Conservatorio di musica “Francesco Morlacchi” • Perugia

Anno accademico 2016-2017

Diploma accademico di I livello in Musica elettronica

Moment V

& la concezione verticale del tempo secondo Jonathan D. Kramer

Candidato

Alessandro Fiordelmondo

Matricola n. 5364

Relatori

M.o Nicola Casetta

M.o Roberto Grisley

«la musica è continua e l'ascolto intermittente»
John Cage

Sommario

Introduzione e commento analitico dell'indice

Questo lavoro ha origine dalla lettura del saggio *The Time of Music* (1988) di Jonathan D. Kramer, uno dei principali studiosi musicali contemporanei[1]. Il suo lavoro è incentrato in un'elaborata analisi delle opere musicali passate e contemporanee, indagate con particolare riguardo attraverso l'aspetto del tempo. Oltre a dare una chiave di lettura interessante a molte composizioni già note, Jonathan Kramer ci propone anche degli aspetti musicali che sono stimolanti per un fine compositivo. Proprio da questo genere di stimolo nasce la volontà di realizzare questo lavoro.

Mi sono chiesto se sia possibile realizzare un'opera d'arte musicale che annulli lo scorrimento del tempo. Questa domanda è celata nel capitolo dodicesimo, *Time and Timeless*, del volume *The Time of Music*. Per mezzo di questo lavoro sarà realizzata la composizione di un'opera d'arte musicale che soddisfi le peculiarità proposte dalla domanda. Separatamente indagheremo tale quesito da un punto di vista concettuale al fine di poter proporre un'interpretazione sul tema messo in discussione.
Per la stesura del lavoro esporremmo inizialmente il profilo concettuale, ai fini di una migliore comprensione dell'intero elaborato.

Per formulare un'adeguata risposta teorica discuteremo del tempo, più approfonditamente del tempo musicale e in seguito dell'opera d'arte musicale. Avendo così una chiara preparazione torica sul tema in questione avremo anche una chiave di lettura dell'opera d'arte musicale che nella seconda parte del lavoro, ossia nel profilo

[1] Jonathan D. Kramer (1942-2004) fu particolarmente attivo come teorico musicale ma fu anche conosciuto come compositore. Tra i sui innumerevoli maestri troviamo Karlheinz Stockhausen per la composizione e John Chowning per la computer music. Ha insegnato composizione alla Columbia University dal 1988 fino alla sua morte. Di maggiore importanza sono le sue pubblicazioni sul tempo musicale, come *The time of Music* (1988), e sul postmodernismo musicale, come *Postmodern Music, Postmodern Listening* (2004) completato appena prima della sua morte (pubblicato solamente nel 2016).

pratico, andrò a sviluppare. In questa seconda parte m'impegnerò infatti alla realizzazione pratica della composizione che rispetterà i concetti chiave proposti dalla precedente lettura teorica.

Nel primo capitolo - *Cos'è il tempo?* - daremo una panoramica sulle diverse concezioni del tempo nel pensiero occidentale. Non scenderemo troppo nel dettaglio ma daremo una lettura ai concetti di tempo spazializzato, tipico della cultura occidentale, e di durata reale. Per questo motivo ci soffermeremo sul pensiero di Henri Bergson che troviamo su due dei suoi più importanti saggi: *Saggio su i dati immediati della coscienza*(1889) e *Durata e simultaneità* (1922). La lettura di questi pensieri sarà essenziale per una miglior comprensione dei capitoli seguenti. Infatti, nel secondo capitolo - *il tempo musicale* - approfondiremo da un punto di vista musicale la concezione del tempo. È ovvio che il tempo musicale abbia sempre avuto, nel corso della storia, una forte relazione con il pensiero dei filosofi della loro epoca. Sarà infatti interessante vedere anche che, in filosofia, la musica viene presa come campione per la spiegazione di teorie sul tempo. E vedremo, come punto centrale del capitolo, che la musica, a differenza delle altre arti, è di natura strettamente temporale. Essa si percepisce mediante il passare del tempo, elemento indispensabile per la costituzione di ogni evento di tipo musicale.

Con il terzo capito – *Le forme del tempo musicale*: *Linearità* e *non-linearità* - entreremo nella parte centrale del lavoro. A questo punto inizieremo a trattare di *The Time of Music* di Jonathan Kramer. I due concetti che identificano il capitolo, ossia *linearità* e *non-linearità*, sono i due concetti che stanno alla base dell'analisi musicale dell'autore. Questi s'identificano molto semplicemente con la contrapposizione dei concetti di divenire ed essere.

Nel quarto capitolo - *La musica verticale* - ci addentreremo nel cuore del lavoro concettuale. Indagheremo principalmente il lavoro svolto da Jonathan D. Kramer nel dodicesimo e ultimo capitolo di *The Time of Music*. L'autore qui sbilancia il peso della sua ricerca sul concetto di *non-linearità* (essere), proponendo una tipologia di musica sprovvista di *linearità* (divenire). Per questo motivo la nomina "musica verticale" poiché priva di elementi orizzontali, in altre parole priva di elementi che scandiscono il tempo. Nello svolgimento del capitolo tratteremo di composizioni che utilizzano un tempo musicale come questo, o molto simile, tra cui le sculture sonore.

A questo punto e attraverso il quinto e il sesto capitolo - rispettivamente *La percezione verticale e l'"interattività coscienziale"* e *L'aura* - affronteremo un punto interessante del quesito introduttivo. Senza indagare profondamente il senso di un'opera d'arte in generale spiegheremo il valore artistico di un'opera musicale senza tempo. A quest'ultimo scopo studieremo la percezione di tale musica, cioè l'ascolto. Oltre a molti spunti che già ci fornisce Jonathan Kramer riguardo a quest'argomento, parleremo anche del concetto di aura che Walter Benjamin ci propone nelle proprie opere e in particolar modo in *L'opera d'arte nell'epoca della sua riproducibilità tecnica* – che trovo molto interessante per una discussione sulla musica senza tempo.

Prima di passare alla realizzazione pratica dell'opera descriverò nel capitolo sesto - *Moment V e* l'installazione Web - alcune caratteristiche di tale composizione. In un primo momento spiegherò il lato affettivo dell'opera e insieme i vecchi tentativi di realizzazione, quindi gli antecedenti *Moment* (*I*, *II*, *III*, *IV*), pubblicati indipendentemente nel marzo del 2016. Poi illustrerò la scelta della forma compositiva, cioè dell'installazione web.

Nell'ultima parte, ossia nel profilo pratico del lavoro, mostrerò i procedimenti tecnici della composizione. In un primo momento mostrerò la struttura web che sta alla base della composizione. Successivamente descriverò il tipo di sintesi utilizzata, che una *multilayer resampling synthesis* (*sintesi di ricampionamento multistrato*). Questa è una sorta di granulazione che utilizza però dei grani di ampio respiro, cioè di lunghe durate, caratterizzati da grandi inviluppi. Il risultato di questa sintesi su un dato file audio genera un nuovo suono in movimento. Se pur in movimento il suono rimarrà sempre delimitato dai confini espressi dal file audio di partenza così da eliminare uno sviluppo orizzontale. La sovrapposizione di più suoni trattati mediante questa sintesi sarà il timbro "in movimento" della nostra installazione. I file audio che saranno dati all'elaborazione deriveranno da registrazioni audio dettagliate dell'organo dell'auditorium del conservatorio di Perugia.

Negli ultimi capitoli mostreremo la realizzazione web di questa istallazione. Scenderemo nei particolari sull'utilizzo del linguaggio di programmazione lato-client, *Javascript*, per l'elaborazione audio e su tutta la struttura server, quindi dei database e della loro manipolazione. Ci soffermeremo in particolar modo nell'*API* (*Application Programming Interface – interfaccia di programmazione di un'applicazione*) di *Javascript*, dedicata all'elaborazione audio, *Web Audio API*. Quest'*API*, da pochi anni in circolazione, agevola la programmazione di algoritmi per l'elaborazione audio in un ambiente web. *Web Audio API* è anche un mezzo che ci rende efficace l'utilizzo del web come strumento per la composizione.

Il risultato di questa seconda parte sarà la produzione di un sito internet che avrà, come ambiente virtuale, la funzione di ospitare l'*istallazione web*.

In conclusione tenteremo di rispondere al quesito di partenza, cioè se effettivamente è possibile realizzare un'opera d'arte musicale senza tempo. Come tutta la stesura del lavoro anche l'esito sarà suddiviso in due parti, cioè quella concettuale e quella pratica. Questo perché troveremo due evidenti e differenti risultati.

Parte I

1. Cos'è il tempo?

Attraverso questa prima parte si vuole dimostrare l'importanza sostanziale del tempo nel suono e più in generale nella musica, dove addirittura questo componente viene espresso ai limiti della propria sussistenza. Introdurremo così un problema del tutto artistico attraverso una domanda per lo più filosofica. Pertanto nell'esposizione di questo capitolo definiremo cosa noi intendiamo per tempo. In seguito utilizzeremo questa spiegazione per un'indagine musicale.

Per questo motivo inauguriamo questo lavoro con la domanda: "cos'è il tempo?". Questa domanda ha animato le riflessioni di una grande moltitudine di pensatori che vanno dall'antica Grecia alla contemporaneità e così da una concezione del tempo ciclico ad una lineare. Non solo, buona parte degli esponenti della musica contemporanea e sperimentale hanno rivolto la loro attenzione nel tema del tempo, affrontandolo per mezzo di scritti e composizioni. Resta il fatto che la risposta che forse compendia in maggior modo le diverse meditazioni è quella di Sant'Agostino: «Se nessuno me lo chiede, lo so; se dovessi spiegarlo a chi me ne chiede, non lo so: eppure posso affermare con sicurezza di sapere che se nulla passasse, non esisterebbe un passato; se nulla sopraggiungesse, non vi sarebbe un futuro: se nulla esistesse, non vi sarebbe un presente»[1]. Questa risposta, che inaugura l'indagine sul tempo di Sant'Agostino, nell'XI libro delle *Confessioni*, mette in larga evidenza l'indeterminatezza del tempo. Infatti egli proclama l'interiorità della percezione del tempo e la difficoltà nel definirlo oggettivamente, al di là della propria percezione.

[1] Agostino di Tagaste, *Confessioni Libri XIII*, BUR Rizzoli, Milano, 2016, p. 320.

Prima di dare però una risposta dobbiamo procedere per passi elementari e cominciare nel dare una definizione del tempo. Comunemente intendiamo il tempo come un'organizzazione di stati, con i quali interpretiamo vicende umane e naturali, i quali si susseguono mediante l'idea di successione che è a sua volta definita dal concetto di causa ed effetto. Dunque, in quest'ottica, il tempo appare come un susseguirsi di vicende per cui ognuna è preceduta e poi seguita da un'altra. Il compito di organizzare la successione viene svolto dalla memoria. La facoltà mnemonica dell'essere umano ci permette di separare e di ordinare gli stati avvenuti nel passato, e di pensare a un futuro, cioè stati che avverranno. Il futuro viene progettato dalle aspettative che sono anch'esse un prodotto del ricordo. Con ciò pre-vedere significa attendere un avvenire visto sotto l'aspetto di stati passati. In altre parole la memoria ci consente di formulare l'idea del tempo e annesso a questo l'idea di passato e futuro.

Per assurdo stiamo parlando solamente di ciò che non esiste. Il passato, quello che ormai è avvenuto non esiste più, mentre il futuro, quello che deve avvenire, ancora non è esistito. In questo modo abbiamo tralasciato quello che in realtà viviamo, cioè il presente.

Nella ricerca spirituale di Sant'Agostino troviamo uno studio sulla metrica del tempo che esordisce appunto con un'indagine sul presente. Sant'Agostino cerca di determinare nel presente una lunghezza. Decorrendo dall'anno presente e procedendo per il mese, la settimana, il giorno e così via, egli arriva a considerare porzioni particellari di tempo anch'esse dotate, allo stesso modo, di un *prima* e di un *poi*. In conclusione definisce il presente privo di estensione e pertanto sprovvisto di durata[2].

Una risposta del genere è l'esito dell'erronea rappresentazione della natura sensibile del tempo. All'interno della società siamo progressivamente educati ad intendere il tempo

[2] Agostino di Tagaste, *Confessioni Libri XIII*, BUR Rizzoli, Milano, 2016, pp. 320 – 322.

come dispiegato su una diritta linea in relazione al crescere della nostra integrazione sociale. Questa linea viene composta da singoli punti separati tra di loro da un tempo cronometrico e misurabile. In questo modo stiamo pensando a un tempo rappresentato nello spazio come quantità. Se, seguendo il procedimento svolto da Sant'Agostino, suddividessimo la nostra linea temporale all'infinito, in frammenti sempre più piccoli, otterremo infinite particelle di tempo tendenti allo zero (il quale rappresenterebbe il presente che stiamo cercando). Mai però otterremo il "Presente". L'immagine della disposizione spaziale del tempo è un'espressione indispensabile per una società umana organizzata e ne è anche il frutto. Ma quando si vuole indagare logicamente sulla sussistenza stessa del tempo, si incappa spesso in paradossi ed errori. Nascono cosi, per esempio, i sofismi della scuola di Enea. Tra i paradossi di Zenone, quello di Achille e la tartaruga è esemplare.

Ma se il tempo non fosse misurabile? Effettivamente constatiamo ogni giorno l'incommensurabilità del tempo. Assistiamo ripetutamente a eventi che vengono cronometricamente misurati come uguali ma che noi percepiamo con una durata diversa, a volte estremamente diversa. Quando cronometriamo un evento, stiamo valutando l'impiego di una quantità di spazio percorsa dalle lancette di un orologio. Percorso eseguito da un movimento meccanico e ostinato. Ma la durata reale non è omogeneità. Quella durata che noi percepiamo e che è l'unica durata che viviamo non è un oggetto misurabile, non è quantità divisibile ma qualità, intensità. Diventa misurabile solamente quando proiettiamo la propria ombra in una dimensione spaziale.

L'eterogeneità della durata viene rappresentata dalle *impressioni*, che sono qualitative, dei nostri stati di coscienza che non si susseguono separatamente uno dopo l'altro, ma si compenetrano l'uno nell'altro. Con il termine *impressione* si vuole porre l'accento sulla peculiarità degli stati di coscienza che appunto imprimono, lasciano un segno,

un'impronta intensiva in base alla loro qualità, mentre altri stati si verificano, impressionando loro stessi e compenetrandosi tra di loro.

Nell'io, nella coscienza che è appunto la somma mnemonica di tutti quegli stati di coscienza, e nella percezione che la determina troviamo la durata reale che è il nostro presente. Così come diceva Sant'Agostino il presente non ha estensione, perché quest'ultima è una proprietà dello spazio. Bensì il presente è qualità.

Padre di questo pensiero fu Henri Bergson, filosofo francese a cavallo del '900. Egli contesta fortemente la spazializzazione del tempo. Infatti critica Kant, per esempio, per aver valutato il tempo come mezzo omogeneo, allo stesso modo dello spazio. La conseguenza negativa di questa valutazione fu che Kant considerò gli stati della coscienza equivalenti agli stati fisici, ponendoli così ai livelli dei noumeni[3].

La critica bergsoniana più conosciuta prese lo spazio di un intero libro dal nome *Durata e simultaneità* (1922) che fu fortemente contestato dagli scienziati dell'epoca. Henri Bergson scrive questo libro sulla teoria della relatività e sulla concezione della dimensione spazio-tempo, non per negare l'intuizione di Einstein (suo contemporaneo e suo contestatore), ma per fare notare equivoci e formulazioni ambigue sulla teoria[4].

