified case of the lossless Webster's equations

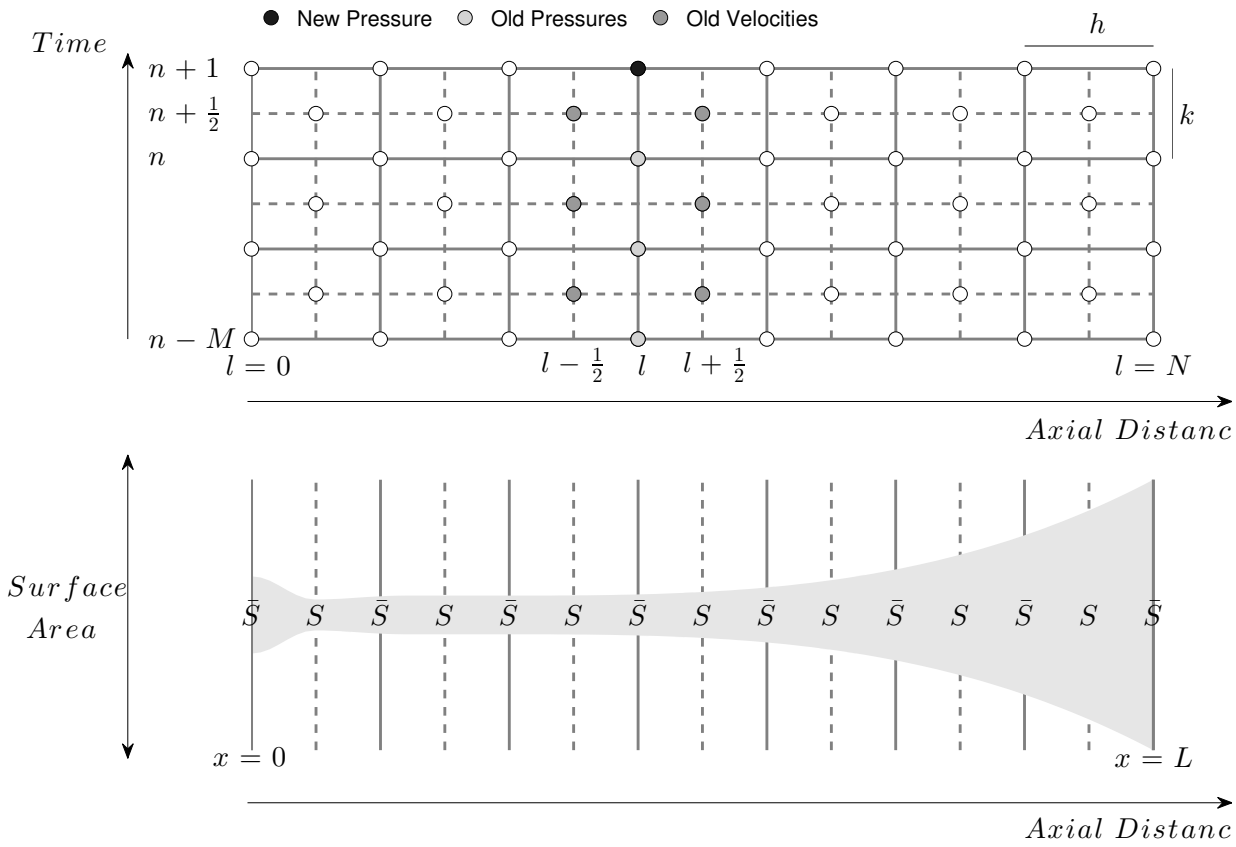

Figure 3. Top: Grid arrangement and representation of the lossy pressure update scheme. Pressure field grid points lie on solid line intersections, velocity field grid points lie on dotted line intersections. The current pressure value (indicated in black) can be calculated using previously computed values of pressure (light grey) and velocity (dark grey). Bottom: Bore profile on interleaved spatial grid. Solid lines lie across integer time and space grids. Dotted lines lie across half integer time and space grids.

given in (1), the simplest "leap-frog" FDTD updates are

$$\begin{aligned} p_l^{n+1} &= p_l^n - \frac{\rho c^2 k}{h\bar{S}_l}\left(S_{l+1/2}v_{l+1/2}^{n+1/2} - S_{l-1/2}v_{l-1/2}^{n+1/2}\right) \\ v_{l+1/2}^{n+1/2} &= v_{l+1/2}^{n-1/2} - \frac{k}{h\rho}\left(p_{l+1}^n - p_l^n\right) \end{aligned}$$

All of the temporal difference operations can be interpreted as digital filters. This idea is extended in the approximation of the fractional derivatives required in modeling viscothermal loss; an M^{th} order infinite impulse response filter is used to approximate

the half derivative required to model viscothermal losses (Haddar et al. 2009). For audio rates, a value of $M = 20$ is satisfactory (Bilbao and Chick 2013). This increases the number of points required in the update equations and is highlighted at top in Figure 3 for the case of the pressure update. Temporal averaging is also necessary to centre the lossy schemes.

