Introduzione

Nella nostra società possiamo certamente dire, senza rischiare di incorrere in pareri discordanti, che la tecnologia ricopra un ruolo fondamentale nella vita di tutti noi.
Perfino coloro che inizialmente nutrivano un atteggiamento restio nei confronti dell'innovazione tecnologica, col passare degli anni, si stanno sempre più lasciando influenzare e coinvolgere da questa nei più svariati ambiti.
Possiamo pertanto dire che la tecnologia sia un'esperienza che può costituire una singolare via di interazione con il mondo circostante e che sempre più tende a divenire una vera e propria necessità.
Partendo da una riflessione sul rapporto che c'è tra tecnologia e mondo reale e prendendo in esame i modi che questa oggigiorno ci fornisce per vivere questa esperienza tecnologico-interattiva, si è ritenuto di interesse per il presente elaborato cercare, non certamente seguendo un approccio del tutto inesplorato, di portare anche l'esperienza di interazione con un oggetto sonoro ad un livello decisamente più intuitivo utilizzando strumenti a tutti reperibili, in modo da abbracciare una più ampia fascia di interesse.
Oltre all'interfaccia implementata in questa sede, è stato scelto come strumento chiave di interazione, reso tale grazie alla sua versatilità e all'ampia gamma di dispositivi che comprende al suo interno, il Wii Remote (o Wiimote) controller della ormai nota Nintendo Wii.
Grazie alla tecnologia Bluetooth è stato possibile servirsi di quest'ultimo come principale fonte di input per l'applicazione e quindi di controllare le funzioni implementate nell'interfaccia, con il solo uso del controller.

1. La ripetizione

«L'organizzazione delle forme in musica obbedisce ad una legge fondamentale: la ripetizione».
(Jules Combarieu)

Vibrazione e ripetizione periodica sono il fondamento di materia ed energia.
Ripetizione ritmica, oscillazione e pulsazione sono qualità dominanti nonché fondamentali della natura non solamente legata all'aspetto sonoro e musicale.
Persino uno sguardo microscopico mostra che le onde, cuore di ogni evento sonoro, presentano una certa periodicità e quindi ripetizione, e di quelle che questa periodicità non le caratterizza si può dire che il loro periodo tenda all'infinito.
Insomma nell'ambito sonoro prima e musicale poi, la ripetizione costituisce una delle caratteristiche fondamentali a partire dalla quale definiamo molti aspetti legati a questi ambiti tanto che non si può parlare di musica se non si parla di ripetizione.
Ripetere semplici ritmi, melodie, frammenti musicali prima e strutture sempre più complesse poi, divenne pratica fondamentale nonché popolare in tutte le culture, nonostante molteplici differenze intercorressero tra queste, e non solamente, come molti pensano, perchè ripetere determinate cellule musicali o ritmiche rendesse più facile la memorizzazione di brani o addirittura di intere opere, ma perchè la tecnica della ripetizione crea da tempo immemorabile grande fascino e attrattiva nell'uomo e, se correttamente usata, rende indimenticabile alle orecchie dell'ascoltatore un brano o una qualsivoglia composizione sonora e musicale proprio per questa forza intrinseca.
Vista l'importanza della ripetizione nell'ambito che ci preponiamo di analizzare, e cioè quello musicale, possiamo dire che finchè si parlerà di musica parleremo sempre in una certa misura di ripetizione.

2. Il Loop

«Ho scoperto che la musica più interessante in assoluto consiste semplicemente nell'allineare i loop all'unisono e lasciarli uscire lentamente fuori fase tra loro.»
(Steve Reich)

Il loop è un ciclo, una sorta di anello in cui la fine si ricongiunge all'inizio in un eterno scorrere tra i due punti.
Si può indubbiamente dire che il loop sia, innanzitutto, ripetizione: ogni passaggio riproduce in maniera del tutto identica il primo in modo tale che alla fine il ciclo risulti essere costituito da una serie di copie del primo elemento, riprodotte una di seguito all'altra senza interruzioni.
Il loop musicale consiste tecnicamente nella ripetizione meccanica di un frammento registrato, sia esso una sequenza di suoni prodotti da strumenti musicali oppure un qualsiasi altro elemento sonoro non strumentale.
Nel primo caso si ottiene una melodia potenzialmente infinita, nel secondo si ottiene invece l'effetto di far emergere le qualità musicali dell'elemento sonoro, dato dal suo ascolto ripetuto, che porta gradualmente a dimenticare le sue origini ed il suo contesto reale.
Camilleri, nel classificarne l'impiego musicale individua quattro tipologie di loop *[9]*.

- Il *loop elaborazione*, utilizzato per realizzare un brano (o parte di esso) mediante ripetizioni di frammenti di varia durata, con differenti gradi di sovrapposizione;
- Il *loop testurale*, sempre secondo la classificazione di Camilleri, è invece un evento sonoro, solitamente di lunga durata che viene utilizzato come sfondo sul quale articolare eventi sonori che invece mutano il proprio comportamento;

- I *loop strumentali* sono invece riff ripetuti meccanicamente senza alcuna manipolazione delle sorgenti sonore che danno vita a sequenze sonore ripetute ciclicamente;
- Il *loop gestuale* è infine una modalità di impiego della ripetizione ciclica che vuole perturbare una struttura sonora statica o in sviluppo.

2.1 Breve storia del loop e delle sue applicazioni nell'ambito musicale

«Although repetition is a major force in music it was never used in this way before» (Terry Riley)

Musicalmente il loop trova la sua prima applicazione per così dire "cosciente" nell'ambito dei lavori sperimentali di Pierre Schaeffer, strettamente legati all'avvento dei mezzi di registrazione e di riproduzione audio primo tra questi il *grammofono* seguito poi dai *registratori a nastro*.
Già prima di Schaeffer, diversi sperimentatori avevano esplorato idee che mettevano in gioco il suono o si servivano di rumori come materiale proprio dell'ambito musicale in modo da soddisfare particolari esigenze compositive, tra questi va ricordato certamente Edgard Varèse e la sua famosa utopia del "suono organizzato".

Tuttavia nessuno di loro ebbe l'idea di utilizzare la registrazione (seppure accessibile e in evoluzione dagli anni '10) al fine di creare una vera "arte dei suoni fissati" .

Nel 1948 Pierre Schaeffer, diventa animatore di un piccolo gruppo di ricerca all'interno della RTF (Radio Télévision Française) e proprio questa attività gli permise di utilizzare il vasto archivio discografico della radio e di cominciare a fare esperimenti su suono e rumore, ma soprattutto cominciò a maturare delle convinzioni

innovative nell'ambito della composizione musicale.

Inventa così una nuova forma di espressione artistica che lui stesso chiamerà *Musique Concrète* (Musica Concreta).

Schaeffer parla di *Musique Concrète* intendendo il suono nella sua completezza; ovverosia il fatto di ascoltare il suono in tutti i suoi aspetti (attacco sonoro, durata, inviluppo, densità di massa sonora, andamento, timbro, frequenza, ampiezza, …).

Tale espressione si collocava in contrapposizione all'idea di "astrazione" che secondo lo stesso Schaeffer caratterizzava l'approccio musicale dominante (quello della musica strumentale), cioè, il pensare la musica per criteri astratti (armonia, contrappunto, notazione, dispositivi logici, ...) piuttosto che elaborarla concretamente attraverso suono e ascolto.

Fig 1 - Pierre Schaeffer lavora con un Fonogeno nel suo studio (1948)

La *Musique Concrète* nasce quindi da materiale sonoro preesistente (rumori o suoni strumentali registrati mediante un microfono) modificato, manipolato, trasformato e giustapposto in studio.

Egli si lancia perciò in un'avventura musicale interamente nuova, nata da mesi di sperimentazione e di osservazione dei suoni "fissati" che i processi di registrazione gli consentono di riascoltare a piacimento, mentre appositi strumenti come banchi di

mixaggio, moduli di riverberazione, filtri, sound transposer, ne consentono la modifica dando vita ad un ampio ventaglio di sonorità allora sconosciute.

Nell'utilizzare più piatti di giradischi, Schaeffer scrive:

«21 aprile 1948: se privo i suoni del loro attacco, ottengo un suono differente; d'altra parte, se compenso la caduta di intensità grazie a un potenziometro, ottengo un suono continuo di cui sposto l'inviluppo a volontà. Registro così una serie di note costruite nello stesso modo, ciascuna su un disco. Disponendo questi dischi su dei pick-up, posso, grazie al gioco di chiavi di contatto, suonare queste note come desidero, in successione o simultaneamente. […] Siamo degli artigiani. Il mio violino, la mia voce, li ritrovo in tutto questo bazar di legno e latta, e nel campanello della bicicletta. Cerco il contatto diretto con la materia sonora, senza interposizione di elettroni»

Schaeffer capisce quindi rapidamente che la registrazione e la manipolazione dei suoni possono dar vita ad una nuova forma d'arte ed inaugura un nuovo approccio che pone l'ascolto al centro del lavoro compositivo rovesciando completamente il processo di creazione e conducendo il compositore a procedere per tentativi, esperimenti, intuizioni, oltre a portare un'attenzione nuova alla materialità viva del suono, alla sua sostanza, alla sua qualità.

Il compositore classifica perciò le registrazioni accumulate, e opera su di esse delle scelte, una ripartizione e quindi molteplici trasformazioni in uno studio equipaggiato con numerosi apparecchi prodotti dall'evoluzione tecnologica degli ultimi decenni: montaggio, inversione, loop, trasposizione, campionamento, compressione, riverberazione, eco, ritardo, filtraggio, mixaggio.

