AALTO UNIVERSITY
SCHOOL OF SCIENCE AND TECHNOLOGY
Faculty of Electronics, Communications and Automation
Department of Signal Processing and Acoustics

Tapio Puputti

Real-time Implementation of the Virtual Air Slide Guitar

Master's Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology.

Helsinki 25.5.2010

Supervisor: Professor Vesa Välimäki
Instructor: Dr. Jyri Pakarinen, D.Sc.(Tech)

AALTO-YLIOPISTO DIPLOMITYÖN
TEKNILLINEN KORKEAKOULU TIIVISTELMÄ

Tekijä: Tapio Puputti
Työn nimi: Real-time Implementation of the Virtual Air Slide Guitar
Päivämäärä: 25.5.2010 **Kieli:** Englanti **Sivumäärä:** 6 + 56

Elektroniikan, tietoliikenteen ja automaation tiedekunta
Signaalinkäsittelyn ja akustiikan laitos
Professuuri: Audiosignaalinkäsittely **Koodi:** S-89

Työn valvoja: Professori Vesa Välimäki
Työn ohjaaja: TkT Jyri Pakarinen

Tiivistelmäteksti: Tässä diplomityössä esitellään reaaliaikainen slide-kitaran implementaatio ilmakitarakäyttöliittymällä. Aiempaa tutkimusta ja työn taustaa käydään lyhyesti läpi. Slide-kitaran analyysi- ja synteesimenetelmät kuvaillaan sekä käydään läpi myös vaihtoehtoisia menetelmiä.

Tämä reaaliaikainen implementaatio koostuu kolmesta pääosasta. Infrapunakamera seuraa käyttäjän liikkeitä ja eleitä, kameraohjelma tunnistaa käyttäjän eleet ja hoitaa laskutoimitukset kameran datalle. Pure Data –ohjelma syntetisoi slide-kitaran äänen kameraohjelmalta saadun tiedon perusteella.

Virtuaalista slide-kitaraa voi soittaa minimaalisella opettelulla. Käyttäjän tarvitsee vain heiluttaa käsiään kameran edessä kuullakseen kitaran äänen. Kuitenkin, jotta soittaja saisi aikaan oikeaa slide-kitaraa muistuttavaa soittoa, tulee käyttäjällä olla kokemusta kitaransoitosta, sillä käsin kosketeltavaa soitinta ei ole. Soitinjärjestelmällä on pieni viive ja se sisältää asetukset mm. slide-putken, kielten virityksen ja kielten valitsemiseen.

Vaikka ohjelma on kokonaisuudessaan melko raskas varsinkin vanhemmille tietokoneille, sen tuottama ääni ja varsinkin slide-putken ja kielten välinen kontaktiääni on onnistunut ja soitin kuulostaa sekä käyttäytyy kuten oikea slide-kitara.

Järjestelmää esittelevä video löytyy nimikkeellä “Virtual Slide Guitar” YouTube:sta (http://www.youtube.com/watch?v=eCPFYKq5zTk).

Avainsanat: Reaaliaikainen implementaatio, slide-kitara, virtuaalinen soitin, äänisynteesi, eleohjaus, infrapunakamera, slide-putki, konenäkö, Pure Data, kitaraefekti, kameraohjaus.

AALTO UNIVERSITY
SCHOOL OF SCIENCE AND TECHNOLOGY

ABSTRACT OF THE
MASTER'S THESIS

Author: Tapio Puputti
Name of the Thesis: Real-time implementation of the virtual slide air guitar
Date: 25.5.2010 **Language:** English **Number of pages:** 6 + 56

Faculty Electronics, Communications and Automation
Department of Signal Processing and Acoustics
Professorship: Audio Signal Processing **Code:** S-89

Supervisor: Professor Vesa Välimäki
Instructor: Dr. Jyri Pakarinen, D.Sc.(Tech)

Abstract of the Master's Thesis: An implementation of a virtual slide guitar is presented. Previous work is viewed with brief background examination. The analysis and synthesis methods that were used in this thesis as well as other possibilities for analysis and synthesis of a slide guitar are described.

This real-time implementation consists of three main parts. An infra-red camera to track player movement and gestures, a camera API for gesture recognition and camera data handling and a Pure Data program to synthesize and output the slide guitar sound according to the guitarist's playing.

The Virtual Slide Guitar can be played with a minimal learning curve. All the player has to do is wave his/her hands in front of the camera. To play anything sounding like a real slide guitar, a player needs to be familiar with a guitar since no haptic feedback is available due to the air interface of the instrument. The VSG has a low latency and custom options for playing with different slide tubes, tunings, and strings.

Although computationally heavy for older computers, the sound and especially the contact sound of the slide tube and guitar strings works and the instrument sounds and acts like a real slide guitar.

A video demonstrating the Virtual Slide Guitar can be found on YouTube (http://www.youtube.com/watch?v=eCPFYKq5zTk).

Keywords: Real-time implementation, slide guitar, virtual instrument, sound synthesis, gesture control, infra-red camera, slide tube, vision-based control, Pure Data, guitar effect, camera tracking.

Acknowledgements

This Master's Thesis has been carried out for the Laboratory of Acoustics and Audio Signal Processing of the School of Science and Technology of Aalto University (formerly Teknillinen Korkeakoulu, TKK). The project was funded by TKK and was conducted during years 2006-2010. The work was instructed by Dr. Jyri Pakarinen and supervised by Dr. Vesa Välimäki of the Laboratory of Acoustics and Audio Signal Processing.

In the late summer of 2006, I was approached by my thesis supervisor Dr. Välimäki about a thesis topic. I was immediately excited when I heard that it would be a virtual instrument. Sound synthesis has always interested me and this was also a chance to do something like a real instrument, although a virtual one. Work was begun in the following fall with help from my instructor, Dr. Pakarinen.

Many thanks go to Dr. Pakarinen and Dr. Välimäki for all the valuable support they gave me with my thesis. I especially appreciate Dr. Pakarinen's advice, tips and ideas, not forgetting his experience and knowledge with guitar playing and his enthusiastic attitude. Dr. Pakarinen's input on synthesis, measurements and analysis was most valuable.

I would also like to thank Teemu Mäkipatola and Aki Kanerva from Virtual Air Guitar Company who gave valuable help and pointers during the beginning stages of the thesis work as well as the Telecommunications Software and Multimedia Laboratory for lending their infra-red camera. Most of all and by far the most help in making the synthesis part possible was the PD-list, a mailing list with users' own example codes and discussion about programming with Pure Data.

Helsinki, 25.5.2010

Tapio Puputti

Table of contents

Symbols

a, a_c	loop filter coefficients
d	distance between blobs
dr	decay rate
f_0	fundamental frequency of string
f_c	slide velocity
f_s	sampling frequency
g, g_c	loop filter coefficients
g_{bal}	friction balance parameter
g_{TV}	scaling coefficient for contact sound output
L	absolute length of string
ΔL	relative length of string
n_w	number of windings in string
$p(n)$	signal output from the time-varying delay block
$p_c(n)$	energy-compensated signal
pl	pulse length
Δx	delay line variation in samples per one time step

Abbreviations

API	Application Programming Interface
ASIO	Audio Stream Input Output
CPU	Central Processing Unit
CAVE	Cave Automatic Virtual Environment
DSP	Digital Signal Processing/Processor
DWG	Digital WaveGuide
EVE	Experimental Virtual Environment
FFT	Fast Fourier Transformation
FIR	Finite Impulse Response
FPS	Frames Per Second
HMI	Human Machine Interface
IFFT	Inverse Fast Fourier Transformation
IR	Infra Red
LED	Light Emitting Diode
LP	Linear Prediction
LPC	Linear Predictive Coding
MIDI	Musical Instruments Digital Interface
MMIO	Memory Mapped Input/Output
OSC	Open Sound Control
PD	Pure Data
SDK	Software Development Kit
SDL	Single-Delay Loop
TKK	Teknillinen Korkeakoulu (Helsinki University of Technology), present Aalto University, School of Science and Technology
STFT	Short Time Fourier Transform
UI	User Interface
VAG	Virtual Air Guitar
VST	Virtual Studio Technology
VSG	Virtual Slide Guitar

1 Introduction

This thesis presents a virtual slide guitar that can be played in real-time. A slide guitar synthesis is implemented in Pure Data and it is controlled by means of optical movement tracking and gesture recognition. The system is called a Virtual Slide Guitar (VSG) and it can be viewed as a successor of the Virtual Air Guitar (VAG) (Karjalainen et al. 2006), which was also developed at Helsinki University of Technology (present Aalto University, School of Science and Technology). A slide guitar or bottleneck guitar is a guitar instrument that is played by wearing a slide tube on the fretting hand and by moving the slide tube the pitch (i.e. the length of the string or strings) can be varied continuously while the other hand plucks the strings (Figure 1.1). This produces a unique voice-like tone. The slide tube can be slid across all the six strings but also single notes can be played by plucking just one string and damping the other strings with the plucking hand.

The user controls the VSG by the same gestures he or she would use to play a real slide guitar (Johnson 1990), only that there is no physical guitar. An infra-red camera tracks the user's hands and sends data to the synthesizer software which produces the sound according to the player's hands' movement. Both virtual guitars, the VAG and the VSG, use a computer-vision based approach for recognizing gestures, but the camera type and software are different. The VSG uses an infra-red camera for gesture tracking which has a higher frame rate and resolution compared to the web-cam approach of the VAG. Reviews on gestural control of music synthesis have been written by Paradiso (2007) and by Wanderley and Depalle (2004).

1.1 Background

The VAG acquired widespread popularity and even received international media attention. It is now located as a permanent exhibition item in Heureka, a science center in Vantaa, Finland. The ancestor of the VSG was a so-called rubber-band virtual air guitar, which was an early prototype of the VAG. This was a simplified gesture control system that measured the distance between the hands and translated it directly to string length. However, the rubber-band virtual air guitar could not be played as an instrument due to the resulting shaky pitch and a noticeable latency between plucking and actually hearing the resulting sound.

Several types of air guitar interfaces have been created. For example a guitar controlled by data gloves has been created as a virtual reality application for EVE, a CAVE-like virtual environment with magnetic position tracking and data gloves. Also special control sticks that utilize ultrasonic positioning have been tried for playing the guitar.

The VSG has certain key differences compared to the VAG. Firstly, it's a slide guitar and not a normal, fretted guitar. Therefore the pitch (i.e. length of a string/strings) is not continuous. Also, the VSG operates an infra-red (IR) camera for tracking and gesture recognition compared to the normal web camera approach of the VAG. The infra-red camera has a higher resolution and frame rate compared

Figure 1.1: A slide guitarist wears a slide tube on the fretting hand. The strings are numbered from 1 to 6 starting from the highest and thinnest string. Adopted from Pakarinen et al. (2008).

to the web camera of the VAG. For hand tracking, instead of one-colored gloves as with the VAG, the VSG uses a slide tube and a plucking ring made of IR-reflective fabric. This eliminates room lighting calibration which was a problem with a color-based recognition of the player's hands.

1.2 Outline of the thesis

This thesis has the following structure. The basic ideas and methods of the analysis of a slide guitar as well as slide tube measurements are presented in Chapter 2. Chapter 3 describes the synthesis of the slide guitar sound and slide tube contact sound, and modeling of the various aspects involved in making the VSG sound like a real slide guitar. Chapter 4 covers the implementation part of producing the VSG with sub-chapters describing the different main components of the implementation. Chapter 5 discusses the final implementation as a whole and Chapter 6 concludes this thesis with ideas and observations for improvement and future work as well as retrospect on the success of the final implementation.

The goals of this thesis were to synthesize a slide guitar with a contact sound generator and to implement an air guitar interface for a virtual slide guitar that can be played in real-time and in a similar fashion that a real slide guitar would be played. The main focus of the synthesis part was to synthesize the friction noise between the strings and different slide tubes and to produce a credible slide guitar sound with an unnoticeable latency. To get a low latency between a gesture and a

sounding result of the gesture, it was decided that an infra-red camera was a best choice due to its better frame rate compared to a traditional web camera. The resolution of the IR camera was also found to be quite satisfactory for accurate playing. For the string synthesis, a time-varying digital waveguide (Smith 1992; Välimäki et al. 2006) was chosen.

2 Analysis

The key points in analyzing a slide guitar are the mechanisms involved in generating its unique sound. When the slide tube contacts a guitar string and starts to slide along it, a squeaking sound is produced. The pitch of the resulting sound can be varied by moving the slide tube to another position on the string. The length of the string from the slide tube to the nut or bridge of the guitar determines the pitch of the resulting sound, but also the material and type of the string along with the material of the slide tube itself affect the final sound. The distance from the slide tube to the bridge (Figure 2.1) defines the pitch we hear since the pickups of the guitar are positioned on the body of the guitar. A slide tube can be made of different materials, for example brass, glass, plastic, ivory and chrome. The slide tube can be thought of as a fret that is moving and therefore it is possible to produce continuous pitches by varying its position.

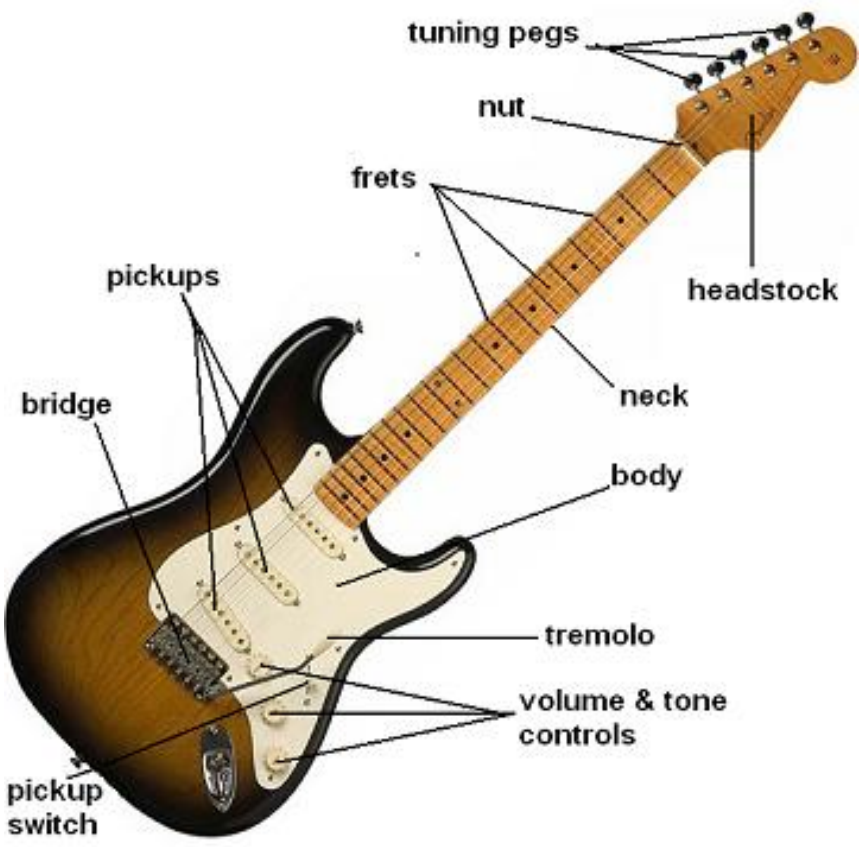

Figure 2.1: Main parts of an electric guitar. Taken from Hellmer.

When moving the slide tube along a string, the pitch changes constantly according to the place the slide tube is positioned. Sliding the tube without picking a string also produces a sound. For this, a closer look at the mechanics and physics involved in creating the friction noise was necessary.

There are two kinds of strings in a guitar, wound strings and unwound strings. A wound string is made of two strings, the outer one wrapped around the center string. This produces bumps on the surface of the string. The distance between the bumps i.e. the thickness of a bump on a string depends on the thickness of the wound string. Unwound strings are just one string with a smooth surface thus making their analysis and synthesis simpler. Figure 2.2 depicts a slide tube on a wound string.

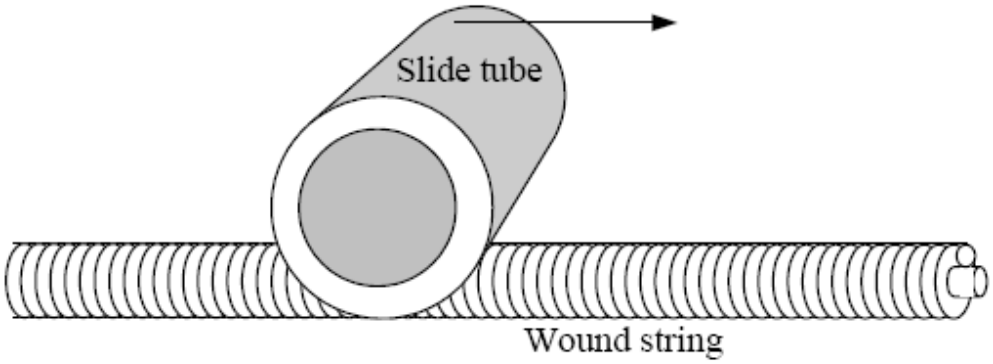

Figure 2.2: A slide tube on a wound string. Notice the bumpy construction of a wound string. Adapted from Pakarinen (2005).

2.1 Contact sounds

When a slide tube touches a string they produce a sound. Sliding the tube along the string produces noise. This noise sounds different depending on the string construction (i.e. wound or unwound) and also the slide tube material. The sound resulting from a slide tube and a wound string could be describes as squeaky while the sound from a slide tube and an unwound string resembles hissing. Sliding movement also affects the resulting sound. The sound of a fast slide is louder than the sound of a slow slide. Also a fast slide contact noise has more emphasis on the high frequencies than a slow slide.

In order to analyze the contact sound further, four different slide events were recorded on a steel-string acoustic guitar. These recordings were done at Aalto University, School of Science and Technology in a small anechoic chamber. A microphone (AKG C 480 B) was placed 1 m away from the guitar and directed towards the sound hole. A Yamaha 01V digital mixer was used for digital recording (44.100 Hz sampling rate, 16 bits) of the signals and they were fed into a PC laptop via a Digigram VX Pocket soundcard. Figure 2.3 illustrates the spectrogram images of four different slide events.

The contact sound between the slide tube and wound strings is caused by the same phenomenon as when sliding a finger across a wound string. A thorough analysis of these handling noises on wound strings has been done by Pakarinen et al. (2007). It is apparent in all four cases in Figure 2.3 that the contact sound has some common timbral qualities. The slide velocity controls the cutoff frequencies of a low-pass type shape of the noise. The noise also has a clear harmonic structure. The frequencies of the harmonics depend on the slide velocity. The windings around the strings explain the harmonic structure of the slide noise, which is periodic in nature. This has also been noted with a Chinese string instrument called Gucin (Penttinen et al. 2006). There appear to be also wider, static harmonic resonances around 1.500 Hz. These are caused by longitudinal string vibration (Pakarinen et al. 2007).

The string type and slide tube material affect the resulting sound from a slide event. Thicker strings tend to have a more conspicuous high-frequency content and produce a louder contact noise than thinner strings. This probably results from the smaller windings of thinner strings, which makes the surface of the string smoother. The contact noise between a slide tube and an unwound string has no harmonic structure, and it closely resembles white noise. Unwound strings also produce a much quieter contact sound noise than wound strings. Also it should be noted that every time the slide tube makes contact with a string, a quiet, but audible percussive click sound is produced.

