ANDREA LEGANZA

KINECT

CONTROLLO REMOTO DEL MOUSE

Andrea Leganza

- laureato in ing. informatica
- redattore rivista Io Programmo (ed. Master)
- docente di sviluppo applicazioni mobile presso l'istituto Quasar di Roma;
- certicato Eucip Core, Java 1.6, Adobe Air and Flex 3
- sviluppatore applicazioni desktop, web, mobile.

Kinect

- Commercializzazione: 2010
- Sistema di comunicazione: USB
- Generatore di griglia di infrarossi
- Camera VGA 640*480@30Hz
- Sensore infrarossi: VGA 640*480@30Hz
- Accelerometro
- Hardware: PrimeSense + Microsoft
- Software: Microsoft

Kinect: funzionamento

- Un generatore di infrarossi proiettta una “maglia” invisibile all’interno di un’area di circa 6mq.

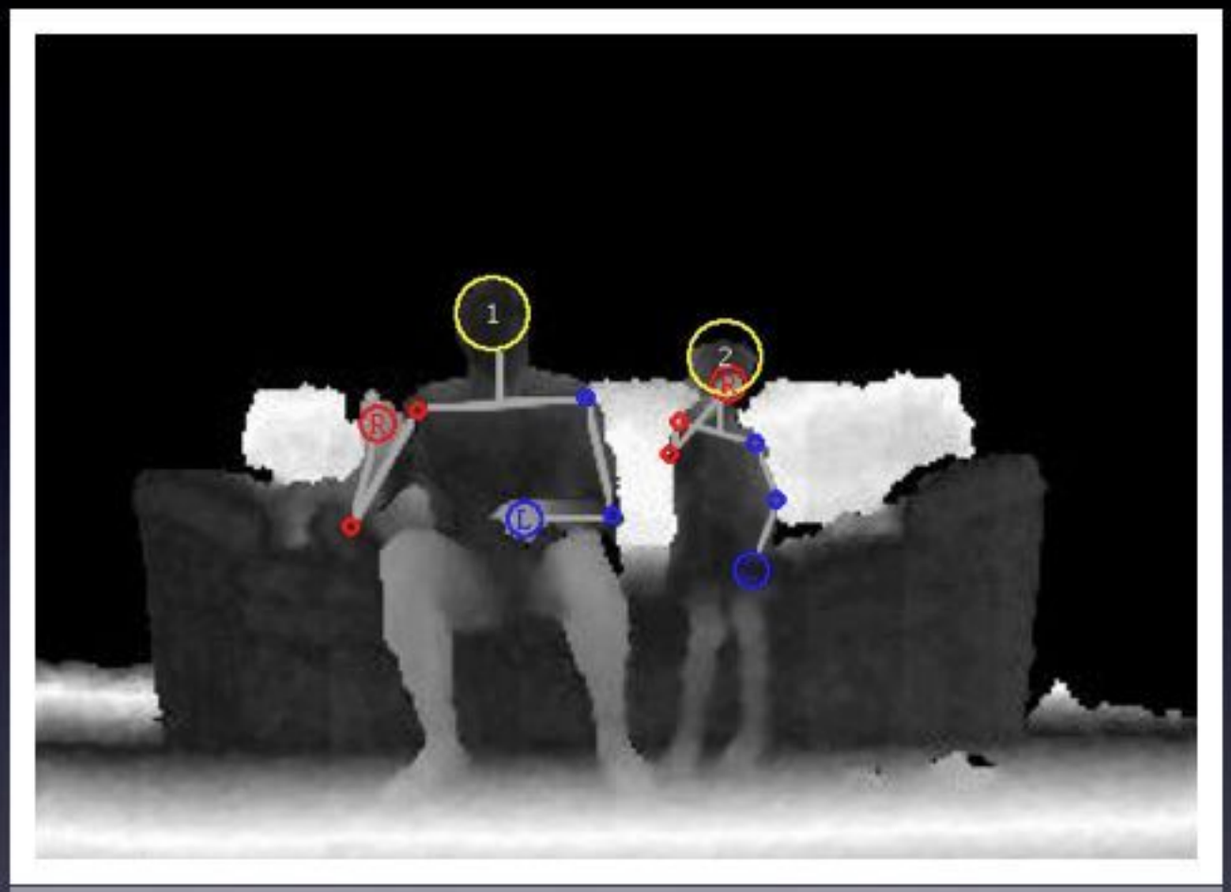
Kinect: funzionamento

VIDEO

Kinect: funzionamento

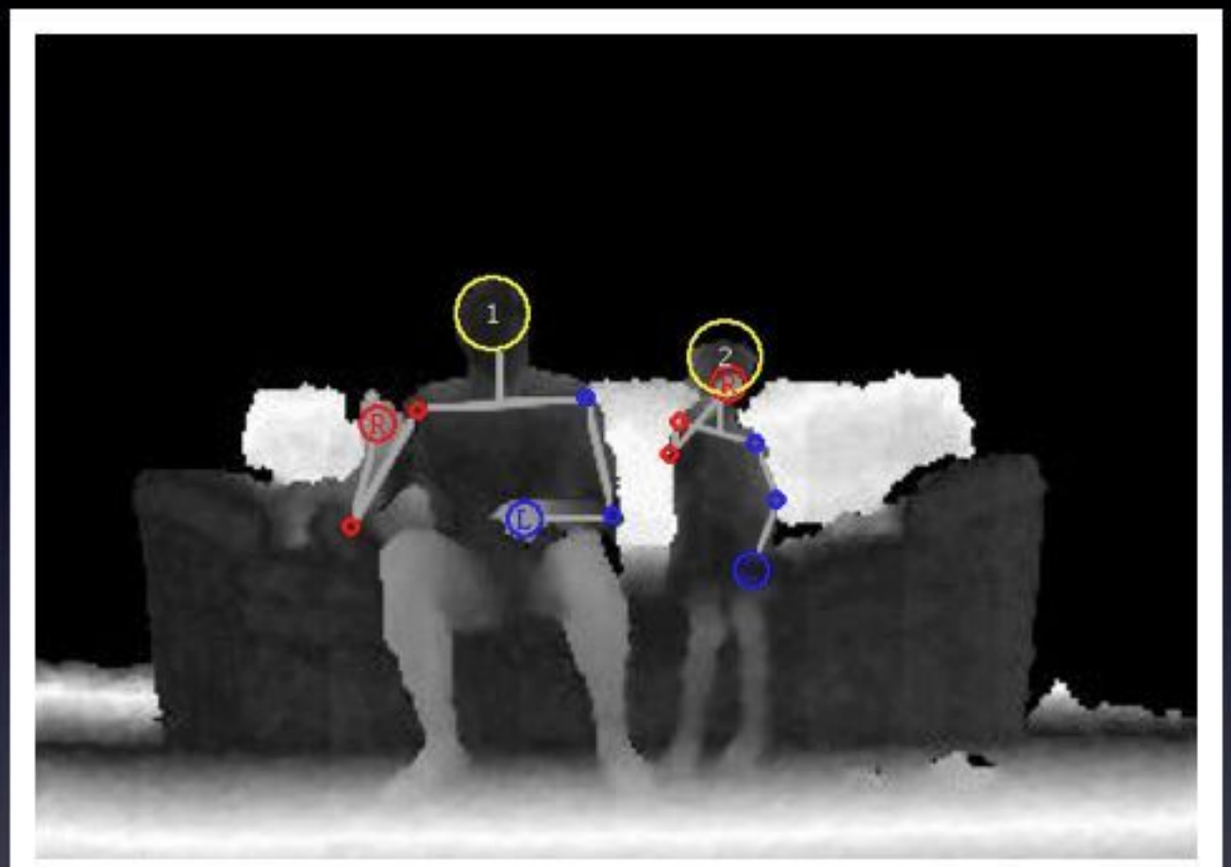
- La telecamera sensibile agli infrarossi, "3D depth camera", di Kinect riprende la scena (crea un'"immagine di profondità") e valuta come questi punti vengono distorti rispetto alla loro posizione ideale, da questa e altre informazioni riesce a determinare la distanza degli oggetti nella stanza e la loro conformazione.

Kinect: funzionamento



Sarà poi compito del software identificare numero, posizione, giunture degli esseri umani prensenti nella scena.

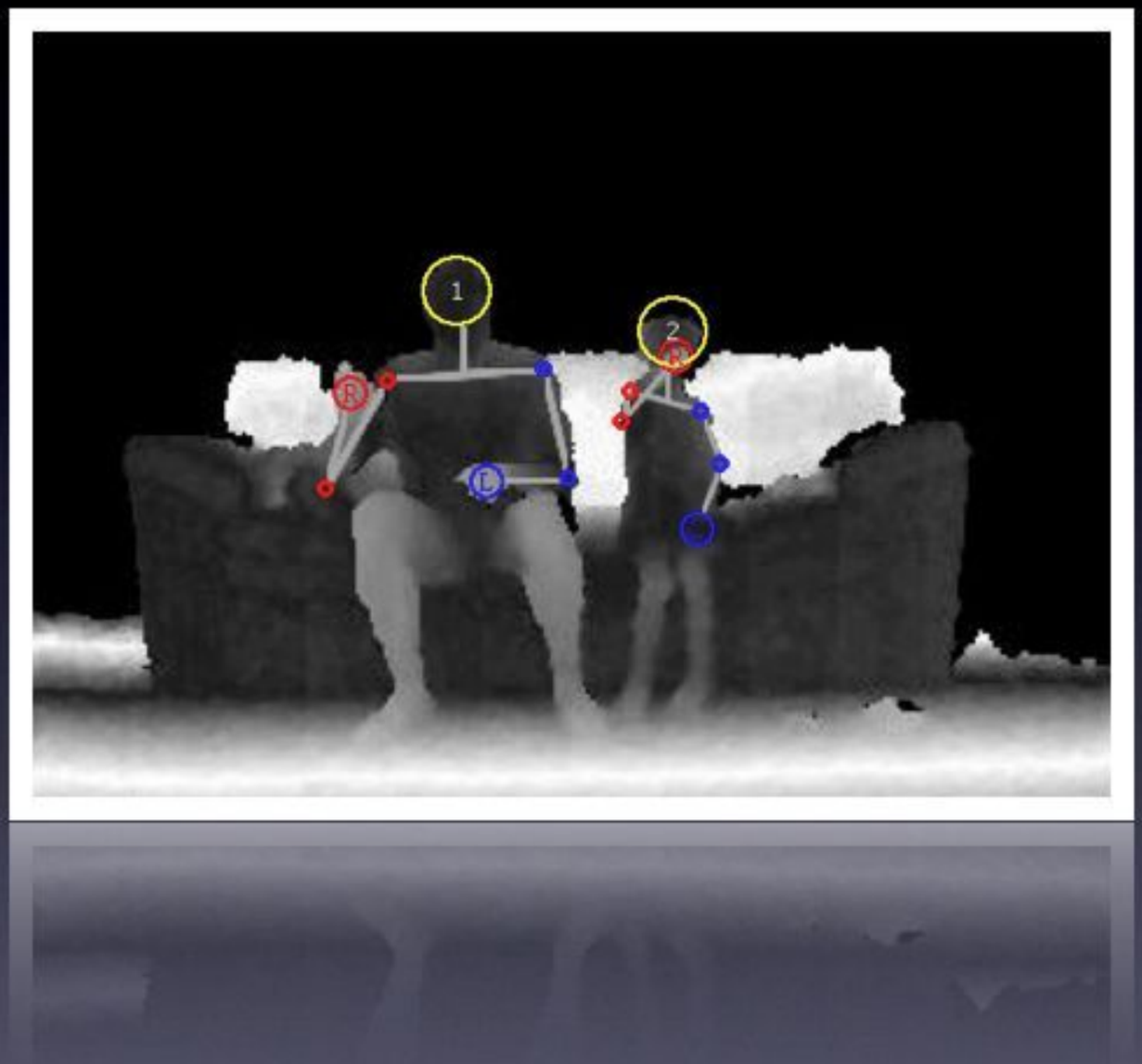
Kinect: funzionamento



Sarà poi compito del software identificare numero, posizione, giunture degli esseri umani prensenti nella scena.

Microsoft dichiara un'analisi su 200 frames al secondo XBox360.

Kinect: funzionamento

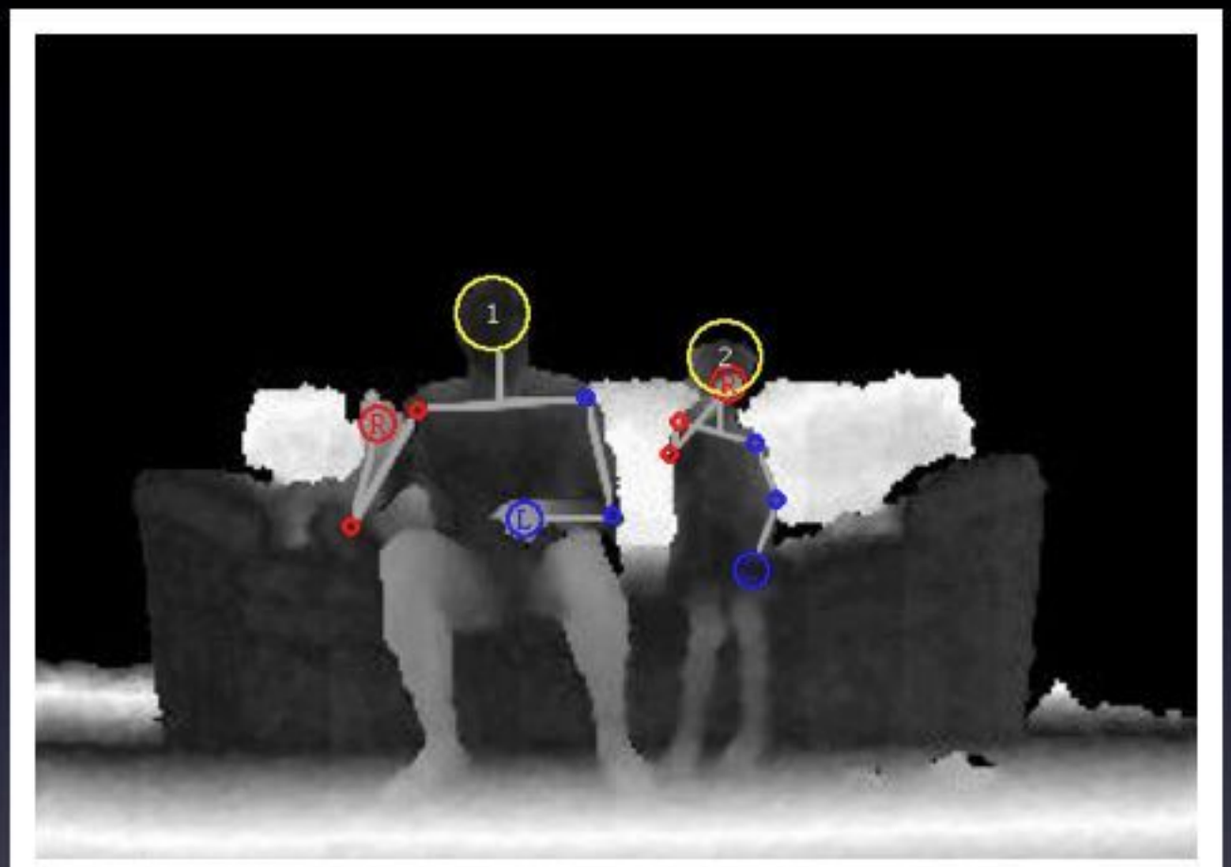


Sarà poi compito del software identificare numero, posizione, giunture degli esseri umani prensenti nella scena.

