Escola Universitària
Politècnica de Mataró

Centre adscrit a:

UNIVERSITAT POLITÈCNICA
DE CATALUNYA

Enginyeria Tècnica en Informàtica de Gestió

REHABILITACIÓ AMB KINECT

Memòria

XAVIER SÀNCHEZ LÓPEZ
PONENT: MONTSERRAT RABASSA

PRIMAVERA 2011

TecnoCampus
Mataró-Maresme

Dedicatòria

A la meva filla Clàudia.

Agraïments

Als meus professors de l'EUPMT,
en especial a la meva ponent Montse Rabassa i a la meva tutora Lina Juan.

Als meus companys de l'EUPMT,
en especial a Eduard Garcia, Núria Silva i Pedro Martinez.

A Manel Domingo del CCI-TCM.

A Maurici Chies de Hewlett Packard.

A Susana Ruiz.

Resum

La medicina aprofita les noves tecnologies per desenvolupar millores en la diagnosi, tractament i prevenció de certes malalties. Amb els sistemes informatitzats actuals i els nous dispositius que sorgeixen, com Kinect, Rehabilitació amb Kinect dona suport en les teràpies de rehabilitació per traumatisme.

L'aplicació està desenvolupada amb el llenguatge de programació C#, amb el processador d'imatges EmguCV i amb el driver de Kinect CL NUI Platform per crear una nova plataforma de tele-rehabilitació.

Resumen

La medicina aprovecha las nuevas tecnologías para desarrollar mejoras en el diagnostico, tratamiento y prevención de ciertas enfermedades. Con los sistemas informatizados actuales y los nuevos dispositivos que surgen, como Kinect, *Rehabilitació amb Kinect* da soporte en las terapias de rehabilitación por traumatismo.

La aplicación esta desarrollada con el lenguaje de programación C#, con el procesador de imágenes EmguCV y con el driver de Kinect CL NUI Platform para crear una nueva plataforma de tele rehabilitación.

Abstract

The medicine takes advantage of new technologies to develop improved diagnosis, treatment and prevention of certain diseases. With current computer systems and new devices that come as Kinect, *Rehabilitació amb Kinect* gives support in trauma rehabilitation therapies.

The application is developed with C# programming language, with EmguCV image processor and CL NUI Platform Kinect driver to create a new tele rehabilitation platform.

Índex.

Índex de figures.

Índex de taules.

Glossari de termes.

RaK	Rehabilitació amb Kinect
RGB	Descomposició d'una imatge en tres colors; vermell, verd i blau *(red, green, blue)*.
TFC	Treball final de carrera.
Xbox360	Videoconsola de Microsoft.
SO	Sistema Operatiu.
Píxel	Menor unitat de què es composa una imatge digital. Conté informació sobre la tonalitat del color.
Thresholding	Mètode de processament de la imatge. Converteix una imatge amb color real en una imatge en escala de grisos o una imatge binaria (blanc i negre).
CIP	Codi únic per a cada assegurat de la sanitat pública.
VS	Visual Studio.
Namespace	Espai de noms de C# equivalent a un package en Java.
PC	Ordinador personal *(personal computer)*
IDE	Entorn de desenvolupament integrat *(integrated development environment)*
SQL	Llenguatge de consulta estructurat *(structured query language)*

1. Introducció.

Les tècniques utilitzades en la rehabilitació de l'aparell locomotor del cos humà consisteixen en el moviment repetitiu de les articulacions per tornar a assolir la màxima mobilitat després d'un trauma o d'una malaltia.

Els sistemes informatitzats són capaços d'obtenir dades concretes d'una imatge, com per exemple, l'angle que forma una articulació del cos humà. A més, sorgeixen nous dispositius que doten als sistemes informatitzats de més informació sobre el món real.

En el projecte desenvolupat anomenat Rehabilitació amb Kinect, a partir d'ara mencionat RaK, utilitzant aquestes noves tecnologies, s'obtenen dades de la mobilitat de les articulacions en una sessió de rehabilitació. Aquestes dades poden ser avaluades objectivament per conèixer quina és l'evolució real d'un pacient.

2. Objectius.

2.1. Propòsit.

Desenvolupar una eina de tele-rehabilitació dins del sistema sanitari català ideada i preconcebuda pel CCI-TCM (Centre de competències d'integració – Tecnocampus de Mataró) dins del marc de treball d'investigació i recerca de noves tecnologies.

Aprofitar els recursos que ofereixen les noves tecnologies per aplicar-les en la ciència de la medicina. En aquest cas Kinect en el camp de la rehabilitació.

Obtenir dades precises del grau de mobilitat de les articulacions d'un pacient.

2.2. Finalitat.

Donar recolzament en el procés de rehabilitació d'un pacient afectat d'una disminució de mobilitat d'alguna articulació, a través de la gestió de les dades que s'obtenen a temps real durant una sessió de rehabilitació.

2.3. Objecte.

Aplicació de sobretaula per Microsoft Windows amb els drivers necessaris per Kinect.

2.4. Abast.

Realitzar sessions de rehabilitació del moviment d'abducció i adducció del canell de la mà dreta amb Kinect o amb una càmera web per recollir dades de mobilitat articular i així poder avaluar l'evolució de la rehabilitació.

3. Estudi previ.

Com a part prèvia al desenvolupament del programari, es presenta un estudi sobre:

- Rehabilitació: Quins tipus de rehabilitació existeixen i amb quins d'aquests es pot treballar amb RaK?
- Tele-rehabilitació:Quines plataformes especialitzades en la tele-rehabilitació hi ha disponibles en el mercat?
- Kinect: Què és Kinect? Quins són els seus components? Quins són els avantatges que ofereix respecte altres dispositius?
- Kit de desenvolupament: Quines tecnologies són necessàries per poder desenvolupar una aplicació basada en Kinect?

3.1. Rehabilitació.

El concepte de rehabilitació fa referència al conjunt de mètodes que tenen per finalitat la recuperació d'una activitat o d'una funció perduda o disminuïda a causa d'un traumatisme o una malaltia en una persona. La rehabilitació finalitza quan es produeix la completa recuperació física, psicològica i laboral.

3.1.1. Tipus de rehabilitació.

Els tipus de rehabilitació es divideixen en tres grups:

- La rehabilitació en reumatologia encarregada d'ajudar en els problemes d'insuficiència respiratòria.

- La rehabilitació neurològica en caos d'hemiplegies, paraplegies, lesions medul·lars, malalties degeneratives...

- La rehabilitació en traumatologia en casos de fractures, traumatismes o osteopaties.

Amb RaK es poden analitzar els moviments articulars que realitza una persona. Per tant, aquesta aplicació pot donar suport en aquells tipus de rehabilitació on s'apliquin tècniques basades en el moviment de les articulacions com a teràpia de cura. El tipus de rehabilitació indicat és la rehabilitació d'una traumatologia.

3.1.2. Les sessions de rehabilitació en traumatologia.

Una de les tècniques que s'utilitza en les sessions de rehabilitació d'una articulació afectada per un traumatisme consisteix a realitzar un moviment de manera repetitiva per tal de retornar el grau de mobilitat perdut.

En les sessions de rehabilitació es realitza el mateix moviment de l'articulació durant un temps establert que pot durar des d'uns segons a minuts. Aquestes sessions es repeteixen durant setmanes i, fins i tot, mesos. Durant aquest procés, l'articulació ha d'anar augmentant el grau de mobilitat.

Per realitzar un anàlisi del progrés de rehabilitació del pacient, el personal mèdic disposa d'eines per mesurar l'evolució de l'articulació afectada, com per exemple, un transportador d'angles, tal i com es mostra a la fig. 3.1.

La tècnica de rehabilitació de RaK és exactament igual a la utilitzada pels professionals. El pacient ha de realitzar el moviment que l'aplicació li indica durant un cert temps.

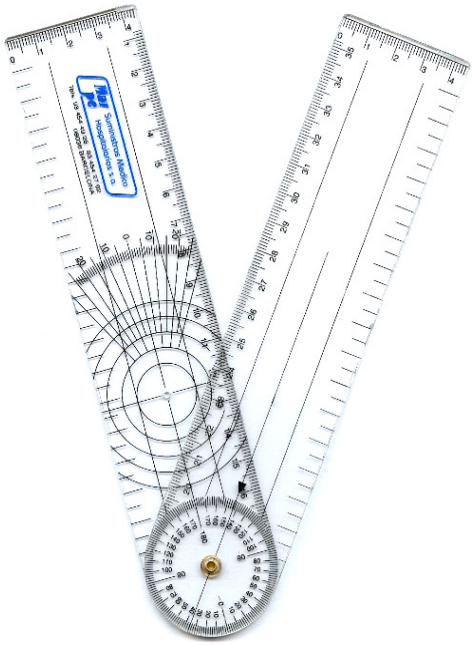

Fig. 3.1. Transportador d'angles.

3.1.3. Moviment de les articulacions.

En la rehabilitació del moviment del sistema de l'esquelet, algunes articulacions tenen més d'un moviment específic. Ossos, músculs i tendons defineixen quants i quins moviments es poden realitzar.

En primer lloc, el genoll, per exemple, només pot realitzar un tipus de moviment, flexió i extensió, en canvi d'altres extensions com el canell poden realitzar més d'un moviment, flexió i extensió, abducció i adducció i circumducció.

D'altra banda, cada moviment de l'articulació descriu una trajectòria diferent depenen de si es tracta d'una extremitat de la part dreta o de la part esquerra del cos, és a dir, el moviment d'adducció es produeix en diferent sentit pel canell de la mà dreta respecte al de l'esquerra. Degut a això, es necessari implementar un algorisme específic per a cada articulació en concret. Amb RaK es treballa el moviment d'abducció i adducció del canell de la mà dreta, tal i com es veu a la fig. 3.2.

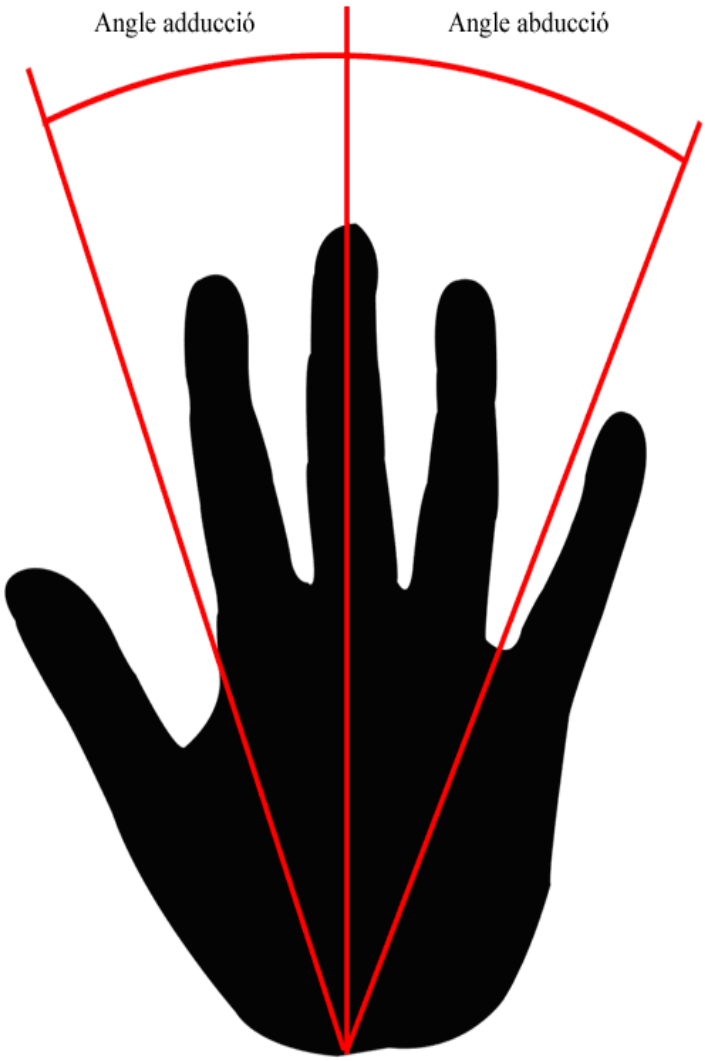

Fig. 3.2. Moviment d'abducció i adducció del canell.

3.2. Tele-rehabilitació.

En el mercat existeixen varies solucions en el camp de la tele-rehabilitació. Cadascuna d'aquestes plataformes utilitzen diferents dispositius i treballen diferents tipus de rehabilitació.

Algunes de les més importants es descriuen a continuació.

3.2.1. Play for Health.

Play for Health és una plataforma de tele-rehabilitació desenvolupada per l'Àrea de Salut de la Fundació IBIT (Illes Balears Innovació Tecnològica) en col·laboració amb IBSalut i el Servei de Rehabilitació de l'Hospital Son Llàtzer.

Play for Health combina la teràpia amb els elements lúdics dels videojocs per millorar l'eficàcia del tractament i millorar la qualitat de vida dels pacients.

La plataforma és apte per a qualsevol edat i es pot aplicar en diferents àmbits. En l'àmbit sanitari es pot utilitzar per rehabilitació de diferents discapacitats tant en les fases agudes com cròniques. Es pot utilitzar a diferents nivells assistencials com hospitals, centres de salut o residències i centres soci-sanitaris i el més important, és que es pot utilitzar des del propi domicili del pacient implementant qualsevol programa de rehabilitació**.**

Els dispositius utilitzats en aquesta plataforma són el comandament de la Wii, la Kinect, la pista de ball de la Play Station 2 o el guant especial amb sensors. Cada joc està dissenyat per ser utilitzat amb un d'aquests dispositius en concret.

La seva utilitat és tant per patologies neurològiques com per patologies traumàtiques o reumatològiques. Cada joc pot ser utilitzat per millorar mobilitat i aspectes cognitius. El que condiciona escollir un o diversos jocs, és el dèficit que presenti el pacient.

Els jocs estan basats en aspectes cognitius:

- Concentració.
- Memòria.
- Reconeixement de figures o sons.

I es poden combinar amb els dispositius d'interacció:

- Moviment d'una o diverses parts del cos.
- Motricitat fina i/o gruixuda.
- Control de la postura i equilibri, coordinació.

Mentre el pacient està jugant, el videojoc recull estadístiques, és a dir, el nombre d'errors o el temps de reacció. A continuació, s'envien les dades al terapeuta, perquè aquest pugui seguir el progrés i si cal, readaptar el pla**.**

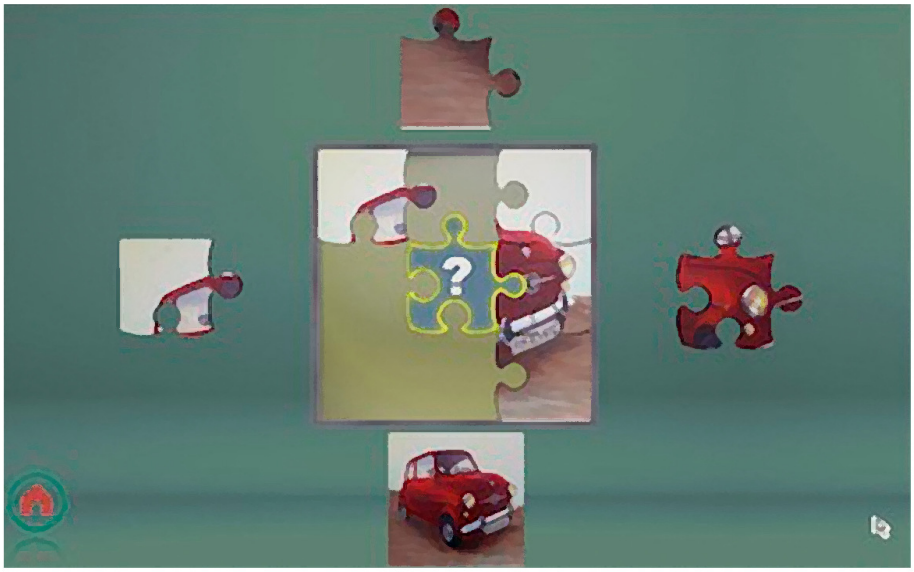

Fig. 3.3. Imatge del videojoc Puzzle de Play For Health.

3.2.2. Previrnec.

L'Institut Guttmann, dins el programa de recerca clínica que desenvolupa des de fa cinc anys com a institut universitari adscrit a la Universitat Autònoma de Barcelona, i amb el suport del Departament d'Universitats, Recerca i Societat de la Informació, i del Departament de Salut, ha creat la plataforma per a la rehabilitació cognitiva, Previrnec, destinada a la rehabilitació de les seqüeles cognitives associades al dany cerebral.

La plataforma Previrnec ofereix al terapeuta la possibilitat d'elaborar plans terapèutics personalitzats i, ajustant-se al grau de capacitació de cada pacient de forma automàtica, instaurar programes de rehabilitació intensiva durant el període de temps necessari, que permeten observar els resultats i adequar el nivell de dificultat en funció del rendiment obtingut en les sessions prèvies.

Fig. 3.4. Pacient utilitzant la plataforma Previnec.

3.2.3. E-health.

E-health o sanitat en línia, és un conjunt de noves tecnologies al servei de la salut, que proposen iniciatives com la tele-rehabilitació o el seguiment mèdic de pacients amb malalties cròniques a casa seva.

Dissenyat per Telefónica, ja s'està provant en hospitals de Barcelona, Palma de Mallorca, Madrid, Torrevieja i Granada. Permet realitzar a casa i sota el control d'un professional certs exercicis de rehabilitació per millorar, per exemple, malalties del genoll.

El terapeuta introdueix en el portal web els exercicis per a cada pacient, personalitza el tractament, visualitza l'execució dels exercicis i es comunica amb ell amb textos o videoconferència. El pacient s'emporta a casa un kit de rehabilitació composat d'un portàtil tàctil amb càmera integrada, connexió a Internet i unes genolleres amb sensors de moviment sense fils. El programa capta els moviments del pacient mitjançant les genolleres i indica si l'exercici s'està realitzant correctament.

