

Università degli Studi di Salerno
Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea in Informatica

Tesi di Laurea Magistrale

Esplorazione di grafi in 3D mediante interfacce naturali utente: Realizzazione e usabilità

Relatori:
Prof. Vittorio Scarano
Prof. Ugo Erra
Prof.ssa Delfina Malandrino

Candidato:
Luca Pepe
Matr. 0522500171

Anno Accademico 2013-2014

Abstract

Visualizzare una rete è un modo per carpire informazioni su ciò che essa rappresenta ed è un processo di supporto all'analisi di fenomeni. I software di visualizzazione di reti tipicamente offrono rappresentazioni delle reti poco navigabili, poco interattive e che soffrono di un eccesso di informazioni quando la rete è di grandi dimensioni dovuto al poco spazio a disposizione fornito dalle sole due dimensioni utilizzate per la rappresentazione.

Per risolvere queste problematiche in questo lavoro di tesi è presentato *Graph 3D Explorer*: un plugin per *Gephi* che permette la navigazione e la manipolazione di una rete in un ambiente virtuale tridimensionale e interattivo. *Gephi* è un software scritto in *Java* molto utilizzato in contesti di analisi delle reti sociali e altamente espandibile grazie al fatto che si basa sulla *Netbeans Platform*. *Graph 3D Explorer* sfrutta questa espandibilità offerta da *Gephi* e dalla *Netbeans Platform* per offrire l'esplorazione di grafi in 3D come componente aggiuntivo.

Graph 3D Explorer usa le API OpenGL e la potenza delle schede grafiche attuali per visualizzare e far navigare in 3D reti di grandi dimensioni. Inoltre, offre il supporto a diverse periferiche di input che vanno dalla classica coppia mouse-tastiera, al più interattivo joypad fino ad arrivare alla periferica Leap Motion che permette di interagire con il sistema tramite dei semplici gesti riconducibili a tipiche azioni che si fanno nel mondo reale. Infine, *Graph 3D Explorer* permette all'utente di essere immerso nella visualizzazione grazie alla compatibilità con la periferica Oculus Rift.

Graph 3D Explorer è stato sviluppato e successivamente validato con un campione di 60 studenti dell'Università di Salerno che ne hanno confermato le qualità e l'innovatività.

Indice

1 Introduzione **1**
- 1.1 Contesto generale . . . 1
  - 1.1.1 Il Grafo: un unico modello per tanti fenomeni . . . 1
  - 1.1.2 L'importanza della rappresentazione visiva . . . 2
  - 1.1.3 Paradigmi di Information Retrieval . . . 4
- 1.2 Motivazioni del lavoro . . . 4
- 1.3 Obiettivi . . . 6
- 1.4 Struttura della tesi . . . 7

2 Interfacce Naturali Utente **8**
- 2.1 Interfacce uomo-macchina: Un po' di storia . . . 8
- 2.2 I gesti e la voce: una naturale evoluzione per le interfacce uomo-macchina . . . 9
- 2.3 I vantaggi e limiti delle Interfacce Naturali Utente . . . 10
- 2.4 Tipologie di dispositivi . . . 12
- 2.5 Ambiti di applicazione . . . 14
- 2.6 La progettazione delle interazioni gestuali . . . 16
  - 2.6.1 Tipologia di gesti . . . 16
  - 2.6.2 Quanti gesti scegliere . . . 18

3 Il Sistema realizzato: descrizione **20**
- 3.1 La scelta di Gephi . . . 20
- 3.2 Gephi: cosa si può migliorare . . . 23
- 3.3 Descrizione generale delle funzionalità . . . 24
- 3.4 Visualizzazione, esplorazione e manipolazione . . . 25
- 3.5 Dispositivi di input e interfacce naturali . . . 28
  - 3.5.1 Mouse e Tastiera . . . 28
  - 3.5.2 Joypad . . . 30
  - 3.5.3 Il sistema di comandi gestuali . . . 31

3.5.4 La modalità immersiva 39
3.6 Integrazione con Gephi 40

4 Il Sistema realizzato: implementazione 43
4.1 Architettura del sistema 43
4.2 Il Modello: Grafo e Osservatore 44
4.3 La GUI Java e la Netbeans Platform 46
4.4 Il livello di coordinamento 47
4.5 Il Rendering: OpenGL e JOGL 49
4.6 Il supporto a Mouse e Tastiera 52
4.7 Il supporto al Joypad 52
4.8 Il supporto alla periferica Leap Motion 54
4.8.1 Una panoramica sulle API 54
4.8.2 Integrazione della periferica Leap Motion nel sistema sviluppato 58
4.9 La visita del grafo, la visualizzazione delle informazioni e l'area di selezione 63
4.9.1 I test sui nodi 63
4.10 L'integrazione dell'Oculus Rift 66

5 Il Sistema realizzato: sperimentazione 68
5.1 Obiettivi dello studio 68
5.2 Reclutamento degli utenti e configurazioni testate 69
5.3 Procedure utilizzate 70
5.4 Configurazione tradizionale contro realtà virtuale 73
5.4.1 Questionario generale 73
5.4.2 I tre task 78
5.4.3 Risultati del Tam 81
5.5 Tutte e sei le configurazioni 84
5.5.1 Dati generali del campione 84
5.5.2 Risultati del TAM 86

6 Conclusioni e sviluppi futuri 87

Bibliografia 92

Capitolo 1

Introduzione

In questo capitolo sarà data una panoramica di quello che è il contesto in cui si colloca il lavoro di tesi e, successivamente, saranno discusse le motivazioni e gli obiettivi. Infine sarà descritta la struttura generale di tutto il documento.

1.1 Contesto generale

1.1.1 Il Grafo: un unico modello per tanti fenomeni

Reti sociali, architetture software e in generale molti sistemi complessi, possono essere rappresentati come un insieme di entità tra cui intercorrono delle relazioni utilizzando una struttura di tipo grafo. Informalmente, un grafo è una struttura matematica composta da nodi e archi. I nodi costituiscono le entità del sistema che si vuole analizzare mentre gli archi rappresentano le relazioni che intercorrono tra di esse.

Un grafo può modellare fenomeni di natura diversa e la teoria che è alla base ha una lunga storia. L'ideatore del concetto di grafo e di gran parte dei concetti alla base della teoria dei grafi fu il matematico Eulero che, nel 1700, affrontò *il problema dei sette ponti di Königsberg* in cui si chiedeva se fosse possibile fare una passeggiata che, attraversando ogni ponte della città di *Königsberg* esattamente una sola volta, facesse tornare al punto di partenza.

Nel corso degli anni, la teoria dei grafi ha acquisito una importanza sempre maggiore e, attualmente, rappresenta uno dei campi di studio della matematica e dell'informatica di maggior interesse. Infatti, con la forte diffusione di Social Network quali Facebook e Twitter si è avuta una grande affermazione della *Social Network Analysis*.

La *Social Network Analysis* è una metodologia di analisi delle reti sociali applicabile in vari contesti e dal carattere molto interdisciplinare. Infatti, essa si fonda su concetti della sociologia e della teoria dei grafi ed è alla base di studi di marketing, advertisement basato sulle abitutini degli utenti, studi sulla diffusione di malattie, sistemi di raccomandazione e tanto altro.

La teoria dei grafi e lo studio delle reti sociali trovano applicazione in molti altri ambiti. Ad esempio, è stato studiato che le reti sociali e le reti di overlay nei sistemi P2P presentano delle analogie in termini di distribuzione dei gradi e di lunghezza media di cammino e sono raggruppabili nelle categorie *Smallworld*[1] e *Scale Free*[2]. Ulteriormente, il concetto di *Triadic Closure*[3] teorizzato da George Simmel e alla base della *Social Network Analysis* è stato usato in contesti di *Software Engineering* per l'identificazione automatica di *antipattern* nei sistemi software[4]. Ancora, la teoria dei grafi trova applicazioni anche nel campo dell'intelligenza artificiale e delle reti neurali[5].

Ovviamente, questa serie di esempi è solo una piccola dimostrazione della grande versatilità e potenza che la teoria dei grafi possiede e del fatto che essa costituisce la base di gran parte dell'informatica moderna.

1.1.2 L'importanza della rappresentazione visiva

Per chiarificare quanto detto finora, vogliamo fare adesso un esempio pratico di analisi di una rete:

Supponiamo di voler capire quali siano gli articoli scientifici più importanti dal 2000 fino ad oggi. Un approccio che potremmo seguire potrebbe consistere nel prendere i dati relativi alle pubblicazioni da delle fonti fidate come *Google Scholar* oppure *DBLP* e costruire un grafo in cui i nodi sono le pubblicazioni scientifiche e gli archi rappresentano le citazioni. Costruito il grafo, potremmo classificare come più importanti le pubblicazioni che corrispondono ai nodi che hanno il più alto numero di archi incidenti. Il numero di archi incidenti su un nodo è detto grado e la metrica appena descritta è nota come *Degree Centrality*.

L'esempio appena presentato è una semplice applicazione della teoria dei grafi e della *Social Network Analysis*. Tuttavia, qual è il passo successivo ad una analisi di questo tipo? comunicare i risultati agli altri. A tal scopo, nasce l'esigenza di avere una rappresentazione visuale delle informazioni che ne favorisca la comprensione.

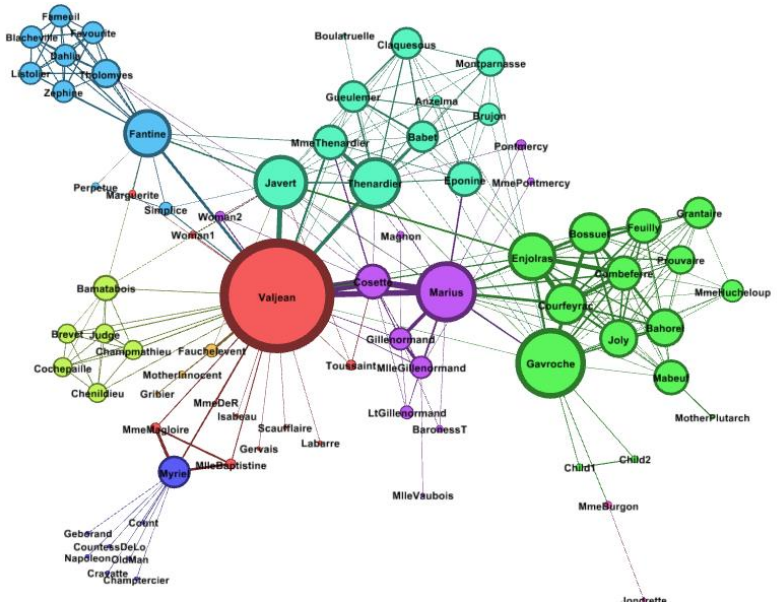
E' importante notare che una rappresentazione visuale non è solo un modo efficace per far comprendere risultati ottenuti con metodi analitici. Infatti, *Jin Zhang*, nel suo libro *Visualization for Information Retrieval*[6],

definisce una visualizzazione come "un processo di trasformazione di dati astratti in una rappresentazione grafica con lo scopo di supportare l'analisi di fenomeni". Infatti, quando non si sa cosa cercare, una rappresentazione visuale efficace può aiutare facendo intuire dei fatti che possono essere successivamente validati tramite metodi numerici.

Tuttavia, trovare la giusta rappresentazione visuale per un insieme di dati non è facile. Infatti, una rappresentazione potrebbe essere più o meno intuitiva a seconda della esperienza della persona che la sta consultando. Inoltre, essa potrebbe soffrire di *Information Overloading*[7] e quindi risultare troppo complessa e ricca di informazioni al punto tale da indurre confusione invece di favorire la comprensione.

Considerando la tipica rappresentazione visuale dei grafi (Figura 1.1), il problema di efficacia della rappresentazione, si traduce nell'associare ai nodi una posizione nello spazio tale da non incorrere nel problema dell'*Information Overloading*[7]. Per far ciò, vi è la necessita di algoritmi appositi che rientrano in una branca della matematica e dell'informatica che prende il nome di *Graph Drawing*[8].

Figura 1.1: Una tipica rappresentazione visiva di un grafo



1.1.3 Paradigmi di Information Retrieval

Una visualizzazione efficace favorisce quello che in inglese viene chiamato *Information Retrieval*. Cioè il "processo di ricerca, esplorazione e scoperta delle informazioni contenute in una collezione organizzata di dati al fine di soddisfare il bisogno di informazioni dell'utente" [6]. Ci sono due paradigmi principali per l'*Information Retrieval*: Esplorazione (*Browsing*) e Interrogazione (*Query*). Entrambi i due paradigmi hanno punti di forza e debolezze e, di fatto, sono complementari.

Con il termine *Query* ci si riferisce al reperimento di informazioni tramite l'utilizzo di parole chiave che possono essere combinate in maniera più o meno complessa tramite dei connettivi logici. Un paradigma di questo tipo è comune e lo ritroviamo spesso durante la navigazione web interagendo con i motori di ricerca tipo *Google*.

Invece, al termine *Browsing* è associato il processo di esplorazione libera dell'informazione. Un esempio è la navigazione sul web passando di pagina in pagina sfruttando i collegamenti.

Entrambi i paradigmi sono essenziali. Infatti, il *Browsing* è efficace quando si ha una idea vaga di cosa si sta cercando. Mentre le *Query* sono adatte a ricerche mirate. Proprio perché il *Browsing* avviene principalmente quando si ha una idea vaga, risulta fondamentale l'interattività della visualizzazione che deve stimolare l'utente e invogliarlo all'esplorazione.

1.2 Motivazioni del lavoro

La maggioranza dei software di visualizzazione di reti permette di inferire informazioni in maniera sia visuale che analitica. Tuttavia, con la diffusione dei *Social Network*, le analisi visuali vengono fatte spesso su grafi grandi. Questo comporta il problema dell'*Information Overload* in quanto, comunemente, la rete è visualizzata in due dimensioni.

Ulteriormente, la rappresentazione bidimensionale della rete rende possibili solo azioni di tipo *Pan* e *Zoom* sui dati. Ciò limita molto la possibilità di fare *Browsing* e rende più difficile il processo di *Information Retrieval* quando non si ha un obiettivo chiaro da cercare.

Queste problematiche presenti nei software di visualizzazione di reti portano alla necessità di rappresentare i grafi in tre dimensioni. Infatti, vari studi[9] confermano che, con una visualizzazione tridimensionale dei dati, si ha più spazio a disposizione e di conseguenza si ha la possibilità di distribuire meglio le informazioni diminuendo le occlusioni. Inoltre, l'utilizzo delle

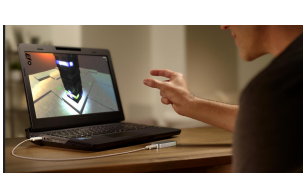
tre dimensioni permette maggiori possibilità di esplorazione dell'insieme dei dati perché permette l'osservazione delle relazioni da più punti di vista.

Come già detto in precedenza, per favorire il *Browsing* è necessario un sistema fortemente interattivo. Un modo per incrementare l'interattività è dato dalla *Gamification* che consiste nell' "introduzione di elementi propri del mondo dei videogiochi in un sistema sofware che non è un videogioco" [10].

A tal proposito, un aspetto dei videogiochi che può essere introdotto con successo all'interno di una applicazione di visualizzazione dati è l'uso di periferiche di input e output che si discostano dalla solita coppia *Mouse - Tastiera* e *Monitor*. Attualmente, periferiche di *Hand Tracking* come *Leap Motion* permettono una interazione innovativa in cui l'utente può controllare il sistema con il solo ausilio delle mani (Figura 1.2(a)). Questo tipo di periferica e le tipologie di interazioni che essa permette rientrano nel campo di ricerca delle interfacce naturali utente, meglio note come *Natural User Interfaces*.

Un altro aspetto che sta avendo una forte diffusione e sta suscitando un grande interesse generale è la realtà virtuale. In particolare, c'è un forte sviluppo sofware per la periferica *Oculus Rift*. L'*Oculus Rift* è uno schermo da indossare sul viso (*Head Mounted Display*) che permette di tenere traccia dei movimenti della testa e di avere una visualizzazione in 3D immersiva (Figura 1.2(b)).

Combinare la realtà virtuale data da *Oculus Rift* e le interazioni gestuali rese disponibili dalla *Leap Motion* permette di avere una esperienza immersiva coinvolgente ed efficace che porta ad un forte aumento dell'interattività del sistema.



(a) Leap Motion



(b) Oculus rift

Figura 1.2: Dispositivi per le interfacce naturali utente e la realtà virtuale

1.3 Obiettivi

La necessità di migliorare il processo di *Information Retrieval* ed aumentare il livello di interattività nelle applicazioni di visualizzazione dati che trattano grafi ha portato a quelli che sono i due obiettivi di questo lavoro di tesi:

- Fornire uno strumento per l'analisi delle reti che offrisse la possibilità concreta di esplorare una rete in modo da potenziare il processo di *Information Retrieval*.
- Studiare l'efficacia delle *Natural User Interfaces* e della realtà virtuale in contesti di visualizzazione di grafi ed in particolare l'impatto in termini di divertimento, coinvolgimento e livello di interattività in confronto a combinazioni standard come Mouse e Tastiera.

Per raggiungere questi obiettivi è stata condotta una attività di ricerca. Tra i vari strumenti software per l'analisi delle reti, si scelto di prendere in considerazione *Gephi*[11] che è un software multipiattaforma molto utilizzato in *Social* e *Biological Network Analysis* ed è stato tra l'altro tra i progetti partecipanti al *Google Summer Of Code* dal 2009 al 2012.

Gephi è stato espanso in termini di funzionalità andando a sviluppare *Graph 3D Explorer*: un plugin che permette una navigazione interattiva in 3D della rete e che si affianca alla normale visualizzazione già offerta dando, quindi, quella complementarietà desiderata tra l'approccio basato su *Query* e la necessità di fare *Browsing*.

Ulteriormente, in *Graph 3D Explorer* sono state previste diverse modalità di utilizzo. Infatti è possibile navigare la rete usando *Mouse e Tastiera*, Joypad oppure *Leap Motion* usando esclusivamente interazioni gestuali. Inoltre, *Graph 3D Explorer* supporta la periferica *Oculus Rift* per la realtà virtuale offrendo così sei configurazioni diverse di utilizzo.

Per verificare l'efficacia delle interfacce naturali e della realtà virtuale è stata condotta una sperimentazione con 60 studenti che ha evidenziato un buon livello di accettazione del sistema generale e un buon grado di accettazione per le periferiche di input output innovative. In particolare, le configurazioni con l'*Oculus Rift* e il *Leap Motion* sono risultate più coinvolgenti e divertenti rispetto ai dispositivi tradizionali. Una descrizione approfondita della sperimentazione condotta sarà data nel capitolo 5.

1.4 Struttura della tesi

Nel capitolo 2 sarà fatto un approfondimento sulle interfacce naturali e sugli aspetti che bisogna tenere in considerazione quando si progetta un software che fa uso di interazioni gestuali.

Nel capitolo 3 sarà chiarita la scelta di *Gephi* come base di partenza e poi sarà presentato il sistema sviluppato mentre nel capitolo 4 saranno dati i dettagli tecnici relativi all'implementazione. Nel capitolo 5 sarà discussa la sperimentazione condotta ed i risultati ottenuti. Infine nel capitolo 6 saranno discusse le conclusioni e quelli che possono essere gli eventuali sviluppi futuri.

Capitolo 2

Interfacce Naturali Utente

In questo capitolo sarà fatta una panoramica sulle *Interfacce Naturali Utente* e sui loro possibili campi di applicazione. Successivamente, saranno prese in considerazione le problematiche da affrontare quando si progetta un software che fa uso di interazioni gestuali.

2.1 Interfacce uomo-macchina: Un po' di storia

Le modalità con cui l'utente interagisce con il computer hanno subito forti variazioni nel corso degli anni. In principio (1950-1960), c'erano i sistemi *Batch* in cui le sequenze di operazioni da fare venivano preparate, memorizzate su supporti compatibili con l'elaboratore, e poi mandate in esecuzione come un blocco unico senza possibilità di interazione da parte dell'operatore umano.

Successivamente (1960 -1980), ci fu l'era dei *Mainframe* e dei primi terminali con display alfanumerico. In questo periodo nacque l'interfaccia a linea di comando (*Command Line Interface (CLI)*) con cui l'utente, per la prima volta, poteva interagire con il computer scrivendo comandi testuali.

Poi, verso la fine degli anni settanta, nei laboratori di *Xerox Parc* fu realizzata la prima interfaccia grafica basata su finestre e icone utilizzabile tramite *Mouse* nota come *WIMP GUI* dalla descrizione inglese "Graphical User Interfaces based on Windows, Icons, Menus, and a Pointing device" [12].

Il paradigma *WIMP* è quello che oggi tutti conosciamo e rappresenta la maniera con cui, quotidianamente, ci interfacciamo con i computer. Tuttavia, attualmente, con la forte diffusione di dispositivi *Mobile* quali *Smartphone* e *Tablet*, l'utilizzo diretto delle dita sostituisce il *Mouse* in varie occasioni.

2.2 I gesti e la voce: una naturale evoluzione per le interfacce uomo-macchina

Secondo McMahan et al.[13], Le Interfacce Naturali Utente (*Natural User Interfaces (NUI)*) possono essere definite come un paradigma di interazione uomo-macchina in cui l'utente interagisce con il sistema usando metodologie a lui naturali e riconducibili al mondo reale.

Interagire con i computer senza usare dispositivi fisici quali *Mouse* e *Tastiera* è stato un sogno a lungo perseguito. Infatti, interfacce uomo-macchina basate interamente su gesti e voce sono state teorizzate in articoli scientifici[12] e in varie produzioni cinematografiche di carattere fantascientifico (Figura 2.1). Basta pensare un attimo a serie televisive come *Star Trek* (1970), o film come *Matrix* (1999) e *Minority Report* (2002) per capire quanto queste metodologie di interazione siano sempre state nell'immaginario e nel desiderio comune.

Oggi, con la diffusione di dispositivi a basso costo per il tracciamento dei movimenti, quali *Leap Motion* e *Kinect*, e della pervasività del riconoscimento vocale grazie al lavoro di aziende come *Google*, *Apple* e *Microsoft* le *NUI* non sono più fantascienza ma realtà.

Figura 2.1: Una interazione gestuale tratta dal film Minority Report (2002)



2.3 I vantaggi e limiti delle Interfacce Naturali Utente

Le interfacce tradizionali costringono l'utente ad apprendere dei meccanismi e dei modi di operare che non gli sono naturali. Inoltre, spesso, azioni uguali richiedono meccanismi diversi che variano da programma a programma.

Consideriamo uno scenario comune a molte applicazioni: selezionare e spostare un oggetto all'interno di un ambiente virtuale utilizzando *Mouse* e *Tastiera*. Nella maggioranza dei casi dovremmo compiere le seguenti azioni:

1. Afferrare il mouse.
2. Muovere il mouse per spostare il cursore di selezione sull'oggetto.
3. Premere un pulsante per far sì che la selezione avvenga.
4. Muovere il mouse e/o premere degli opportuni tasti per spostare l'oggetto.

Questa sequenza di passaggi contiene una serie di azioni che non hanno nessun corrispettivo nell'ambiente reale e che richiedono un apprendimento. Ulteriormente, i tasti necessari per effettuare l'azione di selezione potrebbero variare al variare dell'applicazione e quindi indurre confusione in un utente che utilizza software diversi ma che hanno questa tipologia di scenario in comune.

Un ulteriore problema della sequenza di azioni descritta è che sono richieste azioni diverse a seconda di come si vuole muovere l'oggetto. Infatti, il mouse è un dispositivo che si muove su una superficie bidimensionale e di conseguenza l'oggetto può spostarsi solo su due dimensioni. Per spostare l'oggetto in tutte e tre le dimensioni, oltre allo spostamento del mouse, vi è necessità di un trigger aggiuntivo che permetta di associare il movimento al terzo asse spaziale. Lo stesso discorso vale per i sistemi *touchscreen* in quanto anche lo schermo è una superficie bidimensionale. Ulteriormente, anche le azioni tipiche di *press* e *long-press* dei sistemi *touchscreen* non hanno corrispettivi nella realtà e richiedono uno sforzo cognitivo per essere appresi.