Development

The brass instrument environment was developed using two programming languages.

MATLAB Prototype

The brass instrument environment was originally prototyped using MATLAB, a high level computing language primarily used for numerical computations (MathWorks 2014). Although this platform lacks the speed offered by lower level languages, such as C, it gives users access to a library of built-in mathematical functions together with visualisation tools.

Score and instrument files were written as MATLAB scripts that are run within the main brass code, similar to the means of control of the early Music-N synthesis procedures (Mathews 1961). Information from these files is then used to create arrays of precomputed values that are used within the update equations. The pressure and velocity values are each stored as separate two-dimensional arrays that store values at the previous M time steps. The updates for the pressure and velocity equations are then solved using both matrix and element-wise multiplication operators. Radiation elements and lip positions also require extra updates and storage.

C Port

The initial C implementation of the brass model was essentially a straight translation of the MATLAB code into serial C, with some precomputed elements moved into the main loop to reduce memory access. This runs around 4 to 7 times faster than the MATLAB version; larger speed-ups were highly desirable.

As the FDTD scheme is a one-dimensional simulation the grid size is fairly small, typically on the order of a few hundred elements at audio rates, and this limits the potential for most types of parallelism; for example, a multi-threaded version would spend more time synchronising between the threads than would be gained by having multiple threads. However, a finer grained approach using vectorisation is possible. To exploit this, the pressure and velocity update loops for the main and bypass tubes have been accelerated using Intel AVX (Firasta et al. 2008). This is a 256-bit (4-way double precision) vector instruction set available on newer Intel and AMD CPUs. The default tube updates are typically too small to be worth vectorising. The AVX version of the code runs around 1.5 to 2.5 times faster than the serial C and around 7 to 16 times faster than the MATLAB version. This means that for instruments of dimensions to those of a standard trumpet, real time performance is possible.

Table 1 shows total simulation run times for several instrument configurations: a trumpet with one valve, a horn with three valves and a six metre long instrument with three valves made using the custom instrument function (see next section). The total simulation times are given for the three different versions of the code all run on an Intel Core i5-4300U CPU. Test files

Rather than inventing new file formats for the C port, a parser capable of translating

[1]Simulations were run at a sample rate of 44.1kHz for 1s output.

Run times in seconds for three different instrument definitions in each development stage[1]

Instrument	MATLAB	C	AVX
Trumpet (1.3m, 1 valve)	14.02	1.974	0.857
Horn (4.5m, 3 valves)	27.89	6.120	3.63
Custom (6m, 3 valves)	N/A	19.223	7.946

Table 1

MATLAB formatted arrays was implemented, allowing the same instrument and score definitions to be used both for the original MATLAB model and for the C port. In addition to this, an XML file format is supported so that the brass environment can be included in a larger synthesis project (NESS 2014).

Use of Environment

Use of the environment at the command line is as follows:

```
ness-brass -i instrumentfile -s scorefile
```

The necessary flags `-i` and `-s` refer to the instrument and score file names. Tables 2 and 3 show the structure of the instrument and score files using MATLAB vector notation and give descriptions of each parameter. In MATLAB vector notation a comma ', ' is used to separate individual column entries in each row and a semicolon '; ' is used to separate rows. Figure 4 shows a visual representation of a custom instrument as well as a breakdown of the different custom instrument settings given in Table 2.

[2]Alternatively the bore can be manually defined by setting `custominstrument=0` and replacing the custom instrument section parameters with a two column array, `bore=[position,diameter]`, that defines the bore as a breakpoint function of the axial coordinate[mm]-diameter[mm] pairs which are then linearly interpolated. Note that the custom instrument defined in this table is different from the one used

Structure and description of custom instrument file with three valves[2]

Instrument File Input	Description
`custominstrument=1;`	1 uses custom instrument function, 0 requires manually entering bore profile.
`FS=44100;`	Sample rate[Hz].
`temperature=20;`	Temperature[°C].
`vpos=[600,630,660];`	Position of valves[mm]. Must have the same number of entries as the default and bypass tube lengths.
`vdl=[20,20,20];`	Default tube lengths[mm].
`vbl=[130.7,57.5,199.8];`	Bypass tube lengths[mm].
`xmeg=50;`	Mouthpiece length[mm].
`rmeg=10;`	Mouthpiece opening diameter[mm]. Opens at this diameter and uses a raised cosine to ramp to the first diameter given in the middle section.
`x0eg=[100,200,200,100];`	Middle section lengths[mm].