Lo studio di registrazione guadagna grazie a Schaeffer una posizione di centralità assoluta nella creazione di opere, non solo l'ambiente in cui materiale preesistente viene fissato, ma uno strumento di vera creazione e luogo di innovazione.

Se inizialmente questo lavoro di fissaggio prima e di manipolazione poi veniva eseguito con l'ausilio di grammofoni o giradischi, l'introduzione del nastro magnetico

porta a livelli ancora più elevati le possibilità di manipolazione ed interazione con il materiale sonoro.

I primi *registratori a nastro* cominciarono ad essere disponibili per Schaeffer e il suo gruppo di ricerca intorno al 1949 ma il loro funzionamento era ancora troppo inaffidabile rispetto ai sistemi fino a quel momento utilizzati tanto che anche *Symphonie pour un homme seul (1950-1951)* fu composta soprattutto utilizzando suoni presenti su dischi.

Ma quando le nuove macchine cominciarono a funzionare in maniera efficiente, le tecniche della *Musique Concrète* subirono una notevole evoluzione e la stessa *Musique Concrète* cominciò allora a fondersi e talvolta a prendere il nome di *Tape Music* ereditando questo epiteto appunto dal nuovo strumento di registrazione a nastro magnetico.

Un ampio range di nuove tecniche di manipolazione come l'aumento della velocità di riproduzione, la possibilità di effettuare tagli in maniera molto più semplice e funzionale con tecniche chiamate di micro-editing (editing di porzioni di suono dell'ordine dei millisecondi) furono allora possibili e sicuramente di più facile realizzazione.

Da supporto di registrazione il nastro magnetico diviene quindi anche mezzo creativo non solo di fissaggio ma di vera creazione.
Nei tardi anni '50 e all'inizio degli anni '60, i compositori cominciarono ad usare registratori a nastro e altri sistemi elettronici anche nelle performance live e contemporaneamente furono fondati in più città (come Parigi e Colonia) studi di musica elettronica e *Tape Music*, che divennero centri fondamentali per lo sviluppo di questi nuovi generi musicali.

Una delle nuove tecniche che viene sempre più ad affermarsi con l'utilizzo dei registratori a nastro è proprio quella del loop.
L'origine dei tape loops non è perfettamente chiara, non c'è un vero e proprio

“ideatore” di questa tecnica, ma la semplice idea di prendere un frammento di nastro registrato e creare un anello con questo ottenendo quello che noi chiamiamo loop, probabilmente risale ai primi utilizzatori dei registratori a nastro.

Fig 2 Registratore a bobina Revox

Benchè possa sembrare una tecnica semplice, quasi sterile dal punto di vista musicale, viene ben presto utilizzata in innumerevoli modi e situazioni nonché generi musicali e, insieme alle tecniche innovative e all'esperienza illuminante dello stesso Schaeffer, influenza il mondo stesso della composizione strumentale contemporanea.

Il “solco chiuso” (concetto ereditato dallo stesso Schaeffer per esprimere quello che noi chiamiamo loop) si inserisce poco a poco nel vocabolario musicale contemporaneo influenzandone la musica di molti esponenti di una nuova corrente definita *Minimalista*.

Il *Minimalismo* si basa sul semplice assunto che niente sia uguale a sé stesso.

Partendo da questo presupposto e considerando che, contrariamente a quanto avviene

per le altre arti, l’applicazione di tale concetto alla musica si scontra inevitabilmente con una dimensione temporale che è completamente assente in altri ambiti, il minimalismo si evolve qui in maniera del tutto particolare, puntando proprio sulla filosofia della ripetizione.

Sfruttando il fatto che non solo la nostra percezione riconosce i suoni e si abitua ad essi ma che questo continuo processo fa sì che un suono, se pur apparentemente identico, se ripetuto nel tempo ci risulti cangiante, sempre diverso, la corrente minimal-ripetivitista ripone nella ripetizione e nelle sue infinite variazioni la linfa della propria esistenza.

L’elemento innovativo introdotto dalla musica di Philip Glass, Terry Riley, Steve Reich e dei maestri minimalisti, è, almeno in un primo periodo, lo strutturare interamente le composizioni su materiale ridotto all’osso, e il basarne lo sviluppo sulla micro-variazione; principio da cui poi ciascun compositore svilupperà in seguito il suo particolare stile con le proprie caratteristiche tecniche compositive.

Anche il mondo della popular music fu fortemente attratto dalla tecnica del loop tanto che molti gruppi tra la fine degli anni '60 e l'inizio degli anni '70 ne sperimentarono l'utilizzo, tra questi The Beatles, The Who (che nel loro brano Baba O’Riley rendono inoltre omaggio, come si nota dal titolo, allo stesso Terry Riley), Pink Floyd e molti altri.

Molte furono inoltre le invenzioni legate alla tecnica del loop, in questo contesto ne citiamo principalmente due.

La prima nasce nei primi anni '70 grazie ai musicisti Brian Eno e Robert Fripp, e prende il nome *Frippertronics* che consiste in un sistema in grado di creare tape loops durante performance live.

Esso evolve a partire da un sistema di tape loop originariamente sviluppato negli studi di musica elettronica dei primi anni '60 e inizialmente usato da compositori quali Terry Riley e Pauline Oliveros.

Frippertronics è un sistema di delay analogico che consiste in due registratori a bobina Revox situati l'uno accanto all'altro in modo che il nastro impegni inizialmente la bobina del primo registratore per poi essere catturato da quella del secondo.

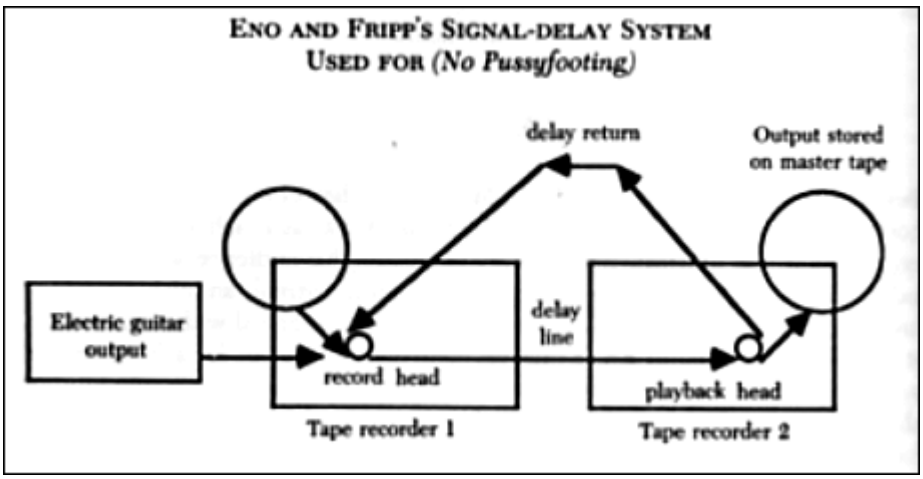

Fig 3 - Schema descrittivo di un Frippertronics

Questi due registratori sono configurati in modo che il suono venga registrato sul nastro della prima bobina per poi essere inviato alla bobina del secondo registratore che ne riproduce la registrazione del primo mentre questo effettua una continua sovraincisione.

Questo processo viene ripetiuto un numero arbitrario di volte in base a quante vogliono essere le sovraincisioni e si può comodamente decidere la quantità di delay (ritardo) semplicemente variando la distanza tra i due registratori e cambiando così il tempo di percorrenza del "circuito". Robert Fripp utilizzò questa tecnica soprattutto dal vivo, per effettuare continue sovraincisioni in tempo reale usando una chitarra elettrica come fonte sonora, creando un vero e proprio tappeto sonoro.

La seconda invenzione citata, che per la sua natura e struttura può essere considerata come una rudimentale Loop Machine, è lo strumento noto con il nome di Mellotron.

Apparso per la prima volta sul mercato nel 1964, il Mellotron Mark I prende il suo nome dalle parole Melody ed Electronic.

Nato in una piccola fabbrica di materiale bellico, il Mellotron è uno strumento a tastiera divenuto popolare tra la fine degli anni '60 e la prima metà degli anni '70 grazie a solisti e gruppi pop e rock come The Beatles e Moody Blues inizialmente, Genesis, Rick Wakeman, Yes, Jethro Tull, King Crimson successivamente.

Fig 4 - Mellotron

Inizialmente creato e pubblicizzato come strumento da casa similmente ad un organo da salotto, è considerato l'antenato dei moderni campionatori, poiché la pressione di ciascun tasto innesca la riproduzione di un segmento di nastro magnetico su cui è stato precedentemente registrato un suono (solitamente di archi, cori o flauti, nonostante sia possibile effettuare registrazioni di qualsiasi strumento).

Inizialmente la durata di ciascun “campione” era di otto secondi, ogni nastro doveva essere lungo esattamente come gli altri e terminata la riproduzione del segmento era necessario alzare il dito dal tasto, il che azionava una molla che riavvolgeva il nastro per portarlo alla posizione iniziale, per poi ripremerlo se si voleva continuare la riproduzione.

Agli albori ogni modello poteva riprodurre solo un suono, e cioè quello corrispondente ai nastri installati, ma successivamente fu creato un sistema a

"cartucce" grazie al quale era possibile smontare il blocco dei nastri e sostituirlo con un altro differente cambiando così le sonorità del Mellotron.

Un esempio del suo tipico suono si può ascoltare nei primi secondi della canzone dei The Beatles, Strawberry Fields Forever, dove lo strumento viene suonato da Paul McCartney.

Esternamente alla scena dell'avanguardia musicale e strumentistica i tape loops furono ben presto impiegati anche negli studi radiofonici e nell'industria cinematografica, dove furono utilizzati per la sincronizzazione e per le colonne sonore.