Fig. 3.5. Portàtil tàctil amb càmera de E-health.

Un cop analitzades algunes de les plataformes més importants en tele-rehabilitació, en la taula. 3.1 es resumeix quins tipus de rehabilitació pot treballar cada aplicació.

	Play for Health	**Previrnec**	**E-health**	**Rehabilitació amb Kinect**
Rehabilitació reumatològica	Si	No	No	No
Rehabilitació neurològica	Si	Si	No	No
Rehabilitació traumatològica	Si	No	Si	Si
Utilitza Kinect	Si	No	No	Si

Taula. 3.1. Comparativa entre plataformes de tele-rehabilitació.

3.3. Kinect.

Kinect és un controlador de videojoc, és a dir, un perifèric d'entrada utilitzat per controlar els videojocs de la videoconsola Xbox360.

Desenvolupat per Microsoft a finals del 2010, a diferència d'altres controladors de videojoc com els gamepads (més conegut com comandament de videojoc) o els joysticks (palanca de videojoc), Kinect no requereix del contacte físic per interactuar i controlar la videoconsola. Amb el reconeixement de gestos, objectes i ordres de veu i sons es consolida com la interfície natural d'usuari més innovadora del món.

Fig. 3.6. Vista frontal d'una Kinect.

3.3.1. Components de Kinect.

Kinect té forma de barra horitzontal de 23 cm sobre una base circular, tal i com es pot observar en la fig. 3.6. Els seus principals components són:

- Un motor.
- Una càmera RGB.
- Un sensor de profunditat.
- Quatre micròfons.
- Un processador amb memòria cau.

El motor permet ajustar la inclinació del dispositiu per facilitar l'enquadrament de la imatge. La càmera de vídeo (sensor de detecció de llum RGB) capta imatges amb color real i s'utilitza pel reconeixement facial i corporal dels usuaris. El sensor de profunditat està format per un projector infraroig i un sensor de detecció de llum monocromàtic que permet captar l'espai en tres dimensions sota qualsevol condició de llum. Els quatre micròfons permeten detectar la procedència de les veus (diferència entre micròfons). El processador s'encarrega de gestionar tota la informació provinent dels components.

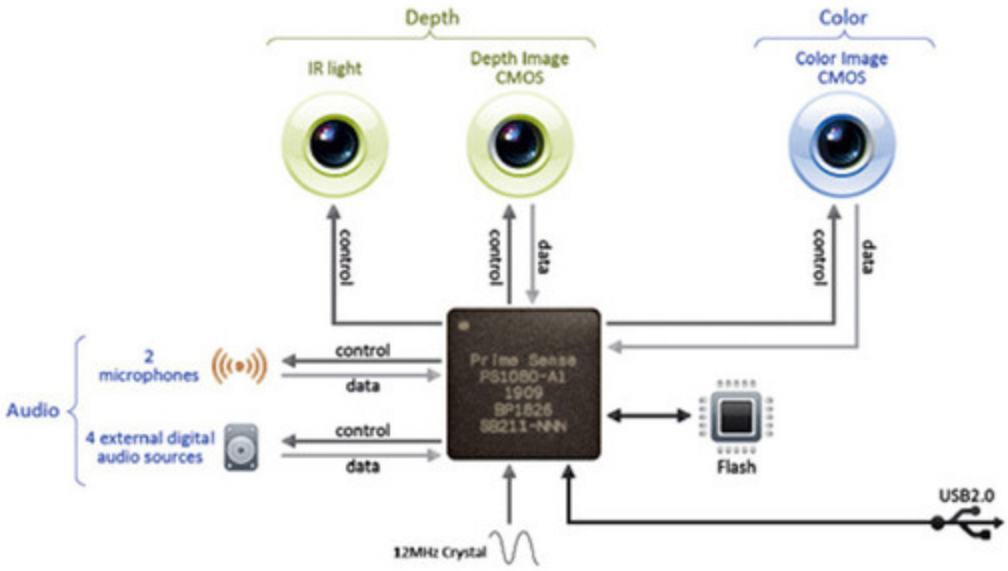

Fig. 3.7. Components d'una Kinect.

3.3.2. Percepció de la profunditat.

L'avantatge qualitatiu més important comparat amb d'altres dispositius de la seva família és la percepció de la profunditat que disposa Kinect.

Amb una càmera RGB es pot captar una imatge en dues dimensions i es pot obtenir una realitat molt precisa de l'entorn. El sensor de profunditat de Kinect proporciona una nova percepció del món real, la profunditat o tercera dimensió.

La precisió en el posicionament d'un objecte segons la distancia en la que es troba es representa amb una escala de colors, com es mostra en la fig. 3.8 a:

- Menys de 50 cm.
- 50 a 70 cm.
- 70 a 100 cm.
- 110 a 150 cm.
- 150 a 300 cm.
- 300 a 600 cm.
- Més de 600 cm.

Amb aquesta percepció de la profunditat es pot focalitzar l'element que es pretén processar. Permet la correcta identificació de l'objecte respecte a d'altres elements de l'escenari situats a una altra distància i que no són transcendents.

Fig. 3.8. Representació en colors de la profunditat percebuda per Kinect.

3.4. Kit de desenvolupament.

Originalment, Kinect és un dispositiu fabricat per ser utilitzat únicament amb la videoconsola Xbox360 però el ventall de funcionalitats s'amplia si s'utilitza en un PC. Microsoft no disposa de cap eina de desenvolupament per poder utilitzar Kinect en d'altres aparells electrònics. Degut a això, alguns programadors i empreses especialitzades en codi obert han treballat fins a publicar un driver no oficial per Kinect.

En el desenvolupament de Rehabilitació amb Kinect són necessaris:

- Drivers i llibreries per Kinect.
- Framework de processament d'imatge.
- Entorn / llenguatge de desenvolupament.

3.4.1. Drivers i llibreries per Kinect.

Els drivers són un conjunt d'arxius amb la configuració necessària per què el sistema operatiu pugui reconèixer un dispositiu. Les llibreries són un conjunt d'arxius capaços de proporcionar ús al dispositiu. Els més comuns per Kinect són:

- Libfreenect.

Libfreenect és un paquet format per un driver i una llibreria per Kinect. Desenvolupat per la comunitat de programadors de codi obert orientat únicament a Kinect més extensa del món, OpenKinect. Codificat en varis llenguatges de programació i disponible per una gran diversitat de dispositius com ordinadors, dispositius mòbils, etc...

- CL NUI Platform.

CL NUI Platform és una solució auto instal·lable i de codi obert per Kinect de molt fàcil instal·lació i utilització. Disponible únicament per Windows i amb llibreries pre-compilades. Code Laboratories és l'empresa especialitzada en programari de codi obert encarregada de desenvolupar aquest driver i d'altres, com per exemple, CL EYE Platform (càmera de PlayStation 3).

- PrimeSenseNITE.

PrimeSenseNITE és un altre paquet de driver i llibreria, desenvolupat per l'empresa encarregada de fabricar Kinect per Microsoft. PrimeSense fabrica altres dispositius amb les mateixes característiques de Kinect per a la interacció de dispositius multimèdia.

En la taula 3.2, es mostra, per a cada driver, la compatibilitat amb diversos sistemes operatius i llenguatges de programació. A més, s'indica el tipus de llicència, el grau de dificultat per instal·lar el framework i si aquest disposa de llibreries pre-compilades.

	Libfreenect	**CL NUI Platform**	**PrimeSenseNITE**
Windows	Si	Si	Si
Linux	Si	No	Si
OS X	Si	No	Si
.Net	Si	Si	Si
Java	Si	No	Si
Llibreries pre-compilades	No	Si (Windows)	No
Instal·lació	Complicada	Fàcil	Complicada
Tipus de llicència	Apache 2.0 / GPL2	GNU GPL	GNU Lesser General Public License

Taula 3.2. Comparativa entre diferents drivers per Kinect.

3.4.2. Frameworks de processament d'imatge.

Els marcs de treball en processament d'imatge ofereixen un conjunt de tècniques i mètodes orientats al tractament d'imatges digital per facilitar la cerca d'informació en les imatges.

Els més comuns són:

- OpenCV.

Un dels frameworks en processament d'imatges més populars. De codi obert i gratuït, està implementat amb C++. La seva gran quantitat de mètodes i efectes digitals proporciona moltes funcionalitats al programador. Disponible també per C# anomenat EmguCV.

- OpenNI.

És un framework de codi obert, multi plataforma i disponible en diversos idiomes. Format per un conjunt d'interfícies especialment creades per desenvolupar aplicacions amb interacció natural (NI), és a dir, aplicacions controlades per moviments humans.

- Java Advanced Imaging.

Són un conjunt d'algorismes que implementen un gran número de funcionalitats en el tractament d'imatges com, càlcul d'àrees, grafisme i efectes i tractament d'imatges.

La taula 3.3 completa la informació de quin tipus de llicencia i quins sistemes operatius i llenguatges de programació són compatibles per a cada framework.

	OpenCV	**OpenNI**	**ImageJ**
Windows	Si	Si	Si
Linux	No	Si	Si
OS X	No	Si	Si
.Net	C++ / C# (EmguCV)	No	No
Java	No	Si	Si
Tipus de llicència	BSD	GNU GPL	GNU GPL

Taula 3.3. Comparativa entre diferents frameworks de processament d'imatge.

L'opció d'utilitzar el framework OpenNI especialitzat en el control d'aplicacions amb els moviments humans sembla la més adient. OpenNI inclou algorismes realment útils capaços de detectar òrgans humans, com per exemple, els ulls d'una persona en una imatge, els moviments d'una mà per activar accions d'una aplicació o algorismes capaços d' interpretar els moviments de tot l'esquelet humà.

La intenció en el treball/projecte desenvolupat no és treballar amb mètodes de tan alt nivell. L'objectiu és obtenir la informació necessària d'una imatge per tal de reconèixer parts del cos humà de la manera més personalitzada possible.

Per tant, qualsevol framework amb efecte thresholding, retolació i grafisme és útil pel desenvolupament d'aquesta aplicació. Rehabilitació amb Kinect està implementat amb EmguCV.

3.4.3. Entorn / llenguatge de desenvolupament.

Analitzant totes les eines anteriors, drivers, llibreries i frameworks, no existeix cap imperatiu que obligui a desenvolupar l'aplicació sota cap plataforma o llenguatge de programació en concret.

Tenint en compte que Kinect està comercialitzat per Microsoft i aquest disposa d'un entorn de desenvolupament propi, Rehabilitació amb Kinect s'ha desenvolupat amb el llenguatge de programació orientat a objectes C# amb Visual Studio 2010. D'aquesta manera, si en un futur es publiqués d'un driver oficial de Microsoft per Kinect, podria ser compatible amb RaK.

4. Anàlisis.

4.1. Planificació.

La planificació temporal de treball s'estableix amb el següent calendari:

- Investigació sobre Kinect, les llibreries disponibles i els frameworks compatibles. Duració estimada de 40 hores.
- Proves de desenvolupament per obtenir control sobre Kinect. Duració aproximada de 80 hores.
- Desenvolupament del programari. Duració aproximada de 190 hores.
- Redacció de la documentació i presentació. Duració de 40 hores.

4.2. Requeriments.

Pel correcte desenvolupament de l'aplicació, com a part prèvia, s'ha de realitzar un estudi sobre Kinect i el processament d'imatges digitals. Paral·lelament, també s'ha de realitzar un altre estudi sobre quins tipus de teràpia es pot treballar.

Com a resultat final, el pacient ha de poder realitzar una sessió de rehabilitació amb Kinect però, a més, es contempla la possibilitat que aquest programari pugui ser utilitzat amb algun altre dispositiu més comú, com per exemple, una càmera web.

El pacient ha de conèixer en qualsevol moment quina és l'evolució en la recuperació de la seva malaltia.

No es requereix la implementació de cap mena d'editor de dades del pacient o qualsevol altra gestió de les dades com inserció o eliminació de pacients o teràpies. Suposant que un usuari detectés un error en les seves dades, com per exemple, alguna dada personal incorrecte o alguna teràpia a realitzar mal prescrita, l'usuari hauria de posar-se en contacte amb l'empresa gestora d'aquestes dades, en aquest cas, el Servei Català de Salut.

4.3. Casos d'ús.

A continuació es defineixen els casos d'ús a implementar.

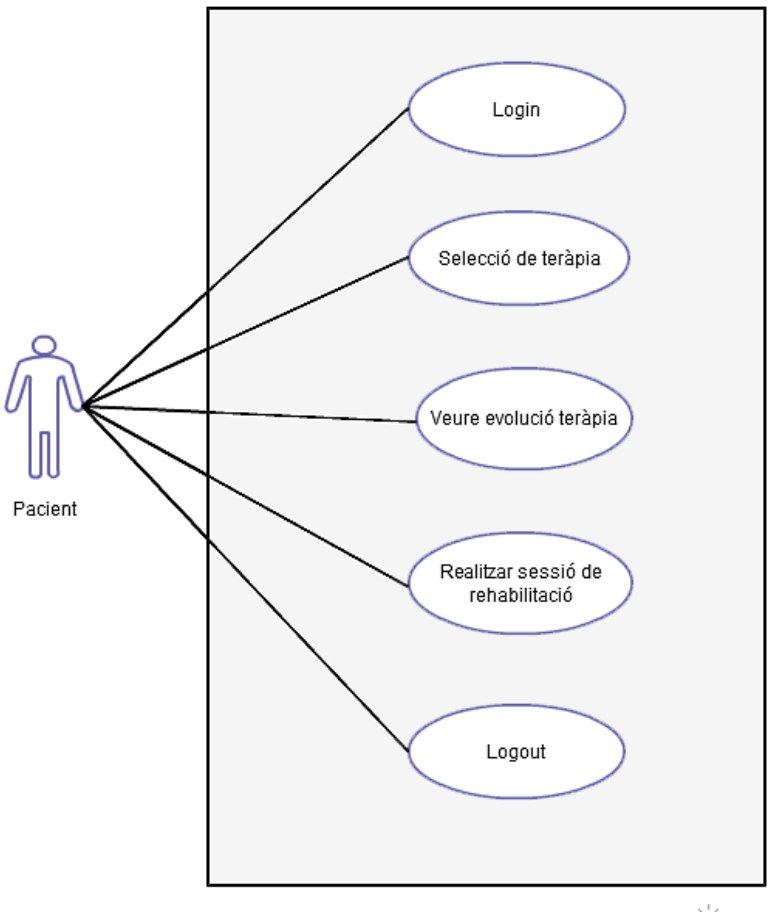

Fig. 4.1. Diagrama dels casos d'ús.

Cas d'ús Log In

Actor: Pacient.

Pre-condicions: No hi ha cap pre-condició.

Flux normal:

1. El sistema mostra una finestra amb dos camps de text per poder introduir el codi de CIP de l'usuari.

2. L'usuari introdueix el codi CIP.

3. El sistema comprova l'existència de l'usuari i consulta les teràpies i les sessions de rehabilitació de l'usuari realitzades anteriorment.

Flux alternatiu:

2.1. L'usuari introdueix un codi CIP amb un format incorrecte.

2.1.2. El sistema informa que el codi CIP no té un format correcte.

3.1 El sistema comprova que el pacient no existeix i informa a l'usuari.

3.2 El sistema comprova que el pacient no té cap teràpia a rehabilitar i informa a l'usuari.

* L'usuari pot tancar l'aplicació en qualsevol moment.

Post-condicions: El sistema realitza el cas d'ús Seleccionar teràpia.

Cas d'ús Seleccionar Teràpia

Actor: Pacient.

Pre-condicions: L'usuari ha d'haver realitzat el flux normal del cas d'ús Log In.

Flux normal:

1. El sistema mostra un llistat de les teràpies que el pacient pot realitzar.

2. L'usuari tria una teràpia de la llista.

3. El sistema mostra, de la teràpia triada, un llistat dels moviments que pot realitzar.

4. L'usuari tria un moviment de la llista.

5. El sistema activa el botó Veure evolució de la teràpia.

6. L'usuari tria un dispositiu de la llista.
7. El sistema activa el botó Realitzar sessió de rehabilitació.
8. L'usuari prem el botó Realitzar sessió de rehabilitació.

Flux alternatiu:

5.1 L'usuari prem el botó Veure evolució de la teràpia.

5.1.2 El sistema realitza el cas d'us Veure evolució de la teràpia.

* L'usuari pot tancar l'aplicació en qualsevol moment.

Post-condicions: El sistema realitza el cas d'ús Realitzar sessió de rehabilitació.

Cas d'ús Veure Evolució de la Teràpia

Actor: Pacient.

Pre-condicions: L'usuari ha d'haver realitzat el flux alternatiu del cas d'ús Seleccionar teràpia.

Flux normal:

1. El sistema mostra una gràfica amb l'evolució del moviment de la teràpia triada.

Flux alternatiu:

* L'usuari pot tancar l'aplicació en qualsevol moment.

Post-condicions: No hi ha cap post-condició.

Cas d'us Realitzar Sessió de rehabilitació

Actor: Pacient.

Pre-condicions: L'usuari ha d'haver realitzat el flux normal del cas d'ús Seleccionar teràpia.

Flux normal:

1. El sistema mostra una finestra amb la imatge del pacient a través del dispositiu seleccionat i informació referent a la sessió de rehabilitació que es realitzarà.
2. El sistema indica a l'usuari que premi virtualment el botó vermell situat en la imatge per poder començar la sessió de rehabilitació.
3. L'usuari prem el botó vermell de la imatge.
4. El sistema informa a l'usuari que la sessió de rehabilitació ha començat.
5. L'usuari realitza la sessió de rehabilitació.
6. El sistema mostra, a temps real, tota la informació de la sessió que està realitzant l'usuari.
7. Transcorre el temps establert per a la sessió de rehabilitació seleccionada.
8. El sistema gestiona les dades obtingudes i les guarda. El sistema informa del final de la sessió.