`r0eg=[5,7,3;`
`7,7,1;`
`7,15,2;`
`7,5,1];`
	Definition of middle section profiles. Must have 3 columns and have the same number of rows as entries for the middle section lengths. First and second entries of each row are the diameters[mm] used in the bore definition. The third entry specifies the type of profile: 1 - linear ramp between the two diameters, 2 - squared sine bulge whose entrance and exit diameters are the first column entry and maximum is the second diameter entry, 3 - cosine ramp between the two diameters. See Figure 4 at right.

`Leg=1000;`	Total length of instrument[mm]. Length of flare section is taken as the difference between the total length and the sum of the mouthpiece lengths and the middle section lengths.
`rbeg=50;`	End diameter of instrument[mm].
`fbeg=4;`	Exponent of flare curve. Flare is constructed using a single power that goes from the final diameter of the middle sections to end diameter of the instrument.

Table 2

The following sections present examples when the custom bore profile is used to create a trumpet, illustrating some features unique to this particular environment. Black lines in spectrograms denote peaks in the frequency spectrum. Most of the parameters

in Table 1.

[3]Time varying components are breakpoint functions of time[s]-value pairs that are linearly interpolated and are explained further in the text. Note that for the valve parameters, each row must contain a time entry and an entry corresponding to each valve.

Layout and description of score file for a three valved instrument. [3]

Score File Input	Description
`maxout=0.95;`	Normalised range of output
`T=1;`	Length of gesture[s]
`Sr=[0,1.46e-5];`	Lip area[m^2]
`mu=[0,5.37e-5];`	Lip mass[kg]
`sigma=[0,5];`	Lip damping
`H=[0,0.00029];`	Lip separation[m]
`w=[0,0.01];`	Lip width[m]
`lip_frequency=[0,550];`	Lip frequency[Hz]
`pressure=[0,0;1e-3,5e3];`	Mouth pressure[Pa]
`vibamp=[0,0];`	Vibrato amplitude. This specifies a fraction of the lip frequency.
`vibfreq=[0,0];`	Vibrato frequency[Hz]
`tremamp=[0,0];`	Tremolo amplitude. This specifies a fraction of the mouth pressure.
`tremfreq=[0,0];`	Tremolo frequency[Hz]
`noiseamp=[0,0];`	Noise amplitude. This specifies a fraction of the mouth pressure.
`valveopening=[0,1,1,1];`	Valve openings. The fraction that the default tube is open. Must have columns for each valve in instrument. Likewise for valve modulation inputs.
`valvevibfreq=[0,0,0,0];`	Valve modulation frequency[Hz]
`valvevibamp=[0,0,0,0];`	Valve modulation amplitude. This is added to the valve opening.

Table 3

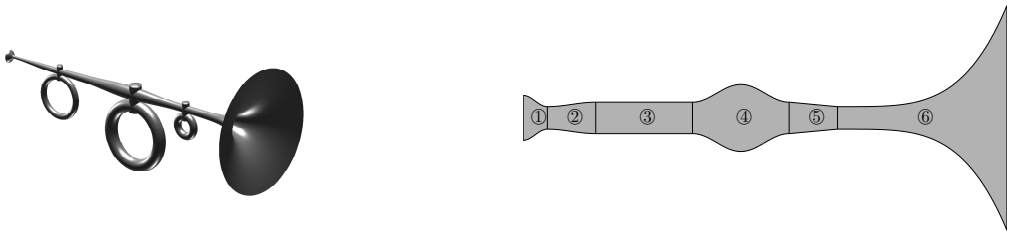

Figure 4. Bore constructed using custom instrument function. Left: Visualisation of a three valved custom instrument. Right: Functional structure of custom instrument given in Table 2 with sections labelled. Section 1 corresponds to the mouthpiece, and is constructed using a raised cosine. Section 2 also uses a raised cosine. Section 3 is a straight piece of tubing with constant diameter. Section 4 uses a squared sine function to add a bulge to the instrument profile. Section 5 linearly ramps between two diameters. Section 6 is the flare section which expands to a set diameter for a given exponential power.

for each example are the same as in Table 3, unless otherwise stated, and the lip frequency has been adjusted in some examples to provide a better example. The corresponding score, instrument and sound files along with other documentation and examples can be found at http://www2.ph.ed.ac.uk/~s0916351/CMJBrass2014.html.

