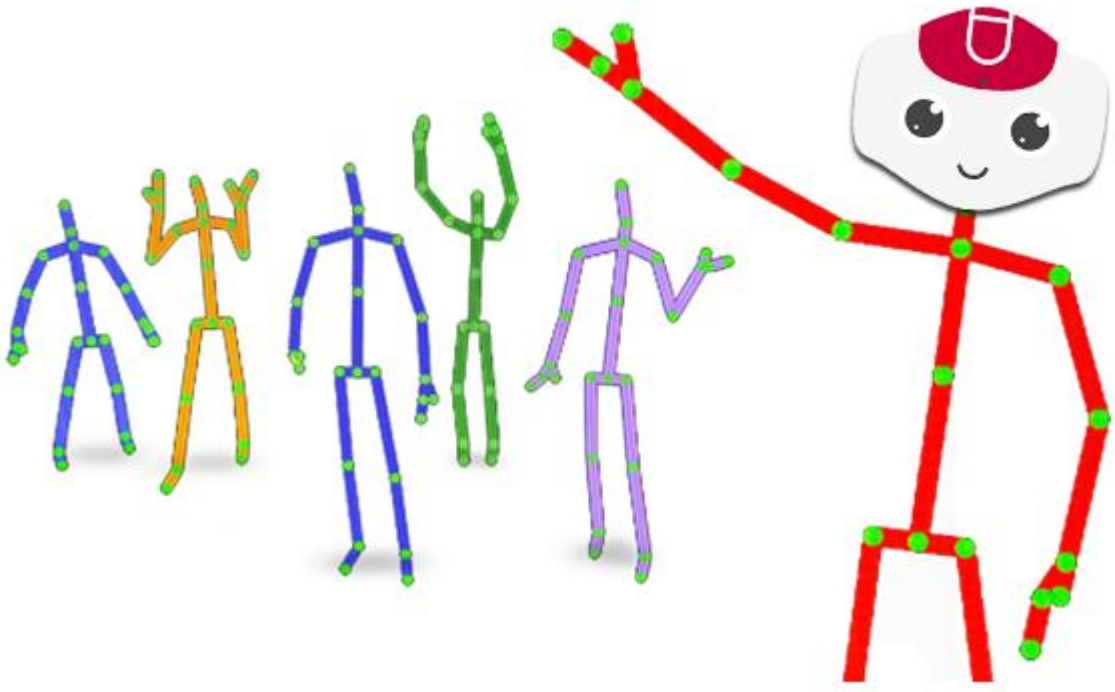
Lorenzo Daidone

I.I.S. “Italo Calvino” - Genova

A.S. 2016/2017

NAO e Kinect



lorenzo.daidone@icloud.com

Abstract [ITA]	Abstract [ENG]
In situazioni di pericolo o di impossibilità, risulta essenziale comandare in remoto dei robot. Il software si propone come un framework1 per lo sviluppo di applicazioni interattive	In situations of danger or impossibility, it is essential to remotely control robots. The software is intended as a framework for the development of interactive applications

Keywords:

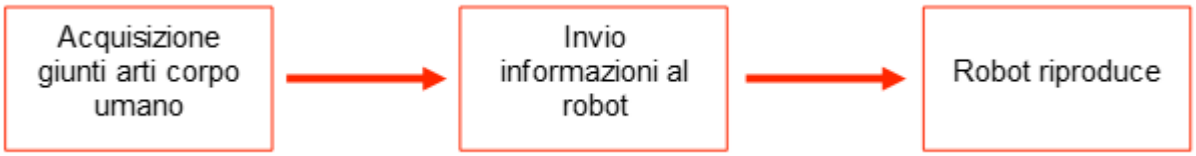
- *Aldebaran NAO robot*
- *Microsoft 360® Kinect*