Flux alternatiu:

8.1 El sistema gestiona les dades obtingudes i detecta que l'usuari ha realitzat poques repeticions. El sistema informa que la sessió no s'ha realitzat correctament.

* L'usuari pot tancar l'aplicació en qualsevol moment.

Post-condicions: El sistema torna al cas d'ús Seleccionar teràpia.

Cas d'ús Log Out

Actor: Pacient.

Pre-condicions: L'usuari ha d'haver realitzat el cas d'ús Log In.

Flux normal:

1. L'usuari prem Tancar Pacient des de Arxiu > Tancar Pacient.
2. El sistema tanca el pacient actual i informa a l'usuari.

Post-condicions: El sistema pot realitzar el cas d'ús Log In.

4.4. Metodologia de treball.

El model de desenvolupament software utilitzat ha estat el model iteratiu i incremental.

La metodologia principal en el model iteratiu i incremental és desenvolupar un programari el qual incrementa les seves funcionalitats a cada iteració. Per treballar amb aquest model és necessari establir quines fases es realitzaran a cada iteració. Aquestes fases poden ser definides com a un conjunt de casos d'ús o com a un llistat de funcionalitats.

Les fases de cada iteració es defineixen amb els següents casos d'ús:

- Primera iteració: casos d'ús login, logout, selecció de teràpia i veure evolució de la teràpia (amb les dades d'exemple introduïdes).
- Segona iteració: cas d'ús realitzar sessió de rehabilitació.

5. Disseny.

5.1. Disseny de la persistència.

La principal persistència de RaK són les dades obtingudes en cada sessió de rehabilitació. Amb aquestes dades es podrà dur a terme un control sobre l'evolució del pacient en el temps. A més, romanen enregistrades totes les articulacions del cos humà amb els seus respectius moviments.

L'aplicació no pot prescriure quina teràpia de rehabilitació ha de fer cada pacient, per tant, aquesta base de dades està dissenyada per ser connectada amb una base de dades externa. Amb el codi CIP del pacient, s'obtenen les dades personals i les teràpies que ha de realitzar el pacient.

5.1.1. Descripció de les entitats i associacions del model conceptual.

L'entitat *Pacient* conté les dades personals del usuari. *Nom, cognoms, número de CIP* i un *identificador*. Aquest usuari pot tenir prescrites varies teràpies o, pel contrari, disposar de plena salut.

L'entitat *Teràpia* té les dades referents al tipus de teràpia que un pacient pot realitzar. Aquestes dades són el *nom, descripció* i un *identificador*.

L'entitat *Moviment* està associat amb teràpia. Per cada teràpia s'ha de saber quins moviments s'han de realitzar. Cada teràpia pot tenir un o més moviments. Aquests moviments contenen un *identificador*, un *nom*, una *descripció*, una *ajudaLink i ajudaText* destinada a un multimèdia o una redacció per orientar al pacient sobre el moviment, un *numRepeticions* amb la quantitat de repeticions que ha de realitzar el pacient en cada sessió, un *angle1 i angle 2* corresponen als angles del moviment que es possible realitzar quan es disposa de total mobilitat articular, i el *temps* que s'ha d'invertir per realitzar el moviment en una sessió. En resum, conté les dades de com s'ha de realitzar el moviment.

L'entitat *Sessió* conté les dades de com s'ha fet el moviment i quan. L'associació amb *Pacient* descriu qui ha fet la sessió. L'associació amb *Moviment* indica quin moviment s'ha realitzat en la sessió.

L'entitat *Articulació* conté el *nom, identificador i descripció* de les articulacions del cos humà relacionats amb el seus moviments.

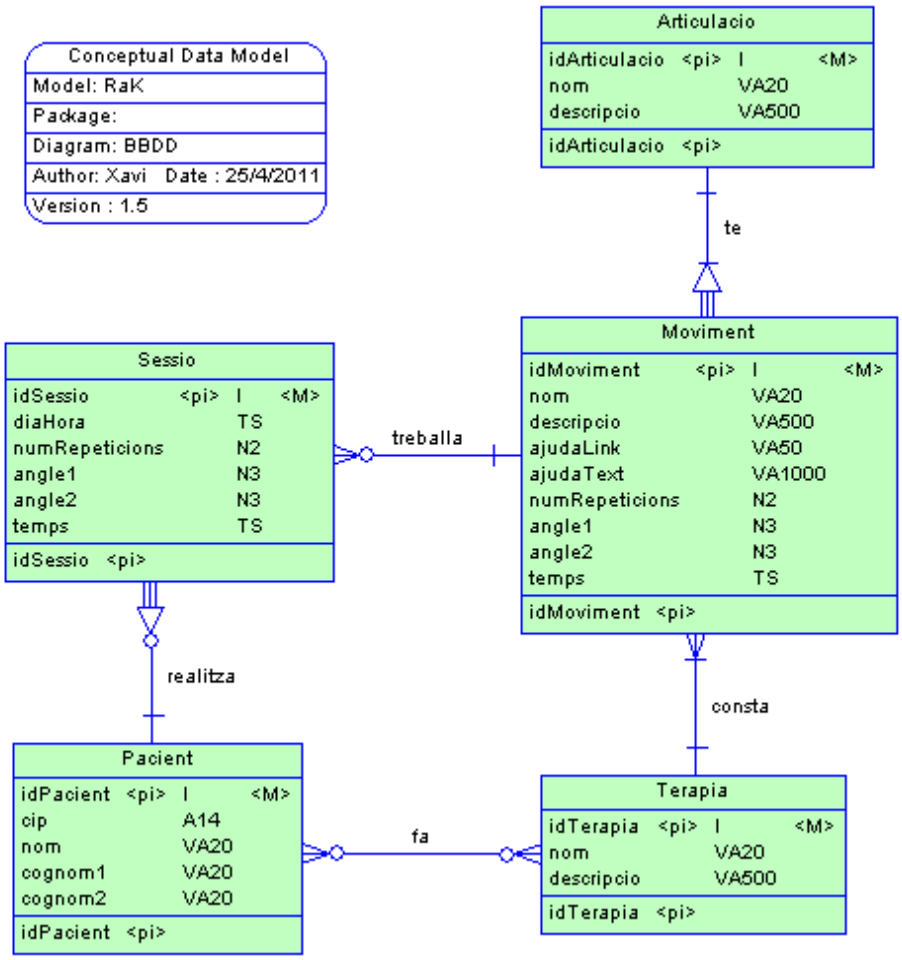

Fig. 5.1. Diagrama conceptual de la base de dades.

5.1.2. Regles de negoci.

S'estableixen les següents regles de negoci:

- *numRepeticions* de les entitats *Sessió* i *Moviment* han de ser valors positius.
- *angle1* de les entitats *Sessió* i *Moviment* han de ser valors positius.
- *angle2* de les entitats *Sessió* i *Moviment* han de ser valors positius.
- Les *Sessions* d'un *Pacient* han de ser d'un *Moviment* corresponen a una *Terapia* que tingui prescrita aquest pacient.

Les regles de negoci de *numRepeticions, angle1* i *angle2* es controlen amb una restricció SQL *check* en el moment de crear la base de dades. La restricció de les sessions d'un pacient es realitzen a nivell d'aplicació.

5.1.3. Descripció del model físic.

Per crear el disseny físic de la base de dades, s'utilitza el convertidor de models de Power Designer a partir del model conceptual com es mostra en la fig. 5.2. En el model físic apareix una nova taula anomenada que relaciona les taules *Pacient* i *Teràpia*. Aquestes tres taules formarien part de la base de dades externa. El disseny físic de les taules *Pacient* i *Teràpia* és orientatiu. S'ha dissenyat amb les dades més bàsiques i necessàries pel funcionament de RaK. Altres dades personals, com per exemple, la residència o la data de naixement, no són transcendents per l'aplicació.

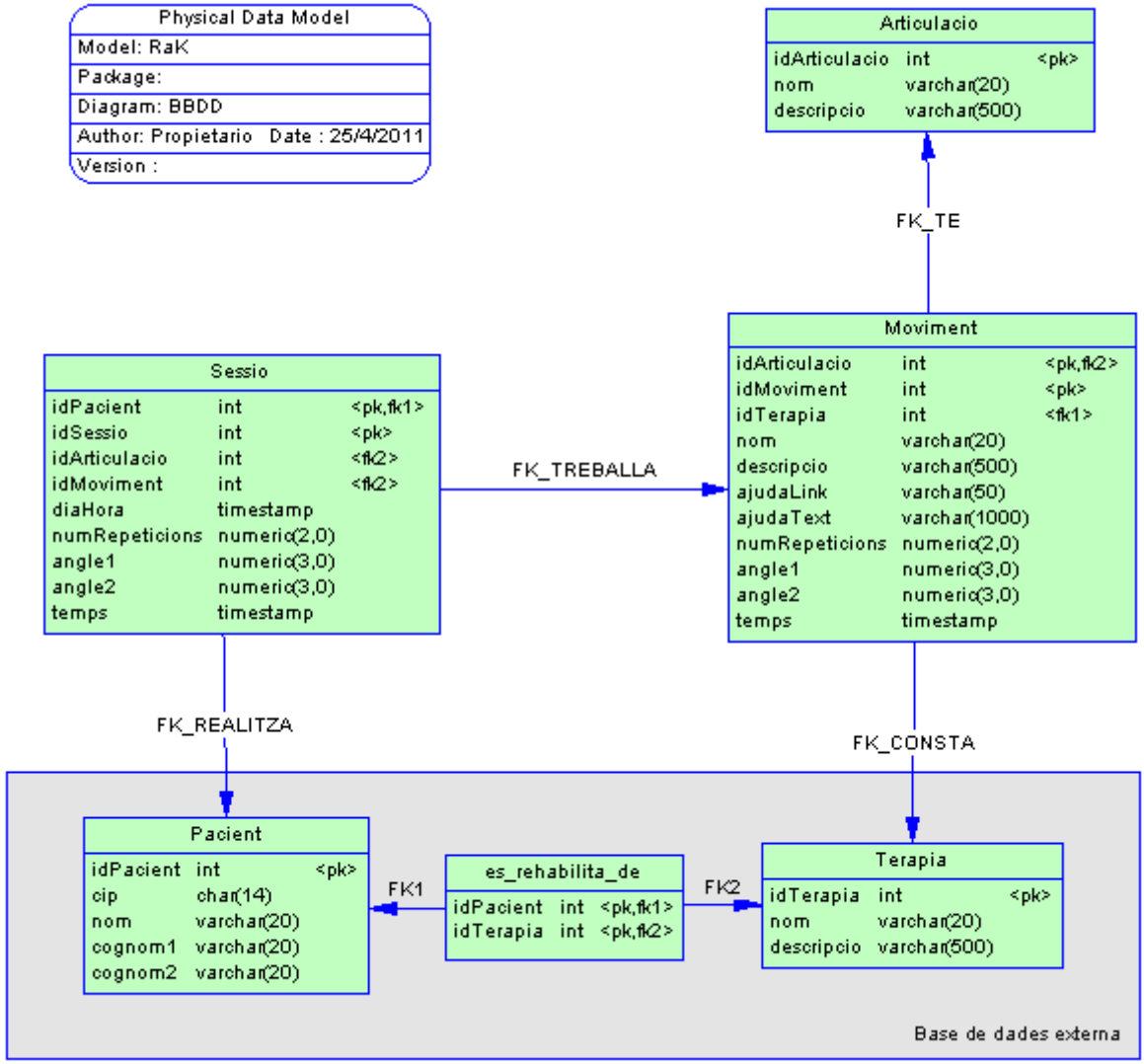

Fig. 5.2. Diagrama físic de la base de dades.

5.1.4. Motor de la base de dades.

El motor de la base de dades és MySQL. MySQL és un dels gestors més utilitzats en aplicacions gratuïtes. És una base de dades relacional, es a dir, un conjunt de dades que estan emmagatzemades en taules entre las quals s'estableixen unes relacions per poder manipular les dades d'una forma més eficient i segura. Per usar i gestionar la base de dades relacional de MySQL s'utilitza el llenguatge estàndard de programació SQL.

6. Desenvolupament.

Rehabilitació amb Kinect és una aplicació de sobretaula i el seu funcionament es basa en el processament d'imatge.

Una de les tècniques més utilitzades en el camp del processament d'imatges en general és la binarització de la imatge. D'aquesta manera s'obté una imatge amb només dos tons, per exemple blanc i negre, traduït a un llenguatge de programació podria ser 1 i 0 o true i false. Amb una imatge en color real, el rang de valors és massa elevat i aleatori, això dificulta el processament de la imatge, com per exemple, fer el seguiment d'un objecte dins un escenari.

Per la correcta detecció d'aquest objecte dins l'escenari utilitzant un dispositiu sense sensor de profunditat, com per exemple, una càmera web, és molt important la manera que estan il·luminats. Per exemple, si es situa un objecte més il·luminat que la resta de l'escenari, es pot obtenir una imatge binaria només amb la imatge de l'objecte. En canvi, amb el sensor de profunditat de Kinect, els objectes no cal utilitzar una il·luminació especial o diferent respecte a l'escenari, simplement cal que estiguin situats a una distància en concret.

Aquestes imatges per processar es representen en forma de matriu. Cada posició de la matriu representa un píxel de la imatge. Quan un píxel de la imatge conté part de l'objecte, aquesta posició de la matriu pren valor. D'aquesta manera es pot situar l'objecte dins de la matriu de píxels.

Fig. 6.1. Exemple d'una imatge real, binaria i de profunditat amb Kinect respectivament.

6.1. Configuració de l'entorn de treball.

Com a pas previ al desenvolupament del codi, cal configurar l'entorn de treball. És necessari realitzar les següents tasques:

- Instal·lar driver de Kinect.
- Instal·lar el motor de la base de dades.
- Executar el script per crear la base de dades.
- Executar el script per introduir les dades d'exemple.
- Importar les llibreries necessàries en el projecte de Visual Studio 2010.

La instal·lació del driver CL NUI Platform per Kinect és un procés ben senzill. El driver es presenta en un arxiu executable i auto-instal·lable.

La instal·lació del servidor WampServer incorpora un motor de base de dades MySQL i el procés també és prou simple. El software esta format per un arxiu executable amb una configuració de fàbrica molt útil.

El codi en llenguatge SQL encarregat de crear la base de dades s'ha obtingut amb el generador de scripts que incorpora el software PowerDesigner a partir del disseny físic de la base de dades.

El codi SQL que introdueix les dades d'exemple a la base de dades s'ha generat amb l' editor de text pla Notepad de Windows. Conté varis usuaris d'exemple, algunes teràpies que poden realitzar aquests usuaris, moviments, articulacions i unes sessions de rehabilitació, també d'exemple.

L'entorn de desenvolupament integrat Visual Studio 2010 incorpora moltes funcionalitats per defecte que s'utilitzen en RaK. Per exemple, la llibreria *Math* conté multitud d'algorismes que faciliten el càlcul d'operacions matemàtiques. Però, a més, pel desenvolupament caldrà algunes llibreries extres, descrites en el següent capítol.

6.2. Instal·lació de les llibreries.

Per incloure una nova llibreria al projecte C# per desenvolupar, cal fer-ho de dues maneres diferents; o bé seleccionant l'arxiu des del menú *Proyecto > Agregar nuevo elemento* de Visual Studio 2010 o copiant l'arxiu dintre de les carpetes *Release i Debug* situades dins la carpeta *bin* del projecte de Visual Studio 2010.

Les llibreries que han de ser afegides com a nou element al projecte són:

- Llibreria EmguCV (processament d'imatges).
- Llibreria Charting (representació de gràfiques).
- Llibreria MySQL (connexió amb la base de dades).

Les llibreries que calen ser copiades dins les carpetes anteriorment esmentades són:

- Llibreria CLNUIDevice.
- Llibreria OpenCV.

La llibreria EmguCV està formada per un conjunt d'arxius amb extensió *.dll*:

- *Emgu.CV.dll*
- *Emgu.CV.UI.dll*
- *Emgu.Util.dll*
- *Zlib.net.dll*

La llibreria Charting també està formada per dos arxius *.dll*:

- *System.Windows.Forms.DataVisualization.dll*
- *System.Windows.Forms.DataVisualization.Design.dll*

La llibreria MySQL té com a funcionalitat connectar el motor MySQL amb el projecte C#. Es composa d'un únic arxiu pel seu funcionament més bàsic; mysql.data.dll.

La llibreria CLNUIDevice està formada per un únic arxiu anomenat CLNUIDevice.dll i és l'encarregat de controlar el dispositiu Kinect.

La llibreria OpenCV es invocada per EmguCV i consta dels següents arxius *.dll*:

- *opencv_core211.dll*
- *opencv_highgui211.dll*
- *opencv_imgproc211.dll*

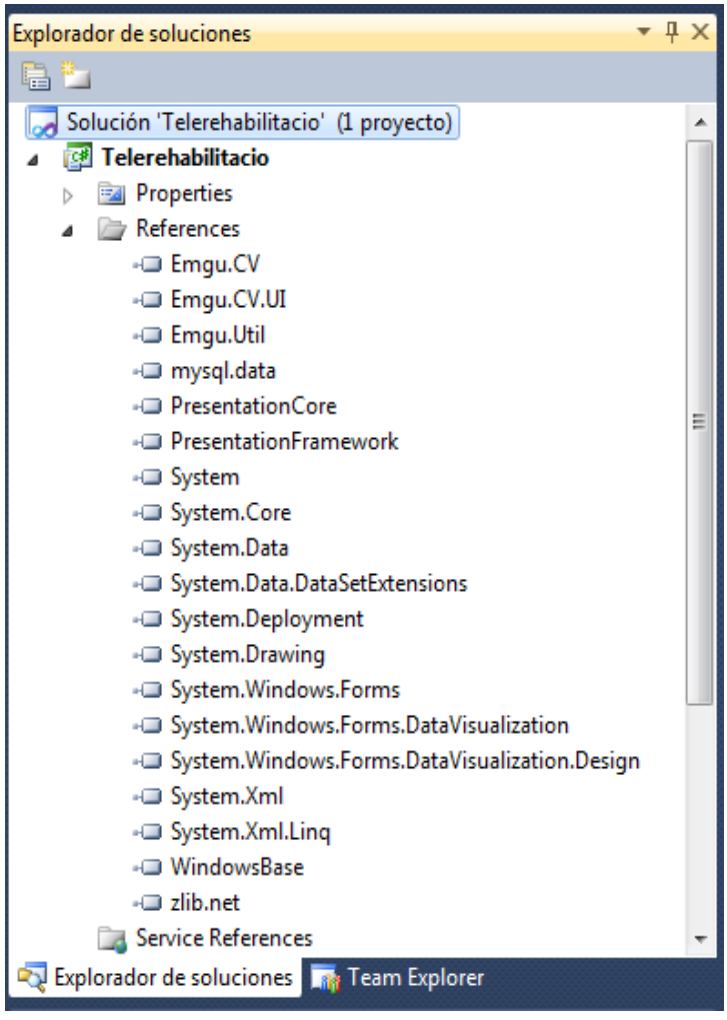

Fig. 6.2. Llistat de llibreries.