In scenari come quello descritto le *NUI* esprimono la loro potenza, in quanto, rifacendosi a meccaniche già note all'utente, portano ad un abbassamento dei tempi di apprendimento e di conseguenza ad una maggiore usabilità. Infatti, per spostare un oggetto nel mondo reale, quello che si fa è semplicemente afferrarlo, muoverlo e successivamente rilasciarlo. Utilizzando dei dispositivi di tracking quali *Leap Motion* o *Kinect* questa metodologia naturale è direttamente portabile nel mondo virtuale.

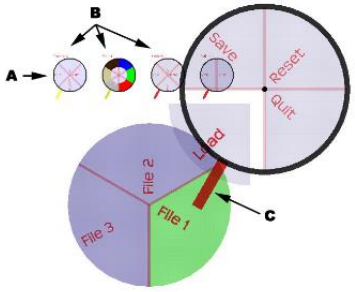
Oltre ad una minore curva di apprendimento iniziale, l'utilizzo di meccanismi che siano naturali all'utente rende standard le azioni. Infatti, se due programmi differenti che fanno uso di *NUI* hanno in comune lo scenario di selezione, allora è probabile che l'azione si faccia allo stesso modo. Una uniformità nelle metodologie d'interazione ridurrebbe la confusione per l'utente nell'utilizzo di applicazioni diverse.

Tuttavia, come evidenziato da *Boussemart et al.*[14], non tutte le azioni che sono possibili in un ambiente virtuale hanno un corrispettivo nel mondo reale. Basti pensare, ad esempio, all'azione di *Zoom* su un insieme di dati o su una regione di spazio oppure al processo di scelta di una determinata azione da applicare ad un oggetto selezionato. Quindi, analogamente alle interfacce tradizionali che fanno riferimento a metafore come quelle del *Desktop*[15] (o della *Home* nel *mobile*), anche per le *NUI* si ha la necessità di metafore di interazione.

L'introduzione delle metafore comporta uno sforzo per l'utente in quanto deve capirne il significato e il funzionamento. Inoltre, la realizzazione della metafora potrebbe avere diverse interpretazioni in software diversi riportando al problemi di uniformità delle soluzioni con *Mouse* e *Tastiera*.

Un esempio di metafora per le interfacce naturali utente è il menu *Pieglass*[14] che propone una interfaccia utilizzabile tramite gesti per scegliere quale azione eseguire su un oggetto presente in un ambiente virtuale. In Figura 2.2 è riportato lo schema e un esempio di interazione con il menu *Pieglass* in cui si vede l'utente scegliere, tramite un menu circolare, quale azione tra le tante possibili (*Salva*, *Modifica*, *Cancella*, *Copia*..) debba essere applicata ad un file.

Figura 2.2: Il menu di selezione pieglass proposto da *Boussemart et al.*[14]



(a) Schema



(b) Esempio di utilizzo

2.4 Tipologie di dispositivi

Un sistema che fa uso delle *NUI* ha bisogno di dispositivi di tracking per il riconoscimento dei movimenti. Essi secondo *Shiratuddin et al.*[16], sono raggruppabili in due categorie: *marker-based* e *marker-less*. I dispositivi *marker-based* richiedono delle componenti sensoristiche che vanno indossate dall'utente per far sì che i movimenti possano essere tracciati. Invece, quelli *marker-less* tracciano le azioni dell'utente utilizzando camere per l'acquisizione delle immagini e tecniche di *Computer Vision* per la loro analisi.

Generalmente, i dispositivi *marker-based* sono più precisi ma anche più invasivi a causa della componente da indossare. D'altra parte, i dispositivi *marker-less* sono meno precisi rispetto ai dispositivi *marker-based* ma generalmente più accettati dall'utenza in quanto meno invasivi. Come avviene in molti contesti della tecnologia la scelta tra il giusto dispositivo da usare dipende dalle necessità.

Un esempio di dispositivo *marker based* è il sistema di *Motion Capture Vicon* (Figura 2.3) che, sfruttando una mappa dello scheletro creata tramite dei sensori indossati dall'utente, permette una tracciatura fedele del movimento. Sistemi di *Motion Capture* di questo tipo non hanno applicazione in

ambito di Interfacce Naturali Utente in quanto altamente invasivi. Tuttavia, sono molto utilizzati durante lo sviluppo di videogiochi e nella produzione cinematografica perché permettono di creare delle animazioni estremamente realistiche.

Dispositivi come *Leap Motion* (Figura 2.6(a)) e *Kinect* (Figura 2.6(b)) rientrano nella categoria *marker-less* e sono nati come periferiche da gioco analogamente alla *Wiimote* di Nintendo (Figura 2.5(a)) e alla *Playstation Move* di *Sony* (Figura 2.5(b)) che però rientrano nella categoria *marker-based* essendo dei controller muniti di accelerometri che necessitano di essere afferrati dall'utente.

Un ulteriore esempio di dispositivo di tracking è dato dall'*Oculus Rift* che è uno schermo da indossare sul viso (*Head Mounted Display*) che permette di avere una visualizzazione in 3D immersiva e, tramite degli appositi sensori, di tenere traccia dei movimenti della testa. Altri esempi di dispositivi di tracking per i movimenti della testa sono il *Google Cardboard* e il *Samsung Gear VR* che sfruttano lo stesso principio dell'*Oculus Rift* ma, invece di avere uno schermo dedicato e dei sensori interni, permettono di utilizzare uno smartphone sia per il tracking che per la visualizzazione portando così ad una riduzione del numero di cavi necessari e ad una maggiore libertà di movimento.

Non tutti i dispositivi hanno gli stessi obiettivi di tracking. Ad esempio, la *Kinect* permette di tracciare i movimenti del corpo ma non ha un elevato livello di dettaglio sulle singole mani. Di contro, la *Leap Motion* permette un tracking specifico per le mani. Ancora, Oculus Rift e affini permettono di tracciare solo la testa. Questo ci fa concludere che, per avere un tracciamento completo dei movimenti del corpo, si debbano usare più sensori insieme. Ad esempio combinando *Kinect*, *Oculus Rift* e *Leap Motion* si può avere un sistema di tracking dei movimenti completo.

Figura 2.3: Il sistema di *Motion Capture* Vicon



Figura 2.4: Alcuni dispositivi di tracking *marker-based* destinati al mercato consumer



(a) *Wiimote*: Il dispositivo *marker-based* della Nintendo per la console *Nintendo Wii*



(b) *Playstation Move*: Il dispositivo *marker-based* della Sony per *Playstation 3*

Figura 2.5: Alcuni dispositivi di tracking *marker-less*



(a) Il dispositivo di *Hand Tracking* Leap Motion



(b) Il sensore di movimento Kinect di *Microsoft*

2.5 Ambiti di applicazione

Le interfacce naturali utente hanno avuto una esplosione soprattutto in ambito videoludico. Infatti, periferiche come la *Kinect* o la *Wiimote* nascono proprio come accessori per console da gioco.

Anche periferiche come *Oculus Rift* e *Leap Motion* stanno avendo la loro principale applicazione nel settore dei videogiochi grazie ad una sempre crescente comunità di sviluppatori.

Però, dato il forte interesse dell'utenza nei confronti di queste tecnologie e dei vantaggi correlati a queste tipologie di interazioni, c'è anche una continua ricerca per utilizzare questi dispositivi e i paradigmi delle *NUI* in contesti differenti da quelli videoludici.

Infatti, l'utilizzo combinato di periferiche *marker-less* poco invasive e dal prezzo accessibile e di interfacce accattivanti unite a meccanismi di *Gamification* [10] ben congeniati basati su trofei ed obiettivi, permette la creazione di sistemi altamente interattivi e stimolanti. Questa tipologia di applicazioni rientra nella categoria dei "Giochi Seri" (*Serious Games*) cioè "giochi che hanno uno scopo che va oltre il puro e semplice divertimento" [17].

In particolare, un campo in cui le *NUI* risultano essere particolarmente efficaci è quello della fisioterapia. Infatti, in un recente studio, *Chang et al.* [18] hanno testato l'efficacia della *Kinect* e delle interfacce basate su gesti in ambito di sistemi di riabilitazione interattivi creando un ambiente virtuale per l'esecuzione guidata di esercizi fisici.

Le *NUI* hanno una applicazione efficace anche nell'interazione con i sistemi robotici. Di fatto, in un recentissimo studio del 2014 [19], la periferica *Leap Motion* e le interazioni gestuali sono state usate per il controllo di un braccio meccanico a scopo di aiutare persone che hanno problemi nell'afferrare oggetti.

Un altro campo di applicazione interessante per le interfacce naturali utente è la domotica [20]. Infatti, l'utilizzo di gesti per controllare i dispositivi elettronici presenti nel proprio appartamento è una idea molto affascinante e da lungo tempo perseguita. Di fatto, i nuovi televisori smart controllabili tramite gesti sono un esempio di come le interfacce naturali utente possono arrivare nelle case delle persone e nella vita di tutti i giorni.

Un ulteriore campo di applicazione per le *NUI* è appunto quello della visualizzazione dati in cui questo stesso lavoro si colloca e nel quale, principalmente, sono state sperimentate interfacce basate su gesti per la consultazione di immagini mediche in sale operatorie dove bisogna preservare l'ambiente sterile. [21] [22]

I campi di applicazione citati finora di certo non costituiscono un elenco esaustivo di quelle che possono essere le applicazioni delle *NUI*. Infatti, le *NUI* sono un campo di ricerca molto influenzato dai fortissimi progressi tecnologici nel campo dei dispositivi di tracking. Di giorno in giorno, le periferiche di tracking sono sempre più performanti e dai costi accessibili e questo può portare soltanto ad ampliare le possibilità di applicazione dei concetti alla base delle *NUI*. Concetti che sono da lungo tempo nella fantasia comune e che diventano sempre più concretamente realizzabili.

2.6 La progettazione delle interazioni gestuali

Presentati i principi e le possibili aree di applicazione delle *NUI*, andiamo ora ad analizzare alcuni aspetti che bisogna tenere in considerazione quando si progetta una applicazione che fa uso di interazioni basate su gesti.

In fase di progettazione si possono identificare due problematiche fondamentali che possono essere denominate come (*i*) *Tipologia*, e (*ii*) *Quantità*. Per *Tipologia* intendiamo il problema di quale gesto associare ad una determinata azione mentre con *Quantità* identifichiamo le questioni inerenti al numero di gesti da utilizzare.

2.6.1 Tipologia di gesti

Inerentemente alla tipologia di interazioni da scegliere, *Nancel et al.*[23] nel 2011 hanno studiato l'applicabilità delle interazioni gestuali in un contesto di *Pan* e *Zoom* di oggetti in un ambiente virtuale visualizzato tramite display *wallsize* (Figura 2.6). Essi hanno messo a confronto interazioni che richiedessero una sola mano (*Unimanual*) e interazioni che richiedessero entrambi le mani (*Bimanual*) ed inoltre hanno comparato gesti di tipo circolari (*Circular*) con gesti di tipo lineare (*Linear*). Un gesto di tipo circolare è un movimento continuo che consiste nel "disegnare" un cerchio nell'aria. Invece, un gesto lineare è, appunto, un movimento lineare che può essere rapportato alla tipica azione *Swipe* in ambienti *touchscreen*.

Lo studio condotto da *Nancel et al.*[23] ha messo in risalto la maggiore efficacia delle interazioni *Bimanual* in contesti di selezione e manipolazione ed, inoltre, ha evidenziato che gesti di tipo circolare risultano di difficile esecuzione a causa della troppa libertà che si ha nello spazio libero e che, quindi, sono meno efficaci di quelli lineari nonostante quest'ultimi richiedano un riposizionamento della mano. Ulteriormente, *Nancel et al.*, hanno stabilito che le interazioni gestuali risultano essere meno adeguate rispetto alle interfacce tradizionali in contesti in cui è richiesta una alta precisione a causa dell'assenza di vincoli sui movimenti possibili.

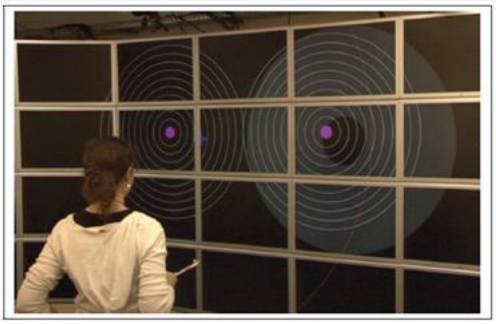


Figura 2.6: L'ambiente virtuale utilizzato da *Nancel et al.* per i loro esperimenti

Un ulteriore aspetto che bisogna tenere in considerazione per la scelta dei gesti da utilizzare è l'affaticamento. Infatti, un sistema che richiede all'utente una interazione gestuale prolungata e senza adeguati intermezzi di riposo può indurre stanchezza e quindi risultare poco piacevole nell'utilizzo portando a quella che viene chiamata "sindrome del Braccio Gorilla" [16]. Questo nome fa riferimento al penzolare del braccio a causa della stanchezza dovuta al lungo tempo che esso è stato teso in avanti facendo piccoli movimenti. La sindrome del Braccio Gorilla è uno dei motivi per cui è bene pensare ad interazioni che non siano eccessivamente prolungate e che garantiscano degli adeguati intervalli di riposo.

La tipologia di gesti da utilizzare dipende anche dai dispositivi di tracking utilizzati e dall'ambiente in cui l'utente si trova quando interagisce con l'applicazione. Infatti, il campo visivo delle periferiche di tracking è limitato e bisogna tenerne conto quando si progettano i gesti utilizzabili. Di fatto questo aspetto è cruciale e lo è ancora di più se si pensa che nelle interazioni gestuali non ci sono vincoli e l'utente è portato a spaziare molto nei movimenti. Un modo per ridurre il numero di volte in cui l'utente esce fuori dal campo visivo della periferica potrebbe essere quello di aggiungere adeguati feedback visuali e sonori.

Come evidenziato da *Freeman et al.*[24], un altro aspetto da tenere in considerazione è la postura dell'utente che non sempre potrebbe essere ottimale. In Figura 2.7 è riportato un esempio in cui è mostrata una postura che, nonostante lo spazio a tre dimensioni, porta ad avere solo due gradi di libertà.



Figura 2.7: Un esempio di postura non ottimale tratto da Freeman et al.[24]

Quindi, in virtù di quanto detto è bene scegliere un insieme di gesti che non sia frenetico e che non richieda una elevata precisione per essere eseguito. Inoltre, per azioni complesse quali selezione e manipolazione è bene usare due mani invece che utilizzare gesti delle dita tutti su una singola mano. Ulteriormente è vitale usare un buon sistema di feedback visivo e sonoro per le azioni e adattare bene le interazioni a seconda della periferica che si va ad utilizzare e, soprattutto, non bisogna essere troppo vincolanti nel definire le posture che l'utente deve assumere per interagire con il sistema.

2.6.2 Quanti gesti scegliere

Inerentemente al quantitativo di interazioni gestuali da scegliere bisogna tener conto delle capacità di memorizzazione degli utenti. Jégo, Palijice e Fuchs nell'articolo "User-Defined Gestural Interaction: a Study on Gesture Memorization"[25] del 2013 hanno studiato le capacità di memorizzazione di gesti su un campione di 21 utenti usando un ambiente virtuale immersivo e sistemi di *Motion Capture* (Figura 2.8).

Il test da loro svolto si divide in due fasi: in prima battuta, all'utente immerso nell'ambiente virtuale venivano mostrate una serie di scatole di colore diverso e gli veniva chiesto di aprirle con un gesto che veniva ripetuto un certo numero di volte e poi registrato. Il colore della scatola è indice del contenuto e faceva sì che l'utente vi interagisse in maniera diversa. Ad esempio, l'utente era portato ad aprire con cautela una scatola rossa, cosa che non accadeva con una scatola di colore verde.

La seconda fase dell'esperimento di *Jégo et al.*[25] consisteva nel mostrare all'utente una successione casuale di scatole con un colore che variava tra quelli visti nella prima fase e di misurare le differenze tra i gesti usati nella prima fase di addestramento e quelli fatti nella seconda.

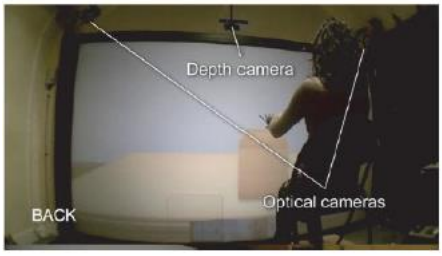


Figura 2.8: L'ambiente virtuale utilizzato da *Jégo et al.*[25]

I risultati di questo esperimento hanno evidenziato che mediamente un utente non riesce a ricordare più di tre gesti ed invitano a non esagerare con il numero di gesti quando si progetta una applicazione.

Per ridurre il numero di gesti necessari, invece di associare ad ogni azione un gesto, si potrebbe stabilire un insieme ristretto di gesti base (massimo tre) a cui sono associate azioni diverse a seconda della modalità di funzionamento correntemente attiva. All'insieme dei gesti base va aggiunto un gesto ulteriore per cambiare la modalità di interazione. Un sistema cosi progettato riduce lo sforzo di memorizzazione dei gesti, ma tuttavia, sacrifica un po' la naturalezza andando ad introdurre concetti non naturali quali, appunto, la selezione della modalità di funzionamento.

Capitolo 3

Il Sistema realizzato: descrizione

In questo capitolo sarà presentato *Graph 3D Explorer*: il sistema realizzato che permette la navigazione immersiva di grafi utilizzando interfacce naturali utente. Per prima cosa, sarà motivata la scelta di *Gephi* come base di partenza. Successivamente, saranno illustrate le funzionalità attualmente disponibili.

3.1 La scelta di Gephi

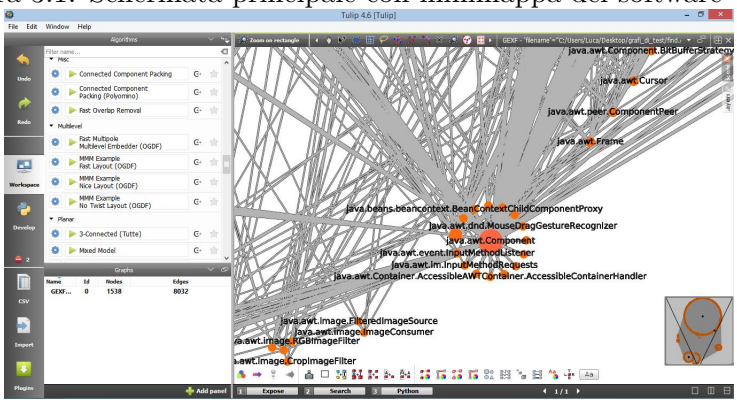
Uno degli obiettivi del lavoro di tesi è stato quello di fornire uno strumento per l'esplorazione di grafi in 3D immersiva che facesse uso di interazioni gestuali e di periferiche per la realtà virtuale. A tal fine, considerando che l'esplorazione deve essere un elemento da affiancare alla visualizzazione canonica e all'analisi basata su metriche, si è scelto di appoggiarsi a software preesistente.

Per la scelta del software di base si è partiti da una comparazione di software di visualizzazione fatta nel 2013 da dei ricercatori dell'università di Tokio[26] dove sono stati considerati gli applicativi capaci di gestire reti di grandi dimensioni con la maggiore utenza e con gli aggiornamenti più regolari. Da questa analisi comparativa abbiamo eliminato tutti i software che non permettessero l'aggiunta di funzionalità tramite plugin restringendo così la scelta a tre candidati: *Cytoscape*[27][28], *Tulip*[29] e *Gephi*[11].

Tutti e tre offrono molte funzionalità per la visualizzazione di reti e offrono il supporto a molti formati per la descrizione dei grafi. *Tulip*[29] è scritto in *C++* e fa della efficienza la sua caratteristica principale ed inoltre offre molti

strumenti di visualizzazione interessanti come la lente di ingrandimento e la distorsione *fisheye* che consiste in uno zoom semantico in cui le componenti non interessanti vengono distorte. Inoltre, oltre ai classici *Pan* e *Zoom* sulla visualizzazione, *Tulip* consente la navigazione della rete sfruttando una tecnica detta *Bring and Go*[30] che consiste nel andare di nodo in nodo sfruttando gli archi del grafo. L'orientamento dell'utente durante l'esplorazione è aiutato con una minimappa situata in basso a destra del visualizzatore (Figura 3.1).

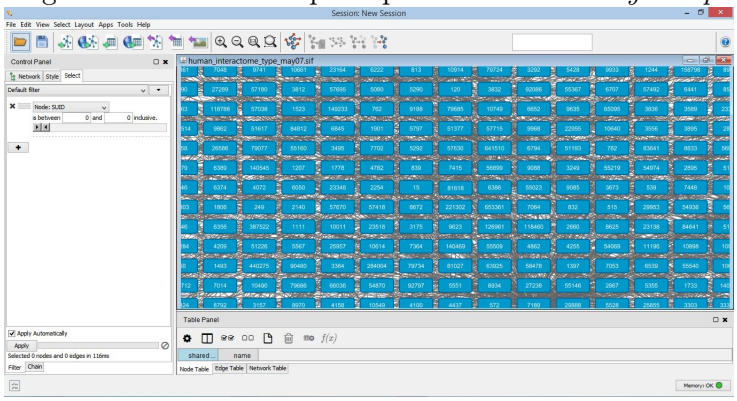
Figura 3.1: Schermata principale con minimappa del software *Tulip*



Tulip è un buon software, peccato che sia poco espandibile. Infatti, con il sistema di estensioni attuale è permessa la creazione di algoritmi di *Graph Drawing* che interagiscano con la rete ma non la creazione di pannelli o finestre di visualizzazione aggiuntive. Ulteriormente, nonostante sia scritto in *C++* e sia dichiarato efficiente, sono stati rilevati degli impuntamenti dell'applicazione durante l'uso di algoritmi di *Graph Drawing* su alcuni grafi di test di dimensioni esigue (8000 nodi).

Cytoscape (Figura 3.2) invece, è basato su *Java* ed ha una architettura modulare che permette anche l'aggiunta di nuove componenti grafiche. Nasce come strumento per la biologia atto alla visualizzazione di relazioni molecolari ma, con la attuale versione (3.0), è stato generalizzato. Tuttavia, non offre ancora un sufficiente numero di plugin d'uso generale per l'analisi delle reti ed ha dei meccanismi di manipolazione della visualizzazione macchinosi.