Introduzione

Italiano	English
Nella maggior parte dei casi, quando si approccia un software per il controllo di un robot, l'attenzione che gli sviluppatori pongono nella progettazione dell'interfaccia grafica (***GUI***), risulta essere scarsa. L'innovazione introdotta con la ***NUI*** (*Interfaccia Utente Naturale*) rende ora possibili una vasta gamma di applicazioni che consentono all'utente di interagire con il dispositivo utilizzando i movimenti e un linguaggio naturale. Una possibile applicazione del ***framework***, potrebbe essere quella di pilotaggio remoto di robot, anche antropomorfi. Inoltre, la piattaforma potrebbe essere utilizzata per applicazioni mediche, ad esempio nel campo delle disabilità gravi, per tenere traccia dei progressi e sviluppando applicazioni che sottopongano all'utente giochi al fine di affinare motricità grossa e fine. In questa tesi il ***framework*** è stato implementato nel seguente modo: l'umano si muove davanti al dispositivo di acquisizione. Mediante una comunicazione senza fili, il robot riproduce in tempo reale ciò che l'umano esegue	In most cases, when approaching a robot controlling software, the attention paid by developers in designing the ***GUI*** turns out to be poor. The innovation introduced with the ***NUI***, now makes possible a wide range of applications that allow the user to interact employing movements and a natural language. A potential application, could be remotely controlled robot, both for anthropomorphic robots and not. Furthermore, the platform could be used for medical treatment, as an example in the field of severe disabilities, in order to keep track of progress, developing applications that present to user games in order to improve fine and large motricity. In this thesis, the ***framework*** has been implemented in the following way: the human moves in font of capture device. Through a wireless communications, robot reproduces in real time what the human performs

[1] Attenzione: le parole mostrate in color rosso sono definite nell'appendice "Glossario", in fondo a questa tesina

Considerata la complessità del problema, ho suddiviso il problema in tre macro-problemi:



1. Acquisizione giunti arti corpo umano

Per acquisire la posizione nello spazio dei "giunti" del corpo umano si è deciso di utilizzare il dispositivo **Kinect**, distribuito dalla Microsoft a partire dal 2010.

Il dispositivo possiede un proiettore e un ricevitore ad infrarossi, telecamera e un array di microfoni che permettono di individuare l'angolo da cui l'utente sta parlando. È possibile variare l'angolo di visione verticale, agendo sul motore di *tilt* L'escursione è di ± 27°.

Costo: ~150 $.

Specifiche di Kinect V1 (2010)

Portata	MAX: ~ 4 m, MIN: ~ 80 cm
Angolo di visione	43,5°
Camera	VGA resolution 640 · 480 pixel. Profondità colore: 8 bit
Velocità di acquisizione frame (stream profondità e colore)	30 FPS (frames per second)
Formato audio	16-kHz, 24-bit mono pulse code modulation (PCM)
Caratteristiche audio input	Array di quattro microfoni con convertitore analogico-digitale a 24 bit ed elaborazione interna del suono con eliminazione dell'eco e riduzione del rumore
Caratteristiche accelerometro	Un accelerometro 2G/4G/8G, configurato per il range 2G con l'accuratezza di 1° sopra il limite

Microsoft distribuisce gratuitamente il kit di sviluppo software (SDK: *software development kit*).

L'SDK comprende:

- driver per utilizzare il dispositivo
- Application Programming Interfaces (APIs) e interfacce per il dispositivo
- Esempi, strumenti e altre risorse per lo sviluppo

Gli esempi contenuti nel pacchetto sono scritti nei linguaggi C#, C++ e VB.

Per ridurre i tempi e i costi di sviluppo è stato modificato un esempio già presente nell'SDK. L'esempio utilizzato per realizzare il tracking dei giunti del corpo umano è "Skeleton Basics". Il software è stato scritto in linguaggio C# e permette di tracciare un massimo di due persone.

Per ogni persona sono rappresentati 20 giunti.

L'esempio, così come viene distribuito, non permette l'invio delle coordinate dei giunti via rete.

Per modificare il progetto è stato utilizzato **Visual Studio**, l'ambiente di sviluppo integrato (***IDE***) sviluppato da Microsoft che permette la realizzazione di applicazioni, siti web, applicazioni web e servizi web.