6.3. Estructuració de les capes.

El desenvolupament de Rehabilitació amb Kinect està estructurat per capes, cadascuna d'elles amb una funcionalitat. Les capes es distribueixen en quatre carpetes amb el seus respectius espais de noms (*namespace)*:

- Aplicació
- Domini
- Persistència
- Presentació

La capa aplicació conté les classes encarregades de realitzar tots els càlculs i cerques de l'aplicació. Càlculs matemàtics i trigonomètrics, estructuració de les dades per la construcció de les gràfiques, eliminació de pacients d'una sessió, localització de punts d'una articulació des d'un fotograma de Kinect...

La capa domini hi ha les classes que modelen la lògica del negoci de l'aplicació.

La capa persistència és l'encarregada d'emmagatzemar i restaurar totes les dades. Accedeix a les dades personals dels usuaris, guarda les dades que es generen d'una sessió de rehabilitació...

La capa presentació reuneix totes les classes que aporten interacció entre l'usuari i l'aplicació. Inclou les classes amb disseny de finestres (Windows Forms), gràfiques i elements multimèdia.

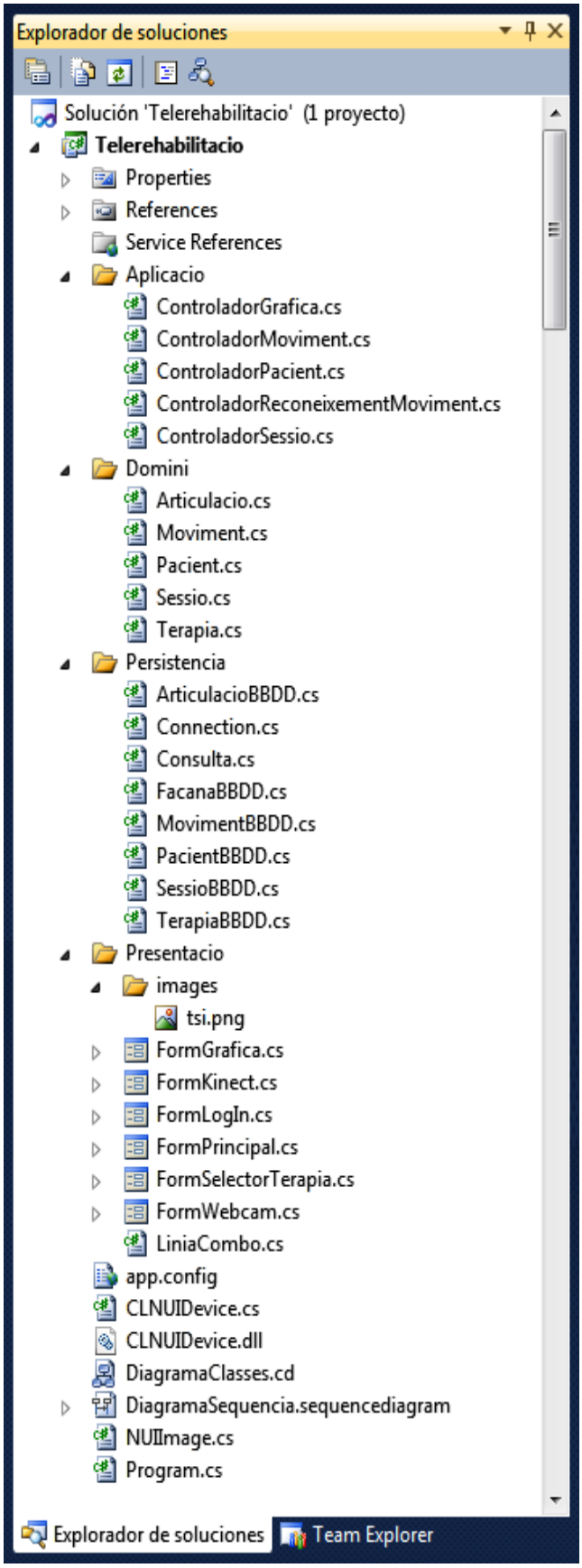

Fig. 6.3. Distribució de les classes dintre de les capes.

6.4. Implementació.

En els següents apartats es descriuen quines funcions es realitzen a cada part de l'aplicació, com es realitzen i quins elements (classes i mètodes) les realitzen.

6.4.1. Implementació Log In.

La implementació del cas d'ús Log In és l'encarregada d'iniciar l'aplicació depenen de quin usuari l'utilitzi.

Les classes utilitzades en aquesta implementació són:

- Totes les classes de la capa Domini.
- Totes les classes de la capa Persistència.
- Les classes *FormLogIn i FormPrincipal* de la capa Presentació.
- La classe *ControladorPacient* de la capa Aplicació.

La classe *FormPrincipal* és la finestra (*Windows Forms*) contenidora de totes les altres finestres que apareixen durant la utilització de l'aplicació. Com a inici del programa, es mostra la finestra *FormLogIn* dintre de *FormPrincipal*.

Com a primer pas, l'usuari ha d'introduir el seu codi CIP. La classe *FormLogIn* s'encarrega de comprovar si el format d'aquest codi és correcte. El codi CIP està format per dues cadenes de caràcters; la primera està formada per quatre caràcters alfabètics i la segona cadena està formada per deu caràcters numèrics. La comprovació es realitza amb els mètodes *controlLletres()* i *controlNumeros()* respectivament.

Amb el codi CIP correcte, la classe *FormLogIn* crida el mètode *getPacient(CIP)* de la classe *ControladorPacient* per obtenir les dades personals del pacient.

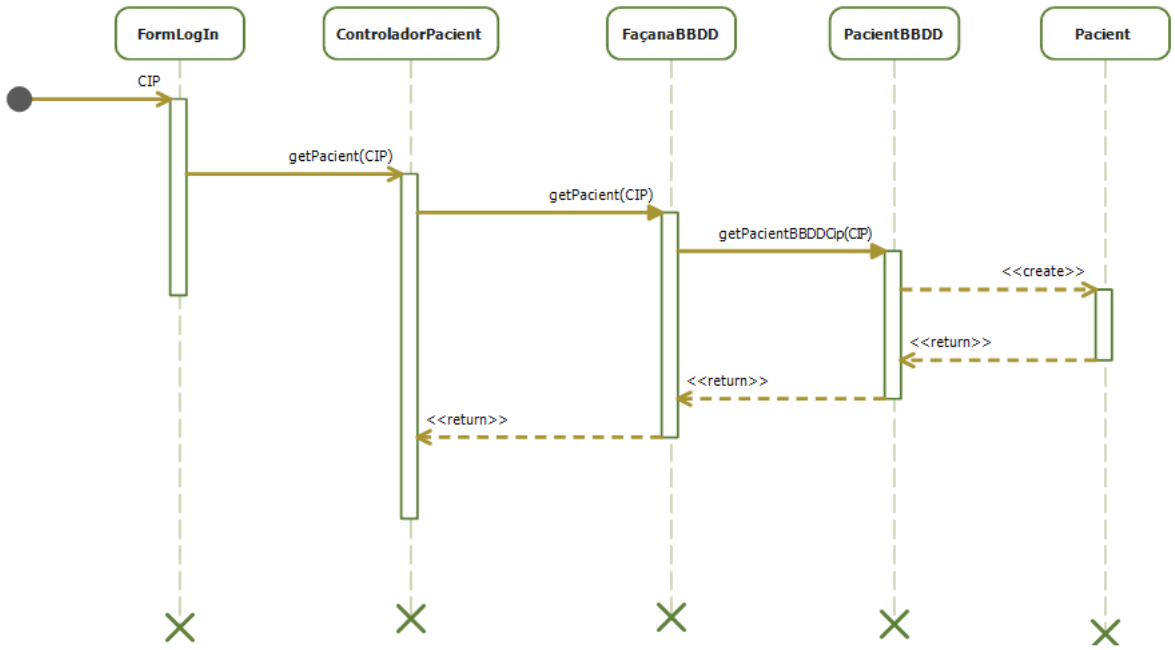

Fig. 6.4. Diagrama de seqüència del mètode *getPacient()*.

A continuació, el mètode privat *inicialitzarPacient()* de la classe *FormLogIn* realitza una sèrie de crides mètodes de la classe *ControladorPacient* amb la finalitat de recuperar totes les dades relatives al nou pacient. Els mètodes són:

- *afegirTerapiesPacient*(). Recupera i crea tots els objectes de la classe *Teràpia* que corresponen a les teràpies que ha de realitzar el pacient.

- *afegirMovimentTerapies()*. Recupera i crea l'objecte de la classe *Moviment* que s'ha de realitzar per a cada *Teràpia* recuperada en el mètode anterior.

- *afegirArticulacioMoviments()*. Recupera i crea l'objecte de la classe *Articulació* i s'associen per a cada Moviment recuperat en el mètode anterior.

- *afegirSessionsPacient()*. Recupera i crea tots els objectes de la classe *Sessió* corresponents a sessions del pacient realitzades anteriorment.

- *afegirMovimentSessions()*. Recupera i crea l'objecte de la classe Moviment corresponen a cada sessió recuperada en el mètode anterior.

En la fig. 6.5 es mostra el diagrama de seqüència del mètode *afegirTerapiesPacient()*. Els altres mètodes encarregats d'inicialitzar el pacient realitzen una seqüència molt similar.

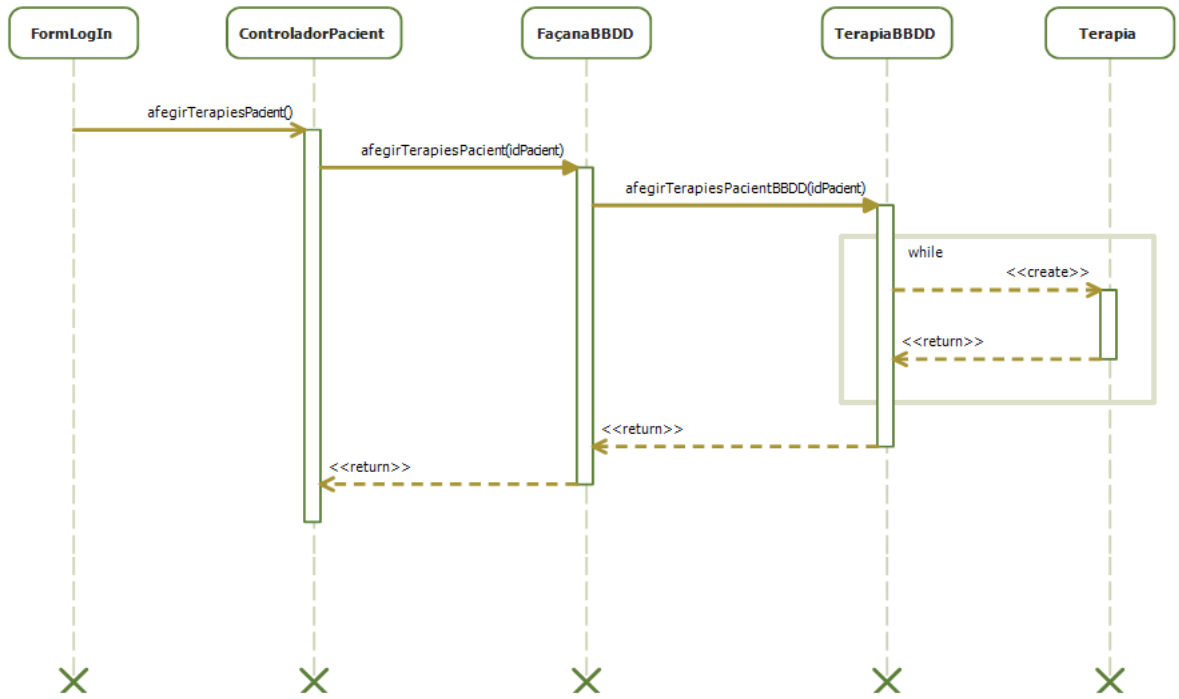

Fig. 6.5. Diagrama de seqüència del mètode *afegirTerapiesPacient()*.

Després d'executar tots els mètodes, l'aplicació té tots els moviments de l'articulació, totes les teràpies de rehabilitació i totes les dades personals del pacient.

Com a darrer pas, es crea un objecte de la classe *FormSelectorTerapies*.

6.4.2. Implementació Selector de teràpies.

La implementació Selector de teràpies és l'encarregada de mostrar a l'usuari les teràpies i els moviments que pot realitzar per consultar l'evolució o realitzar una sessió de rehabilitació d'una d'aquestes teràpies.

Les classes utilitzades en aquesta implementació són:

- Les classes *FormSelectorTerapia, FormKinectForm, FormWebcam, FormGrafica i FormPrincipal* de la capa Presentació.
- La classe *ControladorPacient i ControladorMoviment* de la capa Aplicació.

El disseny de la finestra per seleccionar una teràpia *(FormSelectorTerapia)* està format per tres comboBox i dos botons.

En el diagrama de la fig.6.6 es mostra gràficament els processos la implantació Selector de teràpies.

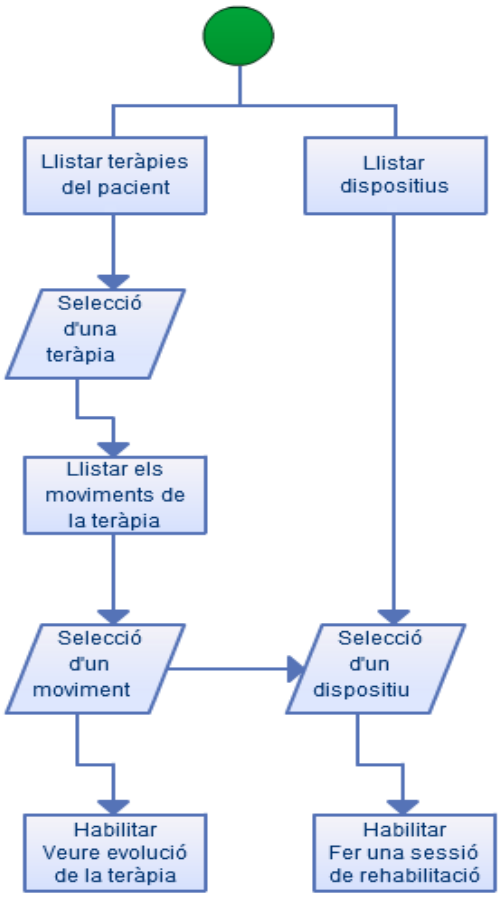

Fig. 6.6. Diagrama de procés de Selector de teràpies.

El comboBox teràpia s'omple automàticament amb les dades obtingudes en el cas d'ús anterior (LogIn) amb el mètode privat *mostrarTerapies()*.

Un cop l'usuari ha triat una de les teràpies disponibles, s'executa el mètode privat *motrarMovimentsTerapies()* . Aquest omple el comboBox dels moviments segon la teràpia triada.

En aquest moment, el botó Veure evolució de la teràpia s'activa i permet el seu ús. Aquest botó crea un objecte de la classe *FormGrafica*.

El botó Realitzar sessió de rehabilitació s'activa després de triar un dispositiu del comboBox dispositius. L'acció d'aquest botó crea un objecte de la classe *FormKinect o FormWebcam* segons quin dispositiu s'hagi triat.

6.4.3. Implementació Veure evolució de la teràpia.

La implementació Veure evolució de la teràpia és l'encarregada de generar una gràfica lineal per mostrar l'evolució d'una teràpia.

Les classes utilitzades en aquesta implementació són:

- Les classes *FormGrafica i FormPrincipal* de la capa Presentació.
- Les classes *ControladorGrafica* de la capa Aplicació.

El mètode privat *representarGrafica()* de la classe *FormGrafica* realitza dues crides a tres mètodes de la classe *ControladorGrafica*. Cadascuna d'aquestes crides genera la línia evolutiva de la gràfica de cadascun dels dos angles del moviment. Els mètodes *getXDataHores(), getYValorsAngle1() i getYValorsAngle2()* de la classe *ControladorGrafica* retornen un vector amb les dades corresponents al dia i l'hora que es va realitzar la sessió de rehabilitació en l'eix X i els dos angles assolits en l'eix Y.

6.4.4. Implementació Realitzar sessió de rehabilitació amb Kinect.

La implementació Realitzar sessió de rehabilitació amb Kinect té com a funció obtenir les dades necessàries d'una imatge de Kinect per poder realitzar els càlculs adients durant una sessió de rehabilitació.