Microsoft dichiara un'analisi su 200 frames al secondo XBox360.

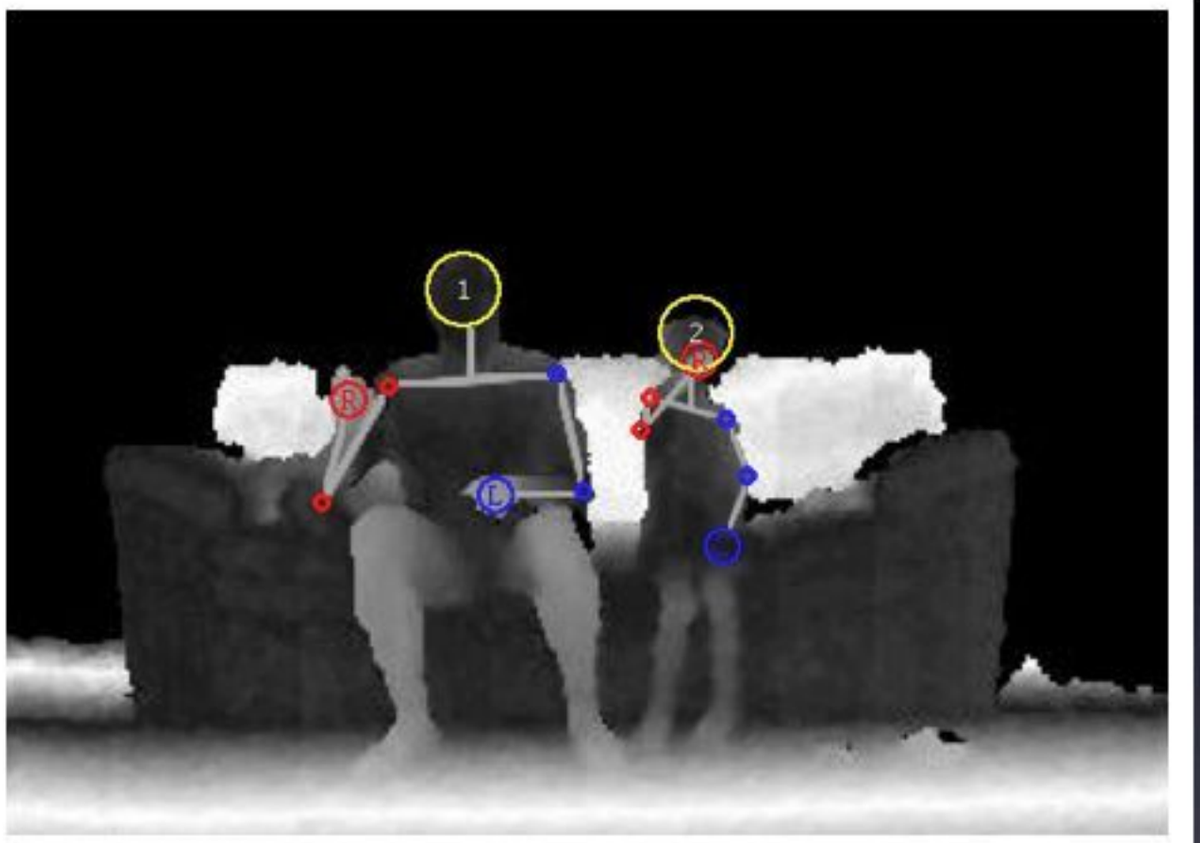
Tracking di massimo 2 persone

Kinect: funzionamento



Tracking di massimo 2 persone
PRIMESENSE dichiara che il dispositivo ha le potenzialità di identificare i movimenti di tutti gli utenti che sono presenti nella stanza!

Kinect: funzionamento



Il tracking di Microsoft è frutto di un'analisi su 500000 campioni di dati registrati da diversi comportamenti umani (ballo, spostamento, saluto, etc), di questi 100000 sono stati considerati utili e hanno consentito di "insegnare" al software proprietario le posizioni delle giunture del corpo.

Kinect: hacking contest

- 4 novembre 2010: Commercializzazione
- 4 novembre: Adafruit Industries, Hacking contest, premio 1000$
- 5 novembre: Microsoft risponde all'hacking contest;
- 5 novembre: Adafruit alza il premio a 2000$;
- 6 novembre: AlexP controlla i motori di Kinect's
- 6 novembre: Adafruit rilancia a 3000€
- 8 novembre: AlexP controlla le due telecamere, non rilascia i sorgenti e chiede un "contributo" di 10.000€ per rendere pubblico il proprio lavoro.
- 9 novembre: Adafruit rilascia i dati di comunicazione su USB di Kinect;
- 10 novembre: Héctor Martín rilascia i sorgenti : Kinect è HACKED;

http://www.newscientist.com/article/dn19762-inside-the-race-to-hack-the-kinect.html

Kinect: hacking USB

1. Identificazione VID (vendor id) & PID (product id)

lsusb (Linux) / system_profiler SPUSBDataType (Mac):

4 USB devices: a hub, camera, microphone (audio), motor.

- http://www.ladyada.net/learn/diykinect/

Kinect: hacking USB

```
adafruit:~ ada$ system_profiler SPUSBDataType
USB:

    USB High-Speed Bus:

      Host Controller Location: Built In USB
      Host Controller Driver: AppleUSBEHCI
      PCI Device ID: 0x00e0
      PCI Revision ID: 0x0004
      PCI Vendor ID: 0x1033
      Bus Number: 0x5b

        Hub:

          Version: 1.00
          Bus Power (mA): 500
          Speed: Up to 480 Mb/sec
          Product ID: 0x005a
          Vendor ID: 0x0409

          Xbox NUI Camera:

            Version: 1.0b
            Bus Power (mA): 500
            Speed: Up to 480 Mb/sec
            Manufacturer: Microsoft
            Product ID: 0x02ae
            Serial Number: A00363908978039A
            Vendor ID: 0x045e

          Xbox NUI Audio:

            Version: 1.00
            Bus Power (mA): 500
            Speed: Up to 480 Mb/sec
            Manufacturer: Microsoft
            Product ID: 0x02ad
            Serial Number: A44882D01681039A
            Vendor ID: 0x045e

          Xbox NUI Motor:

            Version: 1.05
            Bus Power (mA): 500
            Speed: Up to 12 Mb/sec
            Manufacturer: Microsoft
            Product ID: 0x02b0
            Vendor ID: 0x045e
```

Kinect: hacking USB

1. Identificazione VID (vendor id) & PID (product id)
2. Identificazione descriptor; lsusb -vv (Linux);

Kinect: hacking USB

```
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB               2.00
  bDeviceClass            0 (Defined at Interface level)
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0        64
  idVendor           0x045e Microsoft Corp.
  idProduct          0x02ae
  bcdDevice            1.0b
  iManufacturer           2 Microsoft
  iProduct                1 Xbox NUI Camera
  iSerial                 3 A00366A08793039A
  bNumConfigurations      1
  Configuration Descriptor:
    bLength                9
    bDescriptorType        2
    wTotalLength          32
    bNumInterfaces         1
    bConfigurationValue    1
    iConfiguration         0
    bmAttributes        0xc0
      Self Powered
    MaxPower              16mA
    Interface Descriptor:
      bLength              9
      bDescriptorType      4
      bInterfaceNumber     0
      bAlternateSetting    0
      bNumEndpoints        2
      bInterfaceClass    255 Vendor Specific Class
      bInterfaceSubClass 255 Vendor Specific Subclass
      bInterfaceProtocol 255 Vendor Specific Protocol
      iInterface           0
      Endpoint Descriptor:
        bLength            7
        bDescriptorType    5
        bEndpointAddress  0x81  EP 1 IN
        bmAttributes       1
          Transfer Type            Isochronous
```

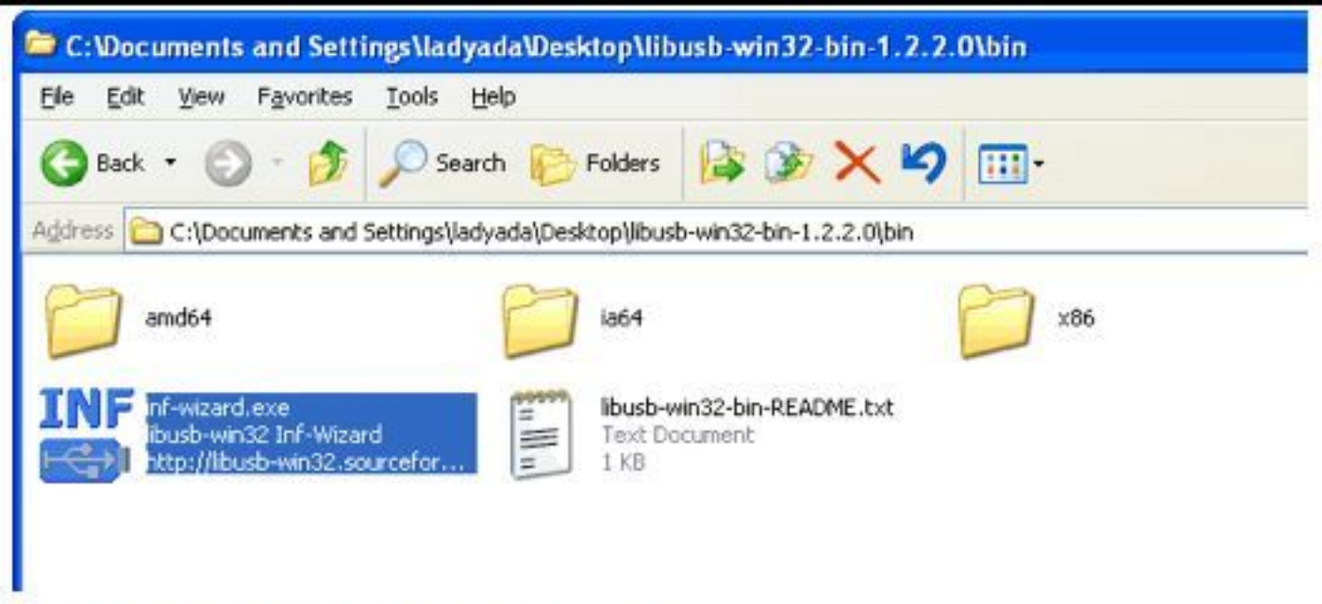
Kinect: hacking USB

```
Interface Descriptor:
  bLength                 9
  bDescriptorType         4
  bInterfaceNumber        0
  bAlternateSetting       0
  bNumEndpoints           2
  bInterfaceClass       255 Vendor Specific Class
  bInterfaceSubClass    255 Vendor Specific Subclass
  bInterfaceProtocol    255 Vendor Specific Protocol
  iInterface              0
  Endpoint Descriptor:
    bLength                 7
    bDescriptorType         5
    bEndpointAddress     0x81  EP 1 IN
    bmAttributes            1
      Transfer Type             Isochronous
      Synch Type                None
      Usage Type                Data
    wMaxPacketSize     0x0bc0  2x 960 bytes
    bInterval               1
  Endpoint Descriptor:
    bLength                 7
    bDescriptorType         5
    bEndpointAddress     0x82  EP 2 IN
    bmAttributes            1
      Transfer Type             Isochronous
      Synch Type                None
      Usage Type                Data
    wMaxPacketSize     0x0bc0  2x 960 bytes
    bInterval               1
Device Qualifier (for other device speed):
  bLength                10
  bDescriptorType         6
  bcdUSB               2.00
  bDeviceClass            0 (Defined at Interface level)
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0        64
  bNumConfigurations      1
Device Status:     0x0001
  Self Powered
```

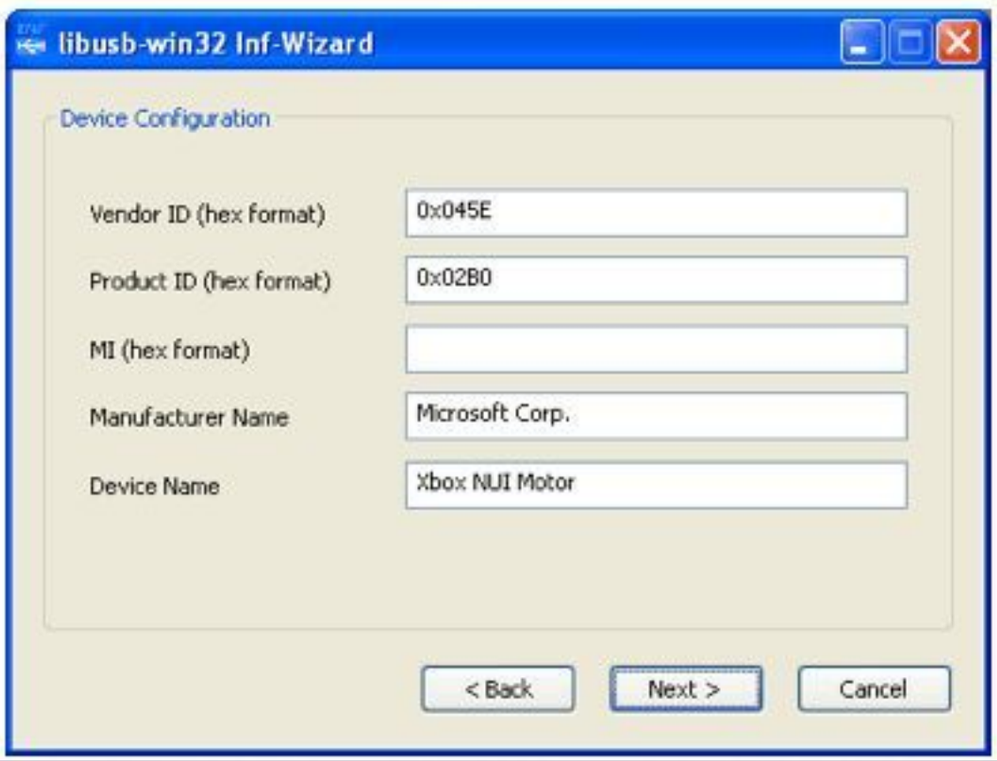
Kinect: hacking USB

1. Identificazione VID (vendor id) & PID (product id)
2. Identificazione descriptor; lsusb -vv (Linux);
3. (Windows) installazione libreria libUSB

