UNIVERSITÀ DEGLI STUDI DI TRENTO

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA DELL'INFORMAZIONE E DELL'ORGANIZZAZIONE

---

# INTEGRAZIONE DI UNA PERIFERICA TATTILE IN UN SISTEMA DI "AUGMENTED REALITY" MULTI-UTENTE MULTIMODALE


**RELATORE:**

**DOTT. VALTER CAVECCHIA**

**TESI DI LAUREA DI:**

**MICHELE WELPONER**



**ANNO ACCADEMICO 2004-2005**

# INDICE GENERALE

**INTRODUZIONE** ..... **9**

**SINOSSI** ..... **12**

**1 STATO DELL'ARTE** ..... **14**

**1.1 COMPUTER GRAFICA E REALTÀ VIRTUALE** ..... **14**
1.1.1 TECNOLOGIE IN EVOLUZIONE ..... 14
1.1.2 DEFINIZIONE DI REALTÀ VIRTUALE ..... 15
1.1.3 AUGMENTED & MIXED REALITY ..... 17
1.1.4 I VANTAGGI DELL'AUGMENTED REALITY ..... 20
1.1.5 APPLICAZIONI DI REALTÀ VIRTUALE ..... 21
**1.2 SUPPORTI HARDWARE** ..... **25**
1.2.1 SISTEMI DI OUTPUT STEREOVISIVI ..... 25
1.2.2 SISTEMI DI INPUT TRIDIMENSIONALE ..... 29
1.2.3 DISPOSITIVI DI TRACKING MECCANICI ..... 34
1.2.4 DISPOSITIVI DI TRACKING ELETTROMAGNETICI ..... 34
1.2.5 DISPOSITIVI DI TRACKING OTTICI ..... 36
1.2.6 DISPOSITIVI DI TRACKING ACUSTICI ..... 37
1.2.7 DISPOSITIVI DI TRACKING INERZIALI ..... 37
**1.3 NAVIGAZIONE E MULTIMODALITÀ** ..... **39**

1.3.1 NAVIGAZIONE IN AMBIENTI VIRTUALI ..... 39
1.3.2 INTERATTIVITÀ E MANIPOLAZIONE ..... 39
1.3.3 MULTIMODALITÀ: SKETCH, INTERAZIONE BIMANULE E COMANDI VOCALI ..... 40
1.3.4 TEXT TO SPEECH ..... 42
1.3.5 SPEECH RECOGNITION ..... 42
**1.4 IL SISTEMA STUDIERSTUBE ..... 43**
1.4.1 L'API STUDIERSTUBE ..... 43
1.4.2 MULTIUTENZA E INTERCOLLABORATIVITÀ ..... 47
1.4.3 IL CONCETTO DI LAYER ..... 48
1.4.4 PERSONAL INTERACTION PANEL ..... 48
1.4.5 TRACKING DI OGGETTI MOBILI ..... 50
1.4.6 MULTI-TASKING ..... 50
1.4.7 CONTEXTS ..... 51
1.4.8 LOCALES ..... 53
1.4.9 CONTEXT CONTRO LOCALES ..... 53
1.4.10 WIDGET 3D ..... 57
1.4.11 CONCLUSIONI ..... 57

**2 ASPETTI SULL'IMPLEMENTAZIONE ..... 59**

**2.1 INTRODUZIONE ..... 59**
**2.2 ESTRAZIONE DEI DATI DAL DISPOSITIVO ..... 60**
**2.3 GESTIONE DEI DATI PROVENIENTI DAL GUANTO P5 ..... 64**
2.3.1 DESCRIZIONE DELLA GESTIONE DEI DATI ATTRAVERSO L'ESTENSIONE DELLA LIBRERIA OPENTRACKER ..... 64
2.3.2 DETTAGLI IMPLEMENTATIVI SULL'ESTENSIONE DELLA LIBRERIA OPENTRACKER PER LA PERIFERICA P5 ..... 67
2.3.3 LA CONFIGURAZIONE DEL SISTEMA ATTRAVERSO LO SCRIPTING XML ..... 68
2.3.4 FUNZIONAMENTO DEI MODULI IMPLEMENTATI ..... 74
2.3.5 LA STRUTTURA DATI ADOTTATA ..... 75
**2.4 TRASFERIMENTO DEI DATI DA OT A STB: L'ENGINE IMPLEMENTATO PER IL P5. 79**
**2.5 GEOMETRIA DEL GUANTO VIRTUALE ..... 84**
2.5.1 DEFINIZIONE DELLA STRUTTURA PER LA GERARCHIA DELLE GEOMETRIE ..... 85

2.5.2 INIZIALIZZAZIONE DEL SOGLOVEKIT ..... 86
2.5.3 POPOLAMENTO DEL CATALOGO ..... 87
**2.6 INTERAZIONE TRAMITE RICONOSCIMENTO VOCALE ..... 89**
2.6.1 SUPPORTO DI RICONOSCIMENTO VOCALE PER OPENTRACKER ..... 90

**3 RISULTATI E SVILUPPI FUTURI ..... 95**

**3.1 RISULTATI ..... 95**
**3.2 LAVORI FUTURI ..... 97**

**4 BIBLIOGRAFIA ..... 99**

# INDICE DELLE FIGURE

*Figura 1: Mixed Reality e Augmented Reality (SigGraph, 2000)* ..... *17*
*Figura 2: Esempio di sovrimpressione di informazioni sintetiche alla realtà* ..... *19*
*Figura 3: Sistema a sei gradi di libertà (6DOF)* ..... *20*
*Figura 4 : Il gioco di Realtà Virtuale America's Army*..... *23*
*Figura 5: Sistema AR applicato alla diagnostica medica, come ausilio durante un'operazione chirurgica*..... *24*
*Figura 6 : Sony HMD GlassTron (Sony, 2004)* ..... *26*
*Figura 7 : Shutter Glasses prodotti da Barco (Barco, 2004)*..... *27*
*Figura 8 : Il sistema di visualizzazione "Virtual Table Baron" prodotto da Barco (Barco, 2004)*..... *28*
*Figura 9 : Sistema di visualizzazione "Back Projector Wall" bicanale prodotto da Barco (Barco, 2004)* ..... *29*
*Figura 10 : Sistema di visualizzazione a 9 proiettori prodotto da Barco (Barco, 2004)* ..... *29*
*Figura 11 : SpaceMouse* ..... *31*
*Figura 12 : DataGlove*..... *31*
*Figura 13 : Prototipo di force-feedback DataGlove (DataGlove, 2004)* ..... *32*
*Figura 14 : Esempio di motion-capture usata nel film Lord of The Rings*..... *33*
*Figura 15 : Tracker meccanico Premium 1.5 prodotto della linea PHANTOM commercializzato da SensAble*..... *34*
*Figura 16 : Tracker elettromagnetico Fastrak prodotto da Polhemus* ..... *35*
*Figura 17 : Dispositivo di tracking ottico a sfere retroriflettenti* ..... *36*
*Figura 18 : InertiaCube2, il più piccolo tracker inerziale al mondo (InertiaCube2, 2004)* ..... *38*
*Figura 19 : La struttura software di Studierstube*..... *45*
*Figura 20 : Tipico ambiente di lavoro di Studierstube* ..... *47*
*Figura 21 : Personal Interaction Panel (PIP) per l'interazione con l'ambiente MR in Studierstube* . *49*
*Figura 22 : Interfaccia a documento multiplo in ambiente 3D. Il fuoco è sulla finestra 3D di destra – indicato dal contorno blu – e può essere manipolata usando gli elementi presenti sul PIP.*..... *52*
*Figura 23 : La divisione degli elementi in più locale può non soddisfare tutte le esigenze (sopra). La replica completa di tutti gli elementi porta all'eccessivo riempimento del display (centro). La replica su richiesta degli elementi permette una personalizzazione conveniente dei locale (sotto).*..... *55*
*Figura 24 : Locali multipli possono coesistere in Studierstube. Possono essere utilizzati per configurare differenti dispositivi di output e per supportare la collaborazione remota.* ..... *56*
*Figura 25 : Dispositivo di interazione basato su tracking ottico P5 sviluppato da Essential Reality* . *61*

*Figura 26: Schema della connessione tra OpenTracker e Studierstube* ..... *66*
*Figura 27: La creazione dell'albero di OpenTracker dallo script XML*..... *69*
*Figura 28 : Rappresentazione grafica corrispondente all'albero generato tramite il codice XML* .... *72*
*Figura 29 : Diagramma UML delle classi che estendono le funzionalità di OpenTracker* ..... *77*
*Figura 30 : Schema generale di rappresentazione di un engine*..... *80*
*Figura 31 : Collegamento di più engine per animazioni sincrone tra diverse parti della scena*..... *81*
*Figura 32: La catena di engine adottata*..... *81*
*Figura 33 : Input ed Output implementati nel P5Engine*..... *83*
*Figura 34 : Lo scene-graph dei nodi relativo alla geometria del modello P5* ..... *88*
*Figura 35: Diagramma di inizializzazione di OpenTracker con SAPI* ..... *91*
*Figura 36: Meccanismo di trasmissione dei dati con OpenTracker mediante comando vocale*..... *91*
*Figura 37 : Ambiente Studierstube con interfaccia di interazione P5* ..... *95*

Giunto al termine del percorso universitario, ringraziare le persone che mi sono state vicine in questi anni più che un dovere è una necessità. Un sentito ringraziamento va alla Fondazione Graphitech, nella persona del Dott. Raffaele De Amicis, per il supporto tecnico ed operativo, al Dott. Giuseppe Conti e alla Dott.sa Giuliana Ucelli per la pazienza e la comprensione dimostrate nel seguirmi in questo lavoro di tesi e per la capacità di stimolarmi ed accompagnarmi con entusiasmo durante tutto il suo sviluppo.

Un grazie particolare va ad Elisa per essere sempre stata disponibile ad ascoltarmi e per avermi aiutato ad affrontare questo periodo con ottimismo.

Un caloroso grazie a tutti gli amici con i quali ho trascorso questi anni di università e a tutte le persone che mi sono state vicine.

Ultimi ma non ultimi i miei genitori, in particolare la mia mamma Emmanuela. Più di tutti ha sempre sognato questo momento, condividendo con me le ansie e le gioie in tutti questi anni di studio. Voglio dedicare questa mia tesi a loro, alla memoria di mio padre che ha potuto vivere troppo poco e a mia madre per tutti i sacrifici e tutti i malumori che ha dovuto sopportare e per l'incessante sostegno che, con tanta dolcezza, mi ha sempre dato.

Michele

*" Virtual Reality: A computer system used to create an artificial world in which the user has the impression of being in that world and with ability to navigate through the world and manipulate objects in the world. „*

C. Manetta and R. Blade in "Glossary of Virtual Reality Terminology" in the International Journal of Virtual Reality, Vol.1 Nr.2 1995.

# Introduzione

L'obiettivo della ricerca nel campo dei sistemi di Realtà Virtuale è quello di creare una ambientazione "sintetica" tale che, l'utente che vi ci è immerso non abbia alcun dubbio sulla reale consistenza degli oggetti che lo circondano. Dato che il mondo è costituito da un numero infinitesimale di dettagli, è lecito pensare che questo obiettivo non potrà mai essere pienamente raggiunto. La realtà può venir approssimata, ma probabilmente mai perfettamente copiata. Ciononostante, pur non essendo gli attuali sistemi di *VR* in grado di trasmettere un senso di completa immersione, crescenti settori di ricerca sono impegnati nello studio di dispositivi capaci di estendere la sfera reale con informazioni di tipo sintetico ad essa sovrimpresse, utili alla velocizzazione e al miglioramento qualitativo dell'adempimento di taluni compiti e attività umane.

L'era delle schede grafiche capaci di bassa risoluzione e dei dispositivi di *input* inaffidabili è ormai passata, finalmente abbiamo raggiunto un livello tale per cui i dispositivi *hardware* e le piattaforme *software* rendono possibile l'esperienza della Realtà Virtuale. Anche se tra pochi anni l'attuale tecnologia sarà considerata vecchia ed obsoleta, possiamo già intravedere come l'utilizzo delle moderne tecnologie porterà al raggiungimento degli attuali obiettivi della ricerca nel campo della Realtà Virtuale. Il costante sviluppo della tecnologia *hardware* e la rapida crescita di *performance* degli attuali computer non basta, essa deve essere supportata da strutture

*software* qualificate, capaci di pilotare sempre più complessi sistemi di ambientazione virtuale in termini di *performance*, scalabilità e facilità di utilizzo, cosicché il programmatore possa concentrare la propria attenzione sul contenuto, l'anima delle applicazioni virtuali.

La libreria *API Studierstube* (Studierstube , 2004) è una piattaforma *software* nata per lo sviluppo di applicazioni *VR*, originariamente implementata per far funzionare uno speciale dispositivo utilizzato in *Augmented Reality*, secondo il paradigma del "*Personal Interaction Panel*". Secondo questo paradigma per mezzo di un semplice dispositivo a due mani, costituito da una tavoletta ed una penna, è possibile fornire all'utente un modo naturale e intuitivo di controllo su applicazioni 3D (Szalavári, 1996). In seguito essa venne estesa e trasformata in un completo sistema di Realtà Virtuale. *Studierstube* permette infatti di eseguire parallelamente numerose applicazioni e fornisce all'utente facili strumenti per poter passare arbitrariamente da una all'altra, in modo del tutto simile a quello di un virtuale sistema operativo, sebbene alcuni termini e concetti rivelino il suo legame primordiale al paradigma del "*PIP*".

I moderni sistemi operativi forniscono una interfaccia grafica bidimensionale e un *set* di elementi di controllo prefabbricati, atti ad aiutare lo sviluppatore di applicazioni, che si adeguano nel *look* e nella funzionalità all'ambiente del sistema operativo stesso. Così fa anche la *API Studierstube* mediante il paradigma della tavoletta e della penna. La nostra intenzione è quella di riformulare questi elementi dalle fondamenta. Abbiamo esteso la possibilità di *tracking* del sistema di *Augmented Reality Studierstube* permettendo un nuovo tipo di interazione multimodale basato sull'utilizzo di una periferica tattile grazie alla quale sarà possibile, in un prossimo futuro, definire una serie di comandi gestuali e vocali interpretati secondo una predefinita ma dinamica serie di regole euristiche.

Nell'ambito della definizione di nuovi strumenti e sistemi di interazione in applicazioni di Realtà Virtuale ho avuto la possibilità di condurre tale attività di ricerca sfruttando l'utilizzo di un particolare tipo di periferica a sei gradi di libertà, il

guanto di realtà virtuale P5 prodotto e commercializzato da *Essential Reality* (Essential Reality, 2004), in seguito indicato nel testo come "P5". Il presente lavoro di tesi è stato reso possibile grazie alla collaborazione con il centro Fondazione Graphitech di Rovereto nell'ambito del progetto *InSIDe* (*Intelligent Styling System for Industial Design*) il cui principale obiettivo è quello di fornire una soluzione innovativa che possa riempire le lacune, determinate dalla scarsa funzionalità dell'interfaccia propria dei sistemi *CAD* tradizionali, attraverso la specificazione di una nuova *tool* specificatamente disegnata per soddisfare i bisogni di stilisti ed ingegneri.

# Sinossi

Nel primo capitolo verranno date le definizioni dei termini *Virtual Reality (VR), Mixed Reality (MR)* ed *Augmented Reality (AR),* seguirà la presentazione delle tecnologie e dei dispositivi attualemente utilizzati in *VR*. Dopo una breve introduzione di carattere storico si procederà con l'analisi delle differenze e dei vantaggi introdotti dall'utilizzo di queste nuove tecnologie discutendo il modo in cui esse sono in rapporto con la computer grafica. Si parlerà delle possibilità e delle modalità di interazione offerte dai dispositivi *VR*, verranno analizzate talune problematiche relative all'interfaccia uomo-macchina e presentati alcuni aspetti sulla multimodalità ponendo attenzione sulla funzionalità e sulle modalità di navigazione in ambienti di realtà virtuale. Nei paragrafi successivi si parlerà della piattaforma *API* di *Mixed Reality Studierstube*, ponendo attenzione ai metodi di interazione da essa previsti.

Nel secondo capitolo verranno spiegate le motivazioni e le scelte effettuate riguardo all'implementazione del progetto. Si passerà quindi ad una dettagliata analisi dei metodi utilizzati per l'estrazione dei dati dalla periferica di *input*, per la gestione e la formattazione di tali dati, per il loro passaggio a *Studierstube*, utilizzando quando necessario grafici esplicativi. Verranno discusse le strutture relative alla geometria del

modello virtuale del guanto, e i metodi di implementazione del sistema di riconoscimento vocale *Speech SDK 5.1 for Windows® applications* (SAPI, 2004).

Nel terzo capitolo verranno analizzati i risultati ottenuti e verranno formulate alcune ipotesi su possibili integrazioni future.

# 1 Stato dell'Arte

## 1.1 Computer Grafica e Realtà Virtuale

Il termine *Virtual Reality* o *Realtà Virtuale,* altrimenti detta *Realtà Sintetica,* è divenuto nell'ultimo decennio molto comune. Le novità introdotte da questa nuova tecnologia e la sua diffusione in un vasto numero di discipline hanno contribuito al successo del suo utilizzo, tanto che ormai risulta difficile fare una netta distinzione tra computer grafica e realtà virtuale.

### 1.1.1 Tecnologie in Evoluzione

Al giorno d'oggi risulta difficile immaginare un ingegnere, un architetto, un arredatore di interni o un designer lavorare senza l'ausilio di un sistema che adotti il computer per l'elaborazione grafica. Il successo ottenuto dalla vendita di software, soprattutto di video-games sempre più raffinati e avidi di risorse hardware, ha portato ad un incremento esponenziale della tecnologia informatica, alla commercializzazione di schede grafiche ed unità di processo sempre più sofisticate e performanti. A partire dalla fine del ventesimo secolo inoltre è iniziato un vorticoso processo di sviluppo di nuovi e sempre più potenti processori grafici. L'aumento della disponibilità di unità di prodotto informatico nell'industria del *pc* dovuto al

progresso tecnologico, unito alla concorrenza tra le varie case produttrici, ha fatto cadere i prezzi e reso la computer grafica accessibile anche all'utente medio. Il mondo della grafica tridimensionale ha permesso la visualizzazione a video di immagini ed animazioni relative alla ricostruzione di oggetti, ambienti, e strutture più o meno accessibili, ancora da costruire o addirittura non esistenti.

Le innovazioni portate dall'evoluzione di queste nuove tecnologie e le possibilità da esse offerte, hanno aperto all'uomo nuovi orizzonti; l'utente oggi non si accontenta della semplice visualizzazione, vuole avere la possibilità di immergersi in questi ambienti virtuali, avere la possibilità di creare, manipolare ed interagire con gli oggetti appartenenti a quel mondo. La Realtà Virtuale risponde in modo corretto a questa esigenza. Essa costituisce infatti una interfaccia uomo-macchina di tipo avanzato in grado di simulare una ambientazione virtuale interattiva, nella quale gli utenti partecipanti possono muoversi ed interagire autonomamente.

### 1.1.2 Definizione di Realtà Virtuale

La popolarità raggiunta all'inizio del 1990 grazie anche ai mezzi di comunicazione di massa ha fatto si che l'interesse verso questa nuova tecnologia crescesse rapidamente. Ciononostante, tra la gente comune, sono ancora poche le persone che possono vantare una buona conoscenza dei principi e delle problematiche che si devono affrontare nello sviluppo di un sistema *VR* (Mazuryk, 1996).

Come Pier Luigi Cappucci (1993) fa notare, è lecito pensare che il termine *Realtà Virtuale* sia fondato su di un'infelicità terminologica. Con le parole di Cappucci: "se per *virtuale* si intende *irreale, fittizio* allora, per esempio, anche uno spettacolo teatrale, un affresco, un film, un romanzo, un videogioco, sono *Realtà Virtuali*, là dove congegnano e offrono, mediante le proprie tipiche modalità comunicative, mondi possibili, inventati;" Scrive Pier Luigi Cappucci: "l'intero universo simbolico della rappresentazione, compreso quello del sogno, è *virtuale*." (Cappucci, 1993)

Tale tesi è parzialmente condivisa da Nicholas Negroponte (1993), fondatore e direttore del *Media Lab* del *M.I.T.* di Boston il quale sostiene che il successo ottenuto dall'utilizzo del termine *Realtà Virtuale* deriva direttamente dalla natura semantica del suo idioma, un ossimoro (Negroponte N., 1993), cioè una figura logica che consiste nell'accostare, nella medesima espressione, parole di senso opposto (p. e. un morto vivente) (Garzanti Linguistica, 2003).

In altre parole l'abilità della realtà virtuale consiste nel lasciare credere che essa non si sia ancora realizzata, togliendo nel frattempo il senso della possibilità di una realtà diversa.

Storicamente, la prima vera idea di realtà virtuale venne presentata da Ivan Sutherland quando, nel 1965, scrisse che per far si che "quel mondo virtuale" contenuto in "quella finestra" apparisse reale, sarebbe stato necessario dotarlo di una ambientazione sonora fedele alla realtà e di un'interfaccia in grado di reagire rapidamente ai comandi impartiti dall'utente (Mazuryk, 1996). Conti (2002) suggerisce citando Burdea et al. (1994, p. 2) che comunemente oggigiorno si tende ad associare il termine "Realtà Virtuale" con un mero insieme di tecnologie quali occhiali stereovisivi e dispositivi per feedback o guanti virtuali mentre evidenzia che, come sostiene Sherman et al. (1992, p. 24) la Realtà Virtuale non è né un' industria standardizzata né una semplice collezione di tecnologie.

E' evidente che non è semplice dare una definizione precisa ed esauriente di Realtà Virtuale; molti autori hanno provato a dare una possibile e convincente definizione. Gigante M. (1993) ad esempio sostiene che la caratteristica più importante di un sistema *VR* sia quella di dare l'illusione di partecipare ed appartenere ad un ambiente piuttosto che la sensazione di essere un osservatore esterno. Egli aggiunge che, per ottenere ciò, la *VR* si basa sull'utilizzo di un sistema tridimensionale composto da periferica display stereoscopica, di un sistema di interazione tramite guanto/corpo e di un sistema audio biauricolare. Secondo Gigante la *VR* è dunque un'esperienza immersiva multi-sensoriale. Jargon (1995) invece considera la VR come simulazione al computer basata sull'utilizzo di grafica

3D e dispositivi quali il "*Data Glove*" per permette all'utente di interagire con la simulazione. Cruz-Neira (1993) scrive che la *VR* fa riferimento ad ambienti immersivi creati al computer, interattivi, multi-sensoriali, tridimensionali ed incentrati sull'utente, nonché alla combinazione delle tecnologie richieste per realizzare tutto ciò. Infine Schweber (1995) puntualizza che la *VR* permette di vedere e navigare con sei gradi di libertà in un mondo tridimensionale costruito in tempo reale.

### 1.1.3 Augmented & Mixed Reality

In letteratura i termini *Augmented Reality* (*AR*) e *Mixed Reality* (*MR*) compaiono in associazione al termine *Realtà Virtuale*. Sebbene i sistemi *AR* impieghino alcune delle tecnologie usate nei sistemi di Realtà Virtuale è preferibile utilizzare il termine *Augmented Reality* nel caso in cui non sia necessario l'utilizzo di una ambiente totalmente virtuale (Dubois et al., 1999), ma si richieda comunque una integrazione della realtà mediante strutture di tipo virtuale.