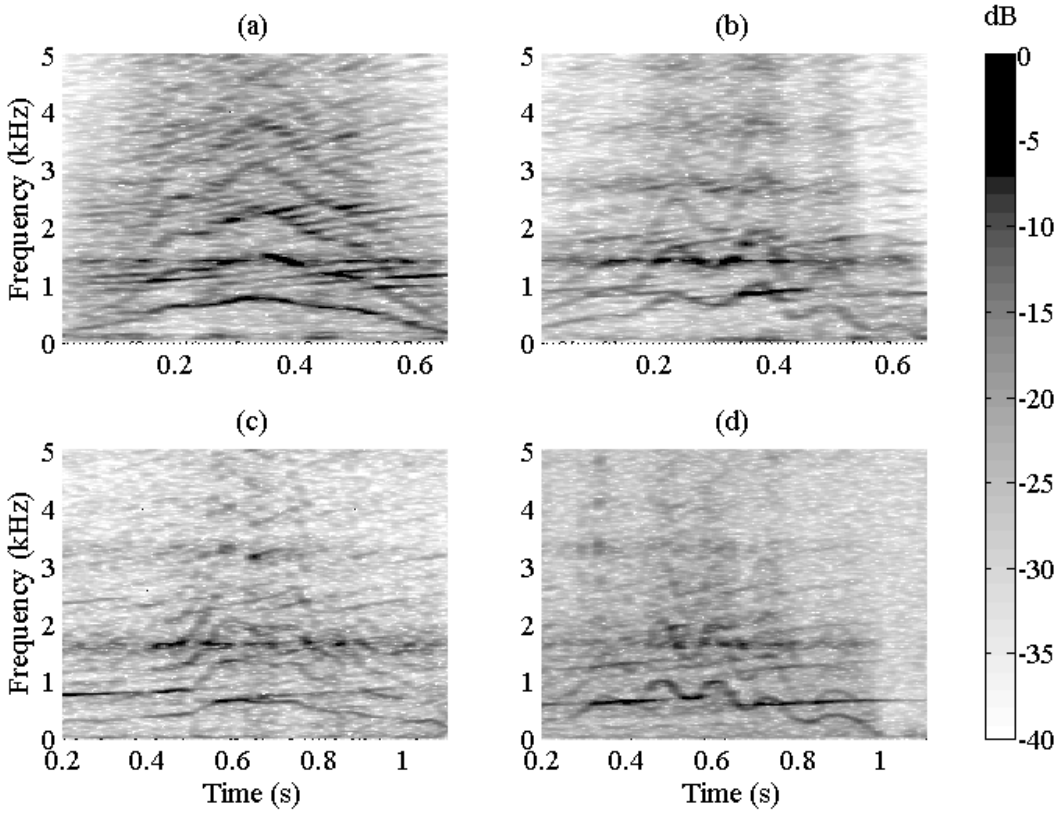

Figure 2.3: Contact noise spectrogram images of four recorded slide events. The tube materials and string numbers were chosen as follows: (a) brass tube with the 6th string, (b) glass tube with the 6th string, (c) brass tube with the 5th string, and (d) glass tube with the 5th string. The strings were damped in all four cases. Adopted from Pakarinen et al. (2008).

2.2 Slide tube measurements

To understand the differences between slide tubes made of different materials, it was decided to measure the impulse responses of three slide tubes made of different materials: glass, brass and a chromed slide tube. The slide tubes' impulse responses were recorded in the small anechoic chamber at Aalto University, School of Science and Technology with a microphone (AKG C 480 B). The signals were recorded into a Macintosh laptop via an Ediro UA-101 audio interface.

The slide tube was placed on the player's finger (in situ), a pen was dropped on the slide tube and the response was recorded. Several measurements were done for all the slide tubes. Figure 2.4 shows the averaged magnitude responses of the glass and brass tubes. As can be seen, the glass tubes magnitude response is much flatter than that of the brass tube. The brass tube's few sharp resonant peaks are too high in frequency (over 8 kHz) to be effectively coupled to the guitar body and radiated as audible sound 0. This leads to the conclusion that the audible differences between different slide tube materials are not created by the tube's vibration but the surface texture of the slide tube.

It was realized from the slide tube measurements that there are harmonic components that differ depending on the slide tube material. Both the glass and the brass tube had slow modulation but it was most notable with the slide tube made of glass. This might be caused by the mass differences between the slide tubes. The glass tube being lighter (14 g) than the brass tube (80 g) might allow for the player's smallest movements to transfer more easily to the slide tube's movement, which produces frequency modulation at the harmonic resonances. Differences in frictional characteristics of the tubes could also affect the contact noise.

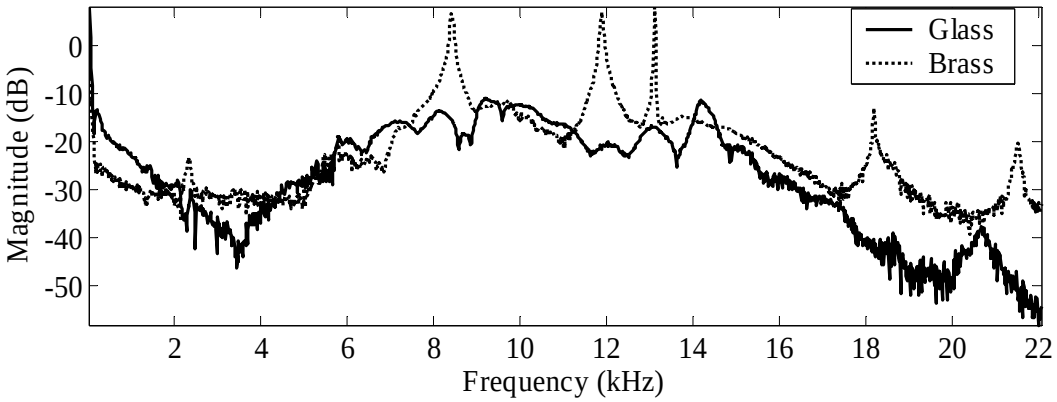

Figure 2.4: Averaged magnitude responses of the glass and brass slide tubes. The brass tube has sharp resonances near 8 kHz, 12 kHz and 13 kHz, while the glass tube is more heavily damped. Adopted from Pakarinen et al. (2008).

3 Synthesis

Chapter 3 presents the synthesis methods used in the real-time implementation. A time-varying digital waveguide string is used for synthesis of the slide guitar. The contact sounds that are produced by the string and slide tube are synthesized with a parametric model. The contact sounds are inserted into the waveguide as excitation.

3.1 Digital waveguide string

The basis of the slide guitar synthesis engine is formed with a single-delay loop (DSL) digital waveguide (DWG) (Karjalainen et al. 1998). This string model is illustrated in Figure 3.1. The waveguide comprises of a simple integer delay loop, a fractional delay filter and a loop filter. The loop filter simulates the vibrational losses in the string (Smith 2006). Both the integer delay line and fractional delay filter are time-varying. The total loop delay value is controlled by the user in real-time while playing the guitar. The loop delay value is directly proportionate to the distance between the player's hands

In order for the pitch to vary continuously and to enable correct tuning of the strings, a fractional delay filter is needed. A thorough tutorial for implementing fractional delay filters can be found in Laakso et al. (1996). In order to enable the pitch to vary continuously, a fractional delay filter is needed, because the regular integer delay line has a resolution of one sample. A fifth order Lagrange interpolator (Karjalainen et al 1998; Gasca et al. 2000) was found to be an optimal choice for our needs. It is short enough not to use too much processor time and its accuracy satisfies the quality needs. The Lagrange interpolator was implemented to work between 2.0 and 3.0 due to the fact that fractional delay filters are at their best accuracy near the midpoint of the filter, e.g. 2.5 for a 6-tap filter. Because of this the fractional delay filter has an overhead of 2 samples. This is compensated by making the integer delay line two samples shorter.

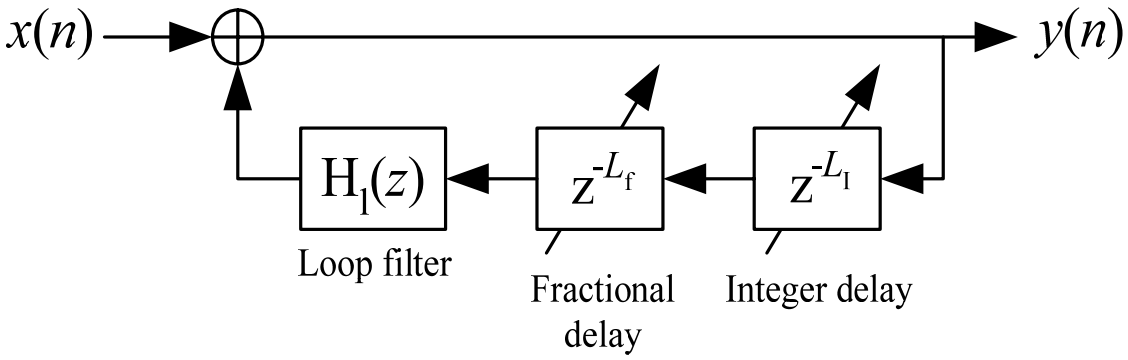

Figure 3.1: A time-varying digital waveguide string. The fractional and integer delay values are controller by the distance of the player's hands, so that when varying the distance between the player's hands, the pitch of the string will change smoothly. Adopted from Pakarinen et al. (2008).

3.1.1 Energy scaling

To prevent energy from being leaked out when increasing the string length, energy scaling is necessary. The signal energy of the DWG string varies when the length of the string is changed. When shortening the string, the signal samples that exceed the string length are discarded. This results in energy loss of the remaining signal in the string. Pakarinen et al. (2005) presents two energy-compensation methods. In order to save computation power needed from the computer, the simpler method was chosen. The zero-order energy-preserving interpolation adds a single scaling coefficient to the delay line. The scaling operation can be expressed as

$$p_c(n) = \sqrt{1-\Delta x}\, p(n) = g_c\, p(n), \qquad (1)$$

where n is time index, $p(n)$ is the signal output from the delay line, Δx is the delay-line variation in samples per one time step and g_c is the scaling coefficient.

3.2 Contact Sound

Based on the assumption that an exponentially decaying noise burst is generated when the slide tube slides over a single winding, a noise pulse train was chosen as the excitation signal to model the handling sounds generated by the slide tube when it passes over a winding on the string (Figure 3.2). The slide velocity controls the time interval between the noise pulses. Fast slides produce a temporarily dense pulse train and in opposition, a slow slide makes the adjacent pulses appear further apart from each other. In some sense, the contact sound synthesizer can be seen as a periodic impact sound synthesis model rather than a friction model. (Pakarinen et al. (2008). Impact and friction sound synthesis models are presented by Aramaki et al. (2006), Avanzini et al. (2005), Cook (1997), Peltola et al. (2007), Rath et al. (2005), Fontana (2003), and Rocchesso et al. (2003).

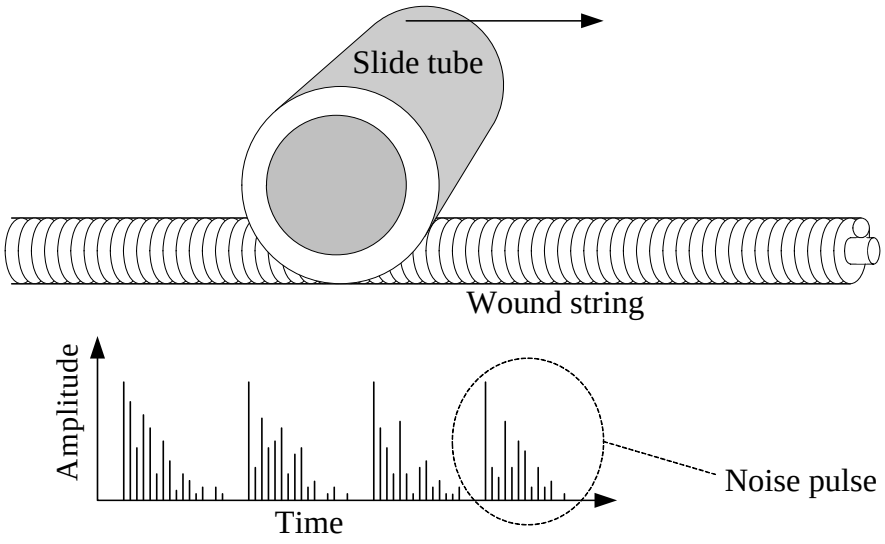

Figure 3.2: This figure depicts the idea behind the contact sound synthesis. A noise pulse is generated each time the slide tube passes a single winding on a wound string. Adopted from Pakarinen (2005).

A signal flow diagram of the contact sound generator block for wound strings is illustrated in Figure 3.2. The variable *L(n)* denotes the relative string length and *n* is the time index. n_w denotes the number of windings on the string and is used as a scaling coefficient. A time difference is taken from the relative string length because of the contact noises dependency on the sliding velocity. A separate smoothing block is needed after the differentiator because the sampling frequency and control rate of the input signal *L(n)* are different. The smoothing block converts the control rate of the input signal to that of the sampling rate. An absolute value of the smoothing block output is taken because the slide direction does not affect the contact noise. The signal $f_c(n)$ that is output from the smoothing block can be seen as the noise-pulse firing rate.

The noise pulse generator [block (a) in Figure 3.2] creates the basis of the contact sound generator. It feeds exponentially decaying noise pulses to the static and dynamic part of the contact sound model with given firing rate. String properties, such as decay time and pulse length determine the decay time and duration of each noise pulse. A second-order band-pass resonator [block (b)] is used to enhance the harmonic structure of the contact noise. This filter enhances the lowest time-varying harmonic while the firing rate controls the resonator's center frequency. A suitable nonlinear function [block (c)] is used to produce the higher harmonics. This is done by distorting the resonator's output with the scaled hyperbolic tangent function. By changing the scaling of this function can be used to alter the number of higher harmonics. This approach is similar to waveshaping synthesis (Arfib 1979; Le Brun 1979).

To simulate the static longitudinal string modes and the general spectral shape of the contact noise, a fourth-order IIR filter [block (d)] is used. Because the contact sound characteristics depend on the tube material and string type, different filter parameters are used for different string and slide tube combinations. The contact sound balance coefficient g_{bal} determines the ratio between dynamic (time-varying) and static contact sound components. The slide velocity $f_c(n)$ controls the total amplitude of the contact noise synthesis and it is scaled by a scaling coefficient g_{TV}. The user controls the overall volume of the contact sound via an external parameter g_{user}.

Unwound strings do not utilize the same contact sound generator as wound stings. It was agreed that it is sufficient to use low-passed white noise as contact noise for unwound strings.

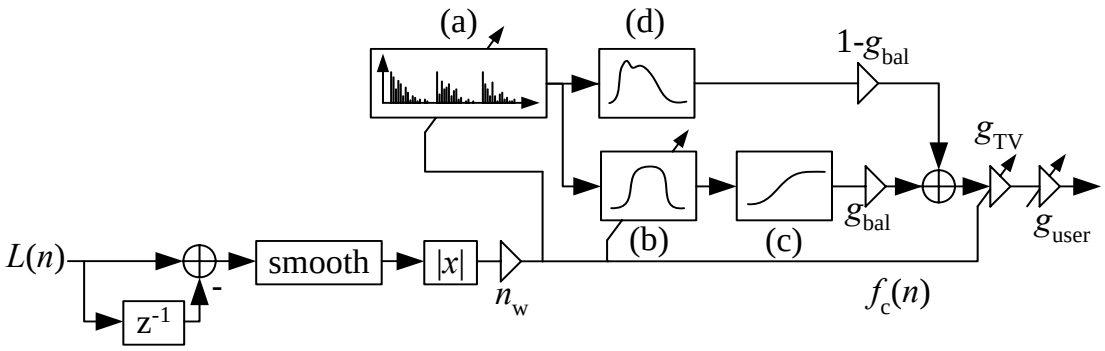

Figure 3.2: The contact sound generator block for wound strings. The sliding velocity controlled by the users hand movements commands the synthetic contact noise characteristics. The sub-blocks marked with the first four alphabets are (a) the noise pulse generator, (b) a resonator creating the first harmonic of the time-varying noise structure, (c) a nonlinearity generating the upper time-varying harmonics, and (d) an IIR filter simulating the general spectral characteristics of the noise. Adopted from Pakarinen et al. (2008).

Figure 3.3 illustrates the 4th order IIR filter magnitude responses used for the static longitudinal string modes as well as estimates of the contact sound's overall spectra. The spectral estimates were obtained using a linear-prediction (LPC) filter of order 100. The pole and zero frequencies and radii used for the contact sound filters are presented in Table A.1.

The whole slide guitar synthesis model for a single string is illustrated in Figure 3.4 This model resembles the ordinary time-varying SDL DWG string with the exception that it has the additional blocks for energy compensation and contact sound generation.

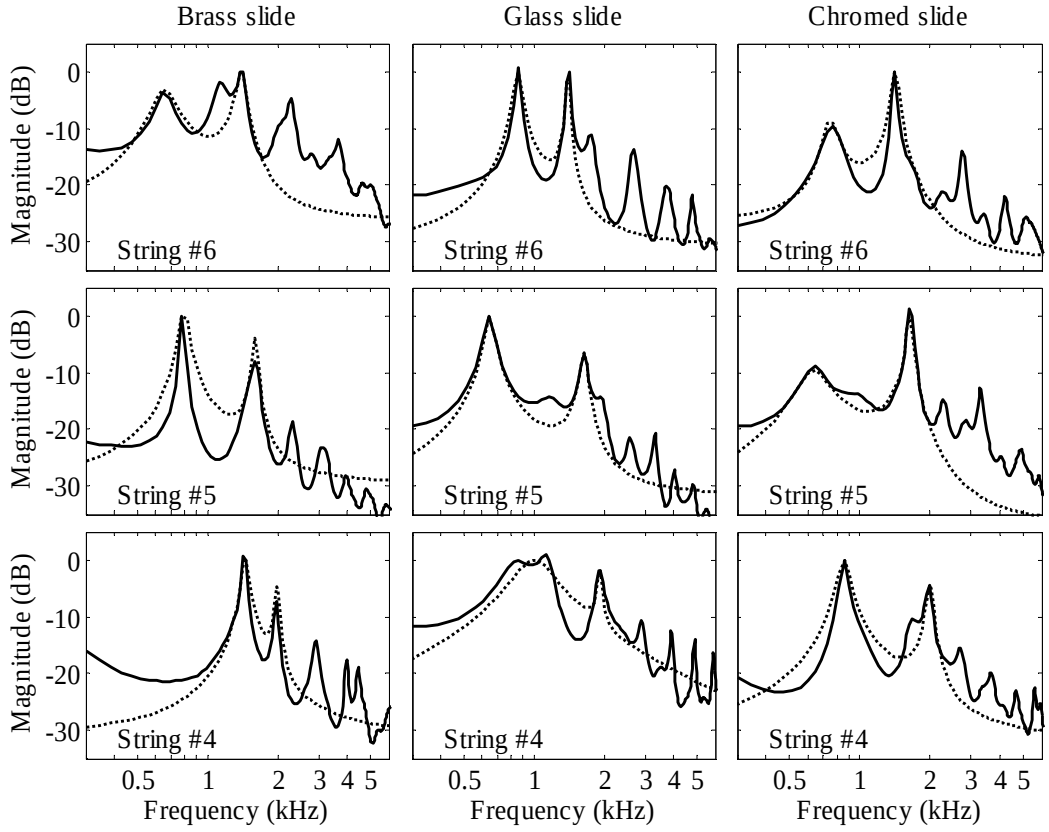

Figure 3.3: Estimates of the contact sound's overall spectra (solid lines), and the magnitude responses of the filters simulating them (dotted lines). The rows represent different strings (6^{th}, 5^{th}, and 4^{th} strings, from top to bottom) and the columns representing different slide tube types (brass, glass, and chromed, from left to right). Adopted from Pakarinen et al (2008).

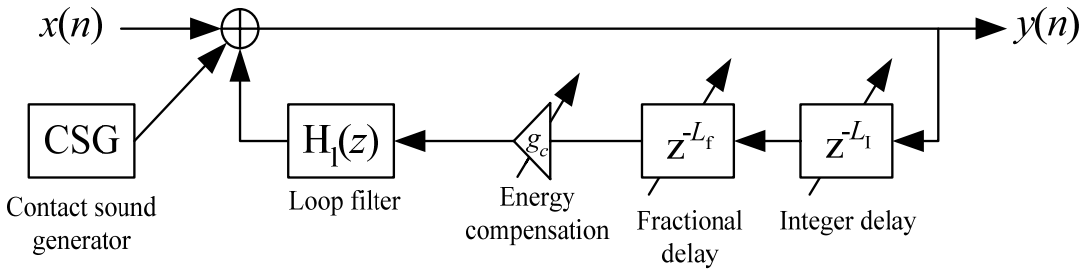

Figure 3.4: The signal flow diagram of the slide guitar string synthesizer. The energy compensation block compensates for the artificial energy losses due to the time-varying delays. The contact sound generator simulates the handling noise due to the sliding tube-string contact. Adopted from Pakarinen et al (2008).

