UNIVERSITÀ DI PADOVA **FACOLTÀ DI INGEGNERIA**

TESI DI LAUREA

STRUMENTI PER LA VALUTAZIONE DI AMBIENTI MULTIMODALI VIRTUALI

Laureando: Alberto Ziliotto

Relatore: Prof. Federico Avanzini
Correlatore: Ing. Michele Geronazzo

Corso di Laurea Triennale in Ingegneria dell'Automazione

28 marzo 2013

Anno Accademico 2012/2013

Prefazione

In questo lavoro di tesi vengono presentati gli strumenti software appositamente sviluppati per costruire un ambiente multimodale virtuale che fornisce feedback audio, aptico e visivo. L'ambiente multimodale virtuale sviluppato é composto da un modello di *gradino* utilizzato per esperimenti psicofisici che fanno utilizzo della periferica Phantom che permette di esplorare apticamente l'ambiente. Per fare questo si é sviluppato un ambiente di controllo dell'esperimento in Matlab con relativa gestione dei soggetti su database. Si é progettata una patch che modula il suono, realizzata tramite l'ambiente per la sintesi audio Pure Data. Il feedback visivo e Il feedback aptico, ovvero la forza percepita per mezzo del dispositivo Phantom , sono stati implementati rispettivamente dal linguaggio Extensible Language Markup X3D e dalla libreria H3DAPI.
Questi diversi strumenti software, interagiscono tra loro per mezzo del protocollo OSC, per formare l'ambiente multimodale virtuale con lo scopo di studiare in che misura il feedback audio incide sulla percezione dell'altezza spaziale di un ostacolo, nel caso particolare di un gradino virtuale.

Sommario

L'obiettivo di questa tesi é quello di sviluppare degli strumenti che permettano l'esplorazione multimodale di un ambiente virtuale. In particolare nel capitolo 1 viene introdotto il concetto di ambiente multimodale e di mappa cognitiva. É presente lo stato dell'arte sul ruolo del feedback audio nell' aiutare le persone con gravi deficit alla vista nella costruzione di mappe multimodali. Si conclude il capitolo con una breve introduzione ai due progetti *Sample01* e *Multimodal01*.
Nel secondo capitolo, viene spiegata la struttura database di salvataggio dati che sta alla base della gestione dei soggetti. Successivamente vengono esposte le funzioni dell'interfaccia grafica dei due progetti.
Nel capitolo 3, si analizzano le problematiche relative al feedback audio realizzato tramite l'ambiente per la sintesi audio Pure Data. Un canale di comunicazione, comune a tutti i programmi utilizzati, riceve tramite protocollo OSC le informazioni per il feedback audio.
Infine, nel quarto capitolo sono stati elencati gli strumenti necessari per la costruzione dell'ambiente multimodale finale. Attraverso l'interazione tra X3D, Matlab, Pure Data, H3DAPI, Python viene spiegato come é stato possibile costruire un semplice ambiente multimodale virtuale.

Indice

Capitolo 1

Introduzione

L'obiettivo di questa tesi è di realizzare degli strumenti software per costruire un ambiente multimodale con feedback audio, aptico e visivo. I linguaggi di programmazione utilizzati per fare questo sono Matlab, Pure Data, X3D, H3D e Python. L'ambiente virtuale progettato è un modello di un *gradino*. Saranno svolti degli esperimenti tramite la periferica aptica Phantom che permette di esplorare l'ambiente multimodale e di percepire il feedback aptico. Verrà studiato in che misura il feedback audio fa percepire l'altezza del gradino al soggetto in esame.

1.1 Background

1.1.1 Esplorazione di spazi Multimodali

Si attribuisce il termine *"multimodale"* un qualsiasi tipo di interazione che coinvolge più di un canale percettivo. Un ambiente multimodale virtuale è composto da un insieme di strumenti che permettono all'utente di interagire con il computer tramite più canali percettivi come il tatto, la vista e l'udito. Il lavoro di ricerca condotto da *Ying Ying Huang* [1] rappresentato da esperimenti per mezzo di un interfaccia aptica su di un labirinto multimodale ha confermato che il feedback audio aiuta le persone con seri problemi alla vista. È stato evidenziato che attraverso l'uso dell'interfaccia aptica gli utenti riescono a percepire e a distinguere informazioni sull'ambiente. Le persone non vedenti hanno problemi nell'orientarsi in spazi a loro sconosciuti senza una guida, in modo indipendente. Le strategie per costruire mappe cognitive sono diverse tra persone sane e affette da problemi cal canale visivo. Le persone vedenti usano principalmente la vista, mentre le persone non vedenti utilizzano il tatto e l'udito. Le tecnologie aptiche e audio sono state sviluppate per facilitare la costruzione di mappe cognitive. Ci sono due tipi di aiuto: **Attivo** e **Passivo**.
L'aiuto *passivo* consiste in una descrizione verbale, tattile; avviene prima di entrare fisicamente nello spazio da esplorare. L'aiuto *attivo* avviene nello spazio inesplorato, con il soggetto al suo interno, tramite GPS, supporto verbale, guida personale. L'aiuto attivo ha il limite di non poter essere fatto prima che il soggetto si trovi fisicamente nello spazio inesplorato. Attraverso

ambienti multimodali virtuali e ancor piú utilizzando strumenti aptici si può ricreare l'ambiente dando la possibilit di esplorarlo preventivamente.

1.1.2 Mappe Cognitive spaziali

Le mappe mentali e i modi possibili per navigarle, sono essenziali per lo sviluppo delle capacità cognitive. In una ricerca pubblicata da Lahav [2] si evince che la maggior parte delle informazioni per la costruzione di queste mappe mentali, provengono dalla vista. Le persone cieche o affette da gravi patologie agli occhi, non possono costruire mappe cognitive. Dovranno quindi ricevere altre informazioni sensoriali per compensare questo deficit. Passini [3] nelle ricerche sulla navigazione visiva di spazi conosciuti e sconosciuti, verificó che l'acquisizione di mappe cognitive da parte di persone non vedenti puó essere supportata a due livelli:

- **Concettuale**
- **Percettivo**

A livello percettivo, Lahv sostiene che il senso che compensa di più la mancanza della vista è il tatto. A livello concettuale, si concentra l'attenzione sulle possibili strategie per realizzare mappe cognitive efficienti per l'orientazione spaziale e la memorizzazione di percorsi all'interno di esse. Le ricerche dimostrano che le persone utilizzano due strategie principali:

1. *Route:* Riconoscimento di caratteristiche spaziali;
2. *Map:* Percezione dello spazio come entitá unica;

Fletcher [4] dimostrò che le persone non vedenti usano principalmente la strategia 'Route' per riconoscere e muoversi in nuovi spazi. I ricercatori Mioduser e Lahav, a seguito di ricerche concluse nel 2004 [5] dimostrarono che le mappe cognitive per spazi sconosciuti e le possibili vie per esplorarle sono essenziali per lo sviluppo di capacit mobili e orientative efficienti.
Dopo vari esperimenti in ambienti multimodali virtuali sulle ricerche precedentemente descritte, si concluso che:

1. L'esplorazione dello spazio virtuale contribuisce alla costruzione di mappe cognitive di spazi inesplorati;
2. La realizzazione di mappe cognitive, da parte di un soggetto non vedente, come risultato di apprendimento con l'ambiente MLVE (Multisensory Virtual Learning Environment) contribuisce all'orientamento del soggetto anche nello spazio reale.

1.2 Modo Operativo

Il soggetto sará accompagnato all'interno della cabina silente e poi bendato, l'esperimento é strutturato in questo modo:
Verranno sottoposti al soggetto in esame diversi modelli di gradino *(vedi Tabella 1.1)*. I tipi di gradino differiscono in altezza e in range di frequenza. Il soggetto subirá il feedback aptico attuato dal Phantom e il feedback audio prodotto da due monitor posizionati all'interno della cabina.
Il Phantom permette di navigare sul gradino impugnando lo stylus come una penna. Il soggetto riceverá un feedback di forza quando lo stylus colliderá con la superficie del gradino virtuale. Il feedback audio viene riprodotto solo nel caso in cui il soggetto stia toccando le superfici del gradino. La frequenza del suono percepito é direttamente proporzionale alla posizione verticale dell'esploratore sul gradino. Sono presenti due range di frequenze per il feedback audio rispetto alla frequenza base di 200 Hz: un'ottava oppure due. Si ha una massima escursione in frequenza lungo l'asse verticale, dalla base del gradino alla sommitá. Alla fine di ogni tipologia di gradino testato, il soggetto dovrà indicare verbalmente l'altezza percepita al tecnico che gestisce l'esperimento all'esterno della cabina.
Quest'ultimo puó assistere all'esperimento attraverso l'interfaccia grafica che mostra a video il movimento del Phantom sul gradino virtuale. Dopo un numero sufficiente di soggetti esaminati si analizzeranno i dati ottenuti per verificare se il feedback audio influisce la percezione dell'altezza in un ambiente multimodale.