Finding Playable Notes

To find usable lip frequencies to "play" an instrument, particularly when the instrument is constructed from scratch, a linear sweep can give an indication of the playable regions. Using the MATLAB vector formatting this can be done by setting the lip frequency as

$$\texttt{lip\_frequency} = [0, 220; 3, 1000];$$

in the score file. This creates a lip frequency sweep from 220Hz to 1000Hz over a period of 3s. Figure 5 shows the spectrogram of the output sound using this frequency sweep as part of the score file. This can give a lower bound to which lip frequencies can be used to produce a sound as well as give some indication into the instruments behaviour in certain frequency ranges. For example, since no sound is produced before 0.25s this shows that the lower frequencies in this example are unlikely to be able to produce a sustained sound. However, due to the complex nature of the instrument-lip system these kind of plots can only give a general idea of how the instrument works and must be used alongside trial and error to determine how to get the best results from the instrument.

Lip Vibrato and Tremolo

Modulation of lip frequency and mouth pressure can be performed using the vibrato and tremolo sections in the score file. These modulations are added to the original signals as fractions of the original signal value. In the case of the vibrato

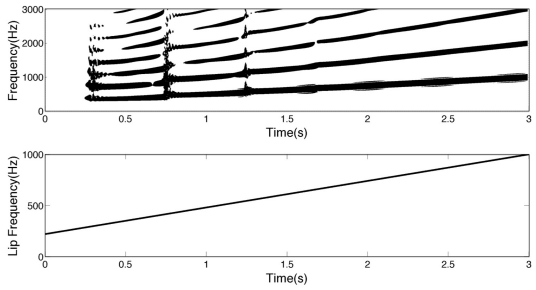

Figure 5. Top: Spectrogram of output sound using a lip frequency sweep. Bottom: Lip frequency as a function of time.

function, this modifies the lip frequency as follows

$$f_{lip} :\rightarrow f_{lip}\left(1 + A_v \sin\left(2\pi f_v t\right)\right)$$

where f_v is the modulation frequency and A_v is the amplitude of modulation which defines a change of the original signal amplitude as a ratio. A similar transformation is used to apply the tremolo function to the mouth pressure. Figure 6 show examples of time varying vibrato and tremolo functions.

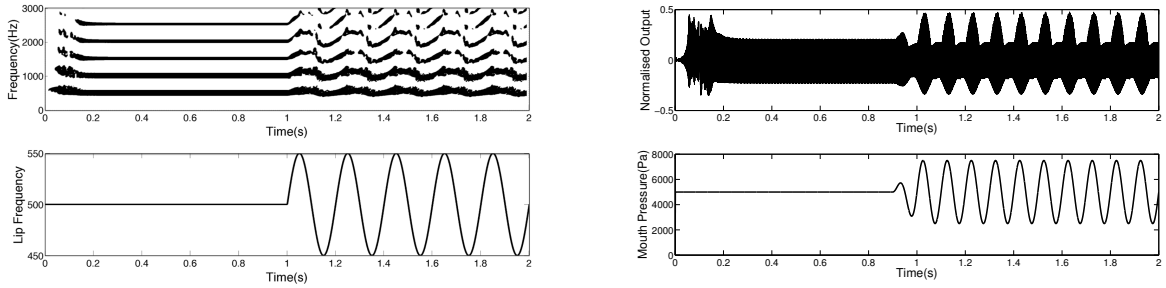

Figure 6. Left Top: Spectrogram of output when a lip vibrato of amplitude 0.1 *and rate* $5\mathrm{Hz}$ *is added to the end of a note using a lip frequency of* $500\mathrm{Hz}$*. Left Bottom: Lip frequency as a function of time for this vibrato. Right Top: Output when a tremolo of amplitude* 0.5 *and rate* $10\mathrm{Hz}$ *is added to the end of a note where the static mouth pressure is* $5\mathrm{kPa}$*. Right Bottom: Mouth pressure as a function of time for this tremolo. There is a* $1\mathrm{ms}$ *ramp from* 0 *to the static pressure at the beginning of the mouth pressure function.*

Valve Transitions and Multiphonics

A valve transition can be performed by varying the valve opening over time. For a single valved instrument this is done as follows

$$\texttt{valveopening} = [0, 1; 1, 1; 2, 0; 3, 0];$$

This input holds the valve open for one second (the only path available is through the default tubing), gradually closes over the next second, then is held closed for the final second (the only path available is through the bypass section of tubing). Figure 7 shows the spectrogram for this valve gesture. In the transition region, the lip model fails to lock onto a normal resonance of the instrument and multiphonic sounds can be heard. This effect can be explored more in depth by setting the valve opening to various other values between 0 and 1.