Infatti mentre la Realtà Virtuale tende a sostituire in toto la visione del mondo reale, la *Augmented Reality* mira, lasciando una connessione con la "realtà", ad arricchirne la rappresentazione mediante l'utilizzo di informazioni utili per l'esecuzione di taluni compiti complessi.

*Mixed Reality*, più specificatamente *Augmented Reality* è il termine comunemente usato per riferirsi ad ambienti che combinano oggetti reali e virtuali con rappresentazioni visive dello spazio reale e virtuale (vedi Figura 1).

**Figura 1: Mixed Reality e Augmented Reality (SigGraph, 2000)**

L'evoluzione della *Augmented Reality* ebbe inizio agli albori della computer grafica quando verso la fine degli anni '60 Ivan Sutherland condusse una ricerca sui visori *Head Mounted Display* (*HMD*) (Sutherland, 1968) presso l'università di Harvard e dell'Utah.

Gli *HMD* possono essere usati in abbinamento ad una/due videocamera oppure possono essere costituiti da materiale semitrasparente attraverso il quale è possibile vedere il mondo reale; un computer collegato all'*HMD* elabora ed arricchisce in tempo reale l'immagine reale ponendo in sovrimpressione informazioni di tipo digitale (vedi Figura 2). Il lavoro condotto durante la ricerca operata da Sutherland rappresenta una pietra miliare della storia dell'*AR* sebbene il prototipo realizzato fosse capace di disegnare solo semplici vettori esso viene considerato il primo sistema di *Augmented Reality* della storia.

Negli anni '70 e negli anni '80 un gruppo di ricercatori studiò come utilizzare le tecnologie per *AR* presso il laboratorio Armstrong dell'*USAF*, centro di ricerca della *NASA* Ames e l'università della Carolina del nord. Il termine *Augmented Reality* viene introdotto solo negli anni '90 quando due ricercatori dei laboratori della *Boeing*, Tom Caudell e David Minzell costruirono un particolare *HMD* in grado di riprodurre la strumentazione di bordo di un aereo. Tale dispositivo permise di visualizzare agli occhi dei piloti informazioni aggiuntive realative a rotte, decolli ed atterraggi in modo assolutamente preciso e veloce.

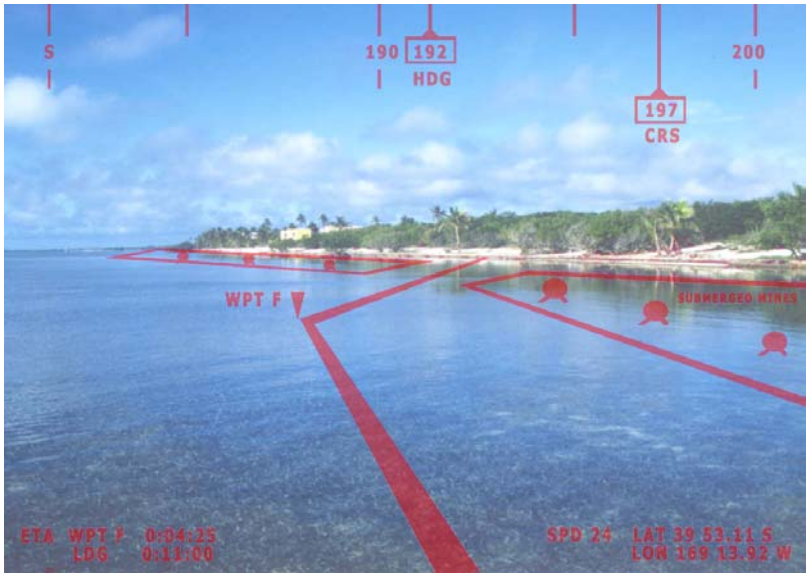


**Figura 2: Esempio di sovrimpressione di informazioni sintetiche alla realtà**

Lo scopo principale di un sistema *AR* è quello di dare la possibilità di vedere quello che normalmente, ad occhio nudo, non si può vedere, arricchire la visione dell'utente con informazioni addizionali mediante la fusione di immagini tridimensionali di sintesi prodotte al computer con immagini della realtà.

Dato che l'utente di un sistema *AR* vede sia la realtà che l'ambientazione virtuale, un sistema di questo tipo ha bisogno per funzionare correttamente di una serie di "aggiustamenti" in più rispetto a quelli richiesti da un tradizionale sistema immersivo *VR*. In quest ultimo infatti il solo requisito necessario è quello di presentare all'utente un'ambientazione di tipo virtuale il più possibilmente simile alla realtà, diversamente da un sistema *AR* in cui l'ambientazione virtuale deve essere combinata e correttamente allineata alla realtà, alla visione del mondo fisico (Splechtna, 2003).

La mappatura (*tracking*) della posizione dell'utente e della prospettiva relativa al suo punto di vista, necessaria per sovrimporre informazioni all'immagine del mondo reale è affidata a diverse tipologie di sistemi di tracking. Tra questi troviamo (vedi capitoli 1.2.3, 1.2.4, 1.2.5, 1.2.6, 1.2.7) strumenti ad ultrasuoni, meccanici, ottici

o inerziali, tutti caratterizzati da sei gradi di libertà (*6DOF*), dunque in grado di calcolare oltre che la posizione anche l'orientamento definito dai tre angoli di imbardata, picchiata e rollio (*yaw, pitch, roll* in inglese) (vedi Figura 3).

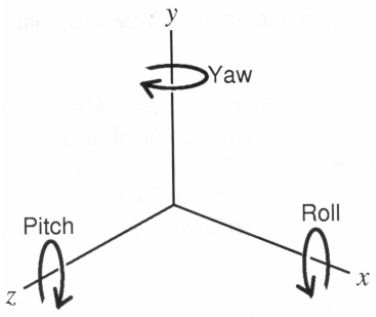


**Figura 3: Sistema a sei gradi di libertà (6DOF)**

Tutti i processi relativi al perseguimento di un perfetto allineamento tra reale e virtuale vanno sotto il nome di "calibrazione di un sistema *AR*".

### 1.1.4 I Vantaggi dell'Augmented Reality

Sebbene l'utilizzo della *VR*, ovvero di quella tecnologia che opera una totale sostituzione del mondo reale con un ambiente sintetico, rappresenti uno strumento per certi casi insostituibile, per ciò che ne concerne un suo quotidiano utilizzo, c'è da notare quanto essa, a differenza di una configurazione di tipo *AR*, presenti numerosi problemi e svantaggi. La strumentazione utilizzata da un sistema di tipo *VR* è certamente più costosa, in termini di prezzo e di necessità di hardware addizionale, l'utente che si immerge in un ambiente virtuale totalmente immersivo sprovvisto di qualsiasi tipo di dispositivo di feedback tattile, è inevitabilmente soggetto ad un forte senso di disorientamento. E' inoltre evidente quanto gli attuali sistemi di *VR* limitino la libertà d'azione dell'utente impedendogli il movimento fisico e negandogli, proprio a causa dello stato di totale immersione, qualsiasi forma di interazione sociale.

Una valida alternativa è rappresentata dall'*AR* che, in opposizione alla *VR*, offre interessanti soluzioni ai problemi appena menzionati. L'utilizzo degli speciali visori (*see-through glasses*) quali l'*HMD* permette all'utente di continuare a percepire l'ambiente reale e contemporaneamente usufruire dei dati "artificiali" elaborati e sovrimposti in tempo reale dal computer. L'*AR* offre una combinazione tra due tipi diversi di ambiente che coesistono e si integrano perfettamente, un ambiente operativo costituito dal mondo reale ed uno costituito dalle strutture ausiliarie integrative di tipo virtuale elaborate dal calcolatore. In un approccio di questo genere, i vantaggi sono evidenti, gli oggetti del mondo reale che stanno intorno all'utente e che in *VR* ne impedivano il movimento fisico, in *AR* non sono più considerati ostacoli, al contrario essi rappresentano elementi sostanziali di un ambiente "aumentato"; l'utilizzo di nuove ed efficienti mezzi di espressione garantisce la possibilità di comunicazione, in *AR* è possibile comunicare mediante l'utilizzo di un linguaggio naturale o semi-naturale. Infine, ultimo ma non per questo secondario vantaggio è rappresentato dalla possibilità di ottenere *feedback* tattile in modo facile ed economico sovrimponendo oggetti virtuali agli oggetti reali adeguatamente *trackati*. Come Marx (2002) suggerisce citando Szalavári (1997) l'AR ha un potenziale vasto assortimento di possibili applicazioni, tra le quali troviamo sistemi mobili di realtà aumentata sensibili al contesto, visualizzatori scientifici, display alternativi per la visualizzazione di dati relativi a misurazioni, sistemi di supporto alla medicina e alla chirurgia, validi supporti nel campo dell'educazione, dell'addestramento e dell'intrattenimento.

### 1.1.5 Applicazioni di Realtà Virtuale

Ormai da anni la realtà virtuale è strumento valido e supporto per varie attività umane. Come è stato brevemente accennato nel capitolo precedente, possiamo trovare applicazioni di *VR* in vari settori dell'attività umana, nel campo militare, in quello medico (ad esempio per telechirurgia), aerospaziale, robotico, automobilistico, per applicazioni legate alla fruizione del patrimonio culturale e del

suo mantenimento, ma anche nell'ambito della progettazione, dell'intrattenimento e dell'industria dei video-giochi.

Con la *VR* è possibile sperimentare fenomeni che alcune discipline confinavano nella regola, si possono simulare situazioni difficili, pericolose, costose, di dimensioni troppo piccole o troppo grandi rispetto alle nostre per essere percepite in prima persona. Simulatori di volo, sistemi per la simulazione di incendi, per operazioni in ambienti contaminati, per interazioni a livello cellulare o astronomico ci permettono di risolvere problemi, di acquisire determinate abilità e di osservare comportamenti che altrimenti non potremmo cogliere. Inoltre l'ambiente creato con la Realtà Virtuale può essere rappresentato in scala reale o nella scala che più si addice alla tipologia di simulazione richiesta ad esempio per permettere di smontare e rimontare strutture molecolari di un dato composto.

Ma la Realtà Virtuale non serve solo per la rappresentazione di oggetti che si muovono nello spazio, può venire applicata anche ad entità astratte, pensieri, elementi logici, forme grammaticali e sintattiche. Tutta una sfera di problemi che un tempo era possibile trattare solo a parole o col pensiero ora può coinvolgere anche le nostre facoltà psico-motorie, i nostri occhi, le nostre mani.

In definitiva, la realtà virtuale viene impiegata in tutti quegli ambiti umani in cui l'uomo abbia la possibilità di simulare ed, alla fine, acquisire conoscenze e capacità altrimenti pericolose o troppo onerose (Conti, 2002).

I casi più evidenti riguardano certamente le simulazioni militari, ambito in cui la *NASA* e le Forze armate statunitensi sono impegnate da anni (Bruno, 2001).

Al giorno d'oggi le forze armate di numerosi paesi, sfruttano la realtà virtuale per l'addestramento dei propri uomini praticamente in ogni settore. E' possibile simulare la navigazione navale, aerea o di un carro armato; alcuni autori addirittura sostengono che in un futuro i soldati, muniti di *HMD* potranno (Alexander, 1994) beneficiare di una serie di informazioni relative a mappe, percorsi e strategie di guerra elaborate in tempo reale e potranno essere guidati e condurre il proprio compito nel più efficace dei modi. L'esercito americano è così convinto dell'efficacia

di queste nuove tecniche di "guerra virtuale" che ha allestito un sito web dal quale è possibile scaricare gratuitamente un gioco di guerra di realtà virtuale (vedi Figura 4) (America's Army, 2003).

**Figura 4 : Il gioco di Realtà Virtuale America's Army**

Questo *video-game* servirebbe per l'addestramento e la recluta di nuovi giocatori destinati al controllo di vere apparecchiature di realtà virtuale. Una applicazione tra le più recenti, il sistema *MARS* (*Mobile Augmented Reality System*), in via di sviluppo alla *Columbia University*, prevede un sistema in grado di calcolare, attraverso l'uso di *GPS* (*Global Positioning System*), la posizione e la direzione dello sguardo di una persona con un errore di pochi centimetri (Guven, 2002). Il sistema sarebbe quindi in grado di inviare all'utente munito di micro-navigatore satellitare informazioni riguardo alle cose che sta osservando, per ora entro un itinerario predefinito. Nel campo della manutenzione un sistema *AR* potrà mettere in evidenza i passi operativi da espletare o le componenti da sostituire durante il

controllo di una apparecchiatura. Allo stesso modo un medico durante una operazione chirurgica potrà avere sovrimpresso alla realtà una visione a raggi X, una tomografia assiale una ecografia oppure delle strutture grafiche guida che lo aiuteranno durante lo svolgimento del suo lavoro senza per questo distrarre il suo sguardo della scena dell'operazione (vedi Figura 5).

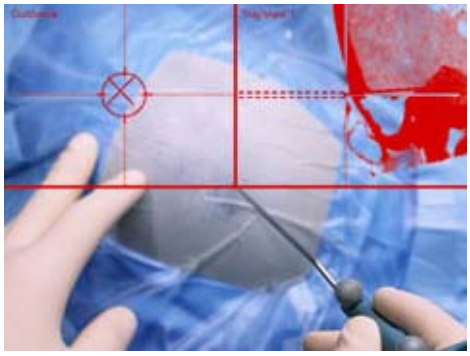

**Figura 5: Sistema AR applicato alla diagnostica medica, come ausilio durante un'operazione chirurgica**

Un pompiere, durante lo spegnimento di un incendio, potrà avere un'idea della disposizione della stanze in un edificio in fiamme e conoscere in anticipo la dislocazione delle uscite di sicurezza, previo conoscenza e disponibilità anticipata del modello 3D della struttura in questione. Un soldato sarà quindi in grado, grazie all'ausilio di speciali ricognitori, di vedere in tempo reale la posizione dei cecchini nemici. Altre applicazioni potranno essere sviluppate nell'ambito dei beni culturali, del turismo, della navigazione aerea e di molti altri settori dell'attività umana (Sonnati, 2002).

Risulta facile immaginare che il futuro della Realtà Virtuale coinciderà sempre più con lo sviluppo della *Mixed Reality* ovvero con lo sviluppo di quella tecnologia che non mira ad una sostituzione totale del mondo reale, piuttosto ad un suo arricchiranno tramite l'aggiunta di determinate informazioni digitali (Feiner, 1996).

## 1.2 Supporti Hardware

### 1.2.1 Sistemi di Output Stereovisivi

La profonda conoscenza dei meccanismi della percezione umana e dei suoi limiti, ha permesso di imitare in maniera sempre più perfetta il modo di vedere proprio dell'uomo della realtà che lo circonda.

Un proponimento fondamentale della Realtà Virtuale è quello di assicurare al soggetto una percezione realistica di un mondo sintetico quale quello prodotto da una applicazione *VR*. L'obiettivo ultimo risulta quindi quello di generare un' immagine virtuale più vicina possibile a quella reale permettendo in tal modo all'utente di lavorare in un ambiente intuitivo e confortevole. La stereovisione è quella tecnologia che assicura la percezione della tridimensionalità e della profondità dell'ambiente virtuale artificialmente costruito.

Nell'ambito delle capacità di visione proprie dell'essere umano, si definisce "stereopsi" la capacità di localizzazione relativa degli oggetti visivi in profondità. Essa subentra non appena oggetti distinti, al di fuori della cosiddetta "area di Panum", rappresentata da tutti i punti-oggetto disposti sulla linea curva posta nello spazio visivo e da tutti quei punti dell'ambiente che stimolano punti corrispondenti sulle retine oculari, vengono visti dietro o davanti al punto di fissazione, con immagini che per motivi geometrici legati alla distanza interpupillare (angolo di parallasse o disparità) sono leggermente differenti ed, attraverso il processo della fusione, ingenerano una impressione visiva singola consentendo di apprezzare la profondità relativa. Tale fenomeno può essere ottenuto anche a partire da immagini bidimensionali costruite in modo da fornire immagini disparate in senso orizzontale e tali da stimolare elementi retinici nei due occhi che corrispondono ad uno schema di riferimento per la localizzazione relativa in profondità: "stereoscopio" o "aploscopio" sono i termini con i quali vengono indicati i dispositivi che sfruttano questo meccanismo (Canziani et al., 2002).

Un sistema stereoscopico misura la distanza che lo separa dagli oggetti nello spazio osservato. Nel modello più semplice, il sistema si compone di due telecamere che giacciono sull'asse *X* del sistema di riferimento, con i piani immagine sul piano *XY*. Le telecamere sono puntate nella direzione dell'asse *Z*, in modo tale che i loro assi ottici siano tra loro paralleli e separati da una sufficiente distanza. La misurazione avviene attraverso la tecnica della triangolazione, secondo la quale è possibile determinare la posizione di un punto su un oggetto determinando il triangolo formato da questo punto con i centri ottici del sistema di osservazione, misurando la parallasse tra due immagini ottenute riprendendo l'oggetto da due posizioni diverse, mediante due telecamere separate da una sufficiente distanza.

Per avere una visione di tipo stereo è necessario dotarsi di sistemi delicati quali l'*HMD* o gli *Shutter Glasses*. L'*HMD* (vedi Figura 6) è un tipo di interfaccia costituita da uno speciale visore, spesso montato sul capo del "navigatore", che assicura la visione stereoscopica grazie all'ausilio di due schermi *LCD* o *CRT*.

**Figura 6 : Sony HMD GlassTron (Sony, 2004)**

Tali schermi forniscono agli occhi dell'utente la visione di due immagini dello stesso modello tridimensionale ripreso da due diversi punti di vista. La visualizzazione e fusione di queste due immagini elaborate dal cervello dell'osservatore permettono la percezione della profondità dell'ambiente.

Una soluzione più economica è rappresentata da un particolare tipo di occhiali detti *Shutter Glasses* (vedi Figura 7). Il loro funzionamento si basa sull'alternanza delle immagini (in inglese *Page Flipping*) visualizzate dalla periferica video. Gli schermi a cristalli liquidi degli occhiali in questione permettono di coprire alternativamente, a determinata frequenza un occhio. La corretta sincronizzazione della frequenza delle immagini visualizzate a video con la frequenza di oscuramento delle lenti degli occhiali permette all'utente di percepire la scena come tridimensionale.

Esistono tuttavia sistemi Applicazioni *VR* che non fanno uso di visori stereoscopici sebbene Conti (2002), citando Pimentel et al. (1995), noti che, una visione di tipo stereoscopico risulta essere una caratteristica molto importante in quanto fornisce una importante sensazione sulla profondità della scena durante la manipolazione di oggetti all'interno di una ambiente virtuale.

**Figura 7 : Shutter Glasses prodotti da Barco (Barco, 2004)**

Una soluzione hardware diversa ma più costosa, per la stereoscopia, è rappresentata dall'utilizzo dei proiettori. Con essi è possibile ottenere un migliore visione grazie all'utilizzo di una superficie di visualizzazione delle immagini più ampia, sebbene, in alcune configurazioni risulti impossibile qualsiasi tipo di interazione. Nei sistemi mono-proiettore e schermo tradizionale infatti, interporsi tra la sorgente di proiezione e la superficie di proiezione impedirebbe la corretta visualizzazione delle immagini. Per evitare questo problema, vengono solitamente adottati dispositivi a retroproiezione, schermi traslucidi in grado di visualizzare le immagini sulla faccia

opposta a quella su cui arriva la luce di proiezione (vedi Figura 8). Questo sistema rappresenta lo standard per la maggior parte delle applicazioni *VR* per l'industria.

**Figura 8 : Il sistema di visualizzazione "Virtual Table Baron" prodotto da Barco (Barco, 2004)**

Sistemi più complessi e più costosi, sono rappresentati da tutte quelle configurazioni che adottano l'utilizzo di più proiettori; i cosiddetti *Back Projection Wall* utilizzano un certo numero di retroproiettori associati talvolta a specchi per redirezionare l'immagine ottenendo in tal modo una vasta superficie visibile (vedi Figura 9).

Gli ambienti virtuali creati mediante l'utilizzo di un treno di proiettori/retroproiettori, comunemente denominati *Cave Automatic Virtual Environment* (*CAVE*), sono in grado di creare ambienti estremamente realistici ed immersivi, nei quali possono operare, contemporaneamente, una o più persone (vedi Figura 10).

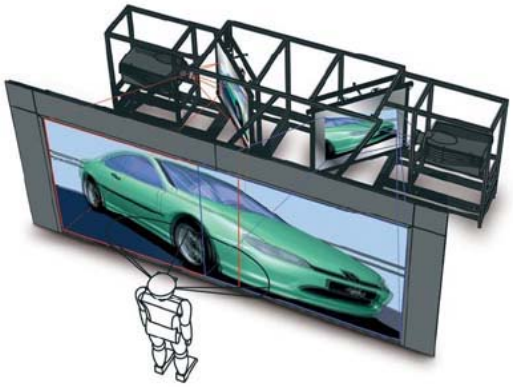

**Figura 9 : Sistema di visualizzazione "Back Projector Wall" bicanale prodotto da Barco (Barco, 2004)**

Presso l'aeroporto di Parigi in Francia, è stato installato un sistema di visualizzazione *VR* che permette una visione sintetica a 360° simile alla visione reale che si avrebbe della dalla torretta di controllo (vedi Figura 10). Il sistema si basa sull'utilizzo di un treno di nove proiettori *CRT BarcoGraphics 1209*, ognuno dei quali caratterizzato da una risoluzione pari a 1248 x 1024 pixels, ed uno schermo cilindrico del diametro di 10 m ed altezza 2.56 m.

**Figura 10 : Sistema di visualizzazione a 9 proiettori prodotto da Barco (Barco, 2004)**

### 1.2.2 Sistemi di Input Tridimensionale

La riproduzione del senso di immersione delle ambientazioni sintetiche tramite dispositivi di *output* rappresenta solo un aspetto della *VR*; per poter beneficiare appieno delle potenzialità offerte da questa tecnologia è necessario

prendere in considerazione altri importanti aspetti circa il modo in cui è possibile interagire con un ambiente virtuale. Sebbene l'abilità di controllare in modo indipendente almeno sei variabili (*6DOF*) costituisca uno dei fondamentali requisiti di un sistema di input tridimensionale, ancora più importante risulta capire in quale modo queste variabili possano venir controllate. L'utilizzo delle cosiddette "interfacce 3D" in *VR* è volto a valorizzare l'abilità che l'uomo già possiede, abilità che egli ha sviluppato vivendo e lavorando nel mondo reale. Gli approcci di interazione tramite l'utilizzo di questi sistemi devono essere quindi il più possibilmente simili a quelli utilizzati per l'interazione con gli oggetti reali appartenenti al modo reale.

Da un punto di vista tecnologico, tali interfacce 3D rappresentate dai dispositivi di input 3D, altrimenti chiamati dispositivi di tracking, convertono un fenomeno fisico, come la forza e la velocità, in un segnale elettrico misurabile mediante l'utilizzo dal computer. L'idea alla base dei moderni sistemi di tracking è quella di proporre un dispositivo in grado di riconoscere ed interpretare un sempre più ricco e standardizzato vocabolario gestuale, sonoro, mimico proprio dell'utente.