Henri Bergson afferma che la vera realtà del tempo è un fluido continuo e indistinto di stati di coscienza, un'eterogeneità pura, una molteplicità qualitativa, uno sviluppo organico che tuttavia non ha una quantità crescente. I momenti del tempo non sono esterni gli uni agli altri ma si amalgamano, si confondono tra di loro. La competenza dalla memoria è quella di legare tra di loro gli stati di coscienza facendo percepire all'uomo la reale durata. Quest'ultima viene paragonata all'avvolgersi di un gomitolo e

[3] Nella pensiero del filosofo Immanuel Kant il *noumeno* significa "ciò che è pensato" e viene contrapposto al *fenomeno* che invece sta per "ciò che appare". L'uomo può conoscere solamente il *fenomeno*, l'apparenza di ogni cosa, ma non può andare oltre. Al di là del *fenomeno* l'umano può solamente pensare la cosa e formulare su di essa dei giudizi. Per questo motivo viene criticato da Bergson che pensa al tempo non dispiegato nello spazio ma come qualità conoscibile alla coscienza umana. Si veda in proposito H. Bergson, *Saggio sui dati immediati della coscienza*, Raffaello Cortina Editore, Milano, 2002, p. 147.

[4] H. Bergson, *Durata e Simultaneità,* Raffaello Cortina Editore, Milano, 2004.

distinta invece da una collana di perle che rappresenterebbe il tempo spazializzato, il tempo fallace della scienza che è una successione d'istanti tutti eguali tra di loro.

Della durata fuori di noi esiste solo il presente o se si preferisce la *simultaneità*. Le cose esterne cambiano ma i loro momenti si succedono solo in presenza di una coscienza che li ricordi. Al di fuori di noi, in un momento dato, osserviamo un insieme di posizioni simultanee: delle simultaneità precedenti non è rimasto più nulla. Così Henri Bergson sottolinea che «nella coscienza troviamo stati che si succedono senza distinguersi; e, nello spazio, simultaneità che senza succedersi, si distinguono, nel senso che, quando ne appare una, l'altra non c'è più - Al di fuori di noi, esteriorità reciproca senza successione, al nostro interno, successione senza esteriorità reciproca»[5].

[5] H. Bergson, *Saggio sui dati immediati della coscienza*, Raffaello Cortina Editore, Milano, 2002, p. 144.

2. Il tempo della musica

Henri Bergson denuncia spesso nei propri saggi l'insufficienza del linguaggio nell'esprimere ciò che per sua natura è ineffabile. In questo caso, nell'esprimere il concetto del tempo, ricorre spesso a immagini e similitudini che possono comunicare le idee più complesse e refrattarie alla parola. Pertanto la similitudine che ritorna più volte nelle sue opere e che meglio riassume la concezione di durata è la *melodia*. In sintesi Bergson dice che la nostra durata interiore si sviluppa nel tempo come la musica. Anzi la sua stessa essenza è temporale. Una nota non è la stessa se dura pochi attimi o si prolunga considerevolmente: allo stesso modo, la durata muta gli stati di coscienza. Come la musica, il nostro tempo interiore è invisibile ed è definito essenzialmente dal tono emotivo: gli stati che si accavallano come note e si strutturano in alti e bassi , nel loro insieme si organizzano armonicamente creando un'unica sinfonia, ossia la nostra storia personale.

Bergson, attraverso questa metafora, da una parte illustra in modo più limpido il proprio concetto di durata; d'altra parte ci fornisce involontariamente un'osservazione sulla musica.

Non è infatti raro trovare nelle speculazioni filosofiche l'utilizzo di metafore musicali. D'altro canto non è possibile occuparsi di musica senza farsi domande riguardo alla natura del tempo. Questo perché la musica è principalmente un'arte del tempo.

La musica si distingue dalle altre arti poiché non è in grado di costituire un'opera stabile e durevole. Quindi la caratteristica di essere un'arte caduca, effimera, ci costringe a chiamarla arte del tempo. Susanne Langer, filosofa del '900, dichiara, citando un saggio di Basil de Selincourt, che «la musica è una delle forme della durata; essa sospende il

tempo e si offre come ordinario sostituto ed equivalente di questo»[1]. Quest'affermazione fu spesso considerata un'esagerazione.

Susanne Langer considera la creazione artistica un processo che utilizza un espediente virtuale per la realizzazione di un'illusione. Per virtuali intendiamo quei fenomeni o enti che si presentano con aspetti non corrispondenti alla realtà. Uno specchio è uno spazio virtuale perché crea indirettamente uno spazio illusorio, ma non ricrea realmente lo spazio. Pertanto per illusione intendiamo la proiezione in ambito immaginario di elementi che non troveranno corrispondenza nell'ambito della realtà contingente. Quando vediamo l'immagine riflessa nello specchio noi immaginiamo comunque lo spazio dietro di esso, anche se questo non esiste.

Nella musica, però, non troviamo la perfetta integrazione di questi due fattori. L'elemento virtuale che la caratterizza è sicuramente il movimento. Visto quanto detto sopra però, e sapendo che di tempo percepito ne esiste solamente uno, noi non andiamo a creare un'illusione temporale (come creiamo l'illusione spaziale nella pittura) bensì modifichiamo la percezione stessa del tempo.

Ma cosa intendiamo per movimento? Henri Berson lo interpreta come una forma di sintesi mentale. Quando vediamo le cose all'esterno della nostra coscienza (i movimenti nello spazio e allo stesso modo le vibrazioni del suono) le percepiamo con una durata. Questo vuol dire che il movimento ha realtà solo per una coscienza che ricordi il passato, che abbia cioè spessore temporale. Nello spazio fuori di noi non c'è spessore temporale, ma un unico presente. Vale a dire che ci sarà sempre una sola posizione degli stati della materia che compongono il movimento percepito. Solamente per un essere cosciente questi stati si susseguono in modo armonico, costituendo per esempio una

[1] S. Langer, *Sentimento e forma,* Feltrinelli Editore, Milano, 1965, p.130 [ed or. *Feeling and Form. A Theory of Art*, New York, Charles Scribner's Sons, 1953] citato in M. Garda, *Teorie del tempo musicale nella modernità*, in G. Borio, C. Gentili (a cura di), *Storia dei concetti musicali. Armonia, tempo*, Carocci editore, Roma, 2007 pp. 333-334.

melodia, cioè un farsi che impiega un certo tempo. La memoria ci consente di organizzare i singoli stati in un progresso dinamico.

In questo modo percepiamo da prima il suono poi la sua durata e in fine le sue relazioni temporali con altri suoni. La facoltà di ritenzione della nostra memoria permette a questo di imprimere con qualità differenti all'interno della nostra coscienza. Questa qualità si esprime in base alle relazioni che esso ha nei confronti del suono passato catturato dalla memoria. Così, come la nostra coscienza ritorna su un singolo suono, ritornerà anche su una melodia e aspetterà in questo modo la comparsa di nuovi elementi sonori in base alle relazioni memorizzate.

Possiamo parlare di movimento virtuale poiché le impressioni qualitative che partecipano alla vita della nostra coscienza e le impressioni qualitative che costituiscono le relazioni musicali si sviluppano in equivalente forma. Le prime sono provocate da tensioni e distensioni di carattere somatico, fisico, emotivo, mentale e così via, con una certa configurazione; invece le seconde si formano sempre per contrasti e distensioni, ma di natura acustica.

In questo modo possiamo dire che la musica non crea affatto un'illusione di tempo ma solamente un movimento virtuale che, per la natura stessa del movimento, porta alla trasformazione del tempo vissuto.

3. Le forme del tempo musicale: *Linearità* e *non-linearità*

Il movimento di cui abbiamo parlato sopra, come già detto dipende dalle relazioni che la successione dei suoni crea all'interno della musica che ascoltiamo. Come le successioni di stati anche le successioni di suoni creano delle impressioni qualitative, mediante la nostra percezione, all'interno della nostra coscienza. Per questo abbiamo affermato che la musica è un'arte del tempo, cioè essa si dispiega in esso come la pittura nello spazio. Ciononostante dobbiamo precisare che la musica non è solamente divenire, ma possiede anche degli elementi statici, degli elementi che non cambiano mai all'interno di un brano. Lo stesso timbro sonoro di una sonata per pianoforte rimarrà invariato per l'intera composizione. Jonathan Kramer, teorico e compositore contemporaneo, a questo proposito afferma che «il suono è e diviene insieme».

Prenderemo ora in studio il saggio di Jonathan Kramer, *The Time of Music*. L'autore fa parte di quell'insieme di compositori e teorici musicali che dalla seconda metà del '900 in poi hanno svolto un'importante ricerca incentrata sulla fenomenologia del tempo musicale. Questa tipologia di ricerche fu portata dalla divulgazione delle nuove riflessioni di pensatori come Henri Bergson, di cui abbiamo parlato, e Edmund Husserl, il quale affronta la ricerca sulla fenomenologia, anche in riferimento al tempo[1]. In secondo luogo anche altri due fattori hanno portato a queste ricerche, ossia la vicinanza sempre maggiore con le culture non-occidentali e sicuramente le nuove tecnologie. Attraverso *The time of Music* Jonathan Kramer indaga le forme dell'esperienza musicale partendo da due concetti di esistenza temporale della musica: *linearità* e *non-linearità*. Questi due concetti non sono uno l'opposto dell'altro. Anzi, tutta la musica utilizza insieme e in combinazioni sempre diverse le forme di *linearità* e *non-linearità*.

[1] Vedi E. Husserl, *Per la fenomenologia della coscienza interna del tempo*, Franco Angeli, Milano, 2011.

Oltretutto il diverso utilizzo di queste forme del tempo porta alla distinzione di stili musicali e forme compositive.

Definiamo quindi entrambi i concetti con le parole dell'autore. Includiamo all'interno del termine *linearità* quelle «caratteristiche della musica determinate secondo implicazioni che nascono da eventi precedenti del pezzo»[2]. D'altra parte includiamo all'interno del termine *non-linearità* quelle «caratteristiche della musica determinate secondo implicazioni che nascono dai principi o tendenze che regolano un intero pezzo o sezione»[3]. In altre parole la *linearità* è il continuum temporale creato dalla successione di eventi nel quale un evento precedente implica di volta in volta il susseguirsi di nuovi eventi. Mentre la *non-linearità* è, invece, quel continuum temporale che risulta dai principi permanenti che governano un intero pezzo o una sezione. Perciò dobbiamo interpretare entrambi i termini come due forme del tempo che convivono all'interno della musica.

Una metafora proposta da Jonathan Kramer ci fa meglio intendere l'importanza dei due termini. Egli infatti ci dice che *linearita* e *non-linearità* corrispondono approssimativamente alla distinzione filosofica tra *divenire* ed *essere*[4]. Il concetto di divenire lo troviamo in particola modo nel pensiero lineare dell'occidente che ha origine dall'antica Grecia e culmina con la cultura giudaico-cristiana. Per di più fu determinante nella filosofia e nella scienza dell'epoca moderna. D'altra parte il concetto di essere ebbe la sua massima affermazione nelle culture orientali, come lo Zen nel Buddismo.

La musica, come le altre arti, nel corso della storia ha sempre rispecchiato la cultura dell'epoca. Ecco perché ci rimarrà più semplice intendere il concetto di divenire utilizzato nell'arte musicale. Quando però parliamo di divenire nella musica, quindi di *linearità*, è importante non confondere questo concetto con continuità. Infatti la

[2] J. D. Kramer, *The Time of Music*, Schirmer Book, New York, 1988, p. 20.
[3] *Ibidem.*
[4] *Ivi*, p. 16.

linearità può essere sia continua sia discontinua. Approfondiamo quindi il concetto di *linearità*.

Il linguaggio musicale occidentale si è costituito nel corso della storia per via del concetto di divenire. Nella nostra cultura infatti si è acquisito con il tempo quel senso di *linearità*, che in altre parole è la concezione aristotelica di causa ed effetto. Per questo il sistema tonale è la forma più autentica di *linearità* poiché ogni nota è la causa di un solo effetto, cioè una nota precedente (così, come nel disegnare una linea, ci servono almeno due punti). Più dettagliatamente il sistema tonale soddisfa anche il concetto di continuità e per questo percepiamo in esso un senso di direzionalità finalizzata. Jonathan Kramer la chiama appunto *linearità finalizzata*[5]. In altre parole, le voci che si muovono in una composizione tonale seguono una direzione verso un obiettivo. Il sistema è caratterizzato da complesse gerarchie tra i suoni. Se consideriamo una tonalità qualsiasi, essa presenta delle forti relazioni che si trovano al suo interno, tra i gradi della tonalità, e al suo esterno, con le altre tonalità. Il movimento dei suoni per mezzo di questo sistema ci permette di muovere le voci attraverso queste relazioni. La conclusione nella tonica e quindi il ritorno alla tonalità di partenza sono le leggi fondamentali di questo sistema. Per questo il linguaggio tonale presenta una forte continuità finalizzata. Oltretutto noi troviamo confortevole l'ascolto di questo sistema musicale per diverse ragioni. In primo luogo apprendiamo l'abilità di ascolto di questa musica in età molto giovane. Non solo, questo genere di ascolto è soddisfacente poiché lo stesso tipo di *linearità* contraddistingue in egual modo la nostra vita da occidentali.

Così, come nel linguaggio tonale, troviamo anche una preponderanza di *linearità finalizzata* nella musica non tonale. Esempio chiaro è la dodecafonia e successivamente la serialità integrale. In questo tipo di sistema riscontriamo una medesima gerarchia

[5] *Ivi*, pp. 25-39.

nella condotta delle parti. Pertanto troviamo processi direzionali che si occupano del movimento delle voci, più in generale del suono e dei suoi parametri nella serialità integrale.

Ugualmente riscontriamo una *linearità finalizzata* in alcune composizioni che non presentano alcun sistema di gerarchia. Le finalità di queste musiche nascono nel contesto della composizione stessa. In altre parole è come se creassero un proprio sistema funzionale solo all'interno della composizione che lo contiene. Queste composizioni forniscono un senso di continuità, coerenza e così degli obiettivi. La finalità non è più raggiunta attraverso le gerarchie di un linguaggio comune prestabilito ma attraverso delle relazioni che il suono e i suoi parametri creano all'interno del brano. L'obiettivo sarà infine raggiunto attraverso, o un procedimento di reiterazione, o attraverso un processo di enfasi delle relazioni stabilite in precedenza.

Troviamo invece una *linearità non-finalizzata* nelle composizioni (in particolar modo del Novecento ma addirittura, anche se più raramente, dei secoli precedenti) dove non esiste una chiara direzionalità. Infatti, in alcune di queste musiche possiamo addirittura trovare più orientamenti nel discorso musicale. Jonathan Kramer chiama *multi-direzionato*[6] questo tipo di discorso temporale. Infatti riscontriamo comunque un certo tipo di movimento affine alla *linearità finalizzata*, ma a differenza di quest'ultima l'obiettivo rimane equivocabile. Pertanto non riscontriamo una singola direzione ma una molteplicità di orientamenti verso un'altrettanta molteplicità di obiettivi. «Il tempo *multi-direzionato* potrebbe paragonarsi a un campo vettoriale multidimensionale; così come il tempo lineare finalizzato è analogo a una linea retta». Anche nella musica tonale dell'ottocento si possono riscontrare esempi di tempo multi-direzionale. In

[6] *Ivi*, pp. 39-40.

particolar modo, tra questi, Jonathan Kramer ci suggerisce il quartetto op 135 di Beethoven.

Nel primo movimento di quest'opera, alla battuta 10 (appena all'inizio), viene portata al termine una cadenza finale. Beethoven, in questo modo, ricopre di multi-direzionalità il significato di queste battute: considerando il tempo assoluto del brano queste prime battute costituisco l'inizio; considerando invece il significato temporale di una cadenza finale il brano termina nella battuta 10[7].

Questo tipo di multi-direzionalità è discontinuo poiché rompe il tempo lineare e lo riordina secondo il proprio svolgimento interno. Questo tipo di discontinuità è particolarmente in rilievo nel *moment time*[8], con riferimento alla *Momentform* di Stockhausen. Con questo intendiamo composizioni costituite da più sezioni autosufficienti (moment) collegate tra di loro da discontinuità. Nel senso che non esistono relazioni tra un moment e un moment successivo e le successioni tra di loro sembrano arbitrarie. Per questo possiamo dire che una composizione in «moment time» non inizia ma semplicemente parte come se fosse già iniziata; non finisce, ma semplicemente si ferma. In questo modo non avremo una chiara direzionalità ma più andamenti tra di loro separati e diretti verso obiettivi differenti tra di loro. Un esempio estremo della moment time è la *mobil form* in cui i moment non sembrano soltanto ma sono realmente arbitrari. L'esecutore, nel momento della performance, sceglie le successioni dei singoli moment, per cui la composizione risulterà sempre diversa (un esempio è *Momente* di Stockhausen).