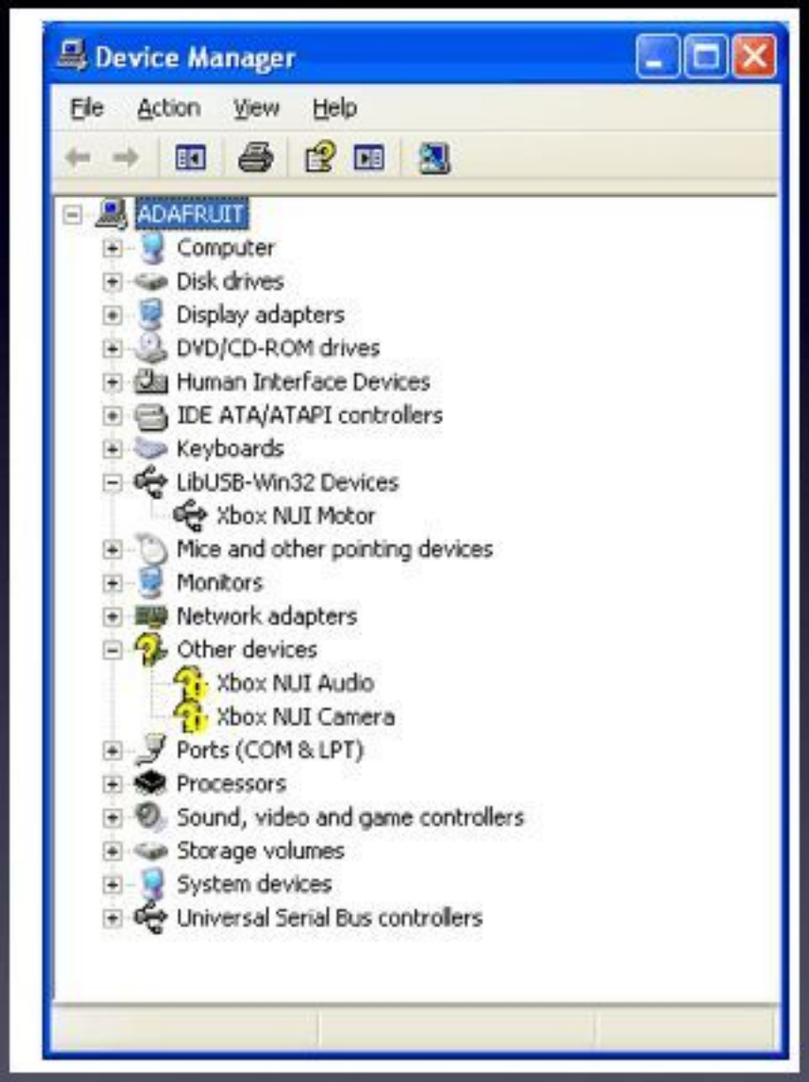
Kinect: hacking USB



The important part is entering in the matching VID and PID we found before



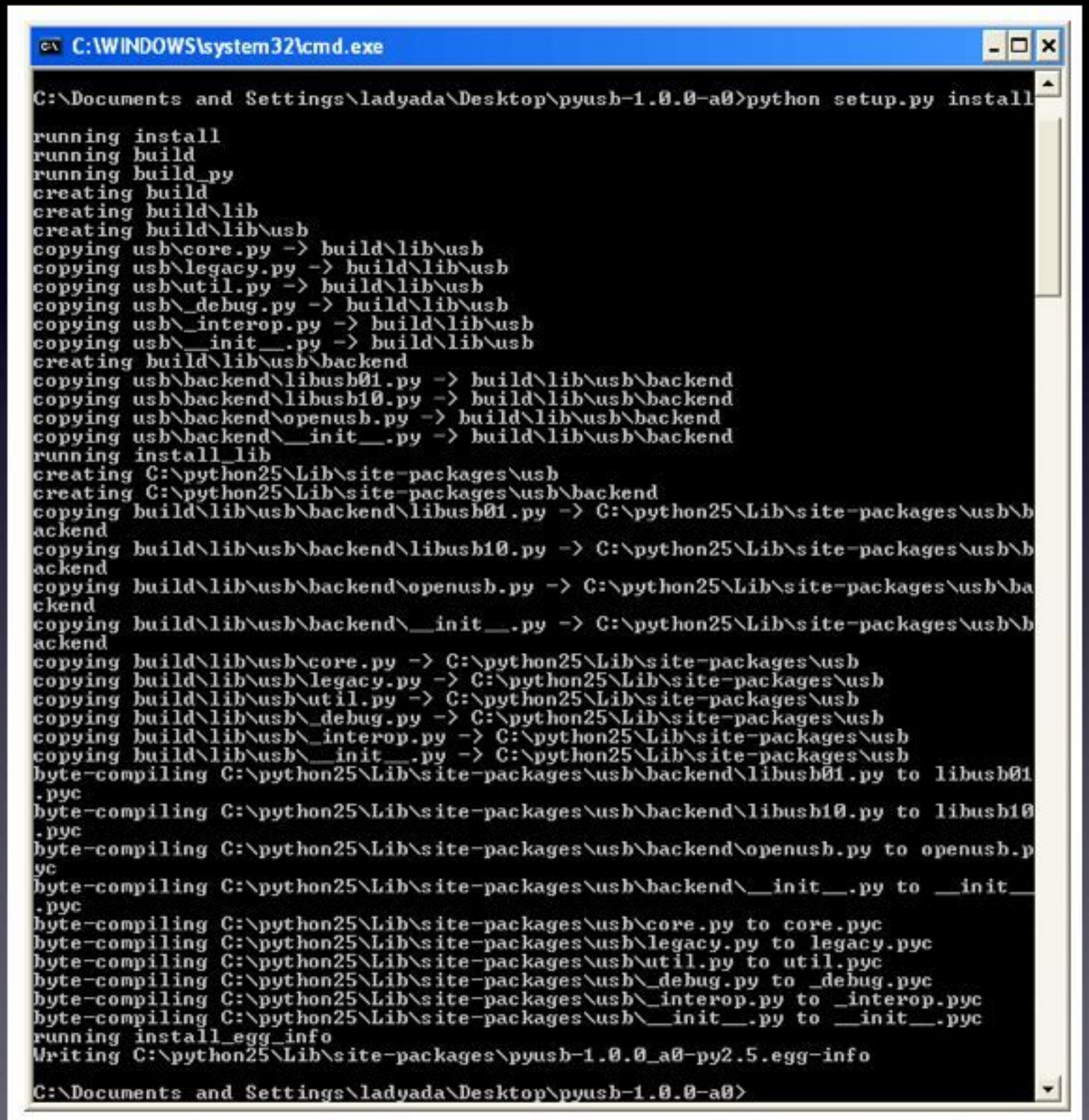
Kinect: hacking USB



Kinect: hacking USB

1. Identificazione VID (vendor id) & PID (product id)
2. Identificazione descriptor; lsusb -vv (Linux);
3. (Windows) installazione libreria libUSB
4. Installazione Python & PyUSB

Kinect: hacking USB



Kinect: hacking USB

1. Identificazione VID (vendor id) & PID (product id)
2. Identificazione descriptor; lsusb -vv (Linux);
3. (Windows) installazione libreria libUSB
4. Installazione Python & PyUSB
5. Listening...

Kinect: hacking USB

```
import usb.core
import usb.util
import sys

# find our device
dev = usb.core.find(idVendor=0x045e, idProduct=0x02B0)

# was it found?
if dev is None:
    raise ValueError('Device not found')

dev.set_configuration()

# Lets start by Reading 1 byte from the Device using different Requests
# bRequest is a byte so there are 255 different values
for bRequest in range(255):
    try:
        ret = dev.ctrl_transfer(0xC0, bRequest, 0, 0, 1)
        print "bRequest ",bRequest
        print ret
    except:
        # failed to get data for this request
        pass
```

Kinect: hacking USB

1. Identificazione VID (vendor id) & PID (product id)
2. Identificazione descriptor; lsusb -vv (Linux);
3. (Windows) installazione libreria libUSB
4. Installazione Python & PyUSB
5. USB dongle...

Kinect: hacking USB

Kinect: hacking USB

enuminit - Total Phase Data Center

File Edit Analyzer View Help

518.10 MB

Sp	Index	m:s.ms.us	Len	Err	Dev	Ep	Record	Summary
	0	0:00.000.000					Capture started (Aggregate)	[11/09/10 21:41:02]
HS	3	0:07.934.342	1 B	I			OUT packet	E1
HS	1154	0:09.912.223	18 B		04	00	Get Device Descriptor	Index=0 Length=18
HS	1184	0:09.913.197	18 B		04	00	Get Configuration Descriptor	Index=0 Length=336
HS	1214	0:09.914.322	0 B		04	00	Set Configuration	Configuration=1
HS	1265	0:09.992.783	1 B		04	00	Control Transfer	22
HS	1295	0:09.993.494	0 B		04	00	Control Transfer	
HS	1346	0:10.059.574	8 B		04	00	Control Transfer	00 01 00 01 07 E9 00 00
HS	1376	0:10.076.102	8 B		04	00	Control Transfer	00 01 00 01 07 E9 00 00
HS	1406	0:10.092.781	1 B		04	00	Control Transfer	22
HS	11010	0:14.645.721	32 B		04	00	Control Transfer	31 00 30 00 31 00 37 00 37 00 32 00 33 00 30 00 33 00 39 00 33 00 35 00..
HS	31581	0:19.448.664	4 B		04	00	Control Transfer	00 00 00 00
HS	31719	0:19.451.727	2 B		04	00	Control Transfer	00 00
HS	33045	0:19.468.229	10 B		04	00	Control Transfer	00 00 00 1E 03 28 FF 79 EE 00
HS	33315	0:19.469.221	1 B		04	00	Control Transfer	80
HS	34265	0:19.488.249	10 B		04	00	Control Transfer	00 00 00 15 03 28 FF 79 EE 00
HS	34322	0:19.489.100	1 B		04	00	Control Transfer	80
HS	34922	0:19.505.253	10 B		04	00	Control Transfer	00 00 00 15 03 29 FF 77 EE 00
HS	34982	0:19.506.352	1 B		04	00	Control Transfer	80
HS	35571	0:19.522.254	10 B		04	00	Control Transfer	00 00 00 15 03 26 FF 7B EE 00
HS	35709	0:19.523.229	1 B		04	00	Control Transfer	80
HS	36294	0:19.541.256	10 B		04	00	Control Transfer	00 00 00 25 03 28 FF 87 EE 00
HS	36351	0:19.542.232	1 B		04	00	Control Transfer	80
HS	36947	0:19.558.258	10 B		04	00	Control Transfer	00 00 00 1E 03 28 FF 87 EE 00
HS	37085	0:19.559.232	1 B		04	00	Control Transfer	80
HS	37705	0:19.578.248	10 B		04	00	Control Transfer	00 00 00 15 03 28 FF 7F EE 00

Kinect: hacking USB

Sp	Index	m:s.ms.us	Len	Err	Dev	Ep	Record	Summary
	0	0:00.000.000					Capture started (Aggregate)	[11/09/10 21:41:02]
HS	1214	0:09.914.322	0 B		04	00	Set Configuration	Configuration=1
HS	1295	0:09.993.494	0 B		04	00	Control Transfer	
HS	922558	0:34.505.982	0 B		04	00	Control Transfer	
HS	952254	0:34.774.137	0 B		04	00	Control Transfer	
HS	1237129	0:37.341.197	0 B		04	00	Control Transfer	
	2140385	0:45.487.878					Capture stopped	[11/09/10 21:41:49]

Navigator

SETUP Transaction

Transaction	Radix: auto
Timestamp	0:37.341.197.566
Duration	17.416 us
Length	8 Bytes
Type	SETUP
Device	4
Endpoint	0
Data	0x40 0x31 0xF0 ...
Status	ACK

SETUP Data	Radix: auto
bmRequestType.Recipient	Device (0b00000)
bmRequestType.Type	Vendor (0b10)
bmRequestType.Direction	Host-to-Device (0b0)
bRequest	0x31
wValue	0xfff0
windex	0x0000
wLength	0x0000

Navigator

SETUP Transaction

Transaction	Radix: auto
Timestamp	0:34.774.137.033
Duration	17.716 us
Length	8 Bytes
Type	SETUP
Device	4
Endpoint	0
Data	0x40 0x31 0xD0 ...
Status	ACK

SETUP Data	Radix: auto
bmRequestType.Recipient	Device (0b00000)
bmRequestType.Type	Vendor (0b10)
bmRequestType.Direction	Host-to-Device (0b0)
bRequest	0x31
wValue	0xffd0
windex	0x0000
wLength	0x0000

Kinect: hacking USB

1. Identificazione VID (vendor id) & PID (product id)
2. Identificazione descriptor; lsusb -vv (Linux);
3. (Windows) installazione libreria libUSB
4. Installazione Python & PyUSB
5. USB dongle...
6. invio comandi...

Kinect: hacking USB

```
import usb.core
import usb.util
import sys

# find our device
dev = usb.core.find(idVendor=0x045e, idProduct=0x02B0)

if dev is None:
    raise ValueError('Device not found')

dev.set_configuration()

ret = dev.ctrl_transfer(0x40, 0x6, 0x1, 0, [])
print ret
```

Kinect: hacking USB

1. Identificazione VID (vendor id) & PID (product id)
2. Identificazione descriptor; lsusb -vv (Linux);
3. (Windows) installazione libreria libUSB
4. Installazione Python & PyUSB
5. USB dongle...
6. invio comandi...test...invio comandi...test....