Con l'avvento dell'era digitale anche le loop machine e il concetto stesso di loop subiscono un notevole cambiamento.

Non più inteso come anello se non solo in maniera meramente teorica, il loop, che prima veniva fisicamente realizzato grazie ad un meticoloso lavoro di editing, diventa ora una semplice pratica accessibile a chiunque possieda un computer, tramite l'utilizzo di un semplice programma se non di un semplice tasto integrato pressochè in ogni programma di riproduzione sonora.

Questa riduzione dei costi legati alla realizzazione di loop, che prima implicava l'utilizzo di specifici macchinari come registratori a nastro, banchi di mixaggio, moduli di effetti, presenti solamente in alcuni studi di registrazione all'avanguardia, porta con se di pari passo la diffusione sempre maggiore di questa pratica prima rivolta ad una fascia elitaria di ingegneri o esperti nell'ambito musicale e di conseguenza un notevole sviluppo della musica basata sulla pratica del loop.

Le tecniche di campionamento ed editing digitale cominciano a prendere il sopravvento sulla scena musicale a partire dai primi anni '80 per poi conoscere uno sviluppo sempre crescente sino ai giorni nostri.

Diventano sempre più di dominio comune programmi come Pro Tools, Fruityloops, Logic Studio, Cubase con i quali non solo diventa possibile realizzare graficamente

un loop tramite una giustapposizione delle tracce su una scala temporale a video, ma che offrono strumenti avanzati per la manipolazione sonora con il semplice utilizzo di appositi plug-in.

Nonostante la tecnologia digitale offra una maggiore affidabilità nonchè minori costi per la realizzazione di loop, molti compositori e musicisti continuano a preferire un approccio analogico alla tecnica del loop che, sebbene comporti maggiore disponibilità di mezzi e strumentazione, offre un'esperienza unica e irripetibile di reale interazione con il materiale sonoro che si vuole elaborare.

3. Il Wiimote

Nel novembre 2006 viene lanciata da Nintendo la nuova console di settima generazione che in quanto quinta console di casa Nintendo prende, nei primi mesi di sviluppo, il nome di *N5* per poi trasformarsi in *Revolution* e diventare solo in prossimità del suo lancio quella nota ai più come *Nintendo WII*. Ad un primo sguardo sembra non presentare grandi risorse che le permettano di distinguersi significativamente dalle concorrenti Sony Playstation 3 e Microsoft Xbox, ma ad un anno dal suo lancio diventa il leader nel mercato delle console della sua generazione, vendendo oltre venti milioni di esemplari in tutto il mondo.

Fig 5 - Nintendo Wii Remote

Questo successo è da attribuirsi soprattutto all'innovativa esperienza di gioco offerta dalla nuova console Nintendo, grazie all'ausilio del nuovo controller chiamato ufficialmente *Wii Remote,* ma anche noto come *Wiimote.*

Il *Wiimote* utilizza un approccio differente da quello del tradizionale controller, nel tentativo di risultare interessante per un pubblico più vasto non di soli appassionati.

Ad un primo sguardo presenta la forma di un comune telecomando da televisione, è progettato per essere tenuto con una sola mano al contrario dei comuni controller ed essendo simmetrico, appare inoltre ugualmente utilizzabile da destri e mancini.

Il controller oltre ai pulsanti, unica fonte di input dei comuni controller (che in questo caso sono undici oltre ad un bottone power), presenta in uno chassis di dimensioni contenute (14.5 cm x 3,5 cm x 3 cm circa): un accelerometro a tre assi che ne permette di stabilire inclinazione e rotazione utili nell'esperienza ludica, una camera ad infrarossi ad alta definizione per il puntamento, un altoparlante, un motore vibrante ed è dotato inoltre di tecnologia Bluetooth per la connessione con la console.

Questo insieme di caratteristiche innovative accorpate in un unico controller, reso accessibile ad un'ampia fascia di mercato grazie anche al suo prezzo competitivo, ha stabilito questo primato della *Nintendo WII* sulle altre console.

Nonostante non ci sia documentazione ufficiale riguardante le tecnologie implementate nell'innovativo controller, un lavoro di reverse engineering ha permesso di conoscerne, seppure in parte, le principali caratteristiche.

Come detto il *Wiimote* presenta, nella sua facciata anteriore, undici bottoni di cui un pad direzionale digitale posizionato sulla parte superiore del controller e procedendo verso il basso un grosso pulsante A e sotto di esso una fila di tre piccoli tasti: - , Home e +.

Vicino all'estremità inferiore del controller sono presenti due bottoni aggiuntivi contrassegnati dai numeri 1 e 2 e sotto questi bottoni sono posizionati quattro led di colore blu.

Nella parte posteriore è invece presente un bottone B dalla forma di un grilletto (trigger B) *[10]*.

Il *Wiimote*, costituendo il principale input device della console *Nintendo WII,* si serve dello standard Bluetooth HID protocol per comunicare con questa, scambiando pacchetti di 22 byte chiamati *report* che consentono di tenerne aggiornato lo stato.

Proprio questo tipo di tecnologia ha reso possibile connettere il *Wiimote* ad un qualsiasi dispositivo Bluetooth, nel nostro caso un notebook, e di studiarne ulteriori potenzialità oltre a quelle già note relative al settore videoludico.

L'abbinamento è reso possibile semplicemente premendo contemporaneamente sul *Wiimote* i pulsanti 1 e 2 fintanto che i quattro led cominciano a lampeggiare (nella proporzione pari al voltaggio della batteria indicandone quindi lo stato di carica).
Ora il *Wiimote* può essere richiesto ed abbinato grazie al Bluetooth HID driver sull'host.
Quando viene cercato e trovato come dispositivo, utilizzando il Bluetooth Service Discovery Protocol (SDP), il *Wiimote* restituisce una grande quantità di informazioni. Eccone alcune mostrate in figura (fig 6):

Name	Nintendo RVL-CNT-01
Vendor ID	0x057e
Product ID	0x0306
Major Device Class	1280
Minor Device Class	4
Service Class	0
(Summary of all Class Values)	0x002504

Fig 6 - Informazioni inviate dal Wiimote in fase di abbinamento con un dispositivo Bluetooth

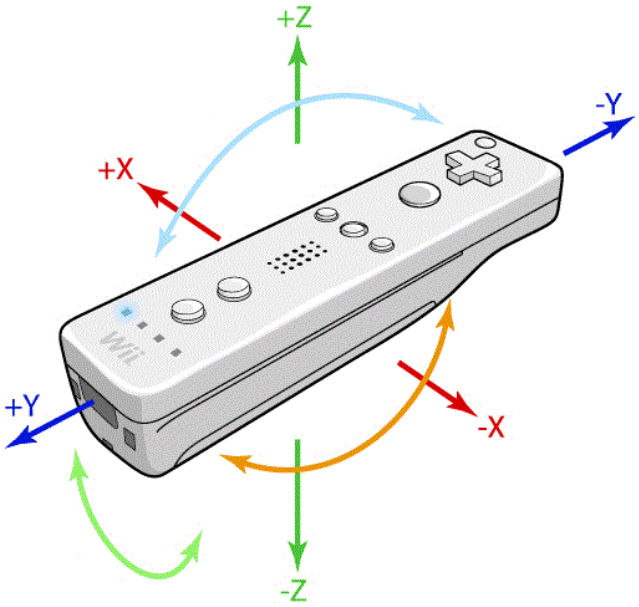

Fig 7 - Rappresentazione degli assi dell'accelerometro presente nel Wiimote

3.1 Accelerometro

L'accelerometro incluso nel *Wiimote* è un accelerometro lineare a tre assi costruito per percepire accelerazioni in un range di +/- 3 g con 8 bit per asse e una frequenza di aggiornamento di 100Hz.

3.2 Camera IR

Come anticipato il *Wiimote* è inoltre dotato di un'IR camera monocromatica (camera a raggi infrarossi).

Le specifiche esatte relative alla camera non sono state pubblicate ma da uno studio effettuato sui componenti sembra avere una risoluzione di 1024 x768 pixels, una frequenza di aggiornamento di 100 Hz e processing di immagine integrata.

La camera è in grado di riconoscere e tracciare fino a 4 oggetti mobili e questi sono gli unici dati disponibili all'utente rendendo perciò impossibile l'uso di questa camera per riprodurre un'immagine convenzionale.

4. DirectSound

DirectSound è un componente software parte delle librerie DirectX, che racchiude una collezione di API per lo sviluppo semplificato di videogiochi in ambiente Windows, inizialmente distribuite dai produttori di giochi stessi, ora incluse da Microsoft nei suoi sistemi operativi a partire dal 1995.

DirectSound nello specifico fornisce una diretta interfaccia tra l'applicazione e i driver della scheda audio, rendendo possibile la produzione di suoni e la riproduzione di musica da parte delle applicazioni stesse.

Fornendo la possibilità essenziale di interfacciarsi direttamente con la scheda audio e di fornirgli dati sonori, rende possibili operazioni come la registrazione e il mixaggio di suoni, nonché la modifica di alcuni parametri relativi ai dati audio e l'aggiunta di effetti sonori offerti sempre da *DirectSound*.

DirectSound oltre a supportare l'audio posizionale, che simula quindi la spazialità del suono in 3D, consente a più applicazioni di condividere l'accesso alla scheda audio contemporaneamente.

I suoni vengono inseriti in appositi buffer creati ad hoc chiamati *buffer di ingresso* (Secondary Sound Buffers), dopodiché vengono mixati insieme nell'unico *buffer di uscita* (Primary Sound Buffer) (fig 8).