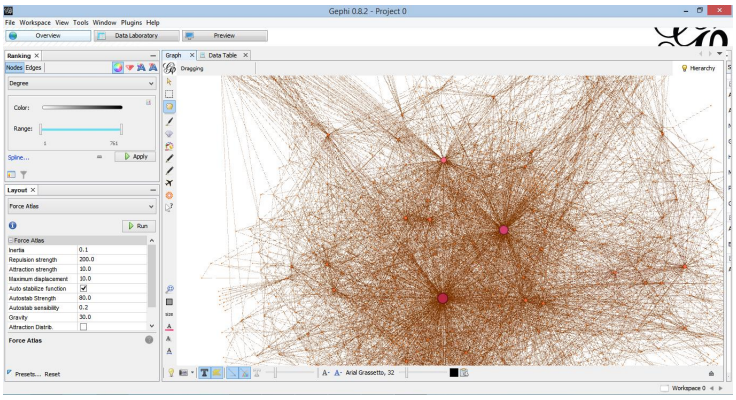
Figura 3.2: Schermata principale del software *Cytoscape*



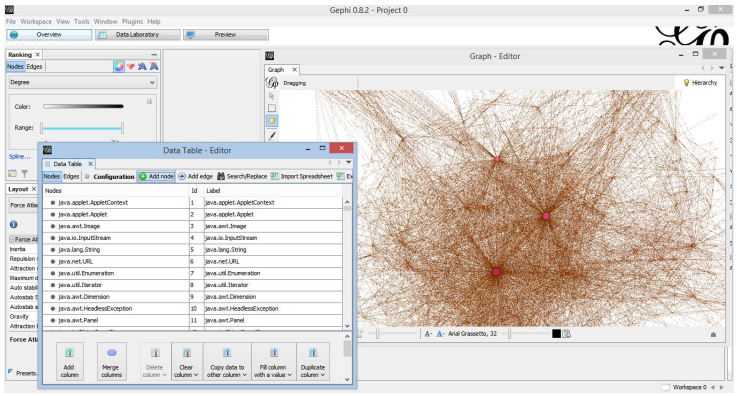
Tra i tre, *Gephi* è quello che più si distingue per versatilità delle funzionalità offerte, in quanto, oltre alla visualizzazione di una rete, permette di applicare metriche di *Social Network Analysis* con generazione di grafici e offre il supporto ai grafi dinamici (dei grafi che cambiano nel tempo). Inoltre, lo sviluppo di plugin per *Gephi* è molto favorito dal framework *Netbeans Platform*[31] che offre tutta una serie di vantaggi per lo sviluppo di applicazioni desktop quali, ad esempio, la gestione dei pannelli che possono essere ridimensionati o staccati dalla finestra principale consentendo una maggiore produttività (Figura 3.3).

Proprio i pannelli sganciabili contraddistinguono *Gephi* da *Tulip* e *Cytoscape* perché danno la possibilità all'utente di avere una doppia visione della rete senza dover preoccuparsi di cambiare finestra portando così alla complementarietà desiderata tra i paradigmi *Query* e *Browsing* dell' *Information Retrieval*. Le caratteristiche citate e il continuo supporto, testimoniato dal fatto che è stato tra i progetti disponibili al *Google Summer of Code*[32] dal 2010 al 2013, hanno reso *Gephi* il punto di partenza migliore per il *software* realizzato.

Figura 3.3: L'interfaccia utente del software *Gephi*



(a) Tutti i pannelli uniti all'interfaccia principale



(b) Pannelli sganciati e che meglio si adattano alle esigenze dell'utente

3.2 Gephi: cosa si può migliorare

Lo slogan usato dagli sviluppatori di Gephi e che meglio lo descrive è "Like Photoshop but for data". Infatti, l'utente utilizzando *Gephi* interagisce con la rappresentazione, ne manipola la struttura, la forma ed il colore e ne inferisce delle proprietà. L'analogia con Photoshop è ancora più rilevante se ci soffermiamo un attimo sul fatto che Gephi permette di esportare il grafo come *PNG*, *SVG*, *JPG* che sono tra i formati immagine più comuni. Di fatto, l'utente con *Gephi* produce una rappresentazione statica della rete per esaltare alcuni aspetti salienti di un fenomeno che vuole analizzare.

Tuttavia, le funzionalità di visualizzazione di *Gephi* hanno alcuni limiti. Infatti, la rete viene sì visualizzata in uno spazio 3D ma di fatto non c'è possibilità di ruotare oggetti né di muoversi liberamente. Inoltre, la visualizzazione standard di *Gephi* è a camera fissa e le uniche interazioni possibili con il grafo sono di tipo Pan e Zoom effettuabili previa selezione dell'apposito strumento tramite delle *toolbar*. Inoltre, tra i vari plugin ci sono degli algoritmi di layout che dispongono i nodi su tre dimensioni ma la visualizzazione standard non permette di apprezzarne appieno il significato.

3.3 Descrizione generale delle funzionalità

Considerando la necessità di affiancare agli strumenti di analisi e visualizzazione proposti da *Gephi* un sistema di visualizzazione della rete che faccia un reale uso delle tre dimensioni e che sia multimodale e interattivo, è stato progettato e sviluppato *Graph 3D explorer*: plugin per Gephi per la visualizzazione e navigazione della rete in maniera interattiva. Esso nasce con i seguenti requisiti:

- Visualizzazione, navigazione e manipolazione in 3D della rete.
- Supporto a diversi dispositivi di input e alle interfacce naturali.
- Supporto alle funzionalità preesistenti di *Gephi*.

L'esplorazione della rete in *Graph 3D Explorer* si rifà a paradigmi propri dei videogiochi quali la navigazione in prima persona ed offre la possibilità di spostare nodi, ridimensionarli, raggrupparli in maniera interattiva direttamente nell'ambiente tridimensionale.

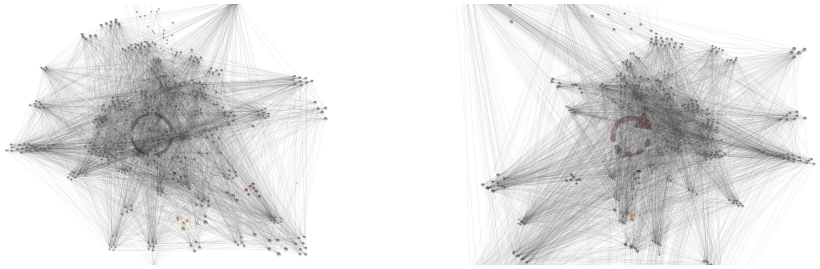
La navigazione interattiva della rete può avvenire sia con mouse e tastiera che con il *Joypad*. Ma, un punto cardine di *Graph 3D Explorer* è l'uso di Interfacce naturali per l'interazione. Infatti, è stata progettata e sviluppata una metodologia per la navigazione e manipolazione della rete totalmente basata su gesti utilizzando la periferica Leap Motion. Inoltre, è stato sviluppato il supporto alla periferica Oculus Rift per avere una visualizzazione dei dati immersiva e altamente interattiva che può essere usata in combinazione con *Mouse e Tastiera*, *Joypad* e *Leap Motion*. Questo, considerando anche l'utilizzo del sistema con i normali monitor portano *Graph 3D Explorer* a fornire sei modalità di utilizzo differenti.

Oltre alla visualizzazione 3D della rete, uno dei requisiti alla base della progettazione è quello della retrocompatibilità. Infatti, tutte le funzionalità di *Gephi* attualmente disponibili sono compatibili con il sistema realizzato.

3.4 Visualizzazione, esplorazione e manipolazione

La metodologia proposta in *Graph 3D Explorer* per il reperimento delle informazioni ricalca il paradigma "Overview first then details on demand" derivante dal *Visual Information Seeking Mantra*[33] proposto da *Shneiderman* in cui si afferma la necessità di avere una visione d'insieme iniziale (*Overview First*) da cui scegliere un determinato aspetto su cui concentrarsi (*Zoom*) e di cui avere, eventualmente, i dettagli (*details on demand*).

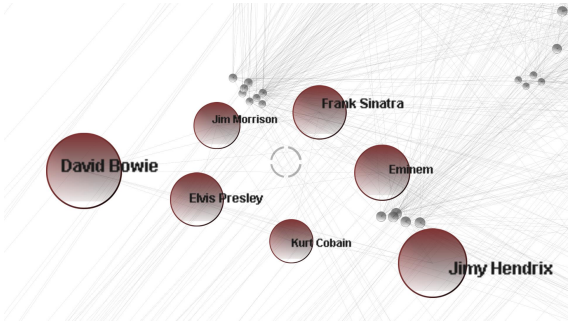
Infatti, l'utente inizialmente è lontano dal grafo (Figura 3.4(c)) e ne può avere una visione d'insieme ruotandolo (Figura 3.4(d)). Successivamente sceglie dove andare e solo quando è in prossimità dei nodi di interesse gli vengono mostrate alcune informazioni (Figura 3.5(a)). Questo approccio permette di evitare l'*Information Overload* dovuto all'ammassarsi delle informazioni testuali che tipicamente si presenta nei software di visualizzazione di grafi.



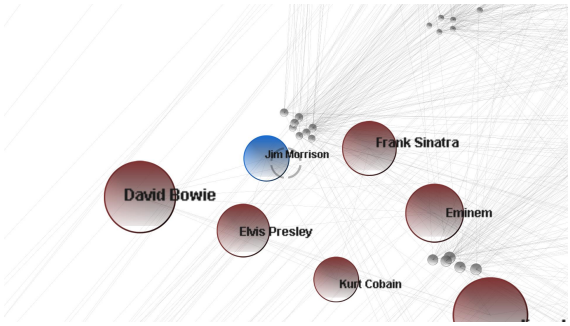
(c) Rete in lontananza (d) Rotazione in corso

Figura 3.4: Visione d'insieme iniziale

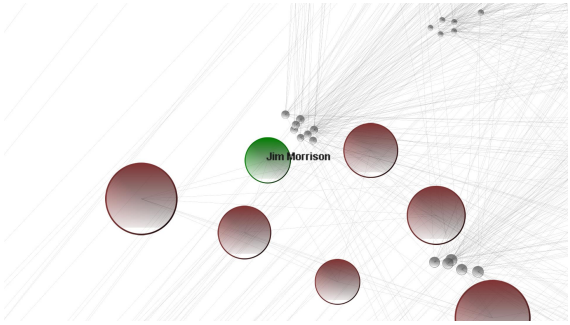
La manipolazione della rete avviene nell'ambiente virtuale per mezzo di un area circolare ridimensionabile posta di fronte all'osservatore che permette la selezione dei nodi che vi cadono all'interno o che vi sono in prossimità. Una volta selezionati, i nodi possono essere ridimensionati, spostati o raggruppati insieme oppure possono essere visualizzati dettagli aggiuntivi. Un esempio di visualizzazione di informazioni unita ad un esempio di schermata dettagli è dato in Figura 3.5. Invece, in Figura 3.6 è dato un esempio commentato di visualizzazione di un insieme di nodi che viene selezionato, ridimensionato e successivamente raggruppato insieme.



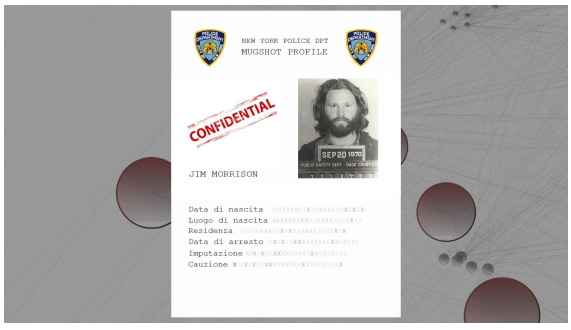
(a) Visualizzazione informazioni



(b) Nodo selezionabile

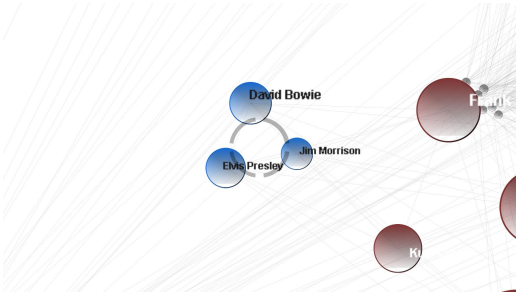


(c) Nodo selezionato

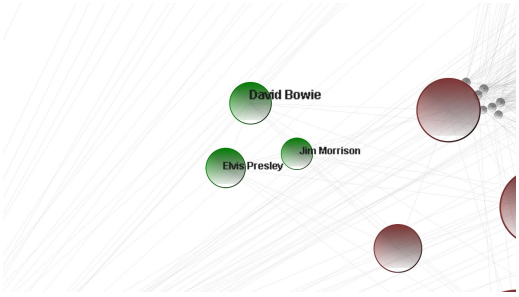


(d) Un esempio di possibili dettagli per un nodo

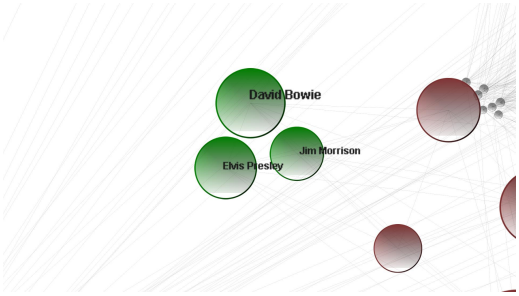
Figura 3.5: Visualizzazione, selezione e un esempio di schermata dettagli. In questo esempio è stato selezionato un solo nodo ma, in generale, possono essere selezionati anche più nodi. In tal caso le schermate di dettaglio saranno ciclabili.



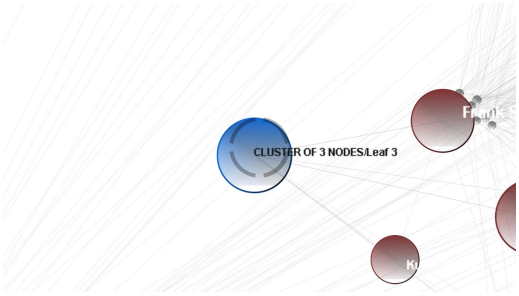
(a) Più nodi selezionabili



(b) I nodi sono stati selezionati



(c) Ridimensionamento della selezione



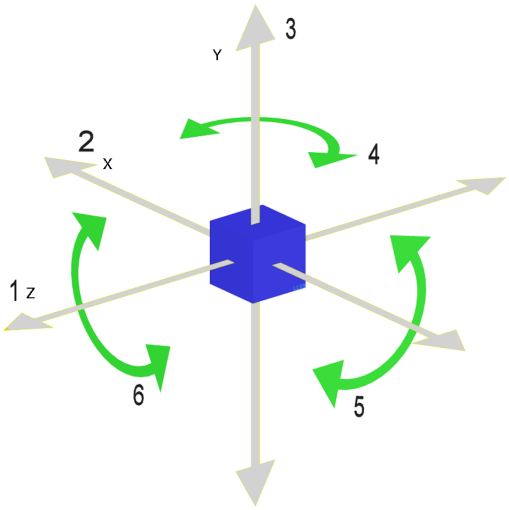
(d) Il gruppo (*Cluster*) creato

Figura 3.6: Un esempio di visualizzazione, selezione multipla con un ridimensionamento ed raggruppamento (*Clustering*) finale. In questo caso sono stati selezionati più nodi ma, in generale, l'operazione di clustering è possibile anche con un solo nodo. Ulteriormente si possono creare cluster che contengono cluster ed inoltre tutti quelli che vengono creati possono essere divisi facendo tornare i nodi che li compongono alle loro posizioni originarie.

3.5 Dispositivi di input e interfacce naturali

In *Graph 3D Explorer*, i nodi giacciono in uno spazio tridimensionale in cui l'utente, sfruttando una visuale in prima persona, può muoversi avanti e indietro, su e giù o destra e sinistra e contemporaneamente può ruotare la camera in alto e in basso. Questo insieme di movimenti è una restrizione di quello che, tipicamente, è identificato con *6DoF- Six Degrees Of Freedom* ovvero sei gradi di libertà cioè l'insieme delle traslazioni possibili per un corpo rigido sui tre assi spaziali unite alle corrispettive rotazioni (Figura 3.7). L'uso di cinque gradi di libertà, escludendo la rotazione sull'asse Z (sesto grado in Figura 3.7), è una semplificazione tipica dei videogiochi e dei software che fanno uso della visuale in prima persona.

Figura 3.7: I sei gradi di libertà: le traslazioni e rotazioni possibili per un corpo rigido



Di seguito verrà analizzato come le varie configurazioni proposte rendano fattibili questi movimenti e le altre azioni disponibili.

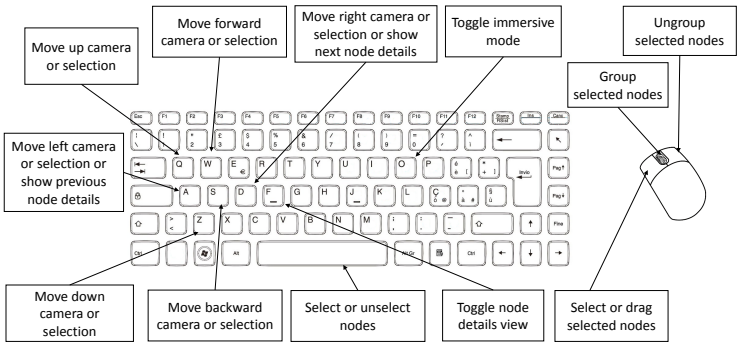
3.5.1 Mouse e Tastiera

La configurazione che prevede *Mouse e Tastiera* fa uso del noto schema di comandi *WASD*[34] in cui i tasti $W\ A\ S\ D$ vengono usati per muoversi in avanti, sinistra, destra e indietro rispettivamente. Questo sistema di controllo è molto usato nei videogiochi in quanto, considerata la posizione tipica sulle

tastiere dei quattro tasti *W A S D*, offre una buona raggiungibilità di tutti i comandi con la mano sinistra e rende la mano destra libera per l'utilizzo del mouse con cui viene gestita la camera. Infatti, muovendo il mouse verso sinistra si ha una rotazione della camera versa sinistra, analogamente, muovendo il mouse verso destra si ha una rotazione della camera verso destra. Il discorso cambia per le rotazioni della camera verso l'alto e verso il basso che sono state associate al movimento in avanti e al movimento all'indietro del mouse. Questa tipologia di comandi per le rotazioni della camera verso l'alto e verso il basso nasce dal fatto che il mouse si muove su una superficie piana. La bidimensionalità intrinseca del *Mouse* è il motivo principale per cui si preferisce restringere i gradi di libertà da sei a cinque impedendo la rotazione sull'asse *Z*.

Al sistema di comandi descritto sono stati aggiunti i tasti *Q* ed *E* per i movimenti verso l'alto e verso il basso e le funzionalità di selezione-deselezione con la *Barra Spaziatrice*. Questi tasti essendo vicini ai *W A S D* destinati al movimento non richiedono grossi spostamenti della mano sinistra durante l'utilizzo favorendo l'ergonomia e l'usabilità. Lo schema completo dei comandi per la configurazione *Mouse e Tastiera* è dato in Figura 3.8 dove sono aggiunte le descrizioni anche per il tasto *F* che permette di entrare nella visualizzazione dettagli e per il tasto *O* che permette di attivare/disattivare la modalità immersiva con la periferica *Oculus Rift*.

Figura 3.8: Comandi attualmente disponibili con *Mouse e Tastiera*

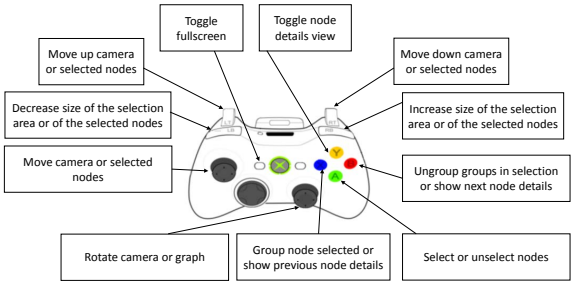


3.5.2 Joypad

Il sistema offre il supporto al *Controller XBOX 360* della *Microsoft*. Tipicamente questa periferica è usata per giocare ai videogiochi ma, data la buona ergonomia, costituisce un buon sistema di controllo per qualsiasi applicazione che richiede spostamenti in ambienti tridimensionali. Infatti, le levette analogiche possono svolgere le funzioni dei tasti W A S D e del mouse permettendo le stesse modalità di navigazione e d'interazione nell'ambiente virtuale che si hanno con *Mouse e Tastiera* ma utilizzando meno spazio e non richiedendo un piano di appoggio. Tuttavia, una singola levetta può rappresentare solo due assi e per far sì che gli oggetti e la camera possano essere spostati verso l'alto o il basso, analogamente al sistema *Mouse e Tastiera*, è stato necessario associare tasti aggiuntivi che in questo caso sono le levette laterali.

In merito all'applicazione sviluppata, oltre ai movimenti della camera e degli oggetti selezionati tramite le levette analogiche e i grilletti laterali, sono stati usati il tasto A per la selezione e deselezione, il tasto X per raggruppare i nodi selezionati e il tasto B per scindere i gruppi di nodi che sono eventualmente presenti tra i quelli selezionati. Ulteriormente, premendo verso il basso la levetta destra essa cambia funzionalità passando da strumento per ruotare la camera a strumento che permette la rotazione del grafo. In Figura 3.9 è dato lo schema completo dei comandi attualmente disponibile per questa configurazione.

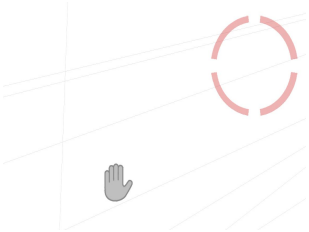
Figura 3.9: Schema dei controlli con il controller XBOX



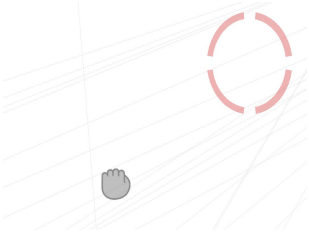
3.5.3 Il sistema di comandi gestuali

Il supporto alle interfacce naturali utente è fornito tramite la periferica *Leap Motion* che permette di tener traccia dei movimenti delle mani e delle dita. Sfruttando questa periferica, è stato progettato un insieme di interazioni gestuali che permettesse la totale fruizione del sistema al pari delle altre periferiche di input supportate.

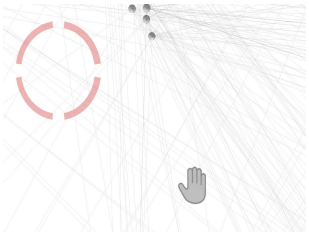
Alla base del sistema di gesti progettato c'è la semplicità. Infatti, l'utente non appena collega la periferica *Leap Motion* e posiziona le mani nel suo campo visivo subito può interagire con il sistema riuscendo a fare tutte le azioni disponibili (navigazione della rete, spostamento o raggruppamento dei nodi, visualizzare dettagli...) con dei semplici movimenti delle mani nell'aria. Proprio per favorire la comprensione e la facilità d'utilizzo, l'interfaccia grafica del sistema fornisce all'utente dei chiari riscontri visivi sul rilevamento di una o di entrambe le mani e sul fatto che esse risultino rilevate come aperte o chiuse (Figura 3.10).



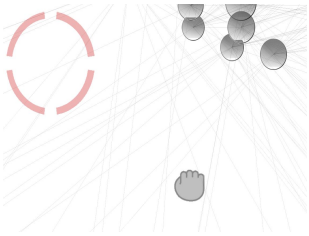
(a) Icona per la mano sinistra aperta



(b) Icona per la mano sinistra chiusa



(c) Icona per la mano destra aperta



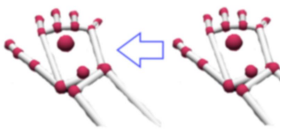
(d) Icona per la mano destra chiusa

Figura 3.10: Sistema di icone usato per indicare all'utente lo stato delle mani rilevato dalla periferica Leap Motion. In caso di utilizzo combinato delle due mani, le icone relative alla mano destra e sinistra compariranno entrambe.

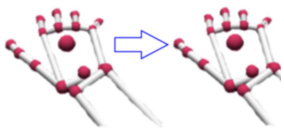
Navigazione della rete

Per la navigazione nella rete, l'utente ha a disposizione due modalità: *Hand Rotation* e *Linear*. Nella modalità *Hand Rotation* l'utente può muovere la camera nelle sei direzioni (avanti, indietro, sinistra, destra, alto, basso) utilizzando i movimenti lineari a mano singola riportati in Figura 3.11 e contemporaneamente può ruotare la camera a destra, sinistra, alto o basso utilizzando le rotazioni del palmo riportate in Figura 3.12.

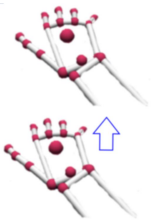
Considerando la modalità *Linear* si ha un controllo della camera più semplificato in cui vengono usati solo i movimenti lineari. In particolare, i movimenti lineari a destra, sinistra, alto e basso vengono usati per ruotare la camera nelle medesime direzioni mentre i movimenti lineari in avanti e indietro sono gli unici che spostano la camera e lo fanno in avanti e indietro rispettivamente. La modalità *Linear* è stata affiancata a quella *Hand Rotation* per venire in aiuto a tutte quelle persone che trovano difficoltà a ruotare il palmo perché hanno poca agilità di polso.