Uno dei dispositivi di input tridimensionale più diffusi ed economici, è senz'altro rappresentato dallo *SpaceMouse*, anche detto *SpaceBall*, particolare dispositivo a sei *DOF*, simile ad un mouse nella forma, ad un joystick nel funzionamento, che permette all'utente di effettuare variazioni di posizione ed orientamento della scena e degli oggetti con l'utilizzo di una sola mano (Figura 11).

**Figura 11 : SpaceMouse**

Tuttavia, il dispositivo che meglio si presta al riconoscimento della gestualità umana (*gesture recognition* in inglese) è sicuramente il guanto (vedi Figura 12). Con esso è possibile definire una ragguardevole combinazione di comandi utili allo svolgimento di numerose attività di interazione con ambienti ed oggetti virtuali.

**Figura 12 : DataGlove**

Già a partire dalla fine degli anni '80 vennero descritti da Sturman (1989) e Zimmerman (1987) taluni guanti che, una volta indossati, potevano rilevare posizione della mano e flessione delle dita. Sebbene questa periferica si sia evoluta nel corso degli anni, essa è ancora oggi oggetto di ricerca, utilizza tipicamente

tracciatori di tipo elettromagnetico a sei *DOF* e sensori per il rilevamento della flessione delle dita.

Nel corso degli anni, si sono sviluppati altri tipi di guanti, più sofisticati, in grado di simulare la forza resistente (*force feedback*) esercitata dagli oggetti virtuali sulle articolazioni della mano durante il loro contatto; essi risultano però oltre che costosi e difficili da calibrare, particolarmente macchinosi ed invasivi (vedi Figura 13).

**Figura 13 : Prototipo di force-feedback DataGlove (DataGlove, 2004)**

Recenti studi in questo campo hanno permesso di sviluppare un particolare tipo di guanto detto "*Pinch Glove*" in grado di intercettare, il contatto tra le punte delle dita, permettendo in questo modo di evitare l'utilizzo dei costosi sensori atti al rilevamento della flessione delle dita, dando all'utente la possibilità di afferrare oggetti virtuali ed un modo alternativo di utilizzo di una serie di comandi e regole euristiche. L'utilizzo di questo particolare tipo di guanto è inoltre facilitato dal fatto che la periferica non necessita, per un corretto funzionamento, di alcuna calibrazione. In questi anni, la ricerca sui sistemi di tracking si sta intensificando, a secondo del campo di applicazione, possiamo trovare dispositivi molto diversi tra loro. Vi sono quelli che permettono di rilevare i movimenti della testa, essi vengono utilizzati in sistemi in cui è necessario sapere in tempo reale la direzione dello sguardo dell'utente. Il *VisionTrack* sviluppato da *Polhemus* (Visiontrak, 2004) riesce a rilevare oltre ai movimenti della testa, la posizione e gli spostamenti degli occhi,

permettendo in questo modo una totale e completa soluzione di tracking del capo. Il dispositivo "*AVSoft FT2000 face tracker*" della *MetaMotion* (AVSoft FT2000, 2004) riesce a catturare i movimenti del viso e la mimica facciale. In campo cinematografico, la tecnica del *motion-capture*, basata sull'utilizzo di strumentazioni di *tracking* 3D, ha dato la possibilità al regista di animare, in tempo reale, personaggi immaginari e modelli virtuali catturando direttamente i movimenti del corpo di attori reali (vedi Figura 14).

**Figura 14 : Esempio di motion-capture usata nel film Lord of The Rings**

A prescindere dal tipo di dispositivo e dal suo campo di applicazione, i dispositivi di tracking 3D sono caratterizzati da quattro parametri fondamentali che ne permettono una corretta classificazione. Il "Volume di Lavoro" definisce la massima regione di spazio in cui si è sicuri che il sistema funzioni correttamente. La "Frequenza di Campionamento" indica la frequenza con cui il calcolatore rileva e aggiorna le variabili in gioco. Con "Risoluzione" si intende invece il grado di precisione raggiungibile, grazie al riconoscimento della più piccola variazione di posizione. Infine con il termine "Latenza" viene indicata la rapidità di risposta propria del dispositivo, grazie alla valutazione del tempo che intercorre tra la generazione dall'evento e il suo riconoscimento. Sulla base della tecnologia utilizzata per il rilevamento degli eventi generati dall'utente, i dispositivi di input 3D possono essere divisi in cinque gruppi principali: meccanici, elettromagnetici, ottici, acustici e inerziali.

### 1.2.3 Dispositivi di Tracking Meccanici

I sistemi meccanici a bracci utilizzano potenziometri o *encoder* ottici per misurare la rotazione dei perni di vincolo delle aste di collegamento (vedi Figura 15). Noti gli angoli di ciascun giunto e le lunghezze delle aste della catena cinematica, è possibile calcolare con facilità la posizione dell'oggetto tracciato.

**Figura 15 : Tracker meccanico Premium 1.5 prodotto della linea PHANTOM commercializzato da SensAble**

- Vantaggi: la semplicità costruttiva e la mancanza di unità di trasmissione-ricezione ne riduce il costo e la latenza. Inoltre i dispositivi di tipo meccanico sono meno sensibili a fenomeni esterni quali i campi elettromagnetici.
- Svantaggi: l'utente è piuttosto vincolato nei movimenti, spesso il volume di lavoro è molto ristretto e le parti in movimento sono soggette ad usura.

### 1.2.4 Dispositivi di Tracking Elettromagnetici

Sono costituiti da un trasmettitore e da un ricevitore. Il trasmettitore genera un campo magnetico fluttuante costituito da tre spire tra loro ortogonali che interagiscono con quelle prodotte nel ricevitore. La variazione del segnale è tradotta in una variazione di posizione e di orientazione a sei gradi di libertà (vedi Figura 16).

**Figura 16 : Tracker elettromagnetico Fastrak prodotto da Polhemus**

- Vantaggi: sono dispositivi molto comodi, le ridotte dimensioni dei ricevitori (meno di 2 cm) permettono all'utente di poterli agganciare all'*HMD* sul capo oppure alle mani o alla penna. Il volume di lavoro non è molto ampio (solitamente un metro cubo per la massima accuratezza), ma combinando fra loro più dispositivi di questo tipo è possibile estendere tale valore.
- Svantaggi: questo tipo di dispositivo è molto sensibile alle interferenze elettromagnetiche (*EMI*) di apparecchi elettronici come radio o monitor, così come a qualsiasi tipo di conduttore elettrico o magnetico. Qualsiasi tipo di metallo, posto nelle vicinanze del trasmettitore o del ricevitore, è in grado di distorgere il segnale del trasmettitore, compromettendo la corretta misurazione delle posizioni e delle orientazioni *trackate*. A tal proposito Kindratenko (1999) propone una possibile soluzione e spiega che, assumendo che la posizione del trasmettitore sia fissa e che i materiali di tipo metallico nelle vicinanze non si muovano, l'errore statico è funzione della posizione del ricevitore e può essere corretto fino a quando il campo magnetico non si ripiega su sé stesso (Kindratenko, 1999). Altri svantaggi sono determinati dal tempo di latenza nella risposta, pari a circa 100ms. Nei dispositivi più recenti, tuttavia, tale valore è stato ridotto.

### 1.2.5 Dispositivi di Tracking Ottici

I sistemi di tracciamento ottici sono caratterizzati da un'elevata accuratezza, ma anche da notevole complessità ed un elevato costo. Il sistema è costituito da particolari dispositivi in grado di emettere luce infrarossa e speciali telecamere ad infrarossi in grado di rilevarla. La penna, il guanto, la testa, o qualsiasi altra parte che si vuole *trackare*, viene dotata di particolari dispositivi sferici di materiale retroriflettente in grado di riflettere la luce infrarossa prodotta dagli emettitori (vedi Figura 17). La particolare disposizione delle sfere propria di ogni parte da *trackare* ne permette il riconoscimento della configurazione ed un rilevamento di una sua traslazione o rotazione nello spazio 3D.

**Figura 17 : Dispositivo di tracking ottico a sfere retroriflettenti**

- Vantaggi: sono capaci di gestire un'area di lavoro molto grande (fino a 100m con il laser) e dei livelli di precisione altrimenti non ottenibili.
- Svantaggi: richiedono che sia mantenuta la continuità del cammino ottico tra proiettore e sensore, altrimenti si genererebbero valori di tracking imprevedibili. L'alto costo e l'ingombro ne hanno impedito lo sviluppo.

### 1.2.6 Dispositivi di Tracking Acustici

È una tecnica semplice e funzionale. Tipicamente il sistema prevede un emettitore e un ricevitore. L'emettitore genera un segnale sonoro che può essere percepito dal ricevitore (microfono) che ne misura il tempo di cammino (chiamato tempo di volo o *TOF*). Per poter beneficiare di un sistema a sei *DOF* è indispensabile assemblare almeno tre coppie di dispositivi emettitore/ricevitore, i dati vengono raccolti da un elaboratore in grado con essi di calcolare le corrette posizioni ed orientazioni delle parte *trackate*.

- Vantaggi: sono dispositivi economici e di facile reperibilità.
- Svantaggi: la velocità del suono varia con le condizioni ambientali, quindi tali sistemi si rivelano poco precisi se non vengono attuati particolari accorgimenti per mantenere costanti pressione, temperatura e umidità dell'aria. Oltre ad avere un volume di funzionamento limitato, questi sistemi sono inutilizzabili in ambienti dotati di pareti in grado di riflettere le onde sonore.

### 1.2.7 Dispositivi di Tracking Inerziali

Questo tipo di tecnologia, nata nei sistemi di guida di missili e aerei utilizza i giroscopi per misurare cambiamenti di rotazione attorno a uno o più assi. Il loro costo è elevatissimo, ma è destinato a scendere dato che se ne prevede un loro massiccio utilizzo per applicazioni nell'ambito della progettazione di stabilizzatori per telecamere. La tecnologia dei circuiti integrati ha ormai raggiunto un livello tale da permettere lo sviluppo di sensori inerziali così piccoli da poter essere "indossati" dall'utente (vedi Figura 18).

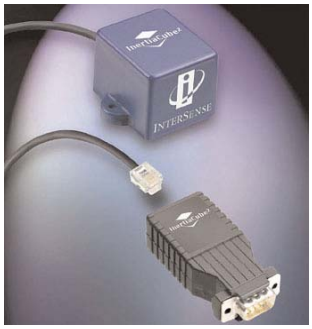


**Figura 18 : InertiaCube2, il più piccolo tracker inerziale al mondo (InertiaCube2, 2004)**

Questi dispositivi sono in grado di percepire la direzione del campo gravitazionale terrestre. Grazie alle misurazioni operate da un giroscopio a due assi in essi contenuto e assumendo che non vi sia una significativa accelerazione dell'utente, condizione ragionevole per un sistema di visualizzazione *VR*, essi riescono a calcolare l'angolo di imbardata. La misurazione di tale angolo richiede la conoscenza di un angolo iniziale ed è soggetta sia alla deriva del giroscopio che all'effetto di isteresi al quale il giroscopio stesso è soggetto dopo una larga e veloce rotazione lungo tale angolo. Ciononostante, il basso tempo di latenza di questi strumenti e la loro semplicità di utilizzo ne fanno uno strumento ideale per applicazioni di Augmented Reality.

- Vantaggi: non sono necessari sistemi di trasmissione-ricezione.
- Svantaggi: il costo, la necessità di ricalibrazione e la particolare sensibilità alla temperatura.

## 1.3 Navigazione e Multimodalità

### 1.3.1 Navigazione in Ambienti Virtuali

La sola "navigazione" entro un ambiente virtuale, ovvero la capacità di muoversi in esso, rappresenta il livello più elementare di interazione in Realtà Virtuale. Lo spazio virtuale, pur definendo una serie di vincoli e possibilità di accesso, di percorso e di spostamento, deve lasciare all'utente la capacità di interagire con lo spazio virtuale mediante semplici movimenti delle mani, del capo o di altre parti del corpo. L'utilizzo di *HMD* e guanti di realtà virtuale rende l'esperienza ancora più immersiva, simile all'esperienza del movimento in ambienti reali.

### 1.3.2 Interattività e Manipolazione

L' interazione non deve essere limitata alla semplice navigazione attraverso l'ambiente virtuale ma deve consentire di interagire con gli oggetti in esso contenuti. Gran parte delle applicazioni di grafica 3D oggi viene sviluppata su piattaforme desktop di tipo convenzionale sulle quali è solito trovare dispositivi di input-output a due dimensioni, tipicamente un mouse come dispositivo di input ed uno schermo *CRT* o *LCD* come dispositivo di output.

Svariate forme di interazione uomo-macchina prevedono l'utilizzo di periferiche comuni, come la tastiera ed il mouse, il *trackball* o la penna ottica, o metodi di interazione ancora più semplici ed intuitivi quali il "*touch screen*".

L'uso di dispositivi di interazione bidimensionali per la manipolazione di modelli 3D in ambienti tridimensionali rende detti dispositivi innaturali e poco intuitivi da utilizzare. Fortunatamente, come Fiorentino (2001) afferma, lo sviluppo di metodi e strumenti per l'interazione in ambienti di Realtà Virtuale tende a rendere tale interfaccia il più "trasparente" possibile. Infatti più alto è il grado di "trasparenza" dell'interfaccia, più l'utente può concentrare la propria abilità nel lavoro piuttosto che sui metodi con i quali realizzarlo. Szalavári (1997) suggerisce

che l'obiettivo è quello di rendere l'utilizzo del dispositivo di input così naturale da far si che l'utente possa evitare qualsiasi forma di addestramento alla periferica o alle metafore che stanno alla base del suo utilizzo; all'utente basterà la conoscenza data dall'esperienza quotidiana sul concetto generale di interazione. I dispositivi di nuova generazione per la modellazione in grafica 3D, caratterizzati da sei gradi di libertà (*DOF*, ovvero *Degree of Freedon*), bene si avvicinano a questa idea di periferica "trasparente".

### 1.3.3 Multimodalità: Sketch, Interazione Bimanule e Comandi Vocali

L'esperienza e il successo ottenuto dall'utilizzo di *PC* portatili/*tablet-PC* e penne ottiche ha posto le basi per lo sviluppo di una interfaccia tridimensionale basata sull'interazione a due mani (la mano dominante per la penna e l'altra per la tavoletta) usata in particolar modo per la manipolazione di oggetti in ambienti *VR* ed *AR*.

Szalavári (1997), citando Sachs (1991), rivela quanto, nel campo del design, l'utilizzo di alcuni sofisticati strumenti *CAD* risulti poco naturale, insufficiente al supporto della modellazione tridimensionale di oggetti, e continua proponendo come valida alternativa un sistema con sensore a sei *DOF* basato sull'utilizzo combinato di una tavoletta ed una penna chiamata *PIP* (*Personal Interaction Panel*, dell'inglese pannello personale per l'interazione). Mentre la tavoletta funge da riferimento e base di appoggio per gli oggetti, la penna permette di modellarli o di disegnare liberamente nello spazio 3D forme e curve di ogni genere. Secondo Szalavári (1996) l'utente può interagire con il *PIP* su un'insieme di elementi 2D e 3D propri dell'interfaccia stessa, basati sulla metafora del "2D nel 3D". Il fatto di poter disegnare/modellare direttamente nello spazio 3D, evitando di dover operare delle astrazioni dovute all'utilizzo di interfacce 2D in ambienti tridimensionali, è di grande vantaggio per l'utente in quanto consente una migliore comprensione delle forme e delle relazioni tra di esse esistenti.

La metodologia *Sketch* fa utilizzo di una interfaccia 2D ed è basata su un sistema di riconoscimento gestuale attraverso il quale l'utente può generare modelli *CAD* approssimati componendo una serie di gesti predefiniti e soggetti, per la loro interpretazione, ad una serie di regole euristiche.

Gli studi effettuati da Hauptmann (1989) sulle tecniche vocali e gestuali per la manipolazione di oggetti 3D hanno confermato quanto comandi vocali e gestuali siano tra loro connessi e quanto possano venir utilizzati contemporaneamente con grande successo. Nessuno dei due sistemi di interazione è completo e, singolarmente preso, sufficientemente affidabile, in modo particolare quando la sintassi del comando è complessa. Tuttavia, l'utilizzo contemporaneo di entrambi i sistemi, permette la creazione di un ricco vocabolario di comandi riducendo in tal modo il rischio di una loro cattiva interpretazione.

La lavorazione bi-manuale facilita talune operazioni complesse quali la rotazione di oggetti sul proprio asse o il loro posizionamento rispetto a determinati punti di riferimento. Fiorentino (1999) fa notare come Weimer e Ganapathy (1989) abbiano studiato l'input gestuale e vocale per un sistema di modellazione per superfici. Con le prove effettuate si è potuto constatare come l'adozione di un sistema di riconoscimento vocale abbinato ad un già collaudato sistema di riconoscimento gestuale abbia portato ad un sensibile miglioramento nell'usabilità dell'interfaccia. Nelle applicazioni usate come test la voce è utilizzata per la navigazione mentre la gestualità per definire parametri sulle forme. Gli autori hanno adottato tre motivazioni fondamentali a favore dell'introduzione combinata del sistema per il riconoscimento della voce e dei gesti:

1. la gente tende ad utilizzare i gesti per spiegare meglio quello che sta dicendo;
2. il vocabolario parlato ha una interpretazione più standardizzata di quello gestuale;
3. la gestualità e la voce si completano a vicenda.

### 1.3.4 Text to Speech

La tecnologia cosiddetta *Text-to-Speech* (*TTS*) sta diventando in questi anni sempre più una tecnologia importante e critica, in quanto le compagnie riescono ad esempio a trasmettere molte più informazioni attraverso il telefono o altre forme di comunicazione. Una nuova generazione di tecnologie, associate ad un continuo miglioramento di quelle già esistenti, hanno messo le basi per lo sviluppo di potenti applicazioni in competizione una con l'altra per l'ottenimento del più affidabile sistema di sintesi vocale in termini di qualità ed economicità.

Nel campo delle applicazioni *TTS* troviamo numerosi settori di interesse che vanno dalla resa vocale di messaggi di testo quali e-mail o fax come parte di una soluzione di *messaging* unificato, alla gestione alternativa di informazioni visive/testuali come ad esempio pagine web o i libri. In generale si può dire che il sistema *TTS* fornisce output vocale per tutti i tipi di informazione organizzata in database.

### 1.3.5 Speech Recognition

Il processo di riconoscimento vocale, anche detto *speech recognition* (*SP*), è generalmente gestito da un componente software conosciuto come "*speech recognition engine*". La funzione primaria delle *speech recognition engine* è quella di processare i dati vocali in input e tradurli in testo, in modo che l'applicazione lo possa capire. Una volta effettuata la traduzione, l'applicazione può gestire i dati in un duplice modo:

- L'applicazione può interpretare il risultato ottenuto dal riconoscimento vocale come un comando. In questo caso l'applicazione è una applicazione di comando e controllo. Un esempio di applicazione di comando e controllo può essere una applicazione nella quale l'utente dice "controlla il bilancio" e l'applicazione esegue una procedura che fornisce all'utente i dati relativi al proprio bilancio.

- Se una applicazione gestisce il testo riconosciuto come semplice testo, allora essa è considerata una *dictation application* (applicazione per la dettatura). In una *dictation application*, se l'utente dice "controlla il bilancio", l'applicazione non interpreterà il comando, ma si limiterà a scrivere a video il testo "controlla il bilancio" (Kimberlee, 2001).

## 1.4 Il Sistema Studierstube

*" Daß ich erkenne, was die Welt*

*Im Innersten zusammenhält,*

*Schau alle Wirkenskraft und Samen,*

*Und tu nicht mehr in Worten kramen. „*

Johann Wolfgang von Goethe, Faust

### 1.4.1 L'API Studierstube

Studierstube è un sistema software nato per lo sviluppo di applicazioni collaborative in Augmented Reality. La parola tedesca "*Studierstube*" letteralmente tradotta in "stanza per lo studio" sta ad indicare la stanza nella quale il famoso personaggio di Goethe, Faust, si rifugiava per acquisire illuminazione e conoscenza dei segreti della vita (Goethe, 1808).

Questo sistema software chiamato appunto *Studierstube Application Programmer's Interface* (*StbAPI*) è costituito da una raccolta di classi C++ scritte utilizzando librerie grafiche di alto livello come *OpenInventor* (*OIV*), *Coin 3D*, *DIVOS*, che si appoggiano, a loro volta, alla *API OpenGl*, libreria grafica multi-piattaforma sviluppata in origine da *Silicon Graphics*, che rappresenta l'attuale

standard industriale per lo sviluppo di applicazioni grafiche che necessitano di alte prestazioni in termini di qualità e capacità di *rendering*. Numerose classi *StbAPI* sono dedicate all'interfaccia per lo sviluppo di applicazioni multi-utente e *multi-tasking* con una serie di possibili forme di interazione 3D quali l'utilizzo del *PIP*. Studierstube vanta un valido sistema ad eventi 3D che, in collaborazione con un set di moduli software, riesce a gestire in modo conveniente l'interfacciamento con i dispositivi di output e di input 3D capaci di generare dati di tipo *6DOF*.

Tale sistema ad eventi è utilizzato da una serie di elementi propri della *API*, quali i cosiddetti "*3D widget*", e le "*3D window*". Mentre i *3D widget* rappresentano la versione 3D dei simboli grafici di interfaccia che consentono l'interazione uomo - ambiente virtuale, le *3D window* funzionano come contenitori di applicazioni. Tra tutti questi componenti, troviamo anche le classi che hanno il compito di gestire gli aspetti non prettamente grafici, tra esse troviamo quelle dedicate alla gestione degli utenti, delle finestre, delle risorse e della loro sicura condivisione.

La struttura di *Studierstube* (vedi Figura 19) si può immaginare divisa in tre parti fondamentali, *harware, kernel, application*, può quindi essere analizzata su tre diversi livelli di astrazione. A livello *hardware* troviamo tutti i dispositivi che permettono al sistema di funzionare correttamente, troviamo l'*HMD* o il sistema di visualizzazione adottato, il sistema di calcolo *hardware* per l'elaborazione grafica, i sistemi di *tracking* e le periferiche di *input* mediante le quali è possibile interagire con *Studierstube*.