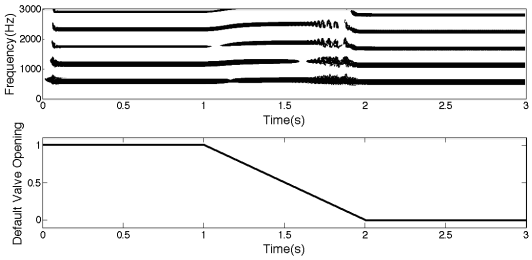

Figure 7. Top: Spectrogram of output when a valve moves from one configuration to another. Bottom: Valve opening as a function of time.

Valve Vibrato

Valve-opening signals can also be modulated in a manner similar to that of the modulation of lip frequency and mouth pressure. In this case the modulation function is added to the original signal rather than multiplying it. The modified valve opening signals must also be clipped to lie between 0 and 1 so as to be physically plausible (and

also to maintain stability, a more important concern). Figure 8 shows an example of the use of such valve vibrato.

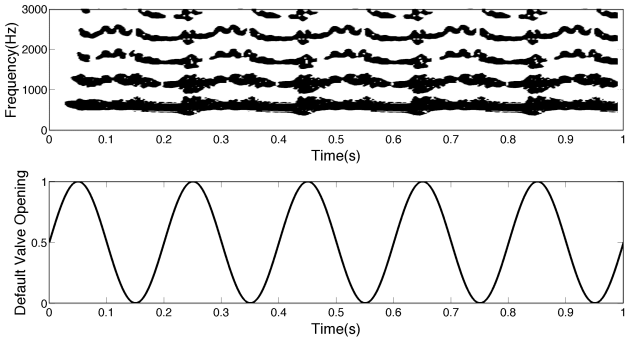

Figure 8. Top: Spectrogram of output for a valve configuration that is modulated in time. Bottom: Valve modulation function and final valve signal. Static valve opening is set to 0.5, valve vibrato amplitude is 0.5 and frequency is 5Hz.

Sound Loom Implementation

The Soundloom interface to the Composers Desktop Project (Forshee 2006) software allows the user to run C-programs for transforming sounds from a graphic interface written in TK/Tcl, hiding the mechanics of the process. The interface allows users to create data-files, graphically or by writing text. Restraints are placed on interface value ranges to prevent invalid data being entered and *Help* or *Information* buttons are available to guide the user. In addition the Soundloom has an associated file parser, allowing it to recognise and differentiate between various file types.

In order to run the brass physical modeling software from the Soundloom it was decided to retain the original MATLAB format of the ascii textfiles, and therefore the existing textile parsing needed to be extended for Soundloom to recognise the instrument and score files. The internal parsing recognises the instrument and score file formats at session start-up and between sessions retains an intelligent library of such files; Soundloom knows which instrument is used by which score file. This library

updates as files are created, destroyed or modified. These files can then be listed, viewed and edited from the score and instrument interface in the Soundloom, and sound output created. See Figure 9 for the bore profile creation interface and score interface.