I *Secondary Buffers* possono essere *statici* e quindi ricevere i dati audio da un percorso specificato nella fase di definizione del buffer o *dinamici*, quindi catturare uno stream da una fonte esterna come ad esempio un microfono.

DirectSound si può avvalere delle funzionalità della scheda audio, nel caso supporti gli effetti richiesti, tra i quali il chorus, l'echo, l'equalizzazione, il panning, il riverbero, la distorsione, vari effetti tridimensionali e la simulazione dell'effetto

Doppler.

Dopo molti anni di sviluppo, attualmente *DirectSound* si presenta come un'API molto evoluta, che fornisce molteplici possibilità di gestione dei dati audio, come la riproduzione multicananle ad alta definizione.

Inizialmente realizzata per essere utilizzata in ambiente videoludico per la riproduzione e gestione di suoni, un certo numero di applicazioni audio professionali ora si servono di *DirectSound* e delle numerose possibilità che offre.

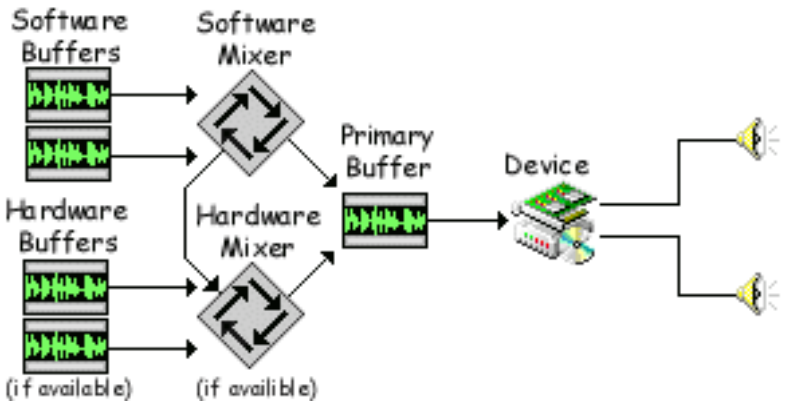

Fig 8 - Architettura DirectSound

5. Progetto Analogic Wii Loop Machine

5.1 Descrizione dell'interfaccia utente

Dopo aver cercato di definire cosa si intende con il termine loop, tracciandone una breve storia legata al suo uso nell'ambito musicale, e dopo avere successivamente presentato lo strumento *Wiimote* e le sue possibili implicazioni in campo informatico e in questo caso anche musicale, passiamo ora a descrivere quello che è stato realizzato tramite l'ultilizzo di questo controller, con l'aiuto di un linguaggio quale C# che ne permettesse l'interfacciamento oltre che la realizzazione dell'applicazione stessa.

Si è deciso pertanto di riprodurre una sorta di Loop Machine in cui la riproduzione delle tracce, l'attivazione di effetti e la modifica del materiale sonoro potesse essere controllata tramite *Wiimote*.

Come accennato un loop può essere inteso come un "anello" formato da un frammento musicale ripetuto ciclicamente sul quale si può agire modificando parametri legati a volume, tempo o frequenza tramite controlli analogici nel caso di un tape loop o digitali come nel nostro caso.

Ma può essere anche inteso come un ciclo formato a partire da un certo numero di tracce alle quali si vanno a sovrapporre ulteriori tracce o frammenti che vengono attivati dall'utente arbitrariamente, modificandone poi i relativi parametri.

In questa sede è stato scelto il primo approccio in quanto più fedele all'idea schaefferiana di loop e si è deciso di implementare, oltre ai controlli inerenti

frequenza, volume e balance, anche l'impiego di effetti sonori offerti in questo caso dallo strumento *DirectSound* in modo che agiscano sul loop in riproduzione precedentemente mixato.

Questo tipo di approccio cerca di ricreare anche in maniera molto sommaria quello che avveniva nelle prime Loop Machine, in cui l'anello di nastro veniva editato a priori scegliendo quali parti inserirvi al suo interno, per poi mandare in riproduzione ciclica il risultato dell'editing.

In questo caso infatti, non è possibile modificare dinamicamente la traccia in riproduzione se non mixandone una nuova, ma è possibile interagire con essa modificandone in maniera arbitraria i parametri sopra descritti.

Per questi motivi è stato deciso di inserire nel titolo dell'elaborato la parola Analogic, che pur creando un ossimoro concettuale di fondo, si è ritenuto potesse richiamare un procedimento tipico dell'approccio analogico alla tecnica del loop.

Come strumento per la programmazione è stato usato Microsoft Visual Studio 2008 in ambiente C# e l'implementazione dell'interfaccia è stata incorporata in un Windows Form sul quale sono stati aggiunti bottoni e trackbar utili alla realizzazione delle features precedentemente scelte per l'implementazione.

Come visibile in figura (fig 9), nel form realizzato sono presenti i seguenti elementi :

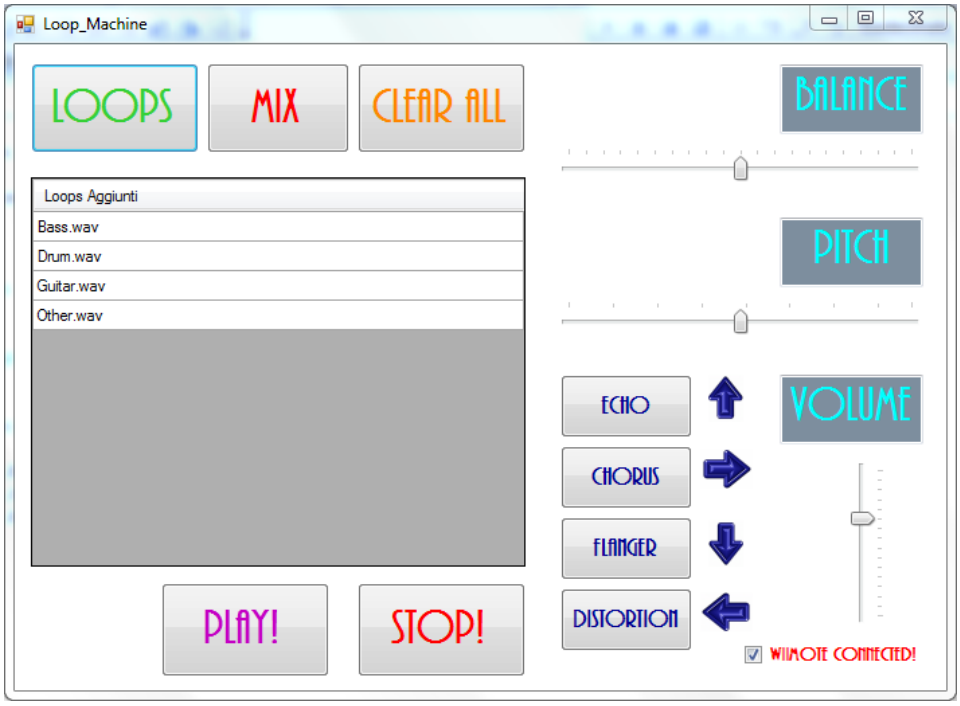

Fig 9 - Interfaccia grafica di Analogic Wii Loop Machine

- **Pulsante Loops:**

Creato per controllare l'apertura di una finestra di dialogo in modo che l'utente abbia la possibilità di navigare tra i file presenti su un qualsiasi supporto e di scegliere quali file mixare tra loro per ottenere la traccia che poi verrà riprodotta in modalità looping.

E' stato implementato a livello di codice un filtraggio riguardante l'estensione dei file che possono essere scelti attraverso la finestra di dialogo, in quanto l'operazione di mixing viene eseguita a partire da soli file in formato.wav (wave).

È stato inoltre reso impossibile selezionare una sola traccia dal momento che non è logico effettuare un mix a partire da un solo file sorgente per la natura stessa della nozione di mixaggio.

- **Pulsante Mix:**

Una volta selezionate le tracce che si desidera mixare premendo il tasto Mix si effettua l'operazione di mixaggio vera e propria. Questa operazione viene svolta da una classe separata integrata nel progetto di Analogic Wii Loop Machine.

Prima di effettuare il mix viene data nuovamente la possibilità all'utente di scegliere, attraverso una finestra di dialogo, la destinazione del file sul quale si vuole scrivere l'output ottenuto dal mix delle tracce precedentemente selezionate.

Anche in questo caso è stato implementato un sistema di filtraggio riguardante l'estensione del file di output sul quale mixare le tracce, dal momento che è necessario farlo su soli file di estensione .wav.

- **Pulsante Clear All:**

Viene utilizzato in modo che l'utente in caso di errore o di necessità possa ripulire il la griglia in cui vengono inserite le tracce prescelte per il mix.

- **LoopsGrid:**

Griglia adibita alla visualizzazione delle tracce selezionate per il mixaggio.

Divisa in righe sulle quali viene visualizzato il nome del file selezionato e la relativa estensione.

- **Pulsante Play:**

Pulsante utilizzato per gestire la funzione *buffer.Play ()* di *DirectSound*.

Premendo questo pulsante viene richiamato il file di output sul quale sono state precedentemente mixate le tracce scelte dall'utente e viene mandato in play il *Secondary Buffer* creato nell'inizializzazione di *DirectSound* e riempito con il file di output scritto in fase di mix.

- **Pulsante Stop:**

Pulsante utilizzato per gestire la funzione *buffer.Stop ()* di *DirectSound*.

Premendo questo pulsante viene stoppato il buffer che contiene il file wave di output.

- **Balance TrackBar:**

Oggetto trackbar scelto per il controllo del parametro relativo alla spazialità del suono quindi del cosiddetto Pan o Balance.