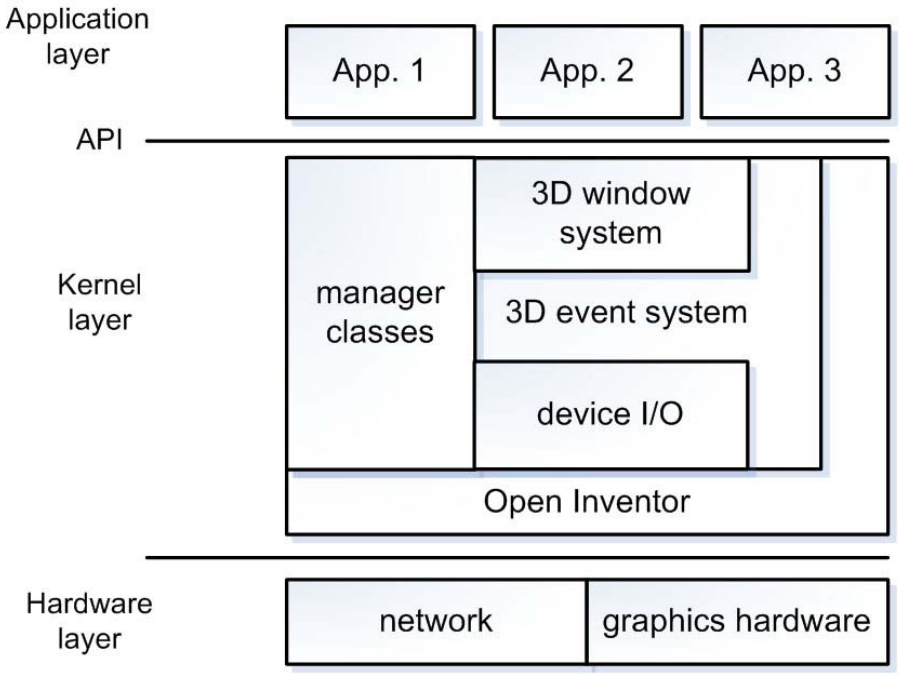


**Figura 19 : La struttura software di Studierstube**

Ad un livello di astrazione più elevato troviamo il *kernel* della *StbAPI* che comprende la gestione della collezione di classi, sviluppate utilizzando la libreria *OIV*, dedicate all'interfacciamento dei dispositivi di *input-output*, alla gestione delle *3D window* e alla distribuzione degli eventi all'interno della scena grafica. In questo livello troviamo non solo l'insieme dei moduli software dedicati all'interfacciamento dei dispositivi di *input* e *output* e quelli dedicati al *tracking*, ma anche quelli sviluppati per un efficiente utilizzo della rete, mezzo mediante il quale è possibile ottenere una collaborazione multi-utente di tipo remoto. La parte relativa all'interfacciamento con i dispositivi di input, ovvero l'insieme di quelle classi che permettono il rilevamento e la corretta interpretazione dei segnali provenienti da detti dispositivi, è supportata in modo nativo da *Studierstube* solo per determinati sistemi quali ad esempio la penna, la tavoletta, lo *spacemouse*. Una alternativa all'utilizzo delle classi per l'interfacciamento sviluppate per *Studierstube* è rappresentata dall'utilizzo del componente *middleware* chiamato "*OpenTracker*". Come vedremo più

approfonditamente nei capitoli successivi, *OpenTracker* rappresenta una soluzione *software* che, ponendosi tra il dispositivo e la struttura *Studierstube*, aiuta lo sviluppatore nel processo di interfacciamento di periferiche non supportate nativamente dal sistema di realtà virtuale, sgravandolo dal dover modificare o implementare da zero nuovi moduli, una valida alternativa ad un più laborioso approccio basato sulla re-ingegnerizzazione o sull'integrazione modulare. Per i nostri scopi infatti, *OpenTracker* ha rappresentato un flessibile e versatile supporto per un'attività di *tracking 6DOF* capace di comunicare con il dispositivo *hardware*, leggere i dati, mandarli a *Studierstube* o, in alternativa, ad una rete in *multicasting* per il loro utilizzo da parte di altri utenti ad essa collegati. L'ultimo livello è costituito dalle applicazioni, parti di codice concettualmente indipendenti dal *kernel* ma che necessitano della piattaforma *StbAPI* per poter funzionare.

I dati raccolti da un qualunque dispositivo di *input*, sono semplici informazioni riguardo lo stato di un *hardware*, quali la posizione del *tracker* determinata dai valori delle tre coordinate spaziali o lo stato di un bottone, nella fattispecie un intero pari a zero o ad uno. Il sistema di generazione di eventi 3D di *Studierstube* implementato nella classe *Stb3DEventGenerator* costituisce il supporto mediante il quale è possibile interrogare i dispositivi di *input* dell'utente ad intervalli regolari, salvarne le informazioni in particolari contenitori e generare eventi in concomitanza al loro cambiamento. *Studierstube* segue infatti una organizzazione di tipo *event-driven*, ciò significa che il sistema è coordinato in modo tale per cui gli eventi o i messaggi che attraversano l'albero dei nodi proprio della scena, possono attivare particolari strutture sensibili a quel particolare tipo di evento.

Come possiamo leggere dalle note riportate in documentazione, Studierstube, sebbene presenti numerose e promettenti prospettive per il futuro, è un progetto sperimentale, per questo incompleto ed in continua evoluzione, e probabilmente tale rimarrà.

### 1.4.2 Multiutenza e Intercollaboratività

La *API Studierstube* si presenta come sistema multi-utente collaborativo, il che significa che supporta e permette lo sviluppo di applicazioni multi-utente, nelle quali più utenti possono lavorare contemporaneamente alla stessa applicazione. E' infatti possibile presentare ad un gruppo di persone, collegate allo stesso sistema, la visione stereoscopica dell'ambiente tridimensionale condiviso relativo all'applicazione, nella quale, le azioni effettuate dal singolo utente sulla scena possano venir percepite, da tutti gli altri utenti, in modo che ogni utente abbia la capacità di seguire l'azione secondo il proprio punto di vista (vedi Figura 20). Sebbene alcuni termini e concetti richiamino e rivelino la sua origine di piattaforma a supporto dell'interfaccia *PIP*, si può pensare a *Studierstube* come ad una specie di sistema operativo virtuale in 3D capace di gestire le risorse in modo adeguato e di distribuirle a tutti gli utenti collegati.

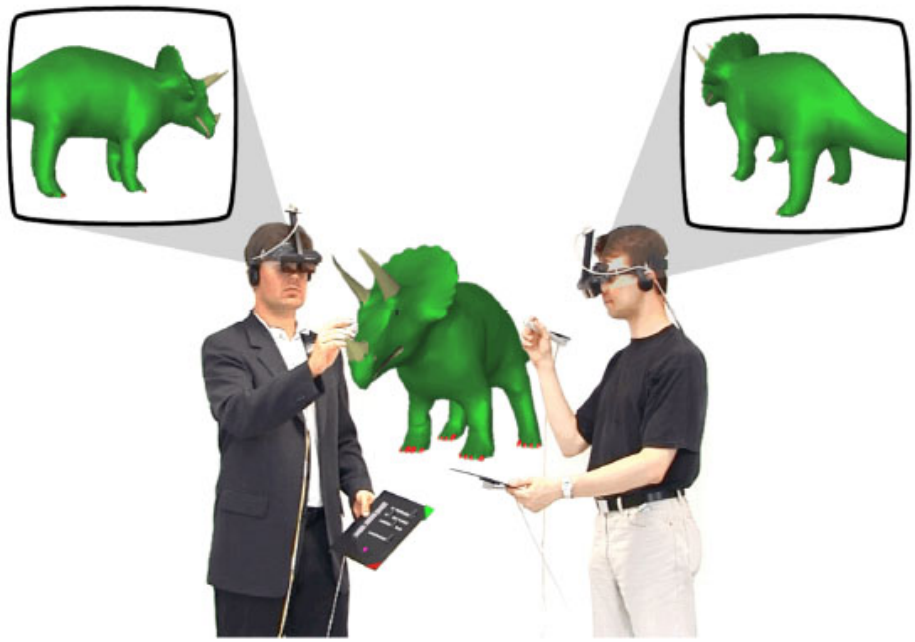

**Figura 20 : Tipico ambiente di lavoro di Studierstube**

### 1.4.3 Il Concetto di Layer

Studierstube, in modo concettualmente simile a quello usato dalla maggior parte dei programmi *CAD* per la manipolazione 3D, supporta il concetto di *layer*. L'insieme dei dati visualizzati può infatti venir separato in più parti tra loro scollegate, considerate per questo dei veri e propri *layer*. Ogni *layer* può venir visualizzato o nascosto a piacere, un efficace metodo per visualizzare aspetti individuali di una stessa realtà, per presentarne differenti rappresentazioni grafiche sulla base ad esempio di una differente educazione o livello di esperienza dell'utente osservatore. Il concetto di "realtà aumentata" non è necessariamente limitato all'arricchimento della realtà tramite strutture tridimensionali ed in *Studierstube* è possibile generare qualsiasi tipo di dato, sia esso immagine, video, o testo. Il sistema si prenderà cura del mantenimento di un perfetto allineamento degli elementi testuali sovrimpressi alle strutture fisiche, e farà in modo che il movimento del punto di vista dell'utente non porti ad una sovrapposizione o ad un oscuramento di tali elementi testuali.

### 1.4.4 Personal Interaction Panel

Il "*Personal Interaction Panel*" (*PIP*) è un dispositivo di interazione bimanuale, sviluppato dall'Istituto di Computer Grafica all'Università della Tecnologia di Vienna, che permette il controllo di qualsiasi applicazione *AR*. Esso è composto da una tavoletta, di dimensioni simili a quelle di un *block-notes* (20x25 cm), ed una penna equipaggiata con uno o due bottoni. Tavoletta e penna non nascondono in sé alcun dispositivo *hardware* "intelligente", la tavoletta è costituita da plexiglass, plastica o legno, è *trackata* mediante, ad esempio, un sistema di *tracking* magnetico e funzionante esclusivamente come supporto di riferimento per l'interazione con i modelli virtuali. Il fatto di poter *trackare* i dispositivi, ovvero poterne conoscere in ogni momento i valori corrispondenti alla loro posizione e al loro orientamento, ne permette un loro "arricchimento" sovrimponendo ad essi i corrispondenti modelli virtuali (vedi Figura 21).

**Figura 21 : Personal Interaction Panel (PIP) per l'interazione con l'ambiente MR in Studierstube**

Sebbene questo sistema sia stato concepito per funzionare in un ambiente *AR*, questa stessa configurazione può essere utilizzata anche in un ambiente totalmente immersivo quale quello proprio di un sistema *VR*.

Mentre con la penna è possibile eseguire operazioni di puntamento e manipolazione 3D, come si potrebbe fare con uno *spacemouse* a sei gradi di libertà, le funzionalità della tavoletta, sulla quale è possibile visualizzare immagini così come si potrebbero vedere su un *display* convenzionale, fanno riferimento all'esperienza quotidiana del lavorare con carta e penna, dove il foglio di carta che usiamo per scrivere rappresenta una superficie 2D immersa nell'ambiente tridimensionale caratteristico della realtà 3D. La tavoletta non solo rappresenta la metafora del piano 2D nello spazio 3D che elimina il problema della visualizzazione di poco gestibili menù volanti tridimensionali, ma permette allo stesso modo di avere a disposizione un *super-set* di elementi 3D standard per ambienti *AR* e *VR*, vincolati al piano 2D della tavoletta. Ciò significa che mediante l'utilizzo della computer grafica 3D è stato possibile creare una interfaccia tridimensionale bimanuale, basata su set di *widget* tridimensionali e innovativi elementi di controllo configurabili, capace tuttavia di mantenere elementi propri delle tradizionali "*Graphical User Interface*" (*GUI*) bidimensionali.

### 1.4.5 Tracking di Oggetti Mobili

I cosiddetti "*tracked mobile objects*" ovvero l'insieme degli oggetti fisici mobili usati all'interno di sistemi *AR* diventano parte della scena aumentata già dalle prime fasi di setup del sistema. Per poter usare oggetti reali in ambienti *AR* non è sufficiente definirne le proprietà geometriche, le dimensioni e le posizioni, è necessario rilevarne, in aggiunta alle proprietà statiche, anche i cambiamenti di posizione, orientamento e stato, di modo che essi possano integrarsi perfettamente con le altre parti dell'ambiente *AR*. Gli oggetti *trackati*, pur essendo generalmente in numero limitato, possono diventare parti funzionali del sistema, il loro ruolo nella applicazione è spesso decisivo, possono essere mossi, tenuti in mano dagli utenti, passati da un utente all'altro. Tipicamente vengono utilizzati oggetti mobili come manipolatori quali il *PIP*, ma teoricamente, qualsiasi oggetto fisico può venir trackato, qualsiasi oggetto che necessiti di informazioni supplementari fisicamente non disponibili. Tecnicamente, la posizione di questi oggetti è determinata da un sensore di tracking dedicato oppure, in altri casi, mediante l'applicazione, sul modello fisico, di specifici *marker* riconoscibili dalle videocamere ad infrarossi di un sistema di *tracking* ottico. Più semplicemente la posizione degli oggetti può essere determinata da normali videocamere il cui contenuto video viene elaborato da uno speciale strumento software chiamato "*ARToolkit*", integrato con *Studierstube*. *ARToolkit* esegue del *pattern recognition*, riconoscimento di figure e simboli appartenenti al mondo reale, mentre la loro rappresentazione virtuale, effettuata da *Studierstube*, gestisce il problema della possibile occlusione tra gli oggetti fisici e quelli virtuali.

### 1.4.6 Multi-tasking

Originariamente nato per rendere possibile l'utilizzo di applicazioni secondo il metodo di interazione definito dalla configurazione *PIP*, *Studierstube* è stato successivamente esteso e rielaborato, fino a diventare un completo e funzionale

sistema di Realtà Virtuale. In maniera del tutto simile a quella di un sistema operativo *multi-tasking* nel quale sono definiti strumenti e metodi per agire sui processi attivi, così anche nell'ambiente di *AR Studierstube* è possibile avere più applicazioni contemporaneamente in esecuzione, e passare da una di esse ad un'altra.

Il componente di *Studierstube* che si occupa della creazione di un ambiente nel quale possono convivere più applicazioni. *Studierstube* diviene così molto simile ad un sistema operativo, ma contrariamente ad una interfaccia grafica convenzionale, il *Workspace* permette a più utenti di condividere contemporaneamente lo stesso spazio. Ogni utente infatti è munito di una propria tavoletta e penna e può interagire liberamente con qualsiasi applicazione attualmente in esecuzione. Questo fatto, determina la necessità che le applicazioni sviluppate siano multi-utente in modo da poter essere gestite simultaneamente senza problemi da più utenti. Il *Workspace* fornisce una serie di risorse e meccanismi utili all'implementazione di applicazioni multi-utente, mentre la gestione delle interazioni generate dai vari utenti, è affidata allo sviluppatore dell'applicazione.

### 1.4.7 Contexts

I cosiddetti *context* costituiscono la pare del sistema *Studierstube* dedicata all'organizzazione delle informazioni ed includono dati su se stessi, la rappresentazione dei dati e informazioni sull'applicazione che opera su di essi. La struttura di queste informazioni può essere paragonata alla struttura *object-oriented* propria dei documenti tipici di un sistema desktop convenzionale (vedi Figura 22).

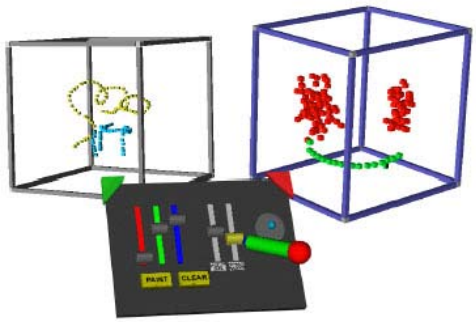


**Figura 22 : Interfaccia a documento multiplo in ambiente 3D. Il fuoco è sulla finestra 3D di destra – indicato dal contorno blu – e può essere manipolata usando gli elementi presenti sul PIP.**

I *context* rappresentano l'unico mezzo di interazione a disposizione dell'utente, al quale la nozione di applicazione è volutamente celata; egli non dovrà mai eseguire applicazioni, in quanto, in termini concettuali le applicazioni sono sempre attive (Kato et al., 2000).

In un sistema *desktop* tradizionale, la rappresentazione di un documento avviene tipicamente mediante l'utilizzo di una singola finestra 2D. Analogamente, nella interfaccia tridimensionale di Studierstube, un *context* è definito come lo spazio tridimensionale contenuto in un volume cubico – una finestra 3D (vedi Figura 22). È interessante notare come, a differenza dell'ambiente tipico di una interfaccia 2D, in questo spazio tridimensionale sia possibile condividere lo stesso *context* tra più utenti, e che ogni *context* rappresenti una istanza di un particolare tipo di applicazione. Possono coesistere istanze di diversi tipi di *context* allo stesso tempo, questo aspetto viene definito *multitasking* di applicazioni multiple. *Studierstube* permette inoltre l'implementazione delle *Multi-Document-Interface* (*MDI*), *context* multiple dello stesso documento capaci di condividere tra loro caratteristiche e dati.

### 1.4.8 Locales

I cosiddetti *locale* sono definiti da un sistema di coordinate proprio di una singola ambientazione virtuale. Solitamente coincidono con una locazione fisica, quale ad esempio un laboratorio, un'aula conferenze o parte di una stanza, ma possono essere relativi anche alla posizione dell'utente o usati arbitrariamente sovrapponendo più *locale* nella stessa porzione di spazio fisico. Convenzionalmente, ogni *display* usato in una applicazione *Studierstube* mostra i contenuti relativi ad un solo *locale*, ma uno stesso *locale* può venir visualizzato su molti display. Ogni *context* può essere replicato in ogni *locale* ma può comparire al massimo una sola volta in ogni *locale*. Tutte le repliche prodotte di un particolare *context*, sono gestite e sincronizzate dal meccanismo di distribuzione della *scene-graph* proprio di *Studierstube*.

### 1.4.9 Context contro Locales

A prima vista, può non sembrare così ovvio il perché sia necessaria la separazione tra il concetto di *locale* e *context*, il sistema *EMMIE* (Butz et al., 1999) ad esempio coinvolge utenti e computer all'interno di una singola struttura chiamata "*Ether*" popolata da elementi grafici e costruita in moto tale per cui il *locale* di una applicazione definisca anche il suo *context* e viceversa, mentre tutti i display condividono lo stesso *locale*. Sebbene questo tipo di approccio sia semplice da capire e facile da implementare, la struttura di interazione non si dimensiona in maniera adeguata rispetto al numero di elementi grafici coinvolti. Taluni elementi grafici possono venir occlusi o risultare non raggiungibili. Anche un tipo di configurazione, composta da *display* ed altre apparecchiature, non completamente vincolate può risultare poco conveniente se l'ambientazione è strutturata in un modo per cui gli utenti vengano obbligati a spostarsi fisicamente per poter aver accesso a determinati dati. Più il gruppo è grande, e più facilmente si incorrerà nella situazione per cui due utenti tra loro distanti tenteranno di raggiungere simultaneamente lo stesso dato,

rendendo per questo difficile o impossibile decidere il luogo ideale dove posizionare il dato in questione. Intelligentemente *Studierstube* separa i *context* dai *locale* per incrementare la flessibilità del sistema. Ogni *display* utilizza un *locale* separato, il che vuol dire che utilizza una scena con un sistema di coordinate indipendente. Per posiziona un *context* in un *locale* basta infatti assegnare alla finestra tridimensionale del *context* una particolare posizione entro il *locale*. Questo approccio permette la formulazione di numerose strategie per la disposizione dei *context* all'interno dei relativi *locale*.

La strategia che rende i *context* disponibili esclusivamente in un solo *locale* è equivalente all'approccio che prevede l'utilizzo del *locale* singolo, con la differenza che il *locale* viene diviso in parti. In questo caso, gli utenti non possono avere accesso ai *context* desiderati (vedi Figura 23, immagine in alto). Differentemente, la strategia secondo la quale ogni *context* è replicato in ogni *locale* garantisce l'accesso al *context* ma porta ad un ammassamento degli elementi sul *display* (Figura 23, immagine centrale).

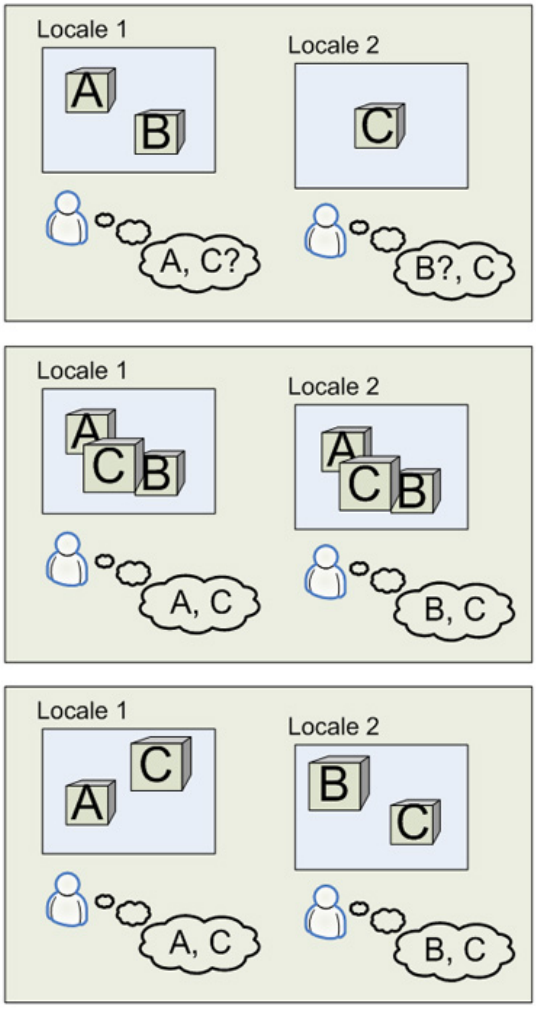


**Figura 23 : La divisione degli elementi in più locale può non soddisfare tutte le esigenze (sopra). La replica completa di tutti gli elementi porta all'eccessivo riempimento del display (centro). La replica su richiesta degli elementi permette una personalizzazione conveniente dei locale (sotto).**

Per ultima, la soluzione che prevede che la replica di un *context* in un dato *locale* sia una operazione opzionale: (ci possono essere al massimo una replica di un dato *context* in un dato *locale)*, permette agli utenti di visualizzare a video un set personalizzato di *context* conveniente e funzionale (vedi Figura 23, immagine in basso). Se i *display* sono connessi ad *host* differenti all'interno di un sistema distribuito, solo gli utenti che replicano il *context* hanno bisogno di sincronizzare i dati del *context* stesso. Assumendo che l'ambiente di lavoro non ecceda in

dimensioni, gli elementi potranno in questo modo posizionarsi in maniera adeguata all'interno dello schermo.

In molte situazioni è tuttavia desiderabile condividere la posizione e la configurazione dei bordi del *display*. *Studierstube* permette quindi la condivisione dei *locale* su più *display*, precisamente l'immagine relativa ad una stessa *scene-graph* può venir mostrata, mediante l'utilizzo di più *display*, secondo diversi punti di vista.

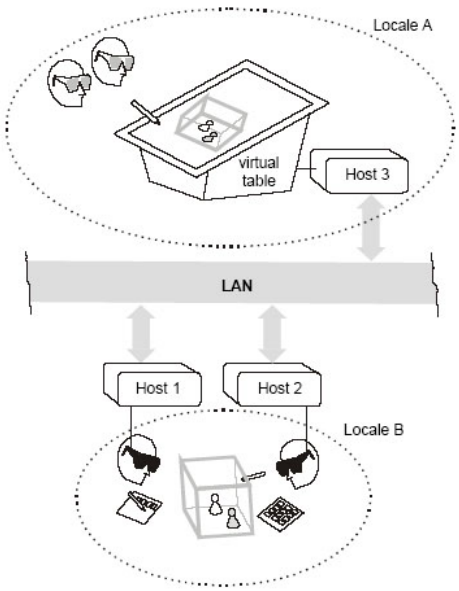


**Figura 24 : Locali multipli possono coesistere in Studierstube. Possono essere utilizzati per configurare differenti dispositivi di output e per supportare la collaborazione remota.**

Questi approcci rappresentano diverse soluzioni per la configurazione di applicazioni AR collaborative, in particolar modo per la configurazione di sistemi complessi nei quali è previsto l'utilizzo di ampi schermi. In Figura 24 viene mostrato una esempio di configurazione complessa nella quale coesistono un *context*, due *locale*, tre *display* e quattro utenti (Shmalstieg, 2001).