Altezza	Range di frequenza
2	1 ottava
	2 ottave
4	1 ottava
	2 ottave
6	1 ottava
	2 ottave
8	1 ottava
	2 ottave

Tabella 1.1: *Varietá dei gradini da sottoporre nell'esperimento*

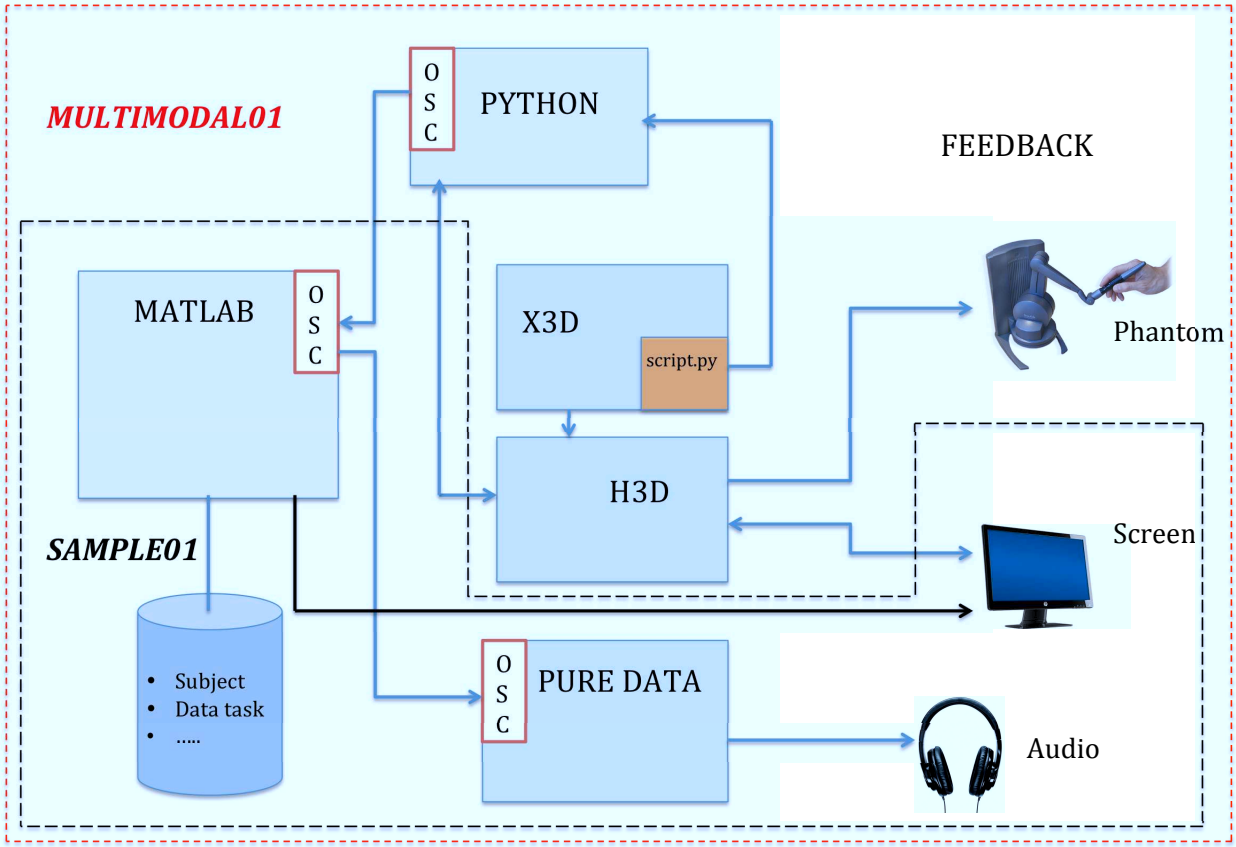

Figura 1.1: *Schema logico dei progetti Sample01 e Multimodal01.*

In Figura 1.1 si possono notare i collegamenti logici tra i vari strumenti sviluppati. É importante sottolineare che Sample01 é contenuto in Multimodal01. Questo significa che per realizzare Multimodal01 si é prima progettato Sample01 con le seguenti caratteristiche:

1. Gestione dei soggetti con modello di database su file *.mat.
2. Feedback visivo in 2 dimensioni realizzato in Matlab.
3. Feedback audio attuato da Pure Data[1].
4. Comunicazione tra Matlab, Pure Data attraverso protocollo OSC.

Matlab e Pure Data necessitano di comunicare tra loro per riprodurre il feedback audio. Matlab gestisce la finestra in cui si svolgerá la simulazione del Theremin (spiegata in dettaglio nel prossimo paragrafo). Con il mouse si piloterá un cursore verde (figura 1.2) le cui coordinate vengono scalate e inviate via OSC a Pure Data che si occupa di creare il feedback audio che dipende

[1] http://puredata.info/ vedi capitolo 3

dal task scelto.
Multimodal01 estende le precedenti caratteristiche per realizzare l'ambiente di sperimentazione finale in questo modo:

1. Gestione dei soggetti con modello di database su file *.mat.
2. Feedback visivo in 3 dimensioni progettato in X3D[2].
3. Feedback audio attuato da Pure Data.
4. Feedback aptico, sul gradino virtuale, realizzato per mezzo della libreria H3DAPI[3].
5. Comunicazione tra Matlab e Pure Data e H3D attraverso protocollo OSC.

Per far comunicare X3D con Matlab ci si é serviti di di uno script Python contenuto nel file X3D richiamato ciclicamente. L'utilizzo verrá spiegato in dettaglio nel capitolo 4.

[2]http://www.web3d.org/x3d/ vedi paragrafo 4.2.2
[3]http://www.h3dapi.org/ vedi paragrafo 4.2.3

1.2.1 Sample 01

Per realizzare l'ambiente multimodale si è partiti progettando un programma di emulazione dello strumento elettronico *theremin*. Il theremin é uno strumento musicale elettronico.Venne inventato nel 1919 dal fisico sovietico Léon Theremin. Si basa su oscillatori che, lavorando in isofrequenza al di fuori dello spettro udibile, producono, per alterazioni delle loro caratteristiche a seguito della presenza delle mani del musicista nel campo d'onda, dei suoni sul principio fisico del battimento, questa volta nel campo delle frequenze udibili. É composto fondamentalmente da due antenne poste sopra e a lato di un contenitore nel quale é alloggiata tutta l'elettronica. Il controllo avviene allontanando e avvicinando le mani alle antenne: mediante quella superiore (posizionata verticalmente) si controlla la frequenza del suono, quella laterale (posta orizzontalmente) permette di regolarne l'intensitá.

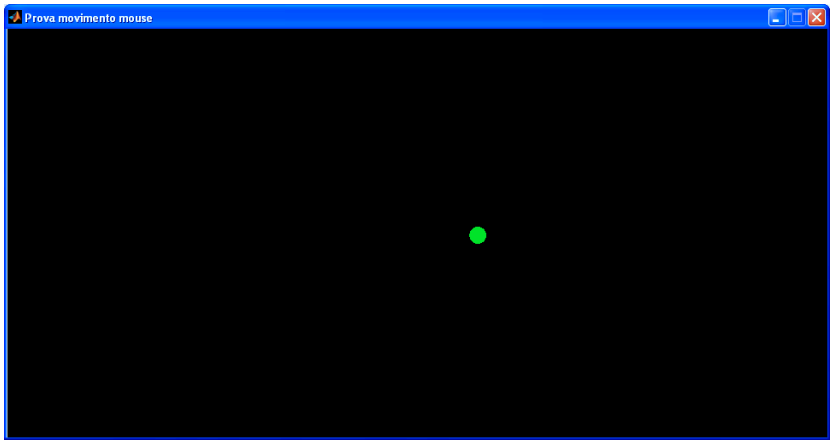

Figura 1.2: *Feedback visivo Theremin.*

L'ambiente su cui è costruito *"Sample 01"* è composto da codice Matlab e Pure Data. Ci sono quindi due programmi che comunicano tra loro. L'ambiente Multimodale viene avviato da Matlab portandosi all'interno della cartella di lavoro *"Sample 01"* , ed eseguendo il comando **start**. Il programma di simulazione del Theremin si presenta con un interfaccia grafica che permette di impostare tutte le opzioni riguardanti il soggetto e l'esperimento. Tra le varie opzioni si puó selezionare il Task: 1 o 2.
Il **task 1** associa al movimento orizzontale del mouse la variazione in frequenza e al movimento verticale la variazione dell'intensitá. Il **task 2** viceversa. Si é deciso di adottare queste due grandezze di controllo per implementare un modello virtuale del theremin. Una volta scelte tute le impostazioni, premendo start si apre la finestra in cui si pu comandare il cursore theremin tramite

il mouse (figura 1.2).
Ad ogni movimento del mouse corrisponde una variazione del volume del suono e della frequenza a seconda del task scelto. Matlab elabora le coordinate del mouse e le trasmette, tramite protocollo OSC, a Pure Data che riproduce il suono. Matlab compie delle operazioni di scalamento delle coordinate. Pure data accetta valori di ingresso per l'intensit da 0 a 128 e per la frequenza da 100 a 1000 Hz. Si é utilizzato il range 0-128 perché l'intervallo usato nello standard MIDI per rappresentare le note e i controlli per esse. Il range di frequenza é stato deciso in modo da poter rendere sensibile la variazione di frequenza.
Oltre alla simulazione del Theremin, in questa fase é stata progettata la gestione dei soggetti. Quando un soggetto compie l'esperimento possibile salvare la traiettoria percorsa dal mouse in modo automatico o tramite il pulsante *"save"* presente nell'interfaccia grafica. I dati relativi ai soggetti vengono salvati su file di tipo **.mat**. Per il salvataggio automatico é necessario aver inserito le credenziali del soggetto in esame all'interno del programma tramite la finestra di dialogo in figura 1.3.

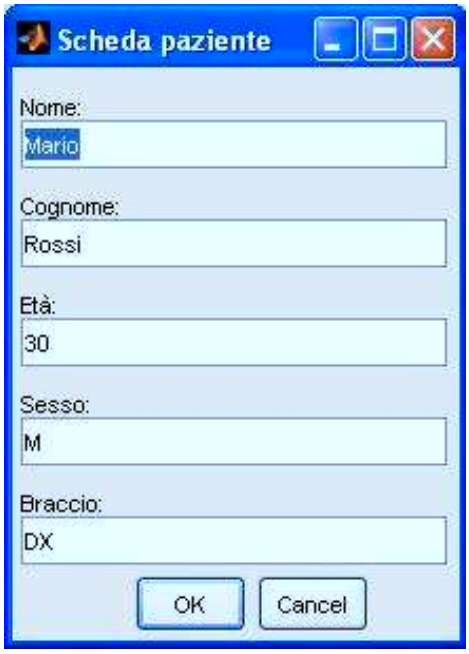

Figura 1.3: *Finestra per l'inserimento delle credenziali del soggetto*

La struttura del file system é stata progettata in modo tale da essere organizzata come un vero e proprio database.