AVVERTENZE

Passare ore davanti ad un dispositivo Kinect/compatibile durante la fase di sviluppo è sconsigliato a lungo termine, questo perchè verrete investiti continuamente da centinaia di fasci ad infrarossi, anche negli occhi.

SVILUPPATE PUNTANDO IL DISPOSITIVO IN UN'ALTRA AREA DELLA STANZA

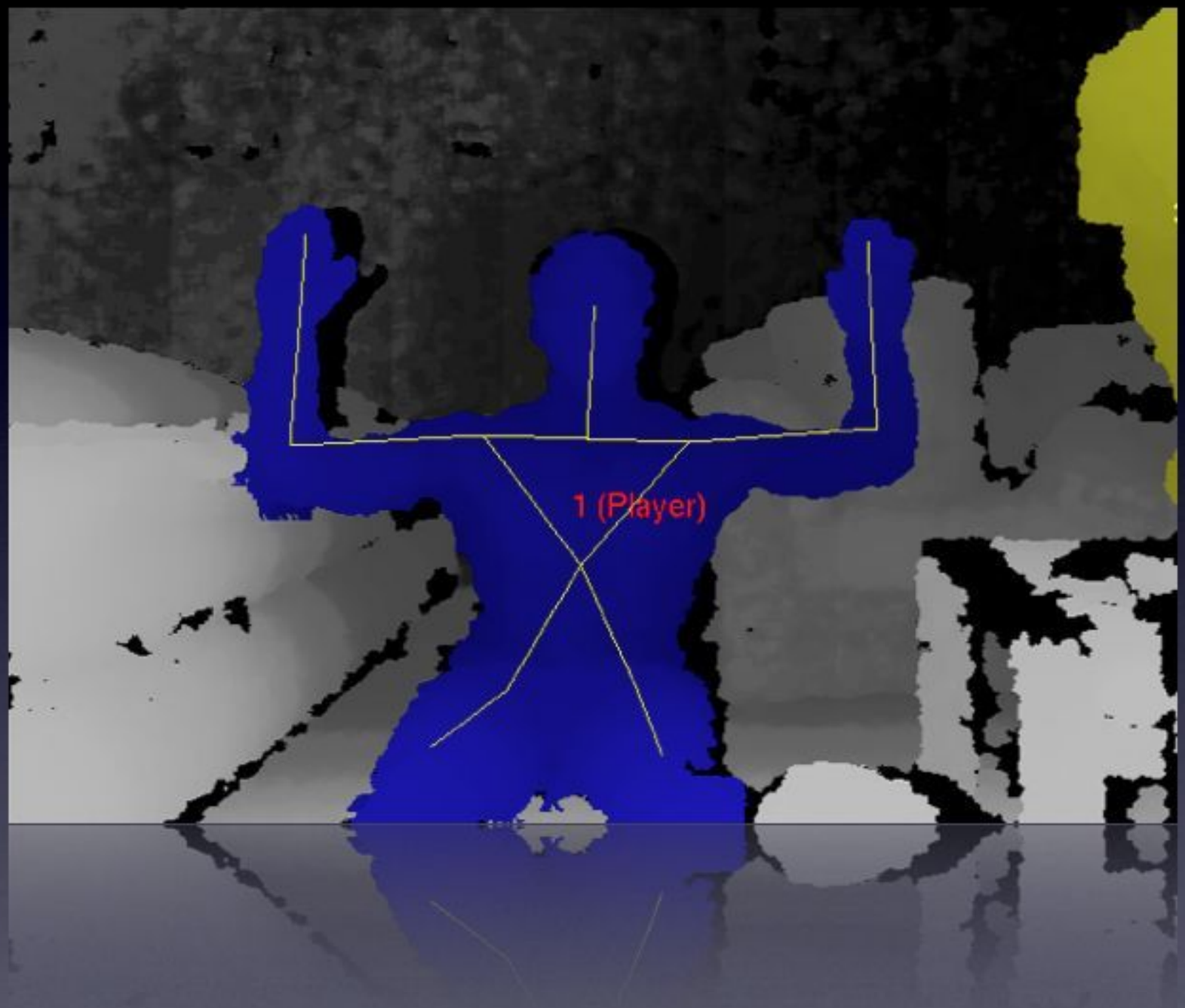
v

ACCENDETE IL DISPOSITIVO SOLO QUANDO NECESSARIO

v

UTILIZZATE DATI DI SIMULAZONE (OpenKinect only).

Y-POSE



La Y-Pose è una posizione necessaria per consentire al sistema di identificare correttamente il corpo e i suoi segmenti.

La Y-Pose è necessaria quando si usano tecnologie open source != Microsoft Kinect SDK

Kinect: progetti

- Controllo remoto periferiche (robot, bracci meccanici, strumenti musicali, aspirapolveri...);
- Riconoscimento/insegnamento linguaggio dei segni;
- Disegno a mano libera;
- Interfacciamento computer;
- Controllo modelli 3D;
- Motion capture;
- Sorveglianza;
- Scanning oggetti 3D;
- Avatar 2D/3D in realtime;
- Navigazione all'interno di ambienti 2D/3D

Kinect: progetti

VIDEO

Kinect: SDKs

OpenKinect (libfreenect)

Open Source
Multi-platform
Community

VS

OpenNI

Open Source
Multi-platform
Primesense

Microsoft SDK

Closed Source
Single-platform
Microsoft

Kinect: SDKs

OpenNI VS OpenKinect (libfreenect)

Maker: Primesense
Source: Primesense
License: LGPL
No motors
No audio
Not only Kinect/Asus Xtion

Maker: community
Source: reveresed Kinect data
License: Apache 2 / GPL 2
RGB and Depth Images
Motors
Accelerometer
LED
No audio (in developmente)
> Languages
https://github.com/OpenKinect/libfreenect

Kinect: SDKs

OpenNI VS Microsoft SDK beta

Open Source
Multi-platform
Commercial License
Asus Xtion compatible
Y-pose
ROS (Robot Operating System)

Kinect: SDKs

OpenNI VS Microsoft SDK beta

Open Source
Multi-platform
Commercial License
Asus Xtion compatible
Y-pose
ROS (Robot Operating System)

Closed Source
Windows 7 only
Non commercial
Commercial "soon" (2012)
Microsoft Robotics Developer Studio

Kinect: linguaggi

OpenNI VS Microsoft SDK beta

C++
C#
Java (JNA and JNI)

Kinect: linguaggi

OpenNI VS Microsoft SDK beta

C++
C#
Java (JNA and JNI)

C++
C#
VB.NET

Kinect: funzionalità SDK

OpenNI VS Microsoft SDK (beta)

- **includes a framework for hand tracking**
- **includes a framework for hand gesture recognition**
- can automatically align the depth image stream to the color image
- full body tracking:
- **calculates rotations for the joints**
- support for hands only mode
- consumes a bit less CPU
- **supports the Primesense and the ASUS WAVI Xtion sensors**
- supports multiple sensors although setup and enumeration is a bit quirky
- **supports Windows (including Vista&XP), Linux and Mac OSX**
- comes with code for full support in Unity3D game engine
- **support for record/playback to/from disk**
- support to stream the raw InfraRed video data
- **SDK has events for when new User enters frame, leaves frame etc**

Kinect: funzionalità SDK

OpenNI VS Microsoft SDK (beta)

- **includes a framework for hand tracking**
- **includes a framework for hand gesture recognition**
- can automatically align the depth image stream to the color image
- full body tracking:
- **calculates rotations for the joints**
- support for hands only mode
- consumes a bit less CPU
- **supports the Primesense and the ASUS WAVI Xtion sensors**
- supports multiple sensors although setup and enumeration is a bit quirky
- **supports Windows (including Vista&XP), Linux and Mac OSX**
- comes with code for full support in Unity3D game engine
- **support for record/playback to/from disk**
- support to stream the raw InfraRed video data
- **SDK has events for when new User enters frame, leaves frame etc**

- **support for audio**
- **support for motor/tilt**
- full body tracking:
- **does not need a calibration pose (Y-pose)**
- includes head, hands, feet, clavicles
- supports multiple sensors
- **single installer**
- SDK uses events when new Video/Depth frame is available

OpenNI

OpenNI è un framework Open Source sviluppato da Primesense e altre società. L'API consente di gestire la comunicazione tra dispositivi hardware di "basso" livello come sensori visivi e acustici ma anche tra differenti middleware, consente inoltre l'invio dei dati a moduli in grado di analizzare i dati inviati da questi sensori.

OpenNI

OpenNI è un framework Open Source sviluppato da Primesense e altre società. L'API consente di gestire la comunicazione tra dispositivi hardware di "basso" livello come sensori visivi e acustici ma anche tra differenti middleware, consente inoltre l'invio dei dati a moduli in grado di analizzare i dati inviati da questi sensori.

OpenNI è quindi un livello di astrazione che consente di nascondere ai livelli superiori (e parallelamente tra i moduli interni) la complessità tecnologica dei singoli sensori.

OpenNI

OpenNI è un framework Open Source sviluppato da Primesense e altre società. L'API consente di gestire la comunicazione tra dispositivi hardware di "basso" livello come sensori visivi e acustici ma anche tra differenti middleware, consente inoltre l'invio dei dati a moduli in grado di analizzare i dati inviati da questi sensori.

OpenNI è quindi un livello di astrazione che consente di nascondere ai livelli superiori (e parallelamente tra i moduli interni) la complessità tecnologica dei singoli sensori.

OpenNI non è perciò legato solamente a Kinect bensì si pone come una soluzione software adatta ad un utilizzo con qualunque tipo di tecnologia di qualsiasi tipo di complessità.

OpenNI

OpenNI è un framework Open Source sviluppato da Primesense e altre società. L'API consente di gestire la comunicazione tra dispositivi hardware di "basso" livello come sensori visivi e acustici ma anche tra differenti middleware, consente inoltre l'invio dei dati a moduli in grado di analizzare i dati inviati da questi sensori.

OpenNI è quindi un livello di astrazione che consente di nascondere ai livelli superiori (e parallelamente tra i moduli interni) la complessità tecnologica dei singoli sensori.

OpenNI non è perciò legato solamente a Kinect bensì si pone come una soluzione software adatta ad un utilizzo con qualunque tipo di tecnologia di qualsiasi tipo di complessità.

http://arena.openni.org/

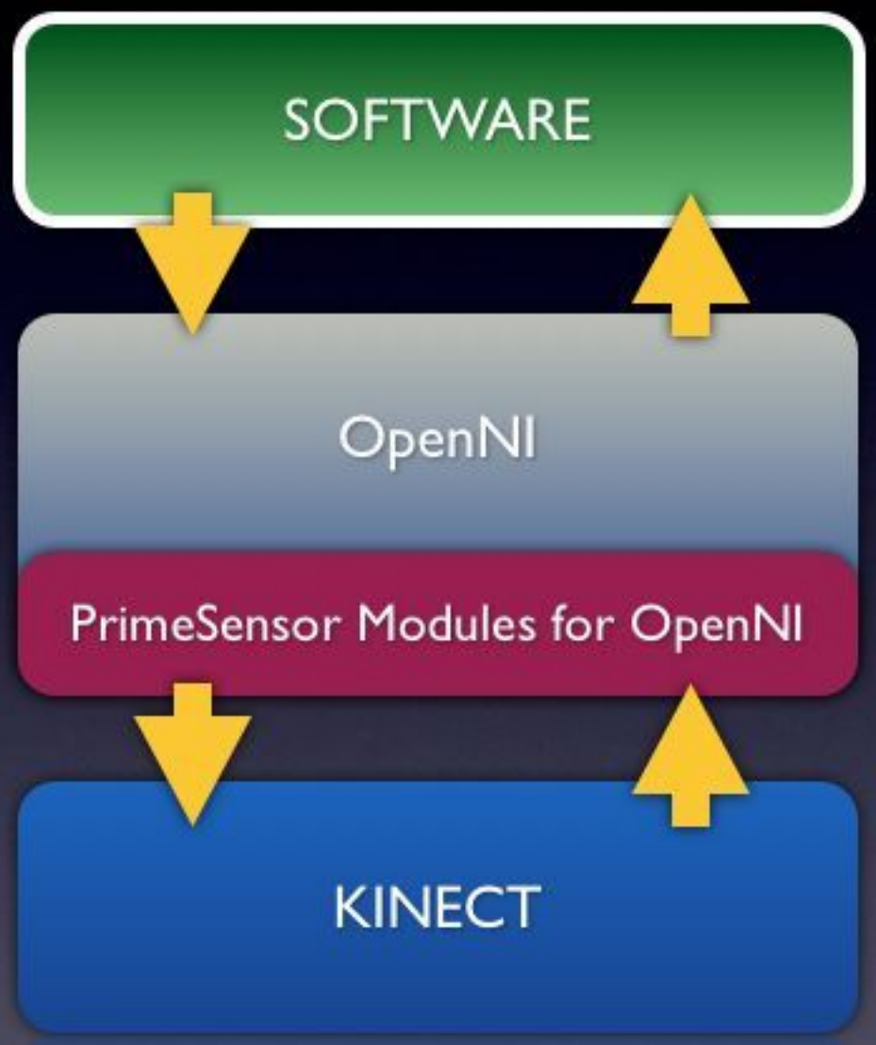
Kinect: controllo remoto

Kinect: controllo remoto

Kinect: controllo remoto

Kinect: controllo remoto

Kinect: controllo remoto



NITE

NITE è un middleware realizzato da PrimeSense che utilizza OpenNI allo scopo di fornire:

- Gesture Generator
- Hands Generator

Richiede utilizzare un codice in fase di installazione:
0KOIk2JeIBYClPWVnMoRKn5cdY4=

NITE: CALIBRAZIONE

NITE NON richiede la Y-pose, è sufficiente effettuare una delle seguenti azioni con una mano:

NITE: CALIBRAZIONE

NITE NON richiede la Y-pose, è sufficiente effettuare una delle seguenti azioni con una mano:

Click: l'utente con la mano aperta e il palmo in direzione della telecamera la muove verso la telecamera e torna indietro;

NITE: CALIBRAZIONE

NITE NON richiede la Y-pose, è sufficiente effettuare una delle seguenti azioni con una mano:

Click: l'utente con la mano aperta e il palmo in direzione della telecamera la muove verso la telecamera e torna indietro;

Wave: l'utente scuote la mano come avviene per un normale saluto almeno cinque volte;

NITE: SESSION

Dopo l'identificazione della mano viene create una SESSIONE, durante la quale il middleware comunica al nostro software le informazioni sulla sua posizione;

NITE: SESSION

Dopo l'identificazione della mano viene create una SESSIONE, durante la quale il middleware comunica al nostro software le informazioni sulla sua posizione;

Durante la SESSIONE è possibile identificare alcuni movimenti adoperando i cosiddetti DETECTORS, questi vengono scatenati se la mano compie alcuni movimenti.