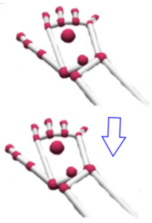
(a) Movimento lineare a sinistra



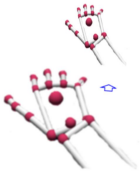
(b) Movimento lineare a destra



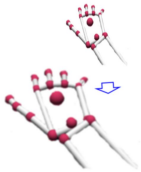
(c) Movimento lineare in alto



(d) Movimento lineare in basso

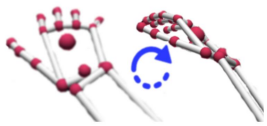


(e) Movimento lineare in avanti

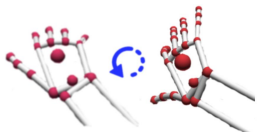


(f) Movimento lineare all'indietro

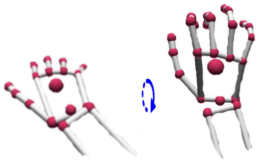
Figura 3.11: Questa figura riporta i movimenti lineari presi in considerazione. Essi possono essere compiuti a mano aperta o chiusa oppure con il solo dito indice esteso. Se fatti a mano aperta come riportato in figura permettono il controllo della camera. Se invece sono fatti a mano chiusa permettono lo spostamento dei nodi selezionati nelle rispettive direzioni indicate dalle frecce. Se fatti con il dito indice esteso allora permettono la rotazione del grafo. Considerando i movimenti a mano aperta, se la modalità di controllo della camera è *Linear* allora tutti i movimenti lineari diversi da quelli in avanti e indietro sono usati per ruotare la camera. Da notare che i movimenti lineari a mano chiusa e ad indice esteso non sono riportati in quanto totalmente analoghi a quelli raffigurati.



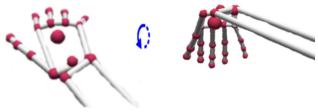
(a) Rotazione del palmo a sinistra



(b) Rotazione del palmo a destra



(c) Rotazione del palmo verso l'alto



(d) Rotazione del palmo verso il basso

Figura 3.12: La figura indica le rotazioni della mano per il controllo della camera nella modalità *Hand Rotation*. Il palmo segue la rotazione opposta alle frecce indicanti il movimento della mano. In particolare, rotazioni del palmo verso il basso e verso l'alto fanno ruotare la camera verso il basso e verso l'alto rispettivamente. Analogamente, le rotazioni del palmo verso destra e sinistra fanno ruotare la camera nelle medesime direzioni.

Rotazione del grafo

L'utente, come descritto in precedenza alla Figura 3.4(c), durante la navigazione può scegliere dove andare ruotando il grafo. Ciò può essere fatto tramite gesti utilizzando il solo dito indice esteso come riportato in Figura 3.13 e compiendo i movimenti lineari di Figura 3.11 sempre con il solo indice esteso. Non appena l'utente cambia la postura della mano chiudendola o aprendola completamente si smette di ruotare il grafo e si cambia tipologia di interazione.

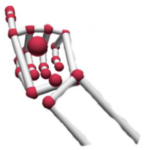


Figura 3.13: Postura per iniziare la rotazione del grafo o uscire dalla visualizzazione dettagli

Selezione e spostamento di nodi

L'utente può selezionare i nodi che cadono nell'area di selezione semplicemente chiudendo la mano come riportato in Figura 3.14. Una volta stretto il pugno, l'utente può muovere la selezione compiendo a pugno chiuso i movimenti lineari riportati in Figura 3.11.

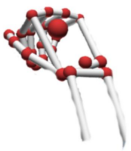


Figura 3.14: Postura per effettuare la selezione dei nodi

Visualizzazione dettagli di un nodo o di un gruppo di nodi

Se l'utente volesse visualizzare i dettagli relativi a dei nodi, deve prima selezionarli e poi affiancare alla mano chiusa adibita alla selezione la seconda mano in postura di indicazione come riportato in Figura 3.15. A seguito di questa azione comparirà una schermata simile a quella riportata in Figura 3.5(d). Se la selezione è multipla l'utente può ciclare i dettagli dei nodi utilizzando i movimenti lineari di Figura 3.11 fatti a mano aperta. In particolare, un movimento lineare a sinistra fa visualizzare la schermata dettagli precedente mentre un movimento lineare a destra fa visualizzare la schermata dettagli successiva. Quando l'utente vuole tornare alla normale navigazione può uscire dalla visualizzazione dettagli ripetendo la postura indicazione di Figura 3.13 con una mano qualsiasi.

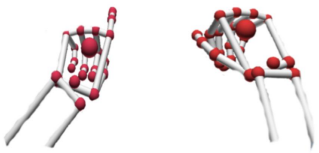
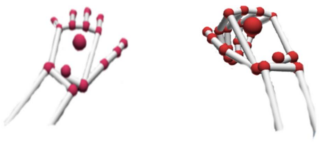


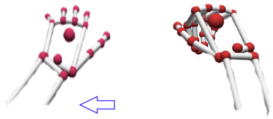
Figura 3.15: Postura per far sì che vengano visualizzati i dettagli di un nodo (o di un gruppo)

Ridimensionamento di una selezione

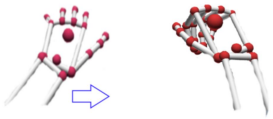
Qualora l'utente volesse ridimensionare un insieme di nodi, deve prima selezionarli e poi usare la seconda mano aperta. In particolare, se vuole rimpicciolirli allora la seconda mano aperta si deve avvicinare alla prima, mentre, se vuole ingrandirli essa si deve allontanare. In figura 3.16 è riportato l'insieme di gesti necessari per rimpicciolire o ingrandire i nodi in una selezione supponendo che si abbia usato la mano destra per effettuare la selezione.



(a) Postura iniziale per l'interazione con la selezione



(b) Movimento della mano aperta per ingrandire i nodi selezionati

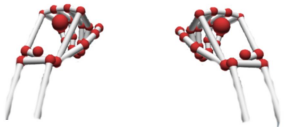


(c) Movimento della mano aperta per rimpicciolire i nodi selezionati

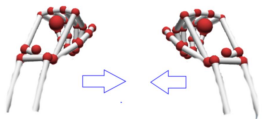
Figura 3.16: In questa serie di figure sono riportate le interazioni necessarie per ridimensionare i nodi di una selezione. La mano usata per selezionare corrisponde alla mano chiusa ed in questo esempio è la destra. Nel caso si usi la mano sinistra per selezionare allora bisogna invertire la mano aperta con la mano chiusa e seguire il verso opposto delle frecce.

Clustering e UnClustering di nodi selezionati

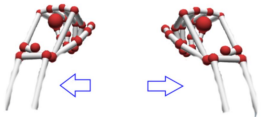
Qualora l'utente volesse effettuare un raggruppamento (*Clustering*) dei nodi selezionati, deve prima di tutto selezionarli chiudendo il pugno come in precedenza (Figura 3.14). Successivamente, deve usare la seconda mano da chiusa e avvicinarla alla prima oppure avvicinarle entrambe. Invece, nel caso in cui l'utente voglia rompere un gruppo formato in precedenza deve compiere le medesime azioni solo che invece di avvicinare la seconda mano chiusa alla prima, la deve allontanare. Un esempio delle azioni descritte è dato in Figura 3.17.



(a) Postura iniziale per l'interazione di *Clustering-UnClustering* di una selezione



(b) Movimenti per effettuare un *Clustering*



(c) Movimenti per effettuare un *UnClustering*

Figura 3.17: In questa serie di figure sono riportate le interazioni necessarie per raggruppare dei nodi (*Clustering*) o rompere dei gruppi creati in precedenza (*UnClustering*). Se si avvicinano (o allontanano) entrambe le mani non c'è bisogno di fare nessuna considerazione sulla mano che abbia effettuato la selezione. Invece qualora si muova una mano sola allora valgono le regole per il ridimensionamento dei nodi date in Figura 3.16 in cui si considera la creazione di un cluster come un rimpicciolimento mentre l'azione di *UnClustering* come ad un ingrandimento.

3.5.4 La modalità immersiva

L'utente può interagire con il sistema usando tutti i dispositivi descritti nelle sezioni precedenti e può fruire i contenuti tramite un qualsiasi monitor oppure in maniera immersiva grazie alla visualizzazione in 3D usando la periferica *Oculus Rift*.

Per entrare in modalità immersiva l'utente deve semplicemente collegare l'Oculus Rift al sistema e poi premere il tasto *O* sulla tastiera. Questo farà passare l'applicazione a schermo intero e farà sdoppiare la visualizzazione che sarà poi proiettata sullo schermo dell'Oculus Rift. L'unione tra la generazione di un immagine distinta per ogni occhio e il sistema di lenti integrate nel visore darà all'utente la percezione delle tre dimensioni e la sensazione di essere totalmente immerso nella rete di dati. In questa modalità, l'applicazione sarà ancora visibile sullo schermo standard e mostrerà quello che l'utente sta vedendo in maniera immersiva (Figura 3.18).

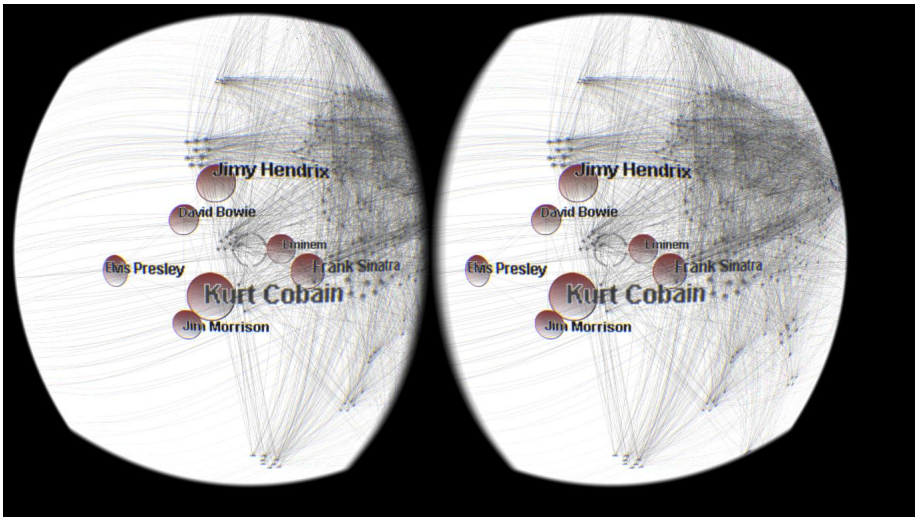
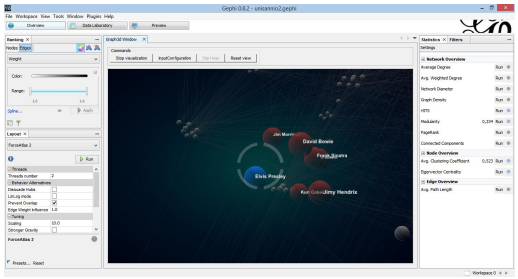


Figura 3.18: Questa figura, grazie alla visualizzazione in contemporanea sia sulla periferica *Oculus Rift* che sul monitor standard, mostra uno stralcio di quello che l'utente sta vedendo in 3D dopo che ha attivato la modalità immersiva e indossato il visore.

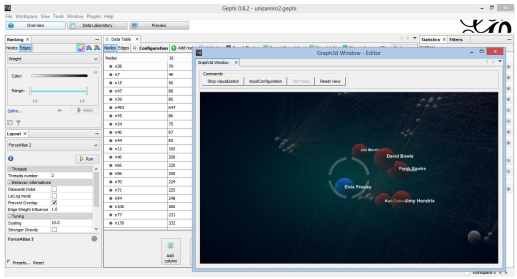
In questa modalità di visualizzazione, la gestione della visuale è totalmente demandata all'utente che, semplicemente muovendo la testa, può ruotare la camera e guardare dove vuole proprio come se fosse realmente nella rete di dati.

3.6 Integrazione con Gephi

Oltre che a schermo intero come mostrato nelle sezioni precedenti, *Graph 3D Explorer* può funzionare sia ancorato all'applicazione principale sotto forma di pannello aggiuntivo (Figura 3.19(a)) che in una finestra indipendente (Figura 3.19(b)). Queste ulteriori modalità di funzionamento permettono all'utente di passare comodamente da qualsiasi funzionalità offerta da *Gephi* alla visualizzazione con *Graph 3D Explorer* e viceversa.



(a) *Graph 3D Explorer* ancorato alla finestra principale di *Gephi*



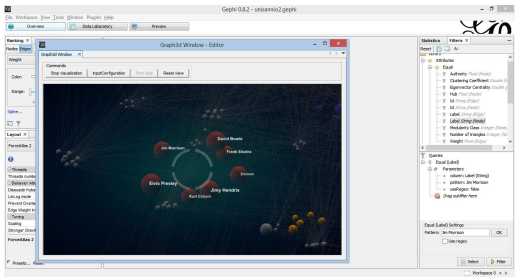
(b) *Graph 3D Explorer* in una finestra indipendente che può essere eventualmente riagganciata alla principale quando si vuole

Figura 3.19: In questa figura sono mostrate le altre due modalità di funzionamento di *Graph 3D Explorer* diverse dal pieno schermo mostrato in tutte le immagini precedenti

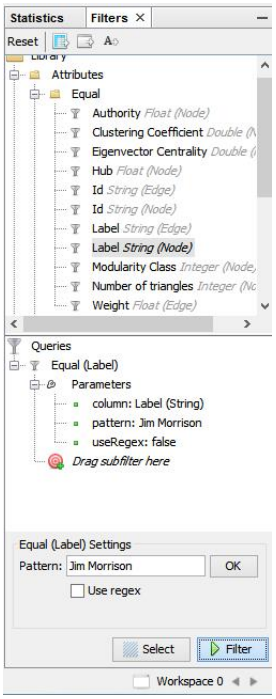
Graph 3D Explorer garantisce l'interoperabilità con gli altri plugin di *Gephi*. Ad esempio, qualora in *Gephi* applicassimo un algoritmo di *Graph Drawing* scelto dall'apposito menu laterale i cambiamenti della rete saranno visibili in tempo reale in *Graph 3D Explorer* sia che esso stia funzionando ancorato alla finestra principale oppure che stia funzionando in una finestra indipendente ma anche con una eventuale visione a pieno schermo su un secondo monitor.

Ulteriormente, tutti i filtri di ricerca testuale disponibili in *Gephi* possono influenzare la visualizzazione di *Graph 3D Explorer* facendo mostrare solo gli elementi corrispondenti ai criteri impostati. Inoltre, *Graph 3D Explorer* può mostrare gli eventuali effetti di plugin che fanno clustering della rete o che colorano nodi o archi a seconda di certe proprietà. Un esempio di interoperabilità con *Gephi* è dato in Figura 3.20 in cui è mostrata l'applicazione di filtro di ricerca e i corrispettivi effetti sulla visualizzazione.

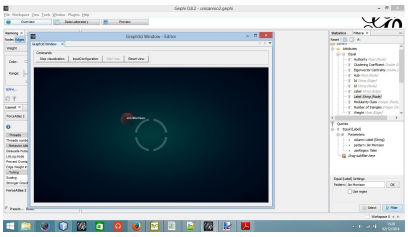
Tutta questa serie di caratteristiche di interoperabilità sono frutto di scelte che saranno discusse nell'appendice relativa ai dettagli implementativi del sistema proposto.



(a) La rete prima di applicare il filtro



(b) La barra di *Gephi* per applicare filtri di ricerca sul grafo



(c) La rete dopo che il filtro è stato applicato

Figura 3.20: In questa serie di figure è riportato un esempio di applicazione dei filtri presenti in *Gephi*. In particolare, sono mostrati la rete prima e dopo l'applicazione di un filtro basato sull'etichetta dei nodi.

Capitolo 4

Il Sistema realizzato: implementazione

In questo capitolo saranno dati i dettagli salienti in merito all'architettura e all'implementazione del sistema realizzato. In particolare, sarà data una descrizione dell'architettura generale e saranno approfondite le componenti del sistema che si interfacciano con le periferiche *Leap Motion* e *Oculus Rift*.

4.1 Architettura del sistema

La *Netbeans Platform* offre una serie di librerie che facilitano la creazione di interfacce grafiche basate su Java ed inoltre permette di creare delle applicazioni facilmente espandibili grazie ad una architettura modulare basata su plugin. Gephi si poggia sulla Netbeans Platform di cui sfrutta le librerie grafiche e il sistema di plugin per permettere l'aggiunta di funzionalità. Proprio grazie a questo, *Graph 3D Explorer* può esistere. La *GUI* di *Graph 3D Explorer* e quella di *Gephi* girano insieme sul framework messo a disposizione dalla *Netbeans Platform* e sono totalmente indipendenti dalla logica applicativa che fa uso delle funzionalità offerte dal *Gephi Toolkit*. Il *Gephi Toolkit* è un insieme di librerie offerte da *Gephi* che possono essere usate anche da applicazioni *standalone* e che permettono di caricare grafi partendo da file di vari formati ed offrono varie classi per la rappresentazione di grafi gerarchici, grafi dinamici oltre che a tutto un insieme vasto di algoritmi di *graph drawing*.

Durante tutto lo sviluppo di *Graph 3D Explorer* si è tenuto conto di queste componenti esterne e si è cercato di confinare il più possibile il codice che vi interagisse in modo da garantire una buona manutenibilità nel caso

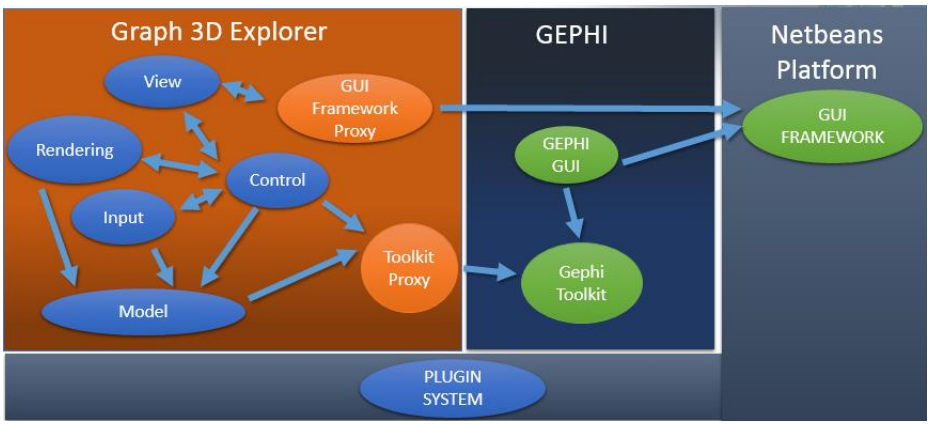


Figura 4.1: Uno schema generale dell'architettura di *Graph 3D Explorer*

in cui esse subiscano modifiche in qualche versione futura. Inoltre, questa filosofia di base agevola un eventuale supporto futuro ad altri software di visualizzazione sempre basati su *Java* quali ad esempio *Cytoscape* che, come detto nel capitolo precedente, è stato tra i candidati iniziali ma poi scartato a causa della poca generalità della versione attuale.

Graph 3D Explorer nell'attuale versione si basa su una architettura con cinque sottosistemi fondamentali: *Model*, *View*, *Control*, *Rendering* e *Input*. Nel seguito di questo capitolo andremo ad analizzare le funzionalità e le componenti principali di questi sottosistemi.

4.2 Il Modello: Grafo e Osservatore

Il sottosistema *Model* racchiude tutte le componenti logiche del sistema realizzato. Il principale scopo di questo sottosistema è quello di gestire la struttura del grafo. Fondamentali per questo modulo sono le componenti *Graph3D* e *GephiGraph3D*.

La componente *Graph3D* è una interfaccia che racchiude tutti i metodi principali per il grafo mentre *GephiGraph3D* costituisce una implementazione concreta di questa interfaccia e fa da *proxy* verso la struttura dati grafo del *Gephi Toolkit*. Questa è una applicazione pratica del design pattern *Proxy*[35] e si rende necessaria in quanto il grafo offerto da *Gephi* non ha nessuna informazione relativa alle rotazioni.

Altri elementi importantissimi sono le classi *Graph3DSelection* e *GraphObserver*. La classe *Graph3DSelection* rappresenta quello che l'utente ha

selezionato o potrebbe selezionare nel corso della sua esplorazione mentre la classe *GraphObserver* rappresenta l'utente che si muove nello spazio tridimensionale e osserva il grafo. La classe *GraphObserver* possiede un istanza della classe *ViewFinder* che si occupa di rappresentare l'area di selezione circolare e ha attributi che riguardano il raggio e la distanza.

L'osservatore, la selezione e il grafo sono degli elementi che si possono muovere o ruotare nello spazio. Queste classi sono tutte estensioni della classe *AbstractMovable* che implementa l'interfaccia *Movable* e mantiene variabili d'istanza comuni a tutte quali la posizione nello spazio data da un vettore tridimensionale e la rotazione rappresentata come quaternione.

I vettori e i quaternioni sono dei concetti matematici usati per rappresentare punti e rotazioni. In particolare, per i vettori è stata usata la libreria *Vecmath* mentre per quanto riguarda i quaternioni è stata usata l'implementazione fornita dalla libreria *JOGL*. In figura 4.2 è dato il class diagram delle classi di questo sottosistema.

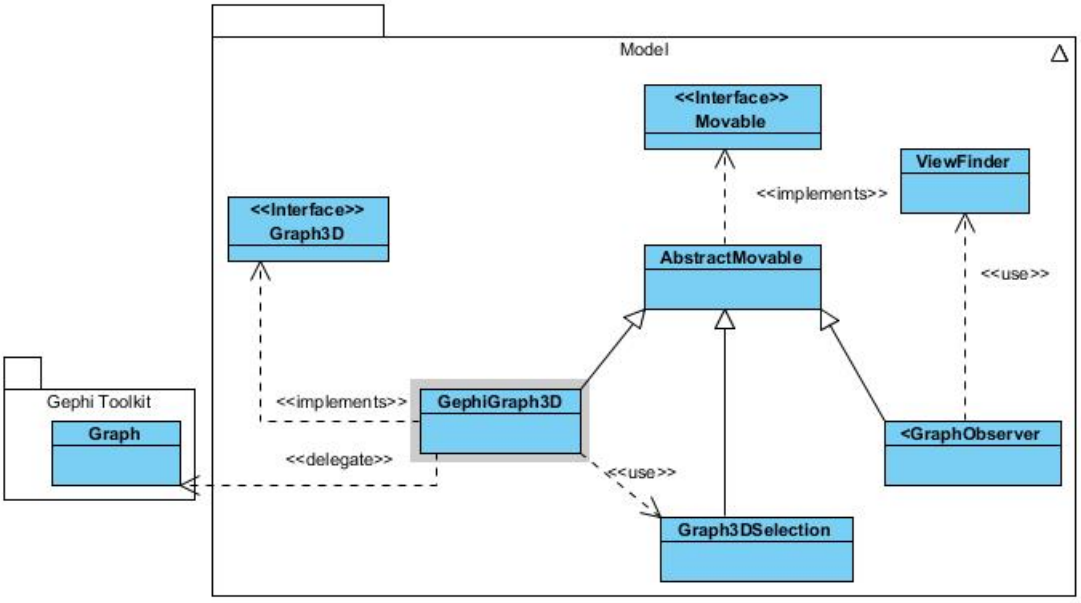


Figura 4.2: Diagramma delle classi per il sottosistema *Model*

4.3 La GUI Java e la Netbeans Platform

Il sottosistema *View* si occupa di visualizzare l'interfaccia utente. Esso è l'unico sottosistema ad interagire con la *Netbeans Platform* ed è costituito da due componenti principali: la classe *MainPanel* e la classe *Graph3DTopComponent*.