Les classes utilitzades en aquesta implementació són:

- Les classes *FormKinect i FormPrincipal* de la capa Presentació.
- Les classes *ControladorSessio, ControladorReconeixementMoviment i ControladorMoviment* de la capa Aplicació.
- La classe *CLNUIDevice i NUIImage.*

El mètode privat *iniciarKinect()* de la classe *FormKinect* conté el codi necessari per usar la Kinect i realitzar la sessió de rehabilitació. El procediment d'aquest mètode és:

1. Es crida al mètode *GetDeviceCount()* per comprovar si hi ha algun dispositiu Kinect disponible.
2. Es creen els objectes del dispositiu Kinect amb els mètodes *CreateMotor(devSerial)* i *CreateCamera(devSerial).*
3. Es situa la inclinació del dispositiu Kinect en el centre i s'activa el led de funcionament de Kinect.
4. Es crea un objecte de la classe *NUIImage* que conté la informació de cada fotograma.
5. S'inicia el dispositiu amb el mètode *StartCamera(camera).*
6. S'entra en un bucle per poder obtenir les imatges del dispositiu de manera iterativa.
7. Dins el bucle, es crida al mètode *GetCameraDepthRGB32(camera, imatge)* per formar la imatge provinent del component de Kinect, sensor de profunditat.
8. Es converteix la imatge *NUIImage* en una imatge EmguCV.

9. Es processa la imatge EmguCV per convertir-la en un mapa binari amb el mètode *Thresholding(minValueRGB, maxValueRGB).*

10. Es crida al mètode *trobarPunts(fotograma)* de la classe *ControladorMoviment* i s'obtenen els tres punts corresponents a l'articulació.

11. El mètode *iniciarSessió(boto)* retorna un valor booleà depenen si es pot iniciar la sessió. La sessió es pot iniciar si els tres punts estan calibrats. En aquest cas s'activa el temporitzador corresponen al temps establert per la teràpia. Un cop dins del condicional, es procedeix a realitzar el càlcul de l'angle i emmagatzemar l'angle si aquest és el màxim assolit amb els mètodes *calcularAngle()* i *guardarAnglesMaxims()*. A més, s'activen les barres de progrés que serveixen com a guia a l'usuari del moviment que està realitzant i es mostra una etiqueta per informar que la sessió ha començat amb els mètodes privats *barresProgres() i grafismeSessioIniciada()*. Si la sessió encara no s'ha iniciat, es mostra una etiqueta informativa amb el mètode *grafismeSessioPreparada()* on s'indica a l'usuari que ha de fer per començar la sessió.

12. Es crida als mètodes privats *grafisme3Punts(puntA, puntB, puntC), grafismeEtiquetes(), grafismeAreaTreball()*. El primer genera un cercle de color vermell per a cada punt trobat anteriorment amb el mètode *trobarPunts(fotograma)* i els uneix amb una línia. El segon mètode actualitza les etiquetes que contenen informació referen al número de repeticions que s'ha realitzat, l'angle que s'està assolint, el numero de repeticions que s'ha de realitzar, els angles que s'ha d'assolir i un text amb ajuda.

13. Un altre condicional comprova si la sessió ha finalitzat i s'ha guardat amb els mètodes *getSessioAcabada() i getSessioGuardada()*. En aquest cas, s'atura el bucle amb la assignació *enFuncionament = false*, es guarda la sessió amb el mètode *guardarSessio()* i es tanca la sessió amb el mètode *tancarSessio()*.

14. Com a últim procediment, es crida al esdeveniment *OnClosed().* Aquest torna a posicionar la inclinació del dispositiu en el centre, desactiva el led i atura la càmera Kinect amb el mètode *StopCamera(camera).*

6.4.5. Implementació Realitzar sessió de rehabilitació amb webcam.

La implementació Realitzar sessió de rehabilitació amb webcam té com a funció obtenir les dades necessàries d'una imatge de la webcam per poder realitzar els càlculs adients durant una sessió de rehabilitació.

Les classes utilitzades en aquesta implementació són:

- Les classes *FormKinect i FormPrincipal* de la capa Presentació.
- Les classes *ControladorSessio, ControladorReconeixementMoviment i ControladorMoviment* de la capa Aplicació.
- La classe *CLNUIDevice i NUIImage.*

El funcionament d'aquesta implementació és molt similar a la implementació Realitzar sessió de rehabilitació amb Kinect. El mètode privat que conté el codi responsable d'utilitzar la webcam i realitzar la sessió de rehabilitació és *iniciarWebcam()*. El procediment d'aquest mètode és:

1. S'obtenen els fotogrames provinents de la càmera web amb el mètode *QueryFrame()* de la llibreria EmguCV.
2. Es realitzen els punts del 9 al 13 ambdós inclosos de la implementació Realitzar sessió de rehabilitació amb Kinect.
3. Com a últim procediment, es crida a l'esdeveniment *OnClosed().* Aquest atura el funcionament de la càmera web amb el mètode *Dispose()*.

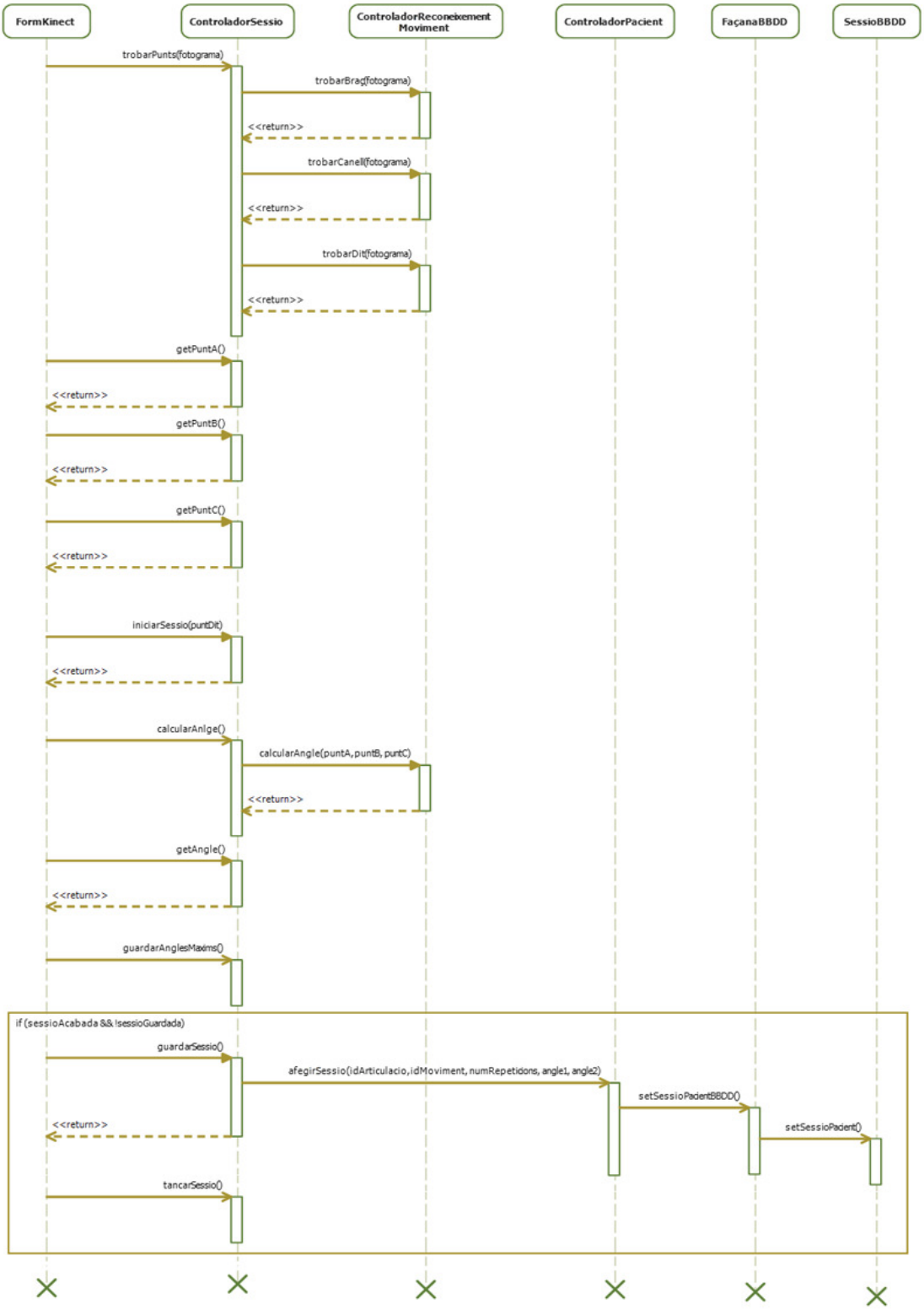

Fig. 6.7. Diagrama de seqüència del cas d'ús Realitzar sessió de rehabilitació.

6.4.6. Implementació per l'obtenció dels punts de la imatge.

Els mètodes que realitzen la tasca de localitzar els punts corresponents a l'articulació del cos humà estan en la classe *ControladorReconeixementMoviment* i s'anomenen *trobarBras(imatge), trobarCanell(imatge) i trobarDit(imatge).*

Aquests tres mètodes, com els seus noms indiquen, són tres algorismes específics per trobar aquestes tres parts del cos.

El mètode *trobarBras(imatge)* (en el nom del mètode es substitueix el caràcter *ç* per la *s* per evitar problemes amb la codificació) rep com a paràmetre una imatge EmguCV corresponen a un fotograma de Kinect o webcam. El mètode torna un objecte de la classe Point corresponen al punt de la imatge. Aquest Point té dos atributs corresponents a les coordenades *X* i *Y*.

Aquest mètode realitza dos recorreguts per la matriu de la imatge. El primer recorregut tracta de localitzar el braç. En general, el braç sempre es troba situat en la part inferior de la imatge. Però en ocasions, quan s'utilitza la càmera web i el braç, o bé no esta ben il·luminat o el pacient utilitza una peça de roba fosca, el principi del braç no queda situat en la part més baixa de la imatge. Per això es realitza un recorregut des de la part inferior de la imatge direcció amunt fins a trobar una posició de la matriu amb valor.

El segon recorregut, s'encarrega de calcular l'amplada del braç. Per obtenir aquesta dada, es realitza un recorregut en horitzontal a l'alçada on, el recorregut anterior havia localitzat el braç. El comptador *brasAmplada* es va incrementant en aquells píxels que contenen valor. El valor d'aquest comptador correspon a l'amplada del braç en píxels.

El mètode *trobarBras(imatge)* retorna les coordenades *(X, Y)* on *X* són els píxels on no hi ha braç més la meitat de l'amplada del braç i *Y* són els píxels corresponents a la posició vertical del braç.

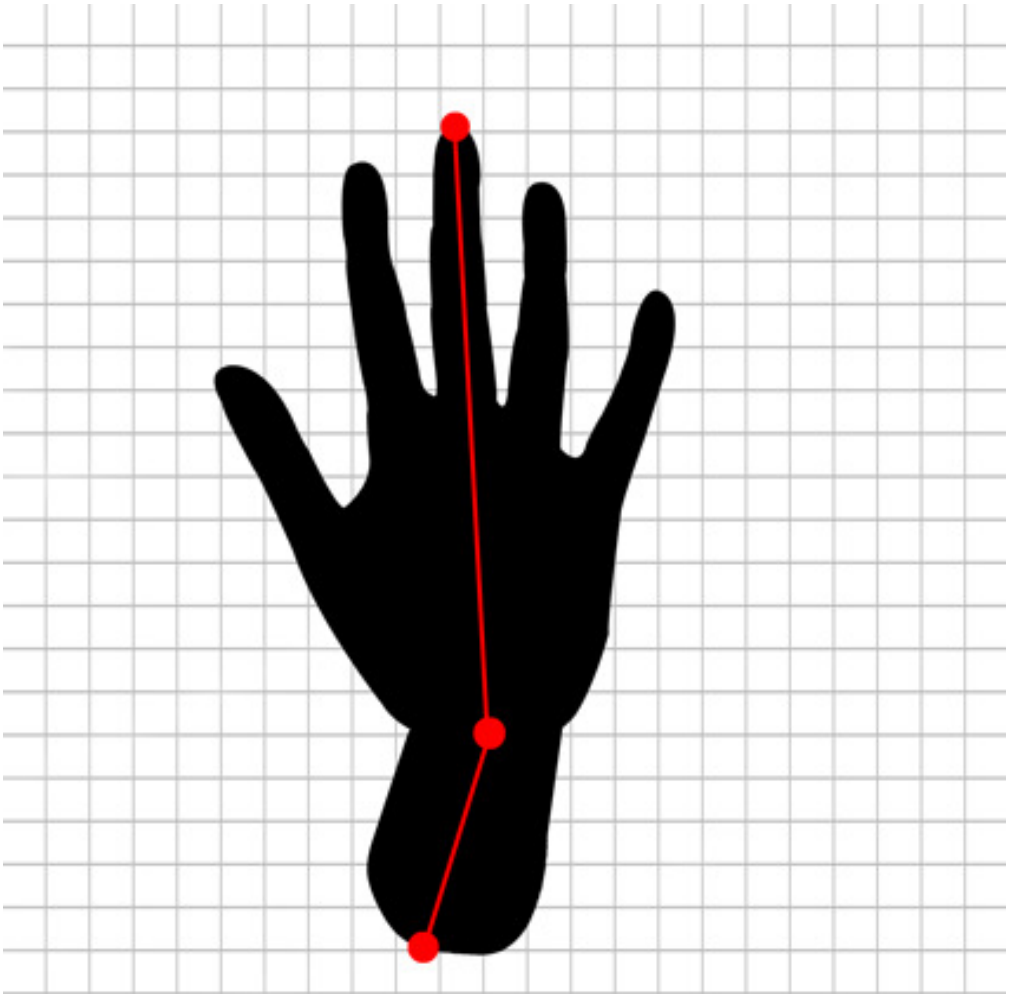

Fig. 6.8. Matriu d'una imatge amb els punts de l'articulació.

El mètode *trobarCanell(imatge)* rep com a paràmetre una imatge EmguCV corresponen a un fotograma de Kinect o webcam. El mètode torna un objecte de la classe Point corresponen al punt de la imatge. Aquest Point té dos atributs corresponents a les coordenades *X* i *Y*.

Aquest mètode realitza un tercer recorregut per tal de localitzar el canell. Aquest recorregut va des de la posició inferior del braç localitzada anteriorment direcció amunt. A cada iteració es consulta si l'amplada del braç ha augmentat prou com per poder considerar que és part del canell.

Com a objecte retornat, s'envia el punt amb les coordenades *(X, Y)* corresponents als píxels on es troba el canell més la meitat dels píxels corresponents a la seva amplada i l'alçada en vertical.

El mètode *trobarDit(imatge)* rep com a paràmetre una imatge EmguCV corresponen a un fotograma de Kinect o webcam. El mètode torna un objecte de la classe Point corresponen al punt de la imatge. Aquest Point té dos atributs corresponents a les coordenades x i y.

Aquest mètode realitza un sol recorregut. Es tracta de trobar el punt més elevat de la imatge amb valor. Per localitzar aquest punt és necessari fer un recorregut vertical per la matriu de la imatge en direcció descendent fins a trobar un píxel amb valor.

6.4.7. Implementació per iniciar una sessió.

Per tal d'iniciar una sessió de rehabilitació amb Kinect no es necessari utilitzar cap altre dispositiu apart de Kinect o la càmera web, com per exemple el ratolí o el teclat. D'aquesta manera s'aconsegueixen dos objectius:

- L'usuari pot iniciar la sessió, situat a certa distància de l'ordinador.
- L'aplicació reconeix l'articulació i la correcta posició del pacient.

Per iniciar la sessió cal invocar el mètode *iniciarSessio(botó)* de la classe *ControladorSessio*. Aquest mètode rep com a paràmetres un objecte de la classe Point corresponen a la situació d'un botó en la imatge en forma de cercle vermell. El mètode retorna un booleà.

El primer condicional del mètode comprova la variable local booleana *iniciada* de la classe *ControladorSessió*. En aquest afirmatiu, no cal realitzar cap control d'inici de sessió per què la sessió ja ha estat iniciada en alguna invocació del mètode anteriorment.

A continuació, hi ha tres condicionals més. Cadascun d'ells realitza una comprovació. El primer verifica si el punt corresponen al dit està situat en la mateixa posició que el botó que arriba al mètode per paràmetre.

El següent condicional comprova si els punts estan ordenats verticalment, és a dir, verifica si el punt corresponen al braç esta per sota del punt corresponen al canell i el punt corresponent al dit està per sobre del punt corresponent al canell.

L'altre condicional comprova si els punts estan alineats verticalment. Perquè els tres punts estiguin correctament alineats, han d'estar dins un marge de 20 píxels per la dreta i 20 píxels per l'esquerra respecte a la posició del botó que arriba al mètode per paràmetre.

Quan es produeixen aquestes tres situacions s'activa el temporitzador corresponen a la teràpia de rehabilitació. A més, es retorna un *true*.