4 Implementation

Chapter 4 presents the main components of the implementation. Implementation of the Virtual Slide Guitar was started by selecting a proper PC with enough computing power run the synthesis and camera software. Also a TrackIR 4 infra-red camera was chosen as the means of movement tracking and gesture recognition.

Since the VSG provides only an auditory feedback of the continuous pitch, the latency between the user's action and the resulting sound should be much smaller than in the VAG. For this reason, a high frame rate (120 fps) infra red (IR) camera is used for detecting the user's hand locations. The camera operates by lighting the target with IR-LEDs and sensing the reflected IR light. Therefore, for successful recognition, the user must have IR-reflecting material in his/her hands. A real slide tube coated with IR reflecting fabric is used for detecting the user's fretting hand. Using a real slide tube instead of a glove makes the VSG more intuitive for the user. For the picking hand recognition, a small ring of IR reflecting fabric is worn on the index finger.

4.1 Technical details

The implementation works on a 2.66 GHz Intel Pentium 4 CPU with 1 GB of RAM and a SoundMax Integrated Digital Audio soundcard. Both the sound synthesis part and the camera interface operate in the Windows XP environment. The sound synthesis uses PD (Pure Data) (Puckette, 1996) version 0.38.4-extended-RC8. The sampling frequency for the synthesis algorithm is 44.1 kHz, except for the string waveguide loop, which runs at 22.05 kHz, as suggested by Välimäki et al. (1996). A Naturalpoint TrackIR4 USB IR-camera is used for gesture recognition. The camera senses only IR light and thus IR-reflective materials. It has a viewing angle of 46° and it outputs a 355 x 290 binary matrix, where the reflected areas are seen as blobs.

4.2 Camera software

The camera software is modified from Naturalpoint's Optitrack SDK (Software Development Kit). The camera software enables the camera to send control data to the synthesis program. It calculates the positions of the blobs it sees. The software was modified to calculate the distance between two blobs, i.e. right and left hand, and send it to PD as an OSC (Open Sound Control) message. OSC is a content format for messaging between devices and programs. It is message transmission protocol similar to MIDI. For the camera API (Application Programming Interface), Naturalpoint's OptiTrack version 1.0.030 (Naturalpoint 2008) was used in the Visual Studio environment. The camera software was modified to include OSC communication. The software was also modified to keep track of the virtual string location, i.e. an imaginary line representing the virtual string. This is very similar to the work presented by Karjalainen et al. (2006). The line is drawn through the tube and the averaged location of the plucking hand, so that the virtual

string slowly follows the player's movements. The camera software detects the direction of the plucking hand movement when the virtual string is crossed. Once the string is crossed, a pluck event and a direction parameter is sent to PD. When the hands are kept still, a minimum velocity threshold is defined for the plucking gesture in order to avoid false plucks.

For more realistic playing, a pull-off feature was been added to the system. This means that the camera software switches the string length to maximum (65 cm) whenever the slide hand is opened. When the slide hand is closed, the string length is again set according to the distance between the user's hands. Thus, the user can lift the slide tube off the virtual strings, pluck open strings, and then press the tube on the strings again. This is typical for slide guitar playing. Opening the slide hand makes the tube finger to point to the camera so that the slide tube vanishes from the IR camera's view. When the tube is missing, the coordinates where the tube was last seen are used for setting the imaginary string's location. This way also open strings can be plucked, although the camera only detects one blob.

Since the slide tube and the fabric ring have quite different shapes, it is easy for the system to distinguish between them. In practice, this is done by selecting the more square-like blob as the ring and the longitudinal blob as the tube. This allows the instrument to be played by left-handed people as well.

There are two versions of the camera API which differ from each other only by the user interface. The basic version of the API has no user configurable interface and the camera image is not shown. This is recommended to be used with slower computers to save processor load. The pro version of the camera software is shown in Figure 4.1. On the main window (on the left) the image of the camera can be seen. The dot tracking part of the window shows the distance between the blobs (in pixels), how many blobs are detected and the surface areas of the blobs. The user can also select how often the display will be updated (every one frame, every two frames etc) or select the video image not to be displayed at all. The camera can also be stopped and started from this window. Clicking 'Options' reveals the window on the right side. From the camera options window the user can modify how the tracked blobs are ranked and by what factors the blobs are recognized. There are also sliders for setting the threshold, frame rate, exposure and intensity of the camera. By default there is no need to alter these settings, but in case of tracking problems these altering settings can help the tracking to work better.

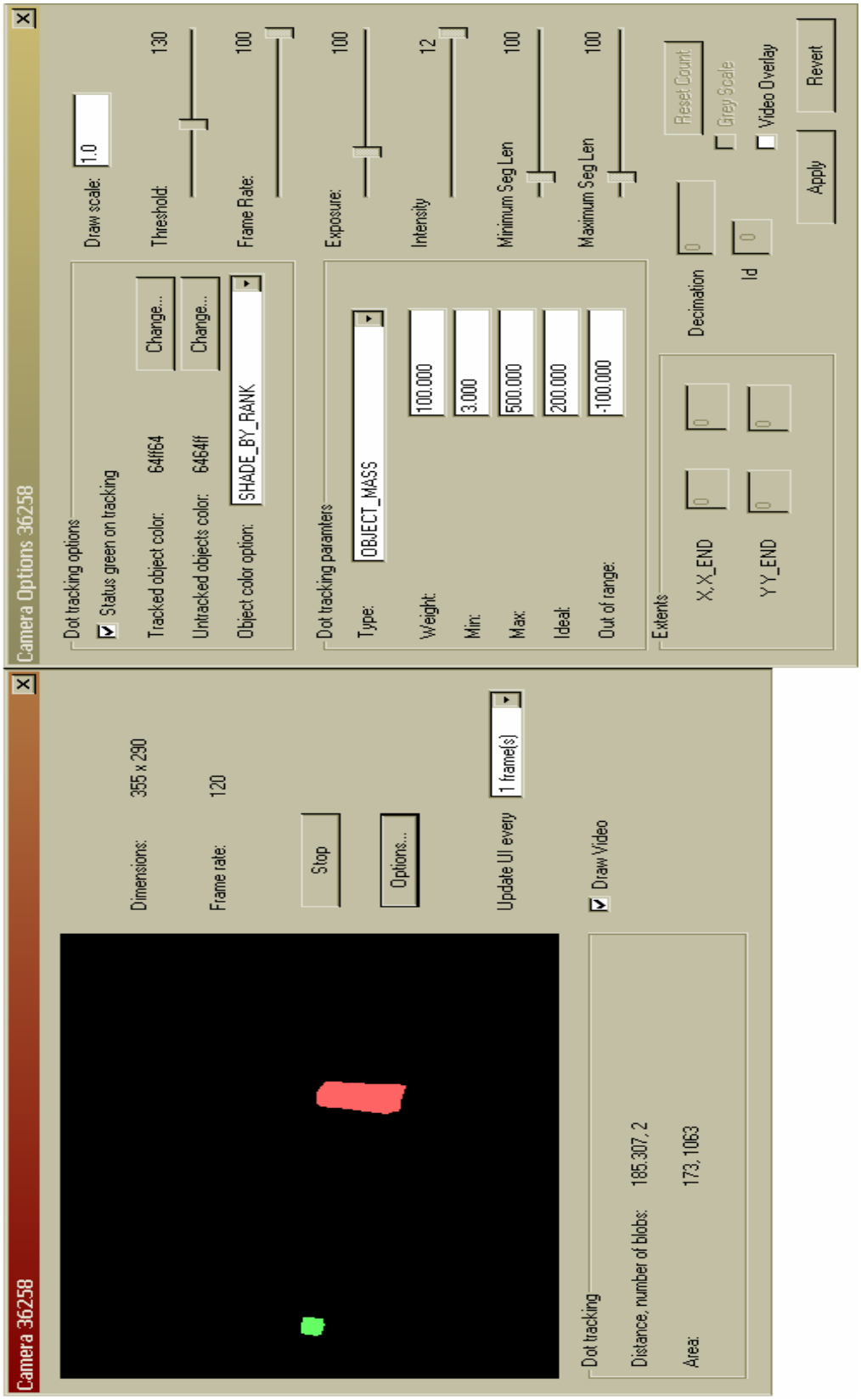

Figure 4.1: A screenshot of the camera API. This is the ‘pro’ version of the API.

4.3 System calibration

The system is calibrated so that the distance of 250 pixels corresponds to 48 cm when played approximately 2 m away from the camera. The distance is constrained in such a way that moving the hands further apart than 250 pixels does not make the strings any longer but will map them as open strings. Similarly, the minimum distance between the user's hands is constrained to 62.5 pixels (12 cm when played 2 m away), thus leading to a playing pitch range of an octave and a minor third for each string (from open string to the 15^{th} fret). Since the plucking hand is not normally positioned at the bridge of the guitar but near the sound hole, an offset of 17 cm is added to the distance to obtain the total length of the strings (65 cm for open strings). The minimum playable length of the string is 29 cm which corresponds to the 15^{th} fret of the guitar. A shorter minimum length would not always work well due to the hands being positioned close to each other. The camera might confuse the two separate blobs as one and therefore playing would not work. For some guitars it is not even practical to play on the highest frets, which are usually around the 20^{th} fret. A minimum string length limitation can also be justified because the length of the virtual string varies more with a shorter distance when plucking the guitar. The distance is calculated as a straight line between the two hands so for the distance to not vary while plucking, a circular movement should be used when plucking. The distance between the hands is normalized by dividing it with the open string length. This results in a relative string length (variable L(n) in Figure 3.2) between 0.446 and 1. Since the usage of a slide tube effectively makes every string have the same playing length, this normalized string length is used as a control signal for each of the synthesized strings.

4.4 Pure Data Implementation

Pure Data (PD) is a freeware real-time graphical programming environment for audio and graphical processing (Puckette). PD uses visual objects that are placed on the screen (called a canvas). These objects can be functions, variables, sliders, number boxes, messages or other PD patches. The boxes can be connected to each other by lines – like cables - that represent signal connections. There are two types of signals in PD: control signals and audio signals. Audio signal connections are represented by thick lines and control signal connections by thin lines. PD allows sub-programs (called sub-patches) to be nested inside the parent patch and the use of external patches (called abstractions) located in separate files. One major advantage of PD is that it works in real-time. In contrast to traditional programming languages where the code must first be processed before obtaining a result, PD code can be changed while the program is running. This helps with debugging and experimenting. (Puckette 2007)

When the Pure Data implementation receives an OSC message containing a pluck event, an excitation signal is inserted into each waveguide string. The excitation signal is a short noise burst simulating a string pluck. A slight delay (20 ms) is also inserted between different string excitations for creating a more realistic strumming feel. The order in which the strings are plucked depends on the plucking direction.

The PD implementation can barely run all the strings in real time with the hardware and software described above. To lower the computational load of the synthesis, strings can be switched totally off. The instrument can be then played with lower end computers also. Due to their heavier computational requirements for contact sound generation, wound strings need much more computational power than unwound strings. Switching the contact sound synthesis off altogether also reduces the amount of CPU power needed dramatically. It is not possible to play all the strings with the above mentioned setup if FreeAmp2 is enabled. Figure 4.2 illustrates the PD implementation's on-screen user interface.

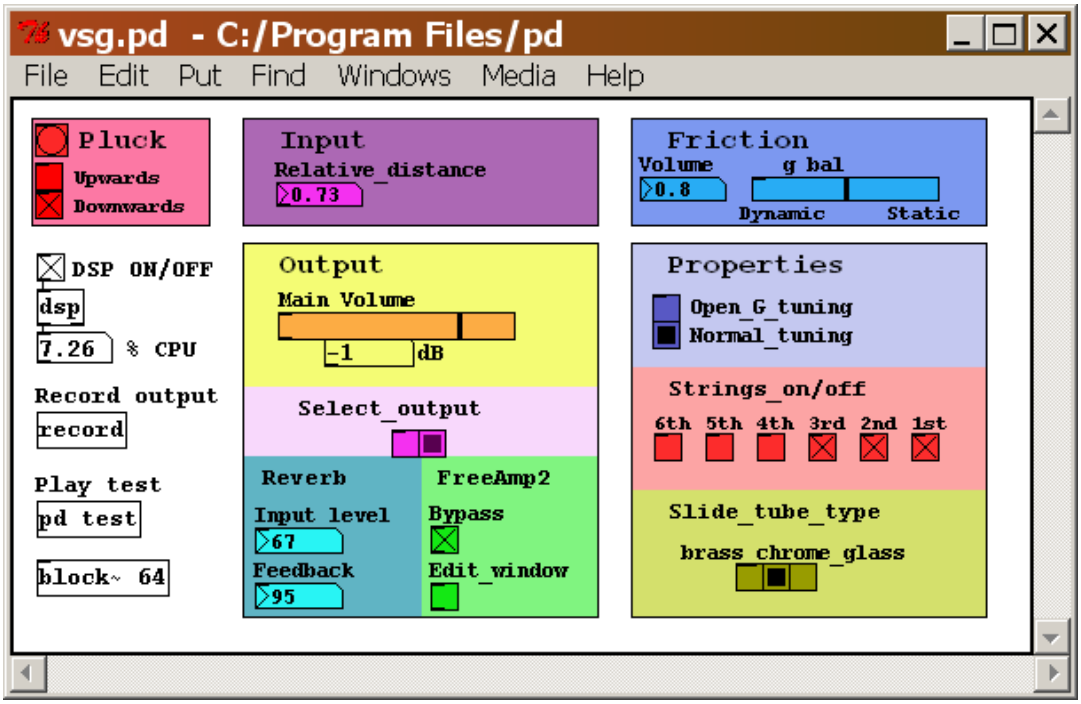

Figure 4.2: A screenshot of the Pure Data user interface. In this example, the strings are in open G tuning, the highest four strings are played with a glass slide tube and the dynamic contact sound is slightly emphasized. No effect or plugin is user for output (bypass is selected).

The overall latency in PD can be set to a minimum of 20 ms by using ASIO (Audio Stream In/Out) sound drivers. With MMIO (Memory Mapped Input/Output) sound drivers the lowest manageable latency was 70 ms. For additional effects, VST (Virtual Studio Technology) plugins can be used with PD, but they tend to require quite much valuable processing power. The PD implementation has options to output the sound as it is, through a reverb effect (modified from PD audio examples) or through a FreeAmp2 VST plugin (Fretted Synth Audio 2007).

Figure 4.3 illustrates the different modules and signaling between abstractions in PD. The whole program consists of three main parts: the main patch (with control options and user interface), string synthesis, and contact sound synthesis. Sub-patches and abstractions are switched off when they are not needed in order to save computing time. For example, reducing the contact sound volume to zero switches all contact sound computation off. Since the waveguide loop runs at half the sampling rate, anti-aliasing filters (e.g. two-tap averagers, $H(z) = (1+z^{-1})/2$) are required at its input and output (Lavry 2004). The string synthesis abstraction converts the relative distance to frequency and delay line length, calculates the loop filter parameters depending on string length, as suggested in 0, sends the pluck excitation and contact sound to the delay line, and implements energy scaling and anti-aliasing filtering. The contact sound synthesis abstraction receives string parameters through the string synthesis abstraction and generates contact noise according to the slide tube type, string properties, and hand movements.

The Pure Data part of the VSG consists of the main patch, which manages input signals, sends control data to the string synthesis components, handles the user interface and outputs the final guitar sound (Figure 4.2). The string synthesis component consists of pluck excitation, anti-aliasing and holds the delay line with its filters, delays and energy scaling. The contact sound synthesis component generates the contact sound between the slide tube and strings. It holds the noise burst generator for wound strings and low-pass noise generator for unwound strings. Every string has its own instance of the string synthesis and contact sound synthesis components which work independently of other strings.

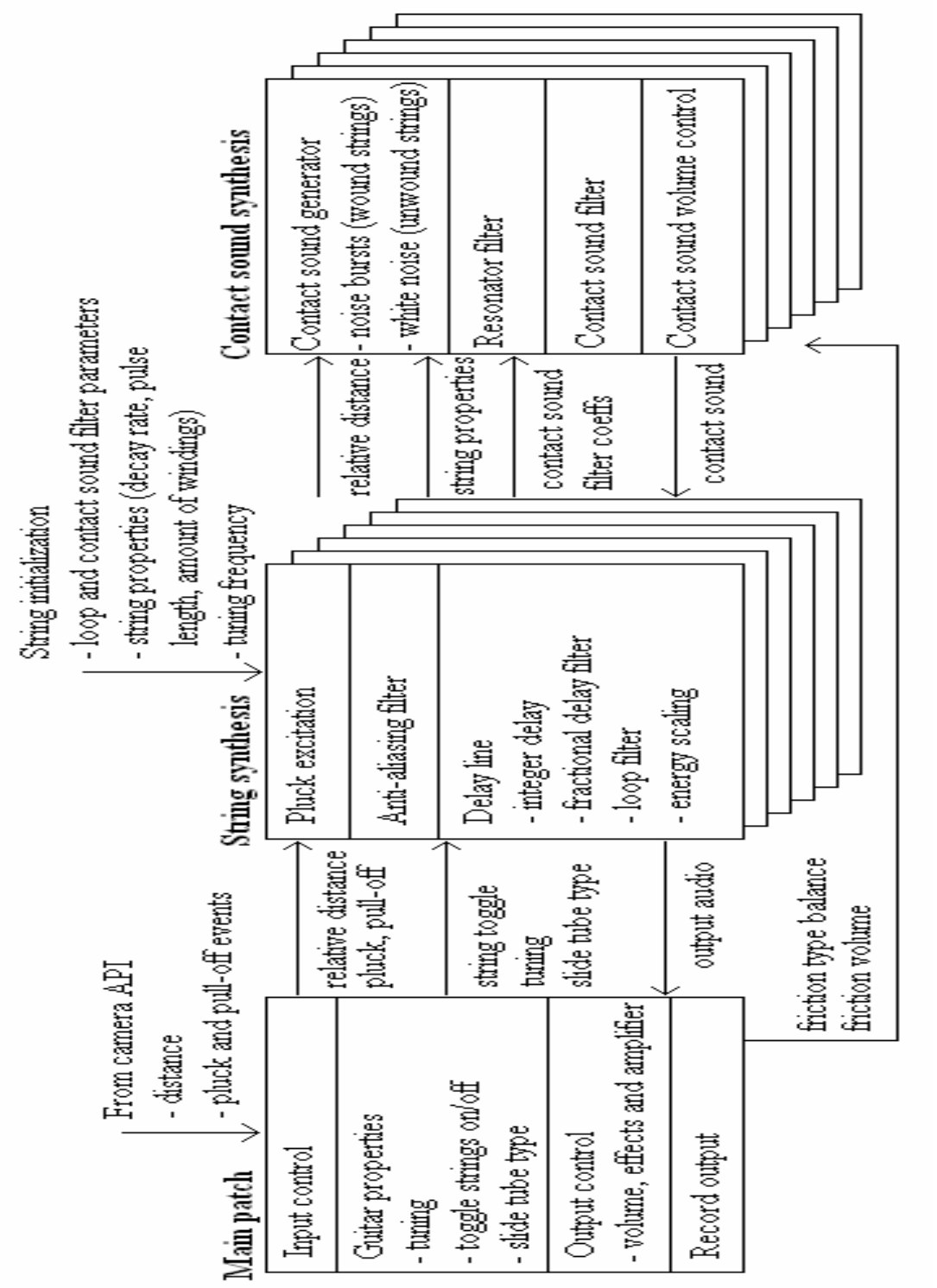

Figure 4.3: The construction and signaling of the Pure Data synthesis implementation.