[7] Si veda in particolar modo *ivi*, Chapter 5. ANALYTIC INTERLUDE: *Linearity, Meter, and Rhythm in Beethove's String Quartet in F Magior, Opus 135, First Movement*, pp. 123-136.
[8] A proposito della *multi-direzionalità* e della *moment time* si veda *ivi*, Chapter 8. DISCONTINUITY AND THE MOMENT, pp. 201-220.

Fin qui abbiamo visto gli aspetti che, attraverso una composizione musicale, divengono. Tutti gli altri aspetti della composizione intera o di una sezione che rimangono statici dall'inizio alla fine, come già detto, fanno parte del tempo *non-lineare*.

Gli elementi non lineari all'interno di una composizione sono pertanto quegli elementi che rimangono costanti. Possono trovarsi nella tessitura, nel materiale motivico, nel disegno melodico e così via. Un esempio proposto da Jonathan Kramer della *non-linearità* nella musica tonale è il *Preludio* in do maggiore del primo volume del *Clavicembalo ben temperato* di Johann Sebastian Bach, in cui il disegno ritmico rimane invariato per tutta la durata del brano[9].

Ora, seguendo il ragionamento proposto da Jonathan Kramer arriveremo al baricentro del nostro lavoro. Procedendo infatti per le varie forme di *linearità* esposte dall'autore, raggiungeremo il punto in cui la forma non-lineare del tempo predomina all'interno di una composizione. Definisce così *verticale* una musica che evoca una forma di tempo priva di movimento, pertanto di *linearità* e che quindi si costituisce interamente fra strati simultanei di suono. Tratteremo nei prossimi capitoli della *musica verticale*[10] e della percezione che questo tipo di tempo musicale evoca a un ascoltatore.

[9] *Ivi*, pp. 40-42.
[10] *Ivi*, pp. 54-57.

4. La musica verticale

Ricollegandoci al moment che abbiamo visto nel capito precedente, Jonathan Kramer definisce la musica verticale con le seguenti parole: «Quando il moment diventa il pezzo, la discontinuità scompare a favore della coerenza totale, possibilmente invariabile. Le composizioni sono state realizzate in modo da essere temporalmente indifferenziate nella loro interezza. Esse mancano di "frasi" (proprio come sono carenti di successioni, di direzionalità, di movimento e di velocità contrastanti) perché la conclusione di frasi interromperebbe la continuità temporale. Il risultato è un unico "presente" esteso ad una durata enorme, un "adesso" potenzialmente infinito che ciò nonostante si percepisce come un attimo. Nella musica senza frasi, senza articolazione temporale, dotata di coerenza totale, qualsiasi struttura presente esiste fra stati simultanei di suono, non fra gesti successivi. Definisco il concetto di tempo evocato da tale musica come "verticale"»[1].

Con la musica verticale siamo arrivati a parlare del punto centrale di questo lavoro. Questo genere di tempo evocato dalla musica verticale nega al suo interno il movimento qualitativo del suono. Elimina le relazioni di altezze e di durate, annulla così la successione e la direzionalità verso un obiettivo specifico. In questo modo abbiamo una musica che mette alle strette la sua stessa sussistenza nel tempo.

Prima, avevamo visto come la musica, attraverso le tensioni acustiche interne, ci provocava un'illusione di movimento, di direzione verso un qualcosa, quindi, alla trasformazione del tempo vissuto. Ora, la musica verticale elimina l'idea di passato e di futuro creando un eterno presente. Infatti il passato sarà vinto dall'immutabilità, quindi dalla *non-linearità*, per cui sarà sempre fuso e così inconfondibile nel presente. La

[1] *Ivi*, p. 55.

stessa cosa il futuro. In mancanza di relazioni passate possedute dalla memoria, l'aspettazione non può che diventare lo stesso presente.

Un pezzo di musica concepito verticalmente non deve iniziare, ma semplicemente partire, non deve nemmeno concludersi, ma terminare. Non presenta un climax, e quindi un punto culminante, non presenta contrasti né alcun tipo di accumulo o sprigionamento di tensione. Una musica del genere può essere definita come un «unico evento di grandi dimensioni»[2]. Così «un pezzo concepito verticalmente definisce sin dall'inizio quello che sarà il confine del suo mondo sonoro durante l'esecuzione, e resta nei limiti che ha scelto»[3]. Pertanto potremmo ascoltare questo genere di musica per un periodo di tempo breve, oppure lungo, comunque sia avremmo ascoltato allo stesso modo l'intera composizione.

La musica verticale non è però priva di struttura. Mentre una composizione *lineare* acquisisce una forma attraverso il suo continuum temporale, la composizione verticale, *non-lineare*, si predispone verticalmente. Quindi se la prima si stabilisce tra eventi successivi la seconda tra eventi sempre presenti e simultanei.

Esistono molte composizioni che, seppur non utilizzando in maniera strettamente fedele la definizione che né da Kramer, impiegano comunque una forma molto simile di tempo verticale. Un esempio è la corrente di compositori che hanno origine dalle musiche di Erik Satie. Di questo compositore abbiamo un pezzo esemplare, *Vexations*. Questo brano è composto di 152 note che devono essere ripetute dall'esecutore per 840 volte. Dopo che sarà suonata per la prima volta, quella cellula di 152 note rappresenterà per altre 839 volte il nostro eterno presente.

Un'altra composizione è *As Slow as Possible* di John Cage. Questa composizione è una delle più lunghe in assoluto. Iniziata nel 2001, con una durata di 639 anni, terminerà nel

[2] *Ibidem*.
[3] *Ibidem*.

2640. Il pezzo per organo consiste nella successione di accordi i quali durano ciascuno venti anni circa. All'interno di questa durata la percezione di un ascoltatore è assolutamente non lineare. Solo nel momento del cambiamento (il prossimo avverrà fra tre anni) sarà annullata la percezione verticale dell'opera[4].

Rimanendo sempre sullo stesso filone di compositori, possiamo vedere che, anche la musica minimalistica ci propone un tipo di ascolto verticale. Anche se nelle composizioni di questa corrente possiamo scorgere come elemento importante il processo, quest'ultimo viene spesso proposto ai limiti della percezione. Le tecniche compositive di alcuni dei minimalisti, in maggior modo quelle di Steve Reich e di Philip Glass, consistono proprio nel nascondere il più possibile lo sviluppo della composizione. In questo modo, anche se non adoperano fedelmente un tempo verticale, provocano comunque un'esperienza verticale all'ascoltatore.

Al termine di questo filone troviamo la musica ambient che riprende il concetto di *musica d'arredamento* di Erik Satie. Uno dei primi e massimi esponenti di questa musica fu Brian Eno. Egli, con *Ambient I. Music for airport* (1978), propone una musica che pur contenendo delle frasi melodiche, queste si ripetono incessantemente senza proporre alcun tipo di direzione, quindi senza aver alcuna finalità. I quattro brani, all'interno di *Ambient I* ci danno proprio un senso di mancanza di sviluppo temporale e così ci fanno percepire un tempo verticale. E, come proponeva la *musica d'arredamento,* si potrebbe ascoltare l'intera opera, ignorarla oppure ascoltarla

[4] Nel sito http://www.aslsp.org si può ascoltare in streaming la composizione in esecuzione nella chiesa di Sankt-Burchard nella città di Halberstadt in Germania. In realtà la durata dell'esecuzione non fu scelta da John Cage ma risale ad una conferenza di filosofi e musicisti del 1997. Questi discussero sulle implicazioni dell'istruzione di Cage che lo spinsero a comporre questo brano. La durata di 639 anni fu scelta per commemorare la prima istallazione permanente di un organo (1361), avvenuta proprio 639 anni prima della data in cui si era prevista originalmente l'inizio dell'esecuzione (nel 2000). L'organo viene suonato attraverso dei pesi posizionati sulla tastiera e controllati da un meccanismo elettronico.

solamente a tratti come quando si nota, o no, l'arredamento di una sala d'attesa (per esempio di un aeroporto)[5].

Comunque sia, nella cultura occidentale, rimane ancora difficile trovare una composizione che sia fedele alla musica verticale se non in opere che possiedo elementi extramusicali. Esemplari sono gli organi del vento o del mare, che sono sculture le quali emettono dei suoni quando vengono eccitati rispettivamente da vento o da acqua[6]. Le composizioni che creano queste opere d'arte sono indubbiamente senza inizio o conclusione ma partono e finiscono in base alla scelta dell'ascoltatore (queste continuano incessantemente a suonare); il suono si muove ma solo all'interno del suo mondo sonoro che è stabilito dalla scultura e che è definito fin dal suo principio. In questi casi abbiamo proprio un ascolto verticale anche perché verticale è la peculiarità dell'esperienza di un'opera d'arte visiva come la scultura.

Ora, ritornando all'obiettivo principale di questo lavoro - ossia la realizzazione di un'opera d'arte musicale senza tempo e quindi la realizzazione di una musica verticale (che non abbia però elementi extramusicali) - dobbiamo affrontare un'ultima problematica. Eliminando qualsiasi genere di movimento significativo e rendendo la musica un eterno presente, quali saranno le reazioni percettive all'interno della coscienza di un ascoltatore? Quale sarà il vero tempo suscitato all'ascolto di una musica senza tempo?

[5] Vedi anche la *musica d'ascensore.* Questo genere di musica viene anche nominato Muzak (dall'azienda musicale Muzak Holdings) e sta ad indicare le musiche di sottofondo dei centri commerciali, dei supermercati, delle navi da crociare, hotel, uffici e anche per esempio degli aeroporti.

[6] Un famosissimo organo da vento è la scultura realizzata dalla coppia di artisti-architetti Anna Liu e Mike Tonkin che si trova nella vallata della città di Burnley in Inghilterra. Un organo d'acqua è per esempio l'*Organo marino a Zadar*, in Croazia, dell'architetto Nikola Bašić. Un'altra interessante opera si trova a Dresda, in Germania, dove il sistema di grondaie e i tubi di scarico di un palazzo creano il suono con la pioggia. Famoso è anche la scultura sonora di Bill Fontana, che prende il movimento di un ponte, causato principalmente dal vento, e lo utilizza come segnale di controllo per un live electronics.

5. La percezione verticale e l'"interattività coscienziale"

Per determinare il tipo di percezione verticale dobbiamo in primo luogo sottolineare che la dimensione lineare della musica non è esclusiva dell'esperienza dell'ascolto. Come avevamo visto in precedenza, la musica, come le altre arti, rispecchia nel corso della storia le credenze culturali della propria civiltà ed epoca. La cultura occidentale è caratterizzata da un pensiero lineare. Culture orientali, africane e alcune sud americane non possiedono lo stesso tipo di pensiero. Perciò troviamo musiche strutturate su forme temporali non-lineari.

Se vediamo la musica dell'isola indonesiana del Bali[1], queste possiedono una struttura circolare senza fine, non direzionata verso qualcosa e senza alcun climax. Non c'è alcun accumulo, non viene costituito niente ma è come se già avessimo tutto fin dall'inizio. Non c'è una durata prestabilita ma la musica finisce semplicemente al termine dell'esecuzione senza alcun gesto conclusivo. Non è importante lo sviluppo, la quantità di tempo utilizzata per determinate parti che si succedono, ma è importante la qualità di quello che suonano che è come un unico momento circolare ed esteso. Come ci dice l'antropologo statunitense Clifford Geertz, proprio a riguardo di queste musiche: «Non ti dicono quanto tempo è; ti dicono che tipo di tempo è»[2]. Questo genere di pensiero musicale si riscontra anche nel particolare calendario balinese, il *Pawukon*[3].

Da un altro punto di vista però la nostra civiltà lineare non è del tutto ignara di un'esperienza verticale nell'arte. In tutta l'arte figurativa, dalla pittura alla scultura fino l'architettura, abbiamo un tipo di esperienza del tempo non-lineare. Non ci preoccupiamo del tempo che dura la nostra esperienza ma, solamente di quello che

[1] Si tratta della musica eseguita dal *gamelan* (orchestra di strumenti musicali indonesiani) che sentì anche Claude Debussy nell'*esposizione universale* di Parigi del 1889 e che lo colpì particolarmente.
[2] C Geertz, *Person, Time, and Conduct in Bali*, in *International of Cultures*, Basic Books, New York, 1973, p. 393.
[3] Il Pawukon ha come grande particolarità quella di non numerare il passaggio degli anni ma, all'interno dei suoi 210 giorni annuali, regola solo i giorni di festa.

vediamo. Per questa similitudine di esperienze Jonathan Kramer scriverà: «L'ascolto di una composizione "verticale" può essere come guardare una scultura. Quando esaminiamo la scultura determiniamo da noi stessi la misura della nostra esperienza: siamo liberi di camminare intorno all'oggetto, osservarlo da diverse angolazioni, concentrarci su alcuni dettagli, vedere altri dettagli in relazione l'uno con l'altro, retrocedere e guardare l'opera nel suo complesso, contemplare il rapporto fra l'opera e lo spazio nel quale la vediamo , chiudere gli occhi e ricordare, lasciare la stanza quando lo desideriamo e ritornare per ulteriori osservazioni. Nonostante la selettività individuale del processo di osservazione, nessuno potrebbe affermare che non abbiamo dato uno sguardo attento all'intera scultura (sebbene possiamo non aver notato qualche particolare). Per ciascuno di noi la sequenza temporale delle posizioni di osservazioni è stata unica. Il tempo trascorso con la scultura è tempo strutturato, ma la struttura è posta da noi , influenzata dall'oggetto, dal suo ambiente, da altri spettatori, e dal nostro umore e gusto. In modo analogo, la musica verticale semplicemente è: possiamo ascoltarla o ignorarla. Se ascoltiamo solo parte dell'esecuzione abbiamo anche ascoltato l'intero pezzo, perché sappiamo che esso non subirà mai modifiche. Siamo liberi di concentrarci sui dettagli o sull'intera opera. Al pari della scultura, il pezzo non ha una differenziazione temporale interna che possa ostacolarne la percezione che desideriamo averne»[4].

Con questo passo di Jonathan Kramer osserviamo che, nella realtà dei fatti, la percezione della musica verticale non porta all'abbattimento del continuum temporale. Pertanto una composizione che utilizza un tempo verticale non sarà mai una musica senza tempo, poiché il tempo viene dato da noi stessi. La musica verticale ci porta solo a confrontarci con un momento musicale eterno e quindi a un livello di esperienza

[4] J. D. Kramer, *The Time of Music*, Shirmer Book, New York, 1988, p. 57.

diverso dalla comune musica lineare. Infatti noi non perderemmo mai il contatto con la nostra reale coscienza, perché mai ci distaccheremo completamente dalla realtà esterna, quindi dallo spazio che ci circonda, dalla modalità di ascolto e, come diceva Kramer, dal nostro umore e gusto. Quello che viene realmente immobilizzato è il momento musicale e solo esso manca di passato e futuro contrariamente alla nostra coscienza, la nostra reale durata.

Il musicologo Thomas Clifton, nel suo libro *Music as Heard: A study in Applied Phenomenology* (riportato in particolare evidenza in *The Time of Music*)[5], nega fortemente la creazione di un'esperienza senza tempo provocata da una stasi musicale. Egli suggerisce l'ascolto di un singolo tono, prolungato e senza alcun tipo di cambiamento. Quando assistiamo a tal evento sonoro, all'interno della nostra coscienza prenderanno luogo uno «sciame di attività»[6]. Infatti la nostra attenzione vagherà all'interno del suono stesso, considerandolo da svariate parti, per esempio dall'altezza, dall'intensità o dal timbro. In analogia a quanto dice Jonathan Kramer con la metafora della scultura, l'ascoltatore, in base alla propria coscienza, determinerà egli stesso la propria esperienza dirigendo l'attenzione verso svariate forme della struttura verticale della musica. In questo modo egli stesso determinerà la durata della propria esperienza.