NITE: Gesture Detector

Push Detector: cerca di identificare un movimento avanti-indietro nella direzione della telecamera, nella pratica è un pugno;

Swipe Detector: cerca di identificare un movimento laterale della mano;

Steady Detector: cerca di identificare la staticità prolungata di una mano;

Wave Detector: cerca di identificare un movimento in cui avviene per ben quattro volte un cambio di direzione, nella pratica è il gesto del comune saluto;

CircleDetector: cerca di identificare un movimento circolare in senso orario o antiorario completo della mano;

SelectableSlider1D: cerca di identificare un movimento lungo uno dei tre assi, viene adoperato ad esempio per spostarsi all'interno delle voci di un menu contestuale;

SelectableSlider2D: cerca di identificare un movimento prima nel piano X-Y (parallelo all'immagine ripresa dalla telecamera) e poi una selezione effettuando un movimento lungo l'asse Z (come avviene per il push);

NITE: Punto Primario

NITE non invia tutti i punti identificati ai controlli ma solo uno, chiamato **punto primario (primary point)** che è il primo punto che il sistema identifica con la telecamera

NITE: Punto Primario

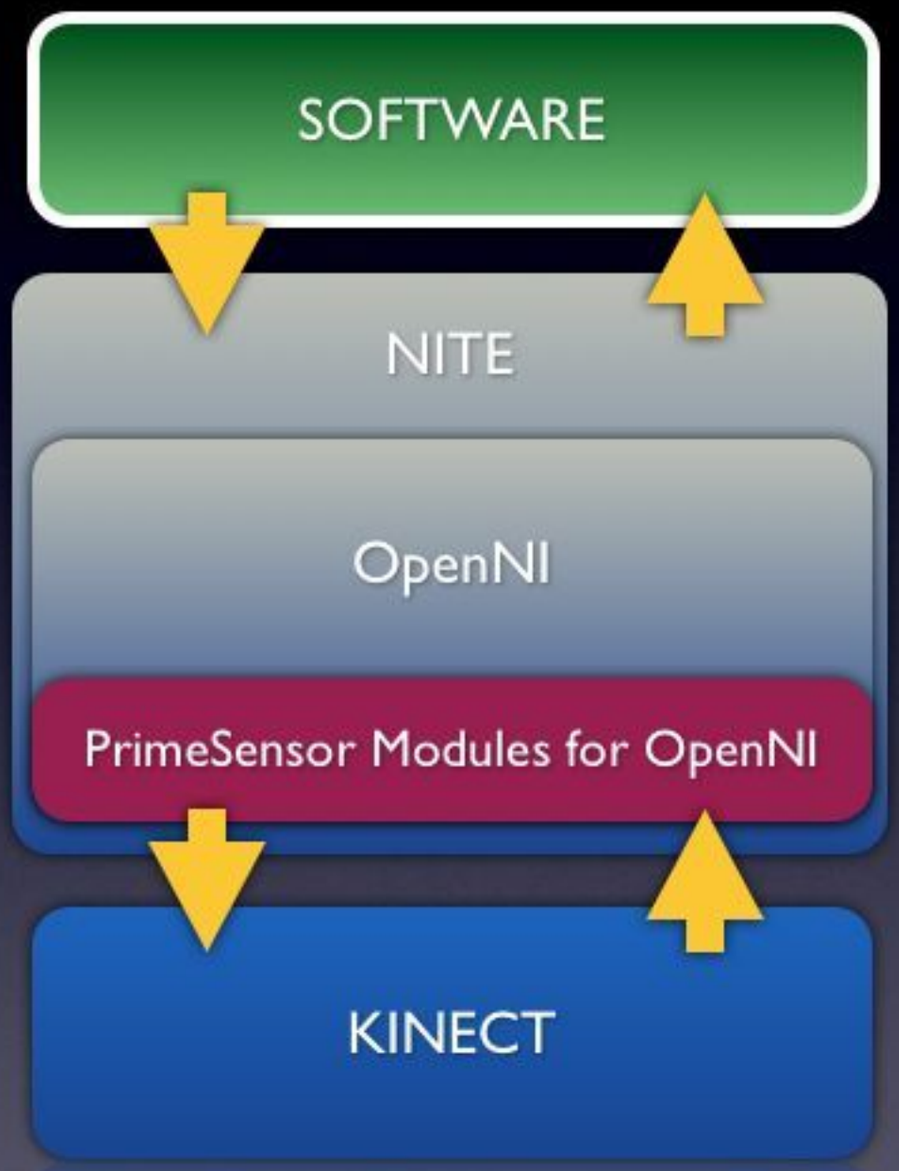
NITE non invia tutti i punti identificati ai controlli ma solo uno, chiamato **punto primario (primary point)** che è il primo punto che il sistema identifica con la telecamera

Il punto primario ha quindi un'importanza fondamentale per il tracking dei movimenti, e ad esso sono associati quattro specifici eventi invece dei tre adoperati per gli altri punti, che ripetiamo sono entrata nel campo della telecamera, spostamento, uscita dal campo della telecamera: per il punto primario si ricevono "creazione del punto primario", "aggiornamento", "distruzione" e "sostituzione con un altro punto".

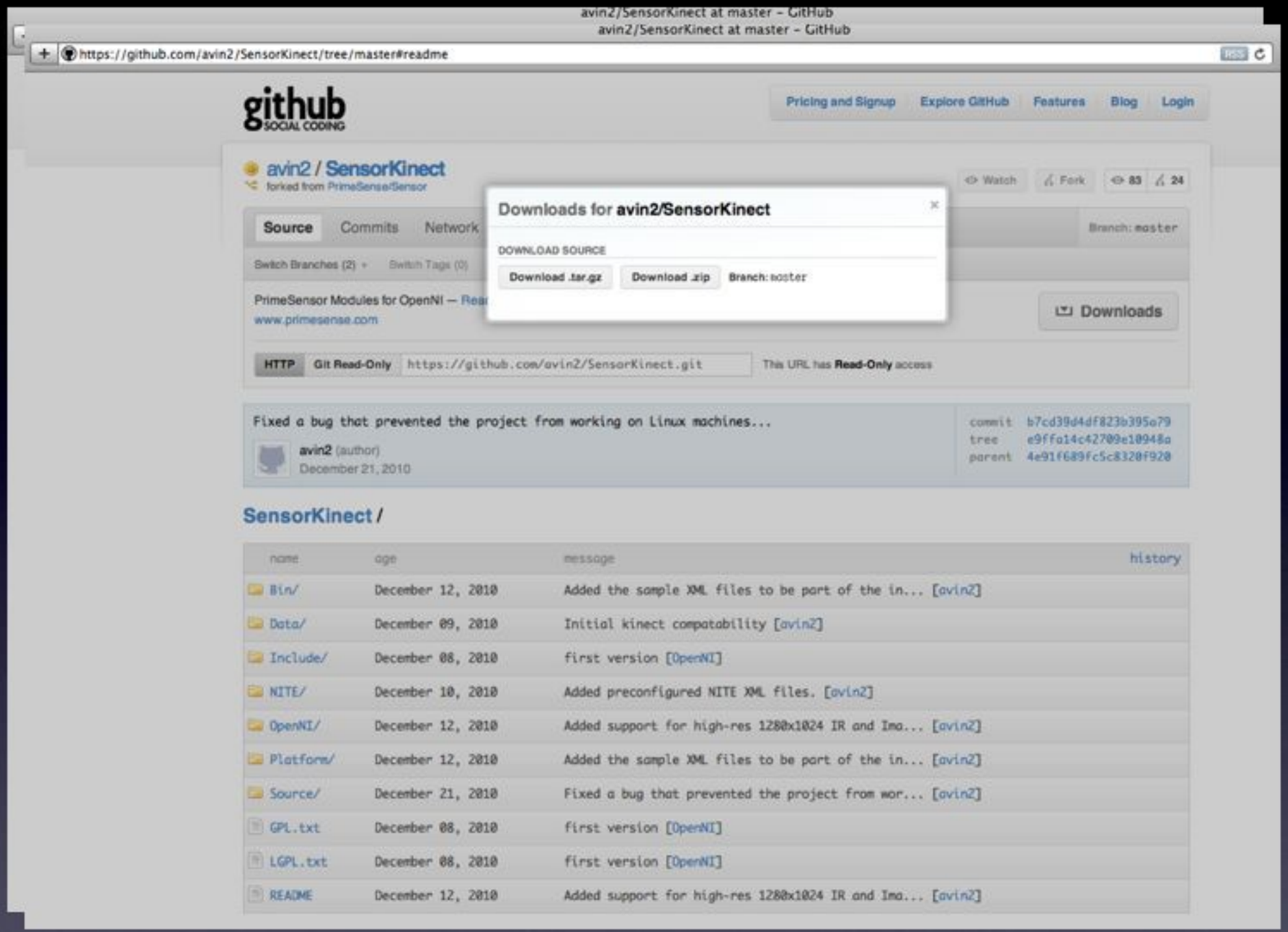
NITE: Punto Primario

VIDEO

Kinect: controllo remoto



Download: Drivers



https://github.com/avin2/SensorKinect

kinect ▸ avin2-SensorKinect-b7cd39d ▸ avin2-SensorKinect-b7cd39d ▸ Bin

Search Bin

Include in library | Share with | Burn | New folder

Name	Date modified	Type	Size

kinect ▸ avin2-SensorKinect-b7cd39d ▸ avin2-SensorKinect-b7cd39d ▸ Bin

Search Bin

Include in library | Share with | Burn | New folder

Name	Date modified	Type	Size
SensorKinect-Win32-5.0.0.exe	21/12/2010 14:36	Application	9.047 KB

PrimeSensor 5.0.0 for Windows (Kinect Mod) Setup

PrimeSense

Choose Components

Choose which features of PrimeSensor 5.0.0 for Windows (Kinect Mod) you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

- [x] OpenNI
- [x] Sensor

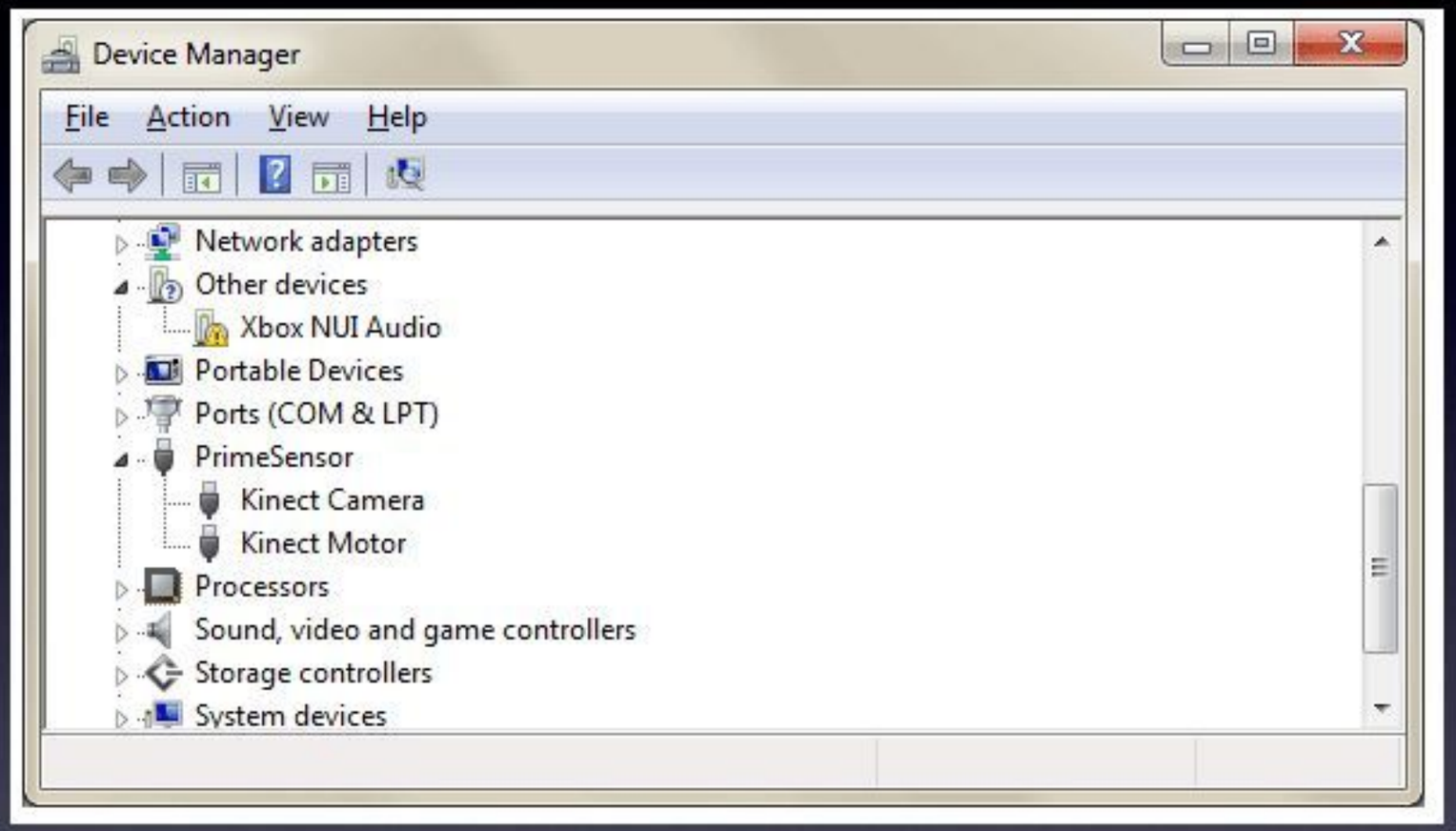
Description

Position your mouse over a component to see its description.

