UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

"SIMULAZIONE SONORA TRAMITE CAMPIONAMENTO E POST-PROCESSING DI UN ANTICO SONAGLIO DEL PERIODO ETRUSCO"

Relatore: Prof. / Dott Antonio Roda'

Laureando/a: Antonio Busetto

ANNO ACCADEMICO 2023 – 2024

Data di laurea: 15/07/2024

Indice

Sommario

In seguito al ritrovamento di un sonaglio etrusco risalente al IV-III secolo a.C., il progetto di tesi si prefigge di preservare il valore culturale dello strumento nel tempo. In seguito a delle registrazioni dello strumento, si procede a creare una simulazione sonora dell'ultimo mediante campionamento e post-processing, collegandoli ai movimenti di un sensore con protocollo osc, con l'aiuto di software come Pure Data. Con i file audio così elaborati si andrà in futuro a ricreare una copia del sonaglio dal valore rappresentativo che permetta di utilizzare e toccare con mano in modo interattivo lo strumento senza rovinarlo e permettendo una corretta conservazione dello stesso.

Capitolo 1

Introduzione

Questa tesi si prefigge come obiettivo finale la creazione di un prototipo atto alla simulazione sonora di un sonaglio di probabili origini ellenistiche, ritrovato in un sito archeologico nei pressi di Adria, Rovigo. Quante volte, all'entrare in un museo o a una mostra, ci siamo chiesti come potesse essere effettivamente toccare con mano quello che stavamo guardando? Poter suonare rudimentali strumenti musicali, indossare pesanti pellicce o brandire antichi pugnali e lance? Come ben sappiamo, questo non è possibile, i musei infatti devono garantire standard di conservazione molto alti per poter trasmettere alle generazioni future il bagaglio culturale che ci viene donato dai reperti. Da qui l'idea di creare una simulazione di un effettivo reperto archeologico, in questo caso di un sonaglio in argilla, per poter permettere alle persone di usarlo e sentire effettivamente le sue peculiarità, senza rischiare di comprometterne l'integrità. Dispositivi di questo tipo se ne trovano già molti in mostre e musei di tutto il mondo; dai sintetizzatori ai clavicembali, permettono di avere esperienze più immersive e interattive nel mezzo della propria visita.

Per questa tesi, il lavoro complessivo si può suddividere in 3 momenti chiave:

1. Registrazione e post-processing dei campioni audio del sonaglio in questione, mediante tecniche di microfonazione classiche e software specifici per il restauro e enhancing delle tracce.

2. Selezione dalla totalità delle riprese audio di un gruppo limitato di campioni che

riassumano ma rappresentino al meglio le peculiarità acustiche del sonaglio, ed in seguito scelta di un software per la trasmissione di dati dei sensori di movimento da smartphone tramite protocollo OSC. Il software è stato scelto in base alla quantità di dati distinti fruibili, i formati in cui vengono inviati e l'affidabilità/versatilità delle connessioni di rete.

3. Implementazione del codice che permetta di tradurre i dati dei sensori di movimento con suoni definiti campionati dalle registrazioni audio del sonaglio, simulando il suono dello stesso collegato al movimento dello smartphone.

Tra le varie metodologie possibili di svolgimento del progetto, vedremo come quella appena descritta sia quella che meglio si sposa con le premesse del progetto, le approssimazioni raggiunte nella raccolta dei dati e il modello di simulazione conclusivo. Vedremo nel dettaglio le tipologie di software impiegate per lo sviluppo del progetto, improntate per l'appunto a un approccio più pratico del problema rispetto ai classici linguaggi di programmazione, e quanto fedelmente sarà possibile riprodurre i movimenti e suoni di uno strumento di questo tipo.

Capitolo 2

Oggetto di ricerca

2.1 Il sonaglio

Figura 2.1: Immagine riportata nel Catalogo Beni Culturali della regione Veneto

Il sonaglio è uno strumento musicale a percussione che suona quando viene scosso. I primi sonagli vengono datati all'incirca tra il 1000 e il 500 a.C., tempi in cui venivano principalmente usati per rituali funebri (nell'Antico Egitto erano considerati sacri, venivano impiegati anche dai nativi americani per danze cerimoniali). Secondo la classificazione Hornbostel-Sachs (sistema di classificazione degli strumenti musicali maggiormente impie-

gato universalmente), "Gli Idiofoni scossi sono sonagli (da non confondere con i clapper). Il materiale è importante, ma ancor più importante è la disposizione delle parti suonanti che si colpiscono tra loro quando l'attrezzo viene scosso."[1]

Il suono di un sonaglio, infatti, dipende dal materiale di cui è fatto l'involucro (ceramica, legno, pietra), da quello di cui sono fatte la/le pallina/e (metallo, legno, pietra, ceramica, vetro) e dai pattern con cui le parti interne semoventi vengono a scontrarsi durante il movimento.

Il sonaglio oggetto dei nostri studi (vedi Figura 2.1), datato indicativamente tra il IV e il III secolo a.C., è stato rinvenuto nel Sito archeologico della Necropoli di Ca' Cima, sito che si trova nel comune di Adria, RO. Gli scavi in cui è stato rinvenuto sono avvenuti tra il Dicembre del 1994 e il Giugno del 1995, e ad oggi si trova nel Museo archeologico nazionale di Adria. Il sito della Regione Veneto nel Catalogo dei Beni Culturali riporta queste informazioni: "Sonaglio in ceramica alto adriatica di forma globulare, schiacciata ai poli, cavo all'interno e contenente piccoli elementi sonanti. Decorazione dipinta con vernice nero-rossastra diluita: linea continua nel punto di massima espansione del corpo e linee a raggiera sulle due superfici; su un lato, 12 linee di colore rossastro partono dal vertice sommitale per scendere a raggiera sulla linea continua, sull'altro lato i raggi, di colore nerastro, sono 6 o 7 e partono dall'altro vertice per terminare sempre sulla linea continua. Sparse sulla superficie sono delle lineette trasversali, a rotellatura, irregolari e sporadiche. L'argilla è di colore camoscio."[2] Il chiaro valore storico di un reperto di questo tipo rende chiaramente difficile il fruirne liberamente a scopo di ricerca, motivo per cui il tempo dedicato alla fase di registrazione dei suoni dello stesso, e più in generale al suo studio, è stato limitato.

Contesto storico La città antica di Adria dove è stato rinvenuto il reperto sorgeva nella parte meridionale di quella odierna (dove ora è situato il Museo Archeologico), sulle sponde del Po. Visto il contesto idrogeologico instabile, lo sforzo logistico della popolazione per la creazione di canali di scolo e per le bonifiche del terreno con strutture lignee era non di poco conto. La zona fin dal VI secolo a.C. si trovava al confine dei territori di veneti

ed etruschi, portando a un vivace interscambio culturale tra i due popoli. A queste due si aggiunse poco dopo l'influenza greca, dovuta all'arrivo di mercanti in cerca di cereali, metalli, ambra e altri prodotti di provenienza settentrionale da scambiare con le loro pregiate ceramiche.

L'incrocio di queste tre culture porta ad un'evoluzione dei prodotti locali di artigianato, che venivano utilizzati perlopiù nella vita quotidiana, mentre quelli importati dai mercanti greci erano più pregiati e ricercati, usati per il vino o come ornamenti nei corredi trovati in grande quantità nelle catacombe della zona. Il sonaglio in oggetto, per esempio, eredita senza dubbio le decorazioni nero-rossastre dalla tradizione delle ceramiche attiche importate dai greci; per quanto riguarda il materiale, la ceramica era già ampiamente impiegata dagli etruschi nelle produzioni artigianali, anche se principalmente per utensili da cucina o di uso comune.

Capitolo 3

Strumenti software

Qui di seguito vengono riportati i vari software utilizzati nella totalità di questo progetto. Nel successivo capitolo andremo a spiegare nel dettaglio in che modo e per cosa sono stati usati, ma questa sezione si reputa necessaria per avere un quadro completo ed esaustivo di quello che è il framework del lavoro, e per dare la possibilità a chi non conoscesse alcuni di questi strumenti di comprendere il loro funzionamento e la loro utilità.