Requisiti Visual Studio hardware e software per lo sviluppo:

Requisiti	HW e SW utilizzati
Hardware: • Processore 32 o 64 bit, dual core, 2.66GHz o più veloci • Bus USB 2.0 dedicato a Kinect • 2 GB di RAM (min.) • Scheda video che supporti DirectX 9.0c **Software**: • Sistema operativo: Windows 7 o più recenti • Visual Studio 2010 o 2012 • Microsoft .NET Framework	**Hardware**: • Kinect utilizzata: modello 1414 • 4 GB di RAM • Scheda video 2GB **Software**: • Sistema operativo utilizzato: Windows 10 • Microsoft Visual Studio Express 2012 per Windows Desktop, versione 11.0.61219.00 update 5 • Microsoft .NET Framework, versione 4.6.01586

Le modifiche apportate al progetto SkeletonBasics originario, sono le seguenti:

- Aggiunto metodo **JointToString**, il quale crea una stringa composta da nome del giunto e le rispettive coordinate del giunto nello spazio.

- Aggiunta classe **Server.cs**, che prevede alcune funzionalità di rete quali: creazione socket lato server, invio dati, ricezione dati, chiusura socket.

2. Invio informazioni al robot NAO

Al fine di far comunicare il client (NAO) con il server (Kinect) è stato creato un **socket** di tipo stream, protocollo tcp in cui gli indirizzi IP sono della versione 4.

I socket di tipo *stream* forniscono comunicazioni bidirezionali affidabili. Il protocollo di comunicazione standard per questo tipo di socket è il **TCP** (*Transmission Control Protocol*).

Il protocollo TCP è progettato in modo che i pacchetti di dati arrivino a destinazione senza errori e in sequenza.

Attualmente, la maggior parte dei sistemi su internet ha un indirizzo IP (internet protocol), costituito dal noto schema a quattro byte nella forma w.x.y.z.

Esempio:

192.168.0.1

Gli indirizzi IPv4, che hanno uno spazio di indirizzamento a 32 bit, sono utilizzati dal 1981:

$2^{32} \cong 4{,}3 \cdot 10^{9}$ indirizzi

La versione 6 dell'*internet protocol* riserva 128 bit per l'indirizzamento.

$$2^{128} \cong 3{,}4 \cdot 10^{38} \text{ indirizzi}$$

Il formato con cui il server invia il nome dei giunti e le rispettive coordinate, per un'opportuna elaborazione, è stato creato ad hoc. Non si utilizza un protocollo standardizzato.

Il messaggio ASCII inviato a NAO è una concatenazione di nome giunto e rispettive coordinate nello spazio. Il sistema di riferimento per tali coordinate è il seguente: sistema di riferimento destrorso, fronte rivolta al dispositivo, asse x verso destra dell'utente e asse z direzionato verso l'utente [fig. 1]:

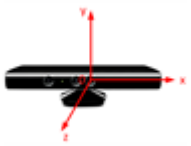


Figura 1

In generale, la regola per il protocollo è la seguente:

```
NomeGiunto Xn Yn Zn/
```

dove `NomeGiunto` è uno di quelli elencati nella fig. 2:

Figura 2

Esempio di messaggio inviato dal server al client:

```
HipCenter 0,02213011 -0,1516186 3,294796/Spine 0,0270134 -0,0878974
3,345237/ShoulderCenter 0,02703401 0,2609315 3,324159/Head 0,01493661
0,4477339 3,266301/ShoulderLeft -0,1625187 0,1579514 3,344531/ElbowLeft -
0,3824059 0,02974367 3,294011/WristLeft -0,5916628 -0,04512607
3,192345/HandLeft -0,6193663 -0,04728847 3,160089/ShoulderRight 0,2109764
0,1430885 3,312534/ElbowRight 0,371758 -0,01951969 3,216079/WristRight
0,5452417 -0,1115142 3,066102/HandRight 0,6481183 -0,1721237
2,975143/HipLeft -0,06597368 -0,236739 3,2796/KneeLeft -0,06226847 -
0,7335333 3,314915/AnkleLeft -0,02930086 -1,104403 3,362409/FootLeft -
0,05503895 -1,183514 3,299364/HipRight 0,1072637 -0,2343558
3,289687/KneeRight 0,1137078 -0,6194826 3,058723/AnkleRight 0,110105 -
1,020778 2,865749/FootRight 0,1243722 -1,058315 2,742855/
```

Una volta che il software “Skeleton Basics” è stato dotato di connettività su rete locale e della capacità di creare un messaggio che il client possa correttamente interpretare, la parte server si può considerare terminata.