Anche in questo caso si tratta di un controllo implementato in *DirectSound* che viene però modificato dall'utente a livello grafico ed intuitivo grazie allo spostamento di questa trackbar o grazie allo strumento *Wiimote*.

Trascinando con il mouse la trackbar otteniamo dall'estremo sinistro valori che partono da un minimo di -10000 (suono spazialmente situato totalmente a sinistra) per arrivare fino a 10000 (suono spazialmente situato totalmente a destra) con progressione ad intervalli di 1000 (valore impostato nella proprietà TickFrequency della trackbar).

Il valore predefinito impostato è 0 che riproduce una situazione di centralità del suono riprodotto.

- **Pitch TrackBar:**

Oggetto trackbar scelto per il controllo del parametro relativo all'altezza del suono, ottenuto grazie ad una differente velocità di lettura del buffer contenente l'output del mix.

Anche in questo caso si tratta di un controllo implementato in *DirectSound* che viene però modificato dall'utente a livello grafico ed intuitivo grazie allo spostamento di questa trackbar o grazie allo strumento *Wiimote*.

Trascinando ad esempio con il mouse la trackbar otteniamo dall'estremo sinistro valori che partono da un minimo di 28200 fino ad arrivare ad un massimo di 60000

con progressione ad intervalli di 4100.

Questo causa quindi un rallentamento nella velocità di lettura dei campioni di pari passo con il descrescere dell'valore indice ed una conseguente accelerazione al crescere di questi.

Il valore predefinito è stato impostato a 44100 in modo che riproduca la velocità naturale del file campionato a 44100Hz.

- **Volume TrackBar:**

Oggetto trackbar scelto per il controllo del parametro relativo all'intensità del suono. Come nei due precedenti casi si tratta di un controllo implementato in DirectSound che viene però modificato dall'utente a livello grafico ed intuitivo grazie allo spostamento di questa trackbar o grazie allo strumento *Wiimote*.

Trascinando con il mouse la TrackBar otteniamo dall'estremo sinistro valori che partono da un valore minimo di -3000, a un valore massimo di 0, corrispondente all'intensità massima consentita da DirectSound, con progressione ad intervalli di 200.

- **Pulsante Echo:**

Pulsante che controlla l'attivazione e la conseguente disattivazione dell'effetto Echo messo a disposizione da *DirectSound* nell'ambito delle funzionalità riguardanti il Secondary Buffer.

- **Pulsante Chorus:**

Pulsante che controlla l'attivazione e la conseguente disattivazione dell'effetto Chorus messo a disposizione da *DirectSound* nell'ambito delle funzionalità riguardanti il Secondary Buffer.

- **Pulsante Flanger:**

Pulsante che controlla l'attivazione e la conseguente disattivazione dell'effetto

Flanger messo a disposizione da *DirectSound* nell'ambito delle funzionalità riguardanti il Secondary Buffer.

- **Pulsante Distortion:**

Pulsante che controlla l'attivazione e la conseguente disattivazione dell'effetto Distortion messo a disposizione da *DirectSound* nell'ambito delle funzionalità riguardanti il Secondary Buffer.

5.2 Bluetooth e Wiimote

Dopo aver descritto come si presenta ad un primo sguardo l'interfaccia grafica della Loop Machine realizzata, possiamo ora passare a descrivere i primi passi utili per connettere il *Wiimote*, attraverso il protocollo Bluetooth, con un computer.

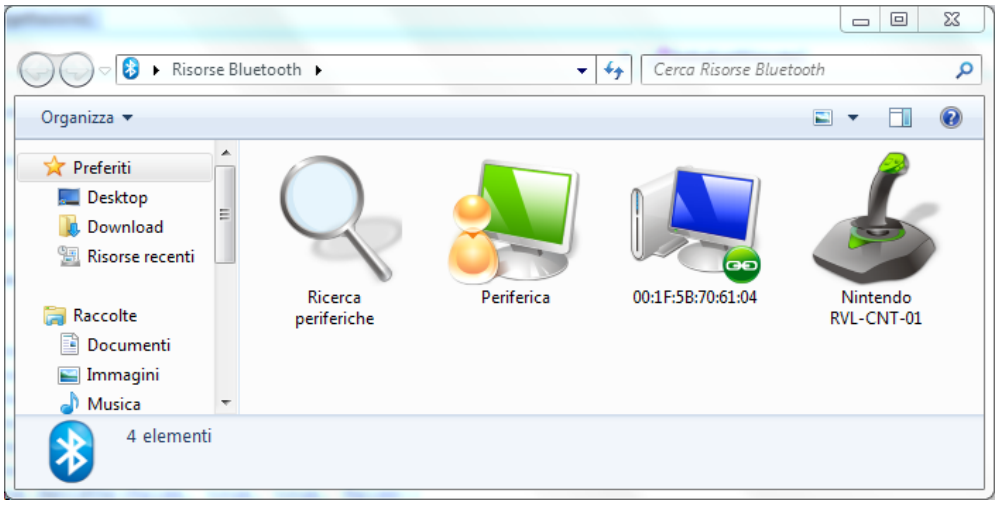

Fig 10 - Finestra delle risorse Bluetooth

Windows ci offre una procedura guidata molto efficiente ed intuitiva per l'abbinamento del dispositivo *Wiimote* o comunque di un qualsiasi dispositivo.

Selezionando tra le risorse Bluetooth la periferica da abbinare, in questo caso il *Wiimote*, che viene riconosciuto con il nome di Nintendo RVL-CNT-01 (come mostrato in fig 10), semplicemente facendo doppio click sulla relativa icona, il dispositivo Bluetooth del computer si pone in ricerca del device con cui è stato richiesto l' abbinamento.

Nell'arco di qualche secondo è perciò necessario provvedere alla pressione contemporanea dei pulsanti 1 e 2 sul *Wiimote* in modo da porlo in "modalità visibile".

Se il processo termina con esito positivo, dopo questo breve iter il *Wiimote* è perciò effettivamente abbinato al computer ed è possibile per i due dispositivi scambiarsi informazioni.

5.3 Inizializzazione del Wiimote

Nel nostro caso, per poter raggiungere lo scopo prefissato non era sufficiente abbinare i dispositivi tra loro dal momento che l'implementazione richiedeva che il nostro applicativo intercettasse messaggi come la pressione di tasti per poi assegnare ad ogni evento una precisa funzione nell'ambito della Loop Machine.

A qusto punto è stato necessario servirsi di una libreria apposita per l'intercettazione degli eventi lato *Wiimote* da riutilizzare poi secondo le nostre esigenze per assegnarvi specifiche funzioni.

Tra le librerie disponibili in rete la nostra scelta è ricaduta sulla *WiimoteLib - .NET Managed Library for the Nintendo Wii Remote* scritta da Brian Peek e reperibile

gratuitamente al seguente indirizzo *[11]*:

http://www.brianpeek.com/blog/pages/wiimotelib.aspx .

Per il nostro fine è stata utilizzata la versione scritta in linguaggio C#.

Lo scopo della libreria è quello di occuparsi della gestione e dell'interazione tra utente e dispositivo, tenendo in memoria lo stato della periferica, informando l'utente di questo stato e garantendo i metodi necessari per l'invio di richieste al fine di modificarlo.

I metodi riguardano principalmente i pulsanti, l'accelerometro e la camera a raggi infrarossi e consistono nell'invio di un pacchetto di report di richiesta contenente il tipo di dato che si vuole ricevere.

Una volta inoltrata la richiesta sarà il *Wiimote* ad inviare automaticamente le informazioni ad ogni cambio di stato.

Una volta scaricata quindi la WiimoteLib viene aggiunta tra i riferimenti del progetto in formato .dll e viene aggiunta la stringa seguente in modo da permetterne l'utilizzo:

```
using WiimoteLib;
```

È ora possibile richiamare i metodi e le funzioni definite nella libreria in modo da poter utilizzare comodamente le funzioni necessarie all'implementazione dell'interfaccia.

Per prima cosa c'è bisogno che la nostra applicazione si connetta al *Wiimote* e viene a questo scopo creato un apposito metodo chiamato `InitializeWiimote()` nel quale viene inizialmente creato un'handler (che consiste in un gestore di eventi utile ad intercettare i cambiamenti di stato) e poi richiamati una serie di comandi dalla libreria, utili per l'inizializzazione appunto del *Wiimote*.

Per prima cosa dobbiamo, come detto, connettere il *Wiimote* all'applicazione e per

fare questo utilizziamo l'istruzione `wiimote.Connect()` ereditata dalla WiimoteLib.

Successivamente passiamo all'inizializzazione di tutte quelle fonti di input necessarie alla nostra implementazione impostando a true tutti gli InputReport a noi necessari e quindi quello relativo alla pressione dei pulsanti, dell'accelerometro e infine settando i LED presenti sul *Wiimote* con una configurazione arbitraria, in questo caso in modo che i due LED centrali si accendano una volta effettuata la connessione.

Questo processo di connessione di cui stiamo parlando, non ha niente a che vedere con l'abbinamento Bluetooth descritto sopra, si tratta in questo caso di una connessione tra il *Wiimote* e l'applicazione in modo che questa possa intercettare i cambiamenti di stato del controller e assegnarli a funzioni specifiche.