Per questo motivo, ai fini di questo lavoro, coniamo il termine di "interattività *coscienziale*". Un sistema interattivo consente l'intervento di un gesto che controlla, modifica, attiva o disattiva dei processi che costituiscono la sua esecuzione temporale. Nell'interattività più comune il gesto è il movimento per di più fisico (come il movimento di una mano, di un braccio o del capo) che provoca una precisa reazione prestabilita all'interno dei confini del sistema. Sulla musica elettronica, e più precisamente sulle installazioni musicali, esistono molte opere interattive. Qui, a volte è

[5] *Ivi*, p. 377-378.
[6] T. Clifton, *Music as Heard: A study in Applied Phenomenology*,New Haven: Yale University Press, 1983, p. 97.

il movimento, la posizione di uno spettatore, la quantità di spettatori o altro a modificare il suono o solo alcuni aspetti di questo. Nelle composizioni per live electronics troviamo molto spesso la costruzione di strumenti interattivi per la gestione del suono (proprio come gli strumenti tradizionali, i quali possono essere anch'essi definiti tali). Questo tipo d'interattività è diretta a differenza di un'*interattività coscienziale*. Infatti quest'ultima la troviamo nell'ascolto della musica verticale ed è eseguita indirettamente dalla nostra coscienza, per questo la chiamiamo coscienziale.

Come abbiamo più volte ripetuto, quando ascoltiamo un brano musicale che segue un tipo di tempo lineare, il nostro tempo coscienziale viene in qualche modo sostituito dal tempo musicale. Questo perché è evocato da impressioni qualitative che sono costituite dalle relazioni sonore all'interno della musica. Nel caso contrario, invece, quando ascoltiamo un brano musicale che segue un tempo di tipo verticale, succede esattamente l'opposto. In questo caso non siamo più investiti da un nuovo tempo, poiché la musica sarà priva di successione e sviluppo nelle impressioni qualitative, quindi di movimento virtuale. Essendo però, la musica verticale, un'unica impressione, un unico esteso presente, e non potendo attraverso questa interrompere lo scorrere del continuum temporale, saremo noi, attraverso gli stati della nostra coscienza, a equipaggiare questa musica di temporalità.

Come con l'interattività più diretta modifichiamo il decorso temporale della composizione, con l'*interattività coscienziale* diamo e modifichiamo la durata della nostra esperienza musicale.

6. L'aura

Con la musica verticale quindi non creeremo più un movimento virtuale, ma un'apparenza di stasi indirizzata alla percezione intima della coscienza dell'ascoltatore. Tale percezione non assomiglierà alla percezione naturale della coscienza. La musica verticale pone una struttura sonora che ispira i singoli dati di coscienza percepiti e libera cosi la percezione da tutti gli scopi pratici. La qualità estetica è come nelle altre arti lo *Schein*, come lo definisce Schiller, ossia la capacità di percezione non pratica, non funzionale se non all'apparenza provocata dall'oggetto artistico. L'arte, essendo pertanto di una sostanza apparente esiste solo per il senso o l'immaginazione della coscienza che li percepisce. Essendo la musica verticale un tipo di forma d'arte che non provoca, è importante capire che l'ascolto funzionale di questo genere d'arte deve avvenire con *devozione*.

Su degli scritti associati al saggio *L'opera d'arte nell'epoca della sua riproducibilità tecnica*, Walter Benjamin parla di due metodi di avvicinamento tra l'opera d'arte e il fruitore. Da una parte c'è un pubblico che cerca nell'arte la distrazione per cui fa sprofondare dentro di se l'opera. Dalla parte opposta abbiamo invece un pubblico che si accosta all'arte con raccoglimento e devozione, per cui questo sprofonda in essa[1].

Perché ci sia una fruizione del secondo genere, l'opera d'arte deve essere in possesso dell'*aura*. L'aura è l'*hic et nunc* dell'opera d'arte, cioè la sua esistenza unica e irripetibile nel luogo in cui si trova. Stiamo pertanto parlando di autenticità e quindi l'aura si sottrae dalla riproducibilità tecnica e non soltanto a quella tecnica. L'aura è un «singolare intreccio di spazio e di tempo: l'apparizione unica di una lontananza, per

[1] W. Benjamin, *L'opera d'arte nell'epoca della sua riproducibilità tecnica*, in A. Pinotti, A. Somaini (a cura di), *Aura e choc*, Einaudi Editore, Torino, 2012, pp. 53-54.

quanto essa possa essere vicina»[2]. E solo mediante la devozione il fruitore può diminuire questa distanza[3].

Finché la musica verticale rimane un pezzo di musica statica, invariata nel tempo, essa si stabilisce in lontananza rispetto all'ascoltatore. Nel momento in cui essa prende vita all'interno della nostra coscienza, tale lontananza viene a meno fornendo di vita temporale la musica verticale. Questo tipo di approccio, che rivela la vera fisionomia di questo genere di musica è esattamente un ascolto autentico. Infatti, attraverso l'*interazione coscienziale* ogni coscienza diventa l'unico luogo di esistenza della musica verticale. Attraverso un ascolto devoto possiamo pertanto dispiegare quello che sembrerebbe una monotonia in un'esperienza temporale della percezione.

[2] *Ivi*, p. 21.
[3] Per approfondimento sul concetto di *aura* vedi in particolare *ivi*, pp. 19-24.

7. Moment V – L'installazione web

Ma come realizzare un'opera d'arte che rispetti il più idealmente le caratteristiche della musica verticale che abbiamo appena visto?

Nel marzo del 2016 pubblicai per la prima volta un disco sulla musica verticale, che prende appunto il nome di *Musica Verticale*. Questo disco è costituito di quattro tracce che rappresentano quattro diversi momenti sonori. Le tracce sono appunto nominate *Moment I*, *Moment II*, *Moment III*, *Moment IV* e durano all'incirca 10 minuti ciascuna[1]. Questo disco si presenta in sostanza come uno studio sulla composizione della musica verticale. L'obiettivo che sta alla base di queste quattro tracce è quello di creare un movimento che non venga percepito tale, e infatti un movimento che non caratterizzi l'orizzontalità della singola traccia audio. La vera drammaturgia delle tracce non risiede più nello sviluppo temporale, che è stato eliminato, ma nella stratificazione verticale del suono. Infatti ogni complesso timbro sonoro, che si percepisce per l'intera singola traccia, è stato accuratamente montato per via della sovrapposizione di più suoni. Ma prima ancora della sovrapposizione, i singoli suoni che costituiscono il timbro finale vengono elaborati per via di un particolare algoritmo. Parleremo di quest'algoritmo più dettagliatamente in seguito, qui diremo soltanto che ci permette di "congelare" il suono e renderlo in movimento come se respirasse, senza generare orizzontalità. *Moment I* si costituisce verticalmente per via di suoni di Tam-tam, gong, timpani e pianoforte; *Moment II* è costituito per strati di note cantate; *Moment III* dalla combinazione timbrica del sax alto e del sax tenore; in fine *Moment IV* è costituito dal suono delle corde di una chitarra acustica. Se pur riescono a esaudire una struttura per lo più verticale comunque le quattro tracce sono costituite da un inizio e da una fine. Ciascuna inizia per via di un lento fade-in e dopo circa dieci minuti termina con un altrettanto lento fade-out.

[1] Il disco *Musica Verticale* è pubblicato indipendentemente nel sito www.bandcamp.com, all'indirizzo https://musicaverticale.bandcamp.com/releases .

Purtroppo questi elementi sono inevitabili in una composizione che risiede nel tempo di una traccia audio. Per questo l'ascoltatore non deciderà il tempo della sua fruizione ma sarà il tempo della traccia audio a determinarla. Pertanto, al termine della traccia sarà completamente annullato il tempo verticale dell'opera a favore di un tempo orizzontale.

Attraverso *Moment V*, che è appunto il seguito delle quattro tracce che abbiamo visto, mi sono proposto di eliminare questi elementi, d'inizio e di fine, non utilizzando più il contenitore della traccia, e quindi realizzare una musica verticale ideale. L'obiettivo è quindi quello di utilizzare l'elaborazione audio incessantemente e quello di permettere al fruitore di accedere a questa elaborazione e lasciarla quando è lui stesso a deciderlo. In questo modo si vuole proprio ricreare le circostanze che determinano l'esperienza di una scultura in un museo. Per questo motivo ho scelto uno spazio virtuale web, dove l'elaborazione audio avviene perpetuamente e l'ascoltatore accede, resta e abbandona il proprio ascolto nei tempi che determina egli stesso. Il tipo di elaborazione rimarrà quello dei quattro *moment* successivi ma questa volta non saranno più preregistrati e fissati all'interno di una traccia audio ma avverranno continuatamente e in tempo reale all'interno di uno spazio web. Ecco perché parliamo d'installazione web.

Ora, per la realizzazione di questo lavoro ho incontrato diverse problematiche. La prima è stata la realizzazione dell'elaborazione all'interno dello spazio web. Nel tempo ho provato diversi linguaggi di programmazione musicale che difficilmente lavorano all'interno del web (per esempio *pure data*, *supercollider*, mentre con *max/MSP*, che è il linguaggio con cui nasce l'elaborazione audio dei primi quattro *moment*, è in pratica impossibile lavorare con il web). In fine ho scoperto l'esistenza di *Web Audio API* che si occupa della gestione dell'audio attraverso il linguaggio *javascript*, proprio all'interno di una pagina web.

In seguito, un'altra problematica fu quella di creare un'elaborazione audio indipendente dai *clients* (utenti) dello spazio web e quindi un'elaborazione unica per un infinito numero di utenti. Proprio come creare una scultura in un determinato luogo. Quando i visitatori la vedono in un certo momento, condividono lo stato di quella scultura (non la loro esperienza), che in quel momento è uguale per tutti. Nello stesso modo l'installazione web sarà, in un determinato momento, uguale per ogni utente che accede al sito internet (suo luogo virtuale) in quell'istante. In seguito questo genere di complicazioni mi ha portato ad altre problematiche ancora legate alla gestione del *server* web. Anche se vedremo più dettagliatamente in seguito, i dati dell'elaborazione audio provengono tutti dal *server*, quindi da un *database* che viene costantemente aggiornato da un'altra pagina web, attraverso una tecnica chiamata *cronjob*. Questo per dire che l'elaborazione non verrà gestita dalla pagina web, ma quest'ultima ha il solo scopo di accedere e utilizzare i dati generati al livello del *server* web.

Nei capitoli successivi vedremo la realizzazione dell'installazione web e la risoluzione delle varie problematiche. Vedremo pertanto l'utilizzo di *Web Audio API* per la gestione dell'audio nel lato *client* della pagina web e così anche l'algoritmo di elaborazione audio; vedremo poi *php*, *sql*, *cronjob* e la struttura dei *database* per il lato *server* della pagina web.

Parte II

8. Cenni funzionamento web (*client-server*)

Ai fini di una maggiore comprensione della realizzazione dell'installazione è bene capire la struttura ed il funzionamento del web. Il *world wide web*, abbreviato *web* e che noi conosciamo più comunemente con la sigla *www* è uno dei principali servizi di internet. Quest'ultimo è una rete virtuale che permette la connessione di una grande quantità di vari dispositivi e terminali in tutto il mondo. La connessione che sta alla base di ogni spazio web (che possiamo chiamare anche sito internet) è quella tra un *server web* ed un *client*. Ogni sito internet risiede all'interno di un computer costantemente connesso alla rete, che prende appunto il nome di *server*. All'interno di questo noi abbiamo dei file testuali con estensioni diverse che descrivono le pagine web. Quasi in ogni *server* avremo un file con estensione .html, che è un linguaggio di markup e che quindi gestisce la struttura di rappresentazione della pagina; un file con estensione .css, che gestisce la formattazione dei file html; potremmo avere file *javascript* per l'interazione e la manipolazione dei file html e css; file con estensione .png, .jpg (quindi immagini), oppure video e file audio.

Il *server* in questo caso funzione proprio come un hard disk esterno: è un contenitore di file ma non li legge. Solo quando colleghiamo il nostro hard disk con un cavo usb ad un computer, possiamo interpretare i nostri file, residenti all'interno, come immagini, video, musica e programmi. La stessa cosa succede nel web ma, invece di utilizzare un cavo usb, viene utilizzata una connessione internet per collegarsi al *server*. Questo collegamento avviene per via del *browser* web (firefox, chrome, safari, internet explorer, opera e così via) che è l'interprete dei file residenti all'interno del *server* e svolge la funzione del *client*. In questo modo l'elaborazione dei file html, css fatta per

esempio attraverso *javascript* avviene al livello del *client*, quindi all'interno del *browser* web pertanto unica per ogni visitatore dello spazio web.

Ma non solo, alcune elaborazioni di queste pagine possono avvenire prima che raggiunga il *client* e quindi al livello del *server web*. Sono quindi programmi che vengono interpretati dallo stesso *server*. I programmi scritti in *php*, *Java*, *phyton* ecc sono di questo genere e sono destinati principalmente alla gestione dei *database* e di quei dati che cambiano e rimangono comuni per tutti i diversi *client*. Un esempio più banale è l'ora e la data che viene mostrata in una pagina web, oppure il numero di utenti che sta visitando la pagina.

Ricapitolando quindi diciamo che il web si stabilisce mediante la connessione internet tra un *server* ed un *client*.

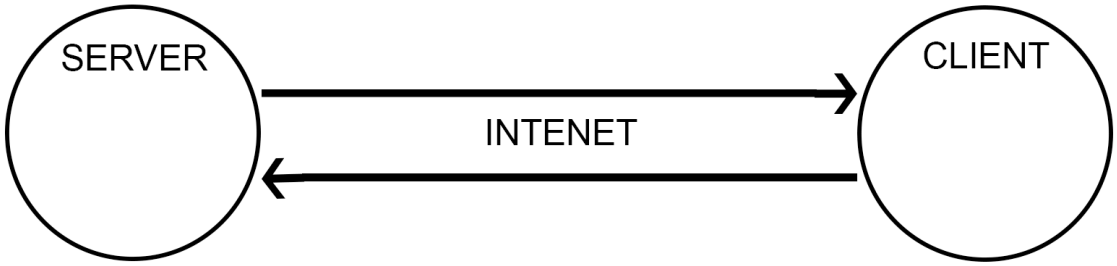

fig.1 – Web

Il *server* contiene tutti i file che consentono la generazione della pagina web, per quanto riguarda lo stile della pagine, quindi la parte visuale e tutti i programmi che consentono l'interazione e la manipolazione della pagina web, per esempio la possibilità di cliccare un bottone, scorrere all'interno della pagina e così via. Resteranno però solamente file testuali finché un *browser* web non li interpreta. Il *client*, digitando il sito internet a cui vuole accedere attraverso il *browser*, richiede al *server* i file che servono alla

generazione della pagina web. Il *server* risponde inviandoli attraverso la connessione internet e il *client* li elabora e li interpreta generando il sito internet desiderato.