### 1.4.10 Widget 3D

La principale forma di interazione disponibile in ambiente *Studierstube* permette all'utente di operare su strutture virtuali tridimensionali, proprie della tavoletta *PIP*, sui cosiddetti "*widget*". Essi consentono l'interazione tra l'uomo e il computer, comprendono bottoni, *dial* e *slider*, un set standard di elementi 3D per la costruzione di una interfaccia utente personalizzata. Alcune volte, l'applicazione richiede lo sviluppo di particolari *widget* che superano il concetto generale di *widget* 2D, cosicché, se da una parte il tipo di interfaccia gestito dal *PIP* garantisce un possibile *upgrade* dei *widget* dato dalla possibilità di sfruttamento di una dimensione addizionale, dall'altra parte, c'è da notare che non è necessario pensare al *PIP* come all'unico metodo di interazione dell'utente con la scena in quanto altri oggetti indipendenti dal *PIP* possono essere usati per una manipolazione diretta. In questo senso, la *API Studierstube* aiuta lo sviluppatore della applicazione nel *tracking* e nell'utilizzo di generici oggetti di interazione 3D.

### 1.4.11 Conclusioni

Questo capitolo ha illustrato l'analisi delle differenze e dei vantaggi introdotti dall'utilizzo delle tecnologie *MR*, discutendo il modo in cui esse sono in rapporto con la computer grafica. Sono stati evidenziate le possibilità e le modalità di interazione offerte dai dispositivi *VR*, analizzate talune problematiche relative all'interfaccia uomo-macchina e presentati alcuni aspetti sulla multimodalità ponendo particolare attenzione alla funzionalità e alle modalità di navigazione in ambienti di realtà virtuale. E' stato infine descritto il sistema *VR Studierstube*, ponendo attenzione ai metodi di interazione da esso previsti.

Nel prossimo capitolo si vedrà come tali funzioni sono state estese sino ad implementare un sistema di interazione tramite il guanto di realtà virtuale P5. Si passerà quindi ad una dettagliata analisi dei metodi utilizzati per l'estrazione dei dati dalla periferica di *input*, per la gestione e la formattazione di tali dati, per il loro

passaggio a *Studierstube*, utilizzando quando necessario grafici esplicativi. Verranno discusse le strutture relative alla geometria del modello virtuale del guanto, e i metodi di implementazione del sistema di riconoscimento vocale *Speech SDK 5.1 for Windows® applications* (SAPI, 2004).

# 2 Aspetti sull'Implementazione

## 2.1 Introduzione

L'obiettivo del sistema descritto nelle pagine seguenti è quello di estendere le funzionalità dell'*API Studierstube* mediante la creazione di un sistema di interazione multimodale garantito dalla corretta implementazione di interfacciamento con il guanto di realtà virtuale P5.

Numerosi studi ed esperimenti riportati in letteratura (Bolt et al., 1992) hanno confermato la superiorità dell'utilizzo della mano, in particolare di entrambe le mani, nel descrivere determinate azioni gestuali se confrontata a metodi di interazione indiretti che prevedono l'utilizzo di strumenti interattivi intermedi (ad esempio la penna virtuale in *Studierstube* vedi Figura 21). La mano infatti offre una maggior possibilità di espressione gestuale ed anche un più elevato livello di interazione, dal momento che, indossato il guanto, esso permette all'utente di interagire direttamente con il proprio corpo nello spazio tridimensionale. L'utilizzo della penna invece, presuppone un tipo di interazione caratterizzata da un maggiore livello di astrazione mentale in quanto il processo di *gesture recognition* (riconoscimento del gesto) avviene tramite penna in maniera indiretta.

Il dispositivo scelto, il *P5 Glove* prodotto da *Essential Reality* (Essential Reality, 2004) non supporta *force feedback* e presenta per questo limitate

caratteristiche se confrontato con altri dispositivi più costosi. Esso è inoltre caratterizzato da un relativamente modesto livello di precisione nella rilevazione dei dati ed infine è parzialmente limitante delle funzionalità motorie dell'utente, dato che egli deve ricordarsi che per non avere perdite nel segnale di input non deve uscire dal campo d'azione delimitato dal ricevitore a raggi infrarossi. Tuttavia questa periferica, dalle modeste funzionalità, rappresenta un ottimo compromesso tra qualità di interazione e costo di acquisto, peraltro inferiore ai cento euro, che ne fanno lo strumento ideale per la manipolazione di oggetti in applicazioni di *Augmented Reality* per sistemi a basso costo, dove l'obiettivo si concentra non nella ricerca di uno strumento dall'elevata precisione, ma nella ricerca di una forma di interazione trasparente, immediata e facile da imparare.

L'implementazione di una struttura software capace di garantire una valida interfaccia tra la periferica P5 e l'*API Studierstube* rappresenta il punto di partenza, per lo sviluppo di un set di comandi gestuali e vocali utili a semplificare l'interazione ed in particolare la modellazione di oggetti 3D e l'interazione con applicazioni in *Augmented Reality*. Nei paragrafi seguenti tale implementazione sarà discussa in dettaglio con una particolare attenzione alla struttura dati ed agli altri aspetti di ingegnerizzazione affrontati.

## 2.2 Estrazione dei Dati dal Dispositivo

La periferica a *6DOF* (*Degree of Freedom*) P5 sfrutta la tecnologia di *tracking* ottico ad infrarossi per il rilevamento della posizione e dell'orientamento nello spazio. Il dispositivo si compone di un ricevitore a raggi infrarossi e di un guanto, ad esso collegato tramite un cavo, provvisto di una serie di *marker* (identificatori di posizione) posti sugli angoli della struttura che costituisce il polso del guanto, e di cinque sensori posti sulle dita mediante i quali è possibile rilevare l'angolo di flessione delle articolazioni (vedi Figura 25).

**Figura 25 : Dispositivo di interazione basato su tracking ottico P5 sviluppato da Essential Reality**

Un pulsante di accensione e altri tre pulsanti ausiliari completano la configurazione di questo economico ma interessante dispositivo di *tracking*. I *driver* ed un *SDK* (*Software Development Kit*), forniti dalla casa produttrice, permettono l'estrazione delle informazioni relative a posizione, orientamento e flessione delle dita nonché le informazioni relative allo stato dei pulsanti.

Di seguito sono esposti i passi operativi seguiti per l'implementazione dell'applicazione sviluppata. Il punto di partenza per l'interfacciamento della periferica P5 con il sistema *Studierstube* è stato l'identificazione della struttura dati in uscita dal *driver* fornito in dotazione. Per fare ciò l'applicazione sviluppata ha utilizzato le funzioni *P5Motion* e *P5Bend*, disponibili con l'*SDK*, necessarie per estrarre le informazioni relative alla posizione (*X, Y, Z*), l'orientamento (*Yaw, Pitch, Roll*), la flessione delle cinque dita (definite tramite le variabili: *P5_THUMB, P5INDEX, P5_MIDDLE, P5_RING, P5_PINKY*), l'inizializzazione del dispositivo e per il filtraggio dei dati in uscita. Dopo aver incluso al codice sorgente *P5Motion.h* e *P5Motion.cpp* si è proceduto ad inizializzare l'oggetto *P5Motion*, che gestisce le variabili relative allo spostamento nello spazio e le rotazioni del guanto,

controllando che l'istanza P5 fosse un valido puntatore alla libreria *CPDLL* fornita con l'*SDK*.

L'ottimizzazione dei valori relativi all'*SDK* ha permesso la calibrazione dei sensori presenti sul dispositivo necessari al rilevamento della flessione delle dita, ed di stabilire la soglia di flessione mediante la modifica della variabile *BEND_THRESHOLD*. Tale variabile è stata impostata in modo da impedire accidentali attivazioni del sistema per piccoli movimenti delle dita, specificandone un valore, pari a 30, corrispondente a circa metà flessione del dito (il piegamento delle dita effettuato dall'utente genera valori compresi tra 0 e 63).

Fatto ciò si è proceduto all'implementazione, all'interno dell'applicazione, del *loop* sulla funzione *Process()* dell'oggetto *P5Motion* per i dati processati da dispositivo P5 (estratte ciclicamente). In questo modo è stato possibile catturare in maniera continuativa i dati provenienti dal P5. Per massimizzare le risorse del sistema e per evitare fenomeni di  tale processo è stato inizialmente implementato all'interno di un *thread* indipendente che inseriva i dati ricevuti in un *buffer*. Come sarà illustrato nelle pagine seguenti, tale soluzione tuttavia è stata poi eliminata tramite l'adozione della libreria *OpenTracker*, la quale gestisce in maniera autonoma il processi di estrazione dei dati.

Infine si è proceduto alla taratura del meccanismo di filtraggio dei dati in entrata per mezzo della variabile *P5MOTION_XYZFILTERSIZE,* che ha permesso di impostare le dimensioni del filtro per gli assi *X, Y, Z,* e la variabile *P5MOTION_YPRFILTERSIZE,* che ha permesso l'impostazione delle dimensioni del filtro per le variabili *Yaw, Pitch, Roll.* Il meccanismo di filtro è stato necessario per evitare fenomeni di tremolio nella visualizzazione virtuale del guanto dovuti ad involontari spostamenti della mano dell'utente ma soprattutto per compensare l'instabilità nel rilevamento dei dati forniti dal dispositivo di *tracking* ottico.

Tale fenomeno è causato dal fatto che il dispositivo P5 è basato su una tecnologia di *tracking* ad infrarosso a basso costo che non riesce a fornire la precisione offerta dai molto più costosi dispositivi magnetici o inerziali o da ben più sofisticati dispositivi ottici. Comunque, visto che lo scopo del lavoro di tesi non era quello di ottenere un'elevata precisione nella modellazione di oggetti, la periferica P5 ha rappresentato un ottimo compromesso tra costo e qualità delle funzionalità offerte. Configurando adeguatamente i valori del filtro, in modo tale da eliminare i dati provenienti dalla lettura di piccoli spostamenti, è stato possibile ottenere una visualizzazione sufficientemente stabile e fluida del modello virtuale collegato al dispositivo.

Una volta inizializzata la periferica ed impostati i parametri necessari, si è proceduto all'estrazione dei dati dalla periferica mediante l'utilizzo delle variabili *fFilterX, fFilterY, fFilterZ* di tipo *float*. Queste variabili vengono estratte ad ogni nuovo *frame*, e contengono i valori filtrati processati dalla periferica relativi alle posizioni sugli assi *X, Y, Z*. Analogamente le variabili *fAbsYawPos, fAbsPitchPos, fAbsRollPos* di tipo *float* sono state utilizzate per determinare i valori assoluti delle rotazioni *yaw, pitch, roll*, filtrati e processati dalla periferica. Tali valori (assoluti) forniscono l'entità dello spostamento rispetto ad un riferimento costante posto all'origine. Analogamente sono stati anche ricavati i valori di *fRelYawPos, fRelPitchPos, fRelRollPos*, contenenti i valori relativi delle rotazioni *yaw, pitch, roll*, relativamente al *frame* precedente (calcolando l'incremento di rotazione attorno ai tre assi *X, Y, Z)*, processati dalla periferica ed opportunamente filtrati.

L'analisi sul formato delle informazioni in uscita dal dispositivo, ottenuta grazie all'utilizzo dei metodi messi a disposizione dall'*SDK*, è stata un passo fondamentale per la formulazione dei passaggi necessari ad una corretta conversione dei dati per l'interfaccia software con *Studierstube*. Nel paragrafo successivo vedremo come è stato possibile effettuare il *wiring* dei dati tra la periferica e l'ambiente *Studierstube*.

## 2.3 Gestione dei Dati provenienti dal Guanto P5

Il sistema sviluppato ha seguito un duplice approccio per cui, una versione del sistema è stato implementato in modo che i dati venissero inviati direttamente a *Studierstube* mentre l'altra versione è stata sviluppata in modo che tali dati venissero raccolti ed elaborati da *OpenTracker*. In particolare lo sviluppo dell'approccio basato su *OpenTracker* è stato di rilevante interesse per l'architettura generale del sistema, in quanto ha permesso la netta separazione tra le fasi di estrazione dei dati e quelle di utilizzo degli stessi. Tale utilizzo ha comportato un'estensione delle funzionalità standard di *OpenTracker* per permettere il completo trasferimento dei dati provenienti dal guanto verso l'applicativo in ambiente *Studierstube*. Per questa ragione nei paragrafi seguenti verranno esposti alcuni dei dettagli più significativi del sistema che fa uso della libreria *OpenTracker* e di come tale libreria è stata estesa per fare fronte alle necessità del P5.

### 2.3.1 Descrizione della Gestione dei Dati attraverso l'Estensione della Libreria OpenTracker

Qualsiasi sistema di *VR* o di *AR*, ha necessariamente bisogno di una struttura software dedicata alla gestione del *tracking* dei dispositivi di *input*. In passato numerose ricerche hanno cercato di migliorare la qualità dei sistemi di *tracking*, in particolare cercando di sviluppare strumenti sempre più performanti ed affidabili. Tuttavia poca attenzione è stata dedicata allo sviluppo di librerie che permettessero l'astrazione delle specifiche caratteristiche di ciascun dispositivo per mezzo di una generalizzazione degli aspetti comuni alla base dei sistemi di *tracking*.

*OpenTracker*, una libreria *open-source* altamente modulare, configurabile mediante un sistema di *scripting* basato su *XML*, tecnologia dalla quale il sistema ne trae il massimo beneficio ha cercato di ovviare a tale cronica mancanza. Questa libreria, che funziona da *middelware* tra il sistema di *VR*/*AR* ed il dispositivo,

rappresenta uno dei primi tentativi di implementazione del *software* secondo il paradigma "*write once, track anywhere*". Tale libreria non è stata concepita come una sistematica collezione di moduli dedicati per l'interfacciamento alla molteplicità dei dispositivi disponibili, infatti essa ha implementato un'unica struttura dati capace di funzionare come substrato universale per la moltitudine di interfacce *hardware* presenti per l'*input* in sistemi di realtà virtuale. L'estensione e la successiva adozione di questa libreria all'interno del sistema sviluppato in questa tesi ha introdotto i seguenti vantaggi:

- Ha permesso all'applicazione di funzionare tramite un flusso di dati generici attraverso la rete. Tali dati sono serviti per descrivere le complesse manipolazioni di oggetti tramite una serie di semplici trasformazioni.
- L'applicazione in questo modo è stata sviluppata secondo un approccio *object-oriented*, e quindi presenta tutti i vantaggi di un software di questo tipo.
- Ha permesso l'interfacciamento con il codice in linguaggio C++.
- Ha permesso in modo nativo un accesso trasparente alla rete.
- Ha permesso di stabilire nuovi tipi di interazioni mediante la configurazione o l'estensione di file *XML*, metodo semplice e indipendente dalla piattaforma di rappresentazione dei contenuti.
- Ha permesso lo sviluppo secondo un flessibile sistema ad eventi, in grado di percorrere l'albero della scena, e la configurazione di funzioni processate in maniera automaticamente a scadenza regolare di tempo.

Tradizionalmente lo sviluppo di architetture software si fonda sul concetto di programmazione a moduli, appena prodotta una nuova interfaccia, viene sviluppato il codice che permette all'interfaccia di funzionare con il sistema *VR* o *AR*.

Nello sviluppo dell'applicazione con *Studierstube* è stata seguita una interessante alternativa all'approccio modulare, costituita dall'utilizzo di un

particolare *middleware,* ovvero un sistema in grado di far colloquiare due applicazioni separate, chiamato *OpenTracker*. Esso si basa su una architettura sviluppata per essere una flessibile soluzione per l'implementazione di processi legati al *tracking* dei dispositivi di *input* ed al flusso di dati utilizzati da ambientazioni virtuali.

Le nuove classi di *OpenTracker*, implementate per i nostri scopi, leggono i dati provenienti dal dispositivo, effettuano le necessarie trasformazioni, e inviano i dati così modificati verso una interfaccia standard di *Studierstube*. La Figura 26 illustra il principio visualizzando la relazione esistente tra i due sistemi.

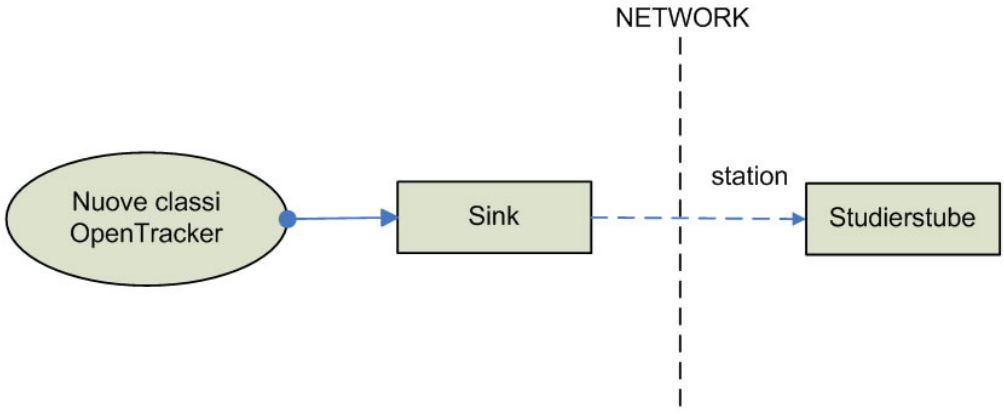


**Figura 26: Schema della connessione tra OpenTracker e Studierstube**

L'utilizzo di *OpenTracker* ha permesso di disporre di classi all'interno dell'applicazione di contenitori dei dati in input provenienti dal dispositivo di *tracking*. Questi dati vengono quindi raccolti e passati ad altre classi chiamate "*StbSink*" che hanno il compito di spedire i dati verso *Studierstube* passando per la rete. Non appena la nuova classe di *OpenTracker* è stata definita, tutte le applicazioni in ambiente *Studierstube* possono utilizzare i dati provenienti dal P5 in maniera del tutto trasparente.

Dopo aver introdotto rapidamente limiti e vantaggi nell'utilizzo di questo sistema, e dopo aver brevemente descritto l'architettura generale di interfacciamento

tra i due sistemi, nelle pagine successive si descriveranno in dettaglio il funzionamento dell'implementazione del sistema sviluppato per la periferica P5.

### 2.3.2 Dettagli Implementativi sull'Estensione della Libreria OpenTracker per la Periferica P5

L'estensione della libreria *OpenTracker* è avvenuta espandendo l'architettura ad albero a nodi per la gestione del flusso dei dati provenienti dai vari dispositivi di tracking utilizzati nel sistema *VR*. Questa struttura viene generata a *run-time* attraverso la configurazione di uno o più file *XML* che definisce i dettagli della configurazione *hardware* da utilizzare per la sessione corrente. Nel nostro caso, il file creato per la gestione dei dati del P5 è stato chiamato *P5.xml* per poterlo distinguere dagli altri file *XML* relativi alla gestione di altri dispositivi. Il sistema carica il file XML, costruisce la struttura ad albero corrispondente (che nel nostro caso contiene i nodi sviluppati *ad-hoc*) e, non appena avviata, comincia a trasmettere i dati a *Studierstube* secondo un protocollo unico e dunque in maniera del tutto indipendente dal dispositivo adottato (nel nostro caso il P5).

L'estensione di *OpenTracker* per il dispositivo P5 è dunque avvenuta costruendo innanzitutto un albero dei nodi e una serie di *modules* ad-hoc. I nuovi nodi, che descrivono la nuova sorgente dati, le trasformazioni ed i cosiddetti *sink* dei dati *trackati*, sono descritti da degli oggetti chiamati *modules,* classi nelle quali è possibile definire speciali funzioni quali ad esempio la configurazione del *driver* del dispositivo P5, calcoli geometrici e filtraggi su generici dati nonché implementazione di codice per il supporto della trasmissione dei dati sulla *network*. Il *sink* che si trova in una posizione vicina alla radice dell'albero viene incaricato di raccogliere gli eventi dal P5 ed inviarli alla struttura dati di *OpenTracker* verso l'esterno, ad esempio verso la rete.

### 2.3.3 La Configurazione del Sistema attraverso lo Scripting XML

*XML* (XML, 2004), l'acronimo di *eXtensible Markup Language*, ha permesso la definizione di una serie di regole attraverso la formulazione dei *tag ad-hoc*. Attraverso dunque un linguaggio di *meta-markup*, è possibile specificare una sintassi per la definizione di altri linguaggi di *markup*, strutturati e specifici di un determinato dominio. Per i nostri scopi l'utilizzo di *XML* ha permesso di specificare le strutture gerarchiche mappate all'interno del nostro grafico ad albero rappresentante il flusso dei dati provenienti dalla periferica P5 verso il sistema *Studierstube* (vedi Figura 27).

L'estensione del linguaggio di *markup* creato attraverso l'*XML* è stato convenientemente descritto sia attraverso l'ampliamento del documento *Document Type Definition* (*DTD*) *OpenTracker.dtd*, sia in maniera ancor più espressiva, attraverso l'ampliamento di un *XML Schema (XSD)*. Il file *DTD* è stato ampliato per descrivere in maniera precisa ed inequivocabile la struttura di linguaggio di *markup* necessario al P5. Il documento *XML* deve essere conforme al *DTD* e rispettarne le regole in esso contenute. L'adozione dell'*XML* conferisce grande flessibilità al sistema in quanto l'*XML* è uno standard proposto dal *World Wide Web Consortium* (*W3C,* 2004*)* e supportato a livello mondiale da tutti gli sviluppatori software. Un *parser XML,* permetta di validare il documento di configurazione che adotta il P5 nei confronti del file *DTD*.