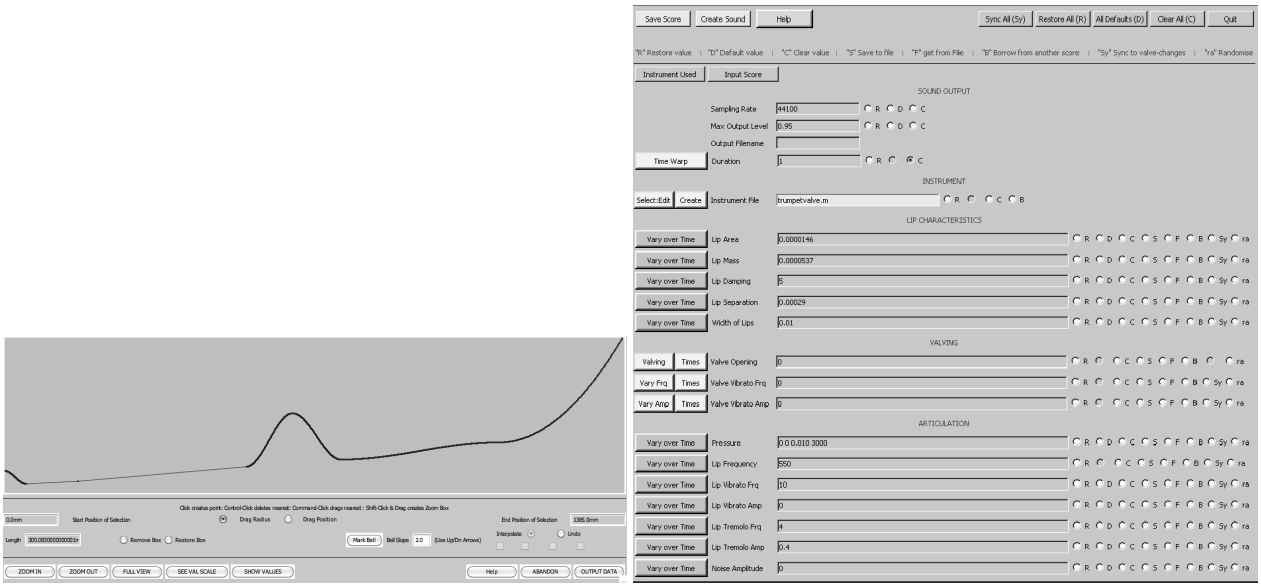

Figure 9. Screen grabs from the interface of Soundloom using the brass environment. Left: Instrument interface. Right: Score interface.

The bore profile and valve positions of the brass instrument can be entered graphically. The user clicks on the screen to create new points which are automatically joined up to create an instrument profile. Points can be dragged to new positions, or deleted. The interface itself makes certain non-permitted shapes impossible to draw, such as profiles which move left to right then right to left, and it also verifies that the user has specified a correct number of points. It is also possible to interpolate values, creating a smoother shape for the bell, the mouthpiece, the internal bore, or any combination of these. For interpolation, the mouthpiece is assumed to be cosinusoidal whilst the bell can be interpolated using a user specified power function which can produce concave, convex or conical profiles. Internal points are interpolated using a combination of sinusoids and cubic-splines to best retain the shape suggested by the marked points.

The profile is initially drawn in a window of fixed proportions which is then scaled

by the user. Typical default values for a Trumpet, Trombone and Horn are also available via a button press. The final profile can then be viewed to scale. This allows for all possible instrument profiles, from the extremely long and thin to the extremely short and fat, to be drawn on-screen.

As the score file format will most likely be unfamiliar to the user, the interface displays and accepts new data as numbers or number-lists and then converts them into appropriate MATLAB formatting. If required the files themselves can be viewed via buttons on the score interface. As a start-up help for the new user, a *Default* button enters default values for most parameters, and a *Help* button explains their nature and range. As with other Soundloom programs, time-varying parameters can be entered graphically, but now onto a blank display screen which resulted in the parameter values' being sent to the appropriate entry box as lists of paired numbers.

Valve depression can also be entered graphically. Because of the nature of the data-processing procedure valve changes are treated group-valve openingwise; at each time where any valve reaches a new value, openings for all the valves at that time must be specified. Thus in the interface the openings of the valve at each valve change time are entered sequentially on the display. After information has been entered for all the valves the timing of these valve changes is entered. At present this is done by tapping on the mouse but MIDI entry and typed text would also be possible.

The user may also synchronise time-changing values to valve opening changes or randomise values within a specified range. Time-warping of the data is also possible to stretch time intervals, e.g. a warp value of two will multiply all time-values by two. As the full-pressure onset time is often critical to making the instrument speak, this can remain fixed while all other time-values are being warped.

By the very nature of working in this score/instrument format the user will create a large number of score files but may wish to retain/reuse data from previous scores. Buttons on the interface therefore allow the user to grab values for a specific parameter from an existing score.

Future Work and Conclusions

As the model for this environment is broken into generator, resonator and radiation parts it is relatively straightforward to make modifications to the environment. The simplest modification would be to choose from different generator models to simulate different instruments whose resonator is an acoustic tube, such as a woodwind instrument or the voice (Fletcher and Rossing 1998).