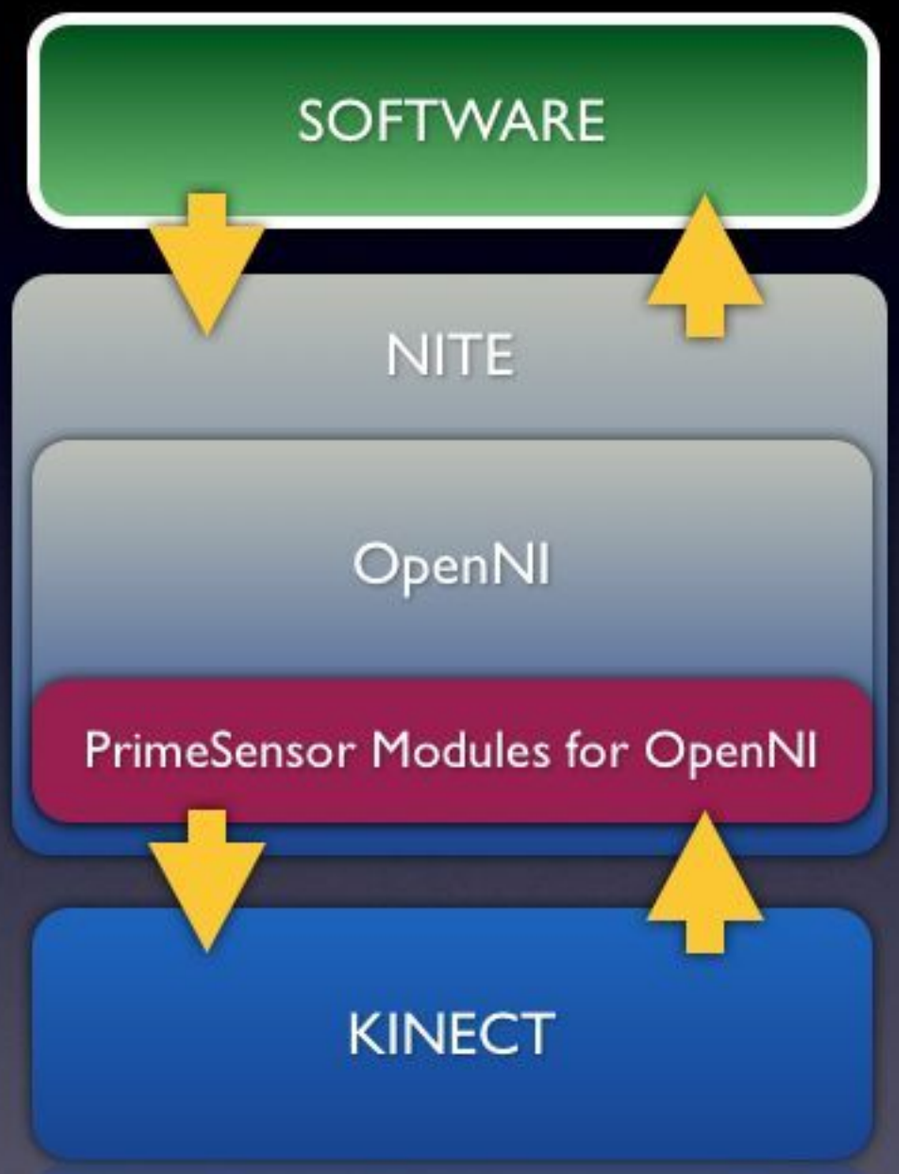
Space required: 129.7MB

Nullsoft Install System v2.46

Next > | Cancel


Device Manager
File Action View Help
Network adapters
Other devices
Xbox NUI Audio
Portable Devices
Ports (COM & LPT)
PrimeSensor
Kinect Camera
Kinect Motor
Processors
Sound, video and game controllers
Storage controllers
System devices

Kinect: controllo remoto



Download: OpenNI (testing)

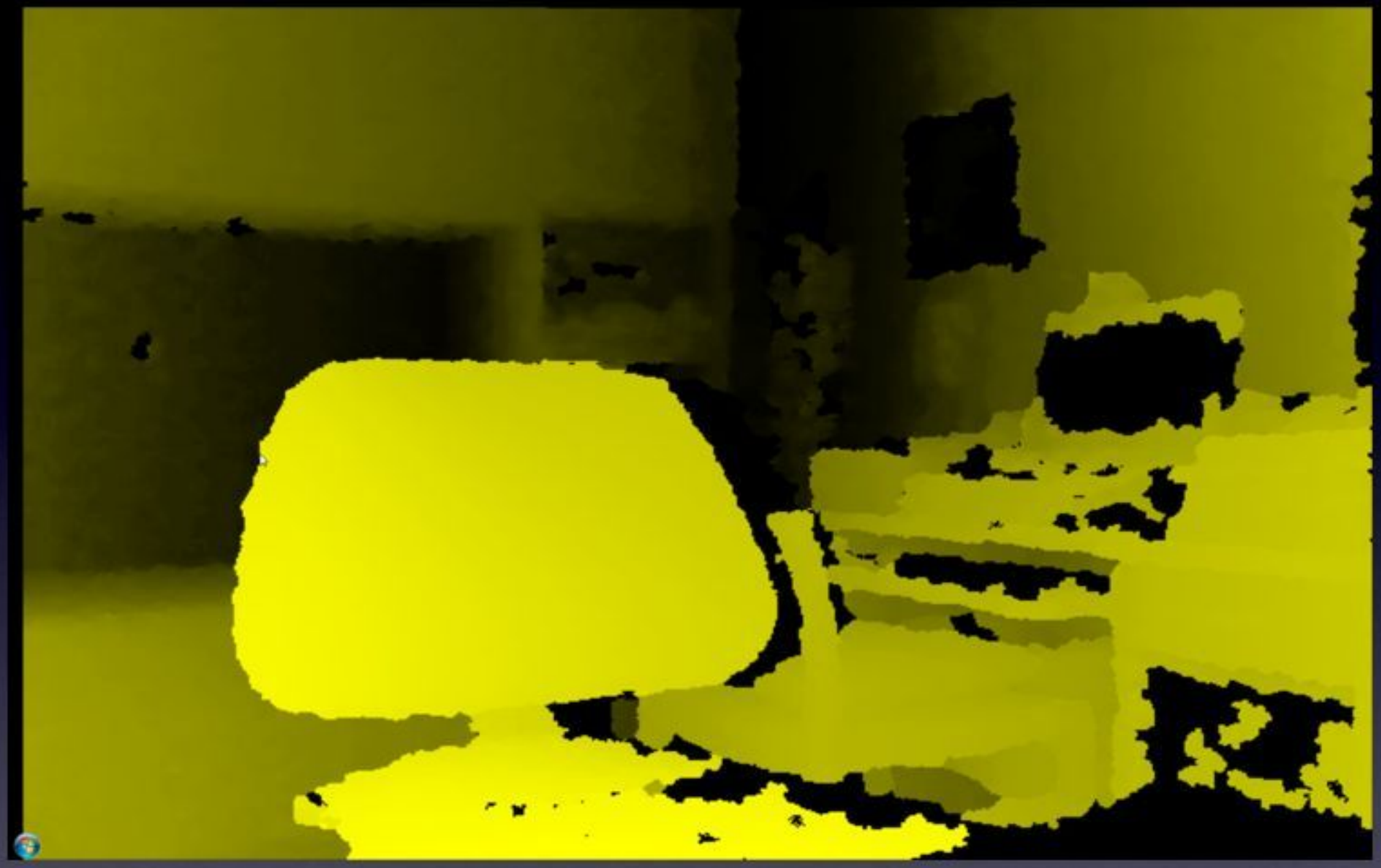
Download: Drivers

https://github.com/OpenNI/OpenNI

https://github.com/OpenNI/OpenNI

C:\Program Files (x86)\OpenNI\Samples\Bin\Release:

- C:\Program Files (x86)\OpenNI\Samples\Bin\Release: nisimpleviewer.exe



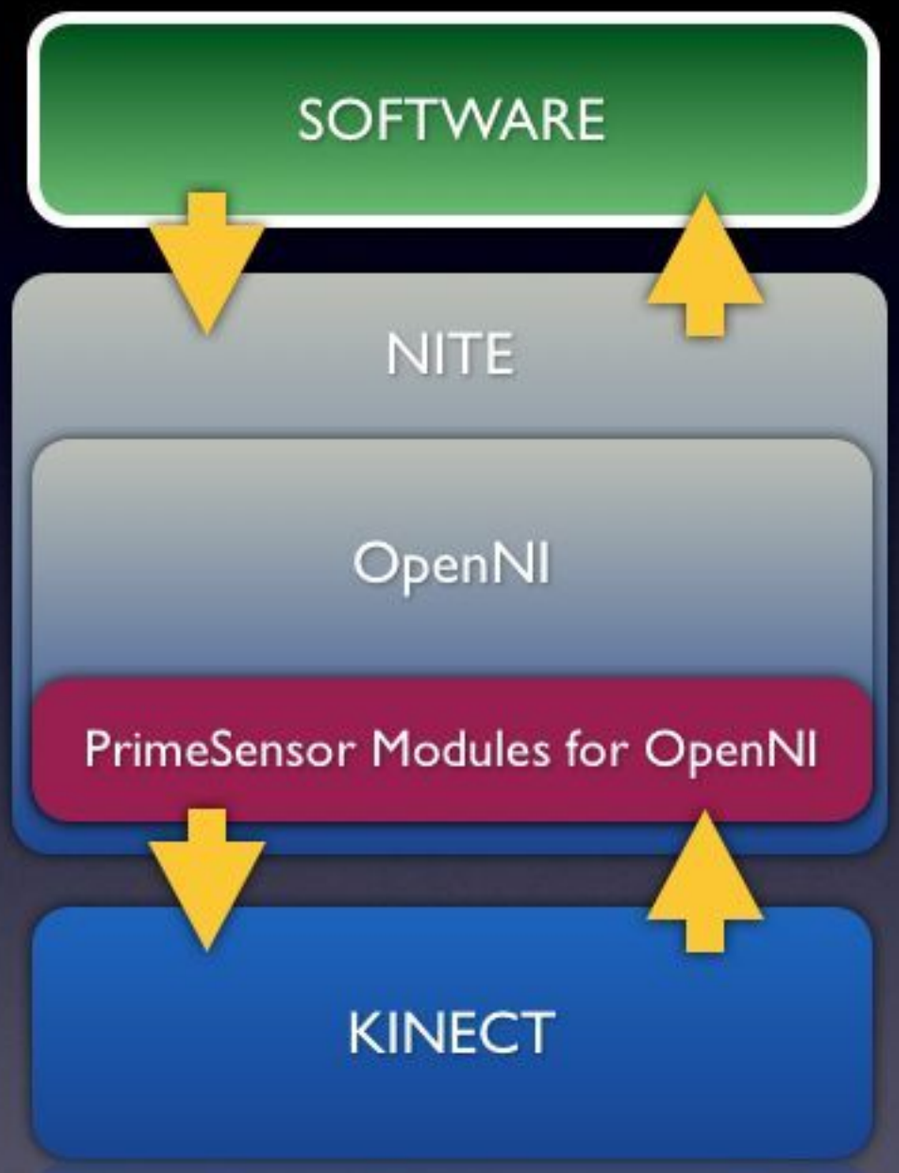
C:\Program Files (x86)\OpenNI\Samples\Bin\Release:

- nisimpleviewer.exe
- niusertracker.exe


3 – Tracking


3 - Tracking

Kinect: controllo remoto



Download: NITE

Download: OpenNI (testing)

Download: Drivers

http://www.openni.org/downloadfiles/2-openni-binaries

- C:\Program Files (x86)\Prime Sense\NITE\Wrappers\C#\Bin
  ManagedNite.dll

New Project

Recent Templates

Installed Templates

- Visual C#
  - Windows
  - Web
  - Office
  - Cloud
  - Reporting
  - SharePoint
  - Silverlight
  - Silverlight for Windows Phone
  - Test
  - WCF
  - Workflow
  - XNA Game Studio 4.0
- Other Languages
- Other Project Types
- Database
- Modeling Projects
- Test Projects

Online Templates

.NET Framework 4 | Sort by: Default | Search Installed Templates

Windows Forms Application	Visual C#
WPF Application	Visual C#
Console Application	Visual C#
ASP.NET Web Application	Visual C#
Class Library	Visual C#
ASP.NET MVC 2 Web Application	Visual C#
Silverlight Application	Visual C#
Silverlight Class Library	Visual C#
WCF Service Application	Visual C#
ASP.NET Dynamic Data Entities Web Application	Visual C#
Enable Windows Azure Tools	Visual C#

Type: Visual C#

Windows Presentation Foundation client application

Name: WpfApplication1

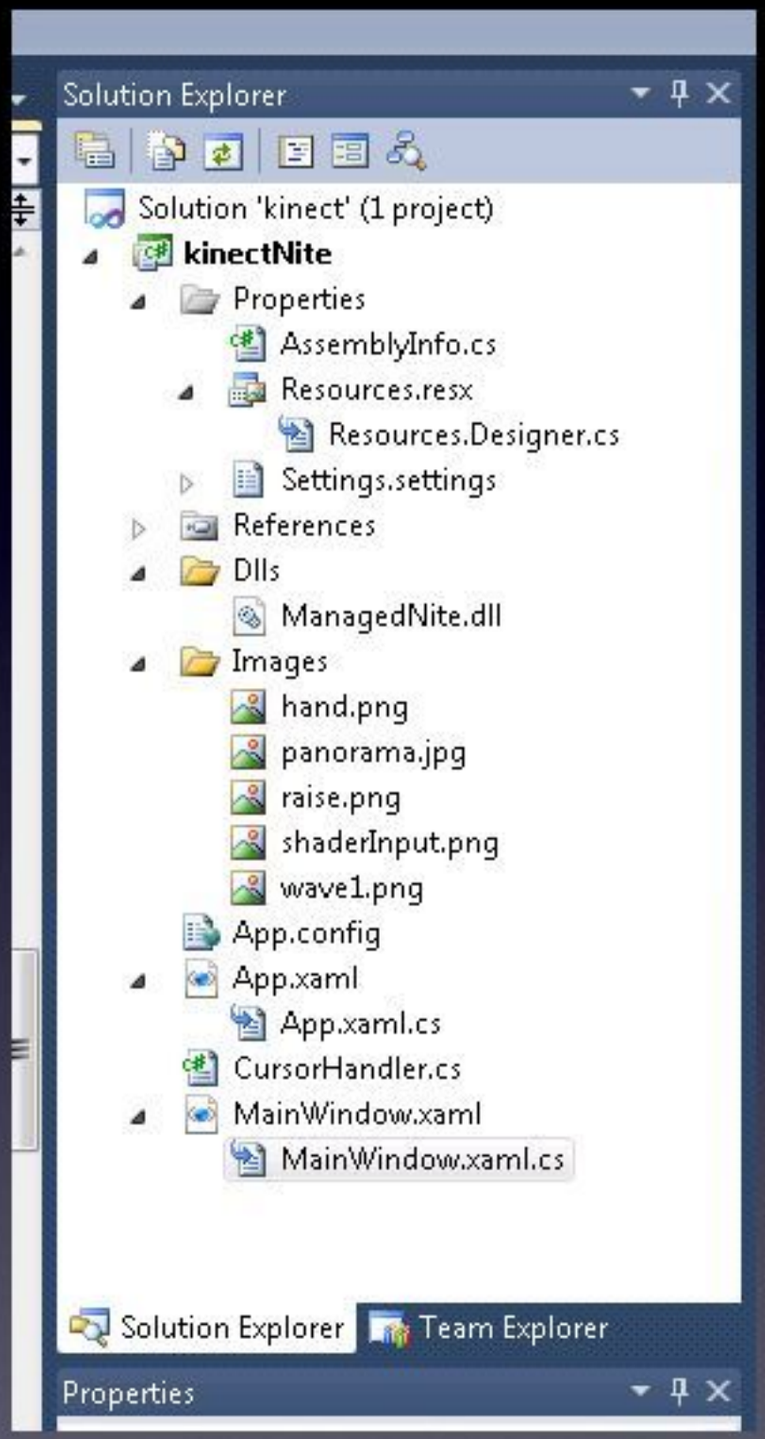
Location: c:\users\neogene\documents\visual studio 2010\Projects — Browse...