3.1 Reaper

Reaper[1] è un software multipiattaforma che funge da DAW (Digital Audio Workstation) e da sequencer MIDI. Realizzato da Cockos nel 2006, a differenza dei suoi blasonati concorrenti permette, dopo la canonica prova di 60 giorni, di proseguire con l'utilizzo, a patto che non si usi il software a scopo di lucro, e invitando comunque gli utenti soddisfatti ad acquistare una licenza. E' la prima scelta per molti addetti ai lavori per via della sua versatilità e affidabilità, è infatti un software molto leggero ma nonostante questo completo, in grado di sfruttare tutti i formati di plugin esterni. Cosa del resto per nulla necessaria, in quanto Reaper possiede una serie di plugin nativi (i ReaPlugins, by Cockos), che nulla hanno da invidiare da quelli più noti, se non per un'estetica un po' minimale. Altro punto

[1] `https://www.reaper.fm/`

a favore è l'abbondante documentazione online, con una numerosa community di utenti che sostiene i neoarrivati nell'apprendere al meglio il funzionamento del software. Inoltre, Reaper viene impiegato molto anche per l'editing video, infatti nonostante l'interfaccia poco "accattivante" rispetto a quella di altri video editor, Reaper implementa tutte le funzionalità necessarie alla manipolazione di video.

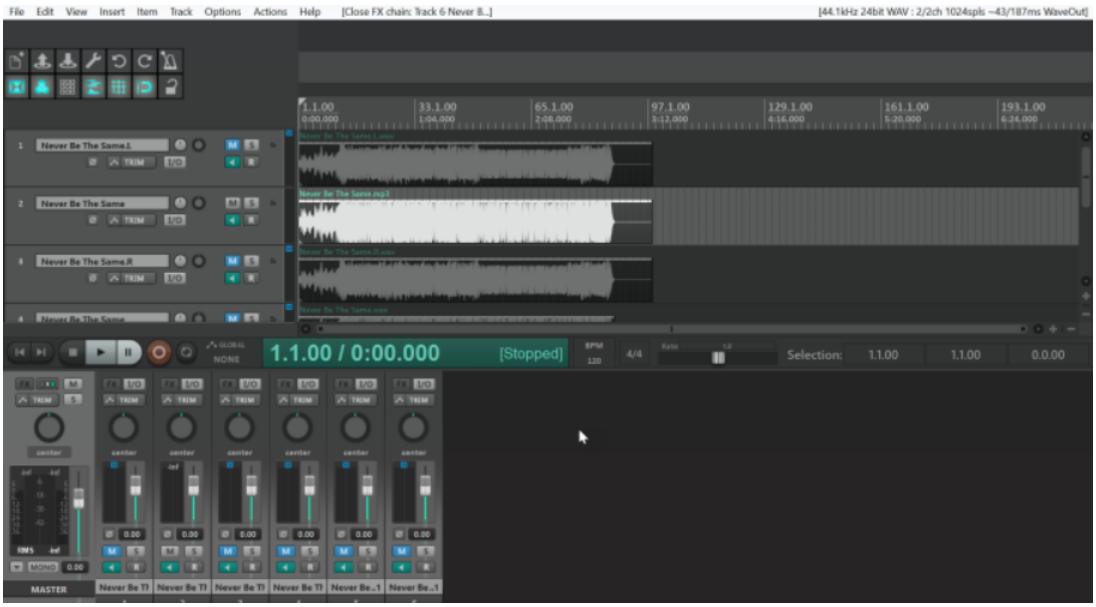


Figura 3.1: Uno snapshot della gui di Reaper

3.2 Fabfilter ProQ

Fabfilter ProQ[2] è un software plugin di equalizzazione di alta qualità, usato da buona parte dei professionisti del settore. Garantisce un eq grafico con fino a 24 bande, una buona ottimizzazione della CPU e un ottimo spectrum analyzer. In aggiunta ha numerose funzioni extra per equalizzazioni più fini, dal deesser interno all'eq dinamica (con funzionamento simile alla compressione parallela[3]), fino alla possibilità di rallentare la visione dello spettro delle frequenze così da poter osservare le curve delle frequenze che stanno suonando in real-time.

[2] `https://www.fabfilter.com/products/pro-q-3-equalizer-plug-in`

[3] Tecnica di compressione che consiste nel duplicare il canale in questione, applicare il compressore a uno dei due e sommarli poi nel mix

Resta famoso in quanto tool fondamentale per poter eseguire un'equalizzazione chirurgica sulle tracce audio più ostiche. Nel nostro caso, dopo una prima analisi con equalizzatori più classici e meno aggressivi, ho compreso che era necessario uno strumento più preciso. Avendo un segnale audio debole e costellato di hum e rumore di fondo, con un'equalizzazione precisa e pulita riusciamo a preservare la "parte buona" di segnale, riducendo i disturbi indesiderati.

Figura 3.2: Fabfilter ProQ 3

3.3 Izotope RX

Izotope RX[4] è il miglior software plugin per restauro e riparazione audio. Utilizzato anche nell'industria cinematografica e nei restauri conservativi oltre che nel campo dell'audio, più che un plugin è una suite di plugin, ognuno pensato per risolvere un problema diverso di rumori di fondo o interferenze. Tra i più utilizzati ci sono Dewind, Declick, Denoise, ecc. La grande novità rispetto a plugin di restauro di altri marchi è il Repair Assistant, un AI tool che sfrutta in parallelo i vari plugin di RX per ottenere il miglior risultato sulla traccia. I plugin sono tutti infatti basati su tecnologie di machine learning, che

[4] `https://www.izotope.com/en/products/rx.html`

permettono di applicare le impostazioni più adatte rispetto ad ogni situazione. RX può essere utilizzato sia come software stand-alone che come plugin interno a una DAW.

Figura 3.3: Spectral Editor in RX

3.4 Protocollo OSC

OSC (Open Sound Control) è un protocollo open-source per la comunicazione tra computer, sintetizzatori audio e altri dispositivi multimediali. Viene sviluppato dopo il suo ben più impiegato antagonista, il protocollo MIDI, principalmente per ovviare ad una delle sue più grandi lacune, cioè la mancanza di possibilità di comunicare i dati e gestire le applicazioni via network. Il protocollo MIDI, infatti, scambia i dati mediante connessioni fisiche con cavi DIN a 5 poli, limitati e datati in quanto a costruzione, che limitano la quantità di dati e la velocità di trasmissione del protocollo (oltre a rischiare danneggiamenti dovuti a usura o uso improprio). OSC invece, sfruttando i protocolli TCP/IP delle reti internet, permette di poter utilizzare reti Ethernet o WiFi per lo scambio di dati, eliminando eventuali limiti dovuti alle interfacce hardware. L'unico suo contro resta una diffusione comunque limitata rispetto al protocollo MIDI, ormai adottato universalmente da tutte le applicazioni musicali ben prima dell'invenzione di OSC.

Il protocollo OSC è progettato per supportare un'architettura client/server, in cui il client invia i messaggi OSC e il server li riceve. Qualsiasi dato OSC viene inviato in unità denominate pacchetti, composti da un blocco contiguo di dati binari e un dato distinto che esprime la grandezza del pacchetto (espressa in bytes, sempre multiplo di 4). Un messaggio OSC è diviso in tre parti:

-Address pattern: stringa che specifica l'entità/le entità all'interno del server OSC a cui è diretto il messaggio e che tipo di messaggio si tratta.

-Type tag string: specifica il tipo di dato di ogni argomento.

-Arguments: sono i dati contenuti nel messaggio.

3.5 GyrOSC

Per poter sfruttare il protocollo OSC per trasferire dati tra dispositivi, l'offerta di applicazioni già esistenti è molto ampia, e la scelta deve essere ponderata in base alle feature di cui uno effettivamente ha bisogno. GyrOSC[5] è un app mobile di utility sviluppata per iOS, che spedisce i dati raccolti dai sensori di movimento dello smartphone sotto forma di pacchetti OSC attraverso una rete wireless. E' stata pensata proprio per controllare qualsiasi applicazione audio o video che supporti il protocollo OSC, come ad esempio Max/MSP, Pure Data, Isadora, ecc.