A modification to the radiation model would be to embed the instrument within an acoustic space by coupling the resonator equations to the 3D wave equation:

$$\partial_t \Psi(\mathbf{x}, t) = \nabla^2_{3D} \Psi(\mathbf{x}, t)$$

where Ψ is the acoustic velocity potential, $\mathbf{x}$ is a three-dimensional position vector and ∇^2_{3D} is the three-dimensional Laplacian operator. This would add spatialisation to the sound as well as improve radiation behaviour. Energy methods can be employed to couple the instrument model to the room model and boundary conditions can be set within the room to model the flare walls. modeling an acoustic space significantly increases the operation count, and therefore run time of simulations. Although the brass instrument model presented in this paper is not suitable for parallelisation, implementation of room simulations using Graphics Processing Units has been shown to drastically improve performance (Webb 2014).

To synthesise the characteristic 'brassy 'sounds of brass instruments played at high dynamic levels the resonator model must be changed so that it is described by a set of nonlinear equations, e.g. the Euler equations (Rudenko and Soluyan 1977). However, nonlinear equations do have problems with stability and energy methods used for the current environment cannot be applied to this problem.

A further extension to the resonator model would be to include the action of a brass instrument slide, such as those present in trombones. Use of a slide alters the overall length of the air column, and therefore the instrument's resonances, in a continuous fashion. This is different from the act of a brass instrument valve which alters the instrument resonances in a more discrete fashion by diverting air into other pieces of tubing of differing length.

The brass environment binaries, user documentation, sound examples and example score and instrument files can be found at http://www2.ph.ed.ac.uk/~s0916351/CMJBrass2014.html. Files for Soundloom can be found at www.trevorwishart.co.uk/slfull.html. Users have full control over how the instrument is played and how the instrument is made which gives a great number of possibilities for creating new musical sounds that are difficult or even impossible to create in the real world. Control of the virtual instruments is intuitive as input parameters are related to physical properties of the instrument system that musicians are already familiar with. As with any real instrument, practice is required to become familiar with what it is capable of doing but the end result is rewarding.

Although finite-difference methods are less efficient than other physical modeling methods, e.g. waveguides, this is becoming less of a problem as modern computing power continually increases. This particular algorithm has a relatively low computation cost; sound synthesis for trumpet like instruments is real-time on modern laptops,

therefore exploration of possible sounds can be done in a reasonably fluid manner.

Acknowledgements

This work was supported by the European Research Council, under grant number StG-2011-279068-NESS.

References

Adachi, S., and M. Sato. 1995. "Time-domain simulation of sound production in the brass instrument." *Journal of the Acoustical Society of America* 97(6):3850–3861.

Berners, D. P. 1999. "Acoustics and Signal Processing Techniques for Physical Modelling of Brass Instruments." Ph.D. thesis, Stanford University.

Bilbao, S. 2009. *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*. Wiley.

Bilbao, S. 2011. "Modelling of Brass Instrument Valves." In *Proceedings of the* 14^{th} *International Conference on Digital Audio Effects*. Paris, France, pp. 337–343.

Bilbao, S., and J. Chick. 2013. "Finite difference time domain simulation for the brass instrument bore." *Journal of the Acoustical Society of America* 134(5):3860–3871.

Cook, P. 1991. "Tbone: An interactive waveguide brass instrument synthesis workbench for the NeXT machine." In *Proceedings of the International Computer Music Conference*. Montreal, Canada, pp. 297–299.

Courant, R., K. Friedrichs, and H. Lewy. 1928. "On the partial differential equations of mathematical physics." *Mathematische Annalen* 100:32–74.

Eveno, P., J. P. Dalmont, R. Caussé, and J. Gilbert. 2012. "Wave Propagation and Radiation in a Horn: Comparisons Between Models and Measurements." *Acta Acustica united with Acustica* 98:158–165.

Firasta, N., M. Buxton, P. Jinbo, K. Nasri, and S. Kuo. 2008. "Intel® AVX : New Frontiers in Performance Improvements and Energy Efficiency." *Intel White Paper* Available online https://software.intel.com/en-us/articles/intel-avx-new-frontiers-in-performance-improvements-and-energy-efficiency.

Fletcher, N. H., and T. D. Rossing. 1998. *The Physics of Musical Instruments, Second Edition*. Springer.

Forshee, J. 2006. "Composers' Desktop Project Version 5.0.1." *Computer Music Journal* 30(3):90–92.

Haddar, H., J. R. Li, and D. Matignon. 2009. "Efficient solution of a wave equation with fractional-order dissipative terms." *Journal of Computational and Applied Mathematics* 234(6):2003–2010.