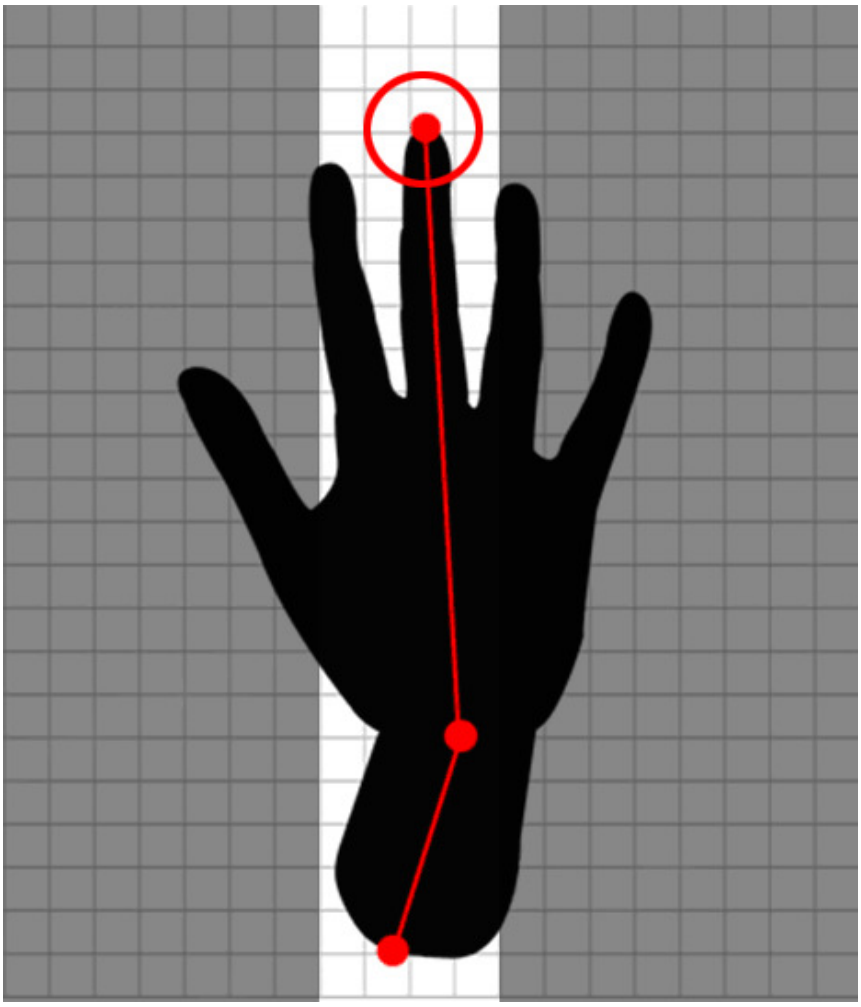

Fig. 6.9. Col·locació dels punts per tal d'iniciar una sessió.

6.4.8. Implementació per calcular l'angle.

Per calcular l'angle que forma una articulació cal utilitzar el mètode *calcularAngle(puntA, puntB, puntC)* de la classe *ControladorReconeixementMoviment*. Aquest mètode rep per paràmetre la situació dels tres punts que formen l'articulació i retorna un *double* corresponen a l'angle en graus.

Com a primera operació, cal obtenir les distàncies entre els tres punts per poder formar un triangle obtusangle. Aquestes tres distàncies són els tres catets del triangle. Per calcular aquests catets es formes tres triangles rectangles i es calculen les hipotenuses amb el teorema de Pitàgores:

$$a^2 = b^2 + c^2 \qquad a = \sqrt{b^2 + c^2} \tag{6.1}$$

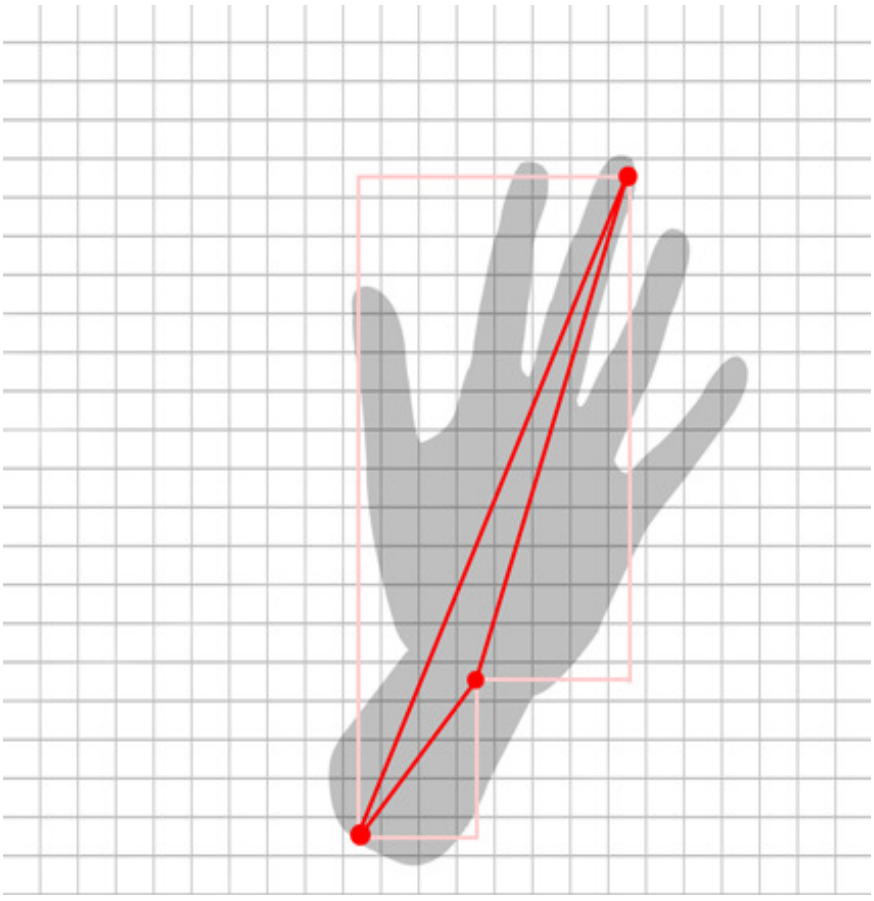

Fig. 6.10. Càlcul de les hipotenuses que formen el triangle.

La variable *signe* determina si l'angle format té signe positiu o negatiu. Si la posició horitzontal del punt que correspon al dit està situat més a la dreta que el punt corresponent al canell, la variable *signe* canvia a valor negatiu (-1).

Amb les mides dels tres catets del triangle obtusangle es pot aplicar el teorema de cosinus:

$$a^2 = b^2 + c^2 - 2 \times b \times c \times \cos\alpha \qquad \alpha = ar\cos\frac{b^2 + c^2 - a^2}{2 \times b \times c} \tag{6.2}$$

El mètode retorna la resta entre 180º i l'angle obtingut amb el teorema del cosinus.

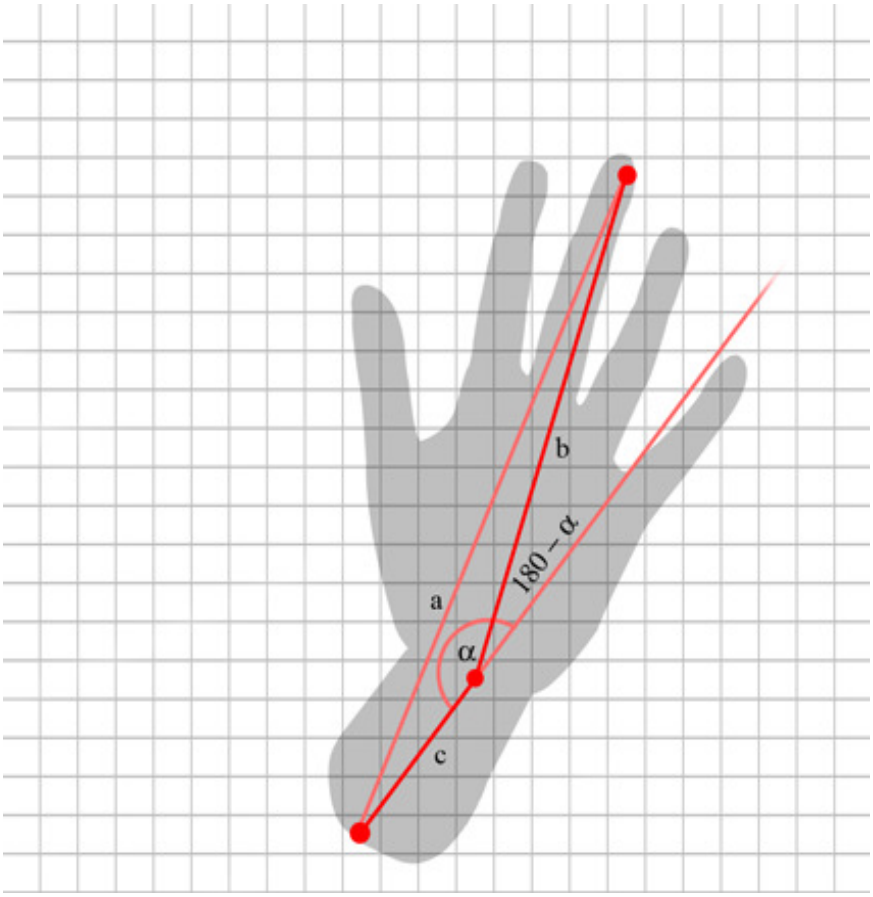

Fig. 6.11. Càlcul de l'angle pel teorema del cosinus.

7. Prova significativa.

7.1. Log In.

Només iniciar l'aplicació, es demana a l'usuari que introdueixi el seu codi de CIP. Poden produir-se quatre situacions:

- El format del codi CIP no és correcte.
- L'usuari no existeix en la base de dades.
- L'usuari no té cap teràpia de rehabilitació a realitzar.
- L'usuari té alguna teràpia de rehabilitació a realitzar. En aquest últim cas, es carreguen totes les dades corresponents a l'usuari.

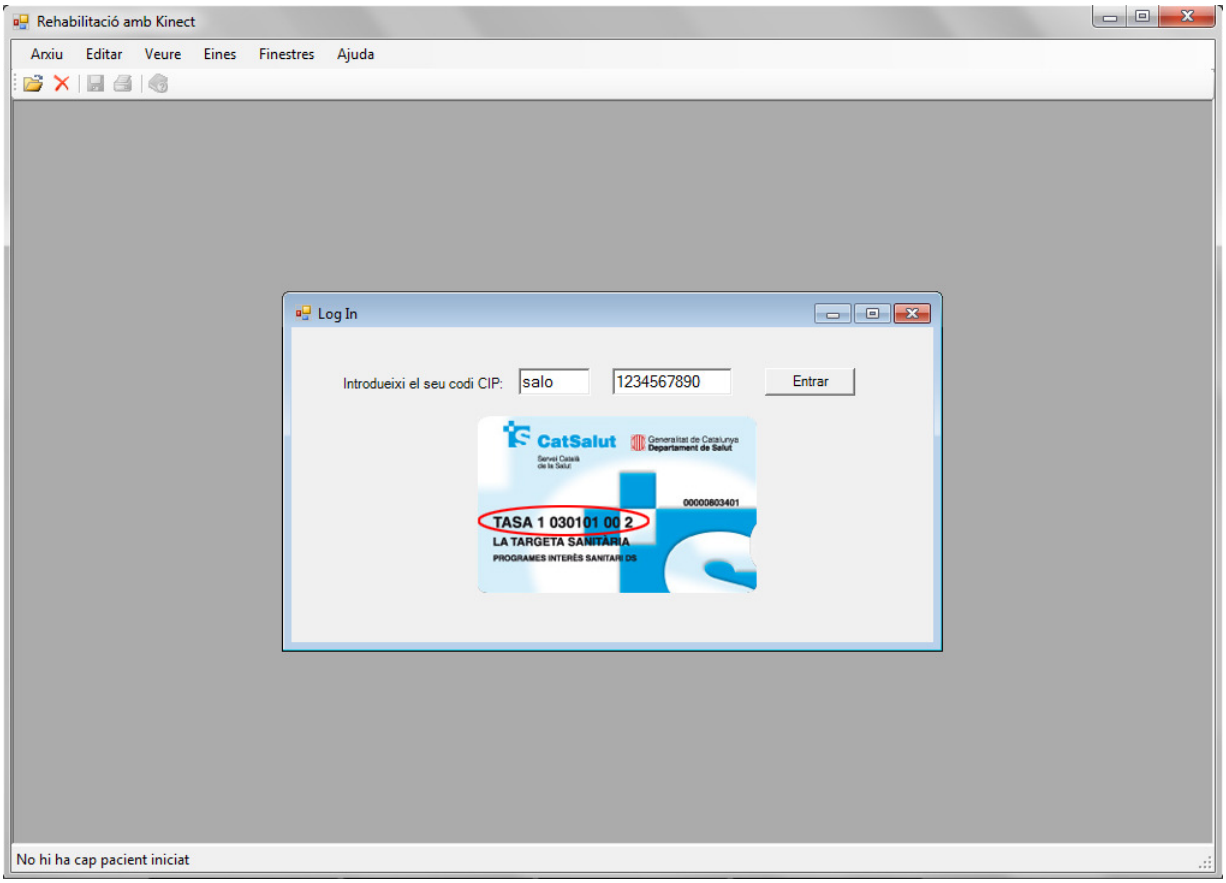

Fig. 7.1. Vista Log In.

7.2. Selector de teràpies.

L'usuari pot conèixer, a partir d'ara, quin pacient està iniciat. Dins la barra d'estat (part inferior de la finestra principal) apareix el nom i cognoms del pacient.

Un cop inicialitzat l'usuari, el selector de teràpies permet escollir la teràpia, el moviment i el dispositiu per realitzar una sessió de rehabilitació.

Des d'aquest selector, també es pot consultar quina és la progressió de la seva rehabilitació.

L'usuari ha de triar una teràpia i un moviment si vol veure l'evolució. Si vol realitzar una nova sessió, també ha de triar un dispositiu.

Si es tanca aquesta finestra i a continuació es vol tornar a obrir, es pot tornar a visualitzar des del menú *Finestres* > *Selector de Teràpies*.

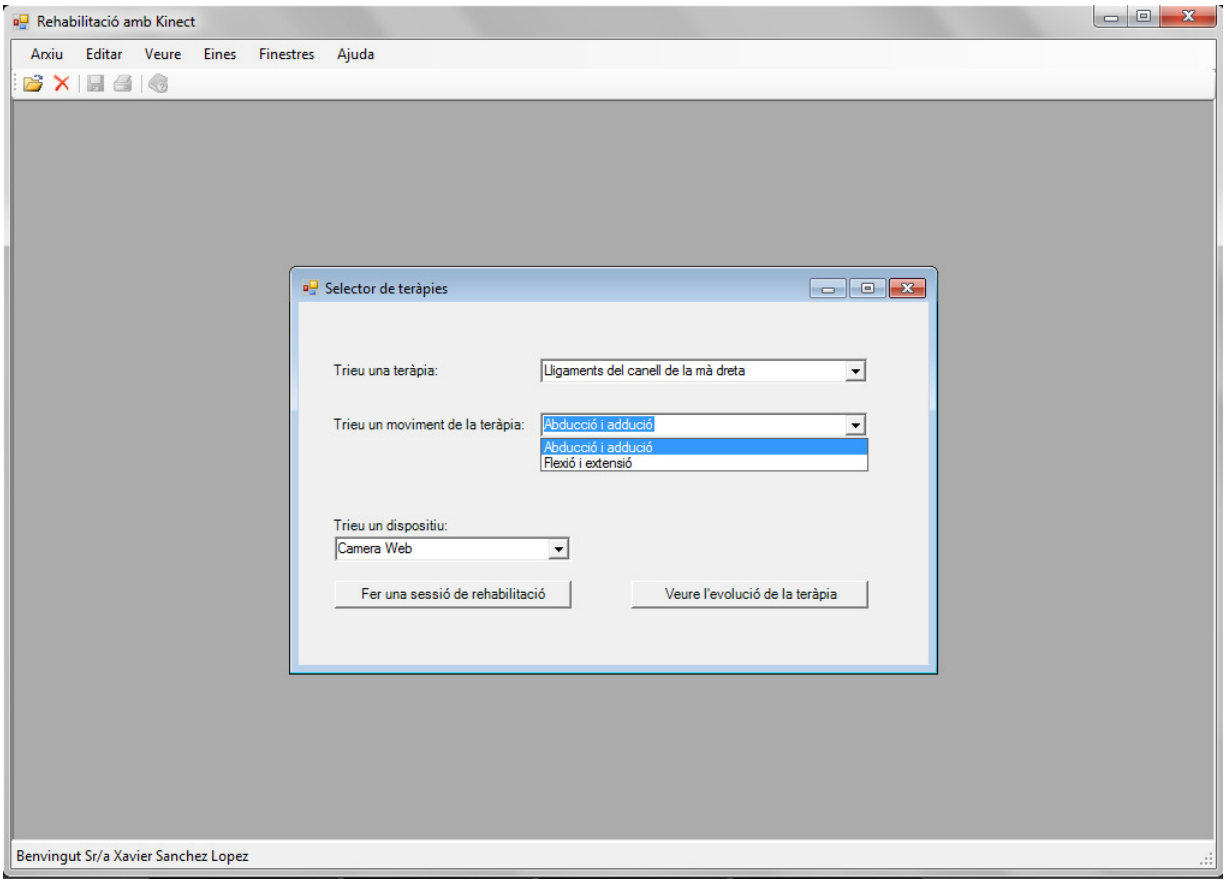

Fig. 7.2. Vista Selector de teràpia.

7.3. Evolució de la teràpia.

L'evolució de la teràpia mostra de manera gràfica i senzilla quins són els resultats de les sessions de rehabilitació realitzades.

La gràfica mostra dues línies de diferents colors que corresponen als diferents moviments de l'articulació.

L'eix de les Y corresponen a l'angle en graus del moviment.

L'eix de les X mostra el dia i l'hora que es va realitzar la sessió de rehabilitació.