Figura 3.4: GyrOSC

[5] https://www.bitshapesoftware.com/instruments/gyrosc/

Abbiamo optato per GyrOSC in quanto, oltre a inviare alcuni tipi di dati utili che tra le altre app invece comparivano raramente (come matrici di rotazione e quaternioni), permette di regolare la frequenza di invio dei pacchetti (da 1 a 120 Hz) e invia i dati (in linea di massima) già normalizzati. Tra i vari dati che l'app raccoglie dai sensori di movimento, quelli che potrebbero essere più utili al progetto sono:

-Giroscopio (Rollio, Beccheggio, Imbardata), con valori compresi tra $+\pi$ e $-\pi$;

-Accelerometro (asse X, Y, Z), con valori compresi tra +3 e -3;

-Quaternioni (w, x, y, z), con valori compresi tra +1 e -1;

-Matrici di rotazione (matrice 3x3), con valori compresi tra 1 e -1;

-Vettore gravità (x, y, z), con valori compresi tra +1 e -1;

3.6 Pure Data

Il miglior modo di approcciarsi a un problema pratico come quello che ci siamo posti è utilizzando strumenti pratici. Pure data infatti è proprio questo, un ambiente di programmazione grafico molto pratico e diretto, è open-source, real-time e parte della più grande famiglia di linguaggi di programmazione che sfruttano le patch derivanti da Max (Max/FTS, ISPW Max, Max/MSP, ecc), sviluppati da Miller Puckette di IRCAM. Può inoltre essere utilizzato su qualsiasi tipo di piattaforma, che sia un computer, un dispositivo embedded (es. Raspberry Pi) o uno smartphone (con libpd, Droidparty (Android) e PdParty (iOS)).

"Pd nasce con la funzione di creare applicazioni audio e, da qualche tempo, video. Una foltissima comunità di sviluppatori, musicisti, hackers e appassionati sforna ogni giorno nuove applicazioni e librerie che potenziano e aumentano le funzionalità di questo ambiente versatile e libero. Si, perché Pure Data è un software libero, e i suoi sorgenti possono essere scaricati, studiati, modificati e redistribuiti da chiunque."[3]

E' un punto d'incontro per musicisti, artisti, ricercatori e sviluppatori, in quanto permette di creare graficamente un programma senza dover scrivere neanche una linea di codice,

preoccuparsi di editor di testo e quant'altro. Con Pd (abbreviazione di Pure Data) si possono elaborare e generare suoni, video, grafiche 2D/3D ed utilizzare strumenti come MIDI, OSC o sensori esterni.

Metodo di utilizzo: l'ambiente di programmazione vero e proprio in cui si scrivono gli algoritmi in Pd è la patch, utilizzabile in edit mode (permette di inserire gli elementi nella patch) e in run mode (permette di gestire la patch quando è in azione). Abbiamo diversi tipi di entità da manipolare all'interno di una patch per elaborare dati e segnali. Le entità principali sono gli oggetti, scatole rettangolari con inlets nella parte superiore e outlets nella parte inferiore. Le altre entità sono messaggi, commenti, numeri e liste. Tutte le varie entità vengono collegate tra loro mediante dei cavi di connessione, che specificano i rapporti tra i vari inlet e outlet. Per poter processare un qualsiasi segnale audio bisogna attivare il DSP (Digital Signal Processor), che elabora tutti i segnali in ingresso e uscita e permette le trasformazioni DAC/ADC (Digital to Analog/Analog to Digital Conversion). Altra feature molto utile di pure data è la possibilità di creare abstraction, che sono in altre parole delle patches aggiuntive, con i loro inlet e outlet, atte ad alleggerire la patch principale potendo scrivere su di esse parte del codice necessario. Una volta creata la abstraction, essa può essere usata creando un box object con lo stesso nome nella main patch, i cui inlet e outlet sono quelli della abstraction patch. Infine, nel caso servissero patch non presenti tra quelle native di Pure Data, queste ultime possono essere scritte da zero come external (in C o C++) o scaricate come external creati da altri utenti.

Capitolo 4

Realizzazione

4.1 Registrazione e campionamento

4.1.1 Registrazione

Per il lavoro di registrazione dei campioni audio del sonaglio, il processo è stato relativamente breve rispetto allo sviluppo del resto del progetto. Il reperto è stato portato direttamente dal museo da una responsabile che si è occupata di tenere e suonare il sonaglio, per un'ovvia questione di responsabilità e rischio dell'integrità dello stesso. Purtroppo, queste premesse hanno portato a una sessione di registrazione concentrata e non completamente esaustiva delle peculiarità acustiche dello strumento. Come spunto futuro, sarebbe sicuramente sensato provare a registrare nuovamente il sonaglio, investendoci del tempo in più e sperimentando tecniche di registrazione alternative per vedere se effettivamente possono completare al meglio la rappresentazione dello stesso. Per registrare lo strumento, si è impiegata una stanza adeguatamente trattata e una disposizione dei microfoni tanto classica quanto efficace, con due microfoni a condensatore a diaframma largo e un microfono dinamico.

Per la figura stereo (LR), sono stati impiegati due AKG C414, microfoni a diagramma largo omnidirezionali famosi ed impiegati per la loro versatilità, in quanto offrono la

Figura 4.1: Sessione di registrazione del sonaglio

possibilità di scegliere tra 9 diagrammi polari.[1] Per registrare lo strumento centralmente, è stato impiegato uno Shure SM57, standard nell'industria dell'audio, microfono dinamico direzionale spesso scelto per la ripresa di strumenti a percussione o comunque dall'alta pressione sonora.

4.1.2 Post processing, campionamento

Entrambe le fasi di registrazione e post processing avvengono su reaper, DAW scelta per questo lavoro vista la sua universalità e accessibilità. In seguito alle registrazioni, si poteva facilmente osservare come le tracce audio avessero bisogno di un lavoro di pulizia per poter rappresentare al meglio il suono dello strumento. Già dal primo ascolto il suono del sonaglio emerge debole, con non molto carattere e in conflitto con molti rumori di fondo, sia ambientali che di hum[2]/clip[3]. Per prima cosa, mediante l'equalizzatore grafico di Reaper stesso (Rea-EQ), ho cominciato a ripulire la traccia a grandi linee eliminando i rumori che si trovavano più lontani dal corpo delle frequenze del sonaglio stesso, princi-

[1] Rappresentazione grafica della sensibilità di un microfono in funzione della direzione di provenienza del segnale, sono dei diagrammi di Nyquist.

[2] Rumore di fondo causato per vari motivi da corrente elettrica, es. induzione, trasformatori