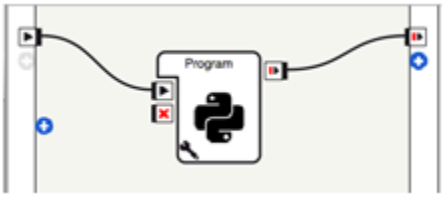
3. Riproduzione dei movimenti da parte del robot NAO

NAO è un robot umanoide progettato da SoftBank Robotics nel 2006.

Caratteristiche principali	
Altezza	58 cm 
Connettivitùà	Wi-fi, ethernet, usb
Sistema operativo	OpenNAO, distribuzione GNU/Linux di Gentoo
Gradi di libertà	25
Videocamere	1280 * 960 @30fps(x2)
Altoparlanti	2
Microfoni	4
Massa	5,4 kg
Sensori	Ultrasuoni, infrarossi, di pressione, di tatto
Costo	**~ 7300 €**

I software lato client è stato realizzato con **Choregraphe**, l’ambiente di sviluppo distribuito gratuitamente da Aldebaran, che fornisce una serie di strumenti per sviluppare agevolmente applicazioni. Nell’*IDE* è compreso anche un simulatore.

Per avere una maggiore libertà di sviluppo tutto il programma è stato scritto in **Python**. Un esempio di programma sviluppato in Choregraphe è il seguente:



Per decodificare il messaggio inviato dal server e rendere i dati fruibili da NAO sono necessarie le seguenti operazioni:

1. il messaggio è frammentato in una lista di stringhe, considerando il carattere slash **/** come un carattere di interruzione
2. le stringhe ottenute vengono nuovamente frammentate usando lo spazio come carattere di interruzione. (Si ottiene un *dizionario*)
3. nei numeri che rappresentano le coordinate le virgole sono sostituite con dei punti (notazione anglosassone)
4. Effettua un cast di tipo convertendo le coordinate, rappresentate come stringhe, in numeri di tipo `float` (reali)

Una volta trasformato il messaggio in una matrice bidimensionale che contiene i nomi dei giunti e le rispettive coordinate, occorre passare da coordinate ad angoli.

L'intento è di rendere il ***framework*** il più generale possibile, pertanto è stata creata una classe di nome "Angles". I metodi della classe permettono di effettuare calcoli su vettori e da questi leggere i valori dei relativi angoli.

Come ricavare l'angolo tra due vettori

Kinect ritorna le coordinate dei giunti di un corpo umano. Invece, i parametri delle funzioni Python che gestiscono il movimento dei giunti del robot accettano solamente angoli.

Per passare da coordinate in tre dimensioni ad angoli è stata utilizzata la formula inversa del prodotto scalare.

Prodotto scalare:

$$v \cdot w = |v| \cdot |w| \cdot \cos(\varphi)$$

La formula inversa è:

$$\cos(\varphi) = \frac{v \cdot w}{|v| \cdot |w|}$$

Da cui:

$$\varphi = arcos(\cos(\varphi))$$

Se i vettori sono espressi tramite omponenti:

$$v = (v_x, v_y, v_z) \qquad \text{e} \qquad w = (w_x, w_y, w_z)$$

il prodotto scalare diventa:

$$v \cdot w = (v_x, v_y, v_z) \cdot (w_x, w_y, w_z)^T = v_x w_x + v_y w_y + v_z w_z$$

Dalle relazioni precedenti si ottiene [1]:

$$\varphi = arcos\left(\frac{v_x w_x + v_y w_y + v_z w_z}{\sqrt{v_x^2 + v_y^2 + v_z^2} \cdot \sqrt{w_x^2 + w_y^2 + w_z^2}}\right)$$