1.2.2 Multimodal 01

Questo Ambiente è stato sviluppato ampliando le potenzialitá di *Sample 01*. Si è mantenuta la stessa gestione dei soggetti, lo stesso protocollo di comunicazione. Le principali aggiunte sono:

- Feedback aptico, sul gradino virtuale, realizzato per mezzo della libreria H3DAPI.
- Ambiente Multimodale in 3 dimensioni.

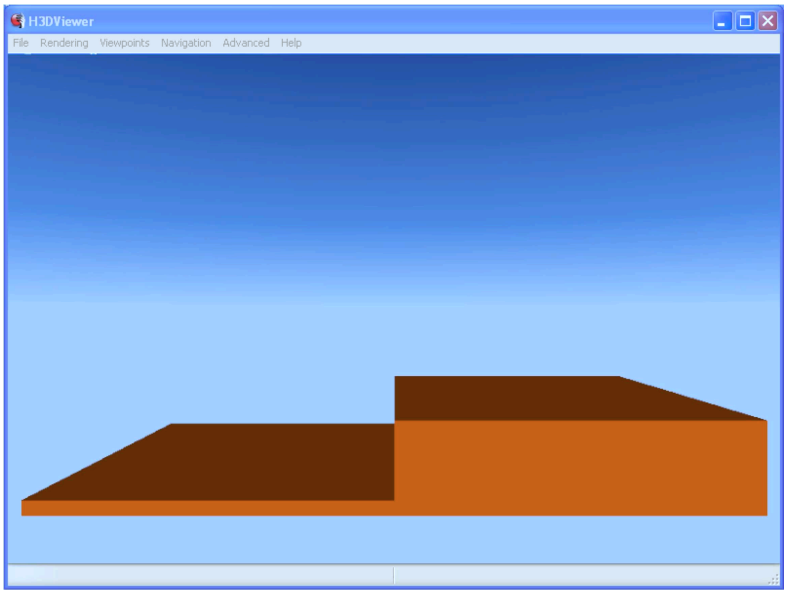

Figura 1.4: *Feedback visivo Gradino.*

Si è inoltre inserito il dispositivo aptico Phantom per rendere possibile la navigazione all'interno dello spazio multimodale con feedback aptico. Il gradino é stato realizzato utilizzando il linguaggio X3D[4], interpretato dalla piattaforma H3DViewer[5]. Pure Data, Matlab e H3DViewer interagiscono tra loro attraverso il protocollo OSC. In figura 1.1 nel riquadro tratteggiato rosso, si vede il collegamento logico tra i vari tools sviluppati.

[4]http://www.web3d.org/realtime-3d/
[5]http://www.h3dapi.org/

Capitolo 2

Gestione Soggetti

In questo capitolo, si affronteranno nel dettaglio le funzioni dell'interfaccia grafica e la struttura del database costruito su file **.mat*.

2.1 Principali strutture dati e loro operazioni

La gestione dei soggetti che hanno svolto gli esperimenti è stata progettata sul modello di un database. All'interno della cartella *subjects* contenuta in *Sample01,* vengono create delle sotto-cartelle, una per ogni soggetto. Le cartelle avranno il seguente nome: **subjN** ; dove N è l'id del soggetto, assegnato automaticamente all'inserimento nel database. Ogni cartella del tipo **subjN** conterrà un file **subj_info.mat** con tutte le caratteristiche del soggetto. Le cartelle relative ai soggetti conterranno anche dei file **X.mat** dove X corrisponde al numero identificativo del task. All'interno di questi file verranno registrate le coordinate del mouse nel caso di *Sample01* e le coordinate tridimensionali del Phantom, nel caso di *Multimodal01*. Oltre ai dati relativi l'esperimento sono anche registrati degli indici quali: **timestamp_id** e **session_id** che permettono di associare correttamente i dati alla relativa sessione di esperimento. Così facendo si ottiene la struttura di un database implementata si file **.mat** con il seguente schema entità relazione in figura.

La ridondanza dei campi *session_id* e *timestamp_id* nell'entitá data é voluta per semplicitá di analisi. Si nota che il *session_id* identifica una sessione di esperimenti che sono identificati dal *timestamp_id* il campo *dates* crea una relazione tra il *timestamp_id* e la data e l'ora in cui è stato avviato l'esperimento. Ecco un esempio delle tabelle **Dates** e **Data** presenti nei file **.mat* nel caso di *Sampe01*

dates	session_id	timestamp_id
19-03-2013 14:07.20	1	1
19-03-2013 14:10.30	1	2

Tabella 2.1 Tabella Dates per Sample01.

n	X	Y	session_id	timestamp_id
1	34	78	1	1
2	36	100	1	1
..	..	..	1	1
1	96	120	1	2
2	97	121	1	2
..	..	..	1	2

Tabella 2.2 Tabella contenente i dati in Sample01.

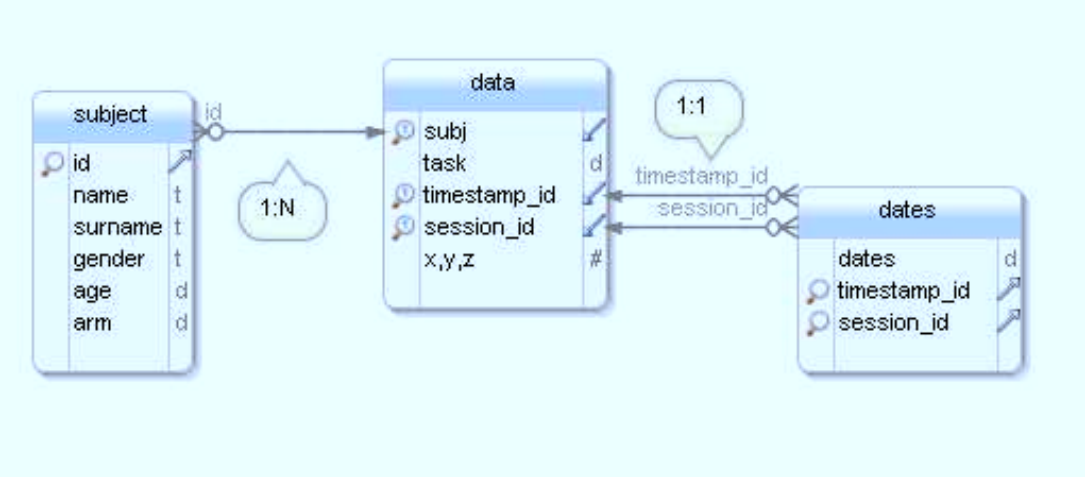

Figura 2.1: *Schema ER del database implementato su file .mat.*

2.2 Funzioni dell'interfaccia grafica

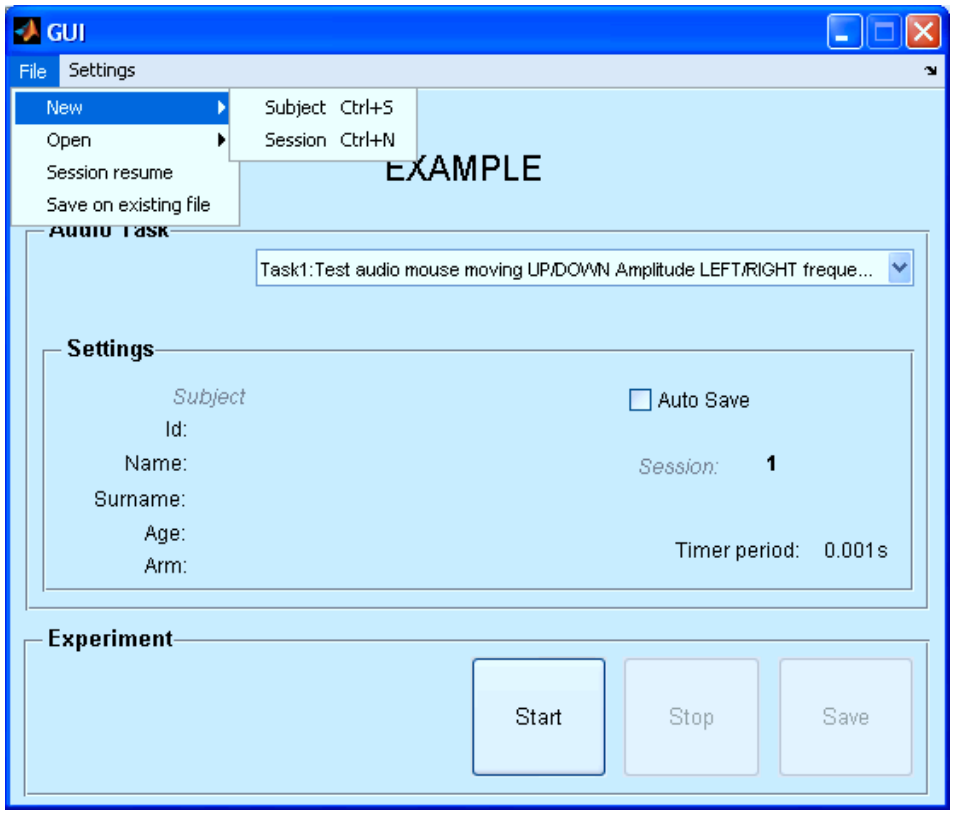

Figura 2.2: *Interfaccia Grafica di Sample01.*

L'interfaccia grafica di *Sample01* é rappresentata in figura 2.2. Si possono immediatamente notare un menu principale e tre frames:

- **Audio Task** Contiene un menu a tendina in cui si può scegliere il *task*. In questo esempio serve selezionare la relazione tra il movimento orizzontale e verticale del mouse con la frequenza e volume del feedback audio.