[3] Distorsione della forma d'onda data da un amplificatore in saturazione

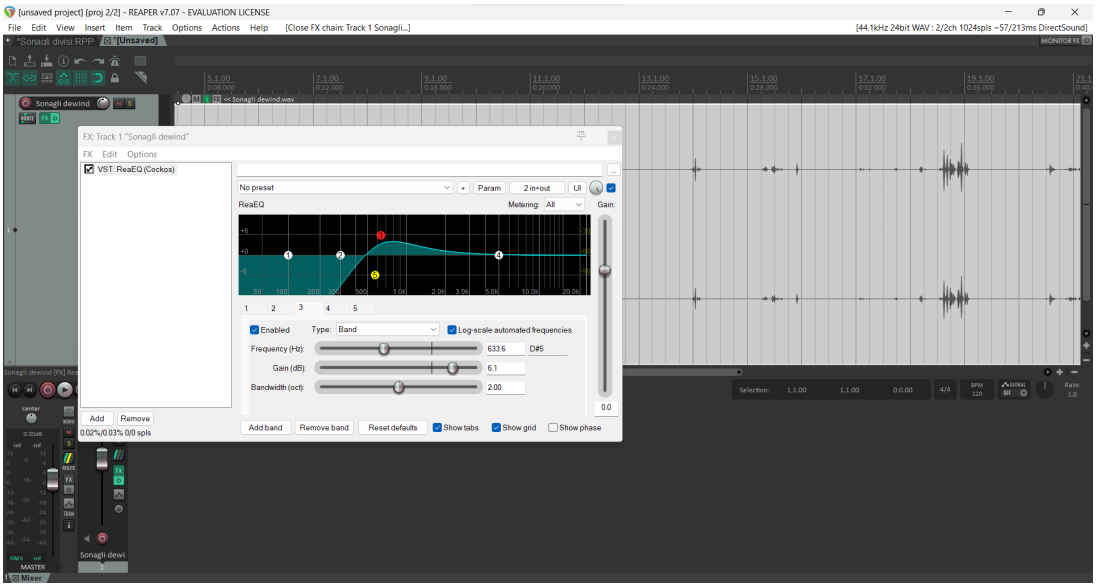


Figura 4.2: Tool di equalizzazione di Reaper

palmente con un filtro passa alti[4] intorno a 300/400 Hz e un cut a campana abbastanza stretta intorno ai 12k Hz. Ho in seguito applicato un Gate[5] morbido per eliminare rumori di intensità minore del sonaglio e un compressore per evitare che quando suonato forte lo stesso clippasse. In seguito, per cercare di pulire al meglio le tracce, ho utilizzato un plugin esterno di equalizzazione grafica, Fabfilter Pro-Q, utile per equalizzazioni "chirurgiche" di pulizia, con cui ho rimosso sempre con campana stretta alcune frequenze fastidiose sulle medio-alte e dato un po' più di corpo al sonaglio dando del gain sulle medio-basse. Ho poi concluso la fase di processing utilizzando un altro tool, Izotope RX, pensato appositamente per il restauro audio e l'improvement di registrazioni acusticamente "povere". Con quest'ultimo, sono riuscito ulteriormente a pulire la traccia da rumori vari, mediante strumenti come un DeWind, un DeCrackle e un DeClip. Sono così giunto a un risultato di pulizia soddisfacente, purtroppo però non perfetto, in quanto molto del rumore condivideva le frequenze con il sonaglio, e preservare il suono dell'ultimo non permetteva di eliminare completamente le interferenze esterne. Per poter procedere con un campionamento soddisfacente, ho dovuto fare una selezione di un pool di suoni che fossero

[4]Strumento per eliminare ogni frequenza al di sotto di una certa soglia
[5]Tool audio che elimina qualsiasi suono che non superi una certa soglia, detta threshold

caratteristici e al tempo stesso tra loro distinti, così da permettere una riproduzione più fedele possibile dello strumento senza appesantire poi l'implementazione della simulazione con troppi campioni diversi. Dopo aver selezionato un primo gruppo di 12 campioni, in seguito a una rivalutazione dell'entità del progetto sono arrivato a una selezione finale di 6 campioni: due campioni di suono di rotolamento (evento di sonaglio ruotato dolcemente, senza che le palline cozzino sulle pareti dello stesso), due campioni di "click" di bassa intensità e un campione di alta intensità(evento di sonaglio scosso, quindi con palline che cozzano sulle pareti) , e un campione di scossa di sonaglio con quanto di tempo più lungo degli altri.

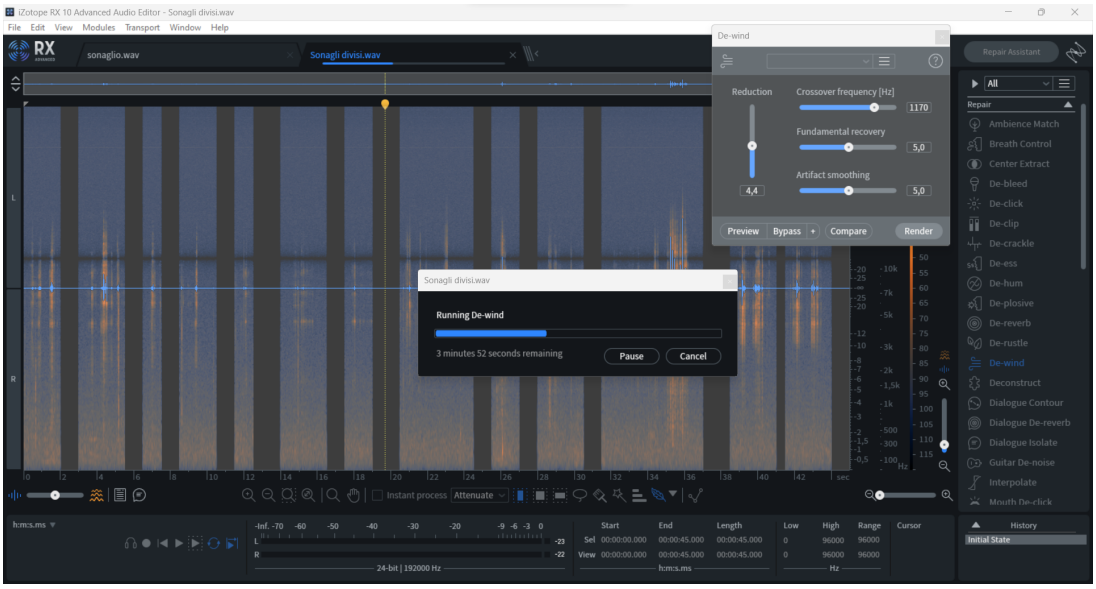


Figura 4.3: Schermata di Izotope RX

4.2 Implementazione

4.2.1 Patch PD fileplayer

Per poter familiarizzare un pò con il software, essendo comunque Pure Data non così usuale, sono state create delle patch "intermedie" per prendere familiarità con le interfacce e avvicinarsi gradualmente all'obiettivo. La prima patch sviluppata è stata una per la

gestione del playback di file audio salvati nel computer, Figura 4.4. Mediante i bang[6] si può scegliere il file audio da una cartella (SEARCH), farlo partire (START) e fermarlo (STOP). La patch è basata sull'uso dell'oggetto *openpanel*, "quando openpanel riceve un bang, apre un file browser da cui selezionare il file che comparirà in outlet"[4]. In aggiunta, sono stati messi uno switch per mettere in loop il file, uno slider per il volume e uno shortcut per attivare/disattivare il DSP.

Un'altra patch derivata da quella che abbiamo appena visto è KeyboardPlayer, Figura 4.5. Come lascia immaginare il nome, mediante l'oggetto *keyname* la patch invia ad un messaggio *open* gli input della tastiera. Salvando nella stessa cartella della patch i campioni audio con nomi elementari relativi a singoli input della tastiera, la patch suonerà i campioni al premere dei rispettivi tasti. Per il resto, il funzionamento è similare a quello di PlaybackSingoloFile.