5 Virtual Slide Guitar

In this chapter the Virtual Slide Guitar setup will be reviewed as a whole.

The virtual slide guitar system is illustrated in Figure 5.1. At its simplest, the VSG is easy to play and needs no calibration. The user only has to put the slide tube and plucking ring on, select which strings to play and start moving his or her hands. For more demanding users, the VSG provides extra options, such as altering the tuning of the instrument, selecting the slide tube material, setting the contact sound volume and balance between static and dynamic components, or selecting an output effect. The user interface can be set to show the blob positions on the screen, but it is not required for playing. Drawing the blobs adds to the computational load of the system.

Generally, it may seem that the VSG is not as easy to play as the VAG, because there is more freedom and options for the player. The VAG offers the user only a few chords or notes to play and might thus at first sound nicer to the audience, but this severely reduces the expressive range. The additional VSG features add versatility for playing, but a short training session is recommended to get the most out of this virtual instrument. The tube-string contact sound gives the user direct feedback of the slide tube position, so visual feedback is not necessarily needed in order to know where the slide tube is situated on the imaginary guitar neck. Switching between the slide tube types results in different kinds of contact sounds, but it is difficult to distinguish the tube material from the synthesized sound only. This might be due to the fact that the perceptually most important contact sound material cue, the frequency-dependent decay rate (Klatzky et al. 2000) of the tube, is not apparent in the synthesized sound.

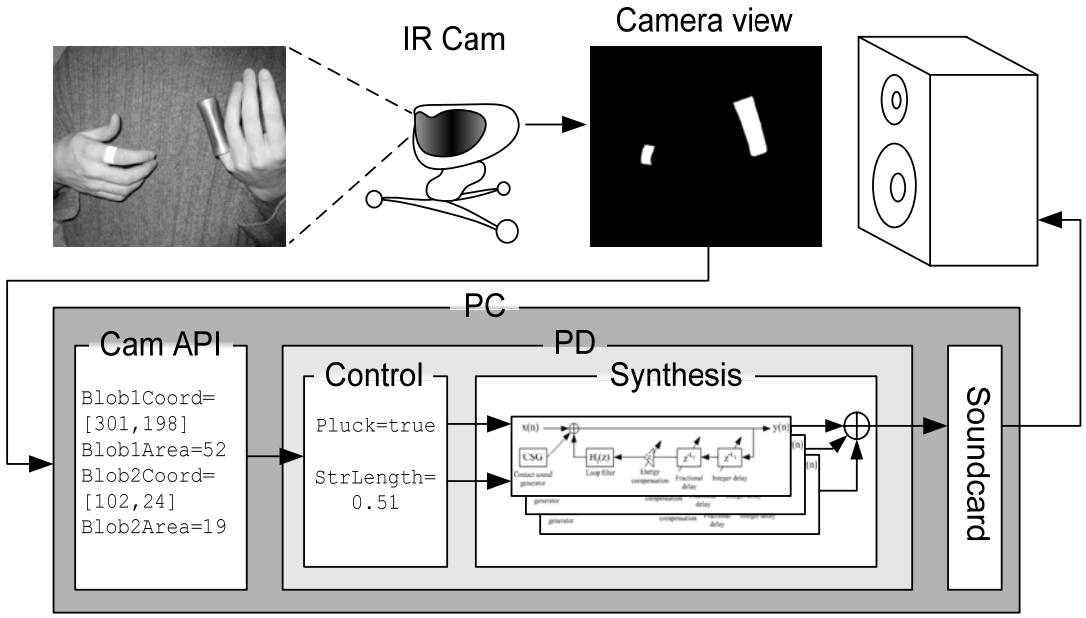

Figure 5.1: The Virtual Slide Guitar system. It consists of an IR-camera, a PC and a loudspeaker(s). The camera API calculates the string length from the hand locations and sends the coordinates to the synthesis control block in Pure Data. The synthesis block creates the sound and outputs it to the soundcard and eventually a loudspeaker or headphones. Adopted from Pakarinen et al. (2008).

Different methods for real-time implementation were considered during the starting period of the work. A normal web camera was considered, but it was decided, that an IR camera will work better since it has a higher frame rate than a normal web camera. For the camera application, PD's GEM along with Eyesweb (Camurri et al. 2000, 2005; Gorman et al. 2007) were considered, but both of them were discarded after realizing, that a free SDK is available for the IR camera. Also, for the IR camera software, no image manipulation was needed since the SDK calculates the blob positions by default.

The camera itself does not need calibration, unless the circumstances in which it is played dramatically change. For the distances of the strings and notes to apply to a real guitar neck and body, an approximate distance of 2 m was measured to be satisfying. The camera needs to be pointed straight towards the player, naturally, otherwise either hand could disappear from the camera's view when playing the instrument. Since the camera differentiates between the users pluck hand and fretting hand by comparing the shape and size of the blobs it sees, the VSG can be played by left-handed people just as well as right-handed. In fact, you could even play it with your feet, although it is not very practical.

The PD patch includes intelligence and automation to minimize the use of system resources. The patch turns off some of the sub-patches and abstractions when they are not needed. The signal is routed only to the synthesis blocks and contact noise generators of strings that are in use. When the contact noise balance is switched completely to static or dynamic contact noise, the algorithms to generate the one not used is automatically switched off to save CPU power. Alternatively, if the

contact noise volume is lowered to zero, every abstractions and patches involved in generating and modifying the contact noise are switched off. Also, as mentioned before, the string synthesis runs on half the sample rate

Since the synthesis part of the VSG is done in Pure Data, it is possible to expand or modify any aspect of the synthesis, or even add your own PD effects. It is quite easy to import your own PD patches, abstractions, effects to any existing PD program, as longs as you know how to use PD (Zimmer 2006). The VSG runs on Windows XP and it has not been tested on any other operating system. The synthesis part might run on a Mac but the camera API is for Windows only.

The VSG was installed on a demo station in the Laboratory of Acoustics and Audio Signal Processing. During the implementation stages of the VSG in 2007, a quite good PC was chosen so that the VSG can be played with several strings at a time. However the 2.66GHz Intel Pentium 4 is not powerful enough to run synthesize all strings at once if the camera image is drawn by the camera API. A more powerful setup has yet to be tried to ensure what kind of processing power is needed for a simultaneous synthesis of all the strings with camera image drawn and a guitar effect in use. However since processors have developed immensely there is little doubt that a modern semi-high-end multi-core processor would not suffice.

The PD user interface consists of tuning, string and slide tube selection, volume control, static/dynamic friction noise balance and volume controls, and output and plugin selection. Additionally, the real-time value of the relative guitar string length is displayed in the main patch window. There is also a small abstraction in the user interface of the PD program, which allows users to record what they are playing.

The camera API displays the positions of the slide tube and plucking hand. The display can be turned off to minimize processor usage. The user can also change the update frequency of the camera frames (every frame, every other, every third etc) to reduce needed processing power, as well as modify blob tracking settings.

6 Conclusions

In this thesis, a real-time sound synthesis model of a slide guitar with air guitar interface is presented. The string vibrations are simulated by means of energy-compensated time-varying digital waveguides. Mechanism involved in generating the contact sound between slide tubes and strings are analyzed. Also different slide tubes have been measured and analyzed. A new parametric model for generating the contact noise between the string and the slide tube was introduced. The slide guitar synthesizer operates an infra-red camera for optical gesture recognition user interface. Slide guitar sound analysis and technical issues of the synthesis model were discussed in detail.

There is room for improvement on the synthesis program; calculations and control data management could be tuned to require less processing time. Some parts of the calculations used for string and contact sound synthesis could be calculated only once collectively instead of separately for each string in use. Also the plucking movement varies the string length because the distance of the hands is measured from an imaginary straight line drawn between the player's hands and the imaginary line follows the hands' movement with little delay. This could be corrected by modifying the string length calculation algorithm to compensate for the unintentional stretching of the imaginary string when moving the plucking hand away from the string. Although not a major problem, occasionally some jewelry, glasses or other metallic or reflective materials in the cameras view can trigger false gesture recognitions and mess up the sound. Longitudinal string vibrations are not synthesized but simulated with fixed filters, which is done to simplify computation of the synthesis program. This results in a less dynamic spectrum.

Altogether the VSG is a fun instrument that anyone can play with a low threshold. It sounds like a real guitar and using a guitar effect in the output makes it sound even more real. Current commercial guitar games utilize a rigid instrument and an air guitar interface might be the next step in musical games. Due to the versatility of Pure Data and the possibility to use OSC signals to control practically anything, the camera interface could be used to control any imaginable Pure Data (or other) software.

References

Aramaki, M., and R. Kronland-Martinet 2006. “Analysis-Synthesis of Impact Sounds by Real-Time Dynamic Filtering.” *IEEE Transactions on Audio, Speech and Language Processing* 14(2):695–705.

Aramaki, M., R. Kronland-Martinet, T. Voinier, and S. Ystad. 2006. “A Percussive Sound Synthesizer Based on Physical and Perceptual Attributes.” *Computer Music Journal* 30(2):32–41.

Arfib, D. 1979. “Digital Synthesis of Complex Spectra by Means of Multiplication of Nonlinear Distorted Sine Waves.” *Journal of the Audio Engineering Society* 27(10):757–768.

Avanzini, F., S. Serafin, and D. Rocchesso. 2005. “Interactive Simulation of Rigid Body Interaction with Friction-induced Sound Generation.” *IEEE Transactions on Speech and Audio Processing* 13(5):1073–1081.

Camurri, A., S. Hashimoto, M. Ricchetti, R. Trocca, K. Suzuki, and G. Volpe. 2000. “EyesWeb—Toward Gesture and Affect Recognition in Interactive Dance and Music Systems.” *Computer Music Journal* 24:1:57–69.

Camurri, A., G. Volpe, G. De Poli, and M. Leman. 2005. “Communicating Expressiveness and Affect in Multimodal Interactive Systems.” *IEEE Multimedia* 12(1):43–53.

Cook, P. R. 1997. “Physically Informed Sonic Modeling (PhISM): Synthesis of Percussive Sounds.” *Computer Music Journal* 21(3):38–49.

Fontana, F. 2003. “Physics-based Models for the Acoustic Representation of Space in Virtual Environments.” Ph.D. dissertation, Univ. Verona, Verona, Italy. Available online: http://profs.sci.univr.it/~fontana/paper/20.pdf (checked 24.5.2010).

Fretted Synth Audio. 2007. “Free Amp 2“.

Gasca, M., and T. Sauer. 2000. “Polynomial Interpolation in Several Variables.” *Advances in Computational Mathematics,* 12(4):377-410.

Gorman, M., A. Lahav, E. Saltzman, and M Betke. 2007. “A Camera-Based Music-Making Tool for Physical Rehabilitation.” *Computer Music Journal* 31(2):39–53.

Hellmer, E. “Guitar Facts”, http://www.evansvillechristian.org/technology/WebPages/Hellmer%20Web/Guitar.html (checked 25.5.2010).

Johnson, R. 1990. “The Complete Recordings.” Legacy Recordings, C2K46222.

Karjalainen, M., T. Mäki-Patola, A. Kanerva, and A. Huovilainen. 2006. “Virtual Air Guitar.” *Journal of the Audio Engineering Society* 54(10):964–980.

Karjalainen, M., V. Välimäki, and T. Tolonen. 1998. “Plucked-String Models: From the Karplus-Strong Algorithm to Digital Waveguides and Beyond.” *Computer Music Journal* 22(3):17–32.

Karjalainen, M., V. Välimäki, and Z. Jánosy. 1993. ”Towards High-Quality Sound Synthesis of the Guitar and String Instruments.” International Computer Music Conference, Tokyo, Japan.

Klatzky, R. L., D. K. Pai, and E. P. Krotkov. 2000. “Perception of Material from Contact Sounds.” *Presence: Teleoperators and Virtual Environment* 9(4):399-410.

Laakso, T. I., V. Välimäki, M. Karjalainen, and U. K. Laine. 1996. “Splitting the Unit Delay—Tools for Fractional Delay Filter Design.” *IEEE Signal Processing Magazine* 13(1):30–60.

Lavry, D. 2004. “Sampling Theory for Digital Audio.” Lavry Engineering, Inc. Available online: http://www.lavryengineering.com/documents/Sampling_Theory.pdf (checked 24.5.2010).

Le Brun, M. 1979. “Digital Waveshaping Synthesis.” *Journal of the Audio Engineering Society* 27(4):250–266.

Naturalpoint Inc. 2008. Optitrack 1.0.030 SDK and Manual, http://www.naturalpoint.com/ (checked 24.5.2010).

Pakarinen, J., M. Karjalainen, V. Välimäki, and S. Bilbao. 2005. “Energy Behavior in Time-Varying Fractional Delay Filters for Physical Modeling of Musical Instruments.” *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 2. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, pp. 1–4.

Pakarinen, J. 2005. “Slide-kitarasynteesi soitinmallinnuksen avulla”. *Suomen musiikintutkijoiden symposiumin satoa.* Suomen musiikintutkijoiden 9. Valtakunnallinen Symposium, Jyväskylä Finland, pp. 105-109.

Pakarinen, J., H. Penttinen, and B. Bank. 2007. “Analysis of the Handling Noises on Wound Strings.” *Journal of the Acoustical Society of America* 122(6):EL197–EL202.

Pakarinen, J., T. Puputti, and V. Välimäki. 2008. ”Virtual Slide Guitar.” *Computer Music Journal* 32(3):42-54.

Pakarinen, J., V. Välimäki, and T. Puputti. 2008. ”Slide Guitar Synthesizer with Gestural Control.” *Proceedings of 8th International Conference on New*

Interfaces for Musical Expression (NIME08). Genoa, Italy, June 5-7, 2008, pp. 49-52.

Pakarinen, J. 2008 “Modeling of Nonlinear and Time-Varying Phenomena in the Guitar.” Doctoral dissertation, Helsinki University of Technology.

Paradiso, J.A. 1997. “Electronic Music: New Ways to Play.” *IEEE Spectrum* 34(12):18–30.

Peltola, L., C. Erkut, P. R. Cook, and V. Välimäki. 2007. “Synthesis of Hand Clapping Sounds.” *IEEE Transactions on Audio, Speech, and Language Processing* 15(3):1021–1029.

Penttinen, H., J. Pakarinen, V. Välimäki, M. Laurson, H. Li, and M. Leman. 2006. “Model-Based Sound Synthesis of the Guqin.” *Journal of the Acoustical Society of America* 120(6):4052–4063.

Penttinen, H. 2006 “Loudness and timbre issues in plucked stringed instruments – analysis, synthesis, and design.” Doctoral dissertation, Helsinki University of Technology.

Puckette, M. 1996. “Pure Data: Another Integrated Computer Music Environment.” *Proceedings of the 1996 International Computer Music Conference.* San Francisco, California: International Computer Music Association, pp. 269–272.

Puckette, M. 2007 “Theory and Techniques of Electronic Music.” World Scientific Books. http://crca.ucsd.edu/~msp/techniques.htm (checked 24.5.2010).

Puckette, M. Pure Data Documentation. http://www.crca.ucsd.edu/~msp/Pd_documentation/index.htm (checked 24.5.2010).

Rath, M., and D. Rocchesso. 2005. “Continuous Sonic Feedback from a Rolling Ball.” *IEEE Multimedia* 12(2):60–69.

Rocchesso, D., R. Bresin, and M. Fernström. 2003. “Sounding Objects.” *IEEE Multimedia* 10(2):42–52.

Rocchesso, D., and F. Fontana, Eds. 2003. *The Sounding Object*. Florence, Italy: Edizioni di Mondo Estremo. Available online: http://www.soundobject.org/ (checked 24.5.2010).

Smith, J. O. 1992. “Physical Modeling using Digital Waveguides.” *Computer Music Journal* 16(4):74–91.

Smith, J. O. 2006. “Elementary Digital Waveguide Models for Vibrating Strings.” *Lecture Overheads, Music 420 / EE367.* Center for Computer Research in Music and Acoustics (CCMRA), Stanford University, California.

Välimäki, V., J. Huopaniemi, M. Karjalainen, and Z. Jánosy. 1996. “Physical Modeling of Plucked String Instruments with Application to Real-Time Sound Synthesis.” *Journal of the Audio Engineering Society* 44(5):331–353.

Välimäki, V., and T. Tolonen. 1998. “Development and Calibration of a Guitar Synthesizer.” *Journal of the Audio Engineering Society* 46(9):766–778.

Välimäki, V., J. Pakarinen, C. Erkut, and M. Karjalainen. 2006. ”Discrete-Time Modelling of Musical Instruments.” *Reports on Progress in Physics* 69(1):1-78.

Wanderley, M., and P. Depalle. 2004. “Gestural Control of Sound Synthesis.” *Proceedings of the IEEE* 92(4):632-644.

Yadegari, S. 2003. “A General Filter Design Language with Real-time Parameter Control in Pd, Max/MSP, and jMax.” *Proceedings, International computer Music Conference, Singapore.* San Francisco: International Computer Music Association, pp. 345-284. Available online:

http://www.crca.ucsd.edu/%7Esyadegar/Publications/YadegariICMC2003.pdf (checked 24.5.2010).

Zimmer, F. (ed). 2006. “bang. Pure Data”, Institut Für Elektronische Musik und Akustik, PD-Convention 2006, Graz, Austria. Available online: http://pd-graz.mur.at/label/book/bangbook.pdf (checked 24.5.2010).

Appendix A

Table A.1: Pole and zero frequencies and radii for the 4th-order IIR contact sound filter used for wound strings. Magnitude responses are illustrated in Figure 3.3. Adopted from Pakarinen et al. (2008).

		Brass tube		Glass tube		Chromed tube	
		F (Hz)	R	F (Hz)	R	F (Hz)	R
6th string	zeros	0	0.9485	0	0.9272	±696	0.9608
		0	0.8510	0	0.8222	±1422	0.8042
		±1400	0.9079	±1400	0.9608	-	-
	poles	±643	0.9894	±850	0.9957	±748	0.9929
		±1400	0.9922	±1400	0.9984	±1422	0.9937
5th string	zeros	0	0.9406	0	0.9646	0	0.9686
		0	0.8105	0	0.7902	0	0.7752
		±1600	0.9478	±1640	0.9217	±1640	0.8042
	poles	±793	0.9957	±644	0.9957	±622	0.9859
		±1600	0.9948	±1640	0.9922	±1640	0.9937
4th string	zeros	0	0.8727	0	0.9887	0	0.9644
		0	0.7269	0	0.0543	0	0.6564
		±2000	0.9687	±1920	0.9826	±2000	0.9217
	poles	±1449	0.9930	±980	0.9720	±859	0.9929
		±2000	0.9948	±1920	0.9948	±2000	0.9922

Appendix B

In this appendix Pure Data source code of this thesis is presented. All patches, abstractions and sub-patches essential to the functionality of the implementation are shown here. The source code is presented in image format for clarity reasons and also because Pure Data is a graphical programming language.

PD functions can have multiple inputs and multiple outputs. A thick line presents an audio signal line and a thin line presents a control signal line. A tilde (~) behind a function means it is an audio signal function. A $0- in front denotes a local variable or sub-patch to keep individual string parameters or objects from mixing.