La classe *MainPanel* è una estensione della classe *JPanel* della libreria *Java Swing*. Essa mostra tutti gli elementi dell'interfaccia utente ed in particolare mostra l'area su cui viene disegnata la scena 3D. L'area di disegno è rappresentata dell'oggetto *GLCanvas* che è una particolare estensione dell'oggetto *Canvas* del pacchetto *AWT* di *Java* fornita dalla libreria *Java OpenGL (JOGL)*.

La classe *Graph3DTopComponent* estende la classe *TopComponent* del framework *Netbeans Platform* e rappresenta il componente grafico principale che viene visualizzato nella *GUI* di *Gephi*. Esso racchiude il *MainPanel* e si occupa di fornire alla *Netbeans Platform* i metodi per la gestione degli eventi relativi allo sganciamento-agganciamento della finestra di *Graph 3D Explorer* dall'interfaccia principale di *Gephi*. In Figura 4.3 è riportato il diagramma delle classi che riassume le relazioni appena descritte.

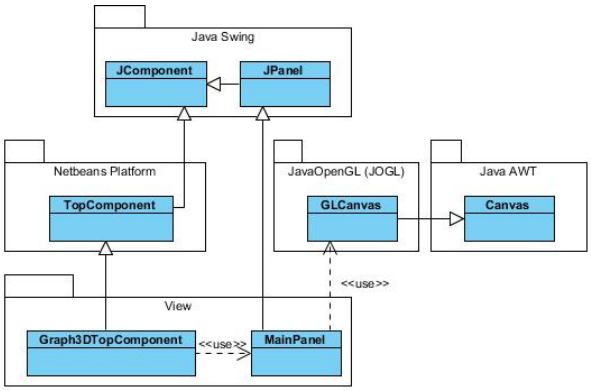


Figura 4.3: Diagramma delle classi per il sottosistema *View*. Da questo diagramma si nota che il framework *Netbeans Platform* è, di fatto, una astrazione della libreria *Java Swing*

4.4 Il livello di coordinamento

Il sottosistema *Controller* si occupa di coordinare i vari sottosistemi. Esso è costituito da tre classi principali: *Graph3DPlugin*, *GraphManager* e *ProjectManager*.

La classe *Graph3DPlugin* è un *Singleton*[35] e mantiene un riferimento al *Model*, al *View*, ai dispositivi di input e al motore di *Rendering*. E' accessibile da ogni parte del sistema e permette la comunicazione tra i vari sottosistemi.

La classe *GraphManager* è adibita alla creazione dell'oggetto *Proxy GephiGraph3D* interagendo con il *GraphController* del *Gephi Toolkit* da cui ottiene l'oggetto *Graph* necessario per la costruzione. Inoltre, ha la funzione di *Listener* per le modifiche che avvengono al grafo. Infatti, il *proxy* lavora sull'istanza *Graph* del *Gephi Toolkit* che è unica e condivisa tra tutti i plugin di *Gephi*. E' importante sottolineare che anche le azioni di modifica che l'utente esegue tramite *Graph 3D Explorer* vengono gestite con questo sistema di notifica perché di fatto anche esse generano delle modifiche sull'istanza condivisa dell'oggetto grafo del *Gephi Toolkit*.

Infine, la classe *ProjectManager* ha lo scopo di notificare al *Graph3DPlugin* gli eventi di apertura e chiusura di un progetto in *Gephi*. Per far ciò essa implementa l'interfaccia *WorkspaceListener* del *Gephi Toolkit* e si registra sul *ProjectController* (sempre del *Gephi Toolkit*) per essere notificata.

La classe *ProjectController* ha il compito di gestire l'apertura e chiusura dei progetti e fornisce dei metodi informativi per capire se c'è un progetto correntemente attivo oppure per aprirne uno nuovo o chiudere quello correntemente aperto. In Figura 4.4 è riportato il diagramma delle classi che riassume le relazioni descritte. Invece, in Figura 4.5 è riportato un esempio di interazioni tra le tre classi principali del sottosistema *Controller* e il *Gephi Toolkit*. Infine, in Figura 4.6 è dato un esempio di interazioni tra gli oggetti per la gestione di un evento di modifica al grafo correntemente attivo.

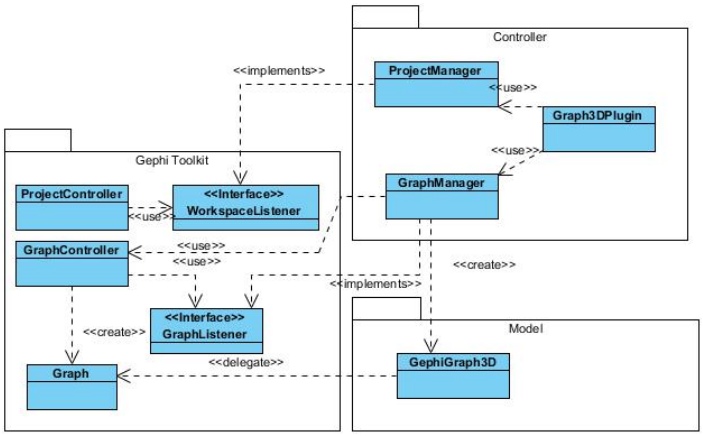


Figura 4.4: Diagramma che descrive le relazioni che intercorrono tra le classi principali del sottosistema *Controller* e quelle maggiormente usate del *Gephi Toolkit*

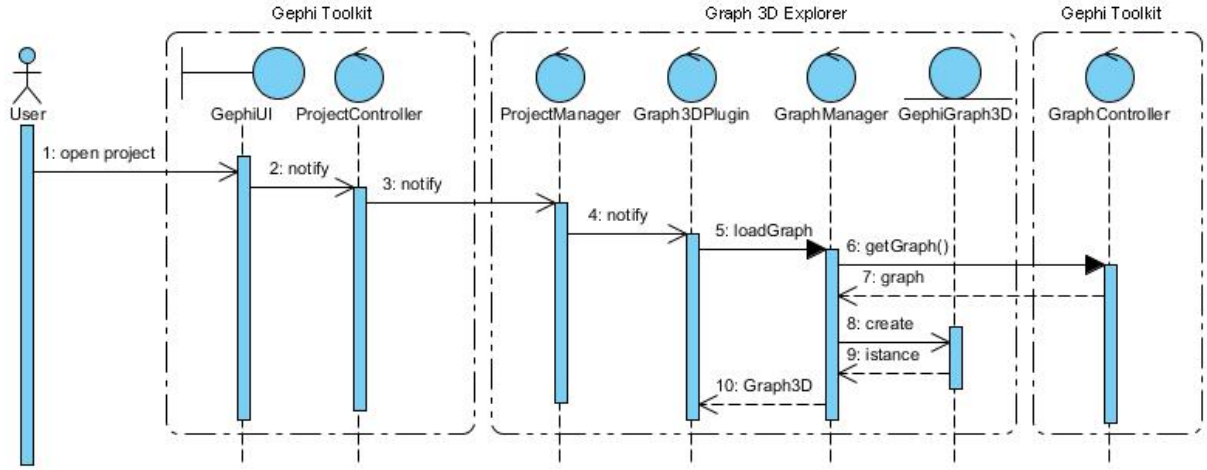


Figura 4.5: Diagramma che descrive ad alto livello la sequenza di interazioni tra gli oggetti che si ha all'apertura di un progetto in *Gephi* se il plugin *Graph 3D Explorer* è attivo. In particolare, l'oggetto *Graph Manager* interagisce con il *GraphController* del *Gephi Toolkit* da cui ottiene il grafo correntemente attivo che poi successivamente passa al costruttore del proxy *GephiGraph3D*.

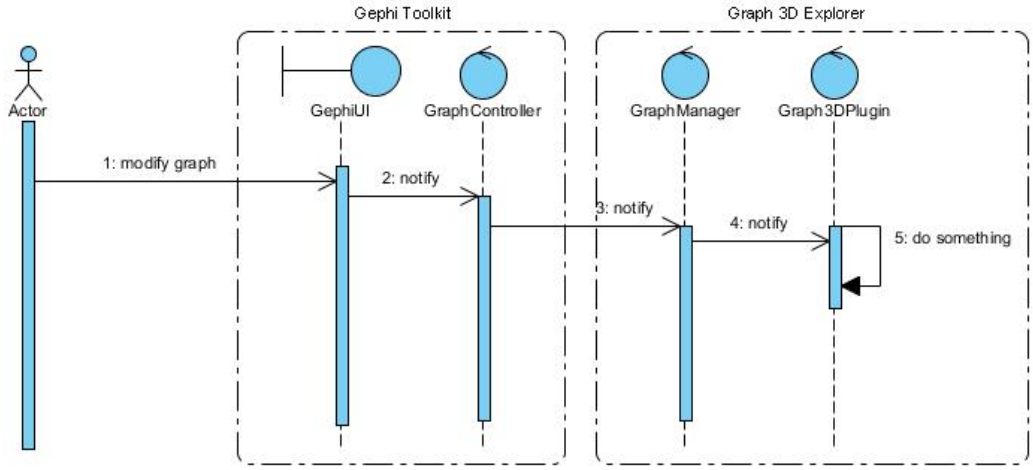


Figura 4.6: Diagramma che descrive il sistema ad eventi di *Gephi*. In questo esempio la classe *Graph Manager* viene notificata delle modifiche all'istanza di grafo correntemente attiva e a sua volta provvede ad informare la classe *Graph3DPlugin* che decide come reagire all'evento che è accaduto.

4.5 Il Rendering: OpenGL e JOGL

Il sottosistema *Rendering* si occupa di disegnare la scena 3D ed è una componente cruciale per il sistema. E' qui che ci si interfaccia con la scheda grafica e in cui si deve prestare molta attenzione alle ottimizzazioni se si vogliono delle buone prestazioni anche con grafi massivi. A tal fine è stato usata la libreria *Java OpenGL (JOGL)* che permette l'utilizzo delle API OpenGL in ambienti basati su tecnologie Java. In questo sottosistema le classi fondamentali sono *RenderingEngine*, *NodeRenderer*, *EdgeRenderer* e *Camera*,.

La classe *RenderingEngine* si occupa di coordinare le operazioni aggiornando il modello e facendo partire la visita del grafo. Le classi *Node Renderer* e *EdgeRenderer* si occupano del disegno dei nodi e degli archi rispettivamente. In particolare, per i nodi e per la schermata dettagli è usata la tecnica del *Billboarding*[36] che consiste nell'uso di quadrati bidimensionali che sono orientati sempre verso la camera e su cui sono applicate delle immagini. Per la rappresentazione degli archi sono state usate le primitive atte al disegno di linee offerte da OpenGL. Per lo scambio dati tra *CPU* e *GPU* sono stati usati i *Vertex Array* e i *Vertex Buffer* offerti da *OpengGL*.

I *Vertex Array* permettono di raggruppare tutti i dati relativi ai vertici

da disegnare in dei buffer in modo da ottimizzare l'utilizzo del *BUS* di sistema e quindi aumentare le prestazioni. Invece, i *Vertex Buffer* consentono la memorizzazione dei dati direttamente sulla memoria della scheda grafica e sono ottimi in caso si hanno dei dati statici. Nell'applicazione sviluppata, i *Vertex Array* sono stati utilizzati per l'invio delle posizioni dei nodi che rappresentano una componente dinamica del sistema mentre i *Vertex Buffer* sono stati utilizzati per la gestione di dati statici quali ad esempio le coordinate di *texture* usate per la tecnica del *BillBoarding*.

Un altro aspetto importante di questo sottosistema riguarda la visualizzazione del testo. Per tale scopo è stata usata la classe *TextRenderer* della libreria *JOGL* che converte stringhe java in texture OpenGL offrendo buone prestazioni e completa compatibilità con i caratteri *Unicode*. Inoltre, un aspetto positivo per la classe *TextRenderer* risiede nelle sue politiche di *caching* automatiche per le texture generate allo scopo di evitare continue conversioni del testo. Ultima ma non meno importante è la classe *Camera* che si occupa dell'applicazione della proiezione 3D in OpenGL leggendo lo stato dell'oggetto *GraphObserver* del sottosistema *Model*. In Figura 4.7 è rappresentato il diagramma delle classi relativo a questo sottosistema.

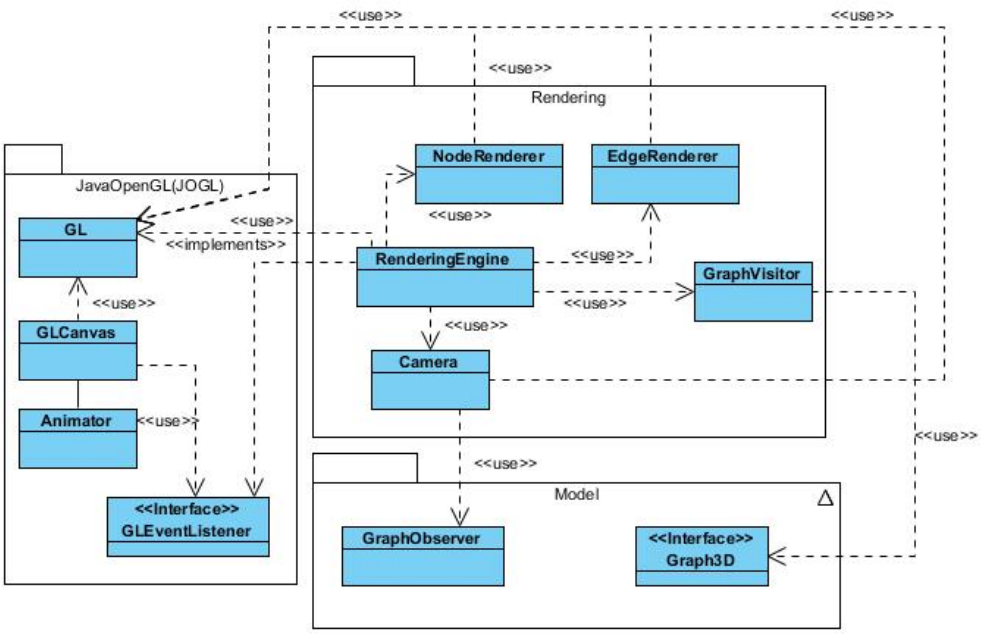


Figura 4.7: Diagramma delle classi per il sottosistema *Rendering*. Ci si interfaccia con JOGL tramite la classe *RenderingEngine* che implementa l'interfaccia *GLEventListener*. Il *GLEventListener* è associato all'oggetto *GLCanvas* che rappresenta l'area di disegno su cui si va a rappresentare la scena 3D. Il *GLCanvas* offre un metodo per il suo aggiornamento che fa uso dei metodi dell'interfaccia *GLEventListener* e che viene continuamente invocato in un ciclo infinito gestito dalla classe *Animator*

4.6 Il supporto a Mouse e Tastiera

In Figura 4.8 è riportato il diagramma delle classi coinvolte nella gestione di mouse e tastiera. Per la gestione di questi dispositivi di input ci si è serviti del meccanismo dei *Listener* di *Java AWT*. Infatti, la classe *GLCanvas* della libreria *JOGL* estende la classe *Canvas* di *Java AWT*. La gestione degli eventi segue un paradigma di *produttore-consumatore* in quanto gli eventi sono prodotti nel *thread* di *AWT* ma vengono consumati nel *thread* destinato al rendering. In particolare, la classe *AWTInputSystem* implementa le interfacce *MouseListener* e *KeyListener* per la ricezione degli eventi che vengono delegati alle classi *MouseInputDevice* e *KeyboardInputDevice* le quali gestiscono quelli relativi a *Mouse* e *Tastiera* rispettivamente e sono indipendenti da *Java AWT*. Entrambe le classi mantengono una coda concorrente in cui vengono salvati gli eventi e implementano l'interfaccia *InputEventProducer* i cui metodi vengono invocati nel *thread* di rendering per consumare gli eventi prodotti.

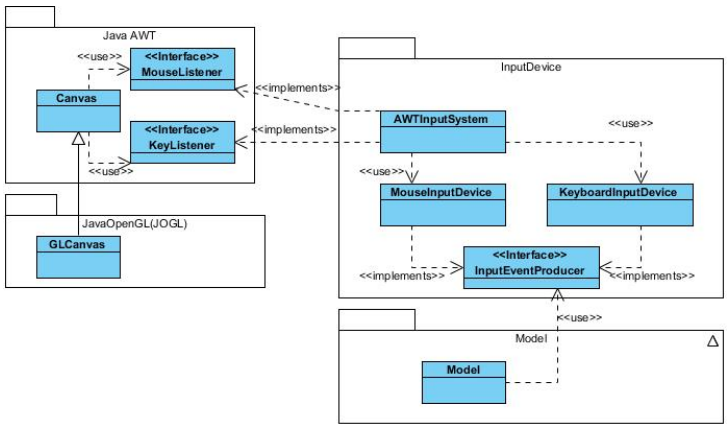


Figura 4.8: Diagramma delle classi per la gestione di *Mouse e Tastiera*. L'interfaccia *InputEventProducer* ha la funzione di disaccoppiare il sottosistema di *InputDevice* da quello *Model*.

4.7 Il supporto al Joypad

Il sistema offre la possibilità di navigare ed interagire con l'ambiente virtuale usando il *Joypad* della *XBOX* di *Microsoft*. Per interagire con questa periferica è stata usata la libreria *JInput* che permette di accedere a qualsiasi

Joypad collegato al computer grazie alla classe *DirectAndRawInputEnvironmentPlugin*. Questa classe legge lo stato delle periferiche *USB* ed offre un metodo *getController()* che permette di accedere a qualsiasi *Joypad* collegato al computer. Il metodo *getController()* restituisce una lista di oggetti *Controller* che astraggono il *Joypad* e offrono dei metodi per capire quali pulsanti sono premuti in un certo istante e in che direzione sono orientate le levette analogiche.

La classe *Controller* associa ad ogni tasto del Joypad una chiave numerica. L'associazione tra tasto e chiave numerica costituisce il *layout* della periferica e varia da modello a modello. La classe *Controller* offre un metodo *poll()* che restituisce una mappa i cui elementi sono coppie chiave-valore dove la chiave è il codice numerico del pulsante e il valore è un booleano che se assume valore vero indica che il tasto in quel momento risulta premuto dall'utente. La gestione delle levette analogiche differisce da quella dei pulsanti in quanto esse non assumono valore discreto ma continuo dato che possono essere mosse lungo l'asse X (sinistra - destra) o lungo l'asse Y (basso - alto).

La classe *Controller* permette di accedere alle due levette del *Joypad* identificando come *LeftStick* la levetta sinistra e come *RightStick* quella destra. Lo stato di una specifica levetta può essere letto grazie ai metodi *getXValue* e *getYValue* che restituiscono un valore reale x nell'intervallo $[-1, 1]$. Valori di x prossimi allo 0 indicano che la levetta non è stata mossa. Valori di x prossimi ad 1 indicano un movimento della levetta nella direzione positiva dell'asse considerato mentre valori prossimi a -1 indicano un movimento della levetta nella direzione negativa. Ad esempio se invochiamo il metodo *getXValue* su una qualsiasi delle due levette e ci viene restituito un valore x maggiore di 0.60 allora vuol dire che la levetta è mossa verso destra (direzione positiva dell'asse X). Di contro, un valore minore di -0.60 indica un movimento della levetta verso sinistra (direzione negativa dell'asse X). Questo discorso è analogo per i movimenti alto-basso deducibili grazie al metodo *getYValue* dove bisogna sostituire il movimento in alto a quello verso destra e il movimento in basso a quello verso sinistra.

La classe *Controller* è stata racchiusa nella classe *Joystick* che implementa l'interfaccia *InputEventProducer* ed ha il compito di identificare gli eventi di pressione e rilascio dei pulsanti e di associargli delle azioni da eseguire sul modello. Tali azioni vengono eseguite quando la classe *Model* legge lo stato di tutti i dispositivi di input ed invoca il metodo *doAction* dell'interfaccia *InputEventProducer*.

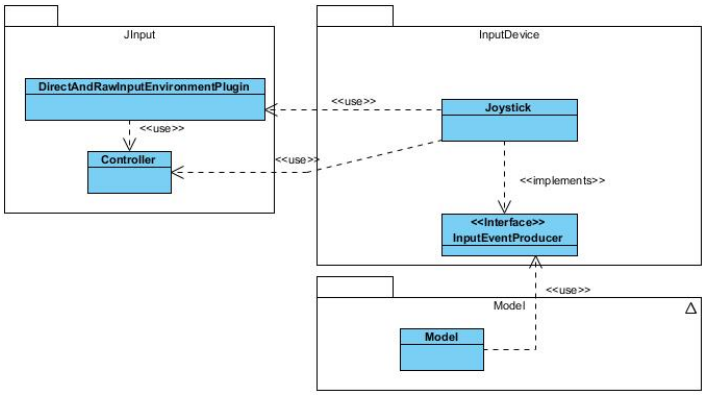


Figura 4.9: Diagramma delle classi per la gestione del *Joypad*. L'interfaccia *InputEventProducer* ha la funzione di disaccoppiare il sottosistema di *InputDevice* da quello *Model*.

4.8 Il supporto alla periferica Leap Motion

La periferica *Leap Motion* è un dispositivo *USB* che fa uso di tecnologie ad infrarossi e di algoritmi di analisi dell'immagine per tenere traccia dei movimenti delle mani e delle singole dita. In questa sezione faremo dapprima una panoramica sul funzionamento generale della periferica e poi analizzeremo come essa è stata integrata nel sistema.

4.8.1 Una panoramica sulle API

La periferica USB *Leap Motion* grazie alle tre camere di cui è dotata produce un flusso continuo di immagini che viene inviato ad un servizio che gira in *background* sul sistema operativo che si occupa di analizzarle.

Una applicazione può far uso delle informazioni di tracking grazie all'insieme di librerie (*SDK*) multipiattaforma fornito dalla casa produttrice che offre la possibilità di programmare in vari linguaggi come il *C*, *Java*, *Javascript* e *Python*. In merito al sistema realizzato è stato usato l'insieme di librerie che permette l'utilizzo della periferica dal linguaggio *Java*. Uno schema rappresentativo dell'architettura descritta è dato in Figura 4.10 mentre in Figura 4.11 è mostrato il flusso di dati.

L'oggetto principale del Leap Motion SDK è dato dalla classe *Controller* che astrae la periferica *USB* ed interagisce direttamente con il servizio in *background*. Le informazioni relative a ciò che è presente nel campo d'a-

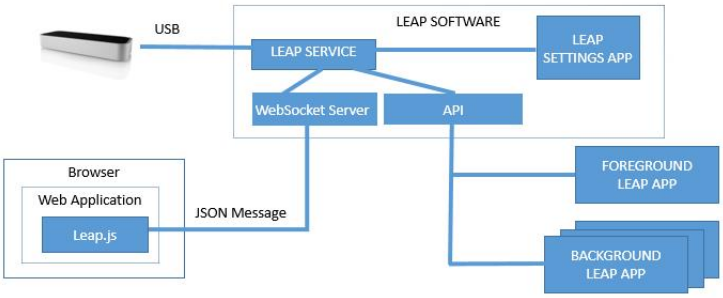


Figura 4.10: Schema dell'architettura del Leap Motion. Il software del leap motion gira sul sistema operativo. Le applicazioni (sia quelle in background che quelle attive) possono accedere ai dati tramite le API. E' possibile usare i dati anche nelle applicazioni web sfruttando le API fornite che sono basate sulle websocket.



Figura 4.11: Il flusso di dati. Le immagini rilevate dalle camere del Leap Motion arrivano sottoforma di dati grezzi al Leap Service che li processa e genera gli oggetti *Frame* che sono utilizzabili nelle varie API offerte.

zione della periferica in un determinato istante di tempo sono incapsulate nell'oggetto *Frame*.