- **Settings** In questa sezione sono presenti i parametri del soggetto che esegue l'esperimento. Si possono vedere anche il periodo (*Timer period*) dell'oggetto *timer* di matlab su cui é basata la progettazione software e l'indice *Session* che identifica univocamente una sessione di esperimenti. È disponibile anche una casella di spunta (*Auto Save*) per attivare o disattivare il salvataggio automatico. Quest'ultimo sarà attivabile solo nel caso in cui sono state inserite le informazioni relative al soggetto corrente.

- **Experiment** Sono presenti tre bottoni relativi alla gestione dell'esperimento. *Start* e *Stop* servono ad avviare e fermare l'esperimento. Il bottone *Save* permette di salvare il singolo esperimento appena terminato su di un file *.mat* a scelta dell'utente.

Il **menu** principale ha le seguenti funzinoni:

- **File**
  - **New**
    - **Subject** Inserisce un nuovo soggetto nel frame settings e lo carica per l'esperimento, se è giá presente viene caricata l'ultima sessione d'esperimenti svolta dal soggetto trovato.
    - **Session** Incrementa l'indice Session, ovvero si inizia una nuova sessione di esperimenti per il soggetto caricato. Esempio in Figura 2.4
  - **Open**
    - **Subject** Mostra una lista di soggetti precedentemente inseriti nel database. Viene chiesto se riprendere l'ultima sessione o crearne una di nuova.
      Esempio in Figura 2.3 .
  - **Session Resume** Ripristina l'ultima sessione di esperimenti e carica le informazioni del relativo soggetto in esame.
  - **Save on existing file** Una funzione simile al pulsante Save. Questa opzione permette di salvare l'ultimo esperimento eseguito alla fine di un file giá esistente.
- **Impostazioni**
  - **Timer** Cambia l'intervallo di aggiornamento della funzione che gestisce la trasmissione dei pacchetti via protocollo OSC, e dell'aggiornamento video nel caso di *Sample01*.

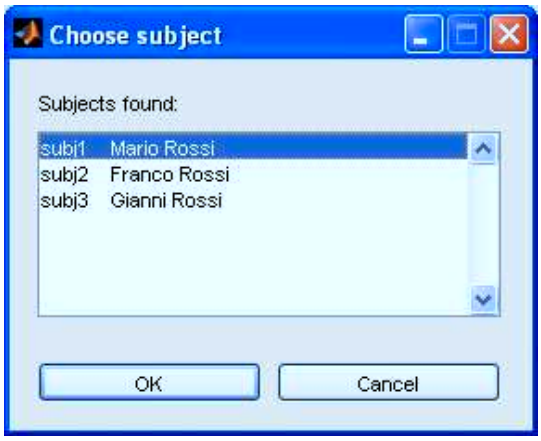

Figura 2.3: *Finestra per la scelta dei soggetti caricati.*

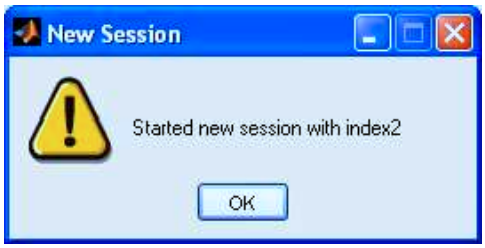

Figura 2.4: *Finestra di conferma incremento sessione*

Capitolo 3

Rendering Audio

In questo capitolo verrá illustrato come viene attuato il feedback audio. Il protocollo di comunicazione OSC e le funzioni dei file Pure Data progettati. Come asserito nel suo sito ufficiale [1], Pure Data é un ambiente di programmazione grafico real-time per l'elaborazione di audio e video. Nato come diramazione del rinomato programma commerciale Max/MSP, Pd si é imposto come valida alternativa open source sin dagli anni '90. Sia Max che Pure Data sono stati ideati dal professore statunitense Miller Puckette, ma la forza di quest'ultimo risiede nella vasta comunitá di sviluppatori sparsi in tutto il mondo che s'impegnano per ampliarne le funzionalitá e metterne a frutto le potenzialitá. Grazie a questo notevole contributo é stata introdotta una versione completa di librerie ed estensioni aggiuntive denominata Pd-extended, di cui si é fatto uso durante la realizzazione di questo lavoro. Un altro punto di forza di Pd risiede nella portabilitá delle applicazioni realizzate e del programma stesso, che infatti é multipiattaforma e disponibile per sistemi operativi Windows, IRIX, GNU/Linux, BSD e MacOS X, nonché per sistemi piu ma oramai estremamente diusi come Android e iOS per iPhone.

3.1 Pure Data Patch

L'ambiente di sviluppo Pure Data serve ha il compito di produrre il feedback audio degli ambienti multimodali *sample01* e *multimodal01*. La struttura generale della patch inglobata nei due progetti ha la seguente forma:

- **Main:** collega **osc_receiver** e **modulator** tra loro.
  - **osc_receiver:** riceve i pacchetti inviati da Matlab con protocollo OSC sulla porta 3456.
  - **modulator:** elabora i pacchetti ricevuti estraendo la frequenza e le informazioni sull'ampiezza *Sample01* o sul panning *Multimodal01*.

[1] http://puredata.info/

Lo script principale **main**, viene richiamato dallo script Matlab *start.m* e contiene due sublet: **modulator** e **osc_receiver**.

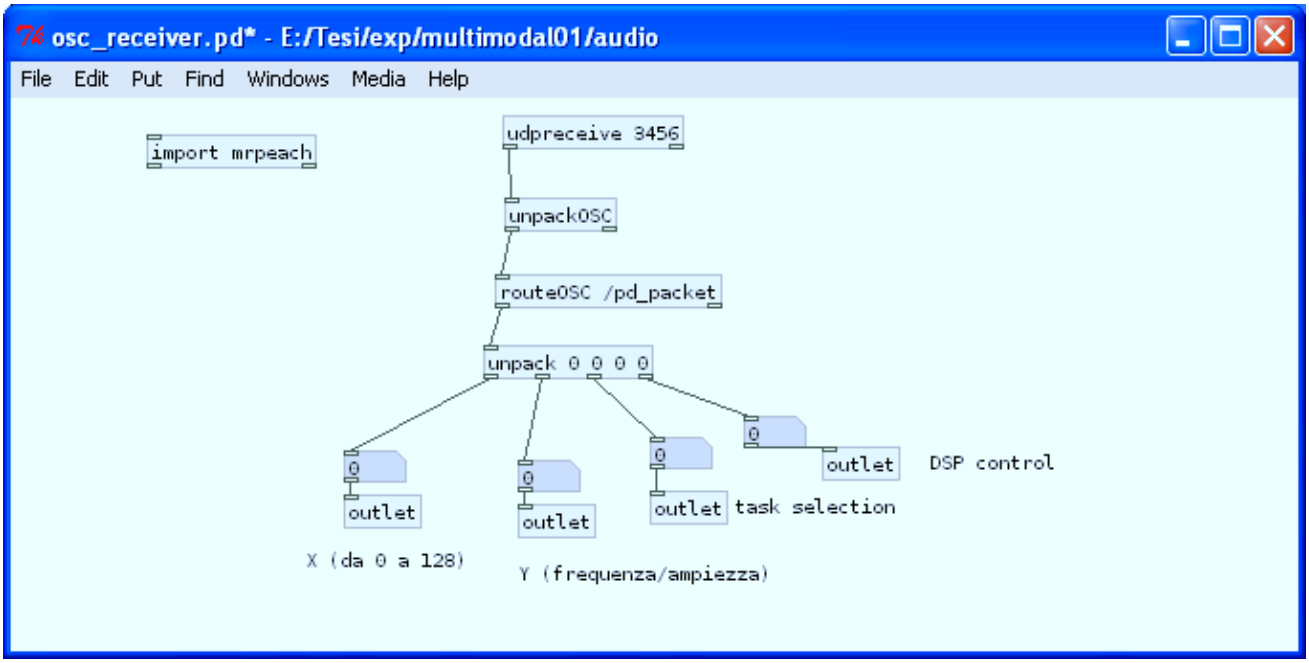

Figura 3.1: *Finestra di osc_receiver.pd*

Con poche istruzioni, vedi figura 3.1, riusciamo a ricevere e scompattare il pacchetto. Questa parte é la stessa per *sample01* e per *multimodal01*. Il pacchetto di tipo OSC inviato da Matlab cos costituito:

```
"/pd_packet X Y Z W "
```

Si sottolinea che il comando **routeOSC /pd_packet** *(figura 3.1)* serve a filtrare la stringa *"/pd_packet"* prevista dal protocollo OSC, e il successivo comando *"unpack 0 0 0 0"* permette l'accesso al contenuto delle diverse variabili X,Y, Z, W contenute nel pacchetto. Le variabili di maggiore interesse sono le prime due *(X e Y)* La sublet *modulator* cambia per i due ambienti.

3.1.1 Sample01

La variabile **X** *(figura 3.1)* é giá scalata in Matlab ed esprime, in un range da 0 a 128, l'ampiezza del suono riprodotto dalla nostra interfaccia audio. Il valore 0 corrisponde a un'amplificazione del segnale di uscita nulla, il valore 128 corrisponde a un fattore di amplificazione pari a 2. Questa variazione ' legata al movimento orizzontale o verticale del mouse a seconda di che task abbia scelto l'utente su Matlab. Si é scelto il range di 128 perché nello standard midi sono ammessi valori nell'intervallo 0-256. I primi 128 valori sono associati alle note musicali mentre i valori da 127 a 255 sono riservati a parametri di controllo. Per analogia si é riportato l'intervallo di controllo a partire dal valore 0 e deciso il range 0-128.
La variabile **Y** *(figura 3.1)* é anch'essa giá scalata da Matlab ed esprime, in un range da 100 a 1000, la frequenza in Hz che deve essere riprodotta dal modulatore. Questa grandezza legata allo spostamento del mouse all'interno della finestra di simulazione del Theremin *(figura 1.2)*. La variabile viene scalata a seconda della grandezza della finestra in modo da ottenere un valore minimo di 100 Hz e uno massimo di 1000 Hz. La variabile Z indica solamente quale task si sta eseguendo in Matlab e non influisce sul feedback audio.