Below are descriptions of used PD functions and denotations that are not self-explanatory:

$f1, $f2… = separate input values for expr, expr~ and fexpr~ functions
abs = take absolute value
b = bang, activates an object or number box
biquad~ = a 2 pole 2 zero filter
block~ = set block size of computation, overlap and up/downsampling
catch~ = receive audio signal
bpq2~ = 2^{nd} order band-pass filter
del = send a bang after delay (ms)
delwrite~ = writes a signal in a delay line
expr = arithmetic expression evaluation (Yadegari 2003)
expr~ = arithmetic expression evaluation for audio signals
fexpr~ = builds FIR and IIR filters by evaluating expressions sample by sample
hip~ = high-pass filter
lop~ = low-pass filter
metro = metronome function
moses = switches output according to set value
noise~ = white noise generator
pink~ = pink noise generator
pipe = delay a message (ms)
r = receive signal
s = send signal
sel = select output
sig~ = convert to audio signal
switch~ = enable/disable patch and set block size, overlap and up/downsampling
t = trigger
t b = trigger bang, sends a bang when triggered
tanh~ = hyperbolic tangent function
throw~ = send audio signal
unsig = convert to control signal
vd~ = read a signal from delay line at a variable delay time
vline~ = audio ramp generator
z~ = samplewise delay

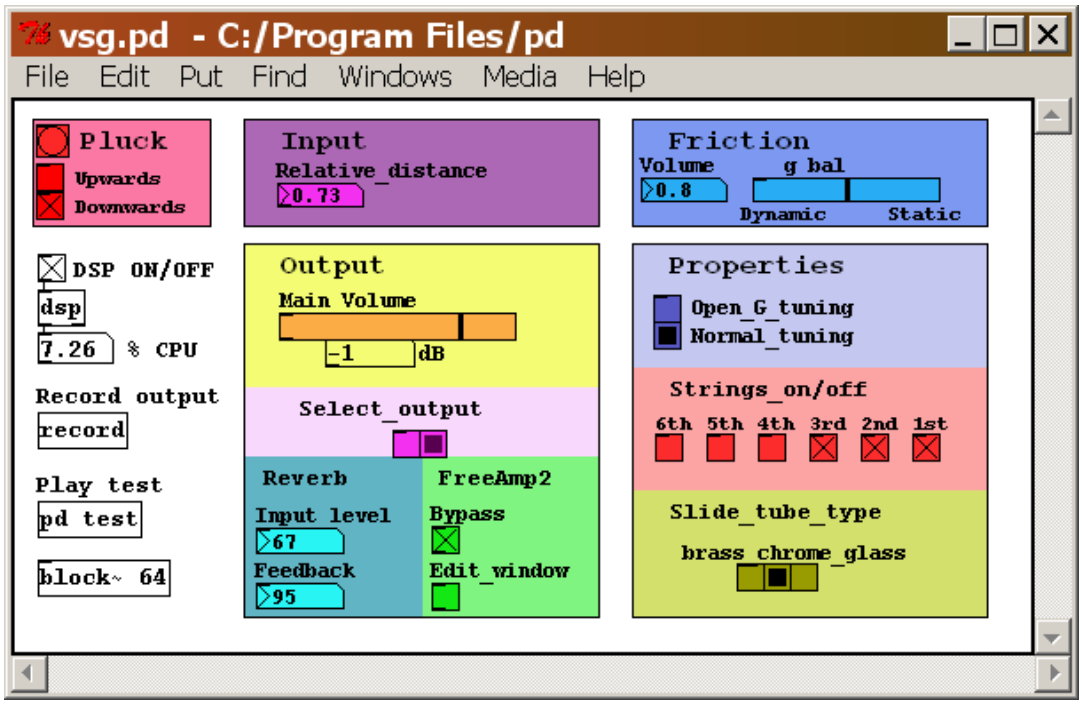

Figure B.1: The main user interface of the VSG Pure Data program. The user can choose the preferences of the synthesis from this screen. From the top left corner the user can choose the direction(s) of plucking movement will excite the string(s). The top middle part displays the relative distance of the strings. The top right corner displays overall contact sound volume and balance between static and dynamic contact sound models. The lower right part displays string settings. From there the user can select which strings to play, which tuning and slide tube type to use. The lower middle part consists of output settings, which are the main output volume and guitar effects. From the lower left part the user can switch DSP computations off for the whole program, record output and play a fretted test guitar. All the check-boxes, selectors, number boxes (except % CPU) and the pluck button can be manually altered by the user.

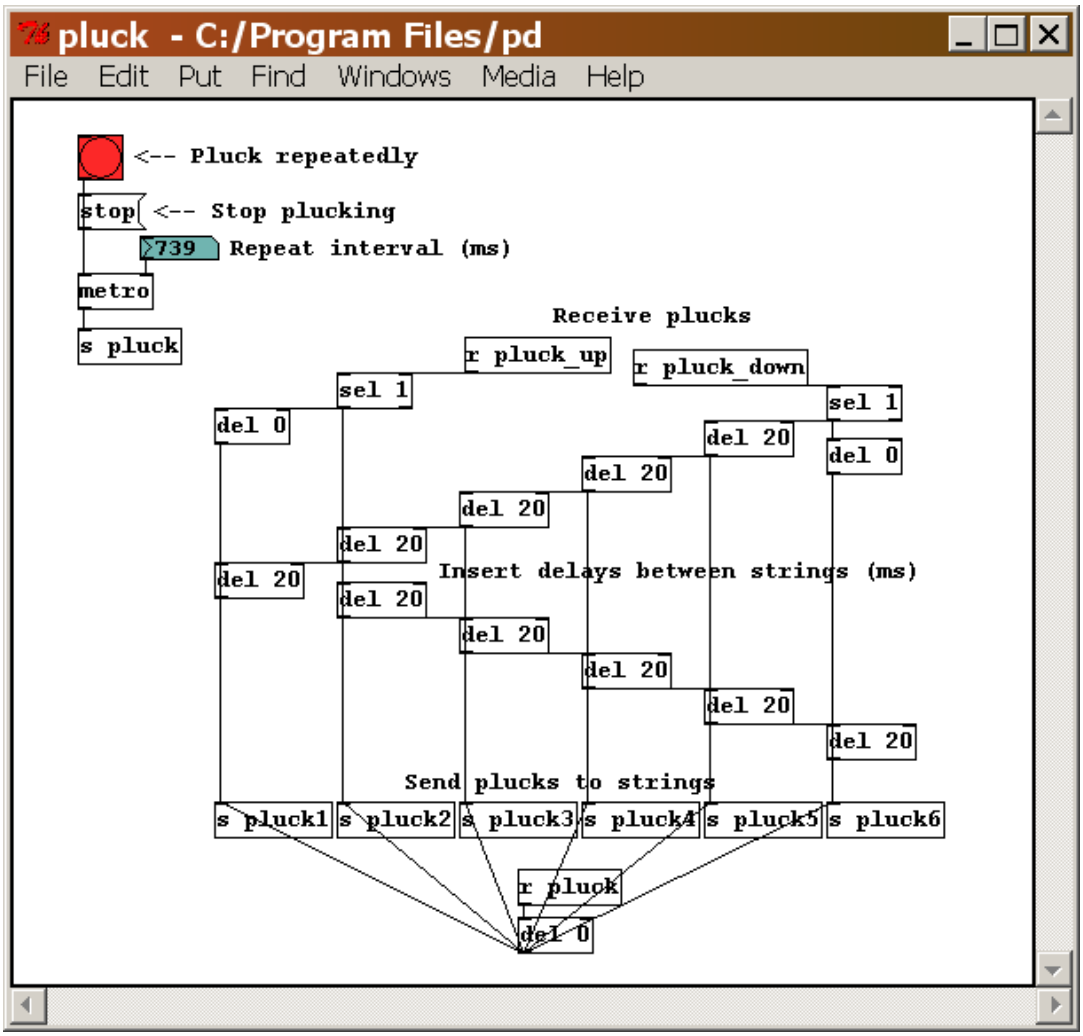

Figure B.2: The plucking sub-patch. Pluck triggers are received and sent to strings with 20 ms delay between adjacent strings. An automatic plucking option with variable repeat interval for testing purposes can also be used from here.

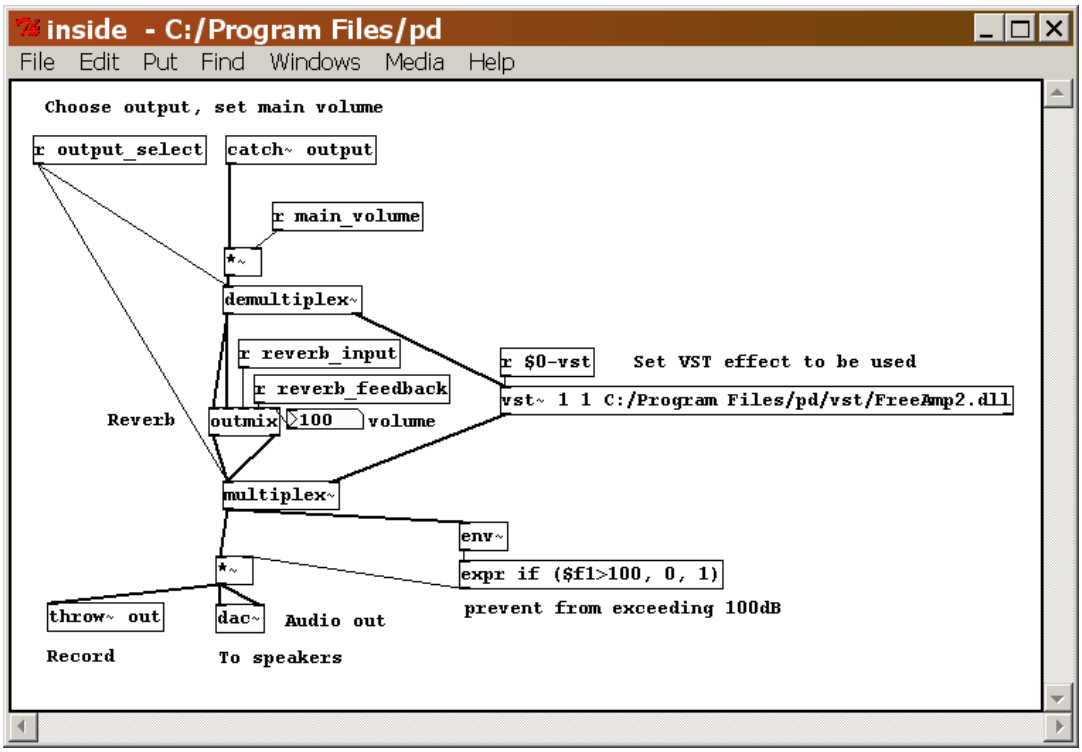

Figure B.3: This is the output patch, which selects the output effects. Selectable effects are PD reverb effect and VST FreeAmp2. The patch also protects from clipping and outputs the final signal to the soundcard and the recorder.

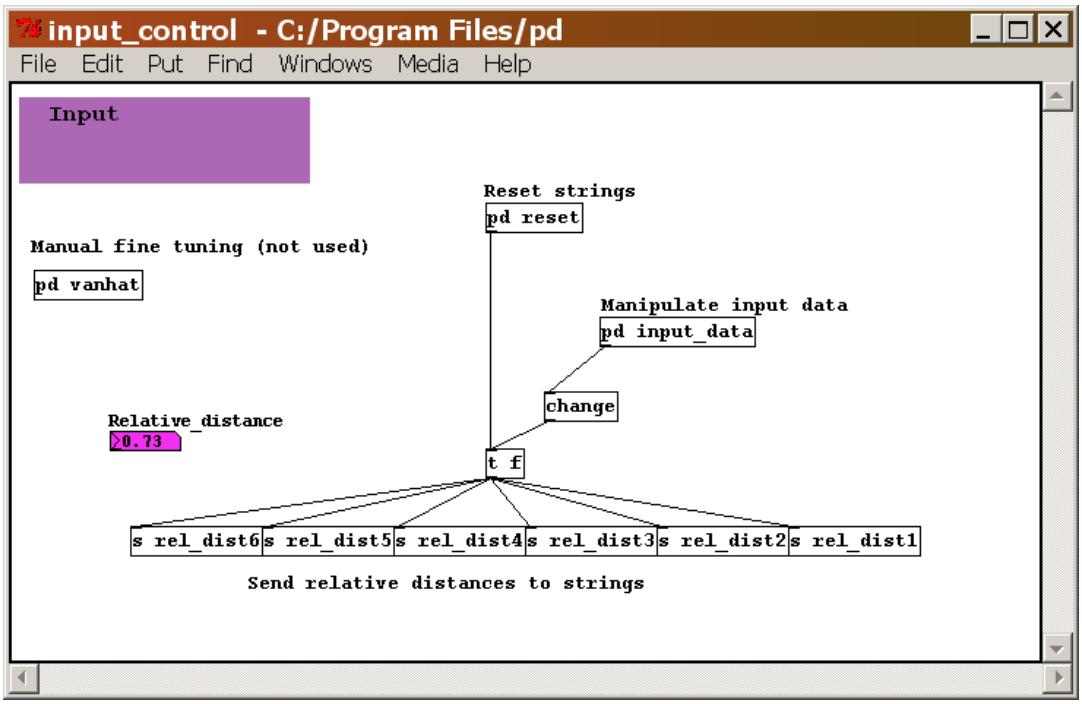

Figure B.4: The input control patch. This patch shows the relative distance and routes input data. The reset sub-patch initializes strings with maximum length.

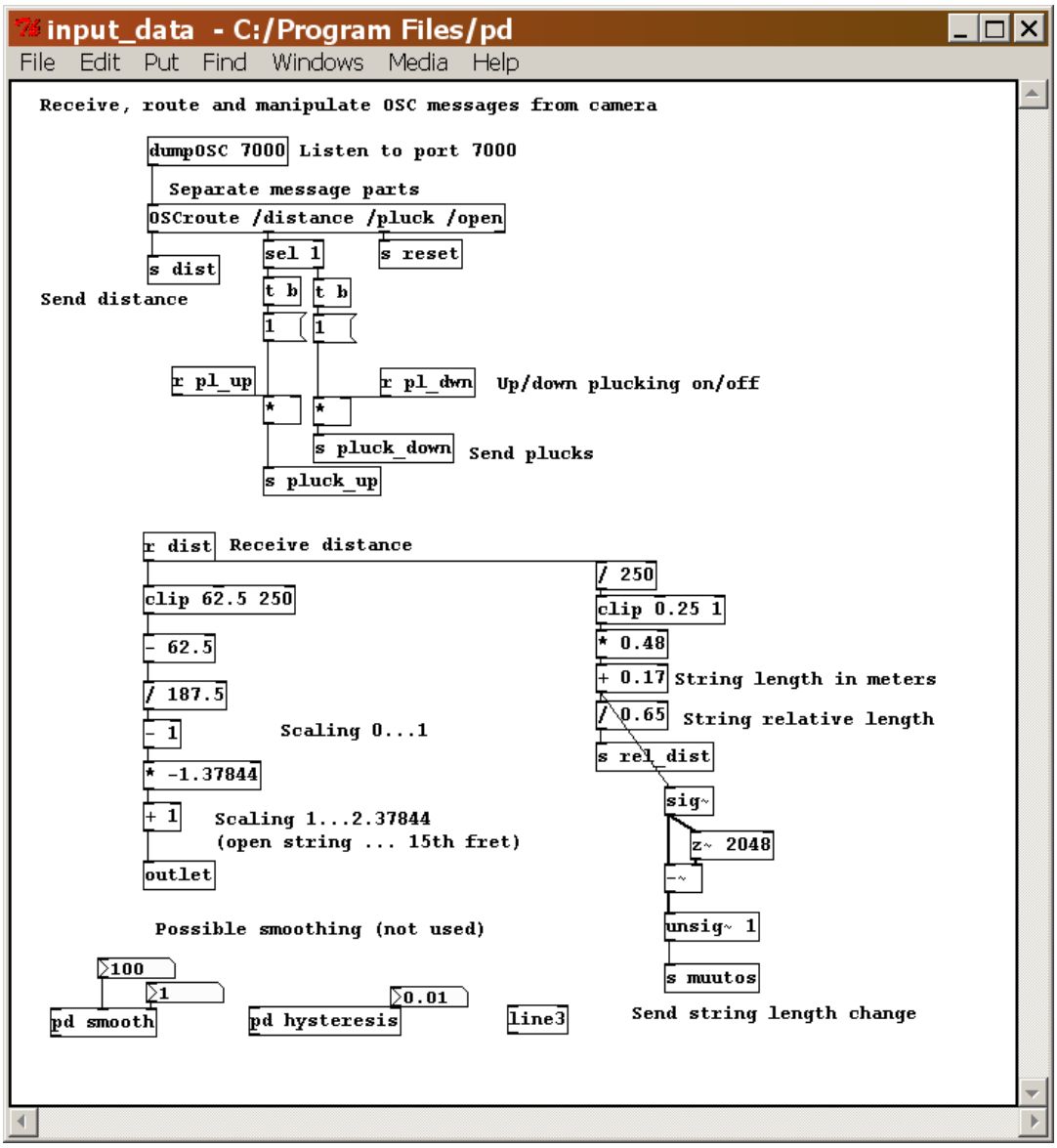

Figure B.5: The input data patch. OSC messages from the camera API are received and treated here. Plucking events are sent according to OSC messages. On the lower part of the patch there are some conversions to convert length in pixels to meters and relative length.

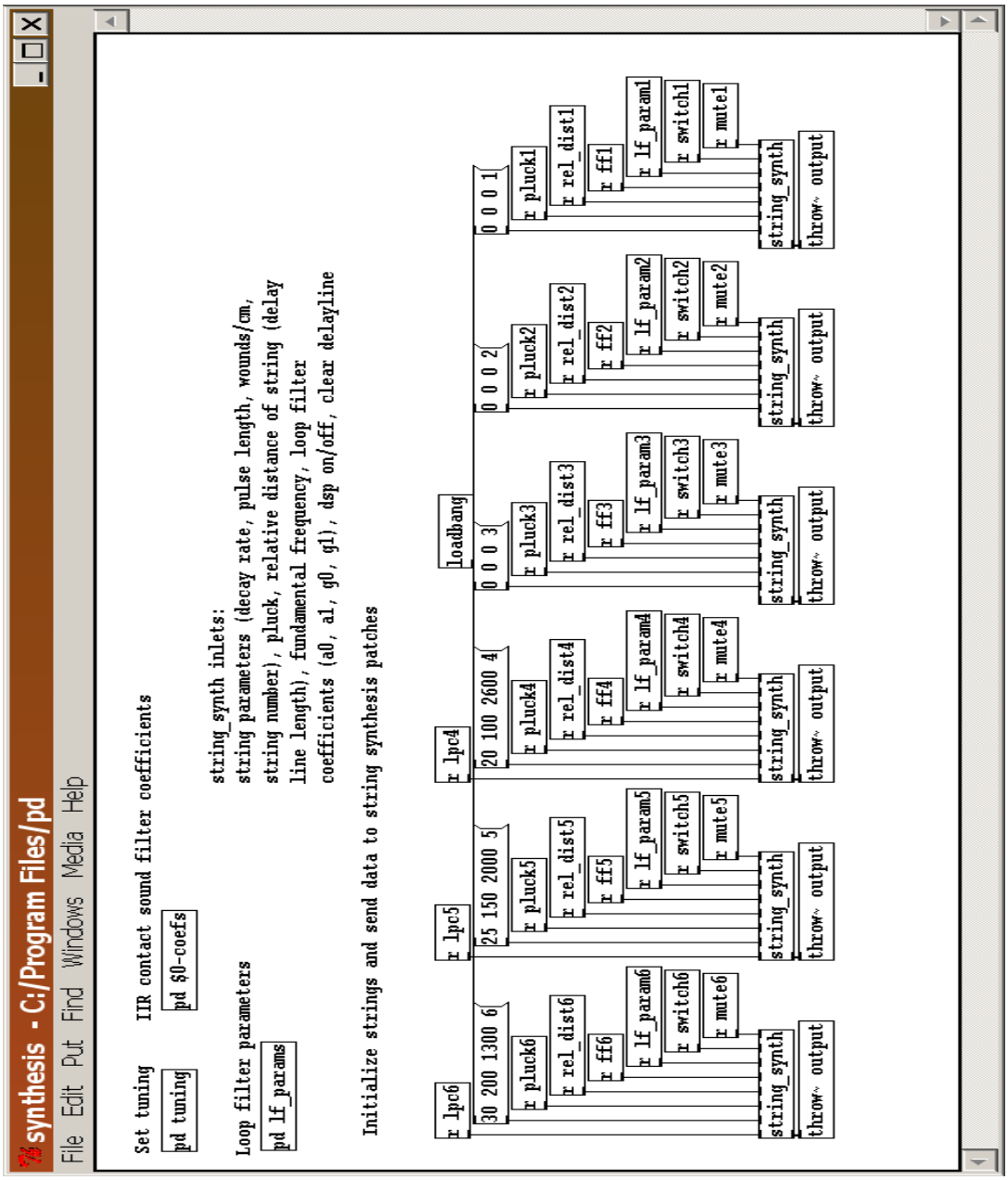

Figure B.6: This is the synthesis ‘mother’ patch. It initializes the strings and routes data to string synthesis patches.