La funzione try è seguita da un catch che intercetta il caso in cui non vada a buon fine la connessione e lancia un messaggio di errore, anch'esso richiamato dalla libreria, prima di chiudere la finestra (this.Close()).

```
private void InitializeWiimote()
{
    wiimote.WiimoteChanged += new EventHandler<WiimoteChangedEventArgs>(this.wiimote_WiimoteChanged);
    try
      {
          wiimote.Connect();
          wiimote.SetReportType(InputReport.Buttons, true);
          wiimote.SetReportType(InputReport.ButtonsAccel, true);
          wiimote.SetLEDs(false, true, true, false);
      }

    catch (Exception x)
      {
        MessageBox.Show("Exception: " + x.Message);
        this.Close();
      }
}
```

Fig 11 - Esempio di codice che descrive il metodo InitializeWiimote

5.4 Inizializzazione ed utilizzo di DirectSound

Date le notevoli possibilità offerte in merito alla gestione delle fonti sonore, *DirectSound* è stato scelto come strumento contestualmente al progetto Analogic Wii Loop Machine.

Dopo una prima fase di studio e di sperimentazione si è arrivati alla conclusione che *DirectSound* soddisfacesse le necessità relative alla riproduzione audio e alla gestione di effetti sonori, pertanto si è passati all'implementazione reale della parte sonora all'interno del codice C#.

Per poter utilizzare questa API va innanzitutto aggiunto al progetto, il riferimento ad essa relativo e poi aggiunta la stringa seguente nelle direttive using in modo che possano essere richiamati i metodi propri di *DirectSound*:

```
using Microsoft.DirectX.DirectSound;
```

Ora è necessario creare un device, cui è stato dato il nome `dSound`, dove per device si intende un oggetto che controlla la gestione di *DirectSound:*

```
dSound = new Device();
```

Si passa poi ad impostare il CooperativeLevel del device appena definito, e cioè una delle opzioni utili per la sua creazione. Nel nostro caso la scelta è ricaduta sul parametro priority che sfrutta, se disponibile, l'accelerazione hardware:

```
dSound.SetCooperativeLevel(button1,CooperativeLevel.Priority);
```

Ora possiamo creare oggetti per la riproduzione di brani che in DirectSound si chiamano buffer, in particolare quelli a noi utili prendono il nome di Secondary Buffers; possiamo crearne moltissimi ed ognuno indipendente per volume e settaggi.

Ogni oggetto SecondaryBuffer ha tra gli argomenti, come riportato nel codice d'esempio sottostante, il path del file che si vuole aggiungere al buffer (trattandosi in questo caso di un buffer statico) che si occuperà poi di riprodurlo:

```
sound = new SecondaryBuffer ( "C:\\audio.wav", d, dSound );

d  =  new BufferDescription ();
```

I flags sono invece ulteriori opzioni sempre relative al buffer che vanno però settati in base ai controlli che l'utente vuole avere su questo.

Tra quelli possibili noi abbiamo scelto di attivare controlli relativi al volume, alla frequenza (quindi alla velocità di lettura dei campioni del file audio) e al Pan (quindi alla differenza di output tra canale destro e sinistro o bilanciamento del suono), mentre l'ultimo flag `ControlEffects` viene aggiunto per controllare gli effetti che si vogliono aggiungere al buffer:

```
d.Flags = BufferDescriptionFlags.ControlVolume|
          BufferDescriptionFlags.ControlFrequency|
          BufferDescriptionFlags.ControlPan|
          BufferDescriptionFlags.ControlEffects;
```

Tra i comandi che *DirectSound* offre sono stati usati nel presente progetto la funzione `buffer.Play()` in questo caso impostata in modo che la riproduzione fosse continua, ottenuta tramite il flag `BufferPlayFlags.Looping`:

```
sound.Play ( 0, BufferPlayFlags.Looping );
```

La funzione `buffer.Stop()` per fermare la riproduzione del buffer:

```
sound.Stop();
```

La funzione `SetCurrentPosition()` utile per settare una posizione all'interno del buffer, nel nostro caso riportare il puntatore all'inizio del buffer per riprendere poi da questo punto la riproduzione:

```
sound.SetCurrentPosition(0);
```

5.5 Implementazione dell'applicazione Analogic Wii Loop Machine

Dopo aver brevemente presentato la parte preliminare per quanto riguarda l'inizalizzazione sia dello strumento *Wiimote* che di *DirectSound* entriamo ora nel merito dell'implementazione dell'applicazione vera e propria.

Innanzi tutto il progetto è stato realizzato in modo che sia possibile un'interazione tra utente ed interfaccia sia attraverso il controller *Wiimote* sia attraverso i pulsanti presenti sul form azionabili direttamente con il mouse.

L'interazione con l'applicazione è strutturata principalmente in due fasi: una prima di mixaggio e una seconda di riproduzione.

Prima di procedere con la riproduzione è stato pensato di dare appunto la possibilità all'utente di mixare offline le tracce che si vogliono poi mandare in loop e di scrivere il risultato su un file di output da noi arbitrariamente chiamato *audio.wav.*

Questo mix è effettuato grazie all'utilizzo di una classe esterna al progetto sviluppato, chiamata WAVFile, che supporta audio in formato .wav a 8 o 16 bit, mono o stereo.

L'unica restrizione è che i file che si vogliono mixare abbiano la stessa frequenza di campionamento.

Una volta effettuato il mix, che consiste in due brevi passi quali la scelta delle tracce da mixare, permessa da una finestra di dialogo che compare alla pressione del pulsante Loops, e la pressione del pulsante Mix, che comporta invece la scelta del file di output su cui scrivere il risultato del mix e l'effettiva realizzazione di questo, si può passare alla fase di riproduzione e manipolazione implementata nella sua totalità dal candidato.

In questo contesto prendiamo in esame a scopo esemplificativo la sola porzione di codice relativa alla gestione della riproduzione attraverso il *Wiimote*, in quanto scelta di carattere innovativo e di poco discostante dal codice presente nella descrizione degli eventi legati ai bottoni del form.

Le funzioni assegnate ai pulsanti del *Wiimote* vengono descritte tutte all'interno di un metodo chiamato `wiimote_WiimoteChanged ()` nel quale per cominciare si estrae lo stato del wiimote attraverso la stringa:

```
WiimoteStatews = e.WiimoteState;
```

e che d'ora in poi viene richiamato con il nome `ws`.

La precedente operazione è fondamentale dal momento che la pressione di un tasto o una qualsiasi fonte di input del controller provoca un cambiamento dello stato dello stesso; per questo motivo si è rivelato necessario usare questa funzione della libreria.

Ora con una serie di cicli if si richiamano gli eventi relativi ad esempio alla pressione dei vari bottoni sul controller e si assegnano a questi delle funzioni come lo start

della riproduzione di una traccia o la sua interruzione.

```
if (ws.ButtonState.B)
        {
            sound.Stop();
            isPlay = false;
            sound.SetCurrentPosition(0);
            sound.SetEffects(null);
            isEcho = false;
            isDist = false;
            isChorus = false;
            isFlanger = false;
        }
```

Il codice sopra mostrato riguarda ad esempio la pressione del pulsante B sul *Wiimote* con cui vengono richiamate una serie di funzioni tutte ereditate da *DirectSound,* oltre alla modifica del valore di opportune variabili.

Prima tra tutte la funzione

`buffer.Stop ()`,

che interrompe la riproduzione del buffer chiamato `sound` precedentemente creato, poi la funzione

`buffer.SetCurrentPosition (0)`,

che reimposta il cursone che tiene traccia della posizione di riproduzione alla posizione specificata negli argomenti cioè in questo caso 0 (dall'inizio) e in ultimo la funzione

`buffer.SetEffects(null)`,

che nel nostro caso imposta gli effetti del buffer a null cioè li disattiva.

La parte inerente a *DirectSound* ha richiesto un'ampia fase di testing dal momento che inizialmente, nonostante una corretta implementazione e stesura del codice relativo, non si è riusciti a rendere funzionante anche solo la riproduzione di un file

audio.

Non venivano mostrati errori in fase di compilazione ma, una volta mandato in esecuzione il programma in modalità debug, dopo pochi secondi veniva visualizzato l'errore in figura (fig 12).

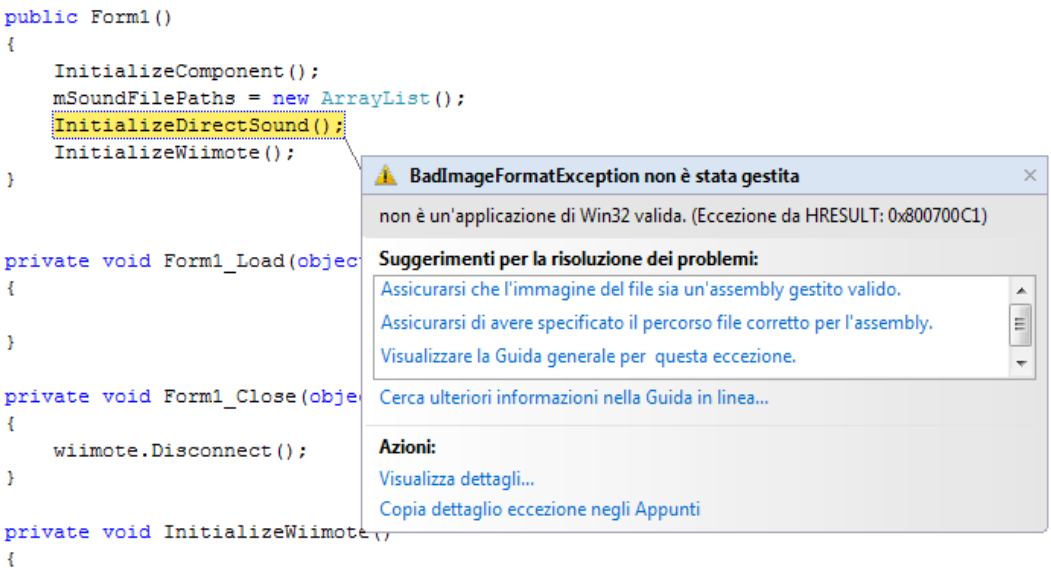

Fig 12 – Messaggio di errore mostrato in fase di Dubug

Dopo una serie di tentativi per risolvere il problema, insolito anche agli occhi di sviluppatori più esperti a cui si è chiesto aiuto, si è riusciti ad individuare la causa dell'errore.