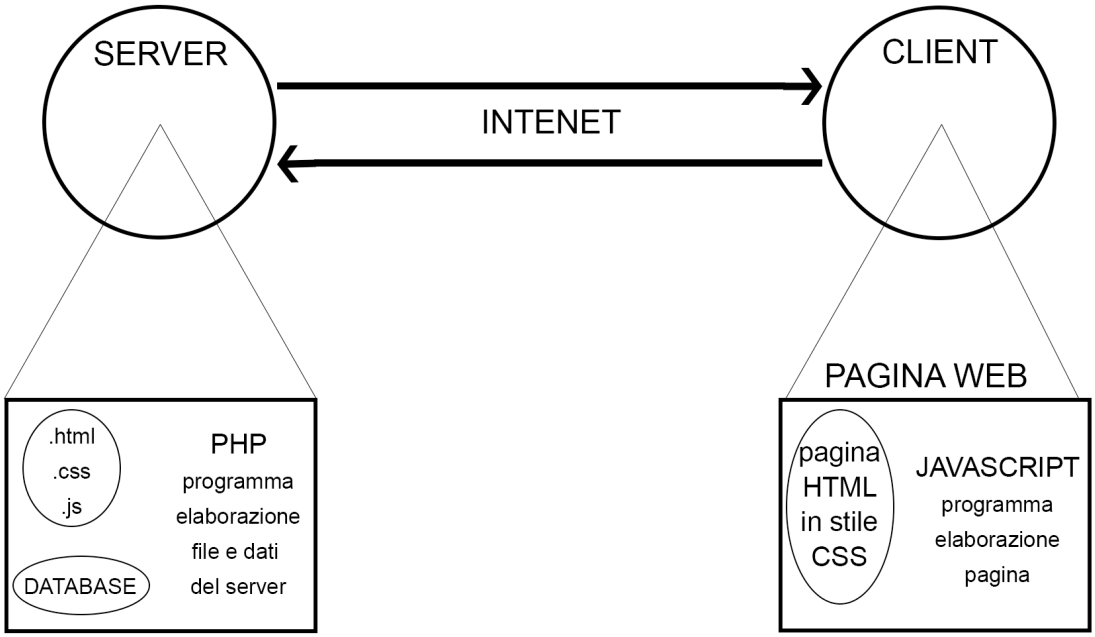

fig.2 – server

Due tipologie di programmi elaborano la nostra esperienza di un sito internet. Un programma al livello del *server* elabora i dati presenti all'interno di un *database*, oppure interi o solamente frazioni di file html, css e *javascript* (residenti all'interno del server), prima che raggiunga il *client*. Un secondo tipo di programma permette invece di elaborare solitamente con un'interazione da parte dell'utente o in maniera automatica la pagina web generata al livello del *client*, quindi del *browser*. Un tipico linguaggio di programmazione a lato *server* è *php* mentre il più conosciuto linguaggio di programmazione a lato *client* è *javascript*.

È molto importante capire quale è il risultato finale delle diverse tipologie di programmi. Partendo dalla programmazione *lato-client*, l'interpretazione del file *javascript* in programma viene svolto dal *browser* e cambia da *browser* a *browser*.

Quando facciamo esperienza di un sito internet con programmazione *lato-client* l'elaborato risultante cambia da utente a utente, rendendola unica per ogni utente (*fig.3*).

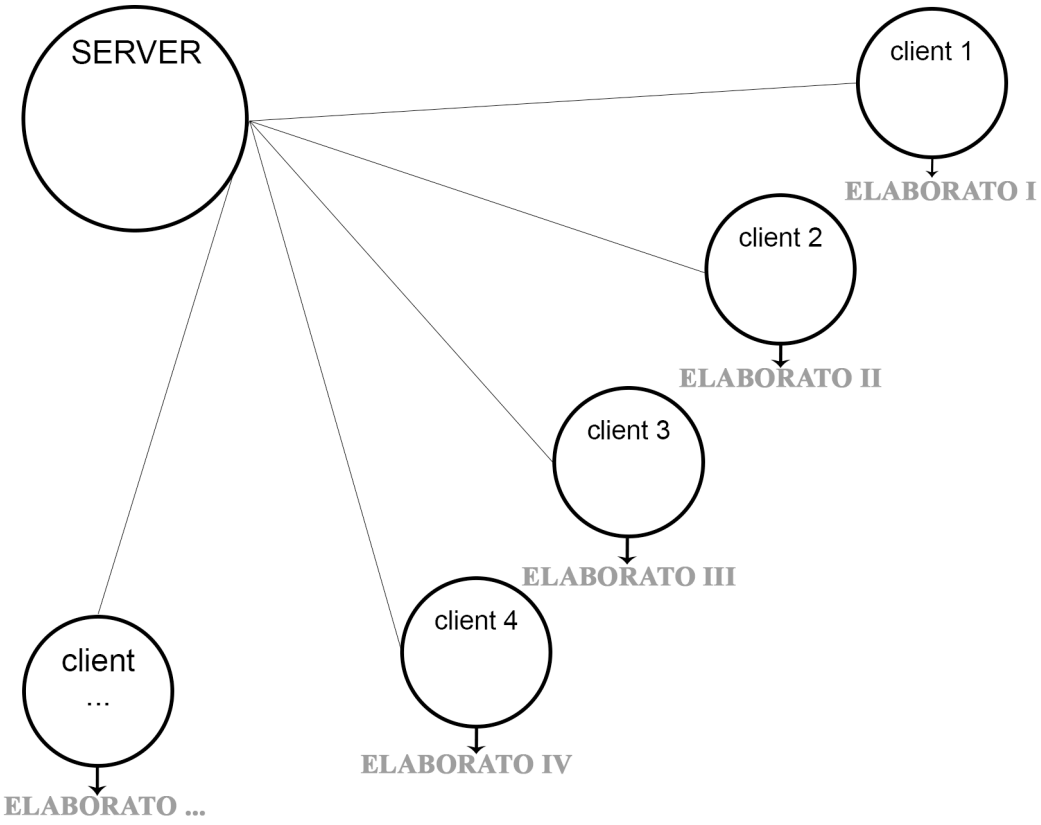

fig.3 - programmazione lato-client

Al contrario quando abbiamo a che fare con una programmazione *lato-server* l'elaborato risultante proviene non più dal *browser* ma dal *server* stesso. Il *browser* in questo caso avrà soltanto il compito di leggere l'elaborato risultante. Questo ci permette di avere un risultato uguale per ogni utente, un unico dato per un numero infinito di *client*. (*fig.4*).

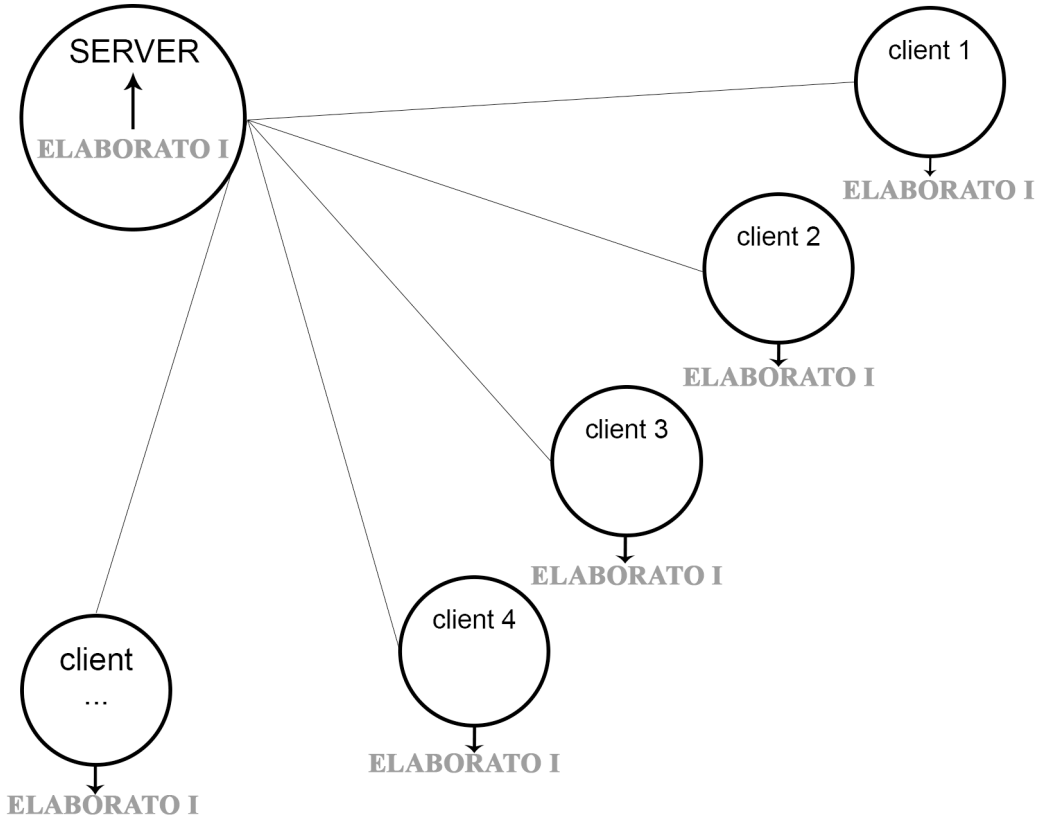

fig.4 - programmazione lato-server

9. Descrizione struttura installazione

L'installazione web consiste di due importanti programmi, uno *lato-client*, creato attraverso *javascript* (più specificatamente con *Web Audio API*) e un altro *lato-server*, creato attraverso *php* per la manipolazione dei *database*. La struttura *lato-client* è un programma che ci permette di ascoltare l'installazione finale. In poche parole crea lo strumento audio che ci consente di effettuare l'elaborazione desiderata.

Il programma *lato-server* invece si occupa di aggiornare ogni minuto un *database*, che sarà la struttura dati che manipola l'elaborazione audio dell'installazione web. In altre parole una volta che il programma *lato-client* è stato interpretato dal *browser* questo servirà da strumento per ascoltare l'elaborazione dati che costantemente vengono aggiornati nel *server*. Praticamente avremo un orecchio che ascolta l'elaborazione dati.

Non abbiamo ancora parlato dei *database*. Qui sarà solo importante capire che sono contenitori di dati residenti all'interno del *server*. Attraverso la tecnica chiamata *cronjob*, che è una pianificazione temporale di comandi scelti, ogni minuto viene attivato un programma *php* che si occupa di aggiornare la struttura del *database*. Come una pulsazione questa tecnica ci permette di rendere in continuo movimento la nostra installazione anche al di fuori di un utente connesso al *server*. L'elaborazione dati continua incessantemente senza mai bloccarsi. Un *client* che visita il sito, dove risiede l'installazione web, viene ad usufruire di un programma *lato-client* che è capace di entrare nel *database*, interpretare i dati e ascoltare il perpetuo respiro dell'installazione web.

10. Algoritmo di *multilayer resampling synthesis*

L'elaborazione che sta alla base dell'installazione web è una *multilayer resampling synthesis*, molto simile ad una sintesi granulare ma semplificata ai processi più basilari. Oltre a questo la nostra elaborazione differisce alla granulazione in quanto i frammenti che vengono generati vanno dai 2 ai 5 secondi (diversamente da una sintesi granulare usuale in cui i grani solitamente stanno sotto 500 millisecondi). I diversi frammenti audio generati dalla *multilayer resampling synthesis* si sovrappongono generando un rafforzamento del timbro e una riorganizzazione del file audio di partenza. Il fine dell'elaborazione è quindi il movimento del timbro che risiede nel file audio. Infatti, per questo motivo, la morfologia del suono che verrà elaborato deve estendersi come una fascia sonora e presentare pertanto un timbro in leggero movimento.

Ora, possiamo vedere l'elaborazione in funzione di un file audio, della durata di 20 secondi circa, che presenta l'evolversi del timbro di una coda di un colpo di tam-tam.

fig.5 - forma d'onda

Come vediamo dalla forma d'onda della *fig.5* notiamo che i transienti di attacco del colpo di tam-tam sono stati smussati da un leggero fade-in. Questo perché vogliamo che il risultato finale non presenti dei momenti di scosse di intensità ma che sia una fascia in leggero moto e senza momenti particolareggianti.

Attraverso la *multilayer resampling synthesis*, in momenti separati da alcuni secondi, verranno lanciate porzioni di file che randomicamente avranno una lunghezza tra 2 e 5 secondi.

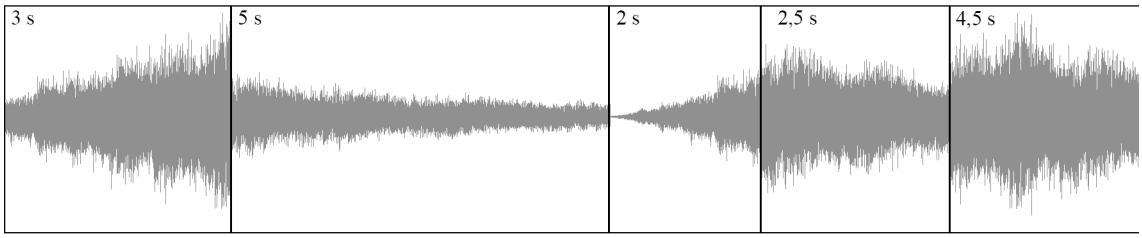

fig.6 - riorganizzazione file audio

Ciascuna frammento lanciato verrà moltiplicata da un inviluppo il quale raggiunge la sua massima ampiezza alla metà della lunghezza della porzione stessa e ritorna a 0 al suo termine. Questo inviluppo non è altro che la comune finestra di granulazione. Di seguito vediamo i frammenti della nostra sintesi con una semplice finestra triangolare.

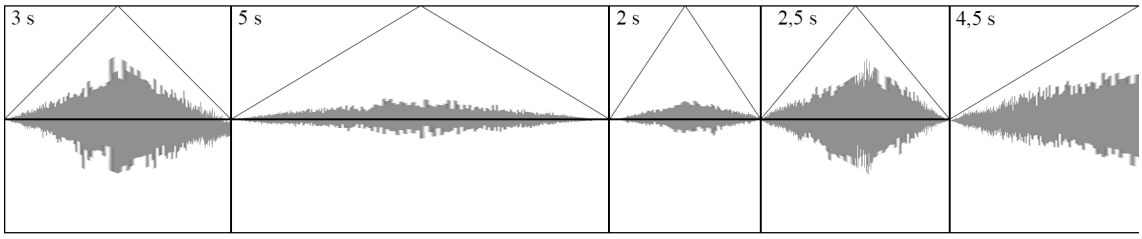

fig.7 - inviluppo per porzione (finestra di granulazione)

Come vediamo, al termine di un frammento ne viene successivamente lanciato un altro. Questo è quello che succede avendo una voce che svolge questa elaborazione. Nel nostro caso abbiamo però più voci (appunto *multilayer*) che elaborando il segnale come sopra, si mescolano insieme provocando appunto quello che cercavamo, ossia il movimento del timbro sonoro del file di partenza.

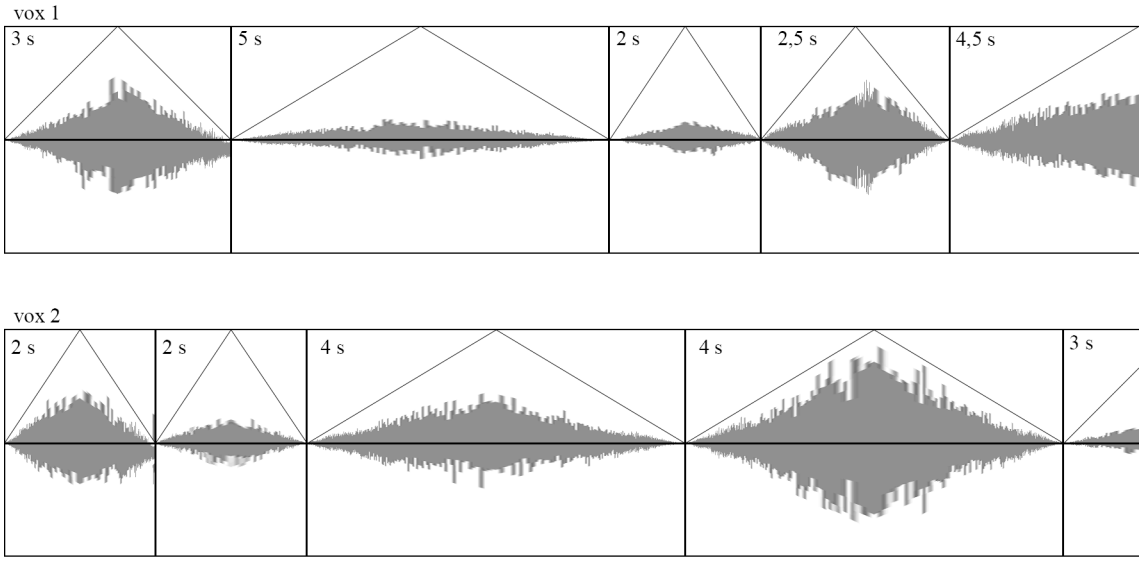

fig.8 - elaborazione a due voci

I parametri che gestiscono il susseguirsi delle porzioni sono veramente ridotti. Ogni voce che gestisce l'elaborazione ha un parametro di *time* che indica la lunghezza del frammento - questo valore viene scelto radomicamente tra i 2 e i 5 secondi; *start* che indica il punto del file audio in cui il frammento viene preso - questo parametro viene scelto randomicamente tra 0 ed 1, dove 0 è l'inizio del file audio ed 1 è l'intera lunghezza meno il parametro precedente, *time* (questo per evitare che l'inviluppo non sfori al di fuori del file audio); *pan* che indica la posizione stereofonica del singolo frammento - il valore, anche esso scelto randomicamente, va tra -1, che è il canale sinistro, ed 1 che è invece il canale destro.