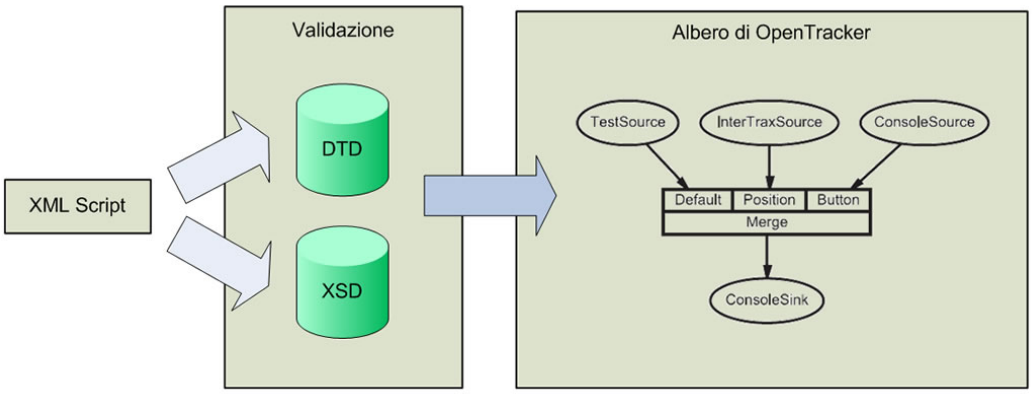


**Figura 27: La creazione dell'albero di OpenTracker dallo script XML**

Per dimostrare l'estrema efficacia del metodo seguito di seguito è riportato un frammento del codice *XML* implementato per il supporto del dispositivo P5 in *OpenTracker*.

```
<...>
<OpenTracker>

      <configuration>
            <ConsoleConfig headerline="P5 Output"
            interval="10"/>
      </configuration>

      <ConsoleSink comment="Test Source 1">
            <EventTransform translation="1 1 1">
                  <P5GloveSource finger="4"/>
            </EventTransform>
      </ConsoleSink>

      <ConsoleSink comment="Test Source 2">
            <EventTransform translation="1 1 1">
                  <P5FingerSource/>
            </EventTransform>
```

```
        </ConsoleSink>

        <ConsoleSink comment="Test Source 3">
              <EventTransform translation="1 1 1">
                    <P5ButtonSource/>
              </EventTransform>
        </ConsoleSink>

</OpenTracker>
<...>
```

Dopo le linee contenenti l'intestazione tipica e necessaria per ogni file *XML*, la definizione del file *OpenTracker.dtd*, troviamo un generico elemento *OpenTracker* che comprende al suo interno l'intero file di configurazione mediante il quale è possibile settare i parametri generali del modulo della console di *input/output*, in questo caso l'intestazione da mostrare *P5 Output* e l'intervallo di *sampling* per l'aggiornamento dei dati visualizzati. Le opzioni di configurazione per i moduli sono nodi figli degli elementi di configurazione. Queste appena descritte sono le parti fondamentali che devono necessariamente comparire in ogni file *XML* di configurazione per *OpenTracker*.

Tutto ciò che segue gli elementi di configurazione è necessario per configurare i nuovi moduli di *OpenTracker* sviluppati a supporto del P5 e fa parte della descrizione del flusso di dati dalla periferica P5 verso *Studiertube*. Ogni elemento è mappato in un nuovo o esistente nodo usato da *OpenTracker* mediante l'utilizzo dei *tag*. Gli attributi di un elemento specificano i parametri di configurazione del nodo stesso, quali ad esempio i dettagli relativi ad una trasformazione o l'indirizzo *IP* di un gruppo *multicast*. Nel caso specifico, dopo i file di configurazione troviamo un semplice grafico lineare composto da una serie di elementi nidificati. L'elemento più esterno è una *ConsoleSink*, corrispondente ad un nodo *ConsoleSink* incaricato di visualizzare i dati sulla console *MS-DOS*. Il figlio è

invece un nodo *EventTransform*, che trasla i dati entranti secondo il vettore di traslazione (1, 1, 1). Nella parte più interna, troviamo il *settaggio* relativo al numero delle dita che compongono la mano.

Il grafico lineare, generato tramite il codice *XML* compilato per la periferica P5, composto dagli elementi nidificati appena descritti e illustrato in maniera simbolica in Figura 28, costituisce la configurazione degli elementi relativi alla classe sviluppata *ad-hoc* e chiamata *P5GloveSource*, nella quale vengono *bufferizzati* i dati relativi alla traslazione e alla rotazione del guanto. La stessa identica struttura viene ripetuta per le classi che gestiscono i *buffer* dei valori relativi a piegamento delle dita e stato dei bottoni, rispettivamente nelle nuove classi *P5FingerSource* e *P5ButtonSource*.

La direzione del flusso dei dati attraverso lo schema in Figura 28 procede sempre dai nodi-figlio verso i nodi-genitore, ed è per questo che i nodi *source*, i generatori di nuovi eventi, si trovano sempre nella struttura più interna dei *tag XML* (vedi frammento di codice XML a pagina precedente). I *sink*, i nodi incaricati di portare i dati provenienti dalla periferica all'esterno di *OpenTracker*, si trovano invece tipicamente vicino all'elemento radice "*OpenTracker*".

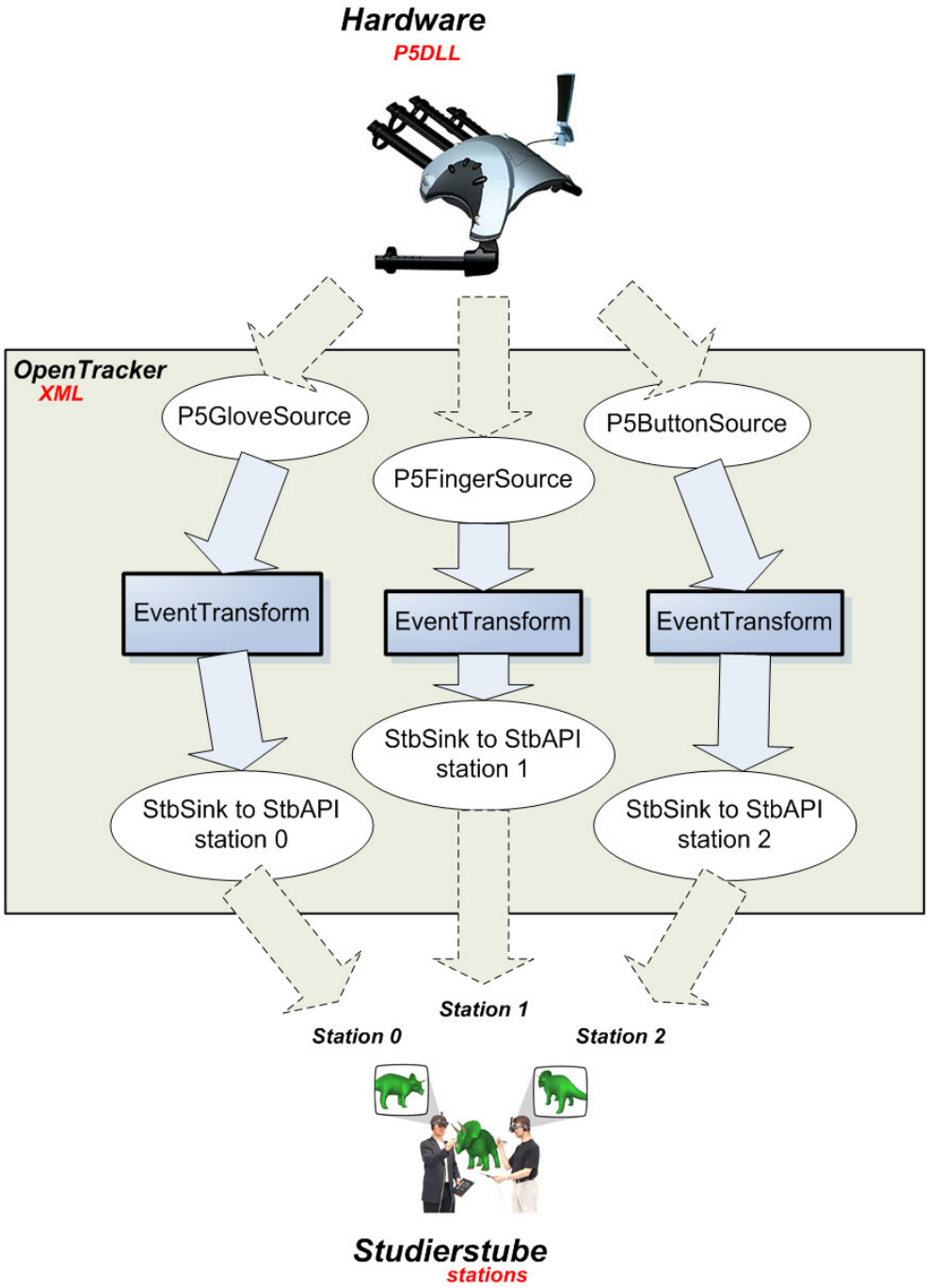


**Figura 28 : Rappresentazione grafica corrispondente all'albero generato tramite il codice XML**

Come è possibile vedere dal grafico riportato, *Studierstube* riceve i dati di *tracking* attraverso un certo numero di "*station*", che operano come canali di

informazione per i dati. Ad ogni *station* può corrispondere uno o diversi dispositivi di input. Nel sistema sviluppato e qui illustrato, la penna e la tavoletta sono controllate ognuna da una stazione indipendente, mentre le stazioni che ricevono i dati dal guanto P5 sono tre: la prima accoglie i dati di traslazione del guanto, una seconda i valori relativi al piegamento della dita, mentre l'ultima trasmette a *Studierstube* lo stato dei bottoni.

E' possibile accedere alle informazioni contenute nelle stazioni seguendo un duplice approccio sia pubblicando i dati attraverso la creazione di *fields* globali in *Open Inventor*, oppure usando il sistema ad eventi di *Studierstube*, trasportando i dati attraverso le "*3D event action*" applicate al *scene-graph* dell'ambiente *VR*.

Mediante l'utilizzo di *OpenTracker* è stato dunque possibile avere un ottimo livello di indipendenza dal tipo di dispositivo *tracker* utilizzato: l'approccio usato ha permesso lo sviluppo in modo tale da offrire lo stesso tipo di interfaccia per dispositivi differenti. Il sistema inoltre supporta una estendibilità lineare proprio per il fatto che, mediante l'utilizzo di moduli di configurazione *XML*, risulta semplice ottenere nuove implementazioni della stessa interfaccia per supportare un numero maggiore di dispositivi diversi.

Tuttavia anche questo sistema presenta ancora dei limiti, probabilmente per via del recente sviluppo, quali:

- Il supporto per configurazioni sperimentali complesse necessarie per la gestione di un sistema in grado di combinare i dati provenienti da due sorgenti diverse.
- Complesse modifiche geometriche della scena grafica complesse quali, ad esempio, l'interpretazione di dati di un dispositivo in relazione a quelli di un altro.
- Il supporto nativo di filtri per la trasformazione dei dati in *input*.

- La capacità per la fusione semantica dei dati provenienti da due dispositivi diversi.
- L'estendibilità gerarchica secondo l'approccio *object-oriented* per l'aggiunta di nuovo codice a tutti i livelli.

### 2.3.4 Funzionamento dei Moduli Implementati

Come abbiamo già accennato nei paragrafi precedenti, le classi *modules* servono a definire speciali funzioni tramite le quali è possibile effettuare calcoli, filtraggi sui segnali in input o altri calcoli che l'applicazione necessiti. Si può quindi affermare che i *modules* sono elementi funzionali della struttura di *OpenTracker*.

Per l'implementazione dei moduli atti al funzionamento del guanto P5 con il sistema, sono state sviluppate le classi *P5GloveModule*, *P5FingerModule* e *P5ButtonModule*, tutte estese dalla classe *module*, nelle quali sono presenti le funzioni necessarie all'interfacciamento.

Durante l'esecuzione del programma, i moduli sviluppati passano attraverso numerosi stadi: da principio, generalmente in fase di inizializzazione, essi vengono istanziati dalla funzione globale *initializeContex(),* che li aggiunge al *contex, un* oggetto del quale abbiamo parlato nel paragrafo dedicato a Studierstube (vedi paragrafo 1.4). In un secondo momento, dopo la chiamata del metodo *Module::init()*, viene effettuato il *parsing* dei parametri provenienti dal file *XML* di configurazione che contengono gli attributi dell'elemento di configurazione per il P5 e dei sotto alberi di nodi-figlio presenti negli oggetti *ConfigNode*. Non appena finita la lettura del *file* di configurazione, quando ormai la struttura dei nodi è stata definita, viene chiamato il metodo *Module::start(),* proprio di tutti i *modules*. A questo punto vengono effettuate tutte le altre inizializzazioni dipendenti dai parametri di configurazione e dai nodi istanziati, al termine delle quali è possibile far entrare in funzione il *loop* principale dell'applicazione per il *triggering* dei dati in entrata.

A questo punto sono disponibili, le chiamate ai metodi:

- *Module::pushState()*
- *Module::pullState()*
- *Module::stop()*

Questi sono rispettivamente il metodo incaricato della generazione di nuovi eventi di aggiornamento e del loro inserimento nell'albero attraverso i nodi *EventGenerator,* il metodo incaricato di analizzare lo stato dei dati attraverso la scansione della struttura dei nodi condivisa e infine il metodo che interroga i moduli per capire se vogliono interrompere il *loop* o fermare il programma mediante la chiamata del metodo *Module::close()*.

### 2.3.5 La struttura dati adottata

Per una adeguata gestione dei dati in entrata dalla periferica, questi sono stati prima convertiti da gradi a radianti e poi passati ai metodi *eulerToQuaternion()* e *normalizeQuaternion()* che hanno permesso la trasformazione dei tre valori di *yaw*, *pitch* e *roll*, corrispondenti alla rotazione del guanto convertiti in radianti, in quaternioni normalizzati. Per quanto riguarda invece il trattamento dei dati relativi alle traslazioni sugli assi *X, Y, Z* del P5, sono stati applicati dei filtri dinamici e delle trasformazioni tramite una funzione polinomiale di accelerazione di terzo grado per correggere e migliorare la fluidità nella visualizzazione del guanto.

Il funzionamento del filtro dinamico si basa sulla capacità di cambiare la soglia di *filtering* dei dati in base alla velocità di spostamento della mano dell'utente. Più precisamente in condizioni di bassa velocità di movimento della mano saranno elaborati molti aggiornamenti della scena (tecnicamente molti *frames*), viceversa, in presenza di bruschi movimenti della mano la produzione di *frames* verrà ridotta notevolmente. Questo approccio ha permesso di aumentare ulteriormente la fluidità del modello guidato dal guanto nella scena *VR*. Le variabili salvate e continuamente

aggiornate nelle *sources* mediante la chiamata al metodo *P5GloveModule::pushState()* sono di tipo intero per i dati relativi alle posizioni, di tipo *float* per i valori incrementali di spostamento e di rotazione.

Il grafico in Figura 29 mostra il diagramma delle classi *UML* corrispondente alla configurazione dei *module* utilizzati per l'interfacciamento della periferica P5 con *Studierstube*, in relazione alla libreria *DLL* (*CP5DLL*) e alle *source* utilizzate. Come si vede dalla Figura 29, l'elemento *CP5DLL* si interfaccia con le tre classi *P5GloveModule, P5FingereModule* e *P5ButtonModule* alle quali fornisce i metodi per attingere a tutti i dati necessari al *tracking* della periferica P5.

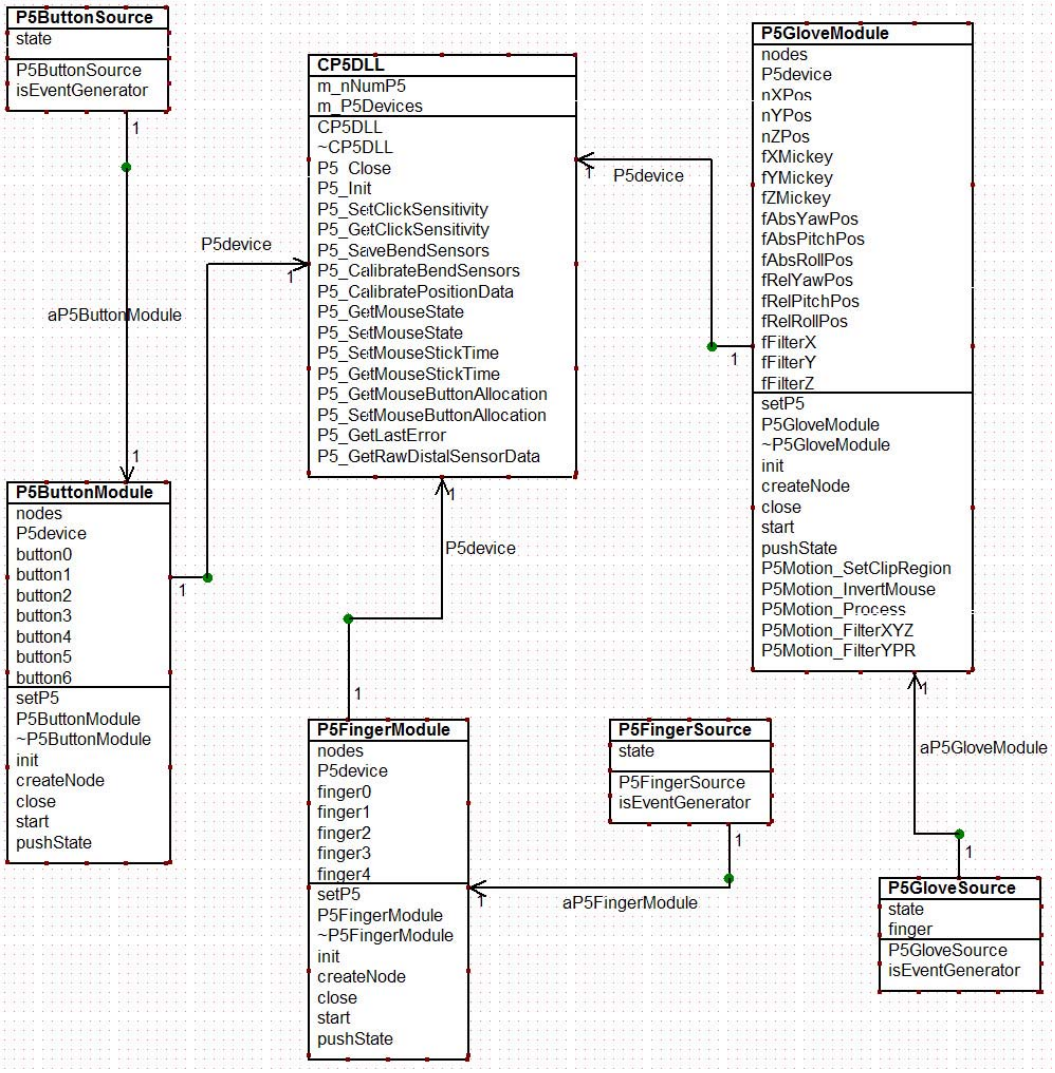


**Figura 29 : Diagramma UML delle classi che estendono le funzionalità di OpenTracker**

Mentre le variabili *nXPos, nYPos* e *nYPos* relative alla classe *P5GloveModule* sono di tipo intero e corrispondono alle attuali posizioni *X, Y, Z* del cursore, le variabili *fXMickey, fYMickey* e *fZMickey* sono di tipo *float* e corrispondono all'incremento della posizione del cursore *frame* per *frame*. Infine come già accennato nei capitoli precedenti, le variabili relative al piegamento delle dita possono assumere dei valori compresi tra (0, 63), sono stati perciò utilizzati dei valori di tipo intero.

Le classi *P5GloveModule, P5FingereModule* e *P5ButtonModule* implementano nella loro struttura le funzioni necessarie alla costruzione della struttura dati, al filtraggio ed alla formattazione dei valori per il passaggio successivo verso le classi corrispondenti, incaricate di fungere da contenitori dei dati: *P5GloveSource, P5FingerSource e P5ButtonSource*. In particolare, con i metodi *start* e *close* vengono aperte e chiuse le librerie relative al *P5Glove*, *P5Finger* e *P5Button*, mentre la funzione *createNode()* è chiamata ogniqualvolta sia necessario costruire un nuovo nodo. La funzione è in grado di controllare la lista di elementi contenuti in *P5GloveSource*, non appena viene trovata una corrispondenza tra il nome passato tramite l'argomento della funzione ed un qualche elemento di *P5GloveSource*, in essa viene generato un nuovo nodo. Infine, tramite il metodo *pushState()* è possible mandare eventi attraverso l'albero dei *tracker*, in accordo alla configurazione dei nodi.

Le *source*, rappresentate dalle classi *P5GloveSource, P5FingerSource e P5ButtonSource,* sono generatrici di eventi capaci di restituire posizione corrente, stato dei bottoni e delle dita della periferica P5, implementando ad intervalli regolari una *flag* e aggiornando i valori del sistema di rilevamento degli eventi.

Sebbene la periferica sia caratterizzata dalla presenza di quattro bottoni, di cui uno ne determina l'accensione e lo spegnimento, la struttura dati adottata permetterebbe il trattamento di sette variabili, corrispondenti allo stato di sette bottoni. Questo è il numero di slot caratteristici che *Studierstube* assegna ad ogni sua *station*, che tipicamente viene adottato per trasmettere i tre valori necessari per la posizione dell'oggetto *trackato* più quattro valori corrispondenti al quaternione che ne rappresenta l'orientamento.

## 2.4 Trasferimento dei Dati da OT a STB: l'Engine Implementato per il P5

L'approccio inizialmente seguito, prevedeva la definizione di una nuova struttura dati comprendente, oltre ai valori delle traslazioni e delle rotazioni del guanto, anche i valori delle dita e dello stato dei pulsanti. Questo approccio tuttavia avrebbe comportato la necessità di modificare un segmento non indifferente del codice esistente per il *tracking* della penna in *Studierstube*. Questo nuovo *re-engineering* del sistema avrebbe dunque portato all'implementazione di una struttura *ad-hoc* per la periferica P5, in contrasto con lo spirito di *OpenTracker* che promuove invece una architettura universalmente riconosciuta dalla molteplicità dei dispositivi di input.

Per questa ragione si è preferito implementare l'applicazione sfruttando le strutture dati già esistenti. Sono state, per questo, istanziate due nuove stazioni di *Studierstube* i cui *slot*, tipicamente adibiti al contenimento dei dati per le traslazioni e le rotazioni dell'oggetto *trackato,* sono stati utilizzati come *buffer* per il trasferimento dei valori relativi al piegamento delle dita e alla pressione dei bottoni. Degli *engine* sono poi stati utilizzati per far colloquiare le differenti parti della struttura caratterizzate dalla presenza di *field* di tipo diverso.

Infine, sulla traccia della classe *SoTrackEngine* implementata in Studierstube per il *wiring* dei dati tra la struttura *OpenTracker* e la geometria della penna, ancora valida per il *tracking* del guanto, è stata implementa una nuova classe chiamata *P5Engine* che opera un *wiring ad-hoc* per i dati relativi al guanto. Tale classe estende la classe *engine* di Inventor e permette di assicurare il mantenimento della "connessione" tra dati interni al codice. Si potrebbe pensare ad un *engine* come ad una scatola nera che riceve in *input* dei valori, esegue alcune operazioni su di essi e produce una serie di valori di *output* (vedi Figura 30).