Mediamente vengono prodotti sessanta oggetti *Frame* al secondo che vengono mantenuti dall'oggetto *Controller* utilizzando un buffer. L'oggetto *Controller* permette l'accesso ai *Frame* mediante l'invocazione diretta del metodo *frame()* oppure mediante un sistema di notifica basato su *Listener*.

I *Frame* contengono informazioni relative alle mani e alle dita. Le mani si muovono in un sistema di riferimento tridimensionale che ha l'origine fissata sulla periferica (Figura 4.12) e sono astratte dall'oggetto *Hand* che permette l'accesso alla posizione del palmo, delle punta delle dita e della direzione della mano e offre metodi per calcolare la rotazione della mano rispetto al sistema di riferimento definito dalla periferica Leap Motion (Figura 4.13).

Le API permettono il riconoscimento di alcuni gesti molto semplici: *Swipe*, *Screen Tap*, *Key Tap*. Lo *Swipe* è un movimento lineare di un dito. Con una mano aperta possono esserci più gesti di *Swipe* mentre con una mano

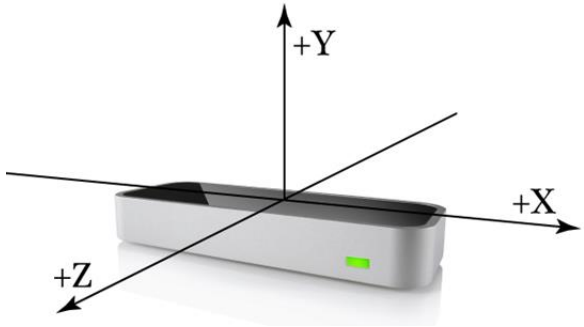


Figura 4.12: Le mani nel campo d'azione della periferica Leap Motion vengono considerate in un sistema di riferimento tridimensionale centrato sul dispositivo.



Figura 4.13: Leap Motion: alcune informazioni relative alla mano. Il vettore palmNormal è usato per calcolare le rotazioni della mano rispetto all'asse Z

chiusa non possono esserci dato che le dita non sono estese. Lo *Screen Tap* è un movimento di un dito in avanti come se si facesse un *Tap* su uno schermo *Touch*. Il *Key Tap* è un gesto che simula la pressione di un pulsante mentre il *Circle* indica dei movimenti circolari di un dito nell'aria. Questi quattro gesti base sono riportati in Figura 4.14.

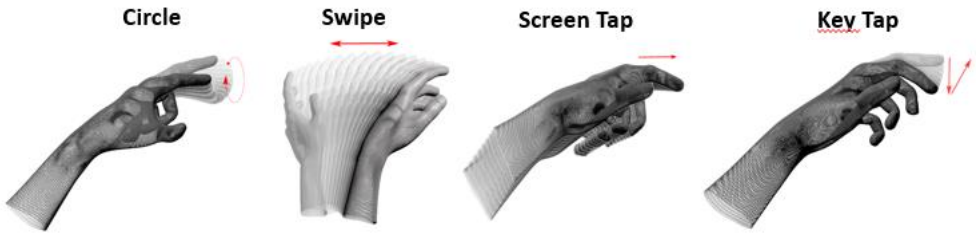


Figura 4.14: I quattro gesti base riconoscibili con le API del Leap Motion

Nelle nuove versioni dell'*SDK* (2.0+) le API permettono di distinguere tra mano destra e mano sinistra. Inoltre, la mano ha un modello scheletrico che fornisce varie informazioni sulla posizione e la curvatura delle falangi. Una panoramica sul modello scheletrico è data in Figura (4.15). Sempre dalla versione 2.0, le API forniscono alcune informazioni circa l'accuratezza del tracking e dei metodi appositi per il livello di chiusura del palmo che possono essere usati per il riconoscimento di pose quali pugno o pizzico. Ulteriori dettagli in merito alle API 2.0 sono disponibili nella documentazione ufficiale[37].

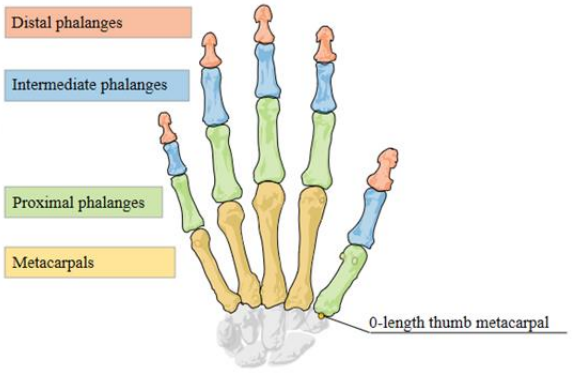


Figura 4.15: Il modello scheletrico fornito dalle API del Leap Motion a partire dalla versione 2.0

4.8.2 Integrazione della periferica Leap Motion nel sistema sviluppato

Nell'implementazione corrente di *Graph 3D Explorer* vengono usati i listener per rilevare la connessione della periferica *Leap Motion* al sistema. L'oggetto *Controller* poi viene interrogato nel ciclo di rendering tramite il metodo *frame()* dall'oggetto *LeapInputDevice* che implementa anche esso come le l'interfaccia *InputEventProducer* come le classi per la gestione di *Mouse*, *Tastiera* e *Joypad*.

Riconoscimento di gesti

Per il riconoscimento dei gesti sono state implementate le classi *HandAnalyzer* e *HandHistory*. La classe *HandAnalyzer* si occupa di analizzare lo stato delle mani che sono nel campo d'azione del *Leap Motion* mentre la classe *HandHistory* mantiene lo storico di una specifica mano.

Lo stato degli oggetti che sono in un certo istante di tempo i nel campo d'azione della Leap Motion è rappresentato dall'oggetto $Frame_i$ che è ottenibile invocando il metodo *frame()* sull'oggetto *Controller*.

Gli oggetti *Hand* hanno un id univoco. Lo stato di una mano H_j che è presente nel campo d'azione della *Leap Motion* all'istante i è rappresentato dall'oggetto $Hand_{j,i}$ che è ottenibile dall'oggetto $Frame_i$.

L'identificazione dei gesti è un processo che dipende dalle variazione di stato degli oggetti *Hand* nel tempo. Ad ogni iterazione i del ciclo di rendering, l'*HandAnalyzer* prende l'oggetto $Frame_i$ dalla classe *Controller*. Dall'oggetto $Frame_i$ vengono presi gli oggetti $Hand_{j,i}$ con $j = 1, .., n$ dove n è il numero di mani correntemente rilevate nel campo d'azione.

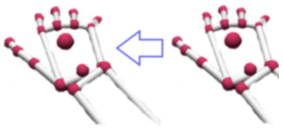
Ogni oggetto $Hand_{j,i}$ è confrontato con l'oggetto $Hand_{j,k}$ con $k < i$. che rappresenta lo stato in cui era la mano H_j $i - k$ istanti prima ed è ottenibile invocando il metodo *frame()* della classe *Controller* con parametro k (*Controller.frame(k)*).

Andiamo ora a considerare i movimenti lineari della mano che, come descritto nel capitolo precedente, servono per traslare la camera o i nodi selezionati. Per questo tipo di movimenti (riportati per comodità di nuovo nella Figura 4.16) è confrontata la variazione della posizione del palmo della mano H_j tra gli oggetti $Hand_{j,i}$ e $Hand_{j,k}$. Siano $p_{j,i}$ e $p_{j,k}$ le posizioni del palmo della mano H_j all'istante i e all'istante k ($k < i$). $p_{j,i}$ e $p_{j,k}$ sono vettori tridimensionali e possono essere denotati come $(x_{j,i}, y_{j,i}, z_{j,i})$ e $(x_{j,k}, y_{j,k}, z_{j,k})$ per mettere in risalto le coordinate x y e z. Considerando adesso le variazioni sui tre assi $\Delta_{x,j} = x_{j,i} - x_{j,k}$, $\Delta_{y,j} = y_{j,i} - y_{j,k}$ e

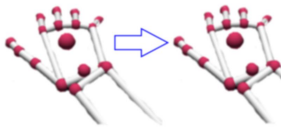
$\Delta_{z,j} = z_{j,i} - z_{j,k}$ andiamo a denotare come $\Delta_{max,j}$ il massimo tra queste tre quantità. $\Delta_{max,j}$ determina l'asse su cui l'oggetto va traslato. Infatti se $\Delta_{max,j} = \Delta_{x,j}$ allora l'oggetto deve traslare lungo l'asse x. Se $\Delta_{max,j} = \Delta_{y,j}$ allora l'oggetto deve traslare lungo l'asse y mentre se vale $\Delta_{max,j} = \Delta_{z,j}$ allora l'oggetto deve traslare lungo l'asse z. Scelto l'asse di spostamento, la direzione (negativa o positiva) è data dal segno di $\Delta_{max,j}$.

Consideriamo adesso le rotazioni della mano (Figura 4.17) che, come detto nel capitolo precedente, servono per ruotare la camera quando è attiva la modalità *Hand Rotation*. Le rotazioni del palmo a sinistra e a destra corrispondono ad una rotazione della mano sull'asse z rispetto al sistema di riferimento della periferica *Leap Motion* mentre le rotazioni del palmo verso l'alto e verso il basso corrispondono a rotazioni della mano sull'asse x. Considerando la mano H_j e l'iterazione i, dall'oggetto $Hand_{j,i}$ possiamo ottenere di quanto la mano è ruotata rispetto l'asse z in quell'istante. Chiamiamo questa rotazione come $roll_{j,i}$. Se $roll_{j,i}$ è maggiore di th o minore di $-th$ allora l'oggetto sarà ruotato di $roll_{j,i}$ radianti lungo l'asse z. Il valore di th indica la soglia oltre il quale la mano va considerata ruotata e può essere calibrata. La rotazione lungo l'asse x segue lo stesso principio. Da questo algoritmo implica il fatto che più la mano è ruotata più la rotazione dell'oggetto avviene velocemente e se l'utente rimane fermo con la mano abbastanza inclinata l'oggetto continua a ruotare. Questo tipo di approccio rende agevoli le rotazioni di 180°.

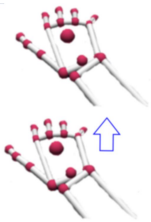
Per gli altri tipi di movimenti sono usati metodi delle *API* (2.0). Infatti, per il pugno chiuso (Figura 4.19) è usato il metodo *getGrabStrength()* sull'oggetto *Hand* che restituisce un valore x nell'intervallo $[0, 1]$. Se x è vicino ad 1 allora la mano è chiusa altrimenti è aperta. Quindi, considerando una mano H_j e denotando il valore del metodo *getGrabStrength()* invocato all'iterazione i con $GS_{j,i}$, se $GS_{j,i} > th_{GS}$ allora la mano è considerata come chiusa nell'istante i. th_{GS} è un valore nell'intervallo $[0, 1]$ e rappresenta una soglia configurabile. In merito alla posa di indicazione (Figura 4.20) è usato il modello scheletrico. Infatti, da un oggetto *Hand* è possibile ottenere gli oggetti *Finger* che hanno un identificativo da 0 a 4 che indica il tipo di dito (pollice=0, indice=1, medio=2 , anulare=3, mignolo=4) e un metodo informativo *isExtended()* che permette di capire se l'oggetto *Finger* rappresenta un dito esteso o contratto. Quindi una mano H_j all'istante i è considerata in posa di indicazione se l'oggetto *Finger* con id uguale ad 1 risulta esteso e tutti gli altri risultano contratti.



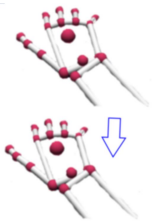
(a) Movimento lineare a sinistra



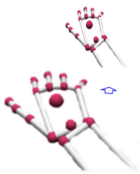
(b) Movimento lineare a destra



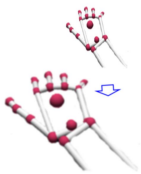
(c) Movimento lineare in alto



(d) Movimento lineare in basso

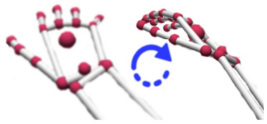


(e) Movimento lineare in avanti

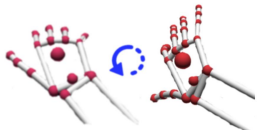


(f) Movimento lineare all'indietro

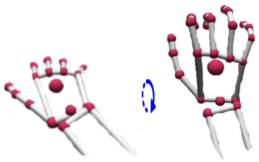
Figura 4.16: Questa figura riporta i movimenti lineari presi in considerazione. Essi possono essere compiuti a mano aperta o chiusa oppure con il solo dito indice esteso. Se fatti a mano aperta come riportato in figura permettono il controllo della camera. Se invece sono fatti a mano chiusa permettono lo spostamento dei nodi selezionati nelle rispettive direzioni indicate dalle frecce. Se fatti con il dito indice esteso allora permettono la rotazione del grafo. Considerando i movimenti a mano aperta, se la modalità di controllo della camera è *Linear* allora tutti i movimenti lineari diversi da quelli in avanti e indietro sono usati per ruotare la camera. Da notare che i movimenti lineari a mano chiusa e ad indice esteso non sono riportati in quanto totalmente analoghi a quelli raffigurati.



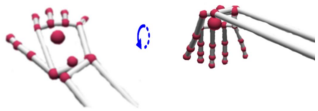
(a) Rotazione del palmo a sinistra



(b) Rotazione del palmo a destra



(c) Rotazione del palmo verso l'alto



(d) Rotazione del palmo verso il basso

Figura 4.17: La figura indica le rotazioni della mano per il controllo della camera nella modalità *Hand Rotation*. Il palmo segue la rotazione opposta alle frecce indicanti il movimento della mano. In particolare, rotazioni del palmo verso il basso e verso l'alto fanno ruotare la camera verso il basso e verso l'alto rispettivamente. Analogamente, le rotazioni del palmo verso destra e sinistra fanno ruotare la camera nelle medesime direzioni.

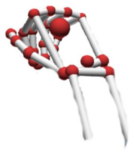


Figura 4.18: Postura per effettuare la selezione dei nodi

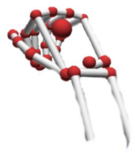


Figura 4.19: Postura per effettuare la selezione dei nodi

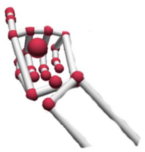


Figura 4.20: Postura per iniziare la rotazione del grafo o uscire dalla visualizzazione dettagli

4.9 La visita del grafo, la visualizzazione delle informazioni e l'area di selezione

La rappresentazione di una scena avviene disegnando una sequenza di immagini fisse che, susseguendosi velocemente, danno all'occhio umano la percezione che gli oggetti visualizzati siano in movimento. Per realizzare ciò è usato un ciclo infinito (*Rendering Loop*) in cui ad ogni iterazione si compiono due passi: *update* e *rendering*. Ad ogni iterazione (*Frame*) viene associato un id (*FrameID*).

Nella fase di *update* viene chiamato il metodo *update* sull'istanza della classe *Model* correntemente attiva che ha il compito di leggere lo stato dei dispositivi di input e di aggiornare la posizione dell'osservatore rappresentata dalla classe *GraphObserver* e la dimensione dell'area di selezione circolare rappresentata dall'oggetto *ViewFinder*.

Successivamente, sempre nel metodo *update* dell'oggetto *Model* viene invocato il metodo *update* sull'oggetto *Graph3D* in cui viene dapprima aggiornato l'orientamento del grafo applicando i comandi dei dispositivi di input e poi fatta eseguire la visita del grafo.

La visita del grafo è adibita alla classe *GraphVisitor*. Qui è stato utilizzato il design pattern Strategy[35]. Infatti, l'algoritmo di visita utilizzato viene isolato all'interno di un oggetto *VisitorAlgorithm* che può essere cambiato a runtime.

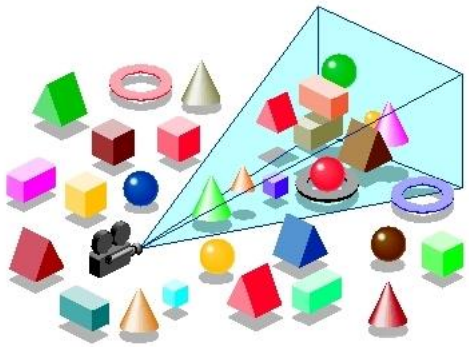
Nell'attuale algoritmo di visita si scorrono i nodi e gli archi in un unico ciclo di complessità $O(max(n, m))$ dove n è il numero di nodi del grafo mentre m è il numero di archi. Infatti ad ogni iterazione del ciclo vengono visitati un arco e un nodo. La visita di un arco consiste nel leggere i due nodi su cui esso giace. Nel nodo (classe *Node3D*) viene mantenuto l'ultimo *FrameID* in cui è stato visitato per evitare che venga letto due volte nello stesso *Frame*. Durante la visita, un nodo viene sottoposto a tre test: *Visibility Test*, *Label Test* e *Selection Test* che saranno analizzati più nel dettaglio di seguito. Alla fase di *update* segue quella di *rendering* in cui vengono mandati i dati relativi alle posizioni dei nodi alla scheda grafica tramite le API OpenGL e le classi del sottosistema di *Rendering*.

4.9.1 I test sui nodi

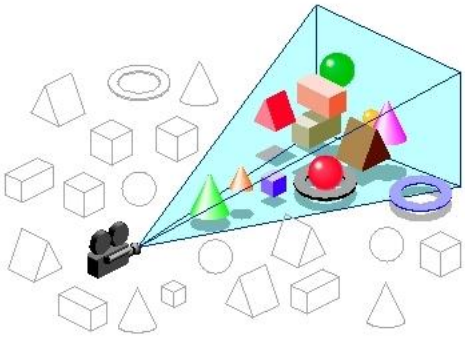
In questa sezione descriveremo più nel dettaglio i tre test a cui vengono sottoposti i nodi durante la fase di *update*.

Visibility Test

Il *Visibility Test* è effettuato dalla classe *Camera* del sottosistema di *Rendering* che, considerando la posizione dell'osservatore (classe *GraphObserver*), determina il *Frustum*[36] e calcola se il nodo è visibile oppure no applicando tecniche di *Frustum Culling*[36]. Una schematizzazione di questo processo è data in Figura 4.21.



(a) Prima dell'applicazione del Frustum Culling



(b) Dopo dell'applicazione del Frustum Culling

Figura 4.21: La tecnica del Frustum Culling: tagliare tutto quello che non rientra nella porzione di spazio visibile dalla camera.

Label Test

Se il nodo è visibile, allora è effettuato il *Label Test* dalla classe *LabelDrawner* con lo scopo di controllare se il nodo è ad una distanza tale da poterne visualizzare l'etichetta. Se questo test ha esito positivo allora il nodo viene marcato come *LabelVisible* per quel *Frame* e l'oggetto *String* che rappresenta l'etichetta viene salvato in una lista *LabelList* dell'oggetto *LabelDrawner*. Solo se il *Label Test* ha successo allora il nodo viene sottoposto anche al *Selection Test* che consiste nel controllare se esso ricade nell'area di selezione circolare. In caso positivo il nodo viene marcato come *Selectable* per quel *Frame*. Per questioni di efficienza, nel misurare la lontananza tra le posizioni è usata la distanza euclidea al quadrato. Considerando $p = (x_p, y_p, z_p)$ come la posizione dell'osservatore e $q = (x_q, y_q, z_q)$ come la posizione del nodo da testare, la distanza euclidea al quadrato è data dalla seguente formula:

$$d(p, q) = (x_p - x_q) + (y_p - y_q) + (z_p - z_q)$$

Selection test

In Figura 4.22 è schematizzato il funzionamento dell'area di selezione. In particolare, l'area di selezione $selArea$ è rappresentata come un cerchio di raggio $r_{selArea}$ di fronte all'osservatore. Alla posizione p dell'osservatore viene sommata la distanza euclidea $distProj = \sqrt{d(p, q)}$ che c'è tra essa e la posizione q del nodo. La nuova posizione ottenuta $pr_{p,q}$ e la direzione dell'osservatore d formano un piano $plane_{pr_{p,q},d}$ su cui è delineata la circonferenza $selArea$. Sia $Circle_q$ la circonferenza con raggio r_q centrata nella posizione q del nodo. Se la circonferenze $Circle_q$ e $selArea$ si intersecano allora il nodo è selezionabile. Per verificare l'intersezione tra $selArea$ e la circonferenza $Circle_q$ del nodo da testare è usato il seguente test booleano:

$$d(pr_{p,q}, q) < r_q^2 \;||\; d(pr_{p,q}, q) < r_{selArea}^2$$

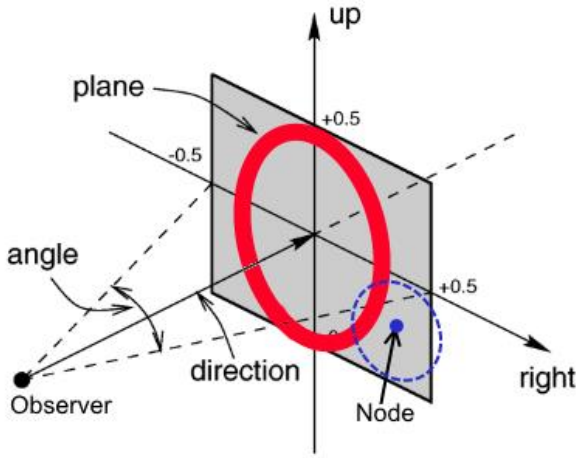


Figura 4.22: Funzionamento dell'area di selezione. In questa figura è data una rappresentazione grafica del funzionamento dell'area di selezione che è rappresentata dalla circonferenza rossa. La circonferenza blu tratteggiato rappresenta il nodo da testare. Se le due circonferenze si intersecano allora il nodo è selezionabile.

4.10 L'integrazione dell'Oculus Rift

La periferica Oculus Rift è uno schermo indossabile che include un giroscopio, un accelerometro e un magnetometro. Unendo i dati ricevuti da queste componenti può determinare l'orientamento della testa della persona che lo sta indossando e consentire la sincronizzazione della prospettiva della camera in modo da permettere un controllo realistico. L'orientamento della testa è riportato come un insieme di rotazioni in un sistema di coordinate destrorso come in Figura 4.23.

Oltre a permettere il tracking dei movimenti della testa, l'Oculus Rift permette una visualizzazione immersiva grazie al sistema di lenti integrato e al rendering stereoscopico lato software. Il tracking della testa e la visualizzazione 3D rendono l'Oculus Rift sia un dispositivo di input che un dispositivo di output e portano a dover interagire sia sul sottosistema di *Input* che sul sottosistema di *Rendering*. L'SDK ufficiale per l'Oculus Rift è in *C++* ma esistono molte librerie che ne permettono l'utilizzo anche con altri linguaggi. In merito al sistema sviluppato è stata usata la libreria open source JOVR[38] che fornisce l'accesso alle API ufficiali tramite linguaggio *Java*.

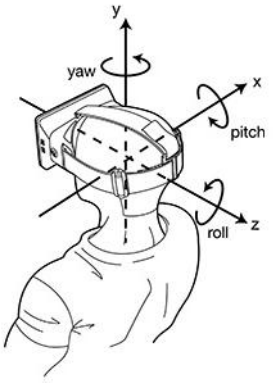


Figura 4.23: Schema delle rotazioni riportate dalla periferica Oculus Rift.

Le API non sono molto complesse. Infatti, c'è un unica classe *HMD* che permette la connessione con la periferica e la lettura delle informazioni relative all'orientamento della testa dell'utilizzatore tramite un metodo *getOrientation()* che restituisce un quaternione. Ad ogni iterazione del ciclo di rendering viene interrogato l'oggetto *HMD* e il quaternione viene applicato all'oggetto *GraphObserver* che espone un metodo *setOrientation()* che prende come parametro un quaternione di rotazione. Utilizzando l'*Oculus Rift*, tutti gli altri dispositivi di input non possono impartire rotazioni alla camera.