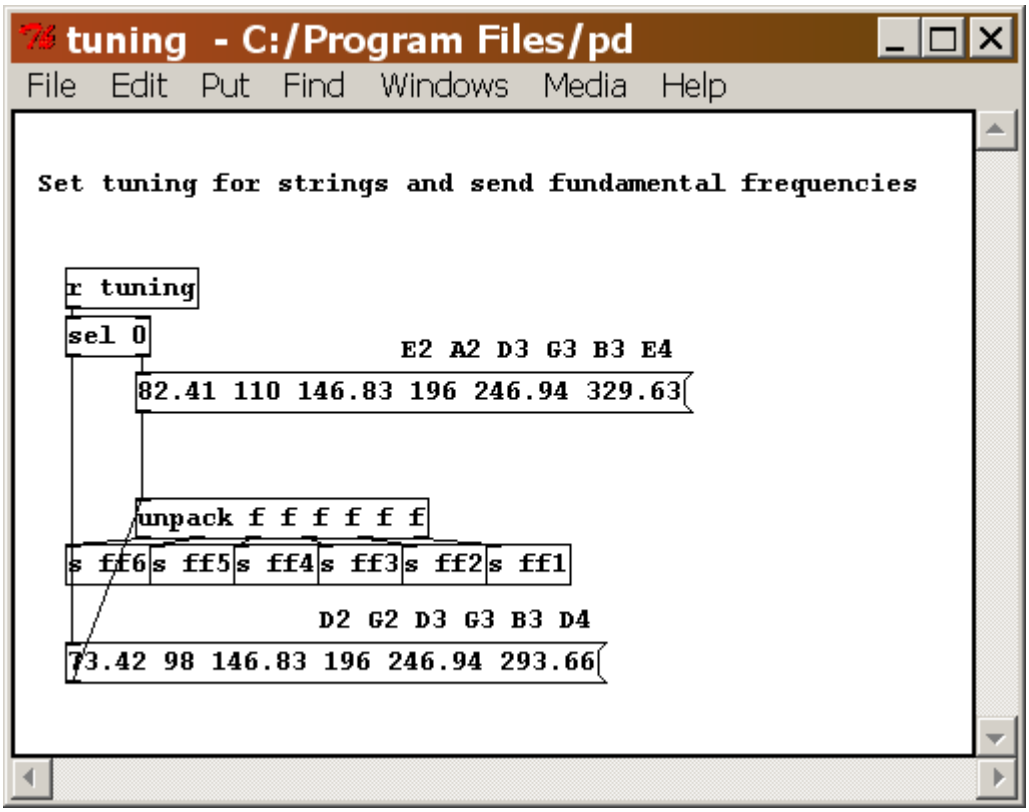

Figure B.7: The tuning patch sets the tuning of the strings according to what the player has selected from the user interface. More tunings could be implemented here.

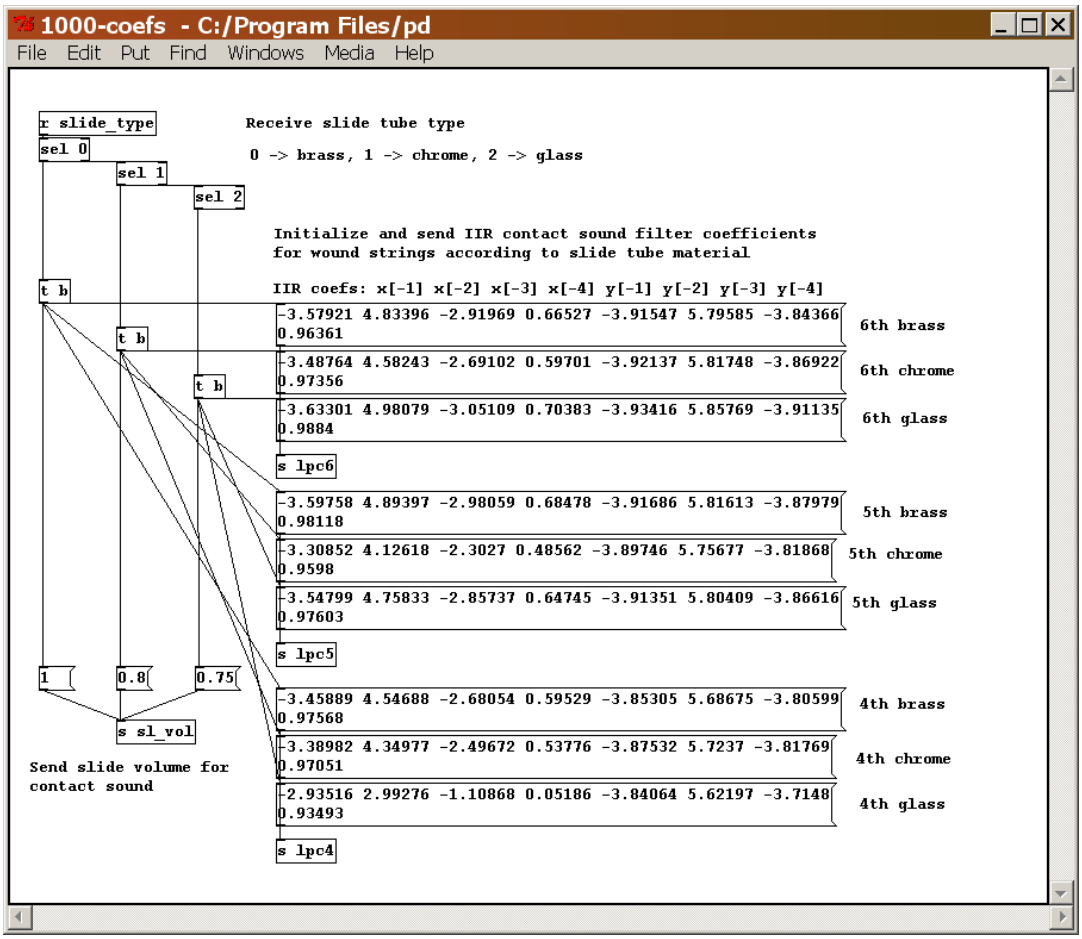

Figure B.8: This patch initializes the Fourth-Order IIR Contact Sound Filter coefficients for wound strings according to slide tube material.

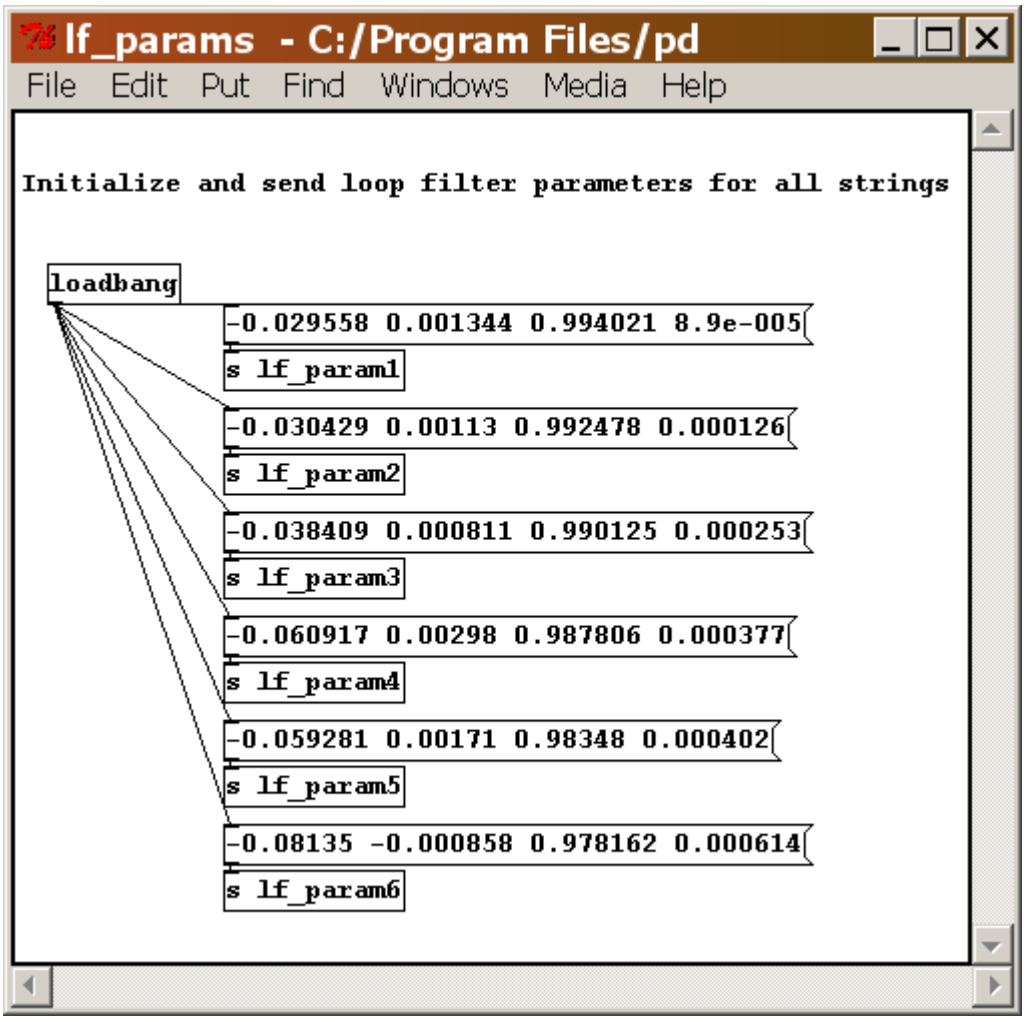

Figure B.9: This patch initializes and sends loop filter parameters for all strings.

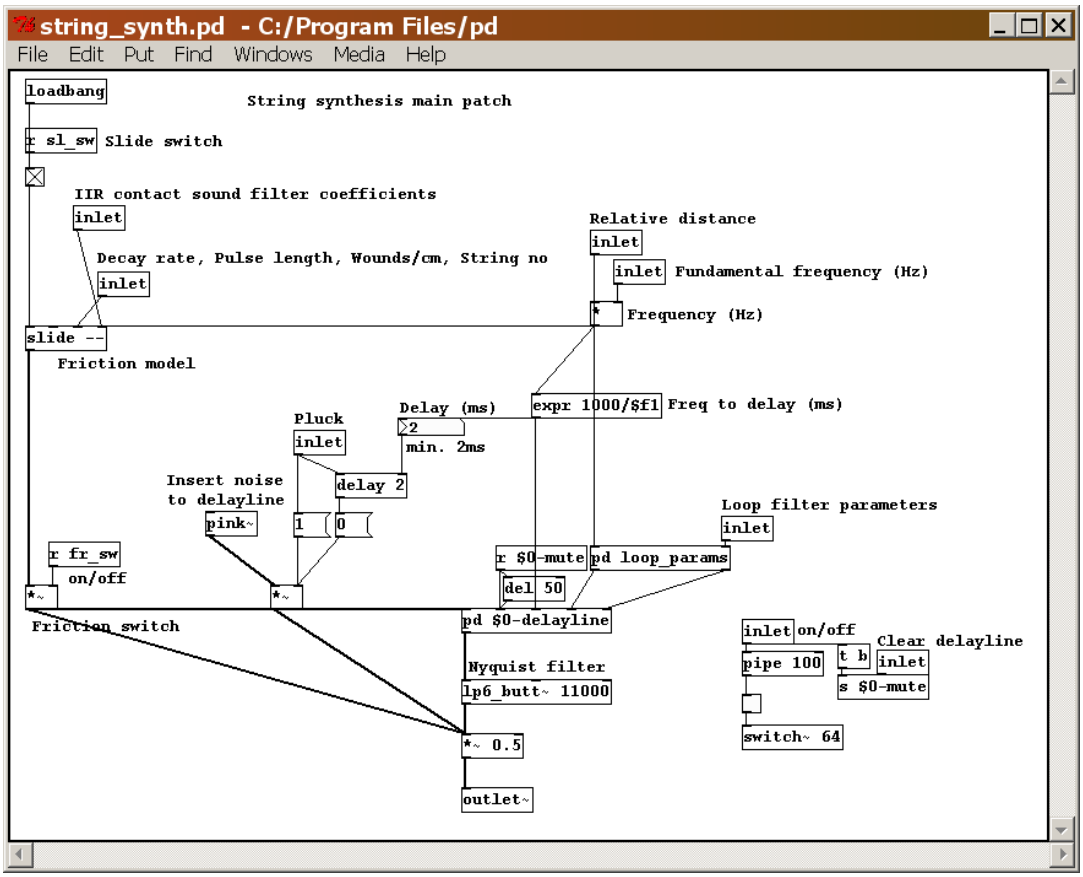

Figure B.10: The string synthesis patch. This is the main string synthesis patch. It holds the contact sound synthesis patch, the delay line and pluck excitation. String properties, filter parameters and distance/frequency/delay data is routed here. An anti-aliasing filter is implemented with a low-pass 6th order filter with Butterworth filter characteristics.

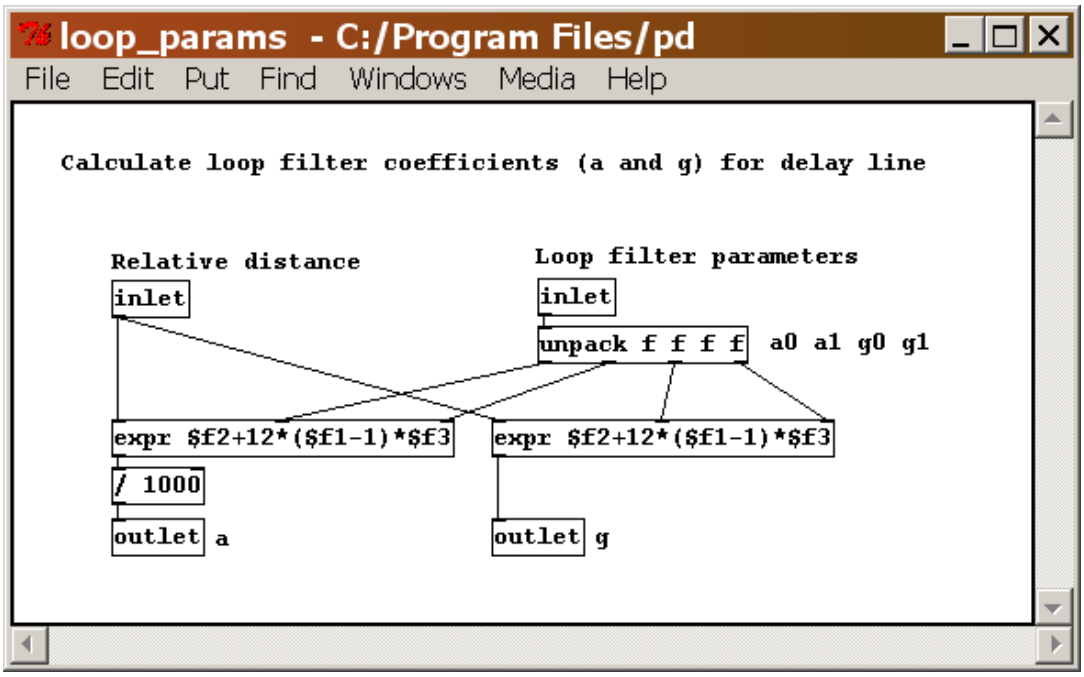

Figure B.11: Loop filter coefficients *a* and *g* are calculated here with given loop filter parameters.

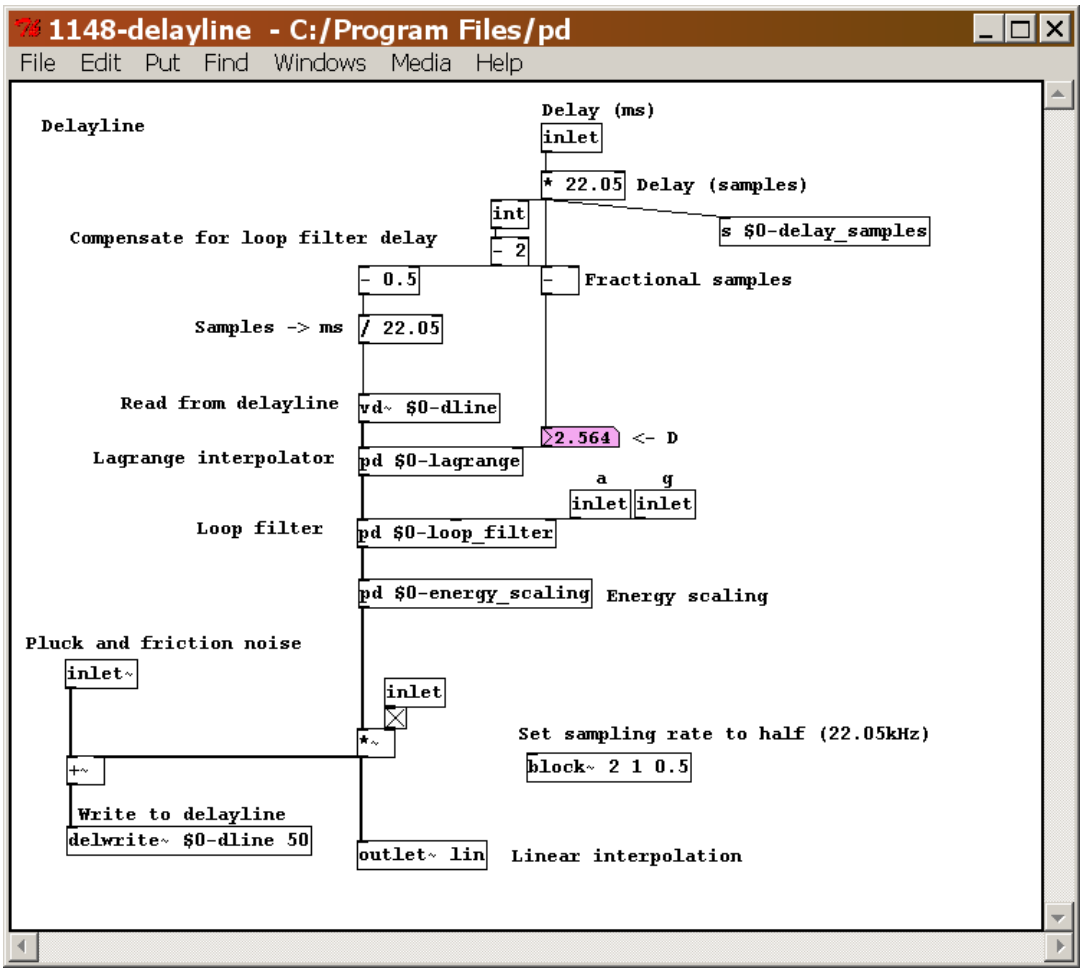

Figure B.12: This is the delay line. It consists of the Lagrange interpolator, loop filter and energy scaling. Delay is also converted between seconds and samples for the filters and delay line.