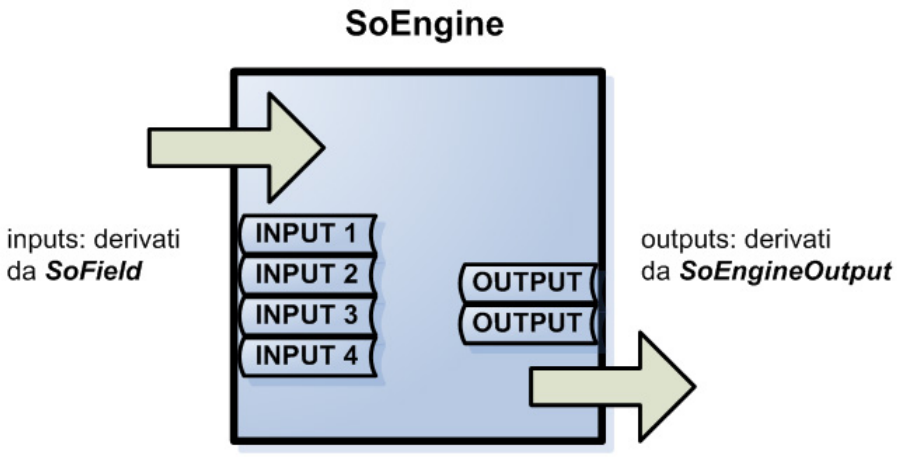


**Figura 30 : Schema generale di rappresentazione di un engine**

In generale gli *engine* sono oggetti adoperati per animare o vincolare determinate parti della scena in relazione ad altri oggetti appartenenti a quella stessa scena. Gli *engine* possono anche essere usati per rispondere in determinati modi ad eventi esterni, determinati ad esempio dall'attività del mouse o della tastiera. Connettendo i campi relativi a due nodi diversi all'interno della scena è possibile infatti fare in modo che il cambiamento dei dati nel *field* del primo nodo inneschi, in maniera automatica, un processo di eventi capaci di aggiornare i valori relativi al *field* del secondo nodo. Lo stesso approccio è stato seguito per trasmettere i valori alla geometria della mano virtuale, attraverso il collegamento in serie di gruppi di *engine*. In questo modo fare in modo che più *engine* reagiscano tra loro, ottenendo come risultato animazioni sincrone tra più parti della scena (vedi Figura 31).

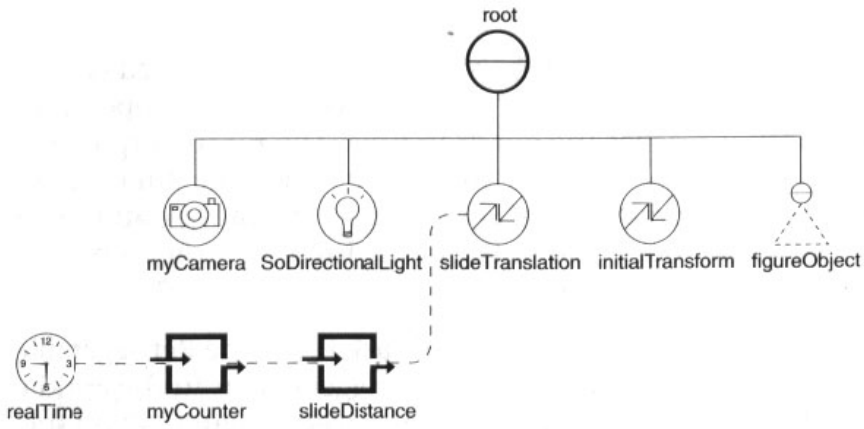


**Figura 31 : Collegamento di più engine per animazioni sincrone tra diverse parti della scena**

I valori in *input* e quelli di *output* sono direttamente connessi ad alcuni campi di altri nodi del grafo di scena, quindi l'output prodotto da un *engine* verrà copiato nei campi dei nodi relativi.

In particolare, nel caso dell'applicazione sviluppata, i dati in uscita provenienti dai diversi *TrackEngine* adoperati per trasferire le informazioni relative alla mano, alle dita ed ai bottoni, vengono a loro volta collegati ad i valori in ingresso del *P5Engine* che provvede a fornire i valori finali utilizzati dalla struttura della classe *SoGloveKit* dedicata alla definizione della geometria del modello.

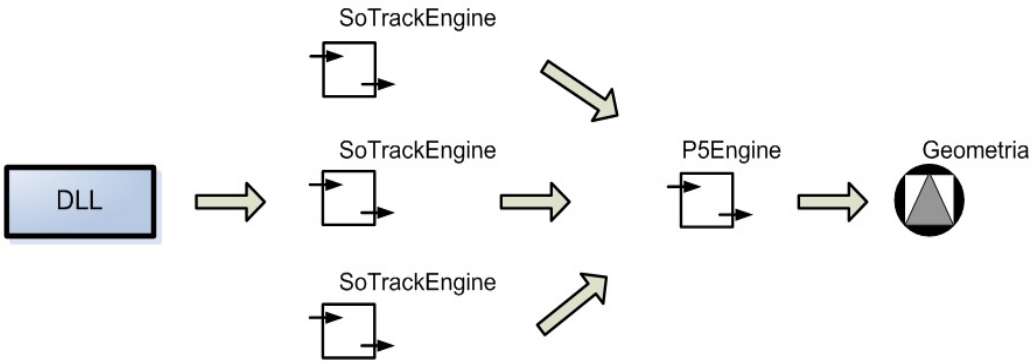


**Figura 32: La catena di engine adottata**

Attraverso la classe implementata, chiamata *P5Engine*, è stato dunque possibile sincronizzare, in ultima istanza, la geometria con i dati in uscita da *OpenTracker*, provenienti a loro volta dai *TrackEngine* attraverso la struttura delle stazioni di *Studierstube*. In questa classe sono state infatti definite le variabili di entrata ed uscita dall'*engine*, come è possibile vedere dal grafico in Figura 33, dove le variabili di *input* sono di tipo *SoSFVec3f* corrispondente al *field* contenente un vettore tridimensionale, e di tipo *SoSFRotation* corrispondente al *field* contenente un vettore di rotazione definito secondo lo specifico quaternione.

Il vantaggio principale nell'implementare una struttura ad *engine* come quella per il trasferimento dei dati dal P5 verso Studierstube (vedi Figura 33) è stato quello di avere permesso la riduzione del codice da sviluppare. Infatti non vi è stata la necessità di implementare complesse procedure necessarie ad ottenere la sincronizzazione tra due parti diverse del codice e/o della scena grafica visto che il meccanismo di *engine* in cascata provvede automaticamente alla sincronizzazione dei *field* scelti.

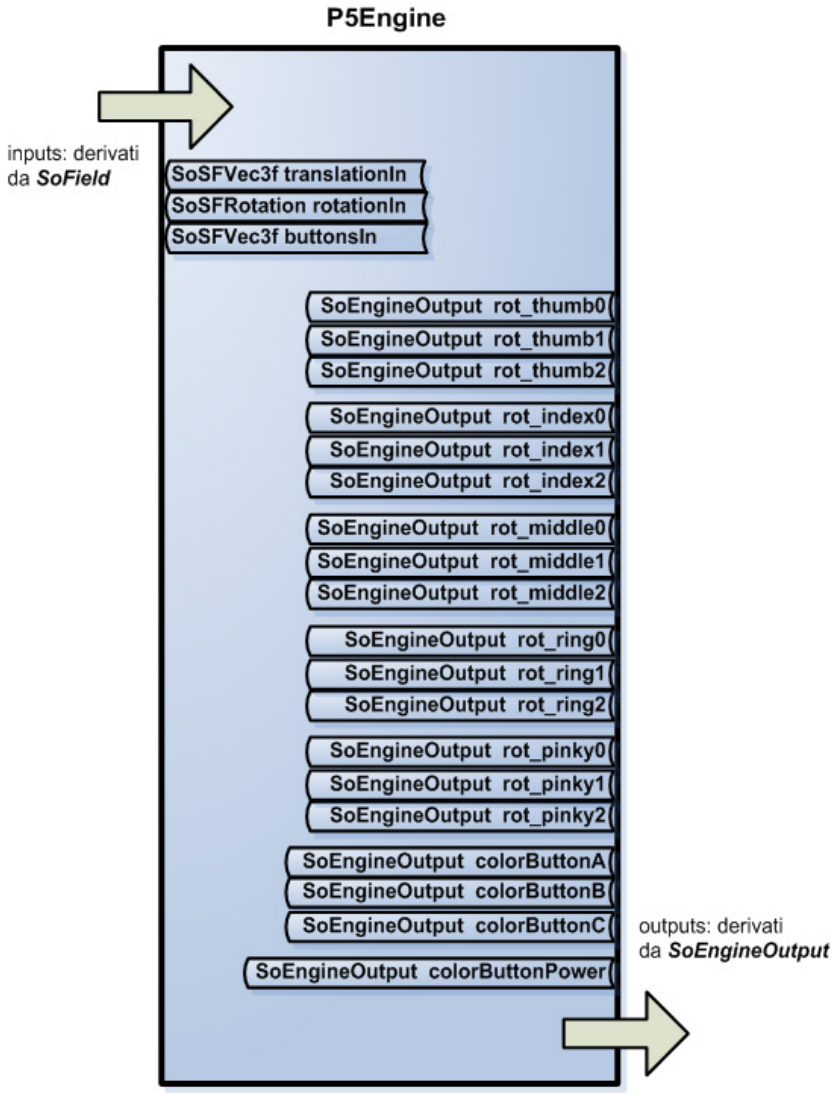


**Figura 33 : Input ed Output implementati nel P5Engine**

Le funzionalità avanzate degli *engine* hanno permesso, oltre alla semplice connessione con aggiornamento automatico, una conversione automatica dei valori, da un tipo ad un altro. Il valore del campo di *input* di un *field* di tipo differente, è automaticamente convertito, nel rispetto di determinate condizioni, nel tipo di campo connesso all'*output*. In particolare, come illustrato in Figura 33, mediante il primo vettore di input inizialmente definito sono stati associati i tre valori dedicati alla rotazione, ai valori del piegamento delle prime tre dita, pollice, indice e medio

(rispettivamente *thumb, index e middle* in inglese). Invece i quattro *slot* associati al vettore di rotazione del secondo vettore di *input* predefinito, sono stati riempiti per metà, con i rimanenti due valori delle dita corrispondenti al piegamento di anulare e mignolo (rispettivamente *ring e pinky* in inglese).

La classe *P5Engine* è stata poi arricchita con l'implementazione di *linked list* e funzioni di *callback* connesse alla pressione dei bottoni. E' stato implementato infatti un sistema per cui con la pressione di un bottone sulla periferica si determina la chiamata ad una serie di funzioni definite a *runtime* in una *linked list* mediante l'utilizzo di *callback* che possono essere personalizzate nell'applicazione finale.

## 2.5 Geometria del Guanto Virtuale

La struttura geometrica della penna virtuale propria del sistema *Studierstube* è piuttosto semplice se paragonata alla struttura propria del modello virtuale del guanto implementato. Essa è costituita da tre semplici primitive, un cono per la punta, un cilindro come corpo ed una sfera all'estremità.

Diversamente la geometria relativa al modello del guanto virtuale ha richiesto una maggiore complessità grafica. Tuttavia, per mantenere una rappresentazione iconica ed al contempo adeguata all'arto umano, la sua rappresentazione geometrica è stata costruita in modo da volutamente rappresentare un arto "robotico", piuttosto che un arto umano, in modo da risultare semplice ma al contempo efficiente. Per la sua realizzazione sono stati utilizzati dunque quindici parallelepipedi, tre per ogni dito, e cinque sfere, una per la rappresentazione del palmo, le altre quattro per i bottoni.

Nell'ambito dello studio dei metodi di implementazione di un modello virtuale complesso come quello del guanto per la periferica P5, è stato subito chiaro come sarebbe stato indispensabile l'utilizzo di particolari strumenti per stabilire le volute relazioni e vincoli tra le parti costitutive del modello stesso. La chiusura del

pugno da parte dell'utente ad esempio, implica un articolato movimento delle falangi, determinato da una corretta sincronia delle rotazioni di falange, falangina e falangetta.

### 2.5.1 Definizione della Struttura per la Gerarchia delle Geometrie

Le difficoltà incontrate nella definizione di giusti vincoli relativi alla cinematica delle geometrie del dito, hanno portato all'implementazione di un nuovo gruppo di classi costruite *ad-hoc* utilizzando le strutture definite dai *node-kits* di *Open Inventor*. I *node-kits* sono particolari nodi di *Open Inventor* caratterizzati dalla capacità di organizzare i propri nodi-figlio all'interno di una determinata struttura, definita dalla molteplicità delle parti del cosiddetto catalogo. La creazione del nuovo *node-kit* chiamato *SoGloveKit* ha richiesto innanzitutto la creazione di un catalogo (in aggiunta alle normali procedure per la dichiarazione di derivazione dal nodo standard), ovvero del contenitore nel quale vengono inizializzate tutte le strutture costituenti il particolare *node-kit* implementato. Una volta costruito il catalogo, si è proceduto all'implementazione delle funzionalità specifiche del nuovo *node-kit* implementando i metodi *setPart()*, *getPart()*, e *createPathToPart()*, definiti dalla classe madre di tutti i *node-kits*, la classe *SoBaseKit*.

È opportuno notare che *SoBaseKit* deriva direttamente dal generico nodo *SoNode*, e dunque durante la creazione del *SoGloveKit*, si sono dovute seguire le procedure standard per la creazione e l'utilizzo di nuove classi di nodi. In particolare nel nuovo *node-kit* sono state implementate alcune macro definite nel file *SoSubKit.h* di *Open Inventor*, necessarie al corretto funzionamento della classe.

Per la precisione, nella macro *SO_KIT_HEADER* sono stati dichiarati l'identificatore ed il nome delle variabili e dei metodi propri del *node-kit* implementato. Inoltre con essi si è definita la variabile privata *nodekitCatalog* ed un metodo protetto virtuale per il suo recupero, il metodo *getNodekitCatalog()* che definisce a sua volta un altro metodo chiamato *getClassNodekitCatalog()*.

Oltre alla scrittura e alla definizione di variabili e metodi tramite le macro, la quasi totalità del lavoro si è concentrata sulla creazione delle parti costituenti il catalogo del *SoGloveKit*, una variabile di tipo statico condivisa da tutte le istanze della classe. Il catalogo del *SoGloveKit* viene dunque creato, in maniera automatica, nel momento in cui viene creata la prima istanza di quella classe. Il catalogo può venir consultato ogniqualvolta un utente richieda l'aggiunta o il recupero di una parte attraverso i metodi *getPart()* e *setPart()*.

### 2.5.2 Inizializzazione del SoGloveKit

Le macro utilizzate per la definizione della geometria del guanto all'interno del catalogo, hanno visto la definizione, mediante la macro *SO_KIT_CATALOG_ENTRY_HEADER*, di variabili per il palmo, le dita, la cosiddetta *childlist* e la *dragGeometry* (un campo usato dallo *StbWinManager* per il salvataggio temporaneo della geometria). Le *ChildList* sono classi per la definizione di liste *SoSeparatorKits* mediante le quali è possibile definire le gerarchie per il movimento. Si è proceduto quindi alla definizione del metodo *initClass(),* mediante il quale è stato possibile inizializzare le informazioni riguardo al tipo di *node-kit* utilizzato, e alla definizione di un costruttore atto ad ospitare le varie parti del catalogo.

L'inizializzazione della classe *SoGloveKit* è stata fatta inserendo la macro *SO__KIT_INIT_CLASS()* all'interno del metodo *initClass()* della classe *GloveKit*:

```
SO__KIT_INIT_CLASS(SoGloveKit, "GloveKit",
SoTrackedArtifactKit);
BASE3D_INIT_CLASS(SoGloveKit);
```

All'interno del costruttore della classe *nede-kit*, sono state quindi definite le parti costitutive del catalogo proprio del *SoGloveKit* e la lista delle parti per ogni istanza del *node-kit* stesso.

### 2.5.3 Popolamento del Catalogo

Inizialmente è stata creata la sfera relativa al palmo del guanto creando una nuova istanza di shapekit, *shapeKit_palm* ed associando ad essa l'oggetto *SoSphere sphere_palm*. Di seguito sono state implementate le sfere relative ai quattro bottoni con i nuovi oggetti *shapekit shapeKit_buttonA*, *shapeKit_buttonB*, *shapeKit_buttonC*, *shapeKit_buttonPower*, le traslazioni, istanze della classe *SoTransform*, *transf_buttonA, transf_buttonB, transf_buttonC, transf_buttonPower* ed i materiali, istanze dalla classe *SoMaterial*, *red_buttonA, red_buttonA, red_buttonA, green_buttonA, blue_buttonA, yellow_buttonPOWER*.

L'implementazione della struttura del dito è avvenuta creando tre istanze della classe *SoCube*, per il pollice ad esempio, sono state creati *cube_thumb0, cube_thumb1* e *cube_thumb2*. *cube_thumb0* è stato aggiunto al catalogo mediante l'utilizzo del metodo *SetPart(),* come primo elemento della *childlist* dell'oggetto *this* ovvero della classe *SoGloveKit*. *Cube_thumb1* è stato aggiunto al catalogo come primo elemento della *childlist* di *cube_thumb0*, determinando in questo modo la gerarchia di movimento della prima falange rispetto alla seconda.

Come illustrato in Figura 34 la struttura ottenuta mediante l'implementazione e la configurazione degli elementi costitutivi del *SoGloveKit* associati all'utilizzo delle *childlist* per il *setting* delle gerarchie di movimento, può essere espressa disegnando l'albero della scena grafica costituito dalle relazioni di dipendenza esistenti tra i nodi di *Open Inventor*.

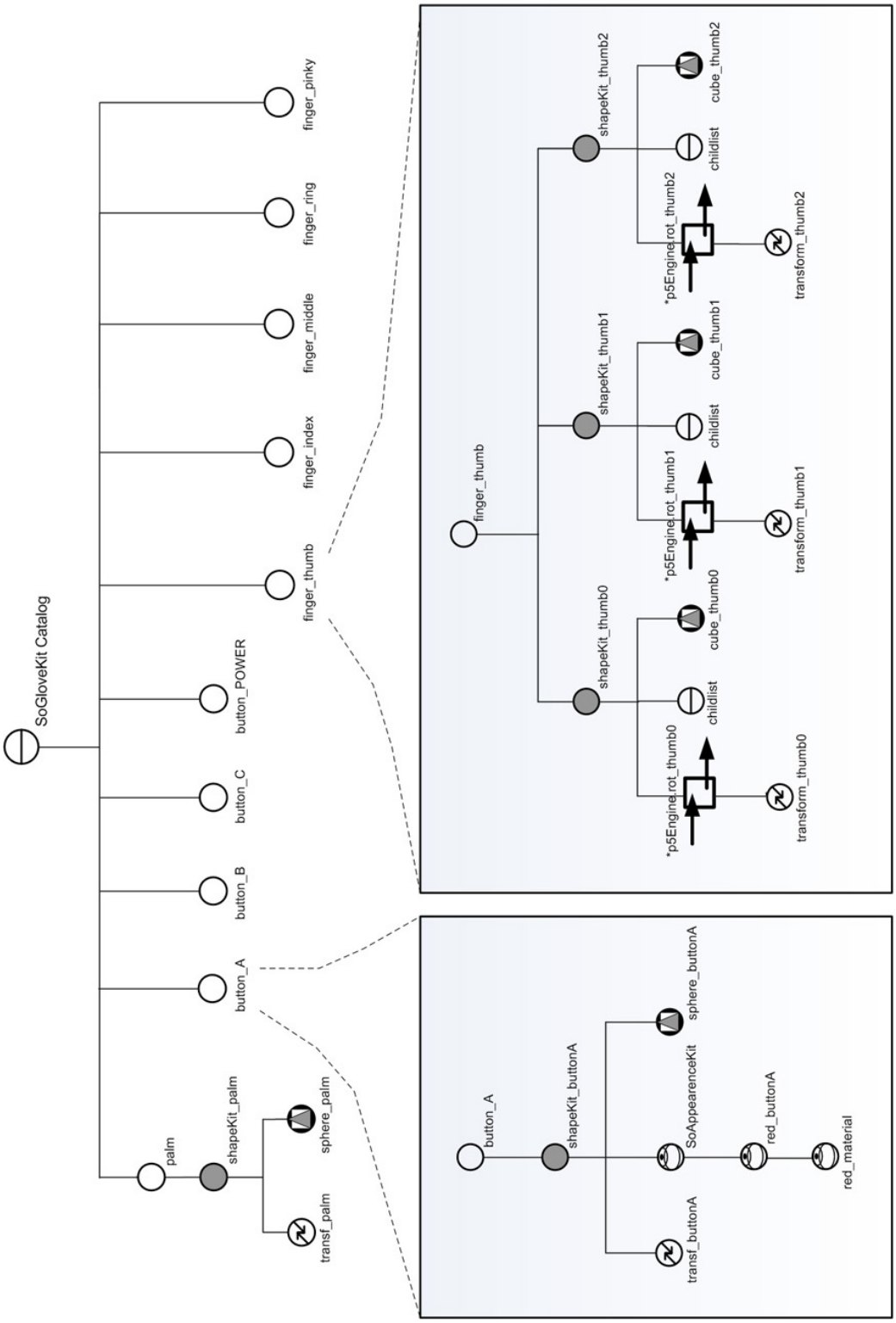


**Figura 34 : Lo scene-graph dei nodi relativo alla geometria del modello P5**

La struttura ad albero di *Open Inventor* in Figura 34 è sviluppata in modo da ottimizzare i processi di *rendering* e l'ottimizzazione dipende direttamente dalla velocità di lettura dell'albero che rappresenta la scena grafica, ovvero dalla velocità di lettura del database di *Open Inventor* che rappresenta la scena, il cosiddetto *scene-graph*. In particolare *Open Inventor* esegue la lettura sottostando a determinate regole di scansione, i nodi dello *scene-graph* sono scansionati iniziando da sinistra verso destra e dall'alto verso il basso percorrendo prima fino in fondo le ramificazioni costituite dai figli dei nodi principali. I vincoli di lettura definiti da queste regole, servono oltre che per ottimizzare il *rendering* della scena tramite la velocizzazione nella lettura dei nodi, anche per definire una struttura della scena inequivocabile, e per definire la gerarchia alla base di un concetto di ereditarietà che influenzerà la visione finale della scena 3D rappresentata dei nodi.

In questo modo, come si osserva in Figura 34, il nodo *transform* definito per la traslazione di una data geometria (per esempio il palmo della mano) farà traslare tutti i nodi, compresi i figli, a destra del nodo di traslazione. Per questo, nel sistema è stato definito per prima cosa il nodo relativo al palmo della mano, di seguito i nodi relativi ai bottoni e alle dita. La modifica dei valori del quaternione relativo al nodo a monte della geometria del palmo produrrà una rotazione del palmo e di tutti gli oggetti definiti successivamente, ovvero bottoni e dita.

## 2.6 Interazione tramite Riconoscimento Vocale

La piattaforma del sistema *OpenTracker* offre la possibilità di sviluppare un sistema di interazione multimodale che può ancor più avvicinarsi al paradigma di interazione uomo tipico dell'uomo. Tramite il *set-up,* la configurazione del modulo *XML* e la compilazione di determinati moduli e classi *OpenTracker* ha permesso lo sviluppo di un sistema altamente configurabile che permettesse di usare la mano ed al contempo la voce per impartire semplici comandi.

È interessante notare come questa forma di interazione si sia rivelata come forma di comunicazione dominante, potendo combinare con essa diversi livelli di astrazione inaccessibili ad altre modalità di input. Sfortunatamente, i sistemi di interazione tramite input vocale sono tutti caratterizzati da un limitato vocabolario di comandi vocali che obbliga l'utente ad usare una determinata sintassi di comandi predefiniti.

### 2.6.1 Supporto di Riconoscimento Vocale per OpenTracker

Il supporto per il sistema *SAPI* (Speech Application Programming Interface) di *OpenTracker*, rappresenta una soluzione di implementazione flessibile ed estendibile per il riconoscimento vocale in *Studierstube*. Gli eventi sono generati da un modulo appartenente al sistema di *Speech Recognition* (nel caso specifico Microsoft SAPI 5.1) e vengono passati al meccanismo di gestione degli eventi di OpenTracker.