La parte più complessa dell'integrazione risiede nel rendering stereoscopico. Infatti, per avere l'effetto 3D bisogna generare un immagine per ogni occhio. Nel sistema realizzato, il grafo viene visitato una sola volta e nel corso della visita vengono riempiti i buffer con i dati da mandare alla *GPU*. Successivamente la scena viene disegnata due volte con due differenti *viewport* e con delle proiezioni leggermente diverse a seconda dell'occhio su due texture utilizzando i *FrameBuffer* di *OpenGL* e realizzando così quello che viene denotato come *Rendering Offscreen*. Queste due texture vengono poi passate all'oggetto *HMD* per l'applicazione di uno *shader* che distorce l'immagine in modo da farla adattare alla curvatura delle lenti per creare l'effetto tridimensionale desiderato. Considerando l'implementazione attuale, la classe *Oculus* si occupa sia del rendering che dell'orientamento ed è a cavallo tra il sottosistema di *Rendering* e quello di *Input*.

Capitolo 5

Il Sistema realizzato: sperimentazione

In questo lavoro di tesi si è sviluppato *Graph 3D Explorer*: plugin per *Gephi* che permette l'esplorazione e la manipolazione di grafi in un ambiente virtuale e interattivo. Nei precedenti capitoli sono state descritte le funzionalità che esso offre e sono stati dati dettagli sull'implementazione e sulle tecnologie utilizzate. In questo capitolo mostreremo come è stata valutata l'efficacia del sistema e l'usabilità delle varie modalità di utilizzo proposte.

5.1 Obiettivi dello studio

I motivi che ci hanno spinto alla realizzazione del sistema proposto sono stati il bisogno di interattività nella visualizzazione di grafi e la necessità di stimolare l'utente all'esplorazione per favorire il processo di *Information Retrieval*. A tal scopo sono state introdotte modalità di interazione basate sulle interfacce naturali utente e la realtà virtuale. Quindi si è voluto capire il grado di soddisfazione dell'utenza nell'utilizzo di queste modalità di interazione in termini di divertimento e interattività e relazionarlo con quello che si ha utilizzando metodologie standard.

In particolare, la domanda che si siamo posti è stata "Le modalità di interazioni innovative basate sulle interfacce naturali utente sono più stimolanti rispetto ai classici mouse e tastiera nella esplorazione di grafi?"

Il fattore divertimento e interattività è collegato all'opinione dell'utenza. Quindi ci siamo chiesti "Gli utenti ritengono che sia una buona idea usare interfacce basate sui gesti per la navigazione di dati strutturati in un am-

biente tridimensionale?" e "Cosa si deve fare per favorire l'utilizzo di queste modalità di interazione innovative?"

Per rispondere a queste domande è stato condotta una sperimentazione con sessanta studenti dell'università di Salerno a cui sono stati assegnati una serie di compiti da svolgere con il sistema sviluppato utilizzando diverse modalità di interazione e successivamente sono state chieste impressioni e consigli in merito tramite dei questionari. I questionari sono stati analizzati con metodi statistici e sono state tratte alcune conclusioni relativamente alle domande sopra esposte.

5.2 Reclutamento degli utenti e configurazioni testate

Sessanta studenti dell'Università degli Studi di Salerno sono stati coinvolti nella sperimentazione sfruttando le conoscenze dirette, il passaparola e le associazioni studentesche. Gli studenti sono stati distribuiti tra le sei configurazioni che il sistema sviluppato rende disponibili come riportato nella tabella in Figura 5.1. In particolare denoteremo come *Traditional Configuration (TConf)* la configurazione che fa uso di Mouse e Tastiera insieme al monitor tradizionale. Indicheremo come *Virtual Reality Configuration (VRConf)* la configurazione che fa uso di Leap Motion e Oculus Rift. 20 utenti del campione hanno testato sia la configurazione *TRConf* che la configurazione *VRConf* per consentire un comparativa accurata e per poter rispondere alle domande di ricerca esposte nella sezione precedente.

Input/Output	Monitor	Oculus Rift
Mouse Tastiera	20*	10
Joypad	10	10
Leap Motion	10	20*

Figura 5.1: Le sei configurazioni testate. Il test ha coinvolto 60 utenti in totale. Nella cella della tabella è indicato il numero di utenti che ha testato quella configurazione. Le configurazioni agli estremi della tabella sono state testate dagli stessi 20 utenti.

5.3 Procedure utilizzate

Per la valutazione di una configurazione, l'utente è stato dapprima invitato alla compilazione di un questionario preliminare in forma anonima atto a raccogliere informazioni demografiche e alcune informazioni di carattere generale come il livello di esperienza con l'informatica, la passione per i videogiochi e quanto tempo giocano. Dopo la compilazione del questionario preliminare l'utente è stato invitato ad eseguire tre task con il sistema sviluppato utilizzando una delle sei configurazioni disponibili. Al completamento dei tre task l'utente è stato invitato alla compilazione di un secondo questionario valutativo.

Per far sì che le risposte possano essere analizzate con metodi statistici è stata usata la scala *Likert*[39]: una tecnica per la misura dell'atteggiamento che consiste nel mettere a punto un certo numero di affermazioni che esprimono un atteggiamento positivo o negativo verso un oggetto. L'utente può associare a queste affermazioni un valore numerico che varia entro una certa scala e che esprime quanto è d'accordo o non d'accordo con l'affermazione.

I tre task eseguiti dall'utente sono i seguenti:

- *Ricerca.* Nel grafo sono presenti 4 nodi neri. L'utente deve trovarli entro un certo tempo e avvicinarsi ad essi per visualizzarne l'etichetta.
- *Esplorazione.* Esplorazione libera del grafo con un tempo massimo.
- *Clustering.* Nel grafo sono presenti 2 gruppi di nodi grigi. L'utente deve trovarli e raggrupparli.

Con il task *Ricerca* si è voluto testare le capacità di orientamento dell'utente. Infatti, l'utente era invitato a guardare il grafo da lontano e a guardarsi intorno per la ricerca dei nodi. Con il task *Esplorazione* si è voluto testare le capacità di movimento con la configurazione proposta. Infatti, all'utente veniva richiesto di visualizzare quante più etichette possibili nel grafo possibilmente cercando zone dense. L'ultimo test del *Clustering* è stato usato per testare l'efficacia della configurazione nell'esecuzione di azioni complesse. Infatti il raggruppamento richiede prima la selezione e questo con le configurazioni che fanno uso della periferica *Leap Motion* richiede un interazione a due mani. In tutti e tre i task l'utente non era a conoscenza del tempo massimo (5 minuti per *Ricerca* e *Clustering.* 2 minuti per il task di *Esplorazione*). Ciò ha permesso di valutare il grado di coinvolgimento dell'utente considerando la sua consapevolezza circa il tempo trascorso durante l'utilizzo.

Dopo l'esecuzione dei tre task è stato chiesto all'utente di valutare alcuni aspetti con le seguenti quattro domande:

1. Quanto è stato semplice?
2. Il sistema ti è sembrato reattivo durante l'esecuzione del task?
3. Quanto ti sono sembrate naturali le interazioni necessarie all'esecuzione dei task?
4. Hai avuto qualche disturbo durante l'esecuzione del task?

Le risposte alle prime tre domande sono in scala *Likert* a 5 intervalli mentre per l'ultima sono ammesse più riposte.

Successivamente alle domande relative ai task, sono stati valutati alcuni aspetti generali del sistema e della particolare configurazione con il *Technology Acceptance Model (TAM)*. Il modello *TAM* è stato creato da Davis[40] nel 1989 basandosi su ricerche nel campo della psicologia ed è stato esteso prima da *Moon* e *Kim* nel 2002[41] per applicarlo in contesti inerenti il *WWW* e poi da *Lui et al.* nel 2010[42] per applicarlo in contesti di *E-Learning* . E' un modello molto usato per spiegare o fare predizioni circa la diffusione di nuove tecnologie o di nuovi sistemi che si basa su cinque concetti fondamentali:

- *Perceived Usefulness (PU)* ovvero quanto è utile secondo l'utente la tecnologia o il sistema proposto.
- *Perceived Easy of Use (EOU)* ovvero quanto è di facile utilizzo il sistema o la tecnologia secondo l'utente.
- *Perceived Playfulness (PP)* ovvero quanto l'utente trova divertente o coinvolgente l'utilizzo della tecnologia o del sistema proposto.
- *ATTitude toward using (ATT)* ovvero la predisposizione che l'utente ha verso la tecnologia.
- *Behavioral Intention to use (BI)* ovvero le percezioni o le sensazioni dell'utente che indicano se l'utente userà in futuro la tecnologia o il sistema proposto.

Le domande usate per la valutazione del *TAM* sono riportate in Figura 5.2 e le risposte sono in scala *Likert* a 7 intervalli.

	L'utilizzo di questa configurazione:	Fortemente disaccordo						Fortemente d'accordo
PU1	mi consente di svolgere i compiti più velocemente	1	2	3	4	5	6	7
PU2	migliora le mie prestazioni	1	2	3	4	5	6	7
PU3	migliora l'efficacia del mio lavoro	1	2	3	4	5	6	7
PU4	rende più semplice il mio lavoro	1	2	3	4	5	6	7
PU5	È utile per il mio lavoro/esperienze	1	2	3	4	5	6	7

(a) Domande relative al *Perceived Usefulness (PU)*

		Fortemente disaccordo						Fortemente d'accordo
EOU1	Imparare ad usare questo sistema è stato semplice	1	2	3	4	5	6	7
EOU2	E' stato semplice svolgere le azioni che desideravo	1	2	3	4	5	6	7
EOU3	Le interazioni erano chiare e comprensibili	1	2	3	4	5	6	7
EOU4	Il sistema si è dimostrato flessibile	1	2	3	4	5	6	7
EOU5	Acquisire le competenze per usare il sistema è stato semplice	1	2	3	4	5	6	7
EOU6	In generale, il sistema è stato semplice da usare	1	2	3	4	5	6	7

(b) Domande relative al *Perceived Easy of Use (EOU)*

	Utilizzando il sistema proposto	Fortemente disaccordo						Fortemente d'accordo
PP1	non ho percepito il tempo trascorso	1	2	3	4	5	6	7
PP2	niente mi ha disturbato	1	2	3	4	5	6	7
PP3	mi sono divertito	1	2	3	4	5	6	7
PP4	la mia curiosità è stata stimolata	1	2	3	4	5	6	7
PP5	sono stato indotto ad una maggiore esplorazione	1	2	3	4	5	6	7
PP6	ha stimolato la mia immaginazione	1	2	3	4	5	6	7

(c) Domande relative al *Perceived Playfulness (PP)*

	Usare questo sistema è una idea							
ATT1	cattiva/buona	Cattiva						Buona
		1	2	3	4	5	6	7
ATT2	sciocca/saggia	Sciocca						Saggia
		1	2	3	4	5	6	7
ATT3	non piacevole/piacevole	Non Piacevole						Piacevole
		1	2	3	4	5	6	7
ATT4	negativa/positiva	Negativa						Positiva
		1	2	3	4	5	6	7

(d) Domande relative al *Attitude Toward Using (ATT)*

		Fortemente disaccordo						Fortemente d'accordo
BI1	Avendone la possibilità, userò questo sistema in futuro	1	2	3	4	5	6	7
BI2	Nel contesto specifico, raccomanderò ad altri l'uso di questo sistema	1	2	3	4	5	6	7

(e) Domande relative al *Behavioral Intention to Use (BI)*

Figura 5.2: Domande usate per la valutazione del *TAM*

5.4 Configurazione tradizionale contro realtà virtuale

Per rispondere alla domanda "Le modalità di interazioni innovative basate sulle interfacce naturali utente sono più stimolanti rispetto ai classici mouse e tastiera nella esplorazione di grafi?" sono state messe a confronto diretto la configurazione che fa uso di mouse e tastiera tradizionale (*Traditional Configuration*) e la configurazione innovativa composta da *Oculus Rift* e *Leap Motion*. Per far ciò è stato considerato un campione casuale di 20 utenti che ha testato entrambe le configurazioni.

5.4.1 Questionario generale

Dal questionario preliminare il campione è risultato prevalentemente maschile e composto da studenti di età media 24 anni iscritti a facoltà scientifiche (Figura 5.3).

Sesso		
	Numero	%
Maschi	18	90
Femmine	2	10
SOMMA	20	100

	Numero	%
Mancino	2	10
Problemi visivi	9	45
Occhiali	11	55

(a) Percentuali di maschi e femmine

(b) Occhiali e mancinismo

Laurea		
	Numero	%
Triennale	4	20
Magistrale	16	80
SOMMA	20	100

Corso di studi		
	Numero	%
Informatica	17	85
Ingegneria Elettronica	1	5
Chimica e Tecnologie Farmaceutiche	1	5
Matematica	1	5
SOMMA	20	100

Carattere del corso di studi		
	Numero	%
Scientifico	20	100
Umanistico	0	0
SOMMA	20	100

(c) Corsi di studio

Figura 5.3: Dati generali relativi al campione di 20 utenti

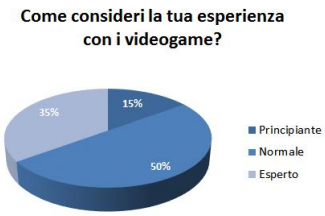
Uso dei videogiochi

In merito all'uso dei videogiochi, tutti i venti studenti hanno dichiarato di esserne abbastanza appassionati (Figura 5.4(a)). In particolare il 35% del campione si è considerato giocatore esperto (Figura 5.4(b)). Per quanto riguarda le ore di gioco (Figura 5.4(c)), il 50% del campione ha dichiarato di giocare ai videogiochi tra le 1-7 ore a settimana mentre l'altro 50% si divide in un 35% che gioca dalle 8 alle 14 ore settimanali e un 15% che gioca per oltre 14 ore settimanali. I generi maggiormente giocati sono riportati in Figura 5.4(d) da cui spiccano i giochi di ruolo *RPG* e gli sparatutto in prima persona *FPS*. A tutti i partecipanti è stato chiesto quali fossero le principali periferiche di input utilizzate per giocare ai videogiochi e l'80% ha dichiarato di utilizzare abitualmente *Mouse e Tastiera* (Figura 5.4(e)).

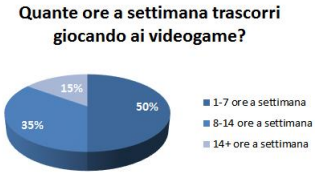
Le risposte in merito alle ore di gioco, alla passione e all'esperienza nell'uso dei videogiochi sono state date in input all'algoritmo di clustering *k-means* da cui è risultato che il campione è composto per il 55% da giocatori occasionali (*Casual Gamer*) e per il 55% da giocatori assidui (*Gamer*). Tale categorizzazione è riportata in Figura 5.5.



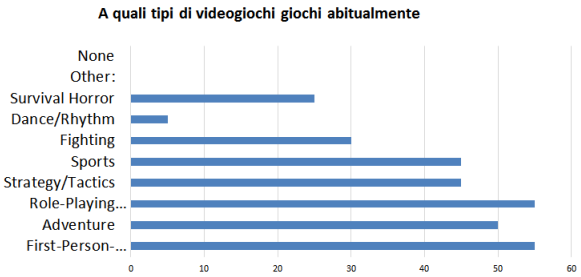
(a) Interesse per i videogiochi



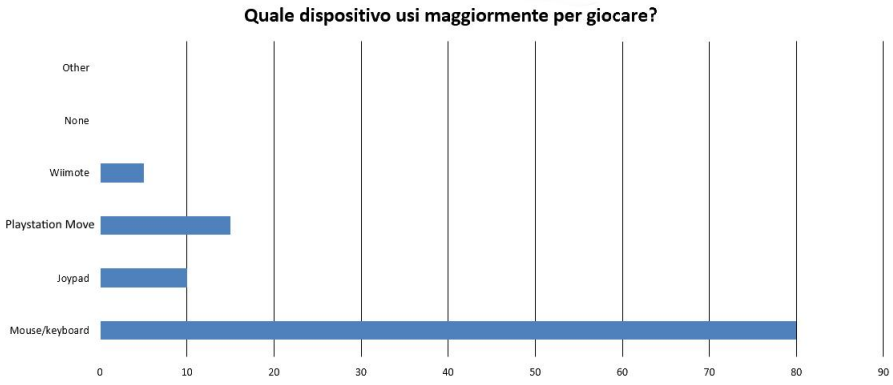
(b) Livello di esperienza



(c) Ore di gioco settimanali



(d) Generi maggiormente giocati



(e) Periferiche maggiormente utilizzate

Figura 5.4: Uso dei videogiochi

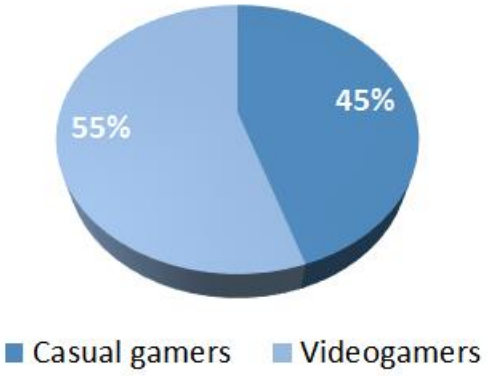


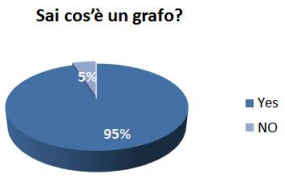
Figura 5.5: Classificazione del campione in giocatori assidui (*Gamer*) e giocatori casuali (*Casual Gamer*)

Esperienze nel campo IT e con i grafi

In merito all'esperienza nel campo IT, il 65 % si ritiene competente (Figura 5.6). Per quanto riguarda i grafi, il 95% del campione conosce la materia (Figura 5.7(a)), il 60% si ritiene competente (Figure 5.7(b)) ed ha avuto necessità di interazione e visualizzazione riscontrando come principale problematica l'eccesso informazioni (Figure 5.7(c) e 5.7(d)).



Figura 5.6: Livello di esperienza nel campo IT



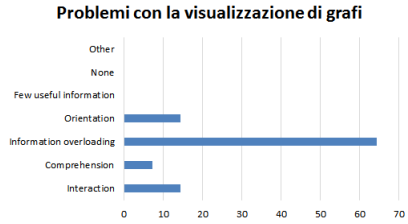
(a) Conoscenza dei grafi



(b) Livello di competenza con i grafi



(c) Percentuale di utenti che ha avuto bisogno di visualizzare un grafo

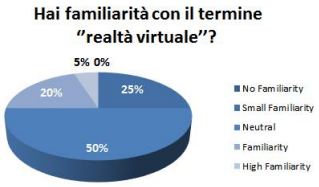


(d) Problemi maggiormente riscontrati con le visualizzazioni

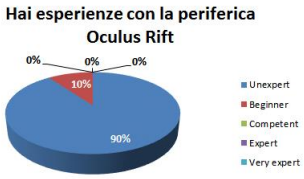
Figura 5.7: Esperienza con i grafi e problemi comuni con le visualizzazioni

Esperienze con la realtà virtuale

Nelle Figura 5.8 sono riportate le risposte degli utenti in merito alla conoscenza della realtà virtuale da cui si evince che il campione considerato non ha particolari esperienze (Figura 5.8(a)) e solo il 10% conosce la periferica *Oculus Rift* (Figura 5.8(b)).



(a) Familiarità con la realtà virtuale



(b) Livello di esperienza con la periferica Oculus Rift

Figura 5.8: Livello di esperienza del campione con la realtà virtuale e la periferica Oculus Rift

5.4.2 I tre task

Semplicità

Le medie sulle risposte degli utenti in merito alla semplicità dei task sono riportate in Figura 5.9. In generale i task non sono risultati difficili da eseguire in quanto i punteggi sono sempre maggiori di 3.5. Però c'è una differenza statisticamente significativa tra le due configurazioni. Infatti, la configurazione *Virtual Reality* (identificata come *Oculus* in Figura 5.9) risulta essere meno facile da utilizzare.

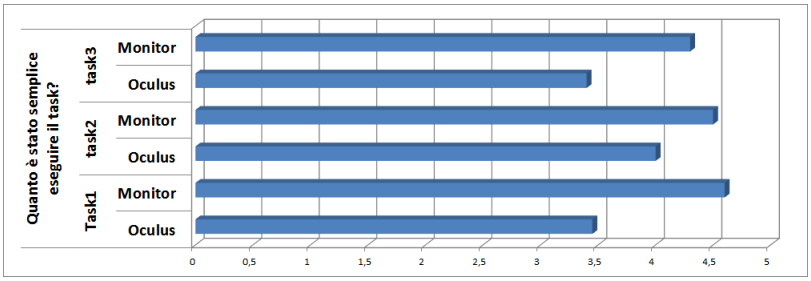


Figura 5.9: Semplicità dei task

Disturbi riscontrati durante l'esecuzione dei task

In merito ai disturbi riscontrati durante l'esecuzione dei task, con la configurazione tradizionale l'87% dell'utenza non ha avuto problemi mentre il restante 13% ha riscontrato limitatezza dei movimenti dovuta principalmente al fatto che alcuni utenti sono abituati ad utilizzare mouse da *gaming* ad alta precisione.

Invece, con la configurazione *Virtual Reality* il 37% in media ha percepito limitatezza dei movimenti e alta sensibilità in tutti e tre i task. Tali problematiche sono attribuibili principalmente al campo d'azione limitato della periferica Leap Motion e alla poca esperienza del campione con questa tipologia di interazioni.

Reattività del sistema e naturalezza delle interazioni

Le medie sulle risposte in merito alla reattività del sistema durante l'esecuzione dei tre task sono riportate in Figura 5.10. In generale il sistema è stato percepito come molto reattivo. Infatti, su una scala da 0 a 5 tutte le medie sono vicine al 4. Inoltre, per confermare la positività del risultato sono

stati fatti test statistici e non è emersa nessuna differenza statisticamente significativa tra le due configurazioni.

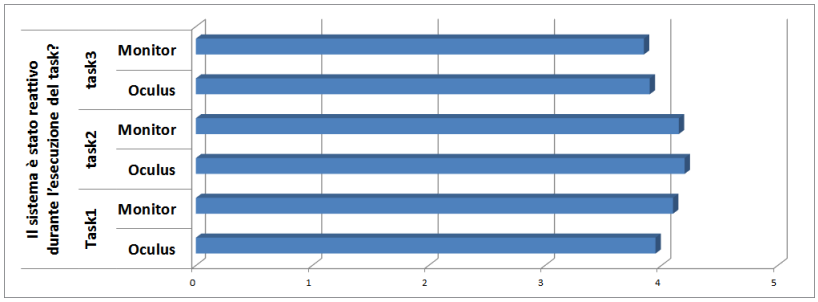


Figura 5.10: Reattività del sistema durante l'esecuzione dei task

Dalla Figura 5.11 si evince che anche la naturalezza delle interazioni è stata giudicata abbastanza buona per entrambe le configurazioni. Ciò principalmente è dovuto al fatto che l'utente riconduce i meccanismi di interazione a qualcosa che conosce. Infatti, per la configurazione tradizionale l'utente videogiocatore si rifà ai meccanismi dei videogiochi. Invece, per la configurazione *Virtual Reality* l'utente riconduce i meccanismi di interazione a delle azioni che sono naturali nel mondo reali quali, ad esempio, cambiare la zona che si sta guardando girando la testa oppure afferrare un oggetto chiudendo la mano.