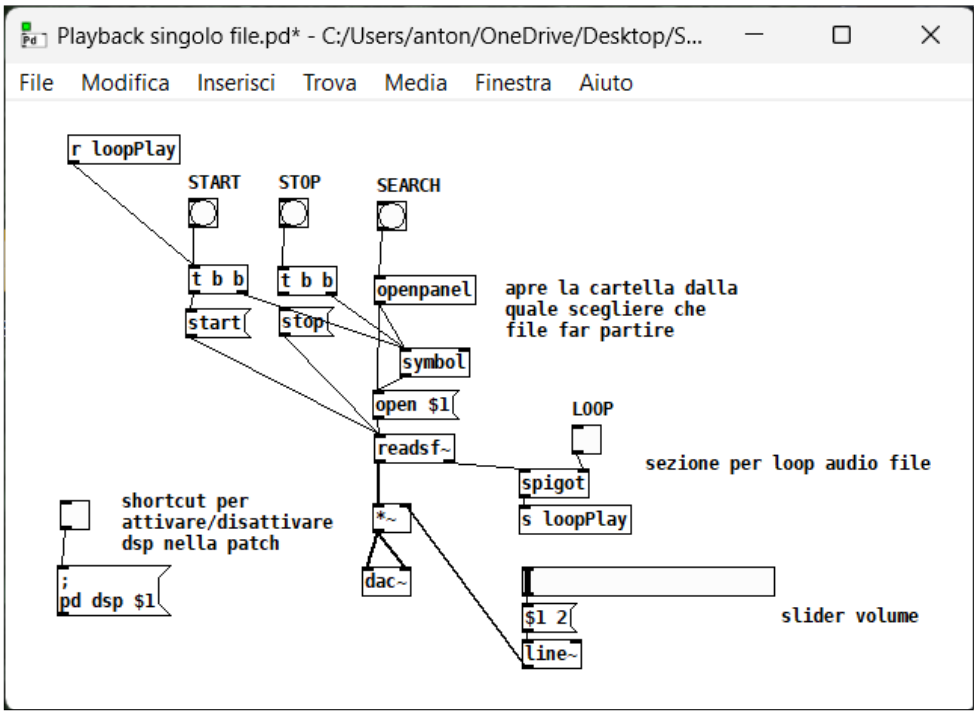


Figura 4.4: Patch Playback singolo file

[6] "do your task", azione atomica che attiva l'oggetto che la segue

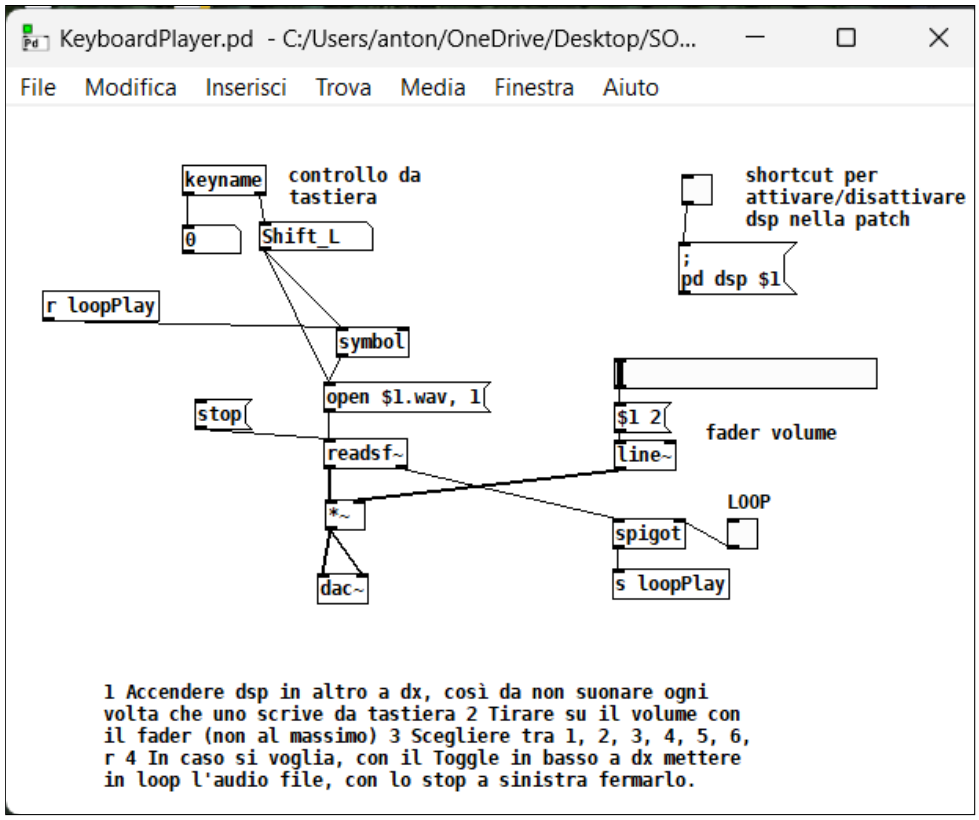


Figura 4.5: Patch Keyboardplayer

4.2.2 Elaborazione dati sensori

Per poter procedere nella creazione delle patch necessarie alla simulazione del sonaglio, bisognava comprendere che tipo di dati sarebbero arrivati dai sensori di movimento del telefono e in che modo avremmo potuto elaborarli per ottenere una rappresentazione fluida e coerente con i movimenti dello strumento. Abbiamo visto prima la quantità di opzioni fornite da GyrOSC, e anche quali di queste potrebbero fare al caso nostro. Altra funzione che ci sarà utile dell'app è la possibilità di settare il punto di riferimento ad angolazioni diverse del dispositivo, per poterlo quindi scegliere come più ci comoda. Il riferimento che considereremo di partenza è con lo smartphone poggiato su un ripiano, con la parte inferiore rivolta verso l'utente.

La patch riportata in Figura 4.6 mostra in che modo vengono presi i dati in ingresso dalla connessione OSC (per poter alla fine controllare tutto dalla main patch, i valori delle port che inizializzano/interrompono la connessione OSC vengono mandati con dei return dalla

patch main), come vengono già in questa prima patch elaborati (in particolare quelli dell'accelerometro) e come vengono inviati mediante degli oggetti send alle patch successive. Si è deciso di provare a gestire i movimenti utilizzando per le scosse del sonaglio il modulo dei 3 vettori dell'accelerometro, mentre per le sue rotazioni delle matrici di rotazione 3x3. Per quanto riguarda l'accelerometro sarebbe inutile, visto il livello di precisione che vogliamo ottenere, tenere separati i valori dell'accelerometro per poterne ricavare anche la direzione. Se il sonaglio viene scosso, suonerà a prescindere dalla direzione in cui lo si fa.

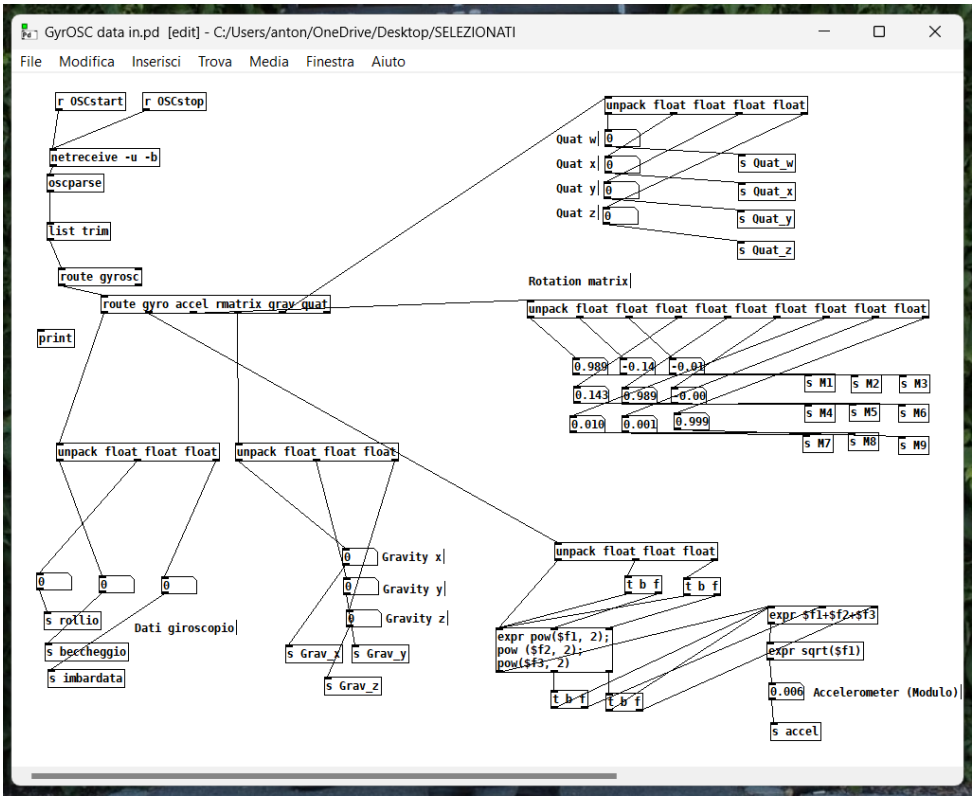
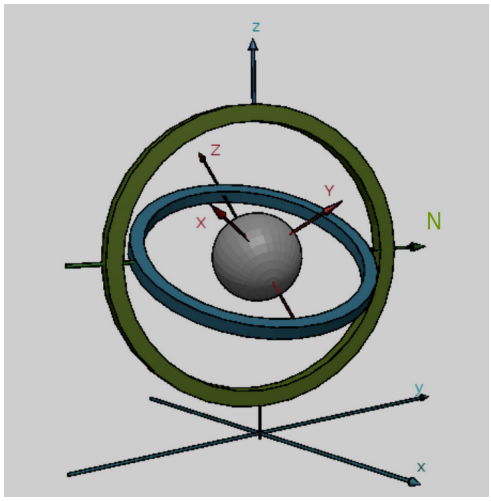


Figura 4.6: Patch GyrOSC data in

Matrici di rotazione Le matrici di rotazione sono matrici di trasformazione usate per rappresentare una rotazione attorno a uno o più assi dello spazio euclideo. Il riferimento per queste matrici è un sistema di coordinate destrorso, in cui le rotazioni di vettori in senso antiorario portano angoli positivi. Le matrici di rotazione sono matrici quadrate 3x3, tutte ortogonali e con determinante uguale a 1. Nel campo di nostro interesse, lo spazio tridimensionale, con queste matrici si rappresentano rotazioni attorno a uno dei

tre assi x, y e z. Le matrici di rotazione di base riportate in Figura 4.7a rappresentano la rotazione di un vettore $v = (x, y, z)$ che, moltiplicato per una o più di esse, ci da il vettore $v' = (x', y', z')$, cioè v ruotato di un angolo *theta* (secondo la regola della mano destra). Una qualsiasi rotazione complessa R può essere considerata come composizione delle 3 rotazioni semplici (4.7b), e quindi usata per tracciare le rotazioni del vettore v (attenzione, R si compone moltiplicando tra loro le matrici; il prodotto di matrici non è commutativo, pertanto l'ordine di rotazione è fondamentale per sapere in che ordine vanno moltiplicate le matrici tra di loro). Pertanto, otterremo v' come $v' = R * v$.