Il sistema utilizzato, supporta il riconoscimento vocale SR (Speech Recognition) ed il *TTS* (Text To Speech) in varie lingue, ed è completamente configurabile tramite un file *XML*. Da parte sua *Studierstube* fornisce supporto sia per *Widget* a riconoscimento vocale che per *SR/TTS*. La configurazione del modulo implementata per la periferica P5, ha permesso lo sviluppo di una applicazione di tipo a comando e controllo.

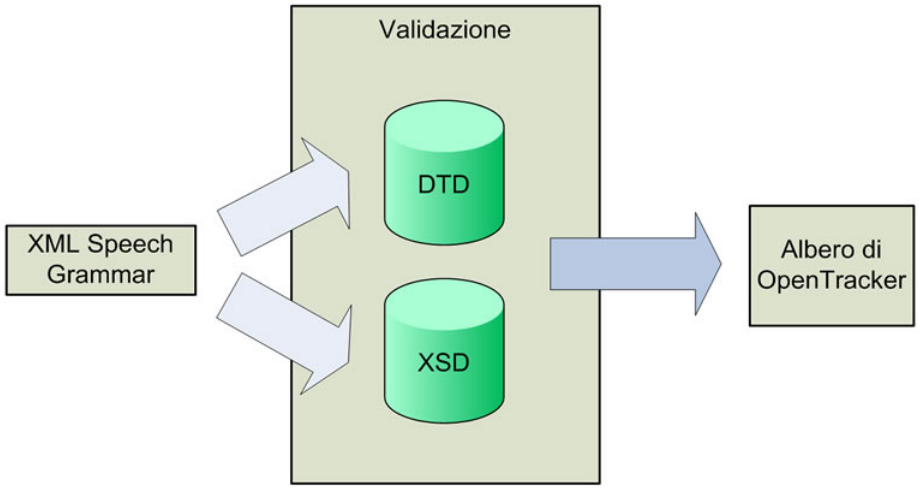


**Figura 35: Diagramma di inizializzazione di OpenTracker con SAPI**

I comandi sono definiti all'interno di un file *XML* che contiene una semplice "grammatica" degli eventi vocali (vedi Figura 36). In pratica tramite lo script *XML* è possibile configurare il sistema in modo che si attivi ogni qualvolta un comando specificio viene identificato. Come illustrato in Figura 36, *SAPI* individua il comando e passa l'evento ad *OpenTracker* il quale assegna un *ID* come definito nell'*XML* e lo invia alla rete tramite un s*ink*. Studierstube riceve l'evento con l'*ID* e riconosce, grazie alla grammatica *XML*, che appartiene ad uno degli comandi specificati ed avvia la funzione specifica come definito dal programmatore.

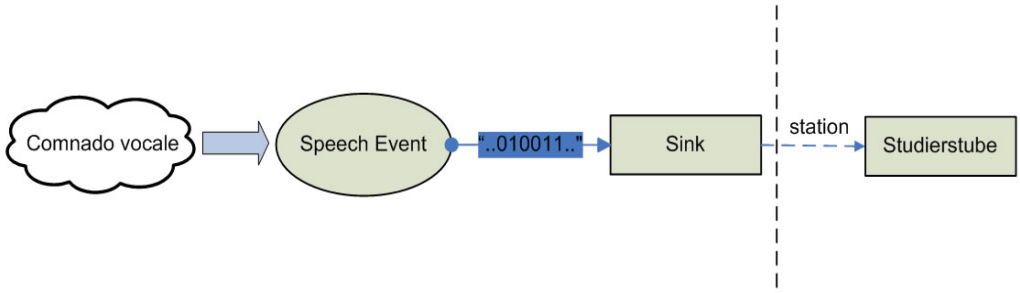


**Figura 36: Meccanismo di trasmissione dei dati con OpenTracker mediante comando vocale**

Di seguito verrà esposta la sezione di codice relativa alla configurazione del file *XML*, file nel quale sono stati definiti i comandi vocali di esempio.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OpenTracker SYSTEM "OpenTracker.dtd">
<OpenTracker>
      <configuration>
            <ConsoleConfig headerline="Sample Output"
            interval="10"/>
            <SpeechRecoConfig language="english">
                  <CommandSet id="CmdSet1">
                        <Command id="1234" name="button
                        A"/>
                        <Command id="1235" name="button
                        B"/>
                        <Command id="1236" name="button
                        C"/>
                  </CommandSet>
            </SpeechRecoConfig>
      </configuration>
      <StbSink station="0">
            <EventTransform scale="0.001 0.001 -0.001"
            translation="0 0 0">
                  <P5GloveSource finger="1"/>
            </EventTransform>
      </StbSink>
      <StbSink station="1">
            <P5FingerSource/>
      </StbSink>
      <StbSink station="2">
            <P5ButtonSource/>
      </StbSink>
            <ConsoleSink comment="SpeechRecoSource
            name=Wordsetset=CmdSet1">
            <StbSink station="3" event="on">
            <SpeechRecoSource name="Wordset"
            set="CmdSet1"/>
```

```
                </StbSink>
            </ConsoleSink>
        </OpenTracker>
```

All'interno della tag <*SpeechRecoConfig*> viene configurata la lingua desiderata propria del componente *SR*, nel caso specifico si è deciso di utilizzare la lingua inglese. Tramite il completamento dei tag <*CommandSet*> e <*CommandSet*> nidificate nel *tag* precedentemente descritto è possibile definire un set di comandi vocali. <*SpeechRecoSource*> rappresenta il nodo sorgente per il sistema di generazione di *speech events* tramite uno specifico set di comandi.

Dopo aver adeguatamente configurato il file *XML* relativo al *set-up* del dispositivo *SP* di *Opentracker*, sono state eseguite le seguenti procedure:

- Sono state definite le direttive di preprocesso *USE_OPENTRACKER* e *USE_SAPISPEECH* all'interno dell'ambinete di sviluppo C++.
- È stato incluso il modulo *SpeechModule.h* all'interno del progetto.
- È stata implementata una nuova stazione per il collegamento tra il sistema *SR* di *OpenTracker* e *Studierstube*.
- Il sistema è stato configurato in modo tale per cui gli eventi generati venissero mandati alla giusta stazione di *Studierstube*.

L'adempimento di queste procedure ha permesso il funzionamento del sistema di riconoscimento vocale tramite la definizione dei comandi scelti ed impostati nel file *XML*, sia il passaggio degli eventi, generati dopo il riconoscimento, al sistema *Studierstube*. L'ultimo passo è stato quindi la definizione di una procedura di gestione degli eventi e l'implementazione delle funzionalità associate ai comandi riconosciuti.

Il sistema implementato ha previsto l'attivazione e la disattivazione dei bottoni del guanto tramite comando vocale. Pronunciando i termini "*button A*", "*button B*" e "*button C*" lo stato del bottone corrispondente cambierà alternativamente da *ON* a *OFF*. Le sfere relative alla rappresentazione dei rispettivi bottoni presenti sul modello del guanto cambieranno colore mentre allo stesso tempo si attiveranno le funzioni di *callback* relative ai bottoni chiamati.

# 3 Risultati e Sviluppi Futuri

## 3.1 Risultati

Con questo lavoro, abbiamo presentato un nuovo approccio per l'interazione all'interno dell'ambiente *AR* di *Studierstube*. Il nostro obiettivo era quello di garantire la possibilità di interazione all'interno del sistema tramite il guanto di realtà virtuale P5, sostituendo il modello virtuale della penna classico di Studierstube con il modello del guanto implementato e descritto nella classe *SoGloveKit* (vedi Figura 37).

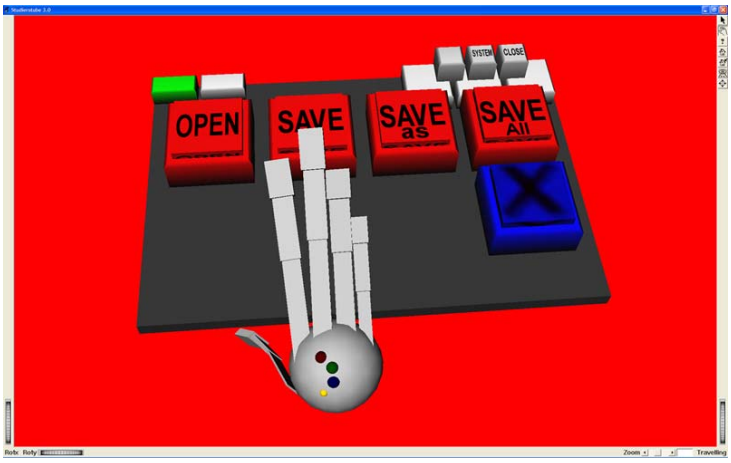


**Figura 37 : Ambiente Studierstube con interfaccia di interazione P5**

L'approccio seguito ci ha permesso di trarre pieno beneficio dal database di *Open Inventor* e dalla funzionalità di *scripting* offerta da *XML*, mediante la quale è stato possibile ridurre notevolmente la quantità di lavoro in termini di programmazione di codice per lo sviluppo di nuove applicazioni future. La libreria di sviluppo *Open Inventor Toolkit* supporta, per il suo database, il concetto di *scene-graph*, concetto che abbiamo utilizzato per l'implementazione di una applicazione flessibile e configurabile via *scripting*.

Come già accennato nei paragrafi precedenti, la forma di interazione offerta dai dispositivi tattili, si rivela molto efficace, le persone tendono infatti ad utilizzare i gesti per spiegare meglio quello che stanno dicendo, per cui voce e gestualità si completano vicendevolmente. Il sistema sviluppato ha dimostrato ottima efficienza e una buona stabilità, ma una limitata usabilità, determinata dal fatto che per ottenere *tracking* fluido e continuo, è necessario non uscire dal campo d'azione del ricevitore ad infrarosso. Inoltre, la presenza del filo di connessione tra il ricevitore e il guanto atto alla cattura dei dati generati dai sensori posizionati sulle dita, impedisce notevolmente la libertà di movimento. Lo sviluppo di una periferica P5 *wireless* risolverebbe il problema. Tuttavia, considerato il costo sostenuto per l'acquisto del dispositivo P5 in relazione al costo che avremmo dovuto sostenere per l'acquisto di altri sistemi più efficienti, ma assai più costosi, possiamo ritenerci soddisfatti del risultato ottenuto.

Infine le librerie utilizzate, quali ad esempio *Open Inventor* e Studierstube si sono rivelate altamente estendibili ed efficaci, determinate da un alto grado di riusabilità.

## 3.2 Lavori Futuri

Studierstube, così come il campo della *Augmented Reality*, è un sistema in costante sviluppo e perfezionamento, parecchie caratteristiche possono essere prese in considerazione per ulteriori miglioramenti del sistema. Per quanto riguarda l'implementazione del sistema per il funzionamento della periferica tattile P5 in ambiente Studierstube, un interessante punto di partenza per lo sviluppo di applicazioni future, potrebbe essere l'estensione della struttura, mediante l'implementazione di una nuova stazione in Studierstube, la modifica delle classi e dei file *XML* implementati, per l'utilizzo contemporaneo di due periferiche P5 per la definizione di una forma di interazione di tipo bi-manuale. Diversi studi hanno dimostrato infatti la naturale propensione dell'uomo all'utilizzo bi-manuale nella modellazione in ambiente *VR*. L'interfaccia bi-manuale comporterebbe una maggiore efficienza e un più alto livello di precisione, permettendo all'utilizzatore di ancorarsi all'ambiente virtuale e di realizzare con il proprio corpo un sistema di riferimento personale, derivato ad esempio dal fatto che la distanza tra le mani costituisce un ottimo strumento naturale per la definizione di grandezze e dimensioni.

Un secondo interessante punto di partenza per lo sviluppo di applicazioni future potrebbe essere quello di implementare un sistema semantico di riconoscimento dei gesti che, affiancato al sistema implementato per il riconoscimento vocale, possa garantire un sistema mediante il quale sia possibile definire una serie di comandi gestuali e vocali interpretati secondo una predefinita ma dinamica serie di regole euristiche.

Altre forme di estensione del sistema implemento, potrebbero essere lo sviluppo di un sistema di stabilizzazione dei dati in uscita dal dispositivo, basato su tecniche di interpolazione, oppure l'implementazione di una struttura mediante la quale sia possibile identificare un determinato set gestuale configurabile mediante *scripting XML* e la definizione di una libreria di gesti.

Per l'implementazione di nuove applicazioni, servono nuove idee di implementazione e lo sfruttamento di tutti i vantaggi offerti dall'utilizzo dei moderni sistemi di Realtà Virtuale, la chiave del successo sta nella definizione di valide ed innovative applicazioni per il sistema, tuttavia, l'incremento nello sviluppo in questo settore mediante la definizione di nuove idee deriverà direttamente dall'esperienza data dall'utilizzo delle applicazioni attuali.

# 4 Bibliografia


America's Army (2003). *Gioco di Realtà Virtuale sviluppato dalle forze armate statunitensi*. Reperito il 9 Febbraio 2004 dal World Wide Web: *http://www.goarmy.com/aagame/index.htm*

AVSoft FT2000. (2004). *Face Trackers for Facial Motion Capture*. Informazioni reperite il 16 Marzo 2004 dal World Wide Web: *http://www.metamotion.com/hardware/face-trackers.htm*

Barco. (2004). *Barco Projectors*. Informazioni reperite il 16 Marzo 2004 dal World Wide Web: *http://www.barco.com/*

Bolt, R. A., Herranz, E. (1992). *Two-handed gesture in multi-modal natural dialog*. Proceedings of the 5th annual ACM symposium on User interface software and technology.

Butz, A., Höllerer, T., Feiner, S., MacIntyre, B., Beshers, C. (1999). *Enveloping Computers and Users in a Collaborative 3D Augmented Reality*, Proc. IWAR '99.

Canziani, T., Spasse, S. (2002). *Visione Stereoscopica*. Informazioni reperite il 9 Marzo 2004 dal World Wide Web: *http://www.univ.trieste.it/~brain/visione/Stereopsi/Stereopsi1.html*

Cappucci, P. L. (1993). *Realtà del Virtuale. Rappresentazioni tecnologiche, comunicazione, arte*, Clueb, Bologna

Conti, G. (2002). *A VR System for the Early Stages of the Design Process in Architecture*. Tesi di Dottorato, Glasgow (UK)

Cruz-Neira, C. (1993). *Virtual Reality Overview*. SIGGRAPH'93 (pp. 1.1-1.18)

DataGlove. (2004). *L.R.P. Force Feedback Data Glove*. Informazioni reperite il 16 Marzo 2004 dal World Wide Web*: http://www.caip.rutgers.edu/~bouzit/lrp/Video/Glove.mpg*

Dubois, E. and Nigay, L. et al. (1999). *Classification Space for augmented Surgery, an Augmented Reality Case Study*. In Proceedings of Interact'99, IOS Press, Edinburgh (UK) (pp. 353-359)

Essential Reality (2004). *Pagina ufficiale del dispositivo di tracking P5Glove*. Informazioni reperite il 20 Gennaio 2004 dal World Wide Web: *http://www.essentialreality.com/index.asp*

Fabio, B. (2001) *Tecniche innovative di interazione uomo-macchina nei sistemi di progettazione assistita dalla Realtà Virtuale*. Tesi di Laurea, Università degli studi della Calabria

Fastrak. (2004). *Polhemus Fastrak, Virtual Reality Tracking system*. Informazioni reperite il 16 Marzo 2004 dal World Wide Web: http://www.cyberminduk.com/tracking/fastrak.htm

Feiner, S. K. (1996). *Adding I sistemi AR applicati alla diagnostica medica e come ausilio in chirurgia nsight Through Animation In Augmented Reality*. Reperito il 10 Febbraio 2004 dal World Wide Web: *http://csdl.computer.org/comp/proceedings/ca/1996/7588/00/75880014abs.htm*

Fiorentino, M. (2001). *Manipolazione interattiva di curve e superfici in ambiente virtuale*. Reperito il 23 Gennaio 2004 dal World Wide Web: *http://climeg.poliba.it/~disegno/vr3lab/publications/2001/Abstract_Lecce2001.pdf*

Fuchs, H., Bishop, G. et al. (1992) *Research Directions in Virtual Environments*. NFS Invitational Workshop, Univ. North Carolina. Reperito il 13 Gennaio 2004 dal World Wide Web: *ftp://ftp.cs.unc.edu./pub/technical-reports/92-027.ps.Z/*

Garzanti Linguistica (2003). *Definizione del lemma "ossimoro"*. Reperito il 9 Febbraio 2004 dal World Wide Web: *http://www.garzantilinguistica.it*

Gigante, M. (1993). Virtual Reality: Definitions, History and Applications. *Virtual Reality Systems*, Academic-Press. Reperito il 10 Febbraio 2004 dal World Wide Web: *http://citeseer.nj.nec.com/290271.html*

Glasstron Sony Corp (2004). *Sony Glasstron LDI-D100Be*. Informazioni reperite il 26 Febbraio 2004 dal World Wide Web: *http://cgsd.com/OldSonyHMD/*

Goethe, J. W. (1808). *Faust*. Opera teatrale, prima pubblicazione 1808. Traduzione inglese di Luke, D. Pubblicato da Oxford University Press, 1998.

Guvem, S., Feiner, S. (2002) *Authoring 3D Hypermedia for Wearable Augmented and Virtual Reality*. Reperito il 25 Febbraio 2004 dal World Wide Web: *http://www.cs.columbia.edu/graphics/people/sinem/papers/Authoring-ISWC2003/183_Guven_S.pdf*

Hauptmann, A. (1989). *Speech and Gestures for Graphic Image Manipulation* Proc. CHI'89, (pp. 241-245). Reperito il 10 Febbraio 2004 dal World Wide Web: *http://citeseer.nj.nec.com/hauptmann89speech.html*

InertiaCube2. (2004). *InterSense InertiaCube2*. Informazioni reperite il 16 Marzo 2004 dal World Wide Web: *http://www.isense.com/products/prec/ic2/index.htm*

Jargon (1995). *Jargon Dictionary*. Reperito il 12 Gennaio 2004 dal World Wide Web: *http://www.fwi.uva.nl/~mes/jargon/*

Kato, H., Billinghurst, M., Poupyrev, I., Imamoto, K. Tachibana, K. (2000). *Virtual Object Manipulation on a Table-Top AR Environment*. Proc. IEEE and ACM International Symposium on Augmented Reality.

Kemble, K. A. (2001). *An Introduction to Speech Recognition*. Informazioni reperite il 22 Febbraio 2004 dal World Wide Web: *http://www.voicexmlreview.org/Mar2001/features/recognition.html*

Kindratenko, V. (1999). *Electromagnetic Tracker Calibration.* National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign. Reperito il 13 Febbraio 2004 dal World Wide Web: *http://archive.ncsa.uiuc.edu/VEG/VPS/emtc/*

Manetta, C., Blade, R. (1995). Glossary of Virtual Reality Terminology *International Journal of Virtual Reality*, Vol.1 Nr.2

Marx, Z. (2002). *Interaction Elements in Studierstube. Tesi di Laurea.* Reperito il 21 Febbraio 2004 dal World Wide Web: *http://www.studierstube.org/media/documents/MarxThesis.pdf*

Mazuryk, Gervautz (1996). *Virtual Reality History, Applications, Technology and Future.* Reperito il 10 Gennaio 2004 dal World Wide Web: *http://citeseer.nj.nec.com/290271.html/*

Negroponte, N. (1993). *Virtual Reality: Oxymoron or Pleonasm?* Wired, December (1) 6. Reperito il 13 Gennaio 2004 dal World Wide Web: *http://www.wired.com/wired/archive/1.06/negroponte.html?pg=1/*

Negroponte, N. (1996). *Being digital.* London: Coronet Books.

Sachs, E., Roberts, A., Stoops, D. (1991). *3-Draw: A Tool for Designing 3D Shapes*. IEEE Computer Graphics & Applications, Vol. 11, No. 6, pp. 18-26.

SAPI (2004). *Speech SDK 5.1 for Windows® applications*. Libreria Software Development Kit per l'implementazione di sistemi di riconoscimento vocale. Informazioni e software reperite il 10 Marzo 2004 dal World Wide Web: *http://www.microsoft.com/speech/download/sdk51/*

Schweber (1995). *Virtually Here.* PC Magazine - March 14, pp. 168-198.

Shmalstieg, D. (2001). *Collaborative Augmented Reality*. Paper for Habilitation thesis. Reperito il 9 Marzo 2004 dal World Wide Web: *http://www.ims.tuwien.ac.at/publication_detail.php?ims_id=16*

SigGraph. (2000). *MagicBook: Project Description*. Immagine reperite il 16 Marzo 2004 dal World Wide Web: *http://www.hitl.washington.edu/magicbook/description.html*

Sonnati, F. (2002). *I vantaggi della "Augmented Reality"*. Reperito il 14 Gennaio 2004 dal World Wide Web: *http://www.lithium.it/dream0013p1.asp*

Sony. (2004). *Sony HMD GlassTron*. Informazioni reperite il 16 Marzo 2004 dal World Wide Web: *http://www.sony.com/*

Splechtna, R. (2003). *Comprehensive Calibration Procedures for Augmented Reality*. Reperito il 19 gennaio 2004 dal World Wide Web: *http://www.ims.tuwien.ac.at/publication_detail.php?ims_id=109*

Studierstube (2004). *Pagina ufficiale del progetto di Augmented Reality Studierstube.* Informazioni reperite il 22 Febbraio 2004 dal World Wide Web: *http://www.studierstube.org/*

Sturman, D., Zeltzer, D., Pieper, S. (1989). *Hands-On Interaction with Virtual Environments.* UIST'89 pp. 19-24.

Sutherland, I. (1968). *A Head-Mounted Three Dimensional Display*. Fall Joint Computer Conference, AFIPS Conference Proceedings, No. 33, pp. 757-764

Szalavári, Z., Gervautz M. (1997). *The personal Interaction Panel – a Two-Handled Interface for Augmented Reality.* EuroGraphics '97.

Szalavári, Z., Schmalstieg D., Fuhrmann, A., Gervautz, M. (1998). *Studierstube - An Environment for Collaboration in Augmented Reality*. Virtual Reality, 3(1):37-48.

Visiontrak. (2004). *Real Time eye/head tracking*. Informazioni reperite il 16 Marzo 2004 dal World Wide Web: *http://www.polhemus.com/eyetracking.htm*

W3C. (2004). *World Wide Web Consortium.* Organizzazione promotrice dell'evoluzione del World Wide Web tramite lo sviluppo di protocolli comuni. Informazioni reperite il 13 Marzo 2004 dal World Wide Web: *http://www.w3.org/*

Weimer, D., Ganapathy, S. K. (1989). *A Synthetic Visual Environment with Hand Gesturing and Voice Input.* Proc. ACM CHI'89 Conference on Human Factors in Computing Systems, pp. 235-240.

XML. (2204). *Extensible Markup Language.* Scripting derivato dal SGML originariamente sviluppato come supporto al publishing elettronico su larga scala. Informazioni reperite il 13 Marzo 2004 dal World Wide Web: *http://www.w3.org/XML/#intro*

Zimmerman, T., Lanier, J. et al. (1887). *A Hand Gesture Interface Device.* Proc. of ACM CHI+GI'87 Conference on Human Factors in Computing Systems and Graphics Interface, pp. 189-192.