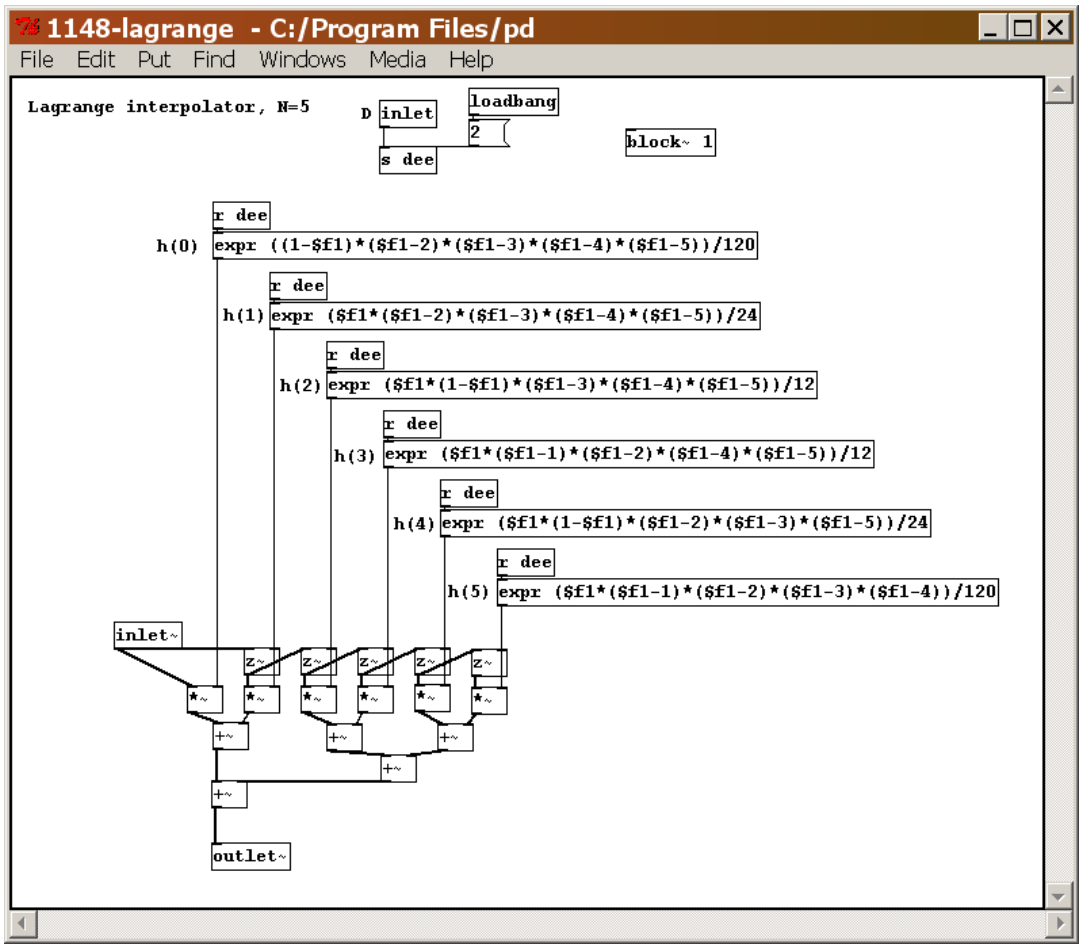

Figure B.13: The 5th order Lagrange interpolator. The filter receives the variable delay value D and calculates the filter coefficients. The block size is set to 1 which means that the filter calculates every sample separately.

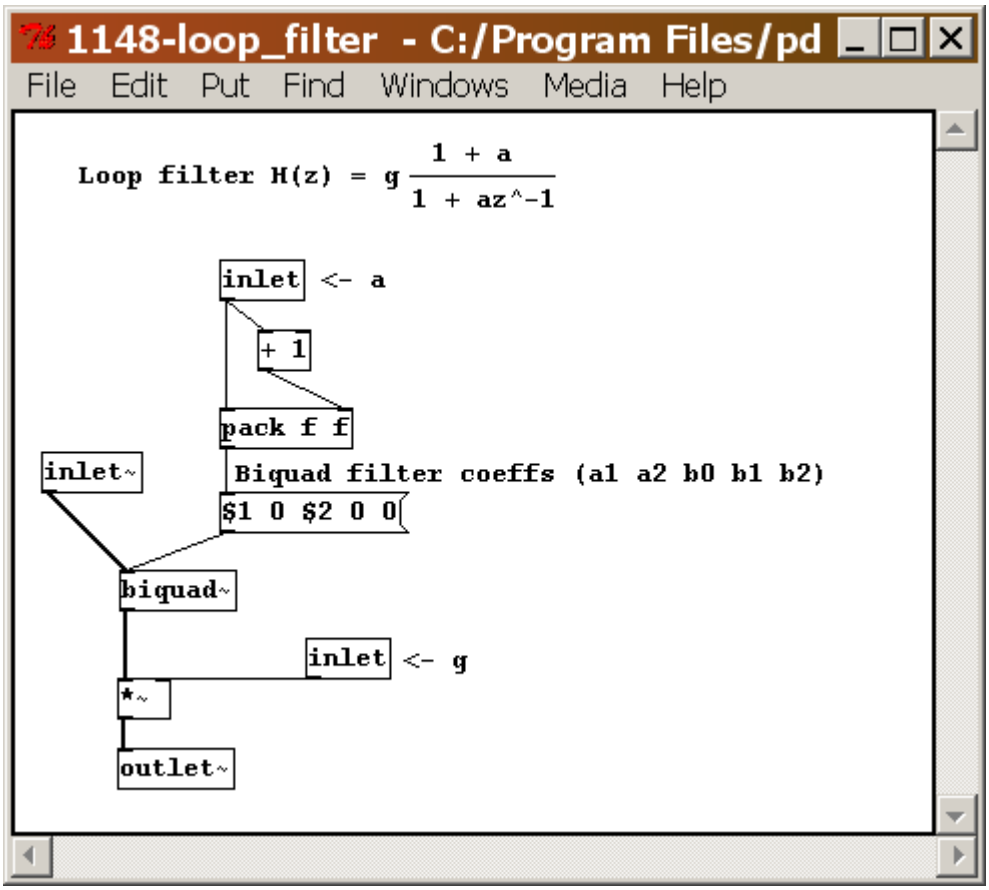

Figure B.14: This is the loop filter implemented with a 2 pole 2 zero filter.

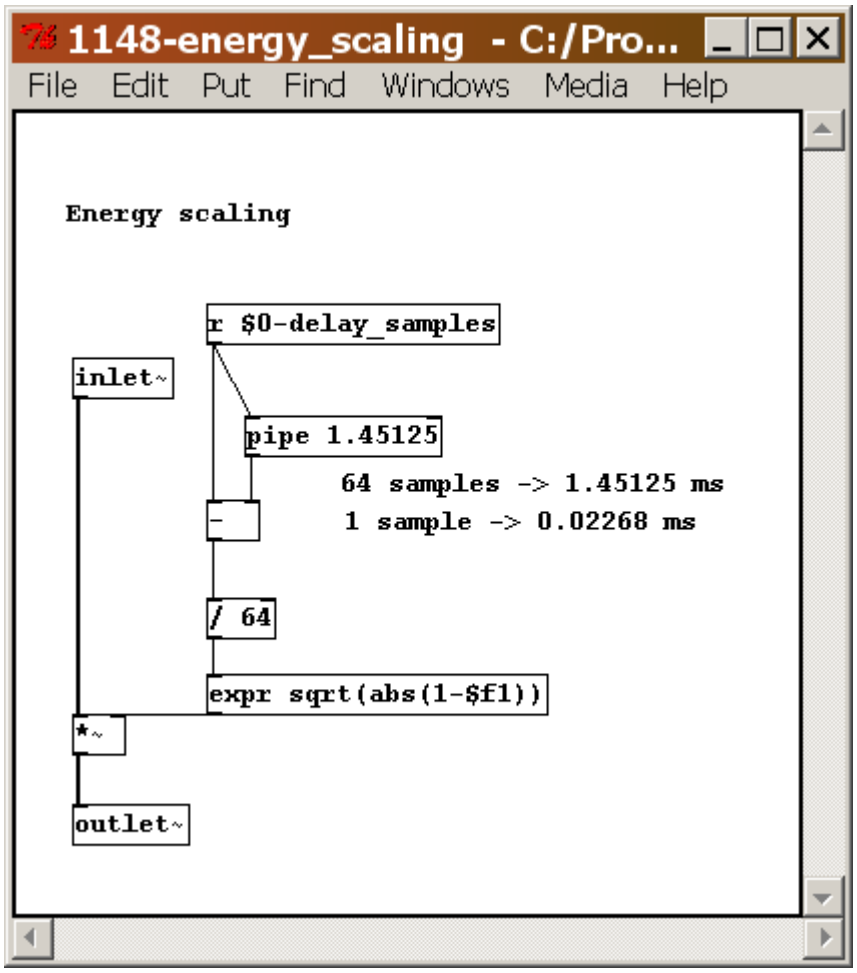

Figure B.15: The energy scaling function is implemented in this patch.

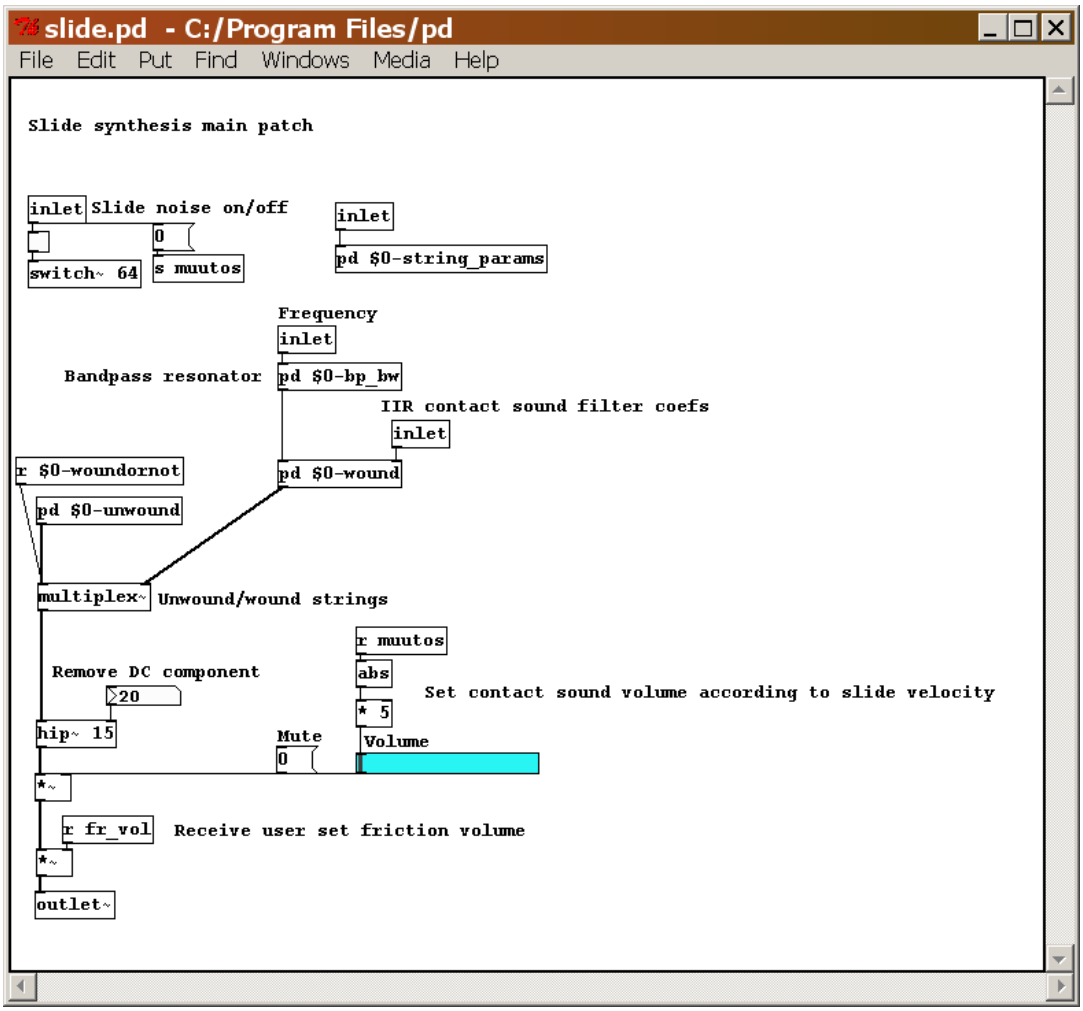

Figure B.16: The main contact sound synthesis patch. This patch holds the contact sound generators for wound and unwound strings, as well as the bandpass resonator and contact sound volume controlling according to slide velocity. A DC component is also filtered out with a high-pass filter from the contact sound output.

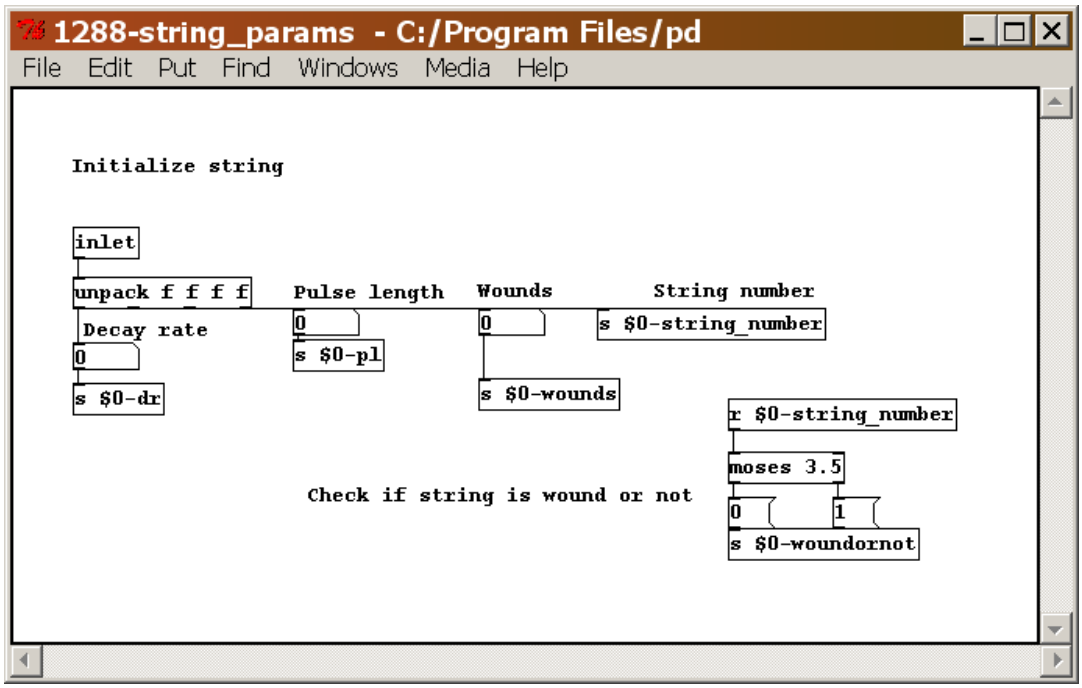

Figure B.17: This sub-patch receives string parameters, unpacks them and sends them to the contact sound generator.

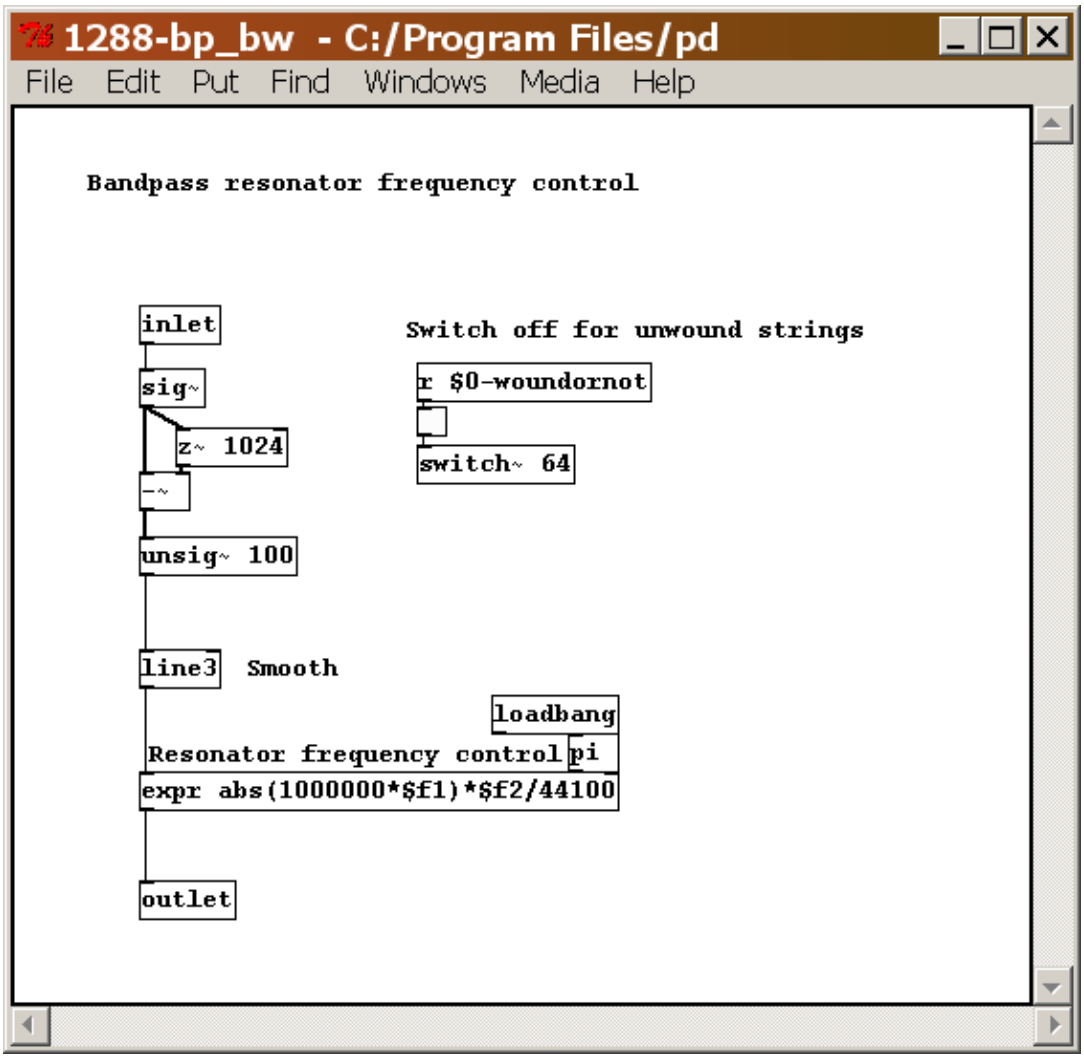

Figure B.18: This is the bandpass resonator frequency control sub-patch.