La macchina su cui è stata sviluppata l'applicazione Analogic Wii Loop Machine è dotata di sistema operativo Microsoft Windows 7 (x64), versione a 64 bit dell'ormai noto sistema operativo.

Probabilmente, secondo considerazioni successive alla risoluzione del problema, si presenta un'incompatibilità se il progetto viene eseguito in modalità debug con specificato Any CPU nell'opzione piattaforma di destinazione, tra le opzioni del progetto.

Non mostra problemi invece se viene scelta come opzione dello stesso campo x86, forzando quindi l'applicazione ad essere eseguita in modalita 32 bit come mostrato nella figura sottostante (fig 13).

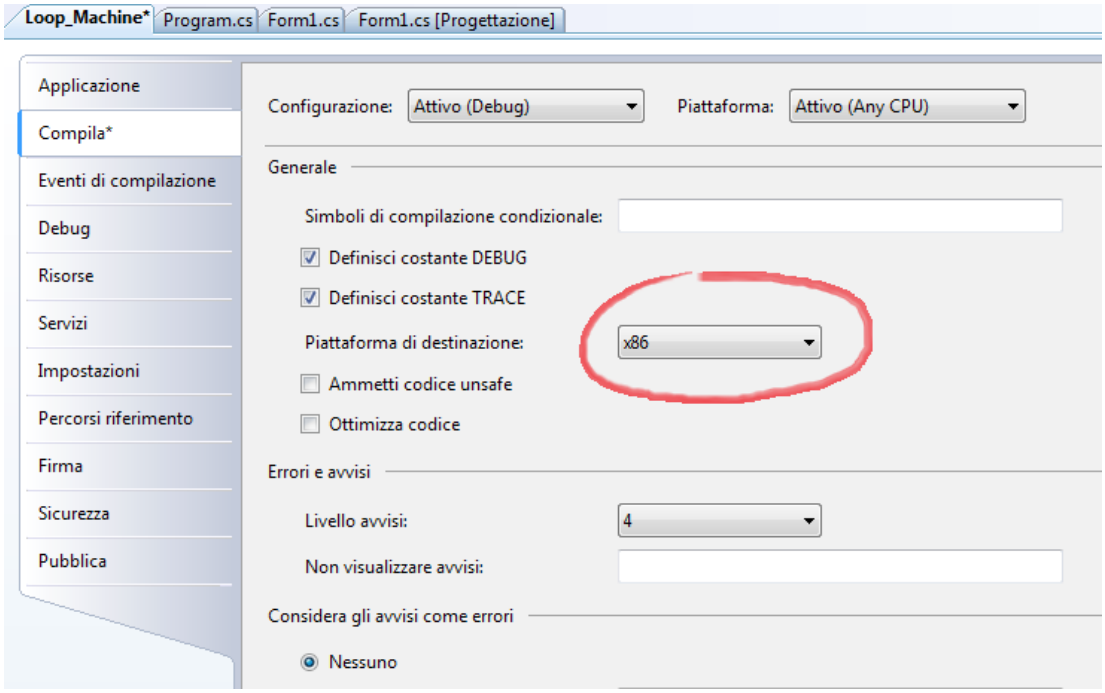

Fig 13 - Maschera delle proprietà del progetto

Una volta risolto questo tipo di problema *DirectSound* si è dimostrato un efficiente mezzo per realizzare quanto preposto e si è riusciti con facilità a riprodurre un file audio.

Si è passati quindi all'implementazione della parte relativa agli effetti da aggiungere al buffer in riproduzione.

Anche in questo caso l'attivazione viene resa possibile in un duplice modo, cioè sia attraverso l'interazione con i pulsati del form usando il mouse come mezzo di selezione, sia usando il *Wiimote* per l'attivazione o disattivazione di questi ultimi.

Mostriamo anche in questo caso la sola parte di codice inerente all'implementazione

con lo strumento *Wiimote*.

In prima istanza descriviamo il metodo `AddEffects()` creato ad hoc per la definizione degli effetti che in fase progettuale si è scelto di implementare:

```
private void AddEffects()
   {
      int numEffects=0, numEffect=-1;

      sound.Stop();

      numEffects += isEcho ? 1 : 0;
      numEffects += isChorus ? 1 : 0;
      numEffects += isFlanger ? 1 : 0;
      numEffects += isDist ? 1 : 0;

 EffectDescription[] fx = new EffectDescription[numEffects];

     if (isEcho)
        {
         numEffect++;
         fx[numEffect] = new EffectDescription();
         fx[numEffect].GuidEffectClass=DSoundHelper.StandardEchoGuid;
        }

     if (isChorus)
        {
         numEffect++;
         fx[numEffect] = new EffectDescription();
         fx[numEffect].GuidEffectClass=DSoundHelper.StandardChorusGuid;
        }


     if (isFlanger)
        {
         numEffect++;
         fx[numEffect] = new EffectDescription();
         fx[numEffect].GuidEffectClass=DSoundHelper.StandardFlangerGuid;
        }

     if (isDist)
        {
         numEffect++;
         fx[numEffect] = new EffectDescription();
         fx[numEffect].GuidEffectClass=DSoundHelper.StandardDistortionGuid;
         }


if (numEffects > 0)
        {
         sound.SetEffects(fx);
        }
```

```
        else
        {
         sound.SetEffects(null);
        }
         sound.Play(0, BufferPlayFlags.Looping);

    }
```

Come possiamo notare dalla porzione di codice soprastante definiamo prima due variabili, una (`numEffects`) che tenga traccia del numero totale di effetti attivati, e una seconda chiamata invece `numEffect` che indica invece l'indice dell'array relativo al singolo effetto.

Una volta definite le variabili utitli all'attivazione degli effetti desiderati, fermiamo la riproduzione del buffer in modo che sia possibile aggiungere gli effetti, per poi riprendere la riproduzione a partire dal punto in cui si era stoppata.

DirectSound non permette infatti l'attivazione di effetti in realtime su un buffer in riproduzione, per questo motivo si è rivelato necessario fermare la riproduzione.

Lato ascoltatore sarà udibile un click istantaneo ma si è pensato fosse il compromesso migliore per preservare l'interazione in tempo reale tra utente e riproduzione audio necessaria per realizzare una Loop Machine.

Prima di procedere con la descrizione del metodo `AddEffects` è necessario però porre l'attenzione sulla porzione di codice che assegna alla pressione di ogni tasto del pad direzionale presente sul *Wiimote* uno specifico effetto:

```
private bool uppress = false;
private bool leftpress = false;
private bool rightpress = false;
private bool downpress = false;


private void wiimote_WiimoteChanged(objectsender,WiimoteChangedEventArgs e)
        {
        [...]

        if (!uppress && ws.ButtonState.Up)
```

```
                {
                   uppress = true;
                   isEcho = (!isEcho);
                   this.AddEffects();
                }

        else if (uppress && !ws.ButtonState.Up)
            {
              uppress = false;
            }
[...]

}
```

Le prime variabili definite sono state introdotte in un secondo momento, quando in fase di testing è stato notato un problema con l'attivazione degli effetti.

Premendo un tasto sul pad infatti non si attivava o disattivava semplicemente l'effetto desiderato, ma si verificavano delle attivazioni e disattivazioni continue in numero casuale, nonostante si prestasse attenzione ad effettuare una brevissima pressione sul pulsante.

Si è quindi deciso di inserire una variabile sempre di tipo booleano per ogni tasto (`uppress, downpress, rightpress, leftpress`) in modo che ad ogni pressione si cambi il valore della variabile relativa da false a true o viceversa risolvendo così il problema innescato dalla pressione.

Nella parte relativa ai cambiamenti di stato del *Wiimote* abbiamo poi definito l'azione da effettuare alla pressione del pad direzionale (nell'esempio di codice riportato verso l'alto - up -).

In questo caso, come negli altri relativi ai pulsanti del pad, il ciclo if comprende due diversi parametri:

- La variabile booleana appena descritta, che cambia il suo valore ad ogni pressione del tasto relativo, inizialmente impostata a false;
- L'evento relativo alla pressione del pulsante sul pad. Perciò ad ogni pressione

del tasto, se la variabile `uppress` è impostata a false e il pulsante viene premuto, il ciclo if si occupa di:

1) cambiare il valore di `uppress`;
2) cambiare il valore della variabile `isEcho` (che indica lo stato dell'effetto cui è riferita, in questo caso l'echo);
3) chiamare il metodo `AddEffects` che imposti realmente l'effetto richiamandolo da *DirectSound* e aggiungendolo al presente form (`this`).

Detto questo ritorniamo al metodo `AddEffects` che stavamo descrivendo.

A questo punto verifichiamo grazie alle nostre variabili quali effetti siano in realtà attivi e creiamo un nuovo array con tante righe quanti sono questi effetti attivi:

```
EffectDescription[] fx = new EffectDescription[numEffects];
```

Da qui in poi comincia una serie di cicli if, che saranno tanti quanti gli effetti da gestire (nel nostro caso quattro), nei quali si prende in considerazione il caso in cui la variabile relativa all'attivazione dell'effetto sia posta a true e quindi quando l'effetto è in realtà stato attivato tramite la pressione dello specifico tasto.