Coseni direttori

I coseni direttori di un vettore, sono i coseni degli angoli convessi che la retta (o la retta su cui giace il vettore) forma con gli assi cartesiani. Ad esempio per il coseno direttore del vettore *v* è:

$$\cos(\varphi) = \frac{v}{|v|}$$

Pertanto la relazione [1], precedentemente trattata, si può anche esprimere in funzione dei coseni direttori dei vettori coinvolti:

$$\varphi = arcos\left(\frac{v_x w_x + v_y w_y + v_z w_z}{\sqrt{v_x^2 + v_y^2 + v_z^2} \cdot \sqrt{w_x^2 + w_y^2 + w_z^2}}\right) = arcos\left(\frac{v_x w_x + v_y w_y + v_z w_z}{|v| \cdot |w|}\right) =$$

$$= arcos\left(\frac{v_x}{|v|} \cdot \frac{w_x}{|w|} + \frac{v_y}{|v|} \cdot \frac{w_y}{|w|} + \frac{v_z}{|v|} \cdot \frac{w_z}{|w|}\right)$$

Con la relazione sopra riportata è possibile associare un angolo ad ogni giunto del robot, prendendo in considerazione due vettori.

La tabella seguente mette in relazione un angolo di NAO con i rispettivi vettori ottenuti da Kinect:

Arto destro				
NAO joint	**Abbreviazione**	***v***	***w***	**Vertice comune?**
Shoulder pitch	rsp	anca - spalla	spalla - gomito	Sì
Shoulder roll	rsr	anca sx - spalla dx	spalla - gomito	No
Elbow yaw	rey	anca - spalla	gomito - polso	No
Elbow roll	rer	spalla - gomito	gomito - polso	Sì

Non sempre i vettori presi in considerazione hanno punti in comune.

L'angolo ottenuto per ciascun giunto non è direttamente assegnabile al rispettivo motore mediante apposite istruzioni. È necessario introdurre un offset nell'angolo:

NAO joint	**Offset [rad]**	**NAO joint**	**Offset [rad]**
Left Elbow Roll	-π	Right Elbow Roll	0
Left Elbow Yaw	-3π/2	Right Elbow Yaw	-π/2
Left Shoulder Roll	-π/2	Right Shoulder Roll	-3π/2
Left Shoulder Pitch	-π/2	Right Shoulder Pitch	-π/2

Tali offset sono stati determinati empiricamente, utilizzando un widget compreso nell'***IDE*** Choregraphe: *Motion* (figure 3b e 4b).

Come si può notare nella fig. 3a l'angolo fra i due vettori è di circa 180° (π rad) mentre l'angolo del motion widget è di ~ 0 gradi [fig. 3b].

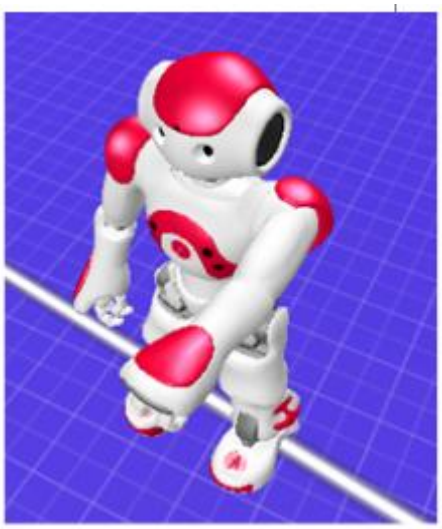


Figura 3a

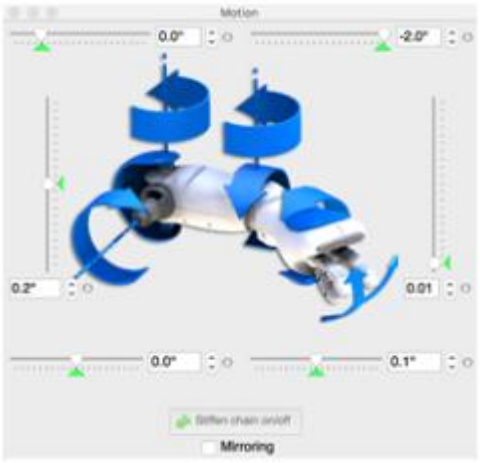


Figura 3b

Analogamente, nella fig. 4a l'angolo fra i due vettori è di circa 90° (π/2 rad) mentre nel motion widget è indicato ~ -90° [fig. 4b].