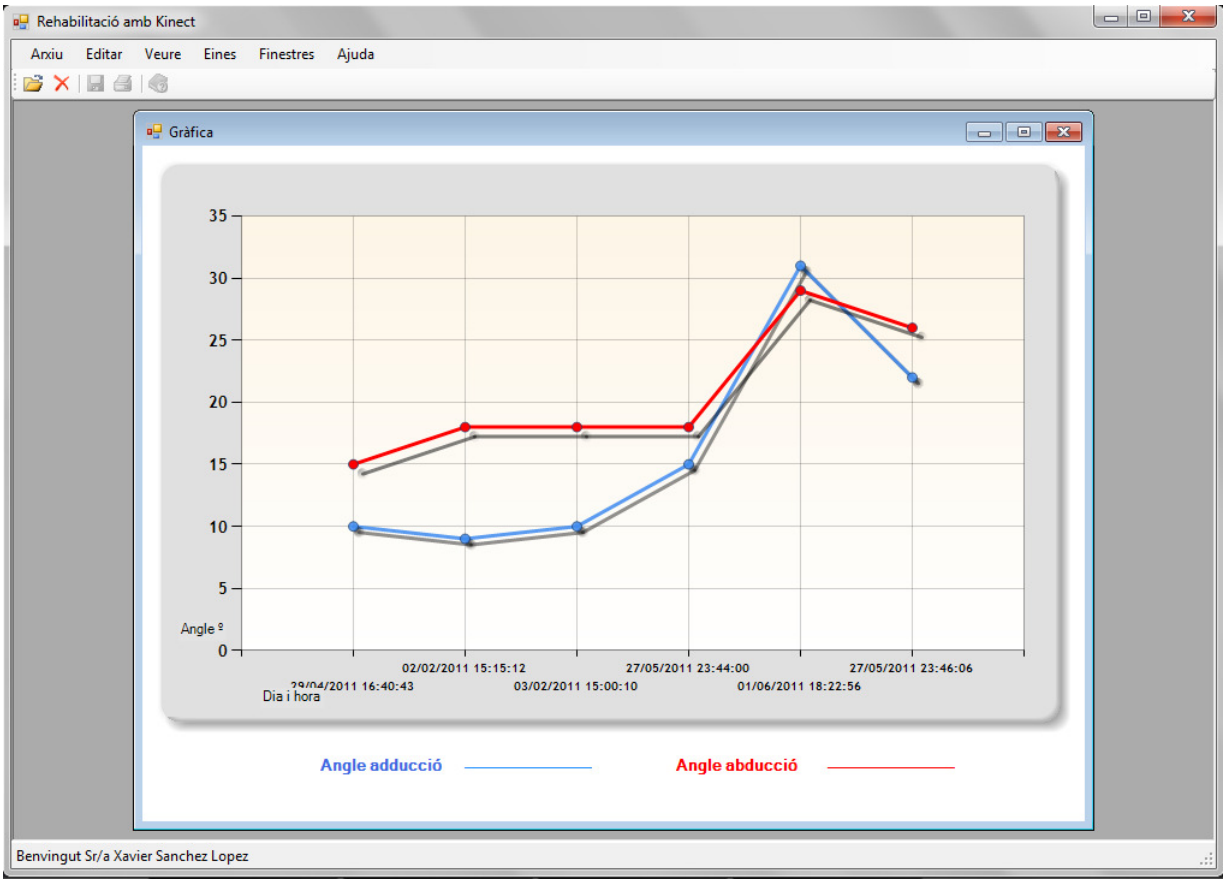

Fig. 7.3. Vista Evolució de la teràpia.

7.4. Sessió de rehabilitació amb Kinect.

Abans de realitzar una sessió de rehabilitació, l'usuari pot regular la inclinació del dispositiu per tal de realitzar la sessió dins l'àrea de treball representada amb un quadrat de color gris.

A continuació, l'aplicació ha de calibrar i reconèixer l'articulació per poder iniciar la sessió.

La sessió s'inicia quan:

- Els tres punts estan ordenats verticalment.
- Els tres punts estan alineats verticalment dins una franja central sota el punt vermell.
- El punt corresponen al dit està en la posició del cercle vermell.

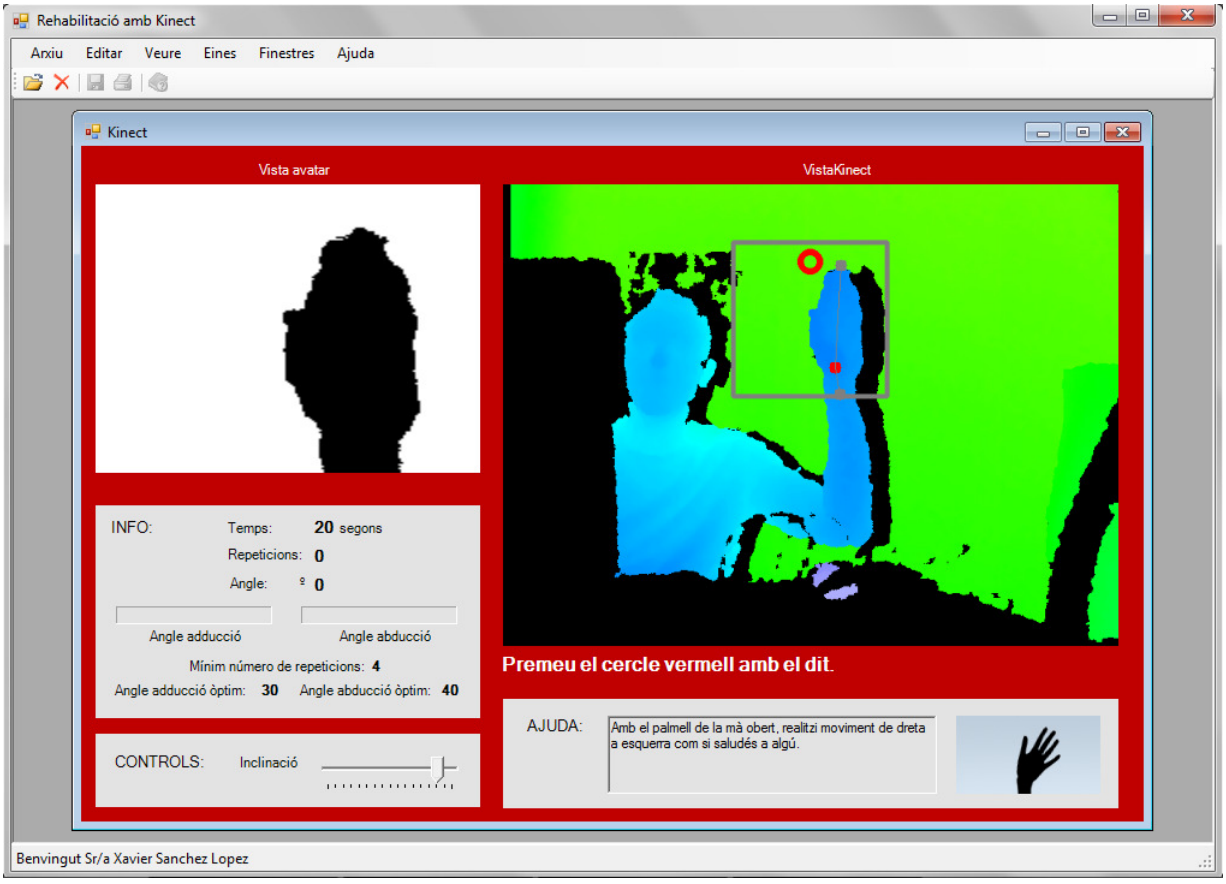

Fig. 7.4. Vista d'una sessió de rehabilitació amb Kinect per iniciar.

En iniciar la sessió, tal i com s'observa en la fig.7.5, l'usuari obté la següent informació:

- Duració restant de la sessió.
- Número de repeticions realitzades.
- Angle que s'està realitzant representat numèricament i en forma de barres de progrés.
- Número de repeticions mínimes que ha de realitzar per considerar la sessió vàlida.
- Angles òptims que pot descriure l'articulació.
- Una ajuda (com s'ha de realitzar el moviment).

En finalitzar el temps de sessió s'informa a l'usuari si la sessió s'ha realitzat correctament depenen del número de repeticions realitzades. En cas afirmatiu, les dades de la sessió es persisteixen a la base de dades.

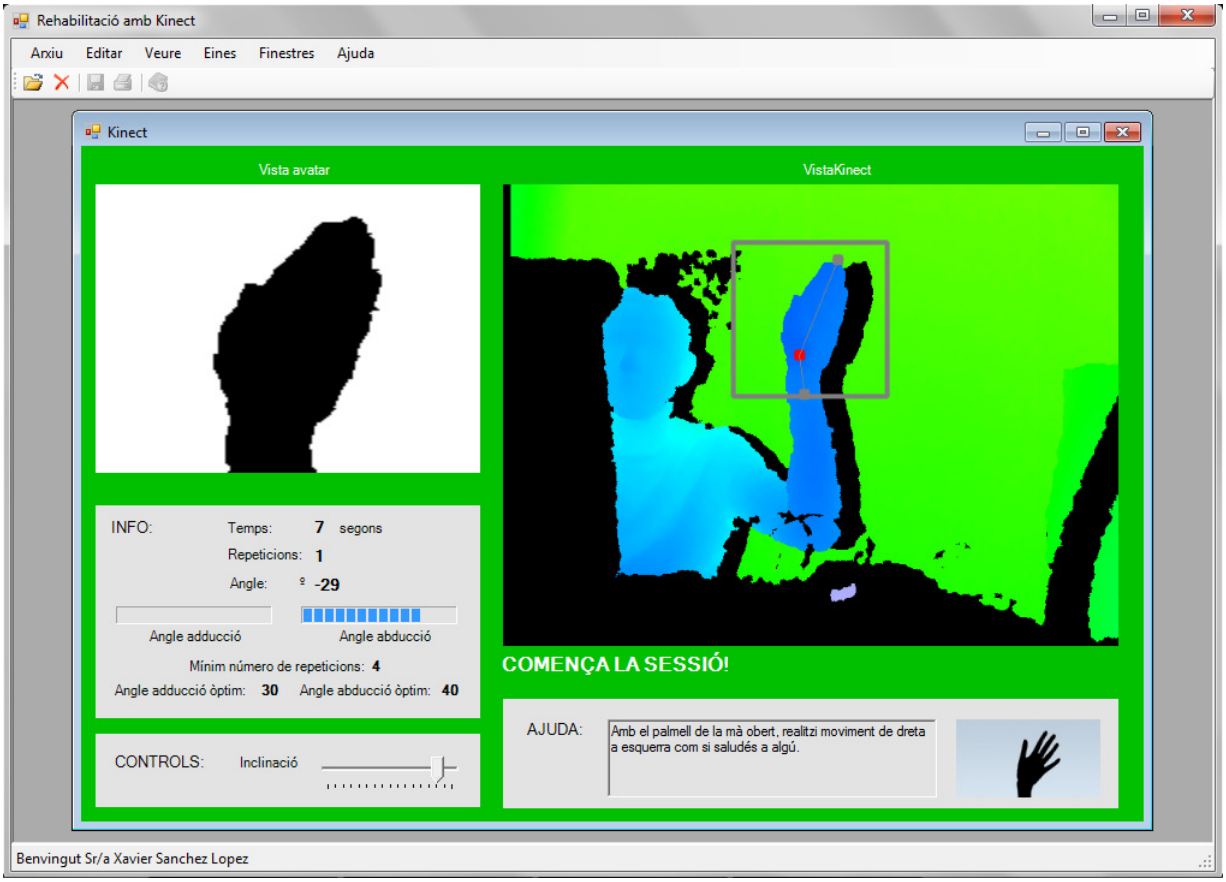

Fig. 7.5. Vista d'una sessió iniciada de rehabilitació amb Kinect.

7.5. Sessió de rehabilitació amb webcam.

Abans de realitzar una sessió de rehabilitació, l'usuari ha d'ajustar el llindar de llum admesa depenen de l'entorn lluminós on es trobi. L'usuari ha d'aconseguir veure únicament la seva mà, en la finestra avatar.

Els processos següent (iniciar la sessió i realitzar la sessió) són idèntics a la rehabilitació amb Kinect, com es mostra el la fig. 7.7.

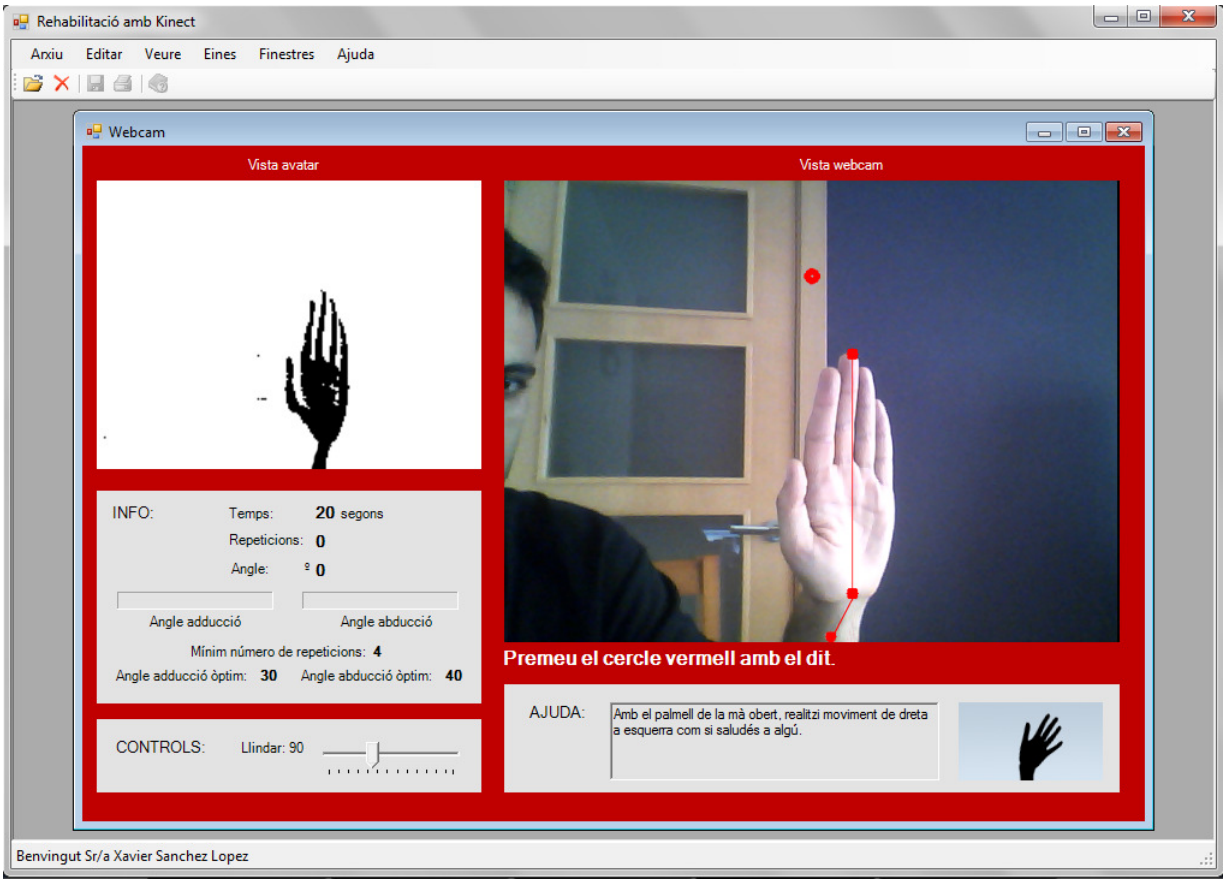

Fig. 7.6. Vista d'una sessió de rehabilitació amb webcam per iniciar.

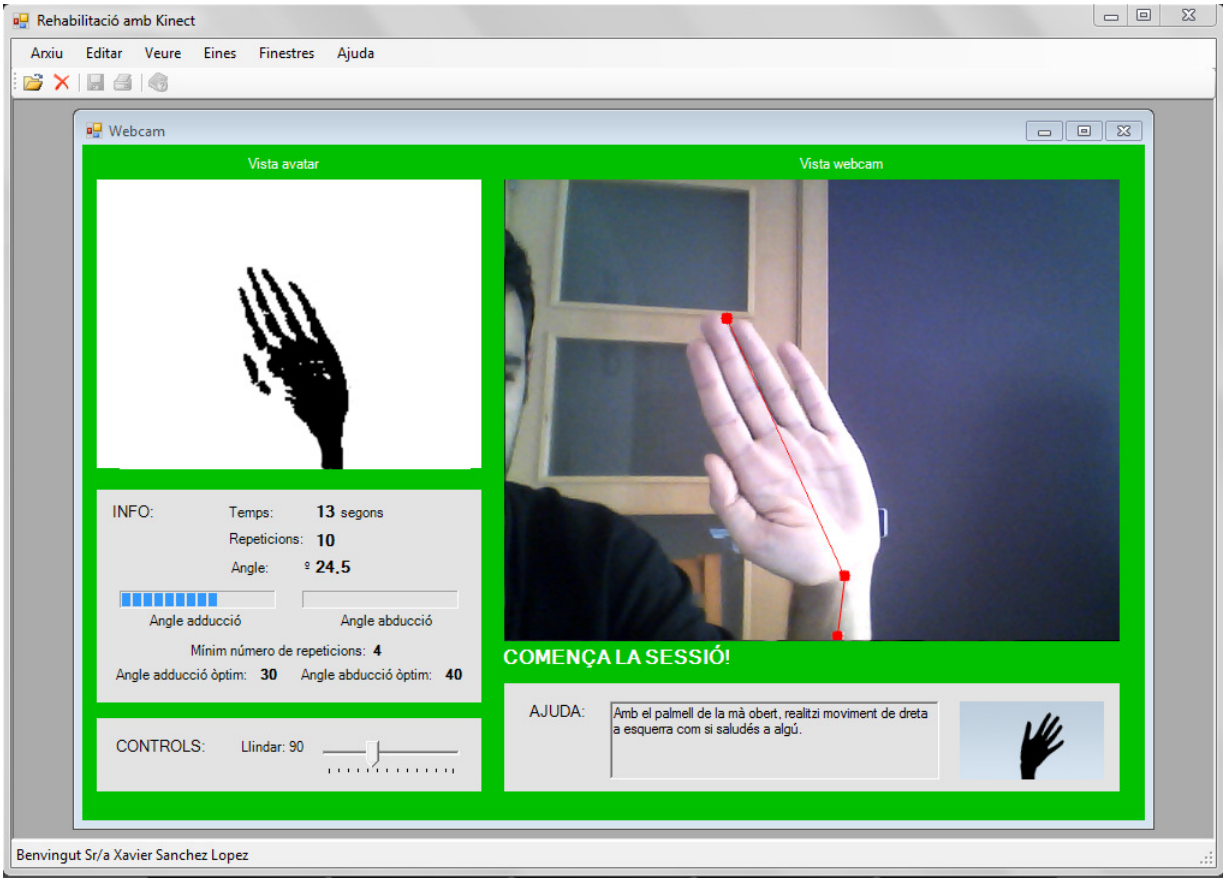

Fig. 7.7. Vista d'una sessió iniciada de rehabilitació amb webcam.