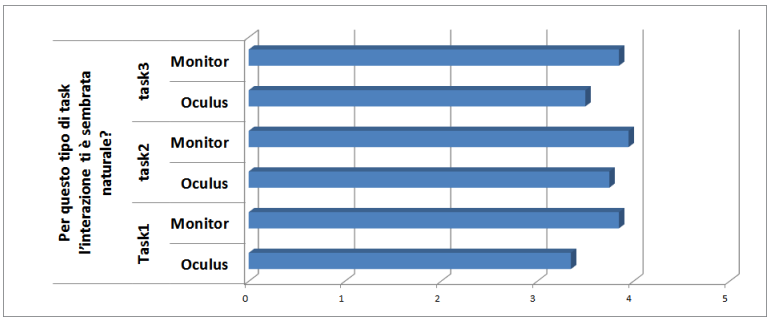


Figura 5.11: Naturalezza delle interazioni per l'esecuzione dei tre task

Tempi di esecuzione e precisione per i tre task

Le sessioni degli utenti sono state monitorate e i dati principali sono riportati in Figura 5.12. In particolare, sono stati misurati i tempi in secondi per l'esecuzione dei tre task e la precisione. Nel caso della ricerca, la precisione è data dal numero di nodi trovati mentre nel caso del clustering è data dal numero di cluster fatti. Dalle tabelle in Figura 5.12 si evince che i tempi di esecuzione sono molto più elevati per la configurazione *Virtual Reality* mentre dal punto di vista della precisione non c'è una differenza significativa tra le due configurazioni. C'è invece una differenza statisticamente significativa per il numero di nodi visualizzati durante il task di esplorazione che risulta più alto per la configurazione tradizionale.

Le prestazioni peggiori della configurazione *Virtual Reality* dipendono da vari fattori tra cui l'inesperienza del campione con questa tipologia di interazioni e l'elemento distrazione introdotto dalla novità. Infatti, molti utenti nonostante fossero stati incoraggiati a terminare i task il prima possibile ci hanno messo più tempo perché hanno voluto interagire con l'ambiente e esplorare la rete.

Task	TConf	VRConf	P-value
Clustering	34,2	138,6	P < 0.001
Search	25,7	135,9	P < 0.001

(a) Tempi di esecuzione in secondi dei tre task

Task	TConf	VRConf	P-value	OPT
Clustering	3	3	N.S.	2
Search	4	3,85	N.S.	4

(b) Precisione nei task di ricerca e clustering

Task	TConf	VRConf	P-value
Nodi visitati	693	272	P < 0.001

(c) Nodi visualizzati durante l'esplorazione

Figura 5.12: Dati relativi ai tempi di esecuzione e alla precisione dei tre task

5.4.3 Risultati del Tam

Le medie sulle risposte in merito alle domande per la valutazione del modello *Tam* sono riportate in *Figura* 5.13. La configurazione tradizionale e quella di *Virtual Reality* hanno valori medi di *BI* simili che portano ad una buona predisposizione all'utilizzo futuro per entrambe le configurazioni. Tuttavia, la configurazione tradizionale allo stato attuale è percepita come più utile (PU) e come più semplice (EOU) rispetto a quella di *Virtual Reality* che però risulta essere più divertente (PP). Le differenze tra i vari parametri sono state confrontate statisticamente con il test di Wilcoxon per verificarne la significatività. Da questi controlli statistici è risultato che le differenze sulla utilità percepita (PU) e sull'utilizzo futuro (BI) non sono statisticamente significative mentre sono statisticamente significative quelle sul divertimento (PP) in favore della configurazione *Virtual Reality* ($p < 0.005$) e quelle sulla semplicità (EOU) a favore della configurazione tradizionale ($p < 0.001$).

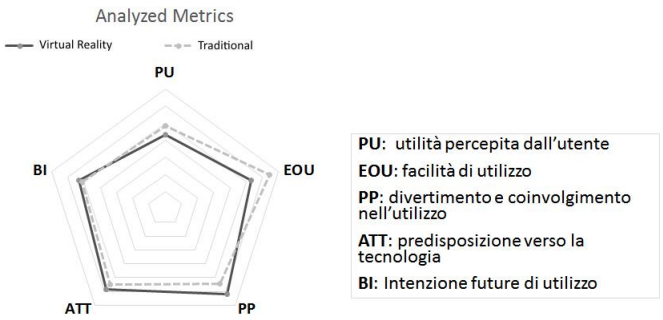


Figura 5.13: Risultati del modello *TAM*

Predittore per il *Behavioral Intention to use (BI)*

Uno degli obiettivi di questa sperimentazione consiste nel capire cosa bisogna fare per far sì che queste metodologie di interazione innovative vengano utilizzate effettivamente dagli utenti. A tal fine, si è voluto studiare quale fosse il predittore per il *Behavioral Intention to use (BI)* che rappresenta appunto la predisposizione all'utilizzo futuro da parte dell'utente di una determinata configurazione. Tramite tecniche di regressione lineare è stato trovato che l'utilità percepita (PU) risulta essere il miglior predittore per il BI in entrambe le configurazioni testate (Figura 5.14).

	Virtual reality configuration	Traditional configuration
Variabile predittore	P value	P value
Constant	.725	.583
Videogamer	.328	.805
Usefulness (PU)	.015	.001
Ease Of Use (EOU)	.967	.082
Attitude (ATT)	.277	.253
Playfulness (PP)	.795	.943

Figura 5.14: Predittore per il *Behavioral Intention to use (BI)*. Il miglior predittore per il *BI* è la utilità percepita *PU* che è segnata in verde in figura

la correlazione tra *Perceived Usefulness (PU)* e *Perceived Easy of Use (EOU)*

Il risultato di Figura 5.14 ci permette di dire che più un utente ritiene utile una configurazione più è propenso ad utilizzarla in futuro. In virtù di questo si è voluto indagare su quali fattori influenzano l'utente e fanno sì che una configurazione sia vista come più utile rispetto ad un altra. A tal scopo si è andato ad analizzare la correlazione statistica tra le risposte date dagli utenti in merito ai cinque concetti del modello *TAM*. Dalla tabella riportata in Figura 5.16 si nota che la facilità di utilizzo *EOU* è in correlazione positiva con la utilità percepita *PU* in entrambe le configurazioni. Questo ci permette di affermare che aumentando la semplicità dei meccanismi di interazione per una determinata configurazione si avrà un aumento della percezione di utilità della stessa e quindi se ne favorirà l'utilizzo futuro.

Traditional configuration

Scala	PU	EOU	ATT	PP	BI
PU	1.0				
EOU	0,596	1.0			
ATT	0,808	0,637	1.0		
PP	N.S	N.S	0.614	1.0	
BI	0,897	0,506	0,795	N.S	1.0

Virtual reality configuration

Scala	PU	EOU	ATT	PP	BI
PU	1.0				
EOU	0,512	1.0			
ATT	N.S	N.S	1.0		
PP	0,488	N.S	0.565	1.0	
BI	0,712	0,543	N.S	N.S	1.0

Figura 5.15: La correlazione positiva tra *Perceived Usefulness (PU)* e *Perceived Easy of Use (EOU)* è evidenziata in rosso nella figura.

la correlazione tra *ATTitude toward using (ATT)* e *Perceived Usefulness (PU)*

Un altro fattore che influenza l'utilità percepita (PU) è l'attitudine ATT. Tuttavia, l'attitudine si manifesta soltanto con l'esperienza di utilizzo. Infatti, guardando le correlazioni tra i cinque concetti abbiamo che nella configurazione tradizionale la correlazione tra ATT e PU si manifesta mentre nella configurazione *Virtual Reality* essa non è significativa in quanto il campione non ha una forte esperienza pregressa nell'utilizzo dell'*Oculus Rift* e della periferica *Leap Motion*. Questa osservazione ci permette di ipotizzare che la familiarità e la pratica con determinati meccanismi di interazione fanno sì che essi siano considerati più utili rispetto ad altri. In virtù di ciò potremmo ipotizzare che lavorando sulla semplicità e facendo familiarizzare meglio l'utente con la configurazione *Virtual Reality* questa possa essere considerata più utile e possa essere maggiormente utilizzata in futuro rispetto a quella tradizionale in quanto già allo stato attuale risulta più divertente e stimolante.

Traditional configuration

Scala	PU	EOU	ATT	PP	BI
PU	1.0				
EOU	0,596	1.0			
ATT	0,808	0,637	1.0		
PP	N.S	N.S	0.614	1.0	
BI	0,897	0,506	0,795	N.S	1.0

Virtual reality configuration

Scala	PU	EOU	ATT	PP	BI
PU	1.0				
EOU	0,512	1.0			
ATT	N.S	N.S	1.0		
PP	0,488	N.S	0.565	1.0	
BI	0,712	0,543	N.S	N.S	1.0

Figura 5.16: La correlazione positiva tra *Perceived Usefulness (PU)* e *ATTitude toward using (ATT)* è evidenziata in rosso nella figura.

Post Survey

Gli utenti, a prescindere dalla configurazione utilizzata, hanno ritenuto che una esplorazione immersiva di una rete di dati possa essere utile non solo ad esperti della teoria dei grafi (Figura 5.17).

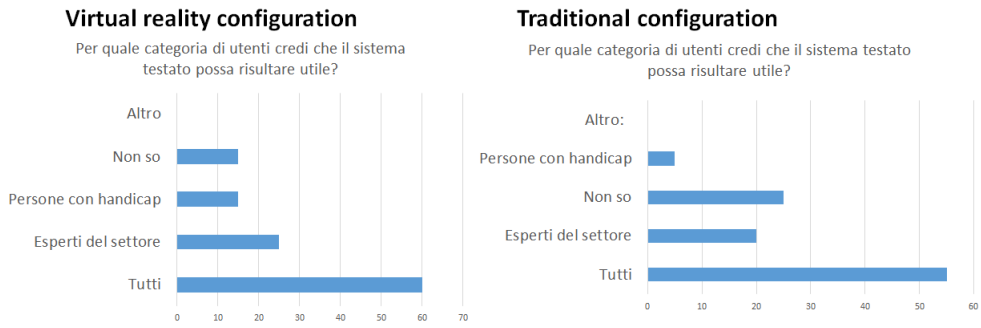


Figura 5.17: Categorie di utilizzatori possibili per il sistema proposto

Con le risposte alle altre domande del Post Survey (Figura 5.18) gli utenti hanno valutato positivamente in termini di semplicità e utilità tutte le configurazioni testate ed hanno indicato come più divertente la configurazione *Virtual Reality* rispetto a quella tradizionale.

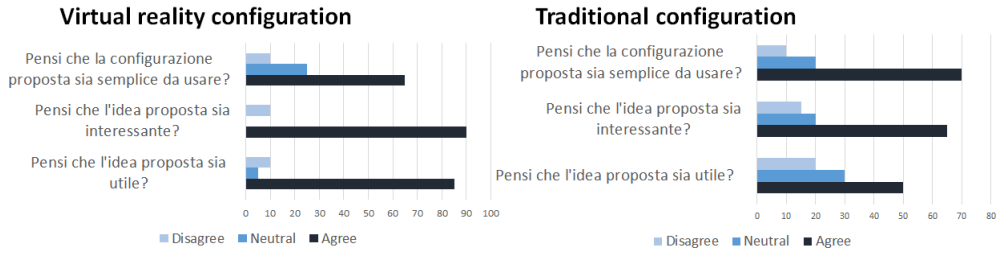


Figura 5.18: Altre considerazioni sull'utilità, la semplicità e l'interesse per il sistema proposto

5.5 Tutte e sei le configurazioni

5.5.1 Dati generali del campione

Per capire il livello di gradimento di tutte le configurazioni, oltre ai 20 studenti considerati nel confronto precedente, sono stati coinvolti altri 40 utenti.

Il campione di 60 utenti è composto prevalentemente da studenti magistrali di facoltà scientifiche. Il 21% degli utenti è di sesso femminile. (Figura 5.19).

Sesso		
	Numero	%
Maschi	47	78,33
Femmine	16	21,67
SOMMA	60	100

(a) Percentuale di maschi e femmine

Laurea		
	Numero	%
Triennale	27	38,33
Magistrale	37	61,67
SOMMA	60	100

Corso di studi		
	Numero	%
Informatica	47	78,33
Ingegneria	3	5,00
Farmacia	4	6,67
Matematica	1	1,67
Lettere	2	3,33
Chimica	1	1,67
Medicina	1	1,67
Economia	1	1,67
SOMMA	60	100

Carattere del corso di studi		
	Numero	%
Scientifico	57	95
Umanistico	3	5
SOMMA	60	100

(b) Corso di studi

Figura 5.19: Alcuni dati sul campione di 60 studenti

5.5.2 Risultati del TAM

Le risposte alle domande relative al *TAM* di tutti i 60 utenti sono state analizzate e messe a confronto. In Figura 5.20 sono riportati i risultati di tutte e sei le configurazioni disponibili. In generale, tutte le configurazioni hanno ottenuto risultati positivi (maggiori di 4). In particolare, la configurazione che ha ottenuto risultati migliori è quella composta da *Oculus Rift* e *Joypad*. Ciò è giustificato dal fatto che usando il *Joypad* si ha una libertà di movimento superiore a quella ottenibile con *Mouse e Tastiera* e non si incorre in problemi correlati a campi d'azione come avviene invece con il *Leap Motion*. Un altro dato interessante riguarda le due configurazioni che utilizzano la *Leap Motion*. Infatti, esse hanno risultati più o meno simili. Ciò dimostra che l'utenza ha problemi nell'interagire con la *Leap Motion* e non con l'*Oculus Rift*.

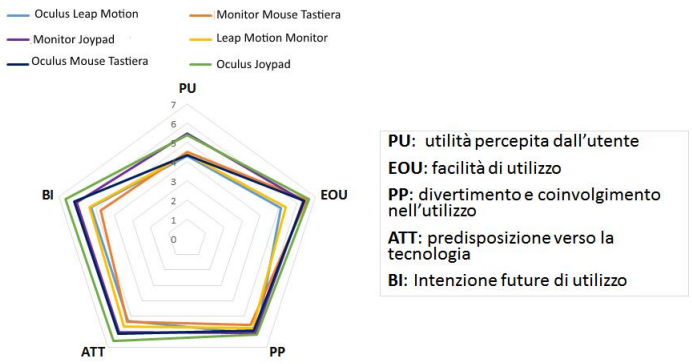


Figura 5.20: Risultati del TAM per tutte e sei le configurazioni

Capitolo 6

Conclusioni e sviluppi futuri

In questo lavoro di tesi è stato presentato *Graph 3D Explorer* un plugin per *Gephi* che permette la visualizzazione e l'esplorazione di una rete in un ambiente virtuale tridimensionale e interattivo. *Graph 3D Explorer* è stato sviluppato e successivamente validato con un campione di sessanta studenti dell'Università di Salerno che ne hanno confermato le qualità e l'innovatività. In particolare, lo studio di usabilità condotto ha mostrato un forte interesse da parte degli utenti verso le interfacce naturali utente e le metodologie di interazioni innovative. Tuttavia queste vengono percepite ancora come più difficili da usare rispetto alle configurazioni classiche che fanno uso di mouse e tastiera. Questa maggiore difficoltà nell'utilizzo delle interfacce naturali utente è dovuto soprattutto alla poca esperienza rispetto a periferiche come mouse e tastiera che sono da anni la metodologia principale con cui ci si interfaccia con i computer. Infatti, quasi nessun utente del campione aveva utilizzato la periferica Leap Motion oppure l'Oculus Rift prima dell'esperienza con l'applicazione realizzata. Le analisi statistiche condotte sulle risposte degli utenti hanno mostrato che la facilità di utilizzo è la chiave affinché le modalità di interazione alternative proposte possano essere utilizzate. In virtù di queste considerazioni bisogna progettare degli insiemi di gesti sempre più semplici e che soprattutto puntino a rendere immediato qualcosa che con mouse e tastiera risulta complesso. Il sistema sviluppato e i risultati dello studio di usabilità condotto hanno mostrato che le interfacce naturali utente e quelle tradizionali hanno le stesse potenzialità. Ora bisogna fare un passo in avanti e cercare di mettere in risalto gli aspetti che rendono veramente motivato il passaggio da delle interazioni tradizionali che hanno anni di utilizzo alle spalle a delle metodologie di interazioni innovative che suscitano tanta curiosità ma che al momento non hanno ancora sbancato.

Bibliografia

[1] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998. 1.1.1

[2] B. A.-L. BARABÁSI and E. Bonabeau, "Scale-free," *Scientific American*, 2003. 1.1.1

[3] D. Easley and J. Kleinberg, *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press, 2010. 1.1.1

[4] G. Bavota, R. Oliveto, M. Gethers, D. Poshyvanyk, and A. De Lucia, "Methodbook: Recommending move method refactorings via relational topic models," 2014. 1.1.1

[5] O. Sporns, "Graph theory methods for the analysis of neural connectivity patterns," in *Neuroscience Databases*, pp. 171–185, Springer, 2003. 1.1.1

[6] J. Zhang, *Visualization for information retrieval*, vol. 24. Springer, 2008. 1.1.2, 1.1.3

[7] M. J. Eppler and J. Mengis, "The concept of information overload: A review of literature from organization science, accounting, marketing, mis, and related disciplines," *The information society*, vol. 20, no. 5, pp. 325–344, 2004. 1.1.2

[8] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis, "Algorithms for drawing graphs: an annotated bibliography," *Computational Geometry*, vol. 4, no. 5, pp. 235–282, 1994. 1.1.2

[9] I. Herman, G. Melançon, and M. S. Marshall, "Graph visualization and navigation in information visualization: A survey," *Visualization and*

Computer Graphics, IEEE Transactions on, vol. 6, no. 1, pp. 24–43, 2000. 1.2

[10] S. Deterding, M. Sicart, L. Nacke, K. O'Hara, and D. Dixon, "Gamification. using game-design elements in non-gaming contexts," in *CHI'11 Extended Abstracts on Human Factors in Computing Systems*, pp. 2425–2428, ACM, 2011. 1.2, 2.5

[11] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," 2009. 1.3, 3.1

[12] A. Van Dam, "Post-wimp user interfaces," *Communications of the ACM*, vol. 40, no. 2, pp. 63–67, 1997. 2.1, 2.2

[13] R. P. McMahan, E. D. Ragan, A. Leal, R. J. Beaton, and D. A. Bowman, "Considerations for the use of commercial video games in controlled experiments," *Entertainment Computing*, vol. 2, no. 1, pp. 3–9, 2011. 2.2

[14] Y. Boussemart, F. Rioux, F. Rudzicz, M. Wozniewski, and J. R. Cooperstock, "A framework for 3d visualisation and manipulation in an immersive space using an untethered bimanual gestural interface," in *Proceedings of the ACM symposium on Virtual reality software and technology*, pp. 162–165, ACM, 2004. 2.3, 2.2

[15] D. C. Smith, F. E. Ludolph, C. H. Irby, and J. A. Chairman-Johnson, "The desktop metaphor as an approach to user interface design (panel discussion)," in *Proceedings of the 1985 ACM annual conference on The range of computing: mid-80's perspective: mid-80's perspective*, pp. 548–549, ACM, 1985. 2.3

[16] M. F. Shiratuddin and K. W. Wong, "Game design considerations when using non-touch based natural user interface," in *Transactions on Edutainment VIII*, pp. 35–45, Springer, 2012. 2.4, 2.6.1

[17] S. Chen and D. Michael, "Serious games: Games that educate, train and inform," *USA, Thomson Course Technology*, vol. 21, 2005. 2.5

[18] C.-Y. Chang, B. Lange, M. Zhang, S. Koenig, P. Requejo, N. Somboon, A. A. Sawchuk, and A. A. Rizzo, "Towards pervasive physical rehabilitation using microsoft kinect," in *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2012 6th International Conference on*, pp. 159–162, IEEE, 2012. 2.5

[19] D. Bassily, C. Georgoulas, J. Guettler, T. Linner, and T. Bock, "Intuitive and adaptive robotic arm manipulation using the leap motion controller," in *ISR/Robotik 2014; 41st International Symposium on Robotics; Proceedings of*, pp. 1–7, VDE, 2014. 2.5

[20] A. C. de Carvalho Correia, L. C. de Miranda, and H. Hornung, "Gesture-based interaction in domotic environments: State of the art and hci framework inspired by the diversity," in *Human-Computer Interaction–INTERACT 2013*, pp. 300–317, Springer, 2013. 2.5

[21] L. Gallo, A. P. Placitelli, and M. Ciampi, "Controller-free exploration of medical image data: Experiencing the kinect," in *Computer-Based Medical Systems (CBMS), 2011 24th International Symposium on*, pp. 1–6, IEEE, 2011. 2.5

[22] G. C. S. Ruppert, L. O. Reis, P. H. J. Amorim, T. F. de Moraes, and J. V. L. da Silva, "Touchless gesture user interface for interactive image visualization in urological surgery," *World journal of urology*, vol. 30, no. 5, pp. 687–691, 2012. 2.5

[23] M. Nancel, J. Wagner, E. Pietriga, O. Chapuis, and W. Mackay, "Mid-air pan-and-zoom on wall-sized displays," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 177–186, ACM, 2011. 2.6.1

[24] D. Freeman, R. Vennelakanti, and S. Madhvanath, "Freehand pose-based gestural interaction: Studies and implications for interface design," in *Intelligent Human Computer Interaction (IHCI), 2012 4th International Conference on*, pp. 1–6, IEEE, 2012. 2.6.1, 2.7

[25] J.-F. Jégo, A. Paljic, and P. Fuchs, "User-defined gestural interaction: A study on gesture memorization," in *3D User Interfaces (3DUI), 2013 IEEE Symposium on*, pp. 7–10, IEEE, 2013. 2.6.2, 2.8

[26] S. Jarukasemratana and T. Murata, "Recent large graph visualization tools: A review," *Information and Media Technologies*, vol. 8, no. 4, pp. 944–960, 2013. 3.1

[27] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker, "Cytoscape: a software environment for integrated models of biomolecular interaction networks," *Genome research*, vol. 13, no. 11, pp. 2498–2504, 2003. 3.1

[28] R. Saito, M. E. Smoot, K. Ono, J. Ruscheinski, P.-L. Wang, S. Lotia, A. R. Pico, G. D. Bader, and T. Ideker, "A travel guide to cytoscape plugins," *Nature methods*, vol. 9, no. 11, pp. 1069–1076, 2012. 3.1

[29] D. Auber, D. Archambault, R. Bourqui, A. Lambert, M. Mathiaut, P. Mary, M. Delest, J. Dubois, G. Mélançon, *et al.*, "The tulip 3 framework: A scalable software library for information visualization applications based on relational data," 2012. 3.1

[30] T. Moscovich, F. Chevalier, N. Henry, E. Pietriga, and J.-D. Fekete, "Topology-aware navigation in large networks," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2319–2328, ACM, 2009. 3.1

[31] "NetBeans IDE - NetBeans Rich-Client Platform development (RCP)." https://netbeans.org/features/platform/. Last visited on 2014-11-28. 3.1

[32] "Google Summer of Code - Open Source Programs Office." https://developers.google.com/open-source/soc/. Last visited on 2014-11-28. 3.1

[33] C. North and B. Shneiderman, "Snap-together visualization: a user interface for coordinating visualizations via relational schemata," in *Proceedings of the working conference on Advanced visual interfaces*, pp. 128–135, ACM, 2000. 3.4

[34] K. Orland, D. Thomas, and S. M. Steinberg, *The videogame style guide and reference manual*. Lulu. com, 2007. 3.5.1

[35] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994. 4.2, 4.4, 4.9

[36] T. Akenine-Moller, E. Haines, and N. Hoffman, *Real-time rendering*. AK, 2002. 4.5, 4.9.1

[37] "Leap Motion SDK documentation." https://developer.leapmotion.com/documentation/. Last visited on 2014-11-28. 4.8.1

[38] "jherico/jovrÂ· GitHub." https://github.com/jherico/jovr. Last visited on 2014-12-07. 4.10

[39] R. Likert, "A technique for the measurement of attitudes.," *Archives of psychology*, 1932. 5.3

[40] F. D. Davis, "Perceived usefulness, perceived ease of use, and user acceptance of information technology," *MIS quarterly*, pp. 319–340, 1989. 5.3

[41] J.-W. Moon and Y.-G. Kim, "Extending the tam for a world-wide-web context," *Information & Management*, vol. 38, no. 4, pp. 217–230, 2001. 5.3

[42] I.-F. Liu, M. C. Chen, Y. S. Sun, D. Wible, and C.-H. Kuo, "Extending the tam model to explore the factors that affect intention to use an online learning community," *Computers & Education*, vol. 54, no. 2, pp. 600–610, 2010. 5.3