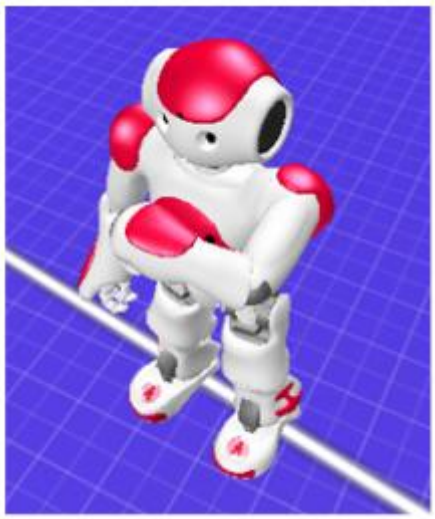


Figura 4a

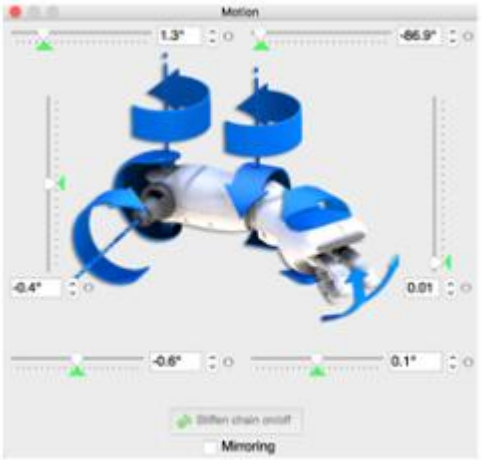


Figura 4b

In alcuni casi Kinect potrebbe ritornare una lettura sfalsata dei giunti del corpo umano con la conseguenza di ottenere calcoli errati. Per evitare che i giunti del robot compiano angoli eccessivi, provocando la rottura degli stessi, viene prima effettuato un controllo. Se l'angolo è compreso in un determinato range, allora il suo valore viene inviato all'attuatore del giunto, altrimenti la posizione del giunto rimane invariata. In gergo quest'operazione viene chiamata ***clipping***.

Considerazioni

Uno dei miei obiettivi è di rendere il programma semplice da utilizzare e molto “robusto”.
L’utente deve solo avviare il programma server, il programma client ed essere pronto ad operare.
Ecco perché sono state aggiunte alcune funzionalità, sebbene non essenziali al funzionamento del programma che lo rendono maggiormente ***user friendly***.

Disattivazione modalità vita autonoma

Quando si accende l’umanoide, di default è attivata la “autonomous life”. Questa funzionalità rende il robot più “vivo” facendolo ondeggiare sui fianchi e seguire perfino il volto dell’utente. Ai fini del programma, il controllo dell’hardware deve essere totalmente eseguito dal software dell’utente e non da quello della modalità autonoma; pertanto L'*autonomous life viene* disattivata non appena avviato il software client.

Set posizioni iniziali

Il robot possiede di default un modulo denominato "*ALRobotPosture*" che consente di portare il robot in una posizione predefinita.
Fra le posizioni disponibili, come posizione iniziale è stata scelta la **StandInit** [fig. 5] perché garantisce un buon equilibrio al robot ed è disponibile su tutte le versioni di NAO.

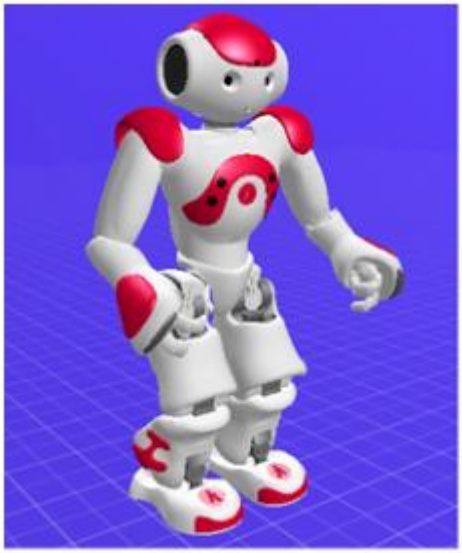


Figura 5

Glossario

Il glossario presenta una veloce definizione e una dettagliata, sia in lingua inglese sia in italiano.