Attraverso questa semplice elaborazione del segnale audio riusciamo ad ottenere un movimento timbrico e d'intensità senza provocare un movimento orizzontale significativo. L'elaborazione audio non uscirà mai dai limiti proposti dalla traccia audio di partenza ma ripresenterà sempre questa, riorganizzata temporalmente e di intensità. Anche gli stessi parametri dell'elaborazione, se pur scelti randomicamente, resteranno fortemente legati e così limitati dai propri range di generazione.

In *Moment V* il materiale audio di partenza proviene da un organo. Più precisamente si tratta dell'organo del conservatorio di Perugia, residente nell'auditorium e costruito da Guido Pinchi nel 1992 su progetto del ex docente di organo Wijnand van de Pol. Le registrazioni sono state effettuate nel dettaglio, catturando l'effetto sonoro che si crea quando i registri non sono correttamente aperti. I suoni registrati sono pertanto non esattamente intonati e provviste di soffio.

11. *Web Audio API* – Realizzazione algoritmo

Ora, vediamo la realizzazione dell'algoritmo della *multilayer resampling synthesis* costruito con *javascript*, più precisamente con *Web Audio API*. Prima di entrare nel dettaglio diciamo solamente due parole su questi linguaggi.

Javascript è un linguaggio di programmazione a script multiuso inizialmente creato per permettere ai sviluppatori di siti internet di aggiungere caratteristiche dinamiche alle pagine web. Oggi ha raggiunto una grande popolarità, anche grazie alla sua forte plasticità, da essere usato anche nella robotica e nei sistemi di automazione domestici. Nel web, *javascript*, è usato per aggiungere interattività e dinamicità in una semplice e statica pagina html (che è lo standard per la costruzione di pagine web). Con *javascript* costruiamo proprio delle applicazioni web che un utente utilizza attraverso il proprio *browser* per performare con la pagina web. Il *Web Audio API* è un insieme di comandi *Javascript* dedicati al controllo dell'audio in una pagina web. *Web Audio API* non da molto tempo va a sostituire il tag audio di html che ci permetteva soltanto di riprodurre in streaming dei file musicali. Questo strumento ci permette di creare applicazioni audio molto complesse, partendo dalla generazione di suoni alla manipolazione di file audio o di una sorgente sonora acquisita in tempo reale. Il concetto di base è quello di creare delle connessioni (un vero e proprio *routing*) fra oggetti denominati *AudioNode*, che sono fondamentalmente di tre tipi: quelli che generano suono; quelli che portano il suono verso la sua destinazione finale; e quelli di elaborazione, come ad esempio dei filtri, e altri strumenti che possono manipolare il segnale che li attraversa.

Procediamo così nel costruire la nostra elaborazione di *multilayer resampling synthesis*.

Al principio della nostra applicazione dobbiamo controllare se *Web Audio API* è presente nel nostro browser. Come abbiamo detto, l'interprete delle nostre pagine e applicazioni web è il nostro *browser*. Essendo *Web Audio API* uno strumento molto

giovane, non tutti *browser* sono ancora capaci di interpretarlo. Per esempio Explorer di windows o Opera mini, ed insieme a questi anche le vecchie versioni di firefox e chrome, non sono capaci di interpretare il *Web Audio API*. Pertanto dobbiamo controllare la presenza dello strumento audio e nel caso fosse assente avvisare l'utente dell'errore proveniente dal *browser*:

```
var contextClass = (window.AudioContext ||
                    window.webkitAudioContext ||
                    window.mozAudioContext ||
                    window.oAudioContext ||
                    window.msAudioContext);
```

In questo modo vediamo se una di queste tipologie di kit *Web Audio API* è presente. La nostra variabile `contextClass` è uguale al kit o altrimenti è indefinita.

```
if(contextClass){
            //web audio API is available
            //open a audio context
              var cntx = new contextClass();
} else {
           alert('web audio API is not available. Try with another
browser');
};
```

Nel caso in cui la variabile non è indefinita, dobbiamo creare, all'interno della nostra applicazione *javascript*, un *contesto audio*, altrimenti "*allertiamo*" l'utente e lo invitiamo a cambiare il *browser*.

Ora dobbiamo caricare il nostro file audio che verrà elaborato tramite la nostra *multilayer resampling synthesis*. Per prima cosa dobbiamo inizializzare un variabile `soundBuffer` in cui verrà caricato un *array* (un insieme di dati) che rappresenta appunto il buffer della forma d'onda del suono.

```
var soundBuffer;
```

Per importare all'interno di un'applicazione *javascript* la maggior parte delle tipologie di documenti, solitamente si utilizza le richieste *ajax*. Non entro nei particolari in quanto questo argomento è molto ampio e complesso. Vedremo solo il suo funzionamento nel caricamento di un file audio.

```
var req = new XMLHttpRequest();
req.open ("GET", "tam-tam.wav", true);
req.responseType = 'arraybuffer';

req.onload = function (){
      cntx.decodeAudioData(req.response, function(buffer){
            soundBuffer = buffer;
            playSound();
      });
};

req.send();
```

Inizializzo una richiesta *ajax* con il comando `new XMLHttpRequest` e la apro con il metodo `.open`. L'apertura della richiesta necessita di tre argomenti: `"GET"` che specifica la tipologia di scambio che avviene tra il *server* e il *client*, `"tam-tam.wav"` che è il nome del file audio e `true` che invece specifica se la richiesta è asincrona o sincrona (rispettivamente `true` o `false`). Poi specifichiamo attraverso il metodo `.responseType` il tipo di risposta che aspettiamo dalla richiesta, in questo caso `'arraybuffer'`. Se la richiesta va a buon fine, e quindi non vengono riscontrati errori, attraverso il metodo `.onload` lanceremo una funzione la quale ci permette di decodificare la risposta e quindi trasformarla in una sorgente audio che passeremo alla nostra variabile `soundBuffer` . Dopo di questo lanciamo una funzione nominata `playSound()` che definiremo in seguito e ci permetterà di riprodurre l'audio elaborato (è in questa posizione che deve essere lanciata in quanto è importante che l'elaborazione parti

solamente dopo che il file audio è stato decodificato[1]). Infine chiudiamo la nostra richiesta con il metodo `.send()`. A questo punto abbiamo il nostro file audio pronto per essere elaborato.

Per quanto riguarda l'elaborazione prima avevamo parlato di soli tre parametri: *time*, *start* e *pan*. Quindi inizializziamoli come variabili:

```
var time;
var start;
var pan;
```

Pertanto abbiamo detto che la lunghezza del frammento (*time)* varia tra 2 e 5 secondi; il punto di inizio rispetto al file audio (*start)* allo stesso modo varia da 0 ad 1; la posizione sterefonica (*pan*) va da -1 ad 1. Per questo creiamo una funzione che sceglie randomicamente il valore di questi parametri:

```
function randomParam(){
      time = (Math.random()*3)+2;
      start = Math.random();
      pan = (Math.random()*2)-1;
};
```

Per avere un numero randomico dobbiamo utilizzare l'oggetto `Math` con il metodo `.random()`. Questo metodo ci da numeri randomici tra 0 ed 1 con sedici cifre dopo la virgola. Nel caso della variabile `time` dobbiamo quindi moltiplicare il range per 3 (il range che sta tra 2 e 5) ed aggiungere il minimo valore da noi richiesto (2). Per la variabile `star` lasceremo pertanto i valori tra 0 ed 1 mentre per la variabile `pan` moltiplicheremo i valori per il range da noi richiesto (2) e li sommeremo al minimo

[1] Questo perché abbiamo utilizzato una richiesta *ajax* asincrona. Questo tipo di richiesta ci permette di interpretare tutto il programma e contemporaneamente effettuare il caricamento del file audio (in questo caso). Questo tipo di richiesta è effettuata per velocizzare i tempi di caricamento dell'applicazione. La funzione `playSound()` verrà inizialmente interpretata ma, dato che utilizza i file audio (come vedremo in seguito), verrà lanciata solo dopo che il nostro buffer è stato correttamente decodificato.

valore (-1). Una volta lanciata la funzione avremo le nostre variabili con i valori randomici nel range da noi stabilito.

```
randomParam();          //lancio della funzione
```

Ora però dobbiamo far suonare il nostro file con la nostra elaborazione. A questo proposito creeremo una funzione.

```
function playSound(){
      var sound = cntx.createBufferSource();
      sound.buffer = soundBuffer;
      var bufDur = sound.buffer.duration;

      var gain = cntx.createGain();
      var panner = cntx.createStereoPanner();
```

Aperta la funzione (che come abbiamo visto è la funzione che verrà lanciata in seguito alla codifica audio) dobbiamo inizializzare gli *AudioNode* che poi andranno a formare il nostro *routing* attraverso dei collegamenti che vedremo in seguito. Con i metodi `.createBufferSource()`, `.createGain()` e `.createStereoPanner()` abbiamo creato rispettivamente i *nodi* che gestiscono il file audio, il gain e il pan. Subito dopo aver creato il *nodo* del file audio, a questo abbiamo associato il nostro buffer (che si trovava nella variabile `soundBuffer`) e di questo ne abbiamo estratto la durata la quale abbiamo allegata alla variabile `bufDur`.

Seguendo, definiamo i nostri tre parametri all'interno della funzione:

```
      panner.pan.value = pan;
      var grainDur = time;
      var startPoint = (bufDur-grainDur)*start;
      gain.gain.value = 1;
```

Quindi il valore del pan, `panner.pan.value`, è uguale al `pan` prima definito randomicamente, così come la nuova variabile `grainDur` è uguale `time`. Per la nuova variabile `startPoint` invece abbiamo moltiplicato il nostro parametro `start` per l'intera durata del file audio (`bufDur`) a cui prima viene sottratta la durata della lunghezza del frammento (`grainDur`). In fine abbiamo definito il valore del gain. A questo punto generiamo la nostra catena audio, cioè il nostro *routing*.

```
sound.connect(panner);
panner.connect (gain);
gain.connect(cntx.destination);
```

Come vediamo colleghiamo il nostro suono al panner (`sound.connect(panner)`), il panner al gain (`panner.connect (gain)`) e così il gain alla `destination` (`gain.connect(cntx.destination)`) che è il nostro hardware di riproduzione audio (altoparlanti del pc, cuffie o interfaccia audio esterna).
Ora ci manca solo di far suonare la nostra elaborazione.

```
sound.start(cntx.currentTime, startPoint);
sound.stop(cntx.currentTime + grainDur);
```

Con lo script sopra facciamo partire subito il suono (`currentTime` indica il tempo trascorso dall'inizializzazione del *contesto audio*) dal punto del file audio determinato da `startPoint`. Stoppiamo la riproduzione del file audio nel momento di inizializzazione del *contesto audio* sommato alla durata del frammento audio, definita da `grainDur`. A questo punto, lanciando la funzione (già abbiamo definito il richiamo della funzione sopra, all'interno della risposta della codifica audio) abbiamo generato il nostro primo frammento. Ma noi non ne vogliamo solamente uno bensì la riproduzione

di un susseguirsi di frammenti. Pertanto prima di chiudere la funzione ne aggiungiamo un'altra al suo interno con il metodo `.onended`:

```
      sound.onended = function () {
            randomParam();
            playSound();
      }
} //close function;
```

Al termine della riproduzione audio del frammento rilanciamo la funzione per aggiornare i parametri randomici e successivamente la funzione per la prossima riproduzione.

A questo punto siamo nella stessa situazione in cui eravamo nella *fig.6* del capitolo precedente. Dobbiamo così aggiungere un inviluppo. Per semplificare inseriremo una finestra triangolare, come nell'esempio del capitolo precedente.

Per creare una finestra dovremo utilizzare un *array*. In questo ne caso utilizzeremo uno particolare che è il `Float32Array`:

```
var envelop = new Float32Array(3);
```

L'argomento (`(3)`) del `Float32Array` sta ad indicare il numero di indici (cioè il numero di dati che conterrà). A questo punto dovremmo definire questi indici i quali conterranno i valori del gain in determinate posizione. Per un'onda triangolare ci basterà avere tra punti fondamentali: l'inizio, in cui il valore è 0, la metà dell'inviluppo, in cui il valore è 1 e il termine dell'inviluppo in cui il valore è ancora 0. Questo particolare tipo di *array* ci permette di interpolare con 32 bit il susseguirsi dei valori degli indici.

Infine dovremmo aggiungere un'istruzione che generi l'inviluppo appena lanciata la riproduzione audio del nostro frammento. È importante capire che l'inviluppo va a

gestire nel tempo il gain, per cui dobbiamo cancellare dal nostro script l'assegnazione di un valore fisso al gain (`gain.gain.value = 1`).

```
      sound.start(cntx.currentTime, startPoint);
      gain.gain.setValueCurveAtTime (envelop, cntx.currentTime,
grainDur);
      sound.stop(cntx.currentTime + grainDur);
```

Come vediamo il gain verrà impostato attraverso il metodo `.setValueCurveAtTime` che appunto genera un curva la quale controlla il valore del gain nel tempo. Questo metodo utilizza tre argomenti, che sono: la *curva* (in questo caso la nostra finestra triangolare `envelop`), l'inizio della curva (`cntx.currentTime`) e la durata (`grainDur`).
In questo modo avremmo solamente una voce che applica la nostra elaborazione, per cui non sarà multistrato (*multilayer*). Vediamo allora in seguito come realizzare il nostro movimento timbrico con la generazione di più voci.
In primo luogo semplifichiamo lo script mettendo la generazione dei valori randomici direttamente all'interno della funzione `playSound()`. La definizione dei valori dei parametri, all'interno della funzione, diventerà:

```
      panner.pan.value = (Math.random()*2)-1;
      var grainDur = (Math.random()*3)+2;
      var startPoint = (bufDur-grainDur)* Math.random();
```

A questo punto possiamo eliminare la funzione soprastante `randomParam()`. Pertanto possiamo eliminare tutte le variabili esterne alla funzione che si riferivano a questi parametri (`time`, `start` e `pan`) e con queste il richiamo della funzione `randomParam()` che avveniva subito dopo la sua inizializzazione e al termine della riproduzione audio del frammento. Ora, non basterà più lanciare solamente una volta la funzione `playSound()` al termine della codifica audio, ma bisognerà lanciarla un totale di volte

quanto è il numero di voci desiderato. A questo scopo utilizzeremo un ciclo `for` come di seguito:

```
req.onload = function (){
      cntx.decodeAudioData(req.response, function(buffer){
            soundBuffer = buffer;
            for (var i = 0; i < 4; i++){
                  playSound();
            }
      });
};
```

Questo tipo di ciclo si ripete finché la variabile definita al suo interno (`var i = 0`) non esaudisce più la condizione stabilita. Questo vuol dire che, finché la variabile non raggiunge il valore 4 (cioè rimane all'interno della condizione `i < 4`), viene lanciata la funzione all'interno delle parentesi graffe (`playSound()`). Ogni volta che vengono svolti i processi descritti all'interno delle parentesi graffe (nel nostro caso ogni volta che viene lanciata la funzione) il valore della variabile del ciclo aumenta di 1 (`i++` dove `i` è la variabile e `++` l'accrescimento di un unità) e così si ripete la verifica della condizione. In questo modo lanciamo quattro volte la nostra funzione per avere pertanto quattro voci nel nostro *multilayer resamling*.