(a) Sospensione cardanica a 3 assi

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(b) Matrici di rotazione base

$$\theta = \arccos\frac{z}{\sqrt{x^2+y^2+z^2}} = \arccos\frac{z}{r} = \begin{cases} \arctan\frac{\sqrt{x^2+y^2}}{z} & \text{if } z > 0 \\ \pi + \arctan\frac{\sqrt{x^2+y^2}}{z} & \text{if } z < 0 \\ +\frac{\pi}{2} & \text{if } z = 0 \text{ and } \sqrt{x^2+y^2} \neq 0 \\ \text{undefined} & \text{if } x = y = z = 0 \end{cases}$$

$$\varphi = \operatorname{sgn}(y)\arccos\frac{x}{\sqrt{x^2+y^2}} = \begin{cases} \arctan(\frac{y}{x}) & \text{if } x > 0, \\ \arctan(\frac{y}{x}) + \pi & \text{if } x < 0 \text{ and } y \geq 0, \\ \arctan(\frac{y}{x}) - \pi & \text{if } x < 0 \text{ and } y < 0, \\ +\frac{\pi}{2} & \text{if } x = 0 \text{ and } y > 0, \\ -\frac{\pi}{2} & \text{if } x = 0 \text{ and } y < 0, \\ \text{undefined} & \text{if } x = 0 \text{ and } y = 0. \end{cases}$$

Figura 4.8: Coordinate sferiche

Coordinate sferiche Le matrici di rotazione sono un ottimo modo per esprimere le rotazioni di un corpo rigido in quanto molto precise e non ambigue, ma per l'elaborazione di una simulazione come la nostra diventano molti dati da considerare. Possiamo però osservare come queste ultime, partendo da una posizione di riferimento che altro non è che la matrice identica I, sono infatti anche la matrice di arrivo della rotazione, in quanto $R' = R * I = R$. Le tre colonne della matrice sono infatti i 3 versori direttori del corpo rigido (infatti per I sono $i_x = (1, 0, 0)$, $i_y = (0, 1, 0)$, $i_z = (0, 0, 1)$). A questo punto, invece che considerare per ogni vettore le sue coordinate cartesiane (x, y, z), possiamo ragionare in termini di coordinate sferiche: dovendo noi tracciare i movimenti dei versori, il raggio r sarà sempre uguale a 1, e ci potremo preoccupare solo di ϕ e θ. Nel sistema di coordinate sferico ϕ è l'angolo azimutale, che la proiezione ortogonale del vettore sul piano xy forma nello scostamento dall'asse x (o y). θ invece è l'angolo di inclinazione (o angolo polare) del vettore rispetto al semiasse positivo z. Da notare che ϕ ha valori compresi tra $-\pi$ e $+\pi$, mentre θ compresi tra 0 e π. Le formule di conversione da coordinate cartesiane a sferiche sono riportare in Figura 4.8, mentre in Figura 4.9 possiamo osservare la patch creata per convertire i dati ottenuti da GyrOSC riguardanti le matrici di rotazione. Sulla sinistra otteniamo (usando la traccia della matrice) l'angolo generico di rotazione θ, mentre sulla destra calcoliamo gli angoli ϕ e θ relativi ai tre assi. Gli angoli ottenuti vengono poi mandati alle altre patch mediante dei send.

4.2.3 Macchina a stati

Per poter riprodurre al meglio i movimenti fondamentali del sonaglio, si è dovuto studiare quali di essi effettivamente producessero suono e di che tipo esso fosse (come detto prima, abbiamo raggruppato in tre suoni fondamentali, rotolamento, click debole e click forte). Per poter distinguere gli eventi che portano ad un determinato suono, ho sfruttato nel processo di implementazione il concetto di macchina a stati. Una macchina a stati è una rappresentazione di un sistema reattivo basato su eventi; essa passa da uno stato all'altro se vengono soddisfatte le condizioni che controllano il cambiamento. Viene rappresentata attraverso quello che si chiama diagramma di stato, cioè un grafo orientato dove i nodi rappresentano gli stati e gli spigoli, etichettati con l'evento scatenante e condizio-

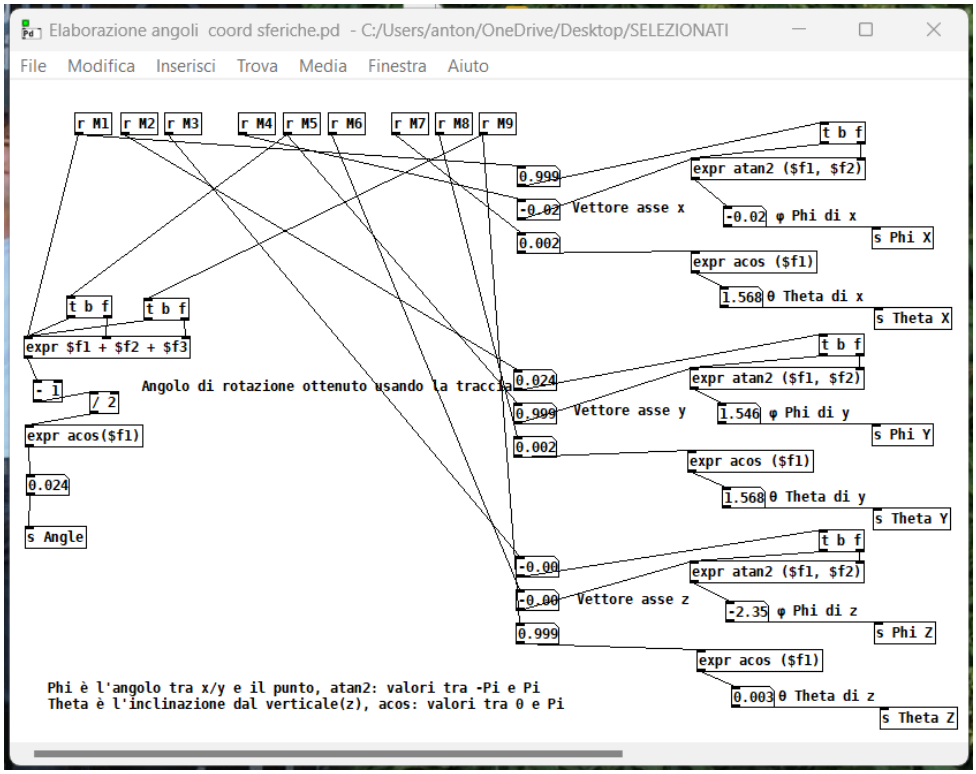


Figura 4.9: Patch Elaborazione angoli coord sferiche

ne di guardia, rappresentano le transizioni; si tratta quindi di un formalismo chiaro e semplice.[5]

L'uso delle macchine a stati permette la realizzazione di sistemi la cui gestione degli eventi dipende sia dal tipo di evento sia dal contesto di esecuzione del sistema. Vengono utilizzate principalmente per sistemi informatici, ma spesso anche per vari tipi di logiche complesse in sistemi dinamici. Nel nostro caso , abbiamo cercato di ricondurre i movimenti del sonaglio al minor numero di stati possibile, di modo da snellire le patch e poter arrivare rapidamente a un prototipo finito. In tabella (4.1) possiamo vedere gli stati del nostro sistema, la loro descrizione e le posizioni degli angoli ϕ e θ per x, y e z.