Clipping: *spuntata, ritaglio*

Clipping, in the context of computer graphics, is a method to selectively enable or disable rendering operations within a defined region of interest.

Clipping, nel contesto del grafica computer è un metodo per abilitare e disabilitare selettivamente le operazioni di rendering in una regione di interesse definita.

Framework: *struttura, intelaiatura*. Architettura logica di supporto su cui un software può essere progettato e realizzato.

In computer programming, a software framework is an abstraction in which software providing generic functionality can be selectively changed by additional user-written code, thus providing application-specific software.

In informatica, un framework software è un'astrazione in cui il software fornisce funzionalità generiche che possono essere modificate selettivamente con del codice scritto dall'utente, fornendo così un software specifico per l'applicazione.

GUI: *interfaccia utente grafica*

The Graphical User Interface (GUI), is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators, instead of text-based user interfaces. GUIs were introduced in reaction to the perceived steep learning curve of command-line interfaces (CLIs), which require commands to be typed on a computer keyboard.

L'interfaccia utente grafica, è un tipo di interfaccia utente che permette agli utenti di interagire con dispositivi elettronici attraverso icone e indicatori visuali, al posto di interfacce basate sul testo. Le GUI sono state introdotte per migliorare la curva di apprendimento delle interfacce a linea di comando (CLI), che richiedono comandi da digitare su una tastiera.

IDE: *ambiente di sviluppo integrato*

An Integrated Development Environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools and a debugger. Most modern IDEs have intelligent code completion.

Un ambiente di sviluppo integrato (IDE) è un'applicazione software che fornisce ai programmatori servizi completi per lo sviluppo software. Normalmente un IDE consiste in un editor di testo, strumenti per compilare il codice e un debugger. I più moderni IDE possiedono anche la possibilità di completamento automatico del codice.

NUI: *intefaccia utente naturale*

In computing, a Natural User Interface, or NUI, or natural interface is a user interface that is effectively invisible, and remains invisible as the user continuously learns increasingly complex interactions. The word natural is used because most computer interfaces use artificial control devices whose operation has to be learned.

In informatica, un’interfaccia utente naturale, NUI, o interfaccia naturale è un’interfaccia completamente invisibile perché impara direttamente dall'utente interazioni sempre più complesse. Viene usata la parola *naturale* perché si differenzia dalla maggior parte delle interfacce per computer che l'utente deve invece imparare a usare.

User friendly: *di facile utilizzo anche per chi non è esperto*

In software engineering, usability is the degree to which a software can be used by specified consumers to achieve quantified objectives with effectiveness, efficiency, and satisfaction in a quantified context of use.

In ingegneria del software, l'*usabilità* è il grado in cui un software può essere utilizzato da consumatori specifici per raggiungere obiettivi quantificati con efficacia, efficienza e soddisfazione, in un preciso contesto di utilizzo.

Riferimenti

- Specifiche Kinect: https://msdn.microsoft.com/en-us/library/jj131033.aspx
- Kinect joint type enumeration: https://msdn.microsoft.com/en-us/library/microsoft.kinect.jointtype.aspx
- Indirizzo IP: pag 285, Arte dell’Hacking, Volume 1. Jon Erickson. Edito da Apogeo
- Specifiche NAO robot: http://doc.aldebaran.com/2-1/family/robots/index_robots.html
- Coseni direttori: http://it.wikipedia.org/wiki/Coseni_direttori
- NAO joints: http://doc.aldebaran.com/2-1/family/nao_dcm/actuator_sensor_names.html
- NAO joint control: http://doc.aldebaran.com/1-14/naoqi/motion/control-joint.html
- StandInit: http://doc.aldebaran.com/1-14/naoqi/motion/alrobotposture.html