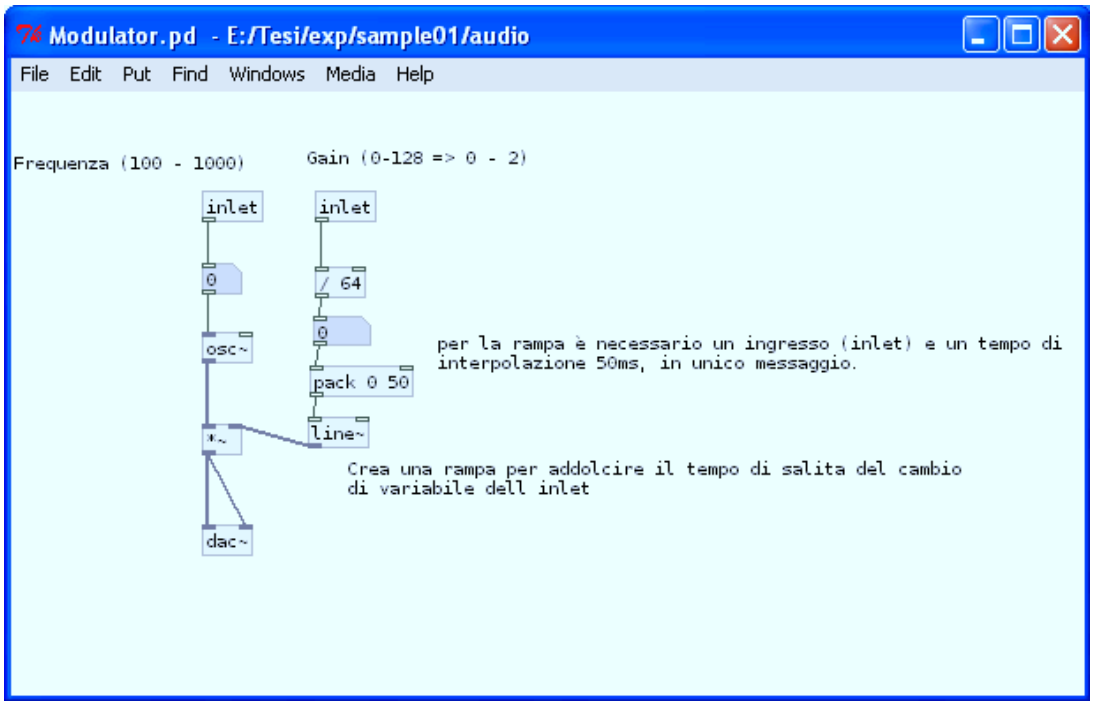

Figura 3.2: *Finestra di modulator.pd per Sample01*

3.1.2 Multimodal01

La variabile **X** *(figura 3.1)* é scalata in Matlab ed esprime, in un range da 0 a 128, la posizione del cursore del Phantom sul gradino. 0 indica che il cursore é all'estrema sinistra, mentre 128 indica che é all'estrema destra. La sublet *modulator* si occupa anche di fare il panning del feedback audio grazie a questa variabile.
La variabile **Y** *(figura 3.1)* é anch'essa giá scalata da Matlab ed esprime, in un range da 100 a 900, la frequenza in Hz che deve essere riprodotta dal modulatore. Questa grandezza legata allo spostamento verticale del cursore del Phantom sopra il gradino *(figura 1.4)*. Di conseguenza si sentirá una variazione in frequenza considerevole in corrispondenza all'alzata del gradino. Il gradino é stato programmato in modo tale da inviare le coordinate del cursore del Phatom solo quando si sta effettivamente toccando il gradino virtuale.
La variabile Z ha la stessa funzione di *Sample01* mentre la variabile W invia un segnale booleano al modulatore in modo di mutare l'uscita audio il soggetto non stia pi toccando il gradino. Questa variabile é necessaria altrimenti resterebbe in memoria l'ultima frequenza rimasta mandata alla porta di ricezione.

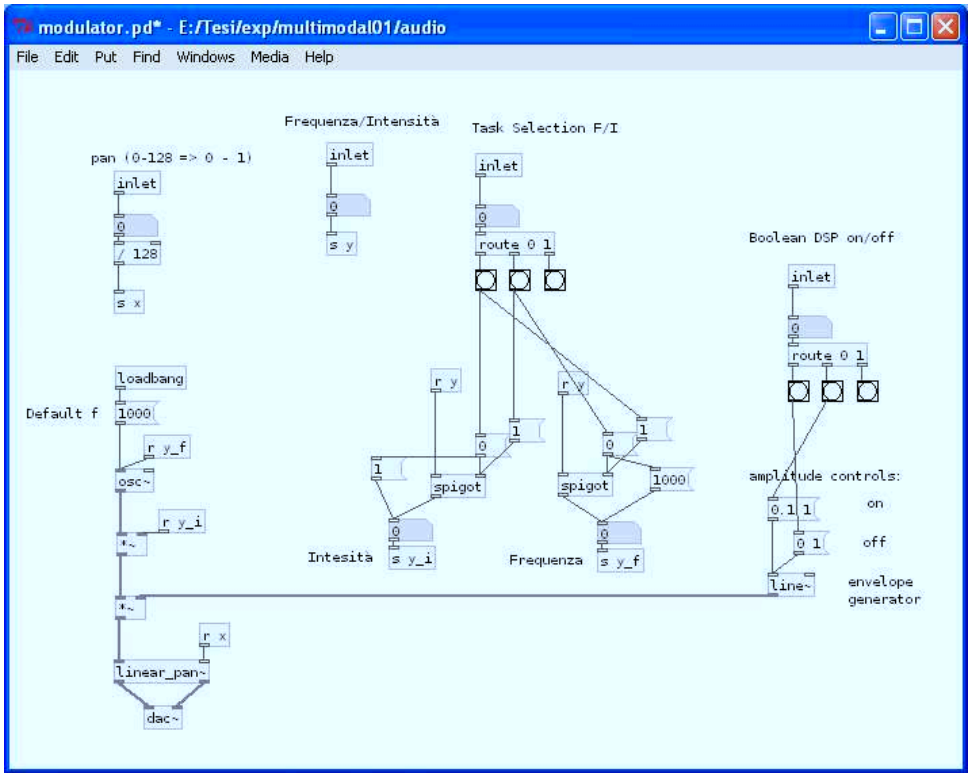

Figura 3.3: *Finestra di modulator.pd per Multimodal01*

3.2 OSC

Il protocollo OSC (Open Sound Control Protocol) [6] é stato ideato nel 1983 per stabilire una connessione network tra computer e dispositivi multimediali. Questo protocollo non trasmette alcun segnale audio o media ma semplicemente messaggi (o dati) come ad esempio pitch e intensitá di note musicali da riprodurre, segnali di controllo come volume, vibrato, panning e segnali di sincronizzazione. i messaggi inviati sono sotto forma numerica. OSC é un protocollo di trasmissione che permette a strumenti musicali, computer ed altri dispositivi multimediali di scambiare i dati precedentemente descritti in tempo reale attraverso una semplice rete interna (TCP/IP, Ethernet) o internet. Il protocollo di scambio dati su rete utilizzato é lUDP.
In Pure Data sono giá presenti le istruzioni per elaborare pacchetti OSC. Su Matlab si sono dovute utilizzare le librerie *cstruct* [2] per costruire il pacchetto OSC e *pnet* [7] per inviarlo tramite protocollo UDP. La cartella che contiene queste due librerie viene inclusa all'avvio dell'ambiente multimodale e deve essere posta nel file system ad un livello superiore rispetto a *Sample01* e*Multimodal01* .
Per praticitá e per rendere disponibile l'invio di pacchetti tramite protocollo OSC via matlab, si é pensato di realizzare una funzione **sendOSC.m** *(si veda l'appendice A.1 per il codice)* dedicata a questo con la seguente firma:

```
function sendOSC ( packet, sock)
```

Dove **packet** é un vettore numerico composto da quattro campi che corrispondono alle variabili X, Y, Z, W citate nel paragrafo 3.1.1 e **socket** é il socket definito per mezzo di *pnet* che permette la trasmissione del pacchetto.

[2] http://cstruct.rubyforge.org/

Capitolo 4

Rendering Aptico e Multimodale

La gestione dei soggetti e la comunicazione via protocollo OSC sviluppati in *Sample01*, sono state progettate per favorire lo sviluppo di altre applicazioni.
Multimodal 01 é sviluppato partendo da *Sample01* e ha l'obiettivo di applicare un feedback aptico ad un gradino virtuale. Per fare questo é necessario installare il dispositivo Phantom e interfacciarlo con Matlab e Pure Data, per realizzare la gestione dei dati e dell'audio rispettivamente. La criticitá iniziale é quella di rendere accessibili le caratteristiche dell'ambiente multimodale a tutti gli strumenti sviluppati, mantenendo la loro indipendenza.
Inizialmente si é esplorata la direzione di progettare l'ambiente multimodale in linguaggio c++ in maniera tale da accedere alle coordinate del cursore del Phantom per poi inviarle a Matlab via OSC. Questa soluzione si é rivelata troppo laboriosa e poco elastica a possibili variazioni dell'ambiente multimodale.
Un requisito fondamentale nella progettazione del codice c++, é l'utilizzo di una libreria che importasse del codice X3D con le informazioni dell'ambiente in tre dimensioni. Nella documentazione della libreria H3DAPI, é presente un richiamo al software H3DViewer che interpreta codice X3D contenente istruzioni appartenenti a H3DAPI e ne fa il rendering aptico. H3DAPI si interfaccia con i dispositivi aptici come il Phantom dell'attuale esperimento. H3DViewer rende inoltre disponibile l'utilizzo degli script python per gestire alcuni eventi descritti dal codice X3D.