4.2.4 Creazione stati, Patch main

In seguito alla definizione di una macchina a stati coerente con le nostre necessità, era necessario implementare una patch che applicasse il concetto al nostro progetto. In Figura 4.11, si può osservare come, utilizzando i dati degli angoli ottenuti dalla patch Elabora-

Stato	Descrizione	ϕ_x, θ_x	ϕ_y, θ_y	ϕ_z, θ_z
S1	Dispositivo poggiato su ripiano, parte inferiore rivolta verso l'utente	$(0, \pi/2)$	$(\pi/2, \pi/2)$	$(undef., 0)$
S2	Dispositivo in piedi, rivolto verso l'utente	$(0, \pi/2)$	$(undef., \pi)$	$(\pi/2, \pi/2)$
S3	Dispositivo a testa in giù, rivolto in senso opposto	$(0, \pi/2)$	$(undef., 0)$	$(-\pi/2, \pi/2)$
S4	Dispositivo poggiato su ripiano, schermo verso sinistra	$(undef., \pi)$	$(\pi/2, \pi/2)$	$(0, \pi/2)$
S5	Dispositivo poggiato su ripiano, schermo verso destra	$(undef., 0)$	$(\pi/2, \pi/2)$	$(+/-\pi, \pi/2)$
S6	Dispositivo poggiato su ripiano, schermo verso il basso	$(+/-\pi, \pi/2)$	$(\pi/2, \pi/2)$	$(undef., \pi)$

Tabella 4.1: Modello semplicistico di macchina a stati per tracciare il sonaglio

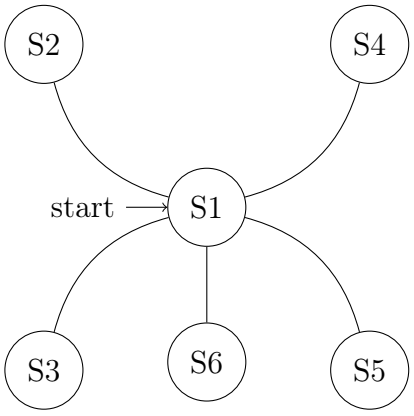


Figura 4.10: Diagramma della macchina a stati

zione angoli coord sferiche (Figura 4.9), vengono definiti gli stati del nostro sistema prima descritti. La forma non è delle più eleganti, ma per riuscire a implementare il tutto solo con oggetti nativi di Pure Data, il modo più semplice era attraverso una cascata di if-else combinati con logica relazionale. Per ogni stato si verificano due delle tre coppie (ϕ, θ), escludendo nei casi limite la coppia in cui $atan2(y, x)$ dava come risultato $undefined$. Infine, otteniamo un riscontro positivo (1) in caso di stato attivo e un riscontro negativo (0) in caso di stato disattivo, risultato che mandiamo con dei *send* alla patch finale.

La patch conclusiva (Figura 4.12) si occupa della gestione di tutto il sistema da un solo

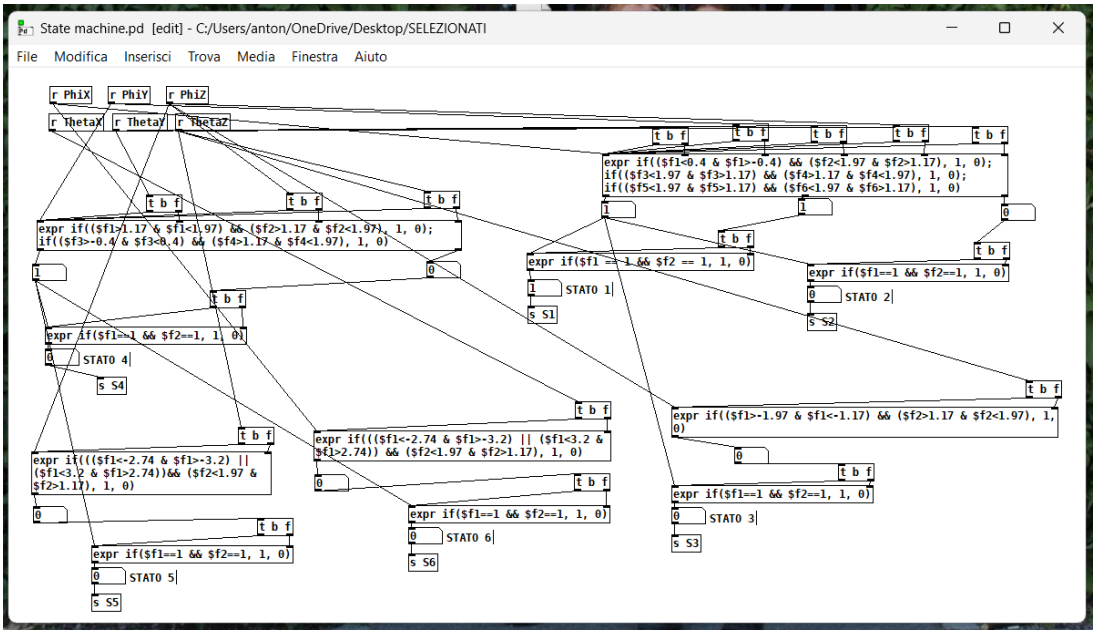


Figura 4.11: Patch macchina a stati

pannello, mediante degli shortcut implementati per migliorare la facilità d'uso. Abbiamo uno switch per attivare/disattivare il DSP, uno slider volume per l'uscita e due messaggi (in alto a sx) che inviano il port number con dei send alla patch per la connessione OSC.

Per prima cosa, il return del modulo dell'accelerometro viene mandato in due if che dividono gli input da accelerometro in due categorie (forte e debole) in base all'intensità del valore. L'input forte attiva direttamente un messaggio che apre il file *.wav* del campione. Quello debole passa prima per un oggetto *random* (che crea per ogni bang un numero random compreso tra 0 e 100), per poi attivare rispettivamente due messaggi che aprono due file distinti; ognuno di essi viene attivato il 50% delle volte.

Nella parte destra della patch, possiamo vedere come ogni valore ottenuti dal check degli stati nella patch State Machine (Figura 4.11), venga passato a una box messaggio con il numero dello stato rispettivo. All'attivarsi di uno stato, l'oggetto *change* dà un bang solo se il valore cambia rispetto al precedente (*change* ha come argomento 1 per evitare che lo Stato 1 suoni all'inizializzazione della patch). Il passaggio da uno stato all'altro attiva sempre un oggetto random che, rispettivamente con una probabilità del 70% e del 30%, attiverà i due messaggi *open* con i rispettivi file *.wav*.