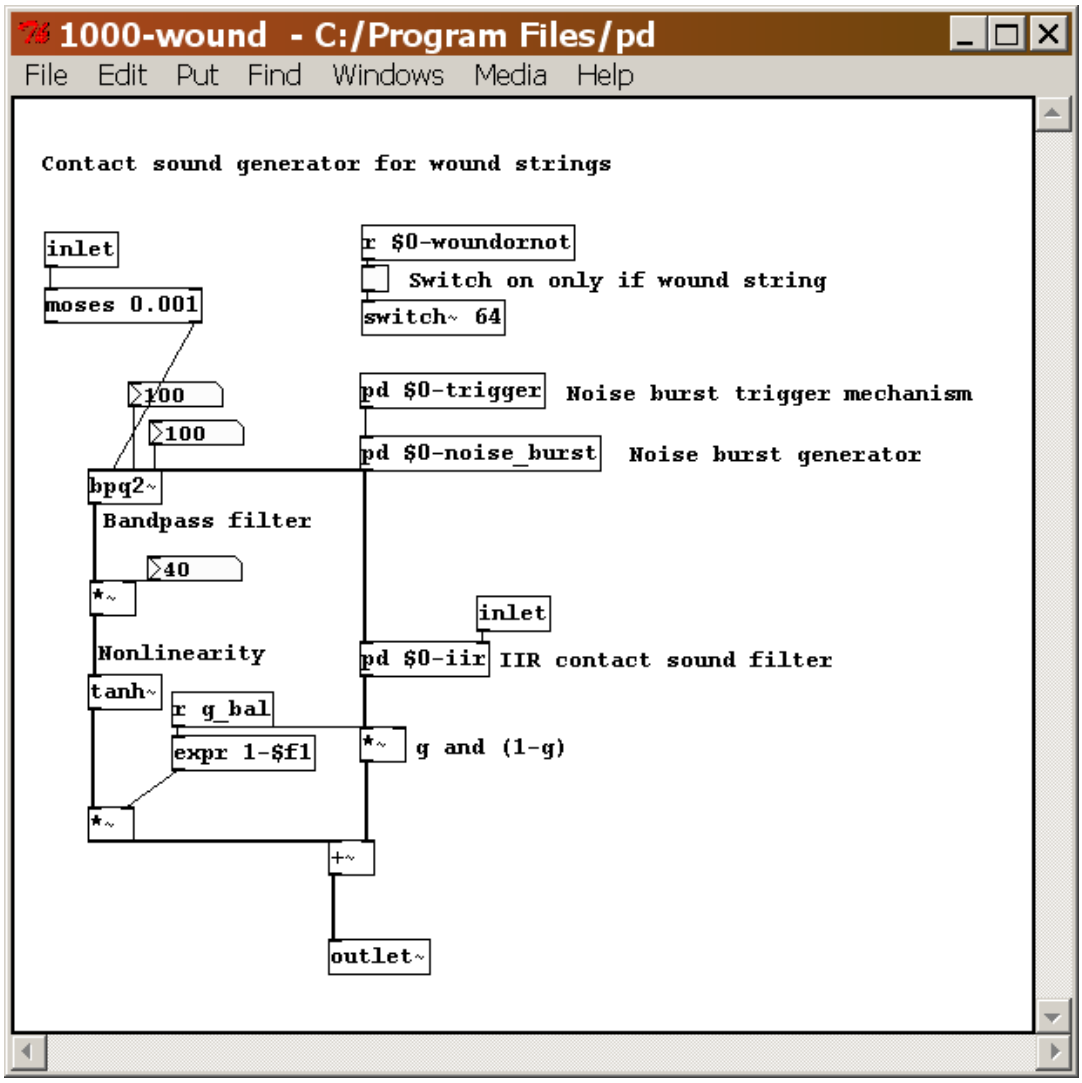

Figure B.19: The contact sound for wound strings is generated inside this patch. It features the bandpass filter, a scaled hyperbolic tangent function, noise trigger mechanism, noise burst generator, the IIR contact sound filter and balancing of static and dynamic contact sounds.

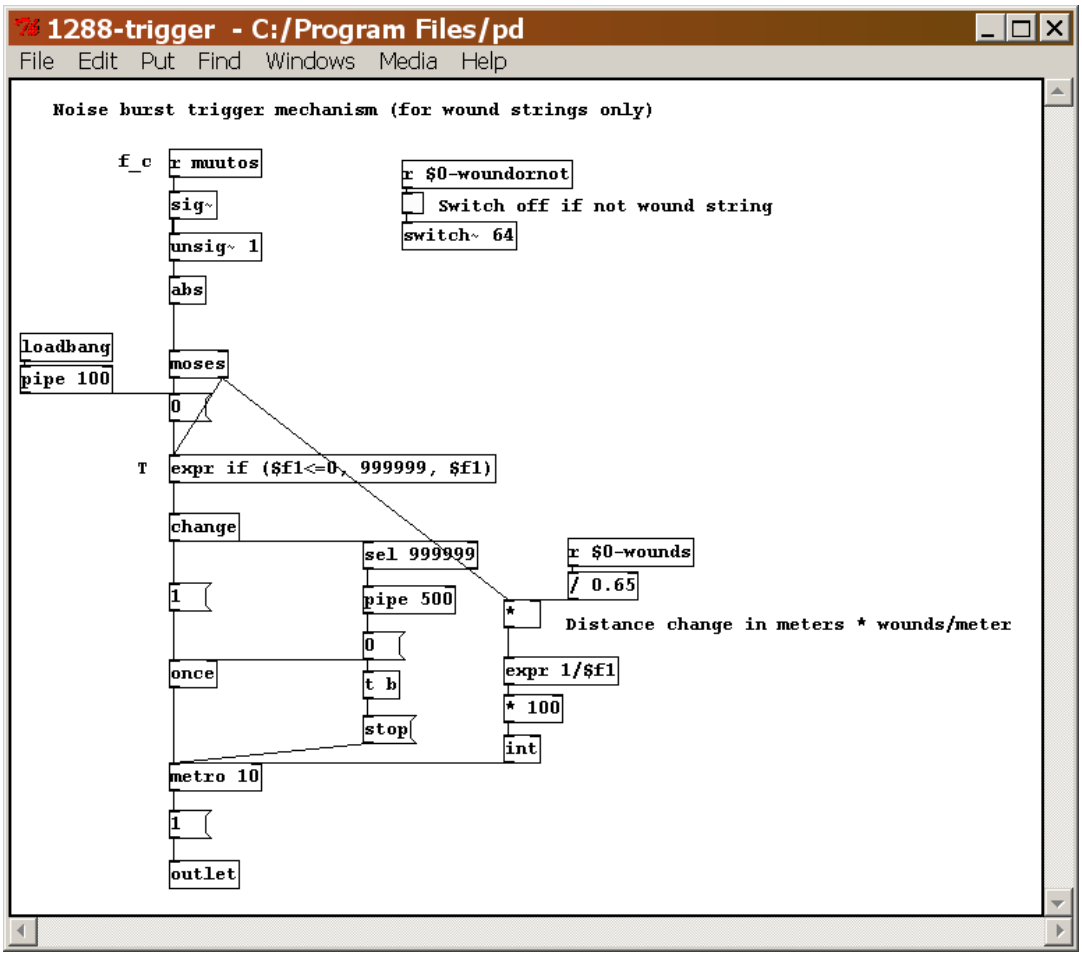

Figure B.20: This patch triggers the noise bursts. Triggering speed varies with string properties and slide velocity.

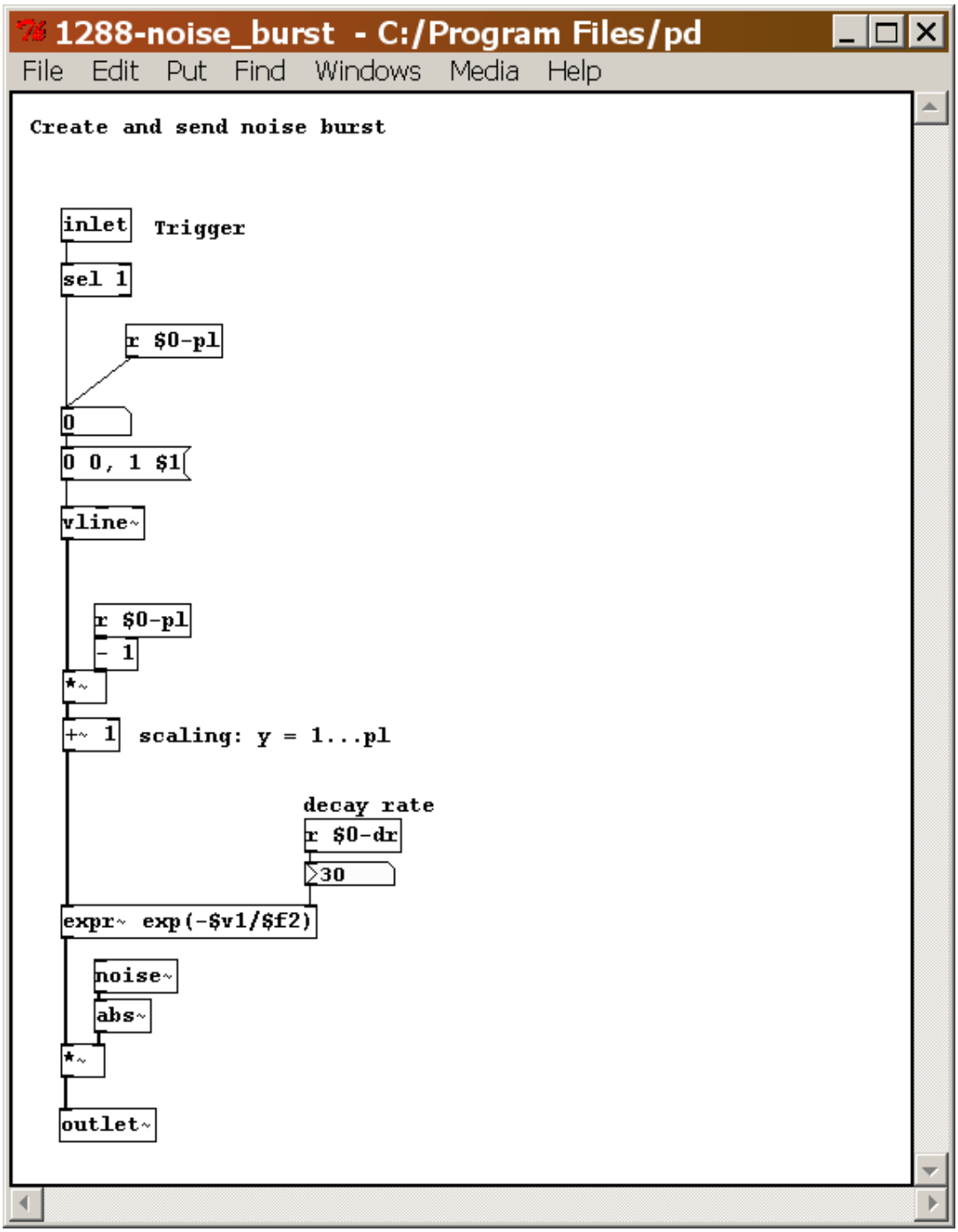

Figure B.21: The noise burst generator patch receives noise burst triggers and outputs the noise pulse train according to triggering and string properties.

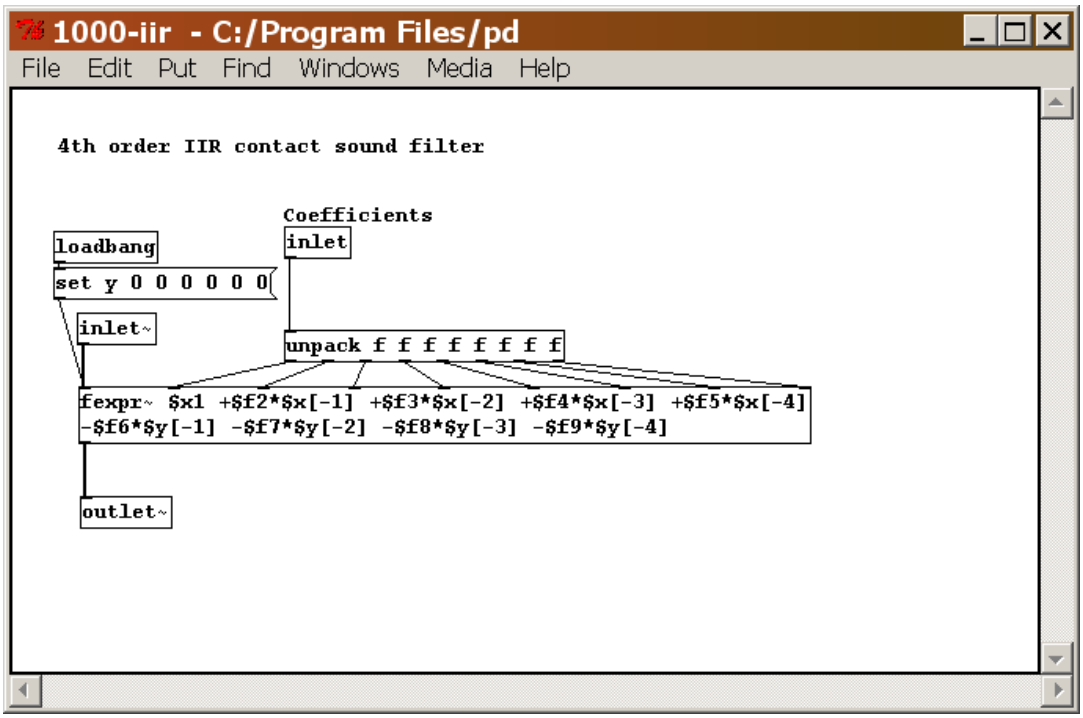

Figure B.22: This is the Fourth-Order IIR Contact Sound Filter. The IIR filter is implemented with fexpr~ function.

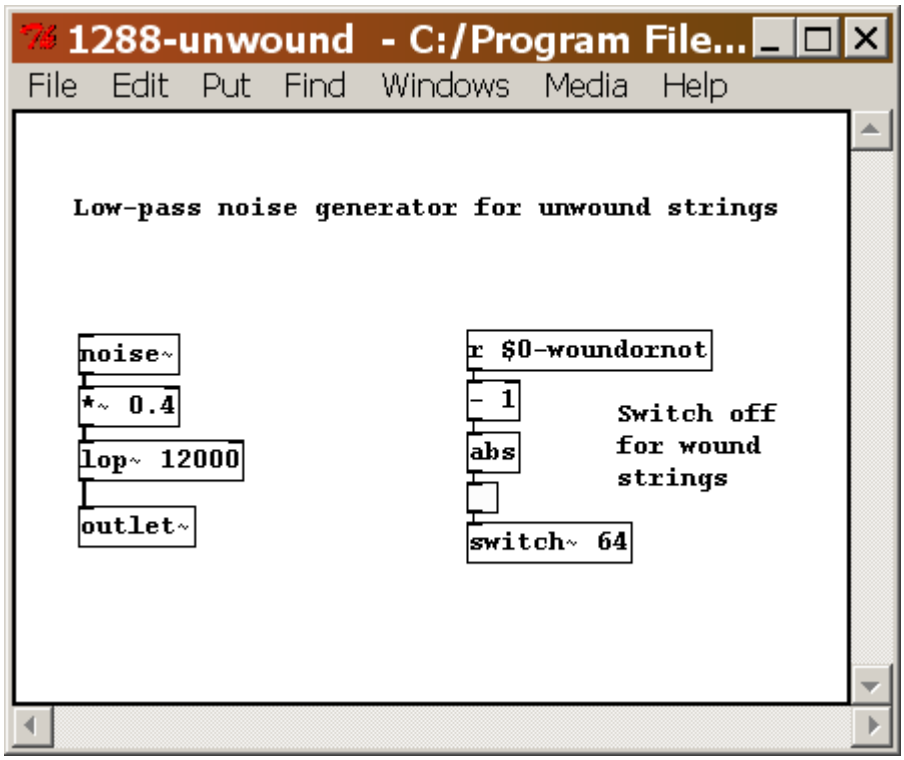

Figure B.23: The noise generator for unwound strings. As can be seen, it is quite simple compared to wound strings. The output is low-passed white noise.

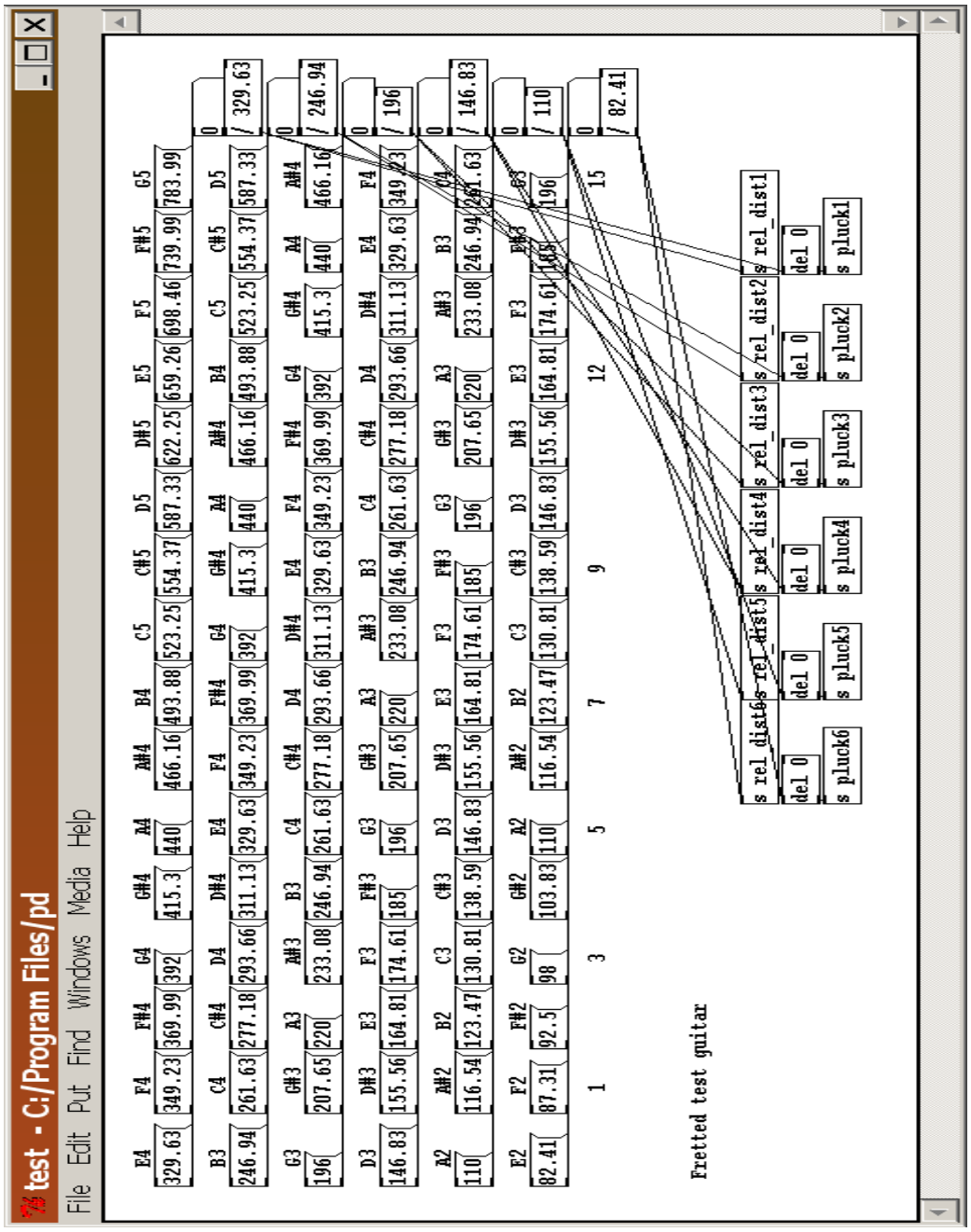

Figure B.24: The fretted test guitar. Originally for testing purposes, individual strings and tones can be played from here as a fretted guitar with normal tuning.

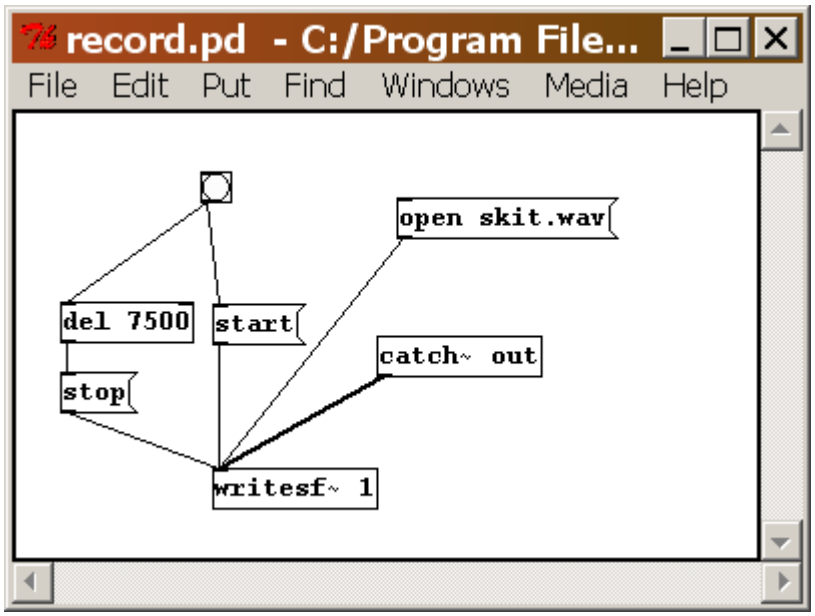

Figure B.25: A patch for recording the main output. By default this patch records 7.5 seconds and writes the recording to a file called skit.wav as a 44.1 kHz mono wave file. The recording length and filename can be edited by the user.