4.1 Strumenti

4.1.1 Phantom

Il PHANTOM desktop ® figura 4.1 é un dispositivo aptico che viene utilizzato soprattutto per la ricerca. É composto da una base rotante su cui é montato un braccio con uno snodo centrale per poi finire con un ultimo snodo nell'estremitá (Stylus) che va impugnata come una penna. Questo dispositivo é in grado di fornire il feedback di forza lungo i tre assi *x, y,z*.

Dopo aver installato i driver per il corretto funzionamento della periferica é necessario installare la libreria *OpenHaptics Toolkit* reperibile gratuitamente sul sito della casa prodruttrice (*vedi*

Figura 4.1: *Dispositivo Aptico Phantom Desktop*

appendice A). Questa libreria permette di accedere alle API proprie del Phantom, da c++ e da altri tools di programmazione come H3DAPI che verrá descritta in seguito.

4.1.2 X3D

X3D é un linguaggio per la descrizione di ambienti virtuali interattivi OpenSource. É stato sviluppato dal Web 3D Consortium come evoluzione del VRML, é basato su XML, é un formato non proprietario ed é stato standardizzato dall'ISO nel 2004. X3D ha un ricco set di funzioni per l'utilizzo negli ambiti: ingegneristico e scientifico, CAD e architettura, biomedico, modellizzazione e simulazione, multimedia, intrattenimento, istruzione, e altro ancora. Lo standard comunicazione in tempo reale di dati 3D tra diverse applicazioni si é evoluto dalle sue origini come Virtual Reality Modeling Language (VRML) allo standard molto piú maturo e sviluppato X3D.
Lo scopo principale di X3D é la descrizione di ambienti virtuali interattivi. Sottolineiamo le entitá piú interessanti che possono essere descritte:

- **Oggetti** *descrive la posizione, la geometria e il colore degli oggetti.*
- **Interattivitá** *descrive l'interazione dell'ambiente con l'osservatore, ad esempio il click su di un oggetto.*
- **Scripting** *permette di manipolare, attraverso linguaggi di scripting come Javascript o Python, l'ambiente virtuale.(Routing, vedi paragrafo 4.2.3)*

4.1.3 H3Dapi

H3DAPI é una API multipiattaforma, con licenza open-source e commerciale, disponibile per il download all'indirizzo *www.h3d.org*. H3DAPI é scritto interamente in C++ e usa OpenGL per il rendering grafico e HAPI per il rendering aptico. Usando la sintassi X3D si possono creare ambienti virtuali anche animati. Utilizzando lo scripting Python possono essere estese le animazioni e i comportamenti degli oggetti. Questi ambienti contengono sia codice per la grafica che per il feedback aptico.
H3DAPI viene fornito con una serie di scene semplici di esempio, che mostrano le caratteristiche di H3DViewer, l'interprete di codice X3D contenente istruzioni aptiche fornite da H3DAPI. H3DAPI utilizza la sintassi X3D e il concetto di nodi per la costruzione del ambiente virtuale. Un nodo fornisce una certa funzione nella scena. Ci sono nodi per definire la geometria nel mondo virtuale, per l'impostazione grafica e per le proprietá aptiche. I nodi possono essere utilizzati dagli eventuali script.

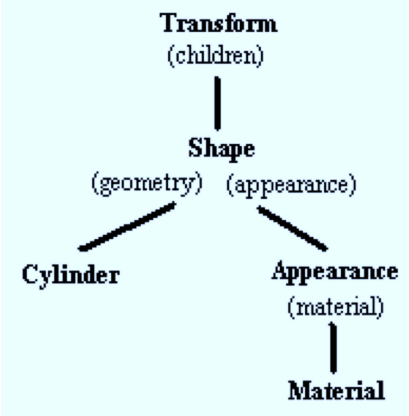

Figura 4.2: *Esempio di struttura ad albero di un file X3D*

4.2 Il Gradino Multimodale

Grazie al feedback aptico renderizzato da H3DAPI e all'ambiente virtuale costruito tramite codice X3D é possibile costruire un **gradino multimodale** con le seguenti caratteristiche:

- **Feedback Visivo** *Tramite il comando "extrusion" si é definito un poligono in 2D che poi viene esteso a tre dimensioni per ottenere il gradino desiderato*
- **Feedback Aptico** *Con lo stylus é possibile navigare sulla superficie del gradino percependo il feedback di forza quando si va a collidere il gradino*
- **Feedback Audio** *Attraverso uno script Python vengono inviate le coordinate dello stylus a Matlab che le elabora e le inoltra a Pure Data per elaborare il rendering audio*

4.2.1 Feedback Visivo

Un gradino tridimensionale puó essere visto come la somma di piú solidi, il primo tentativo é stato combinare piú piani per ottenere il gradino. Purtroppo il punto in cui si intersecavano i solidi, il feedback aptico fornito dal Phantom non rispettava la realtá virtuale rappresentata nel monitor. Di conseguenza si é trovato un modo piú veloce di rappresentare solidi 3D con una sola riga di codice.
Per costruire il gradino si é definito un poligono in due dimensioni(figura 4.2) attraverso il comando *"Extrusion"*. L'estrusione é un processo di produzione industriale di deformazione plastica che consente di produrre pezzi a sezione costante (ad esempio tubi, barre, profilati, lastre). É importante che le coppie di punti specificate nel campo *"crossSection="* siano ordinate in senso orario e che formino un poligono e non una serie di segmenti aperti. Di seguito il codice per definire il gradino virtuale.

```
<Extrusion DEF='GRADINO' convex='false' solid = 'true'
  crossSection='-0.22 -0.01 -0.22 0.06 0 0.06 0 0 0.22
                0 0.22 -0.01 -0.22 -0.01'
  spine='0 0 0.0 0.0 0 -0.22' />
```

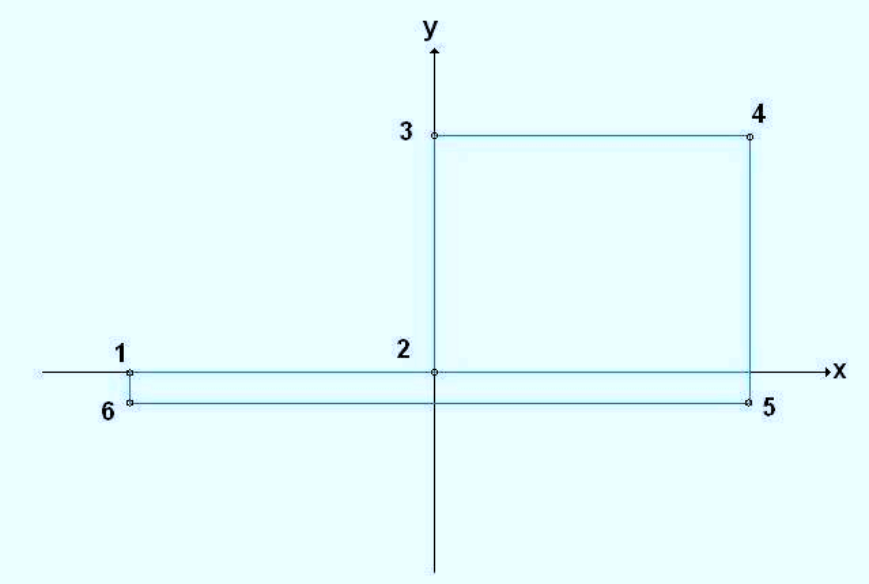

Figura 4.3: *Poligono in 2 dimensioni definito da "extrusion"*

Dove *"spine="* definisce una lista di punti 3D di una curva lineare a tratti che formano una serie di vertici connessi, aperta o chiusa. Questo é il percorso lungo cui la *" crossSection"* viene estrusa. Nel caso particolare il poligono di figura viene:

1. Estruso lungo l'asse Z in verso negativo, di 0.22 metri.
2. Traslato nella direzione positiva dell'asse Z della metá della larghezza del gradino (0.11 metri) per centrarlo sull'asse X (vedi figura 4.4).