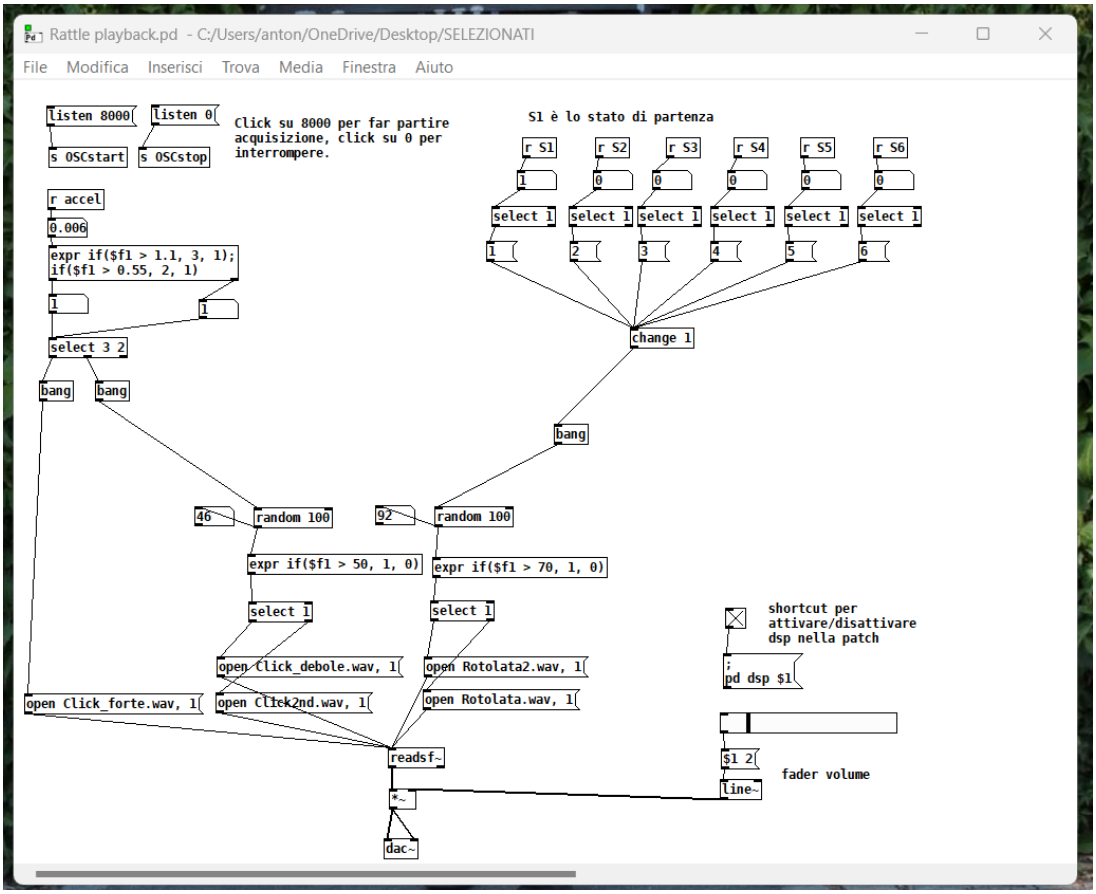


Figura 4.12: Patch operativa

Con questa patch si conclude il lavoro di implementazione della simulazione. Un valore ottimale di frequenza (sintonizzabile da app) per rendere verosimile il tutto si aggira intorno ai 5 Hz, di più renderebbe artefatti i movimenti e, facendo partire troppo frequentemente le tracce audio, bisognerebbe rivedere parte dell'implementazione. Un modello più completo con definizione anche delle transizioni oltre che degli stati potrebbe portare a una rappresentazione più verosimile dello strumento, ma viste le approssimazioni fatte sia in fase di campionamento che di scrittura delle patch questo approccio renderebbe il tracking dei movimenti ancor più artificioso e irregolare. Il livello di accuratezza raggiunto con questa simulazione è più che soddisfacente, e sarebbe poco saggio comprometterlo per cercare la perfezione (sempre molto difficile da raggiungere quando si parla di modelli fisici).

Capitolo 5

Conclusioni

Con questa tesi si vuole dimostrare come, con le tecnologie e possibilità di oggi, si possa volgere uno sguardo curioso al passato per sperimentarlo in modo interattivo, ma comunque sicuro per quanto riguarda l'incolumità dei manufatti. Con le patch riportate nelle figure 4.11 e 4.12, siamo giunti a una simulazione soddisfacente del comportamento del sonaglio, riuscendo a riprodurre i movimenti e suoni fondamentali dello strumento senza deviare troppo dalla natura dello stesso.

Per raggiungere questo risultato, il sonaglio è stato prima registrato, per poi venire processato e campionato in quelli che saranno i suoni che caratterizzano la simulazione. In seguito, dopo aver elaborato matematicamente i dati ottenuti dai sensori (per mezzo di matrici di transizione e coordinate sferiche) sono state create delle patch per poter riprodurre questi campioni in modi diversi, ad esempio attraverso degli input da tastiera. Per concludere, sfruttando il modello delle macchine a stati, abbiamo ottenuto le patch State Machine e Rattle playback con le quali ricondurre gli effettivi movimenti dello smartphone ai suoni campionati del sonaglio.

5.1 Sviluppi futuri

Gli spunti per sviluppi futuri, come si può immaginare, sono molteplici, vista l'implementazione comunque semplicistica per cui abbiamo optato. Si potrebbe migliorare il tracking elaborando allo stesso tempo più tipi di dati ottenuti dai sensori di movimento, come ad esempio quaternioni e giroscopio, o anche ridurre i dispositivi in uso per la simulazione eseguendo le patches di Pure Data direttamente su smartphone (mediante app come DroidParty (Android) e PdParty (iOS)) senza supporto del computer. Uno step successivo futuro potrebbe essere quello di creare un vero e proprio prototipo fisico, con sensori di movimento e attuatori interni, che mediante le patch sviluppate possa riprodurre il suono del sonaglio (magari con dei suoi altoparlanti integrati), così da diventare una vera e propria rappresentazione dello strumento a sè stante.

Elenco delle figure

Elenco delle tabelle

Bibliografia

[1] Curt Sachs. *The history of musical instruments*. Courier Corporation, 2006. 6

[2] Catalogo beni culturali della regione veneto. `https://www.culturaveneto.it/it/beni-culturali/archeologia/5ced4b621ebd0e66c81d22bf`. 6

[3] Francesco Bianchi. Inventare il suono con pure data. *Manuale introduttivo di musica elettronica*, 2010. 14

[4] Miller Puckette et al. Pure data: another integrated computer music environment. *Proceedings of the second intercollege computer music concerts*, pages 37–41, 1996. 21

[5] Marcomin Gabriele. Studio delle macchine a stati gerarchici e sviluppo di importatori per matrix. Master's thesis, Università degli studi di Padova, 2017. 26

[6] Francesco Bianchi, Alessandro Cipriani, and Maurizio Giri. *Pure Data: Electronic Music and Sound Design*. Contemponet, 2021.

[7] Grant Sanderson and Ben Eater. Quaternions and 3d rotations, an explorable video series. `https://eater.net/quaternions`.

[8] David Sachs and Google Tech Talks. Sensor fusion on android devices: A revolution in motion processing, 2010. `https://www.youtube.com/watch?v=C7JQ7Rpwn2k`.

[9] Eric Lengyel. *Mathematics for 3D Game Programming and Computer Graphics*. Cengage Learning, 2011.

[10] Marco Prevedelli and Lorenzo Bechelli. Implementazione di un sistema di navigazione inerziale basato su sensori mems.

[11] Johannes Kreidler. *Loadbang: Programming Electronic Music in Pure Data*. Wolke V.-G, 2009.

[12] Fregnan Andrea. Controllo gestuale di modelli fisici per la sintesi sonora di membrane percosse. Master's thesis, Università degli studi di Padova, 2012.

[13] Giovanni De Mezzo. La creazione di un archivio digitale: aspetti tecnologici e implicazioni musicologiche del riversamento conservativo e digitalizzazione dei documenti audio. *Ri-mediazione dei documenti sonori*, pages 1000–1018, 2006.

[14] S Iman Shirinbayan and Jochem W Rieger. An mr-compatible gyroscope-based arm movement tracking system. *Journal of Neuroscience Methods*, 280:16–26, 2017.

[15] S. Bonomi. Rovigo e il polesine. adrias, atria. rovigo nell'antichità. *Luoghi e tradizioni d'Italia, Veneto*, 2003.

Ringraziamenti

A conclusione di questo progetto di tesi, volevo ringraziare tutte le persone che, in un modo o nell'altro, mi hanno aiutato ad arrivare fino a qui.

Ringrazio Misha, per avermi sempre e comunque sostenuto in questi anni, spronandomi a non mollare e arrivare alla fine di questo percorso, e aiutandomi a credere nelle mie capacità, in quanto prima persona a farlo.

Ringrazio il mio tutor Giulio Pitteri, per avermi seguito nello sviluppo di questo progetto, aiutandomi a fare chiarezza sugli argomenti più intricati e dandomi spunti mai scontati.

Ringrazio i miei genitori, per i loro sacrifici, per il sostegno incondizionato degli ultimi anni e per avermi permesso di scegliere il mio percorso e di viverlo nel bene e nel male. Un buon genitore rispetta sempre le scelte di un figlio, anche quando non pensa siano le migliori.

Ringrazio Mattia, figura sempre presente e il cui sostegno ho apprezzato anche perché non dovuto, per avermi aiutato a lavorare quando avevo bisogno di lavorare, e a studiare quando avevo bisogno di studiare.

Ringrazio Simone, per il supporto sempre offerto in questa triennale, sempre benaccetto da un amico ben preparato.