Solution: Create new solution

Solution name: WpfApplication1

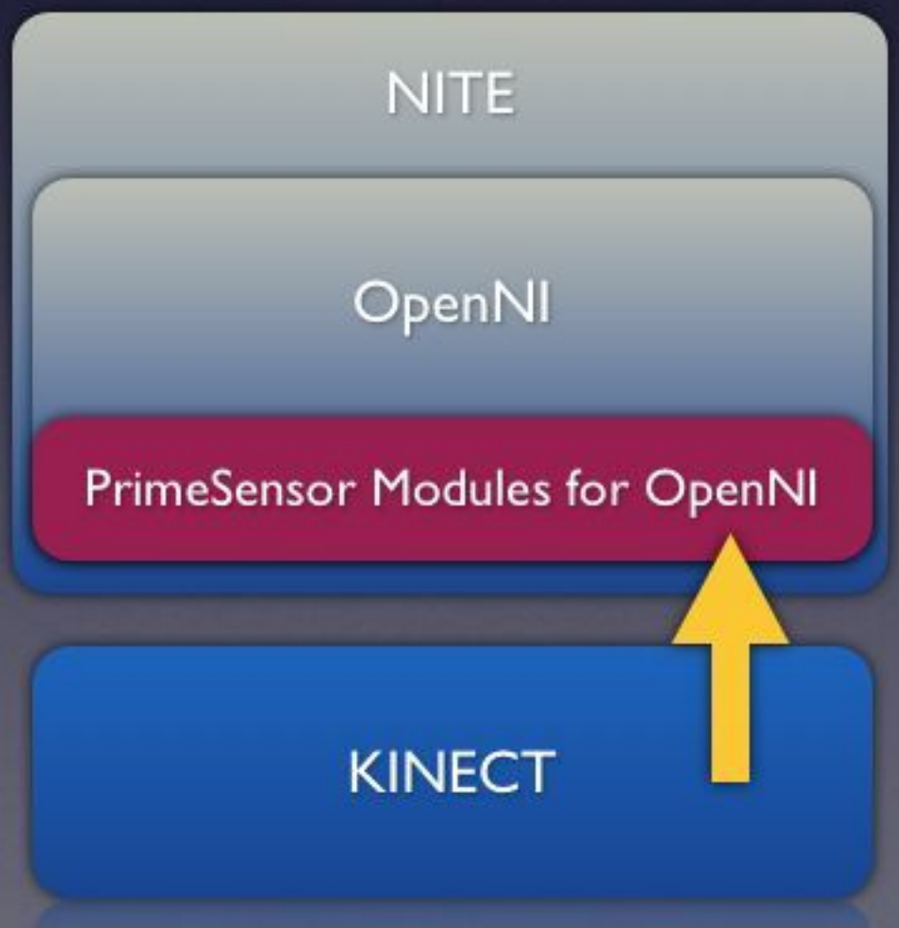
Create directory for solution

OK Cancel


Solution Explorer
Solution 'kinect' (1 project)
kinectNite
Properties
AssemblyInfo.cs
Resources.resx
Resources.Designer.cs
Settings.settings
References
Dlls
ManagedNite.dll
Images
hand.png
panorama.jpg
raise.png
shaderInput.png
wave1.png
App.config
App.xaml
App.xaml.cs
CursorHandler.cs
MainWindow.xaml
MainWindow.xaml.cs
Solution Explorer
Team Explorer
Properties

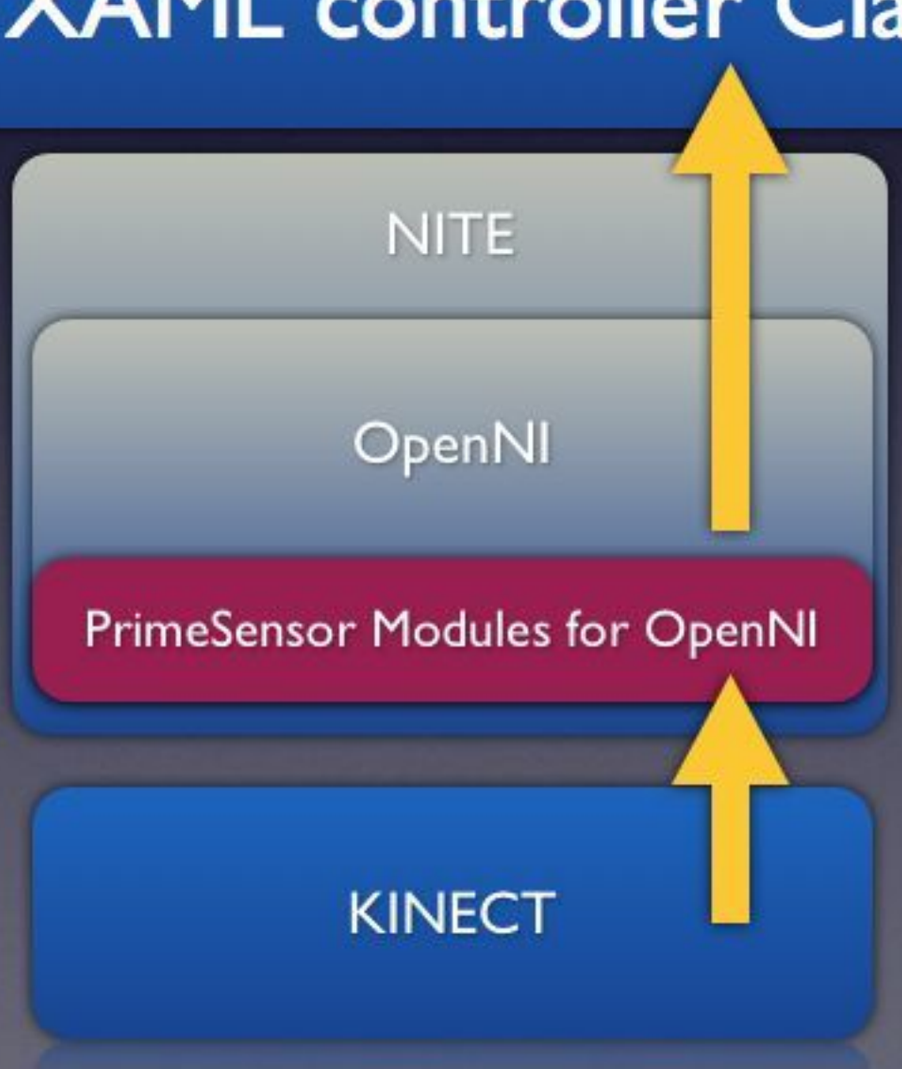

MainWindow
Photo by Andrea Leganza

KINECT


NITE
OpenNI
PrimeSensor Modules for OpenNI
KINECT

C#

XAML controller Class

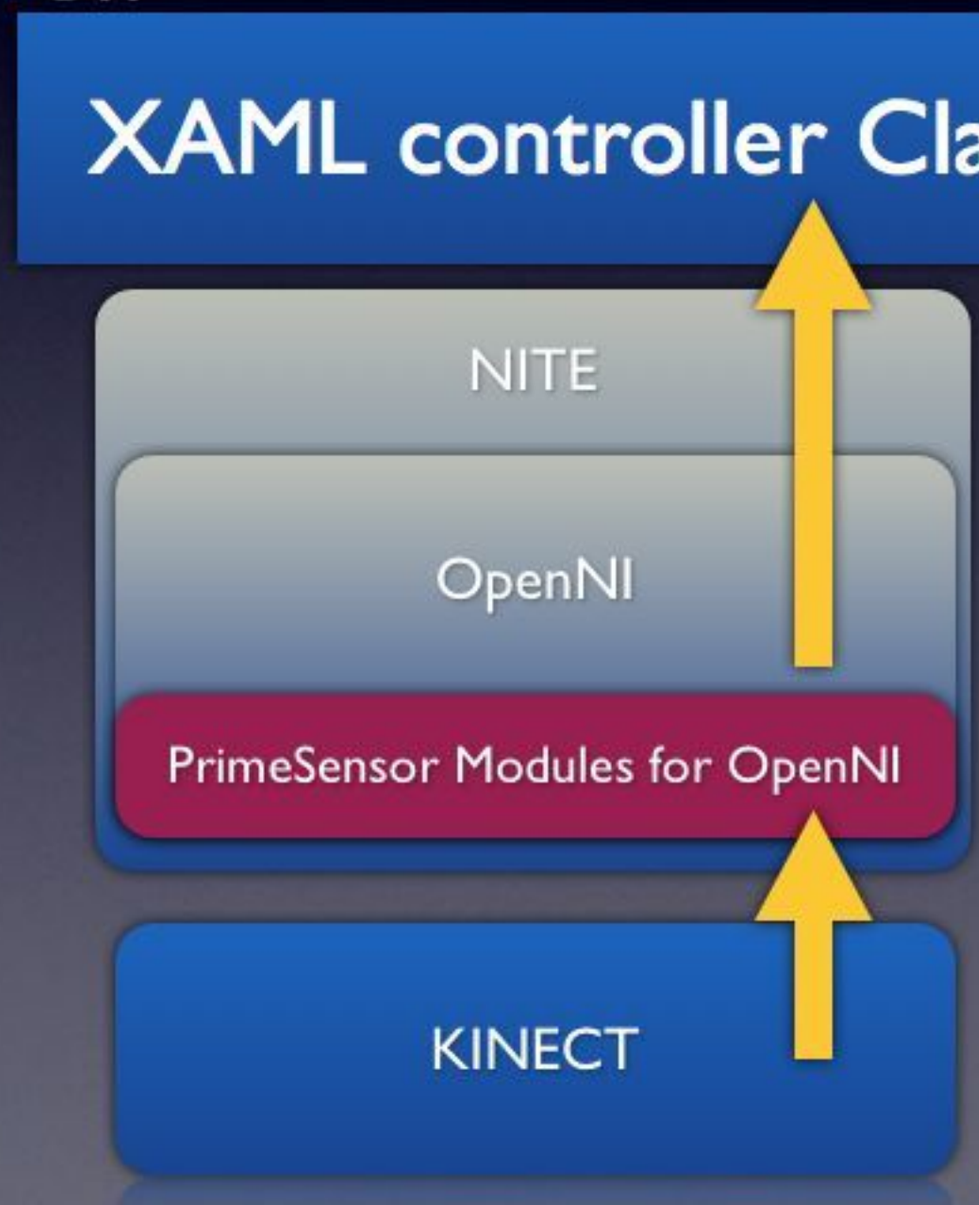


C#

XAML controller Class

C#

Hand controller Class



XML

INTERFACCIA GRAFICA:
XAML

C#

XAML controller Class

C#

Hand controller Class

NITE

OpenNI

PrimeSensor Modules for OpenNI

KINECT

INTERFACCIA GRAFICA: XAML

```
<Window
      x:Class="kinectNite.MainWindow"
       xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
       xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
       xmlns:local="clr-namespace:kinectNite"
       xmlns:ee="http://schemas.microsoft.com/expression/2010/effects"
       Title="MainWindow"
       x:Name="main"
       WindowState="Maximized">

 <Window.Resources>
       <BooleanToVisibilityConverter x:Key="cnvBoolToVisibility"/>
   </Window.Resources>
   <Grid >

<Image x:Name="imgImage"   Stretch="UniformToFill" Source="Images/
panorama.jpg">
 <Image.Effect >
  <ee:RippleEffect x:Name="myRipple" Frequency="0" Magnitude="0"   Phase="0"/
>
   </Image.Effect>
 </Image>

 <StackPanel
HorizontalAlignment="Center"
VerticalAlignment="Top"
Margin="0,50,0,0"
Visibility="{Binding ElementName=main, Path=HandDetected, Converter=
{StaticResource cnvBoolToVisibility} }">

 <TextBlock Text="Effettua un Wave per iniziare il tracking..."
HorizontalAlignment="Center"
FontSize="20"
FontFamily="Segoe UI Light"
Foreground="White">
</TextBlock>
<Image Source="Images/wave1.png" Width="90" Margin="10" ></Image>
</StackPanel>
<StackPanel HorizontalAlignment="Center"
VerticalAlignment="Top"
Margin="0,50,0,0"
Visibility="{Binding ElementName=main, Path=ReFocus, Converter={StaticResource
cnvBoolToVisibility} }">
<TextBlock Text="Solleva la mano per riprendere il tracking..."
HorizontalAlignment="Center"
FontSize="20"
 FontFamily="Segoe UI Light" Foreground="White">
</TextBlock>
<Image Source="Images/raise.png" Width="90" Margin="10" ></Image>
</StackPanel>

 <ListBox x:Name="lstHands"
Background="Transparent"
Grid.Column="0"
HorizontalAlignment="Stretch"
VerticalAlignment="Stretch"
ItemsSource="{Binding ElementName=main,Path=AvailableHands}">
          <ListBox.ItemTemplate>
            <DataTemplate>
              <Canvas Background="Transparent">
                <Image Canvas.Top="{Binding Top}" Canvas.Left="{Binding Left}"
Source="Images/hand.png" Width="80"  ></Image>
              </Canvas>
            </DataTemplate>
          </ListBox.ItemTemplate>
          <ListBox.ItemsPanel>
            <ItemsPanelTemplate>
              <Canvas Background="Transparent">
              </Canvas>
            </ItemsPanelTemplate>
          </ListBox.ItemsPanel>
        </ListBox>

        </Grid>
    </Window>
```