Hélie, T. 2003. "Unidimensional models of acoustic propagation in axisymmetric waveguides." *Journal of the Acoustical Society of America* 114(5):2633–2647.

Hiller, L., and P. Ruiz. 1971. "Synthesizing Musical Sounds by Solving the Wave Equation for Vibrating Objects: Part I." *Journal of the Audio Engineering Society* 19(6):462–470.

Keefe, D. H. 1984. "Acoustical wave propagation in cylindrical ducts: Transmission line parameter approximations for isothermal and nonisothermal boundary conditions." *Journal of the Acoustical Society of America* 75(1):58–62.

Levine, H., and J. Schwinger. 1948. "On the Radiation of Sound from an Unflanged Circular Pipe." *Physical Review* 73(4):383–406.

Mathews, M. V. 1961. "An Acoustic Compiler for Music and Psychological Stimuli." *The Bell System Technical Journal* May:677–694.

MathWorks. 2014. "MATLAB - The Language of Technical Computing." URL http://www.mathworks.co.uk/products/matlab/. Last accessed October 2014.

McIntyre, M. E., R. T. Schumacher, and J. Woodhouse. 1983. "On the oscillations of musical instruments." *Journal of the Acoustical Society of America* 74(5):1325–1345.

Mignot, R., T. Hélie, and D. Matignon. 2010. "Digital Waveguide Modeling for Wind Instruments: Building a State Space Representation Based on the Webster Lokshin Model." *IEEE Transactions on Audio, Speech, and Language Processing* 18(4):843–854.

Morrill, D. 1977. "Trumpet Algorithms for Computer Composition." *Computer Music Journal* 1(1):46–52.

NESS. 2014. "Next Generation Sound Synthesis." URL http://www.ness-music.eu/. Last accessed December 2014.

Risset, J.-C. 1966. "Computer Study of Trumpet Tones." Technical report, Bell Technical Laboratories, Murray Hill, New Jersey.

Rudenko, O. V., and S. I. Soluyan. 1977. *Theoretical Foundations of Nonlinear Acoustics*. Consultants Bureau, New York.

Ruiz, P. 1969. "A Technique for Simulating the Vibrations of Strings with a Digital Computer." Master's thesis, University of Illinois.

Russ, M. 1994. "Yamaha VL1 Virtual Acoustic Synthesizer." *Sound on Sound* Available online http://www.soundonsound.com/sos/1994_articles/jul94/yamahavl1.html.

Silva, F., P. Guillemain, J. Kergomard, B. Mallaroni, and A. N. Norris. 2009. "Approximation formulae for the acoustic radiation impedance of a cylindrical pipe." *Journal of Sound and Vibration* 322:255–263.

Silva, F., C. Vergez, P. Guillemain, J. Kergomard, and V. Debut. 2014. "MoReeSC: A Framework for the Simulation and Analysis of Sound Production in Reed and Brass Instruments." *Acta Acustica united with Acustica* 100:126–138.

Smith, J. O. 1986. "Efficient simulation of the reed-bore and bow-string mechanisms." In *Proceedings of the 1986 International Computer Music Conference*. The Hague, pp. 275–280.

Smith, J. O. 2004. *Physical Audio Signal Procesing*. Stanford, CA. Draft version. Available online at `http://ccrma.stanford.edu/~jos/pasp04/`.

van Walstijn, M., and M. Campbell. 2003. "Discrete-time modelling of woodwind instrument bores using wave variables." *Journal of the Acoustical Society of America* 113(1):575–585.

Vergez, C., and P. Tisserand. 2006. "The BRASS Project, from Physical Models to Virtual Musical Instruments: Playability Issues." In R. Kronland-Martinet, T. Voinier, and S. Ystad, (editors) *Lecture Notes in Computer Science: Computer Music Modeling and Retrieval*, volume 3902. Berlin/Heidelberg, Germany: Springer, pp. 24–33.

Webb, C. J. 2014. "Parallel computation techniques for virtual acoustics and physical modelling synthesis." Ph.D. thesis, The University of Edinburgh.

Webster, A. G. 1919. "Acoustical Impedance, and the Theory of Horns and of the Phonograph." *Proceedings of the National Academy of Sciences of the United States of America* 5(7):275–282.

Yee, K. S. 1966. "Numerical Solution of Initial Boundary Value Problems Involving Maxwell's Equations in Isotropic Media." *IEEE Transactions on Antennas and Propagation* 14(3):302–307.