Ogni ciclo incrementa di uno l'indice `numEffect` con l'istruzione `numEffect++` e poi riempie una riga dell'array creato con l'elemento effetto definito con il comando

```
fx[numEffect].GuidEffectClass=DsoundHelper.StandardEchoGuid;
```

prendendolo tra quelli offerti da *DirectSound.*

L'ultimo ciclo invece opera una distinzione tra due casi: il primo in cui il numero di effetti attivati sia maggiore di zero, procedendo di conseguenza con l'istruzione che permette quindi l'azione di aggiunta reale degli effetti al buffer, il secondo in cui invece non ci sono effetti attivi e quindi esegue l'istruzione

```
sound.SetEffects(null);
```

rimuovendo di fatto gli effetti dal buffer.

Un'altra significativa porzione di codice implementativo, che inzialmente dava alcuni problemi sempre in fase di debug, riguarda la gestione dell'accelerometro a tre assi incluso nel *Wiimote*.

In realtà quello che sembrava in primo luogo un problema legato alla stesura del codice, in realtà riguardava l'interazione tra *Wiimote* e form.

Questo problema impediva di interagire col form usando lo strumento *Wiimote* come input, quindi rendeva impossibile la gestione di un qualsiasi elemento del form, nel nostro caso specifico le trackbar relative al volume di riproduzione e alla frequenza.

Ecco riportato in figura (fig 14) l'errore segnalato:

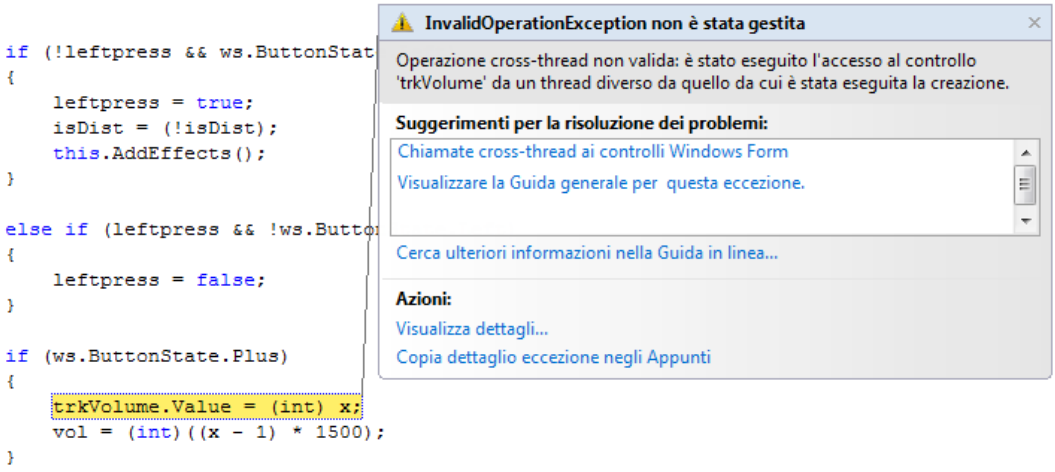

Fig 14 - Messaggio di errore mostrato in fase di Debug

L'approccio scelto per la risoluzione del problema doveva pertanto aggirare questo errore derivato dall'interazione diretta tra *Wiimote* e form.

In sostanza ci si proponeva di controllare il volume del buffer in riproduzione tramite la pressione di un tasto sul *Wiimote* e alla contemporanea variazione di un certo valore di posizionamento restituito dall'accelerometro (e con una simile azione fare lo stesso per la frequenza), restituendone graficamente la variazione grazie allo spostamento della trackbar relativa.

È stato così inserito all'interno del form un elemento timer e cioè un oggetto che permette di svolgere una data azione, descritta all'interno del relativo evento `timer_Tick(object sender, EventArgs e)`, allo scadere di ogni intervallo definito nelle proprietà del timer stesso.

L' intervallo è stato fissato a 10 msec e nell'evento abbiamo aggiunto le istruzioni che permettessero di associare il valore della trackbar a quello di una variabile.

La variabile in questione, chiamata `vol` nel caso del volume e `pit` nel caso del pitch (frequenza), è stata definita nel metodo `wiimote_WiimoteChanged()` e legata al valore dell'asse dell'accelerometro interessato dal movimento implementato (X per

quanto riguarda il volume e Y per il pitch), scalato opportunamente.

Questo ha permesso quindi che il cambiamento di valore da assegnare al volume o alla frequenza venisse immagazzinato in una variabile e che la trackbar relativa, ad intervalli definiti dal tick del timer, fosse spostata nella posizione relativa a quel valore, rendendo possibile il feedback grafico dell'azione.

Quella descritta costituisce una panoramica generale del progetto realizzato che pone luce sugli aspetti peculiari del lavoro ma che non vuole essere una spiegazione esaustiva del funzionamento dell'applicazione.

È stato perciò possibile, sebbene si siano riscontrati alcuni errori in fase di implementazione e stesura, realizzare l'applicazione rispettando quanto prefissatosi in principio, e aggiungendo durante la fase di scrittura alcune funzioni che non si erano considerate in fase progettuale.

Nonostante sia ancora possibile una miglioria significativa dell'applicazione sia sotto l'aspetto funzionale che grafico, ci è sembrato che il risultato finale rispettasse i canoni definiti nel concetto di Loop Machine e cioè sostanzialmente la possibilità di mixaggio delle tracce, benchè fosse comunque semplificata riaspetto alle tecniche di editing avanzato e la possibilità di interazione con il risultato mixato per mezzo di un'intuitiva interfaccia grafica che permettesse la manipolazione sonora e l'aggiunta di effetti caratteristici della musica strutturata sulla tecnica del loop.

6. Conclusioni

La periferica *Wiimote* rappresenta, allo stato attuale, il primo reale successo tecnologico ed economico nell'ambito dell'interazione con il sistema, non solo videoludico, ma come visto anche musicale, portandola ad un livello più vicino a quello dell'interazione umana con il mondo circostante.
La varietà dei sensori su di esso installati, ha reso il controller oggetto di studio nei più svariati ambiti, favorendo la crescita di iniziative di utilizzo volte a far fronte alle più svariate applicazioni.
Ci si augura che in futuro sia possibile assistere a numerose creazioni che portino il mondo della tecnologia sempre più vicino a quello umano da un punto di vista interattivo e che queste siano rese accessibili ad una vasta gamma di consumatori grazie al loro modico prezzo, proprio come lo è oggi il *Wiimote*.
Solamente grazie all'insorgere di innovazioni simili sarà possibile ottenere quello che dovrebbe essere il vero compito della tecnologia, cioè ampliare e perfezionare le potenzialità umane tramite l'utilizzo di specifici strumenti per semplificare e agevolare sempre di più la vita dell'uomo.
Si incentiva pertanto chiunque volesse prender spunto da questo lavoro per un futuro sviluppo o ampiamento nel settore, dal momento che le risorse dello strumento in questione sono ancora molteplici e svariate.
Se già oggi il *Wiimote* è stato utilizzato per aprire una porta e persino per muovere un braccio meccanico da quindici tonnellate (http://www.youtube.com/watch?v=v1AJ_OBJUpY&feature=player_embedded#), pensiamo che il futuro ci riservi numerose sorprese.

Bibliografia

[1] Bradley L - Sams Teach Yourself C#™ in 21 Days - Sams Publishing (2004)

[2] Chion Michel - L'arte dei Suoni Fissati o la Musica Concretamente. - Edizioni Interculturali - Roma (2004);

[3] Fernandez Dan, Peek Brian - Coding4fun: 10 .Net Programming Projects for Wiimote, Youtube, World of Warcraft, and More - O'Reilly (2009);

[4] Fabbri Franco - Il Suono in cui Viviamo. Saggi sulla Popular Music. - Il Saggiatore Tascabili - Milano (2008);

[5] Fabbri Franco - Around The Clock. Una breve storia della Popular Music. - UTET Libreria – Torino (2008);

[6] Foxall James, Haro-Chun Wendy - Sams Teach Yourself C#™ in 24 hours. - Sams Publishing (2002);

[7] Holmes Thom - Electronic and Experimental Music: Technology, Music, and Culture (Third Edition). - Routledge - New York (2008);

[8] Manning Peter - Electronic and Computer Music. - Oxford University Press - Oxford (1994);

[9] Camilleri Lelio - Loop, trasformazioni e spazio sonoro - http://www-3.unipv.it/britishrock1966-1976/pdf/camilleri.pdf

[10] Chung Lee Johnny - Hacking the Nintendo Wii Remote - Published by the IEEE CS. - 2008 IEEE. PERVASIVE computing 39 - http://www.cs.cmu.edu/~15-821/CDROM/PAPERS/lee08.pdf

[11] WiimoteLib - .NET Managed Library for the Nintendo Wii Remote by Brian Peek - MSDN Alliance 2008

Ringraziamenti:

Il mio più sentito ringraziamento va alla mia famiglia per essermi stata vicina in ogni momento, ai miei genitori per aver sempre creduto in me e nelle mie capacità, incentivandomi a seguire i miei sogni e le mie passioni, prima tra tutte la musica, per avermi dato questa possibilità e molte altre per le quali gli sarò sempre grato, a mio fratello Andrea per avermi sopportato quando neanche io ci sarei riuscito, per aver tollerato il mio pessimo umore e perchè nonostante le incomprensioni mi è sempre stato vicino.

Ringrazio, inoltre, tutti ma proprio tutti i “collions”, per tutti gli anni passati insieme, per esser stati degli ottimi amici in ogni situazione e perchè quello che sono lo devo anche a loro.

E infine penso che la voglia di impegnarmi, di studiare, di portare a termine tutto questo non l'avrei avuta se non grazie a Gloria che ha sempre creduto in me e mi ha incoraggiato con tanto amore e pazienza.