CONTROLLER:
C#

```
...
using ManagedNite;
...

namespace kinectNite
{

    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow :Window
    {
        XnMSessionManager sessionManager;
        XnMOpenNIContext context;
        XnMPushDetector pushpoint;
        XnMPointDenoiser pointdenoise;
        private int count = 1;

        #region Properties
        public ObservableCollection<Hand> AvailableHands{
            get { return (ObservableCollection<Hand>)GetValue
(AvailableHandsProperty); }
            set { SetValue(AvailableHandsProperty, value); }
        }

        // Adoperiamo una DependencyProperty per gestire la variabile
AvailableHands. In questo modo possiamo adoperare il binding nel file xaml (è
adoperata per visualizzare l'icona con la mano il cui spostamento dipende dai nostri
movimenti).
        public static readonly DependencyProperty AvailableHandsProperty =
DependencyProperty.Register("AvailableHands", typeof
(ObservableCollection<Hand>), typeof(MainWindow), new UIPropertyMetadata
(null));

        public bool HandDetected {
            get { return (bool)GetValue(HandDetectedProperty); }
            set { SetValue(HandDetectedProperty, value); }
        }

        // Adoperiamo una DependencyProperty per gestire la variabile
HandDetected. In questo modo possiamo adoperare il binding nel file xaml
        public static readonly DependencyProperty HandDetectedProperty =
DependencyProperty.Register("HandDetected", typeof(bool), typeof(MainWindow),
new UIPropertyMetadata(false));

        public bool ReFocus {
            get { return (bool)GetValue(ReFocusProperty); }
            set { SetValue(ReFocusProperty, value); }
        }

        // Adoperiamo una DependencyProperty per gestire la variabile ReFocus. In
questo modo possiamo adoperare il binding nel file xaml
        public static readonly DependencyProperty ReFocusProperty =
DependencyProperty.Register("ReFocus", typeof(bool), typeof(MainWindow), new
UIPropertyMetadata(false));

        private bool shouldRun = false;
        private Thread readerThread;
        private Point oldpos = new Point(0, 0);
        #endregion
```

```
//Costruttore
        public MainWindow() {
            InitializeComponent();

            //Ascoltiamo la richiesta di chiusura dell'applicazione
            this.Closing += new System.ComponentModel.CancelEventHandler
(MainWindow_Closing);

            //Ascoltiamo la pressione dei tasti della tastiera
            this.KeyDown += new KeyEventHandler(MainWindow_KeyDown);

            //Inizializziamo  la struttura dati che conterrà le mani rilevate
            AvailableHands = new ObservableCollection<Hand>();

            //Creiamo il gestore del contesto
            context = new XnMOpenNIContext();

            this.shouldRun=true;
            HandDetected = true;

            //Inizializziamo il contesto
            context.Init();

            //Inizializziamo il detector del movimento di push
            pushpoint = new XnMPushDetector();

            //Ascoltiamo questo tipo di eventi
            pushpoint.Push += new EventHandler<PushDetectorEventArgs>
(pushpoint_Push);

            //Inizializziamo il filtro
            /* Questo oggetto riduce la variazione del movimento dei punti
consentendo
             * di ridurre il tremolio dovuto a normali microspostamenti della mano nei
diversi assi.
             */
            pointdenoise = new XnMPointDenoiser();

            //Ascoltiamo la rilevazione del punto associato alla mano, il suo cambio di
stato, e la sua scomparsa dall'area visibile della telecamera
            pointdenoise.PrimaryPointCreate += new
EventHandler<PrimaryPointCreateEventArgs>(detectPoint_PrimaryPointCreate);
            pointdenoise.PrimaryPointUpdate += new
EventHandler<HandPointContextEventArgs>(detectPoint_PrimaryPointUpdate);
            pointdenoise.PrimaryPointDestroy += new
EventHandler<PointDestroyEventArgs>(detectPoint_PrimaryPointDestroy);

            //Gestione della Sessione, questa viene avviata in base al movimento di
wave, il "saluto"
            sessionManager = new XnMSessionManager(context, "Wave", "RaiseHand");
            sessionManager.AddListener(pointdenoise);
            sessionManager.AddListener(pushpoint);

            sessionManager.FocusStartDetected += new
EventHandler<FocusStartEventArgs>(sessionManager_FocusStartDetected);
            sessionManager.SessionStarted      += new EventHandler<PointEventArgs>
(sessionManager_SessionStarted);
            sessionManager.SessionEnded       += new EventHandler
(sessionManager_SessionEnded);

            //Avviamo un thread che forzi l'aggiornamento del contesto e del session
manager.
            this.readerThread = new Thread(ReaderThread);
            this.readerThread.Start();
        }

        //Richiede in continuazione l'aggiornamento dei vari monitor (è un ciclo
infinito)
        private void ReaderThread(){
            while (this.shouldRun) {
                uint rc = context.Update();
                if (rc == 0) sessionManager.Update(context);
            }
        }
```

```
#region Window Events
        void MainWindow_KeyDown(object sender, KeyEventArgs e){
            if (e.Key == Key.M) {
                //Se l'utente preme il tasto M allora avviene uno switch tra finestra
massimizzata/minimizzata.
                //Passiamo da uno stato all'altro di massimizzazione/minimizzazione della
finestra della nostra applicazione
                if (this.WindowState == WindowState.Maximized)
                    this.WindowState = System.Windows.WindowState.Minimized;
                else
                    this.WindowState = System.Windows.WindowState.Maximized;
            }
        }

        //In caso di richiesta di chiusura impostiamo shouldRun a false in modo che il
thread di aggiornamento esca dal ciclo infinito
        void MainWindow_Closing(object sender,
System.ComponentModel.CancelEventArgs e){
            this.shouldRun = false;

            //Richiediamo il termine delle operazioni di monitoring
            this.context.Close();
            this.context.Dispose();

            //Lasciamo che l'evento di chiusura prosegua in alto nella catena di
ereditarietà
            base.OnClosing(e);
        }
        #endregion

        #region Push
        //Invocato quando NITE identifica il movimento di push
        void pushpoint_Push(object sender, PushDetectorEventArgs e){
            Dispatcher.Invoke( System.Windows.Threading.DispatcherPriority.Normal,
new Action(
                delegate(){
                    //Richiediamo che il puntatore effettui un click
                    CursorHandler.SendMouseLeftClick(oldpos);
                }));
        }
        #endregion

        #region Hands
        //Invocato quando viene rilevata una nuova mano
        void detectPoint_PrimaryPointCreate(object sender,
PrimaryPointCreateEventArgs e){

            //Otteniamo le coordinate associate al punto che identifica la mano
            float x = e.HPC.Position.X;
            float y = e.HPC.Position.Y;

            //ID associato alla mano/punto che la rappresenta
            int id = e.HPC.ID;

            Dispatcher.Invoke(System.Windows.Threading.DispatcherPriority.Normal,
new Action(
                delegate()
                {
                    ReFocus = false;

                    //Creiamo una nuova istanza che monitorizza la posizione della
mano, registriamo ID associato al punto, e posizioni
                    Hand newHand = new Hand {
                        ID = id,
                        Left = (((lstHands.ActualWidth) / 2) - 30) + x,
                        Top = (((lstHands.ActualHeight) / 2) - 30) + (y * -1)
                    };

                    //Aggiungiamo la mano appena rilevata all'elenco di mani rilevate
                    AvailableHands.Add(newHand);
                    count++;
                }));
        }
```

```
    }

        //Invocato quando viene rilevata una variazione della posizione del punto
associato alla mano
        void detectPoint_PrimaryPointUpdate(object sender,
HandPointContextEventArgs e) {

            //Otteniamo le coordinate associate al punto che identifica la mano
            float x = e.HPC.Position.X;
            float y = e.HPC.Position.Y;

            var cursorX = 0;
            var cursorY = 0;

            Dispatcher.Invoke
(System.Windows.Threading.DispatcherPriority.Normal,new Action(
                    delegate(){
                        //Cerchiamo la mano precedentemente inserita che abbia identico
ID di quello che ha causato l'invocazione del metodo
                        Hand currHand = AvailableHands.First(h => h.ID == e.HPC.ID);
                        //Aggiorniamo le sue posizioni
                        currHand.Left = (((lstHands.ActualWidth) / 2) - 30) + x;
                        currHand.Top = (((lstHands.ActualHeight) / 2) - 30) + (y * -1);

                        //Registriamo la nuova posizione della mano relativamente allo
schermo.
                        cursorX = (int)
((System.Windows.SystemParameters.PrimaryScreenWidth / 2) + x);
                        cursorY = (int)
((System.Windows.SystemParameters.PrimaryScreenHeight / 2) + (y * -1));
                    }

                ));

            //Spostiamo il puntatore del mouse nella nuova posizione
            CursorHandler.SetCursorPos(cursorX, cursorY);
        }

        //Invocato quando la mano esce dal campo visivo della telecamera
        void detectPoint_PrimaryPointDestroy(object sender, PointDestroyEventArgs
e)
        {
            //Otteniamo l'ID del punto/mano
            int id = (int)e.ID;

            Dispatcher.Invoke(System.Windows.Threading.DispatcherPriority.Normal,
new Action(
                    delegate() {
                        if (AvailableHands.Count > 0){
                            //Abbiamo  almeno una mano nel nostro elenco, nascondiamo la
finestra di richiesta di movimento della mano
                            HandDetected = false;
                            //Richiediamo che l'utente effettui nuovamente un movimento di
wave e mostriamo il relativo box con il testo esplicativo
                            ReFocus = true;

                            //Otteniamo la mano che è appena scomparsa adoperando l'ID...
                            Hand currHand = AvailableHands.FirstOrDefault(h => h.ID == id);

                            //e la rimuoviamo dal nostro elenco poichè non è più disponibile
                            AvailableHands.Remove(currHand);
                        }
                    }));
        }

        #endregion
```

```
  #region Sessione
        void sessionManager_FocusStartDetected(object
sender, FocusStartEventArgs e) {
            System.Console.WriteLine("Focus Session
Started");
        }

        void sessionManager_SessionStarted(object
sender, PointEventArgs e) {
            Dispatcher.Invoke
(System.Windows.Threading.DispatcherPriority.Normal,
new Action(
                    delegate(){
                        //E' appena iniziata la sessione, nessuna
mano è stata ancora identificata
                        HandDetected = false; //Nascondiamo il
box contenente il testo di richiesta di muovere la mano
                    }));

            System.Console.WriteLine("Session Started");
        }

        void sessionManager_SessionEnded(object sender,
EventArgs e) {
            Dispatcher.Invoke
(System.Windows.Threading.DispatcherPriority.Normal,
new Action(
                    delegate() {
                        HandDetected = true; //Visualizziamo il
box contentente il testo di richiesta di muovere la mano
                        ReFocus = false; //Nascondiamo quella di
richiesta di ri-muovere la mano per continuare il
tracking
                    }));

            System.Console.WriteLine("Session Ended");
        }
        #endregion
    }
}
```

Hand managemente class (uses InteropServices)

```
namespace kinectNite
{
    internal class CursorHandler
    {
        #region Constants

        private const UInt32 MouseEventfLeftDown = 0x0002;
        private const UInt32 MouseEventfLeftUp = 0x0004;

        //Fattori di conversione tra coordinate
        private const double ScreenXConv = 64.0; //65535 / 1024
        private const double ScreenYConv = 85.3; //65535 / 768
#endregion
#region DllImports

[DllImport("user32.dll")]
private static extern void mouse_event(UInt32 dwFlags, UInt32 dx, UInt32 dy, UInt32 dwData, IntPtr dwExtraInfo);
[DllImport("user32.dll")]
public static extern IntPtr GetDesktopWindow();
[DllImport("user32.dll")]
public static extern IntPtr GetWindowDC(IntPtr hWnd);
[DllImport("user32.dll")]
public static extern IntPtr ReleaseDC(IntPtr hWnd, IntPtr hDC);
[DllImport("user32.dll")]
public static extern void mouse_event(uint dwFlags, long dx,long dy, uint dwData, IntPtr dwExtraInfo);
[DllImport("user32.dll")]
public static extern void keybd_event(byte bVk, byte bScan,uint dwFlags, IntPtr dwExtraInfo);
[DllImport("user32.dll", SetLastError = true)
```

Hand managemente class (uses InteropServices)

```
[return: MarshalAs(UnmanagedType.Bool)]
public static extern bool SetCursorPos(int x, int y);

#endregion

/// Click del pulsante sinistro del mouse
public static void SendMouseLeftClick(Point location)
{
    MoveMouseTo((int)location.X, (int)location.Y);
    mouse_event(MouseEventfLeftDown, 0, 0, 0, new IntPtr());
    mouse_event(MouseEventfLeftUp, 0, 0, 0, new IntPtr());
}

/// Spostamento del mouse nella posizione richiesta
public static void MoveMouseTo(int x, int y)
{
    x = (int)(x * ScreenXConv);
    y = (int)(y * ScreenYConv);
    SetCursorPos(x, y);
}
  }
}
```

Kinect: risorse web

- http://kinectforwindows.org
- http://kinectforwindows.org/features//
- http://www.xbox.com/en-US/Kinect/Kinect-Effect
- http://openkinect.org
- http://kinect.dashhacks.com/
- http://channel9.msdn.com/series/KinectSDKQuickstarts/
- http://www.microsoft.com/robotics/
- http://research.microsoft.com/apps/pubs/default.aspx?id=145347