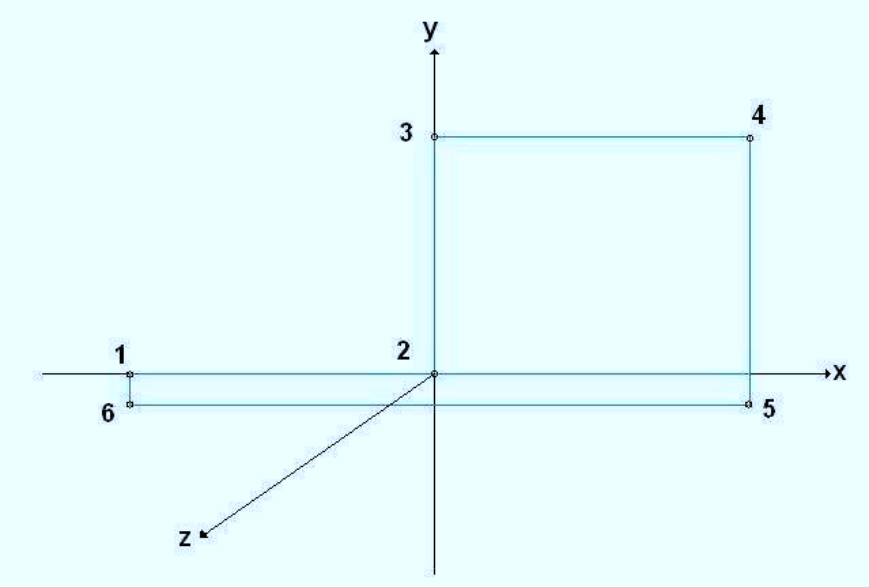

Figura 4.4: *Orientamento degli assi X3D per il comando "spine"*

Le coordinate prima descritte, sono espresse in Metri. Per ottenere l'unitá di misura in scala reale (Ad esempio: 10 cm di spostamento dello stylus corrispondono a 10 cm nell'ambiente multimodale) bisogna aprire il tool "H3DLoad settings" e sul menu a tendina **"Haptics Device"** selezionare la voce **Any PHANToM device (1to1)**. I componenti del menu a tendina non sono altro che i file di configurazione delle diverse interfacce poste nella cartella: *"C://H3D/H3DAPI/settings/common/device"*. Il file di configurazione del Phantom é nella cartella *"C://H3D/H3DAPI/settings/display/None/device"*. Si possono creare multipli e sottomultipli dell'unitá di misura modificando la matrice *"positionCalibration"* all'interno dei file di configurazione.

$$\begin{pmatrix} mx & 0 & 0 & 0 \\ 0 & my & 0 & 0 \\ 0 & 0 & mz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Dove mx, my, mz sono i coefficienti che moltiplicano rispettivamente la grandezza dello spostamento reale per ogni asse di riferimento (x, y, z).

Quando viene interpretato il file X3D, si viene automaticamente posizionati in un punto dello spazio virtuale, il campo visivo che si presenta definito tramite il comando *viewpoint*, si puó scegliere tra diversi viewpoint dal menu di H3D Viewer.

```
<Viewpoint description='Vista panoramica'  fieldOfView='0.9'
   position='0.0 0.12 0.44'/>
<Viewpoint description='Vista dall alto' orientation='0 1 0 -0.15'
  fieldOfView='1.2'  position='0.0 0.24 0.45'/>
```

dove i parametri sono:

- **description:** Descrizione testuale o suggerimento per la navigazione che viene mostrato per questo; Viewpoint.
- **fieldOfView:** Minimo angolo di visuale (in radianti) preferito per questo viewpoint. Un piccolo angolo di visuale corrisponde circa a un teleobbiettivo, un angolo di vista grande corrisponde ad un grandangolo. Modificare la distanza del Viewpoint dalloggetto puó migliorare lingrandimento;
- **orientation:** Rotazione (asse, angolo in radianti) del Viewpoint, relativamente alla direzione dellasse di default -Z nel sistema locale di coordinate;
- **position:** Posizione (x, y, z in metri) relativamente al sistema di coordinate (figura 4.4).

4.2.2 Feedback Aptico

Per ottenere il **Feedback Aptico** sul poligono definito dal comando *"Extrusion"* é necessario inserire *"FrictionalSurface"* all'interno dei tag *"Appearance"* in cui viene definito il materiale dell'oggetto:

```
<Shape>
  <Extrusion DEF='GRADINO'......... />
    <Appearance>
       <Material DEF='MATERIAL'  .......... />
       <FrictionalSurface />
    </Appearance>
</Shape>
```

Si puó personalizzare il feedback aptico fornito da **"FrictionalSurface"** tramite i seguenti parametri:

– *staticFriction:* determina l'attrito subito dal cursore quando si inizia uno spostamento lungo la superficie, se non viene specificato il valore di default é 0.1.

– *dynamicFriction:* definisce l'attrito dinamico percepito dall'utente quando ci si sta muovendo sulla superficie, se non viene specificato il valore di default é 0.4.

– *stiffness:* determina la durezza della superficie, cioé la quantitá di forza che esercita la superficie virtuale sullo stylus, se non é specificata viene posta uguale a 1.

– *damping:* determina la velocitá di smorzamento del movimento dello stylus sulla superficie (attrito viscoso), se si muove troppo in fretta si sentirá una forza opposta al movimento, se non viene specificato il valore di default é 0.

Le precedenti opzioni rispettano le formule:

$$F_{sup} = -stiffness \cdot F_{stylus} \tag{4.1}$$

$$F_{freno} = damping \cdot vel \tag{4.2}$$

– *staticFriction* e *dynamicFriction* assumono valori nell'intervallo [0,1].

– La forza esercitata dalla superficie F_{sup} (Eq. 4.1) é uguale e contraria alla forza esercitata dall'utente sullo stylus F_{stylus}. (Nel caso ***stiffness=1***).

– La forza di decelerazione F_{freno} (Eq. 4.2) é direttamente proporzionale al prodotto del fattore damping moltiplicato per la velocitá vel con cui l'utente muove lo stylus. (Nel caso specifico é pari a 0, non ho decelerazioni)

Oltre a **"FrictionalSurface"** ci sono altri tipi di feedback aptico per le superfici come **"SmoothSurface"** che é un caso particolare della precedente, **"MagneticSurface"** che rende un effetto magnetico alla superficie definita (impugnando lo stylus in prossimitá della superficie ci si sente attirati da essa), **"DepthMapSurface"** che permette di regolare la profonditá del piano in funzione della texture 2d definita da una scala di grigi. I grigi piú chiari corrispondono a una superficie piú alta mentre i grigi piú scuri corrispondono a una superficie piú bassa.

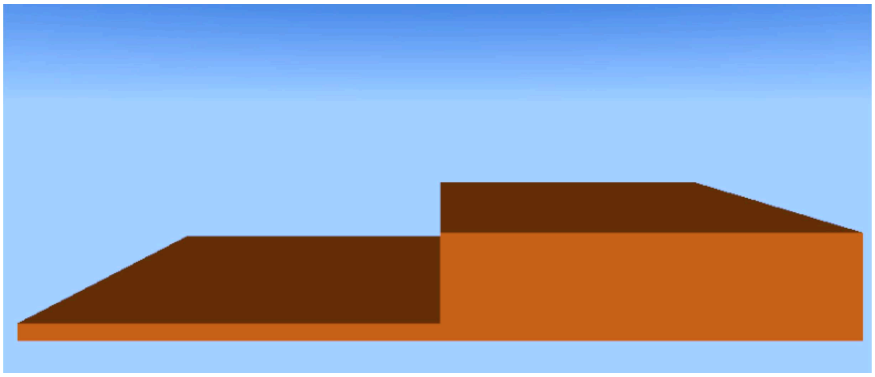

Figura 4.5: *Esempio di gradino contenente il comando “FrictionalSurface“*

4.2.3 Feedback Audio

Per implementare il **feedback audio** é stato necessario accedere alle coordinate dello Stylus del Phantom nel caso in cui l’utente stesse toccando il gradino virtuale. Lo script *Gradino.py* ha questa funzione: trasmettere le coordinate dello stylus via protocollo OSC a Matlab che dopo averle elaborate le trasmette a Pure Data. Tramite il routing dei nodi é possibile passare gli eventi accaduti nell’ambiente virtuale allo script collegato. Di seguito si analizzano le righe di codice X3D che implementano le funzioni precedentemente descritte:

```
<TimeSensor DEF=’TS’ cycleInterval=’0.01’
   loop=’true’ enabled=’false’ />
<PythonScript DEF="PS" url="Gradino.py" >
    <TimeSensor USE=’TS’ containerField="references"/>
</PythonScript>


<ROUTE fromNode="GRADINO" fromField="isTouched"
       toNode="PS" toField="color" />
<ROUTE fromNode="PS" fromField="color"
       toNode="MATERIAL" toField="diffuseColor" />
<ROUTE fromNode=’TS’ fromField=’cycleTime’
       toNode=’PS’ toField=’color’ />
```

Il comando Route connette dei campi di nodi diversi per permettere la trasmissione di eventi. In questo caso il nodo “GRADINO“ trasmette l’evento “isTouched“ al nodo *“PS“* che sarebbe il nodo che contiene lo script. Il nodo *PS* avrá accesso al campo *“color“* per modificare il colore del gradino quando viene toccato dal Phantom. Il codice:

```
return RGB( 1, 1, 0 )
```

Viene eseguito quando si é verificato l’evento *isTouched*, facendo assumere al gradino il colore giallo, mentre

```
return RGB( 1, 0, 1 )
```

viene eseguito quando lo stylus non tocca piú la superficie del gradino, rendendolo cosí di colore viola. Lo script Python viene richiamato solo nel caso in cui l’evento “isTouched“ é verificato.
Di seguito vediamo la porzione di codice dello script **Gradino.py** che legge la posizione dello stylus del phantom.

```
#leggo la posizione quando tocco il gradino con lo stylus
    di = getActiveDeviceInfo()
    if( di ):
      devices = di.device.getValue()
      if( len( devices ) > 0 ):
          hdev = devices[0]
          pos= hdev.trackerPosition.getValue()
          print pos #stampa la posizione nella console
```

Il comando *“hdev.trackerPosition.getValue()“* restituisce un vettore di posizione contenente le coordinate dello stylus del Phantom espresse con l’unitá di misura definita dalla matrice *“positionCalibration“*, che in questo caso é 1:1 con le dimensioni reali.
Dopo aver letto la posizione si prepara il pacchetto e si spedisce a Matlab via OSC sulla porta 999.

```
    initOSCClient("127.0.0.1" , port=999)  #osc sulla porta 999

    posx=str(pos.x) #costruisco il pacchetto
    posy=str(pos.y)
    posz=str(pos.z)

    pacchetto= "/pd_packet "+ posx +" "+ posy+" "+ posz+" "+"1 "
    #1 nell’ultima posizione vuol dire DSP on
    sendOSCMsg(pacchetto)
    closeOSC()
```

Il comando *"initOSCClient"* apre un socket con protocollo OSC, sull'indirizzo locale alla porta 999. Per inviare il pacchetto con protocollo OSC a Matlab é necessario specificare il tipo di pacchetto con la porzione di stringa *"/pd_packet "*. Alla fine della trasmissione é buona regola chiuedere il socket aperto precedentemente per rendere la porta 999 nuovamente utilizzabile.
Per utilizzare i comandi *"sendOSCMsg"* , *"initOSCClient"* e *"closeOSC"* é necessario installare delle librerie Python aggiuntive (vedi appendice A).