8. Problemes trobats.

Els problemes trobats durant la realització del projecte es van produir, principalment, durant el procés de desenvolupament.

El primer problema va sorgir en el procés d'estudi previ dels drivers de Kinect. Alguns d'ells són realment complicats d'instal·lar i configurar.

D'altra banda, la llibreria de Kinect CLNUIPlatform, crea un objecte / imatge bitmap de la classe *System.Windows.Interop.InteropBitmap* totalment incompatible amb els objectes de la classe *System.Drawing.Bitmap o Emgu.CV.Image*. Degut a aquesta falta de compatibilitat, el procés per obtenir una imatge per poder ser processada, va ser bastant laboriós. No es va trobar cap mètode o procediment capaç de convertir els objectes de les diferents classes entre si. La solució va passar per formar la imatge necessària des de l'espai de memòria que utilitza la llibreria de Kinect.

En ocasions, l'aplicació havia de realitzar una quantitat de càlculs per segon molt elevada. Per cada segon, l'aplicació construïa 30 imatges, aplicava els efectes digitals d'imatge adients (thresholding, re-dimensionar, voltar horitzontalment...), cercava els tres punts necessaris entre 307200 posicions de la matriu de la imatge (640 x 480 píxels), afegia grafisme corresponen als punts i realitzava els càlculs matemàtics per obtenir l'angle de l'articulació. Calia optimitzar el procés. La solució ha passat per reduir la freqüència de fotogrames per segon de 30 a 15, reduir la resolució de 640 x 480 a 213 x 160 píxels amb les imatges de la càmera web i establir una àrea de treball de 100 x 100 píxels en forma de quadrat de color gris amb les imatges de Kinect per tal d'alleugerir la quantitat de càlculs per segon.

En canvi, d'altres possibles dificultats que s'havien considerat, finalment, es van superar amb èxit. La inexperiència amb el llenguatge de programació C# i la utilització del framework de processament d'imatge EmguCV no han suposat cap mena de problema afegit.

9. Conclusions.

Un cop finalitzat el projecte, la conclusió principal és que s'han implementat tots els requeriments inicials. Un pacient pot realitzar una sessió de rehabilitació d'un moviment d'una articulació amb el dispositiu Kinect.

També es poden extreure d'altres conclusions de cada procés de la implementació. Per exemple, les decisions presses durant el procés dedicat a l'estudi previ han estat de vital importància per l'èxit del projecte. Escollir un driver, un processador d'imatge o un entorn de desenvolupament equivocat hauria suposat un endarreriment substancial del desenvolupament.

En l'etapa final, també ha estat molt important els *testing* que s'ha realitzat amb persones alienes al projecte que aportaven millores d'usabilitat i accessibilitat en l'aplicació.

La realització del projecte ha representat una elaboració de gran dedicació recompensada pels coneixements obtinguts. L'aprenentatge d'un altre llenguatge de programació, la utilització del processament d'imatges, l'ús de temporitzadors...

Realitzar individualment un projecte informàtic de principi a fi, aporta molts coneixements.

10. Futures ampliacions.

Amb RaK es poden realitzar sessions de rehabilitació del moviment d'abducció i adducció del canell de la mà dreta. Com a futures ampliacions es podrien implementar més algorismes de cerca de punts del canell quan es realitza altres moviments, com per exemple, flexió i extensió i quan es realitza amb la mà esquerra. A més, es podrien implementar altres algorismes per a més articulacions del cos humà.

RaK és una aplicació de sobretaula. Per millorar la distribució de l'aplicació es podria desenvolupar una versió web.

D'altra banda, la informació que rep el pacient sobre la seva rehabilitació podria ser ampliada. A part de la gràfica lineal, podria mostrar-se un informe amb més detall sobre una sessió de rehabilitació en concret. Es podria informar sobre quantes repeticions es van realitzar en la sessió, quins van ser els angles màxims aconseguits per a cada moviment...

11. Referències.

[1] www.sermef.es, Sociedad Española de rehabilitación y medicina física, 2011.

[2] www.sld.cu/sitios/rehabilitacion/temas.php?idv=1029, Ejercicio terapéutico. Generalidades, Dra. Solangel Hernández Tápanes, 2009.

[3] www.play4health.com, Play For Health, Fundació Illes Balears Innovació Tecnològica, 22 Novembre 2010.

[4] www.guttmann.com/%5Cbackoffice%5Chtml%5Cdocs%5Csobreruedas76.pdf, Previrnec, Institut Guttmann, 2011.

[5] www.telefonica.com/es/innovation/html/desa_nuevos_negocios.shtml, Desarrollo de nuevos negocios E-Health, Telefónica, 2011.

[6] saladeprensa.telefonica.com/documentos/DossierTelefonica_0.pdf, E-Health, Telefónica, 2011.

[7] www.xbox.com/es-ES/kinect, Kinect España, Microsoft, 2011.

[8] www.ifixit.com/Teardown/Microsoft-Kinect-Teardown/4066/1, Microsoft Kinect Teardown, 4 novembre 2010.

[9] openkinect.org/wiki/Main_Page, OpenKinect, 28 abril 2011.

[10] codelaboratories.com/nui, CL NUI Platform Kinect preview, 6 novembre 2010.

[11] www.primesense.com, PrimeSense Natural Interaction, 2010.

[12] opencv.willowgarage.com/wiki, OpenCV, 16 juny 2011.

[13] www.emgu.com/wiki/index.php/Main_Page, EmguCV, 16 abril 2011.

[14] www.openni.org, OpenNI, 18 abril 2011.

[15] java.sun.com/javase/technologies/desktop/media/jai/ Java Advanced Imaging (JAI) API, Oracle, 2010.

[16] www.mysql.com, MySQL Oracle, 2010.

[17] msdn.microsoft.com/en-us/magazine/dd569763.aspx, Microsoft Chart Controls, Abril 2009.

Escola Universitària
Politècnica de Mataró

Centre adscrit a:

UNIVERSITAT POLITÈCNICA
DE CATALUNYA

Enginyeria Tècnica en Informàtica de Gestió

REHABILITACIÓ AMB KINECT

Estudi econòmic

XAVIER SÀNCHEZ LÓPEZ
PONENT: MONTSERRAT RABASSA

PRIMAVERA 2011

TecnoCampus
Mataró-Maresme

Índex.

1. Pressupost econòmic.

A continuació es detallen els costos econòmics que s'han generat en el desenvolupament de l'aplicació. L'estudi econòmic ha de considerar:

- Cost del personal involucrat en el projecte.
- Cost d'amortització del material i software.
- Despeses indirectes.
- Despeses imputables al projecte.

1.1. Cost de recursos humans.

Durant el procés de desenvolupament del projecte, ha estat necessari el treball de diferents perfils professional per a cada etapa del procés:

- Personal especialitzat en anàlisis informàtic (anàlisis de requeriments, disseny base de dades…).
- Personal especialitzat en desenvolupament informàtic (implementació del codi).
- Personal especialitzat en tasques administratives (redacció de la documentació, càlcul del pressupost econòmic).

Concepte	Preu / hora	Hores	TOTAL
Analista informàtic (Enginyer sènior)	50 euros	50 hores	2500 euros
Programador informàtic (Enginyer sènior)	50 euros	250 hores	12500 euros
Administratiu	30 euros	50 hores	1500 euros
TOTAL COST RECURSOS HUMANS			**16500 euros**

1.2. Cost d'amortització del material i software.

Detall del cost d'amortització de les eines necessàries pel desenvolupament en material físic i programari informàtic:

Concepte	Preu	Hores	TOTAL
Ordinador portàtil	750 euros	350 hores	250 euros
Kinect	150 euros	Ús exclusiu	150 euros
Visual Studio 2010	1499 euros	300 hores	450 euros
Power Designer 15	1078 euros	50 hores	50 euros
Microsoft Office	699 euros	50 hores	35 euros
TOTAL COST AMORTITZACIONS			**935 euros**

1.3. Despeses indirectes.

Les despeses indirectes són les derivades de l'ús de les instal·lacions i dels diversos subministraments que s'han utilitzat durant el temps de desenvolupament del projecte. S'ha considerat que la utilització d'aquests recursos durant 350 hores equival a dos mesos de treball a jornada completa:

Concepte	**Preu / mes**	**Hores**	**TOTAL**
Lloguer de les instal·lacions	700 euros	350 hores	1400 euros
Subministrament elèctric	40 euros	350 hores	80 euros
Subministrament Internet	30 euros	350 hores	60 euros
TOTAL DESPESES INDIRECTES			**1540 euros**

1.4. Despeses imputables al projecte.

Es consideren despeses imputables al projecte el cost que generen les visites al client realitzades durant el procés d'anàlisis de requeriments i presentació del producte.

Concepte	**Preu unitari**	**Quantitat**	**TOTAL**
Desplaçaments	10 euros	8	80 euros
Dietes	10 euros	4	40 euros
TOTAL DESPESES IMPUTABLES			**120 euros**

1.5. Cost total.

Concepte	**Preu**
Recursos humans	16500 euros
Amortitzacions	935 euros
Despeses indirectes	1540 euros
Imputables al projecte	120 euros
TOTAL PROJECTE	**19095 euros**

Escola Universitària
Politècnica de Mataró

Centre adscrit a:

UNIVERSITAT POLITÈCNICA
DE CATALUNYA

Enginyeria Tècnica en Informàtica de Gestió

REHABILITACIÓ AMB KINECT

Annexos

XAVIER SÀNCHEZ LÓPEZ
PONENT: MONTSERRAT RABASSA

PRIMAVERA 2011

TecnoCampus
Mataró-Maresme

Índex.

Annex I. Codi trobar punts.

```
      public Point trobarDit(Image<Gray, Byte> fotograma)
        {
            Gray valor;

           for (int i = 0; i < fotograma.Height / 2; i++)
  //fotograma.Height / 2 limita la cerca a la mitad superior de la imatge.
            {
                for (int j = 0; j < fotograma.Width; j++)
                {
                    valor = fotograma[i, j];
                    if (valor.Intensity == 0)
                    {
                        puntC.X = j;
                        puntC.Y = i;
                        return puntC;
                    }
                }
            }
            //En aquest punt del mètode, no s'ha localitzat el punt.
            puntC.X = 0;
            puntC.Y = 0;
            return puntC;
        }
```

```
public Point trobarBras(Image<Gray, Byte> fotograma)
        {
            if (puntC.X == 0 && puntC.Y == 0)
      // Si no s'ha trobat el dit (puntC), no es busca el braç.
            {
                puntA.X = 0;
                puntA.Y = 0;
                return puntA;
            }
            else
            {
                Gray valor;
                brasPosicioY = 0;
                brasAmplada = 0;
                int brasPosicioX = 0;
                bool brasSituatX = false, brasSituatY = false;
```

```
            //Recorregut cap amunt per trobar l'inici del braç.
            for (int i = fotograma.Height/2 - 1; i>0 && !brasSituatY; i--)
            // Height/2 limita la cerca a la mitad inferior de la imatge.
            {
                for (int j = 0; j < fotograma.Width; j++)
                {
                    valor = fotograma[i, j];
                    if (valor.Intensity == 0)
                    {
                        brasSituatY = true;
                    }
                }
                brasPosicioY++;
            }


            //Recorregut en horitzontal a l'alçada brazoPosicioY per
            calcular l'amplada del braç.
            for (int j = 0; j < fotograma.Width; j++)
            {
                 if (fotograma.Height - 1 - brasPosicioY > 0)
                 valor = fotograma[fotograma.Height - 1 - brasPosicioY, j];

                 else valor = fotograma[0, j];

                if (valor.Intensity == 0)
                {
                    brasAmplada++;
                    brasSituatX = true;
                }

               if (!brasSituatX)
                {
                    brasPosicioX++;
                }
            }

            if (!brasSituatX || !brasSituatY) //No s'ha localitzat el punt.
            {
                puntA.X = 0;
                puntA.Y = 0;
            }
            else
            {
                puntA.X = brasPosicioX + (brasAmplada / 2);
                puntA.Y = fotograma.Height - 1 - brasPosicioY;
            }

            return puntA;
        }
    }
```

```
public Point trobarCanell(Image<Gray, Byte> fotograma)
        {
            if (puntC.X == 0 && puntC.Y == 0)
      // Si no s'ha trobat el dit (puntC), no es busca el canell
            {
                puntB.X = 0;
                puntB.Y = 0;
                return puntB;
            }
            else
            {

                Gray valor;
                int canellAmplada = 0;//, brasAmplada = 0;
                int canellPosicioX = 0;//, brasPosicioY = 0;
                bool canellSituatX = false;//, brasSituatY = false;
                double DIFERENCIA = 1.3;
             //Index de diferència de tamany per diferenciar el braç del canell.


                //Recorregut desde brasPosicioY cap amunt per trobar el canell.
                for (int i = fotograma.Height - brasPosicioY; i > 0; i--)
                {
                    for (int j = 0; j < fotograma.Width; j++)
                    {
                        valor = fotograma[i, j];
                        if (valor.Intensity == 0)
                        {
                            canellAmplada++;
                            canellSituatX = true;
                        }
                        if (!canellSituatX)
                        {
                            canellPosicioX++;
                        }

                    }
                    if (canellAmplada > brasAmplada * DIFERENCIA)
                   //La amplada supera el % establert com a part del braç.
                    {
                        puntB.X = canellPosicioX + (canellAmplada / 2);
                        puntB.Y = i;
                        return puntB;
                    }
                    canellAmplada = 0;
                }
          //En aquest punt del mètode, no s'ha localitzat el punt del canell.
                puntB.X = 0;
                puntB.Y = 0;
                return puntB;
            }
        }
```

Annex II. Codi iniciar sessió.

```
public bool iniciarSessio(Point boto, int diferenciaX, int diferenciaY)
{
	if (iniciada) return true;

	//Control punt C (dit) prem el botó.
	if (puntC.X + diferenciaX > boto.X - 5 && puntC.X + diferenciaX < boto.X
	+ 5 && puntC.Y + diferenciaY > boto.Y - 5 && puntC.Y + diferenciaY <
	boto.Y + 5) iniciada = true;
	else iniciada = false;

	// Control dels tres punts ordenats verticalment: adalt C, B i abaix A.
	if (iniciada && puntA.Y > puntB.Y && puntB.Y > puntC.Y) iniciada = true;
	else iniciada = false;

	//Control dels tres punts en linia (marge de 20 pixels).
	if (iniciada && puntA.X + diferenciaX > boto.X - 20 && puntA.X +
	diferenciaX < boto.X + 20 && puntB.X + diferenciaX > boto.X - 20 &&
	puntC.X + diferenciaX < boto.X + 20) iniciada = true;
	else iniciada = false;

	//Activa el temps de la sessió.
	if (iniciada)
	 {
		temps.Interval = movimentC.getTemps() * 1000;
		temps.Start();
		temps.Elapsed += new
		System.Timers.ElapsedEventHandler(acabarSessio);
		iniciarCronometre();
	}

	return iniciada;
}


private void iniciarCronometre()
{
	Timer crono = new Timer();
	crono.Interval = 1000;
	crono.Start();
	crono.Elapsed += new System.Timers.ElapsedEventHandler(restarSegon);
}


private void restarSegon(object source, EventArgs e)
{
	if (segons > 0) segons--;
}
```

Annex III. Codi calcular angle.

```
public double calcularAngle(Point puntA, Point puntB, Point puntC)
{
	int signe = 1;

	//Càlcul del catet a, b i c per Pitagores.
	double a;

	if (puntC.X > puntA.X)
	a = Math.Pow((Math.Pow(puntA.Y - puntC.Y, 2) + Math.Pow(puntC.X -
	puntA.X, 2)), 0.5);

	else
	a = Math.Pow((Math.Pow(puntA.Y - puntC.Y, 2) + Math.Pow(puntA.X -
	puntC.X, 2)), 0.5);

	double b;
	if (puntB.X > puntA.X)
	b = Math.Pow((Math.Pow(puntA.Y - puntB.Y, 2) + Math.Pow(puntB.X -
	puntA.X, 2)), 0.5);

	else
	b = Math.Pow((Math.Pow(puntA.Y - puntB.Y, 2) + Math.Pow(puntA.X -
	puntB.X, 2)), 0.5);

	double c;
	if (puntC.X > puntB.X)
	c = Math.Pow((Math.Pow(puntB.Y - puntC.Y, 2) + Math.Pow(puntC.X -
	puntB.X, 2)), 0.5);

	else
	{
	c = Math.Pow((Math.Pow(puntB.Y - puntC.Y, 2) + Math.Pow(puntB.X -
	puntC.X, 2)), 0.5);

	// El punt B>C, per tant, l'articulacio gira en l'altre direcció.
	signe = -1;
	}

	//Càlcul de l'angle pel teorema del cosinus.
	return (Math.Round(180 - (Math.Acos((b * b + c * c - a * a) /
	(2 * b * c)) * 180 / Math.PI), 1)) * signe ;

}
```

Annex IV. Contingut del DVD-Rom.

A continuació es detallen els arxius que s'inclouen en el suport digital:

- Documentació del projecte (memòria, estudi econòmic, annexos).
- Software necessari (Visual Studio 2010, WampServer).
- Llibreries utilitzades en el projecte (EmguCV, OpenCV, Chart, MySQL, Kinect).
- Drivers necessaris (CLNUIPlatform).
- Scripts de creació de la base de dades i inserció de les dades d'exemple.
- Solució C# per Visual Studio 2010 amb el projecte Rehabilitació amb Kinect.