Attraverso questo capitolo abbiamo visto solamente l'implementazione base dell'algoritmo. Come abbiamo già detto i dati dell'elaborazione non provengono dall'applicazione *lato-client*, come in questo caso. L'incessante elaborazione viene gestita infatti da dati aggiornati costantemente da un'applicazione *lato-server*. Per questo, l'applicazione *javascript* dell'installazione web presenta particolari differenze da quella che abbiamo mostrato sopra. Il risultato sonoro finale è lo stesso ma lo script si sviluppa in concomitanza ad un algoritmo di sincronizzazione con il *server*. Ai fini di

questo lavoro mi sono proposto di tralasciare una spiegazione dettagliata di tale algoritmo. Lascerò comunque sia l'intero script nell'appendice dello scritto.

Nel prossimo capitolo vedremo brevemente l'implementazione dell'applicazione *lato-server*.

12. Applicazione *lato-server*

Come abbiamo già più volte ripetuto, tutti i dati dell'elaborazione provengono dal *server* che è in constante aggiornamento. I dati risiedono in un *database* che attraverso un'applicazione *php*, lanciata ogni minuto per via della tecnica nominata *cronjob*, veine aggiornato tramite comandi *sql*.

I *database* sono contenitori di dati che si organizzano in una struttura a tabelle. La singola tabella rappresenta un insieme di elementi utilizzando un modello di colonne verticali (identificate con il loro nome) e righe orizzontali. La *cella* di una tabella è un'unità in cui una riga e una colonna si intersecano. Un esempio di tabella può essere la seguente:

ID	**Nome**	**Cognome**	**Anni**	**Nazione**	**Stile**
1	Guillaume	De Machaut	1300-1377	Francia	Ars nova
2	Giovanni Pierluigi	Da Palestrina	1525-1594	Italia	Sacra rinascimentale
3	Orlando	Di Lasso	1532-1594	Belgio	Sacra rinascimentale
4	Arcangelo	Corelli	1653-1713	Italia	Barocco

In questa tabella possiamo notare le cinque colonne (ID, Nome, Cognome, Anni, Nazione, Stile) che assumono un valore numerico o letterale per ogni riga orizzontale. La *cella* con il valore "Orlando" è l'intersezione della terza riga con la seconda colonna.

I comandi *sql* sono quelli che ci permettono di modificare, aggiungere ed eliminare valori di una riga orizzontale o solo di una *cella*. Con il comando `INSERT` possiamo per esempio popolare la nostra tabella aggiungendo una riga orizzontale:

```
INSERT INTO nome_tabella (ID, Nome, Cognome, Anni, Nazione, Stile)
VALUE (5, Karlheinz, Stockhausen, 1928-2007, Germania, Elettronica)
```

Il *database* dell'installazione web presenterà una tabella per ogni voce della nostra sintesi. Per cui per ogni tabella dovremo assegnare i nostri parametri e per questo avremo solamente tre colonne ("Time", "Start", "Pan") più la colonna iniziale per l'indice ("ID"):

ID	**Time**	**Start**	**Pan**

Pertanto ogni riga presenterà i valori randomici di ogni frammento della voce a cui si riferisce la tabella. Quindi, essendo il *database* aggiornato ogni minuto il numero delle righe di ogni tabella sarà uguale al numero degli addendi, che sta al valore randomico di ogni durata del frammento audio (*time*), la cui somma è uguale, o maggiore di massimo quattro secondi, a sessanta. Più precisamente la tabella verrà riempita attraverso dei comandi *sql* finché la somma dei *time* di ogni riga non sarà maggiore o uguale a sessanta. Possiamo fare un esempio con numeri reali interi che sono scelti sempre all'interno del range tra 2 e 5 secondi.

ID	**Time**	**Start**	**Pan**
1	2	0.5	0.9
2	3	0.1	-1
3	2	0.03	0.4
4	2	0.23	0.66
5	5	0.9	-0.5
5	4	0.77	-0.4
6	4	0.05	-0.1
7	2	0.01	0.001
8	5	0.1	0.32
9	3	0.2	-0.23
10	3	0.32	-0.045
11	3	0.01	-0.22

12	5	0.4	-0.99
13	4	0.41	-0.45
14	2	0.56	0.33
15	4	0.6	1
16	2	0.97	-0.6
17	3	0.3	-0.77
18	5	0.1	0.82

Come possiamo vedere la somma dei valori che si presentano sotto la colonna "Time" è uguale a 63. Finché il valore era minore di 60 (nella riga 17 il valore della somma era uguale a 58) la tabella continuava a popolarsi.

Questo processo di aggiornamento di tutte le tabelle che costituiscono il *database* viene fatto attraverso un applicazione *php* che utilizza i comandi *sql*. Vediamo di seguito un esempio di script *php* semplificato che si occupa di un aggiornamento del genere:

```
$servername = "indirizzo_server";
$username = "nome_server";
$password = "password_server";
$dbname = "nome_database";
```

Inizializziamo delle variabili che identificano il *server* e assegnano i permessi per poter accedere al *database*. Di seguito apriamo la connessione con il *database* (che è di tipo MySQL[35]).

```
$conn = new mysqli($servername, $username, $password, $dbname);
```

Poi controlliamo se la connessione ha successo, altrimenti avvisiamo il tipo di errore:

[35] Il mySQL è un tipo di database relazionale (Relational database management system - RDBMS) che quindi consente la manipolazione e la gestione dei dati attraverso un linguaggio di programmazione (appunto l'sql).

```
if ($conn->connect_error) {
      die("Connection failed: " . $conn->connect_error); }
```

Se tutto è avvenuto con successo iniziamo a riempire le tabelle. Qui vediamo l'utilizzo del ciclo `for` e dei comandi *sql* in *php*:

```
for ($i = 0; $i < numero_tabelle; $i++){
      $sql = "DELETE FROM tabella".$i."";
      $conn->query($sql);
      $stmt = $conn->prepare("INSERT INTO tabella".$i." (id, time,
      start, pan)
      VALUE(?, ?, ?, ?)");
      $stmt->bind_param('dddd', $id, $time, $start, $pan);
```

In questo modo per ogni tabella abbiamo inizialmente eliminato ogni riga, in modo da svuotare il nostro *database* (utilizziamo per questo il comando `DELETE FROM nome_tabella`). Successivamente abbiamo riutilizzato il comando *sql* `INSERT` per ognuna, dove però i valori sono indicati con dei punti di domanda (`VALUE(?, ?, ?, ?)`). Assoceremo questi valori a delle variabili con il comando `bind_param` e le definiremo in seguito. Il ciclo `for` che segue, all'interno di quello che abbiamo appena visto, popolerà di righe orizzontali ogni tabella.

```
      for ($j = 0, $somma = 0; $somma < 60; $j++){
            $id = $j+1;
            $time = numero_randomico_tra_2/5;
            $start = numero_randomico_tra_0/1;
            $pan = numero_randomico_tra_-1/1;
            $stmt->execute();

            $somma += $time
      }
```

Attraverso questo ciclo vediamo che, per ogni volta che si ripete, le variabili vengono riempite con i valori randomici richiesti e caricati nella riga orizzontale della tabella (con il comando *php* `execute()`) . L'ultimo comando del ciclo aggiunge di volta in

volta un addendo alla somma che andrà a verificare la condizione del ciclo stesso. Il comando `$somma += $time` significa che la somma è uguale alla somma stessa più il nuovo valore della variabile `$time` .

```
      $stmt->close();
};

$conn ->close();
```

Alla fine di ogni fase del ciclo precedente (che è il contenitore del ciclo che abbiamo appena visto) chiudiamo l'aggiornamento di ogni tabella. Poi, al termine del completo aggiornamento del *database* chiudiamo la connessione con il *server*.

Questa applicazione verrà lanciata ogni minuto con un *cronjob*. Questo tipo di tecnica permette di pianificare eventi e sta appunto ad indicare un'attività (*job*) eseguita in un tempo cronometrato (*cron*). Nel web il *cronjob* assume la forma di un altro *server web* che si aggiunge nello schema precedentemente visto:

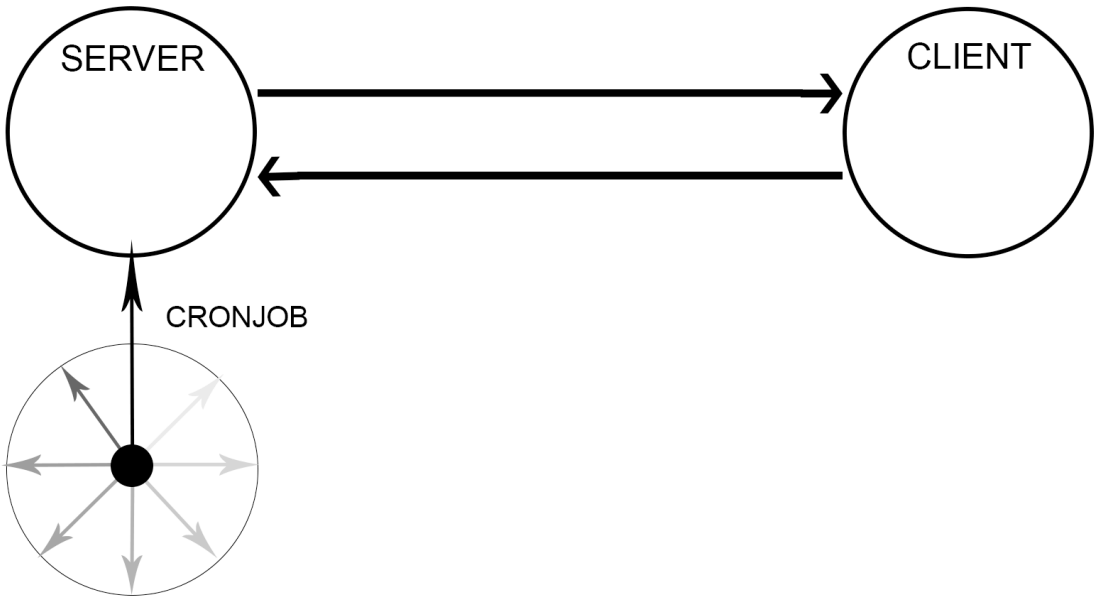

fig.9 - il cronjob nella struttura web

Sempre attraverso la connessione internet un altro *server* viene programmato solamente per lanciare una pagina di aggiornamento all'interno del *server* principale dov'è contenuto il sito internet. Nel nostro caso il *cronjob* darà una pulsazione costante all'aggiornamento del *database* lanciando ogni minuto l'applicazione *php* che abbiamo visto sopra.

A questo punto la nostra applicazione *lato-client* perde tutta la generazione dei dati randomici di cui si occupa invece, incessantemente, l'applicazione *lato-server*. Per questo motivo dovrà invece avere un algoritmo per l'acquisizione dei dati (anche questa viene applicato con una richiesta *ajax* ma in maniera più complessa che non vedremo nel dettaglio). In questo modo diventerà proprio uno strumento di ascolto per la continua pulsazione del *server*.

Andando nella pagina internet che ospita l'installazione web il *browser* web interpreterà subito lo strumento di ascolto. Aspetterà però un nuovo aggiornamento da parte del *cronjob* verso il *server*. Appena il *cronjob* lancia l'applicazione *php* e appena il *database* viene aggiornato, l'applicazione *lato-client* inizierà ad ascoltare e a suonare. Ogni voce, per sessanta secondi o poco più, spacchetterà di volta in volta la tabella a se associata, elaborando il file audio in ampi frammenti audio, i quali hanno i valori di *time*, *start* e *pan* derivanti da ogni riga della tabella stessa. Ai seguenti lanci dei *cronjob* però non tutte le voci avranno terminato (è raro che una tabella duri perfettamente sessanta secondi). Pertanto, per evitare lo sfasamento, il numero di voci è sempre il doppio del numero delle tabelle, così che l'applicazione *lato-client* può entrare istantaneamente in ascolto del nuovo *database* e contemporaneamente portare al termine quello precedente.

Riepilogo

La composizione web sarà installata nel sito www.momentv.org. Attraverso questo indirizzo accederemo all'applicazione *javascript* che ci permetterà di ascoltare l'incessante respiro del *database*. Appena accederemo comparirà una facciata di attesa. Infatti dovremo aspettare il prossimo aggiornamento del *database* per poter iniziare ad ascoltare. Appena sarà consentito di ascoltare l'installazione web potremmo lasciare l'applicazione *lato-client* in ascolto del *database* per un tempo indeterminato. Potremmo quindi ascoltare la composizione come un pezzo di musica infinito. Sarà possibile assistere a questo eterno timbro in movimento per ore o addirittura per giorni interi. Ma potremmo ascoltare la composizione per un tempo molto breve e allo stesso modo, come ci diceva Jonathan Kramer, avremmo ascoltato l'intera composizione. Come una scultura che risiede in un determinato luogo, *Moment V* si offre ad un pubblico all'interno di uno spazio virtuale in cui i visitatori possono accedere in qualsiasi momento della sua vita. Una struttura autentica e unica nella sua forma concreta che diventa invece molteplicità all'interno di ogni coscienza.

Moment V porta l'importanza del compito svolto dell'ascoltatore al livello della stessa composizione. Infatti, solo mediante una coscienza che la acquisisce al suo interno essa potrà realizzarsi. Mentre al di fuori di noi esisterà sempre come un timbro sonoro eterno, al nostro interno, mediante quella che abbiamo chiamato *interattività coscienziale*, essa si trasformerà in una dinamicità. Questa dinamicità, che non viene più data dalla struttura orizzontale della musica, sarà posta dal compenetrarsi dei nostri dati di coscienza. Le *impressioni* emotive, mentali, fisiche e somatiche che determinano la

nostra reale durata verranno immerse nella struttura verticale della composizione. Così intinte modificheranno la nostra percezione temporale che darà così una struttura orizzontale a questo eterno timbro sonoro.

Conclusioni

Risponderemo quindi alla domanda posta al principio di questo lavoro dicendo che: praticamente è possibile realizzare un opera d'arte musicale senza tempo ma non è concepibile ad un livello teorico.

Dal punto di vista pratico è possibile la realizzazione di un algoritmo capace di lavorare incessantemente per creare un'elaborazione audio che non generi orizzontalità. Come abbiamo visto, nella seconda parte di questo lavoro, la sintesi utilizzata e continuatamente in movimento non genera orizzontalità in quanto esiste all'interno di limiti ben stabiliti. Quello che noi sentiamo, come prodotto acustico finale, non è altro che un movimento di un timbro sonoro delimitato dai file audio che lo costituiscono verticalmente e dai range dell'elaborazione di questi. In altre parole la composizione si definisce nella sua interezza fin dal principio.

Dal punto di vista teorico invece non potremmo mai realizzare un'opera d'arte musicale senza tempo. Infatti un'opera d'arte è tale quando una coscienza prende consapevolezza di essa. Essendo pertanto l'esperienza di un soggetto rigorosamente temporale, sarà allo stesso modo temporale un'opera d'arte. Anche nelle arti plastiche, dove sembrerebbe che il tempo venga a meno, l'esperienza dell'opera si stabilisce ancora temporalmente.

In linea con il pensiero di Jonathan Kramer concluderemo dicendo che: il tempo impiegato nell'ascolto della musica verticale è tempo strutturato, ma la struttura non è posta dalla musica bensì da noi, influenzata dalla vita della nostra coscienza.

Appendice

Di seguito saranno presentati i codici che costituiscono l'installazione web. Pertanto verranno riportati i file *javascript*, css e html gestiti dal *browser* e i file *php* che invece si occupano della gestione del *database*. Il file *php*, db_update.php, rappresenta il codice che viene ogni minuto lanciato dal *cronjob*.

Ogni file è racchiuso in una cartella che al suo interno presenta altri file della stessa tipologia. Per esempio i file *javascript* si trovano tutti all'interno della cartella js.

Solamente due file sono all'esterno delle cartelle. Questi sono i file html in cui vengo indirizzati gli utenti all'apertura del sito internet e i quali dirigono la successione di svolgimento delle parti dell'applicazione *lato-client*.

Il *server*, al suo interno, presenta, come già detto, il *database*, ogni cartella e file con tutte le parti dell'applicazione.

In questo caso il *database* è costituito di un numero di tabelle uguale al numero di voci che viene utilizzate dalla granulazione. Non abbiamo specificato il numero di voci, però possiamo dire che ogni tabella è nominata "vox" con il numero della voce a cui appartiene (vox0, vox1, vox2, vox3…voxN).