Capitolo 5

Conclusioni

Nei precedenti capitoli si é evidenziato come l'utilizzo di diversi programmi di sviluppo indipendenti tra loro possa costruire un unico ambiente virtuale multimodale. Matlab compie la gestione dei soggetti e dell'esperimento dirottando i pacchetti tra H3DViewer e Pure Data. H3DViewer realizza l'interfaccia grafica attraverso codice X3D e ne interpreta i comandi aptici, forniti dalla libreria H3DAPI realizzando il feedback aptico. Lo script Python viene utilizzato per accedere alle coordinate tridimensionali del Phantom per poi inviarle a Matlab che si occuperá di salvarle nel file system di tipo database. Dopo averle processate, Matlab invia pacchetti contenenti informazioni sulla frequenza e direzione del suono a Pure Data che si occuperá di realizzare il feedback audio.

5.1 Conclusioni

Le interazioni tra gli strumenti elencati negli scorsi capitoli hanno reso possibile la costruzione di un esemplare di gradino multimodale virtuale. Tramite il protocollo OSC é stato possibile stabilire la comunicazione tra i diversi programmi sviluppati per ottenere i vari feedback sensoriali posti negli obbiettivi iniziali. Il gradino multimodale si presenta cosí di facile utilizzo e fluido in ogni sua componente.
La modularitá dell'ambiente multimodale sviluppato, consente ad esempio, di modificare solo la parte visiva indipendentemente dagli altri strumenti. Questo é un punto di forza perché si possono estendere le funzionalitá di questo progetto ad ambienti multimodali piú complessi. Il protocollo di comunicazione OSC, permette di controllare il Phantom da remoto e posizionarlo in un'ambiente diverso da dove potrá essere la postazione per la gestione dei soggetti o del feedback audio. L'utilizzo di uno script Python per gestire gli eventi aptici puó aprire nuovi scenari ad ambienti multimodali dinamici.
Gli esperimenti sulla percezione dell'altezza spaziale del gradino, effettuati in collaborazione con il Dipartimento di Psicologia Generale dell'Universitá di Padova, sono giá in corso e da un'analisi preliminare si evince che il feedback audio induce una sovrastima dell'altezza spaziale. L'ambiente si presenta robusto ed affidabile, per questo Gli obbiettivi di questo lavoro di tesi si ritengono soddisfatti..

Appendici

Appendice A

Installazione Phantom e H3D

Per un utilizzo corretto del Phantom, dopo aver installato i driver della periferica é necessario installare il pacchetto *OpenHaptics Toolkit* [1]. Per accedere alle caratteristiche esposte nel capitolo 4 é necessario installare i pacchetti *H3DAPI* e *H3DViewer* reperibili nel sito del progetto H3DAPI. [2] E' necessario installare le librerie aggiuntive per Python*"SimpleOSC module"*[3] e *"pyOSC"*[4] che permettono di utilizzare con facilitá il protocollo OSC.

A.1 sendoOSC

Ecco il codice della funzione per inviare pacchetti tramite protocollo OSC su Matlab:

```
%Funzione che spedisce un pacchetto tramite protocollo OSC, con socket
%creato precedentemente dalla GUI
%packet= [x ,y, z, DSP_on]

function sendOSC ( packet, sock)


    %PREPARAZIONE PACCHETTO PER Pure Data
    pd_packet = [single(packet(1)) single(packet(2)) single(packet(3))
                 single(packet(4))];

    [attr datastring]=cstruct(pd_packet);
    [attr zerostring]=cstruct(single(0));

    %codifica OSC
    string=char(['/pd_packet' zerostring(1:2) ',ffffffff' zerostring(1:2)
                datastring]);
```

[1] http://www.sensable.com/products-openhaptics-toolkit.htm
[2] http://www.h3dapi.org/
[3] http://opensoundcontrol.org/implementation/python-simple-osc
[4] http://www.ixi-audio.net/content/body_backyard_python.html

```
    %scrittura della stringa nel buffer del socket
    pnet(sock, 'write', string);

    %spedizione del pacchetto
    pnet(sock, 'writepacket');

end
```

A.2 Gradino.py

Ecco il codice dello script Python per leggere la posizione del Phantom e inviare pacchetti tramite protocollo OSC a Matlab:

```
# Script che invia i pacchetti tramite protocollo UDP quando e' verificato l'evento
# isTouched
#Copyright Alberto Ziliotto

#import the H3D fields and types
import time # in this example we will have a small delay in the while loop
from H3DInterface import *
#from H3DUtils import *
from simpleOSC import *

# The Color class is of type SFColor and its value is determined by
# the SFBool field that is routed to it. If its value is 1 the color
# is red, otherwise it is blue.

nodes = references.getValue() #rende l'accesso al timer dal
                            python cos da modificare le sue variabili

class Color( AutoUpdate(TypedField( SFColor,MFBool, SFTime)) ): #era SFBool se
                                                    #non usavo isTouched su X3D
  def update( self, event ):
    global nodes
    ts = nodes[0]
    # print 'event.getValue='
    # print event.getValue()

    initOSCClient("127.0.0.1" , port=999) # inizializzo il client OSC

    if ((isinstance(event.getValue(), float))): #l'evento  di tipo float
                                                quando si verifica isTouched
      #leggo posizione cursore quando tocco la sfera ed  abilitato il timer
      di = getActiveDeviceInfo()
      if( di ):
        devices = di.device.getValue()
        if( len( devices ) > 0 ):
            hdev = devices[0]
            pos= hdev.trackerPosition.getValue()
            print pos
      #fine lettura
```

```
    if (pos.z==0):
       if (pos.y==0):
          if (pos.x==0):
             pacchetto= "/pd_packet 0 0 0"+" 0 "  #0 vuol dire DSP spento
    else:
       posx=str(pos.x) #costruisco il pacchetto
       posy=str(pos.y)
       posz=str(pos.z)

       pacchetto= "/pd_packet "+ posx +" "+ posy+" "+ posz+" "+"1 "  #1 vuol dire
                                                                    #DSP acceso
    #print pacchetto
    sendOSCMsg(pacchetto) #invio il messaggio di test quando premo con il pulsante
                          #destro del mouse e il gradino cambia colore
    #time.sleep(0.05)    #piccolo ritardo per vedere se funziona lo script
    closeOSC()

    return RGB( 1, 1, 0 )

  else:   #(evento in cui tocco la sfera)

    if (len(event.getValue())>0):    #all'inizio crea un evento di default, io
                           #considero solo gli eventi touched uguali a [0] o [1]
      if(ts.enabled.getValue()):
        ts.enabled.setValue(False)
        print 'non stai toccando il gradino'
        di = getActiveDeviceInfo()
        if( di ):
          devices = di.device.getValue()
          if( len( devices ) > 0 ):
             hdev = devices[0]
             pos= hdev.trackerPosition.getValue()
        posx=str(pos.x) #costruisco il pacchetto
        posy=str(pos.y)
        posz=str(pos.z)

        pacchetto= "/pd_packet "+ posx +" "+ posy+" "+ posz+" "+"0 "
        #print pacchetto
        sendOSCMsg(pacchetto) #invio il messaggio di test quando premo con il
                  pulsante destrodel mouse e il gradino cambia colore

      else:                                          #cambio di stato del timer
        if (event.getValue()[0]):     #l'evento e [1] quando premo la superficie,
                                       quando tolgo il phantom dalla superficie e [0]
          ts.enabled.setValue(True)
          print 'stai toccando il gradino'
          #time.sleep(0.05)    #piccolo ritardo per vedere se funziona lo script
    else:
      print 'iniziato esperimento'
```

```
      return RGB( 1, 0, 1 )
  prev=0


# create an instance of the Color class.
color = Color()
```

Bibliografia

[1] Ying Ying Huang, 2010. *—Design and Evaluation of 3D Multimodal Virtual Environments for Visually Impaired People—Doctoral Thesis in Human-Computer Interaction KTH, Stockholm, Sweden 2010*

[2] Lahav O. and Mioduser D., 2008 *— Lahav O. and Mioduser D., 2008 — Haptic feedback support for cognitive mapping of unknown spaces by people who are blind, International Journal of Human-Computer Studies, Volume 66, issue 1, January 2008, pages 23- 35.*

[3] Passini R. and Proloux G., 1988 *— Passini R. and Proloux G., 1988 — Way Finding Without Vision: an experiment with congenitally blind people. — Environment and Behaviour 20, 227-252*

[4] Fletcher J.F., 1980. *—Spatial representation in blind children 1: development compared to sighted children.— Journal of Visual Impairment and blindess 74 (10), 318 - 385.*

[5] Lahav O. and Mioduser D., 2004 *— Lahav O. and Mioduser D., 2004 — Anticipatory Cognitive Mapping of Unknown Spaces by People who are blind using a Virtual Learning Environment, Proceeding of the 6th International Conference on Learning Sciences June 2004.*

[6] opensoundcontrol.org *— Informazioni reperite dal sito web di OSC opensoundcontrol.org*

[7] pnet.mexw32 *— Libreria precompilata sviluppata da Peter Rydesater — reperibile al 21/03/2013 all'indirizzo http://www.mathworks.com/matlabcentral/fileexchange/345*