N: Quest'appendice serve solamente per presentare in generale la struttura dell'algoritmo. Troveremo pertanto parti di codice scritte diversamente (es."`sound1.wav`") che stanno ad indicare specifiche tralasciate perché superflue per la nostra presentazione.

Nome File: app.js

Tipo: Applicazione Javascript

Funzione: multilayer resampling synthesis

Codice:

```
//check availability of 'Web Audio API'
var contextClass = (window.AudioContext ||
                  window.webkitAudioContext ||
                  window.mozAudioContext ||
                  window.oAudioContext ||
                  window.msAudioContext);

if(contextClass){
      //web audio API is available
      //open a audio context!
      var cntx = new contextClass();
} else {
      window.location.href = "error-page.html";
}

//creating an Array with the sounds

url  var arrayURL = [   "sound1.wav",
                        "sound2.wav",
                        "sound3.wav",
                         ecc..
                        ];

//creating an empty array (it will contain the sound voices)

var objectVox = [];

//creating envelop array

var envelop = new Float32Array(100);
var envelopLength = 100;
var maxAMP = 0.3;

//filling the envelop array with window function

for (var count = 0; count < envelopLength; count++){
      envelop[count] = (0.5*(1-
Math.cos((2*Math.PI*count)/envelopLength)))*maxAMP;

}

//Creating a convolver Node and loading Impulse Response File for
Reverb

var reverb = cntx.createConvolver();
```

```
var reqImpulse = new XMLHttpRequest();
reqImpulse.open("GET","impulso.wav",true);
reqImpulse.responseType = 'arraybuffer';
reqImpulse.onload = function(){
      cntx.decodeAudioData(reqImpulse.response, function(impulse){
      reverb.buffer = impulse;
      reverb.connect(cntx.destination);
      });
};

reqImpulse.send();

//creating playSound object to load and play soundFile  function

playSound (sound) {

      var self = this;
      self.url = sound;
      self.buffer;
      self.id = 0;
      self.intDatabase;

      //function to load sound file

      self.bufferLoader = function () {
            var req = new XMLHttpRequest();
            req.open ("GET", self.url, true);
            req.responseType = 'arraybuffer';
            req.onload = function (){
            cntx.decodeAudioData(req.response, function(buffer){
                  self.buffer = buffer;});
                  };

            req.send();

            };

      self.start = function(){

            if (self.id < self.intDatabase.length){

                  self.play(self.intDatabase[self.id].time,
                            self.intDatabase[self.id].start,
                            self.intDatabase[self.id].pan);
                  }else{ self.id = 0};

      //function to play sound file
      self.play = function (time, start, pan){
                  var sound = cntx.createBufferSource();
                  sound.buffer = self.buffer;
                  var bufDur = sound.buffer.duration;
                  var gain = cntx.createGain();
```

```
                var panner = cntx.createStereoPanner();

                panner.pan.value = parseFloat(pan);
                var grainDur = parseFloat(time);
                var startPoint = (bufDur-grainDur)*parseFloat(start);

                sound.connect(panner);
                panner.connect (gain);
                gain.connect(reverb);

                sound.start(cntx.currentTime, startPoint);
                gain.gain.setValueCurveAtTime (envelop,
cntx.currentTime, grainDur);
                sound.stop(cntx.currentTime + grainDur);

                sound.onended = function () {
                      self.id += 1;
                      self.start();
                };
          };
}

//filling the array with multiple voices of playSound object

var soundVoices = arrayURL.length; //number of sound file
var nVfS = 4; //voice for sound file

for (var vox = 0; vox < soundVoices; vox++){
      objectVox[vox] = [];
      for (var i = 0; i < nVfS; i++){
            objectVox[vox][i] = new playSound(arrayURL[vox]);
            objectVox[vox][i].bufferLoader();
      }
}
```

Nome File: __updatedatabase.js

Tipo: Applicazione Javascript

Funzione: Acquisizione del database (per ogni aggiornamenteo)

Codice:

```
var extId = 0;
var extDatabase = [];

function _startFunc(x, y){
      var z = 1-((y+1)%2);
      objectVox[x][y].intDatabase = extDatabase[z + (x*soundVoices)];
      objectVox[x][y].start();
}

function loopIncreaseWithLimit(){
      if (extId < nVfS-1){
            extId++;
      } else {
            extId = 0
      }
}

function updateDatabase(){

      $.ajax[1]({
            type: 'GET',
            url: 'php/db_json.php',
            async: true,
            dataType: 'json'
            success: function (data){
                  extDatabase = data;
                  for (var vox = 0; vox < soundVoices; vox++){
                        startFunc(vox, extId);
                  }
                  loopIncreaseWithLimit();

                  setTimeout(function(){
                        for (var vox = 0; vox < soundVoices; vox++){
                              _startFunc(vox, extId);
                        }
                        loopIncreaseWithLimit()
                  },1000);
            }
      })
}
```

[1]In questo caso viene fatta una richiesta *ajax* utilizzando la libreria jQuery (`$.ajax`).

Nome File: __update_impulse.js

Tipo: Applicazione Javascript

Funzione: attiva il meccanismo di acquisizione per ogni aggiornamento del database.

Codice:

```
var timeStart = 0;
var timestamp = null;

function _func(){

	switch(timeStart){
		case 0:
			timeStart++;
		break;
		case 1:
			endLoading();
			updateDatabase();
			timeStart++;
		break;
		default:
			updateDatabase();
		}
}

function waitForMsg(){
	$.ajax({
		type: 'GET',
		url: 'php/__get_impulse.php?timestamp=' + timestamp,
		async: true,
		cache: false,
		success: function(data){
			var json = eval('('+ data + ')');
			if (json.theContent != ""){
				_func();
				timestamp = json.timestamp;
			}

			setTimeout('waitForMsg()', 1000);
		},

		error: function(XMLHttpRequest, textStatus, errorThrown){
			alert('error: '+ textStatus + '('+ errorThrown +
		')');

			setTimeout('waitForMsg()', 15000);
		}
	});
}

$(document).ready(function(){	waitForMsg(); });
```

Nome File: __loading.js

Tipo: Applicazione Javascript

Funzione: sistema di visualizzazione dinamica (manipolazione html).

Codice:

```
var $img = $("<img id='loading' src='immagine di attesa' alt=''>");
var $img2 = $("<img id='wellcome' src='immagine benvenuto' alt=''>")

function endLoading(){
      $('#loading').fadeOut(1000, function(){
            $('img').remove('#loading');
            });
      wellcome()
      $('#top_presentation').fadeIn(5000);
      watch();
      title();
}

function watch(){
      setInterval(function(){
      var data = Math.floor(Date.now()/1000)-data di pubblicazzione;
      $('p').remove('#time')
      $('#top_presentation').append("<p id='time'
class='top_text'>"+data+" sec</p>")
      }, 1000);
}

function title(){
      $('#top_presentation').append("<p id='title'
class='top_text'>moment v</p>")
}

$('document').ready(function(){

      $('#descript').hide();
      $('#top_presentation').hide();
      $('#loading_div').append($img);

      var speed = 5000;
      var showId = 0;

      $('#but').click(function(){
            showId++;
            if(showId%2){
            $('#descript').fadeIn(speed);
                  if($('#loading')){
                        $('.move_div').animate({top: '80%'}, speed);
                        $('#loading, #wellcome').animate({height:
                  '100px', width: '100px'}, speed)
                  }
```

```
            }else {
                  $('#descript').fadeOut(speed);
                        if($('#loading')){
                              $('.move_div').animate({top: '20%'},
                        speed);
                              $('#loading, #wellcome').animate({height:
                        '200px', width: '200px'}, speed)
                        }
            }
      })
});

function wellcome(){
      $('#wellcome_div').append($img2).hide();
      $('#wellcome_div').fadeIn(2000, function(){
            $('#wellcome_div').fadeOut(7000, function(){
                  $('img').remove('#wellcome_div')
            });
      })
}
```

Nome File: index.html

Tipo: pagina html

Funzione: interfaccia di visualizzazione.

Codice:

```
<!doctype html>
<html>
<head>

<link href="css/app.css" rel="stylesheet" type="text/css">
<script type="text/javascript" src="js/jquery-3.2.1.min.js"></script>[1]
<script type="text/javascript" src="js/__loading.js"></script>
<script type="text/javascript" src="js/app.js"></script>
<script type="text/javascript" src="js/__update_impulse.js"></script>
<script type="text/javascript" src="js/__update_database.js"></script>

<meta charset="UTF-8">

<title>moment V</title>

</head>

<body>

<div id="top_presentation"></div>
<div id="loading_div" class="move_div"></div>
<div id="wellcome_div" class="move_div"></div>

<div id="descript" class="top_text">     <p> Descrizione
dell'installazione web....</p>

</div>  <button type="button"id="but">info</button>

</body>

</html>
```

[1] Dichiarazione di inizializzazione della libreria jQuery (la libreria, che è un file javascript, è inserita nella cartella js)

Nome File: error-page.html

Tipo: pagina html

Funzione: Avviso di errore quando Web Audio API non è avviabile.

Codice:

```
<!doctype html>
<html>
<head>
<link href="css/app.css" rel="stylesheet" type="text/css">
<script type="text/javascript" src="js/jquery-3.2.1.min.js"></script>

<meta charset="UTF-8">
<title>error</title>

</head>

<body>

<div id="error_div" class="move_div"></div>

<script>[1]
var $e_img = $("<img id='error' src='immagine di errore' alt=''>");
function alert_error(){
	$('#error_div').fadeIn(2000, function(){
		$('#error_div').fadeOut(1000, function(){
			alert_error();
		});
	})
};

$('document').ready(function(){
	$('#error_div').append($e_img).hide();
	alert_error();
});

</script>

</body>

</html>
```

[1] Utilizzo di javascript all'interno di una pagina html

Nome File: app.css

Tipo: foglio di stile

Funzione: stile pagina html.

Codice:

```
body{
	background-color:
	black;
}
.move_div{
	position:
	absolute;
	top: 20%;
	bottom: 20%;
	left: 20%;
	right: 20%;
}
#loading{
	width: 200px;
	height: 200px;
	display: block;
	margin: 0 auto;
}
#wellcome{
	width: 200px;
	height: 200px;
	display: block;
	margin: 0 auto;
}
#error{
	width: 200px;
	height: 200px;
	display: block;
	margin: 0 auto;
}
.top_text{
	font-family: Arial, Helvetica, sens-serif;
	color: white;
}
#title{
	position: absolute;
	top: 2%;
	left: 2%;
}
#time{
	position: absolute;
	top: 2%;
	right: 2%;
}
```

```
#descript{
      width: 700px;
      text-align: center;
      position: absolute;
      top: 25%;
      left: 35%;
      margin-top: -100px;
      margin-left: -100px;
}
#but{
      display: inline-block;
      background-color: rgba(255,255,255,0.54);
      border: 2px solid #ffffff;
      padding: 5px, 15px;
      width: 100px;
      position: absolute;
      bottom: 2%;
      right: 2%;
}
```

Nome File: db_update.php

Tipo: applicazione php

Funzione: aggiornamento del database.

Codice:

```
<?php
function randFloat($min, $max){
      $depth = 100000;
      return rand($min*$depth, $max*$depth)/$depth;
};
$arrayNum = array();

for ($i = 0; $i < 20; $i++){
      $arrayNum[$i] = [];

      for ($j = 0, $num = 0; $num < 60; $j++){
            $t = randFloat(2, 5);   //tempo -> time nel database
            $s = randFloat(0, 1);   //punto di partenza -> start
            $p = randFloat(-1, 1);  //pan

            $arrayNum[$i][$j] = [$t, $s, $p];
            $num += $t; //aumenta fino a 60
      }
}

$servername = "indirizzo_server";
$username = "nome_server";
$password = "password_server";
$dbname = "nome_database";

$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
      die("Connection failed: " . $conn->connect_error);
}

for ($i = 0; $i < count($arrayNum); $i++){

      $sql = "DELETE FROM vox".$i."";
      $conn->query($sql);
      $stmt = $conn->prepare("INSERT INTO vox".$i." (id, time, start,
pan)  VALUE(?, ?, ?, ?)");
      $stmt->bind_param('dddd', $id, $time, $start, $pan);

      for ($j = 0; $j < count($arrayNum[$i]); $j++){
      $id = $j+1;
      $time = $arrayNum[$i][$j][0];
      $start = $arrayNum[$i][$j][1];
      $pan = $arrayNum[$i][$j][2];
      $stmt->execute();
      }
```

```
      $stmt->close();
};

$conn>close();

date = date("Y-m-d H:i:s");
$fp = fopen('__updateimpulsefile.txt', 'w');
fwrite($fp, "update_IMPULSE_File ".$date);
fclose($fp)[1];

?>
```

[1] Le ultime quattro righe ci permettono di aggiornare un file testuale presente nel database. Questo aggiornamento viene incessantemente atteso dall'applicazione vista sopra __update_impulse.js attraverso una tecnica chiamata *long polling*. Appena l'aggiornamento è effettuato tutta l'applicazione javascript si attiva per acquisire il nuovo database (aggiornato contemporaneamente al file, in questa applicazione php).

Nome File: db_json.php

Tipo: applicazione php

Funzione: trasformazione del database in file JSON per la lettura in javascript.

Codice:

```
<?php

$servername = "indirizzo_server";
$username = "nome_server";
$password = "password_server";
$dbname = "nome_database";

$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
      die("Connection failed: " . $conn->connect_error);
}

$array = array();

for ($i = 0; $i < 16; $i++){
      $sql = "SELECT * FROM vox".$i."";
      $result = $conn->query($sql);
      while($row = $result->fetch_assoc()){
            $array[$i][] = $row;
      }
}

$conn->close();

echo json_encode($array);

?>
```

Nome File: db_json.php

Tipo: applicazione php

Funzione: trasformazione dell'file testuale in JSON per la lettura in javascript

Codice:

```
<?php

$filename=dirname(__FILE__).'/__updateimpulsefile.txt';
$lastmodif = isset($_GET['timestamp']) ? $_GET['timestamp'] : 0;
$currentmodif = filemtime($filename);

while ($currentmodif <= $lastmodif){
      usleep(10000);
      clearstatcache();
      $currentmodif = filemtime($filename);
}

$response = array();
$response['theContent'] = file_get_contents($filename);
$response['timestamp'] = $currentmodif;

echo json_encode($response);

?>
```

Bibliografia

1. Agostino di Tagaste. *Confessioni* (401), Rizzoli Libri S.p.A. / BUR Rizzoli, Milano, 2016.
2. Benjamin W., *L'opera d'arte nell'epoca della sua riproducibilità tecnica* (1936), in Pinotti A., Somaini A. (a cura di), *Aura e choc. Saggi sulla teoria dei media*, Einaudi Editore, Torino, 2012.
3. Bergson H., *Saggio sui dati immediati della coscienza* (1889), Raffaello Cortina Editore, Milano, 2004.

   -, *Durata e Simultaneità* (1922), Raffaello Cortina Editore, Milano, 2004.
4. Bergson H., James W., *Durata reale e Flusso di coscienza*. *Lettere e altri scritti* (1902-1939), Raffaello Cortina Editore, Milano, 2014.
5. Garda M., *Teorie del tempo musicale nella modernità*, in G. Borio, C. Gentili (a cura di), *Storia dei concetti musicali*. *Armonia, tempo*, Carocci editore, Roma, 2007.
6. Husserl E., *Per la fenomenologia della coscienza interna del tempo* (1893-1917), Franco Angeli, Milano, 2011.
7. Kramer J. D., The Time of Music, Schirmer Book, New York, 1988.

   - *Il tempo musicale*, in Nattiez J.J. , *Enciclopedia della musica vol. II. Il sapere musicale*, Einaudi, Torino 2002
8. Langer S., *I principi della creazione artistica* (1950), Editrice Morcelliana, Brescia, 2017.

   -, *Sentimento e forma* (1953), Feltrinelli Editore, Milano, 1965.

   -, *Problemi dell'Arte* (1953-1956), Aesthetica Edizioni, Palermo, 2013.
9. Ricoeur P., *Tempo e racconto. Volume 1* (1983), Editoriale Jaca Book, Milano, 2008