laSalle

UNIVERSITAT RAMON LLULL

Escola Tècnica Superior d'Enginyeria La Salle

Treball Final de Màster

Màster Universitari en Creació, Disseny i Enginyeria Multimèdia

Rehabtimals: validació biomecànica i desenvolupament de la plataforma de software

Alumne
Adso Fernández Baena

Professor Ponent
Oscar García Pañella

ACTA DE L'EXAMEN
DEL TREBALL FI DE CARRERA

Reunit el Tribunal qualificador en el dia de la data, l'alumne

D. Adso Fernández Baena

va exposar el seu Treball de Fi de Carrera, el qual va tractar sobre el tema següent:

Rehabtimals: validació biomecànica i desenvolupament de la plataforma de software

Acabada l'exposició i contestades per part de l'alumne les objeccions formulades pels Srs. membres del tribunal, aquest valorà l'esmentat Treball amb la qualificació de

Barcelona,

VOCAL DEL TRIBUNAL VOCAL DEL TRIBUNAL

PRESIDENT DEL TRIBUNAL

ABSTRACT

Avui en dia hi ha molta gent que fa teràpia de rehabilitació física per recuperar-se d'alguna lesió. Aquestes teràpies són avorrides i repetitives, i fan perdre la motivació als pacients. Els serious games s'han aplicat amb èxit en diferents àmbits per incrementar la motivació en processos d'aprenentatge. Per altra banda, dispositius com Kinect permeten la captura de moviment, i per tant, són una tecnologia a tenir en compte en la rehabilitació basada en el moviment. Rehabtimals és una plataforma que pretén millorar la rehabilitació física utilitzant Kinect i els serious games.

S'ha creat una aplicació on el pacient realitza la recuperació usant un serious game guiat per la captura de moviment feta amb el dispositiu Kinect. Des d'una altra aplicació, el fisioterapeuta configura les sessions de treball i realitza el seguiment de les teràpies visionant sessions en vídeo, gràfiques i moviments 3D. Prèviament s'ha realitzat una validació biomecànica de les dades capturades per aquest dispositiu per donar credibilitat al moviment estimat.

ABSTRACT

Hoy en día hay muchas personas que realizan terapias de rehabilitación física para recuperarse de alguna lesión. Estas terapias son aburridas y repetitivas, y hacen perder la motivación a los pacientes. Los serious games se han aplicado con éxito en distintos campos para incrementar la motivación en procesos de aprendizaje. Por otra parte, dispositivos como Kinect permiten hacer captura de movimiento, y por lo tanto, son una tecnología ha tener en cuenta en la rehabilitación basada en movimientos. Rehabtimals es una plataforma que pretende mejorar la rehabilitación física utilizando Kinect y los serious games.

Se ha creado una aplicación donde el paciente realiza la recuperación usando un serious game guiado por la captura de movimiento hecha con el dispositivo Kinect. Desde otra aplicación, el fisioterapeuta configura las sesiones de trabajo y realiza el seguimiento de las terapias visionando las sesiones en video, gráficas y movimientos 3D. Previamente se ha realizado una validación biomecánica de los datos capturados por este dispositivo para dar credibilidad al movimiento estimado.

ABSTRACT

Today there are many people doing physical rehabilitation therapy to recover from injury. These therapies are boring and repetitive, and they lose motivation to patients. Serious games have been successfully applied in various fields to increase motivation in learning processes. Moreover, Kinect devices like, can perform motion capture, and therefore, are a technology has to take into account in movements based rehabilitation. Rehabtimals is a platform that aims to improve the physical rehabilitation using Kinect and serious games.

It has created an application where the patient performs recovery using a serious game driven by the Kinect motion capture. From another application, the therapists sets up the working sessions and tracks therapy sessions envisioning the video, 2D graphics and 3D motion graphics. Previously there has been a biomechanical validation of the data captured by this device to give credibility to the estimated movement.

RESUM

Rehabtimals és una projecte que pretén millorar les teràpies convencionals de rehabilitació física. Aquest és un treball final de màster del Màster en Creació, Disseny i Enginyeria Multimèdia (MCDEM) de La Salle – Universitat Ramon Llull. El projecte l'ha realitzat l'equip ARtiK format per: Anna Aguilar, producer; Marc Rodríguez, tècnic; Dani Arguedes, modelador i animador 3D; i Adso Fernández, l'autor d'aquest document, tècnic. Aquesta memòria descriu la implementació de la validació biomecànica, i el desenvolupament de l'aplicació fisioterapeuta i l'aplicació pacient de la plataforma.

Les teràpies convencionals de rehabilitació física estan basades en la repetició de moviments. Els fisioterapeutes han de corregir els moviments erronis als pacients per que aquests puguin realitzar els exercicis a casa. La repetició i la falta de motivació fan que els pacients no realitzin les teràpies de forma completa.

L'aparició de dispositius com Kinect fan possible la captura de moviment a un cost molt baix. La repetició de moviments és la base de les teràpies de rehabilitació física, per tant, fa evident l'ús d'aquests dispositius. Com a premissa, s'ha realitzat una validació biomecànica per donar credibilitat aquests dispositius i demostrar la seva fiabilitat. S'han comparat les dades capturades pel Kinect i per un sistema òptic passiu de captura. S'ha creat un serious game on el pacient realitza els exercicis sense necessitat d'estar un fisioterapeuta present. Aquest joc motiva i entreté al pacient mentre realitza correctament la teràpia de rehabilitació.

Per desenvolupar l'aplicació pacient s'han implementat dues aplicacions que funcionen simultàniament. Una aplicació és l'encarregada de captar i tractar les dades del Kinect. Per fer-ho s'han utilitzat les llibreries OpenNI i NITE. L'altre, s'encarrega del renderitzat i la lògica del serious game. Aquesta ha estat implementada amb el game engine Panda3D. Les dues aplicacions es comuniquen per intercanviar paràmetres de control i el vídeo en viu. L'aplicació connectada al Kinect guarda el vídeo i les dades del moviment de cada exercici, atenent a les indicacions de l'aplicació del joc. Per solventar aquestes necessitats s'han utilitzat diferents tecnologies com: protocol UDP, codificació de vídeo amb la llibreria FFMPEG, memòria compartida i bases de dades Sqlite3.

També s'ha creat una aplicació on els fisioterapeutes configuren les teràpies per als seus pacients especificant el nombre de repeticions i altres paràmetres. Finalment, les dades de les sessions fetes pels pacients es mostren als fisioterapeutes. Aquestes dades es presenten de tres formes diferents: gràfics en dues dimensions de les rotacions de les articulacions, vídeos gravats amb la càmera de color del Kinect i els

moviments 3D capturats pel sistema de captura. Aquesta aplicació també ha estat programada amb Panda3D.

El desenvolupament d'aquesta plataforma s'ha realitzat contant amb la col·laboració dels fisioterapeutes Bernat Pascual i Eva Pascual, els quals els hi agraïm la seva implicació i dedicació. També voldria agrair la feina feta per Eduard Ruesga i Meritxell Aragonés, tècnics del laboratori MediaLab on s'han realitzat les captures de la validació biomecànica. Voldria donar les gràcies a l'empresa col·laboradora LABSID pels consells i indicacions que han fet durant els sis mesos del projecte, i en especial al seu director executiu Toni Susín. Finalment, agrair també als advisors del projecte, Oscar García i Marc Antonijoan, pel suport donat durant la realització del projecte.

ÍNDEX

ÍNDEX DE FIGURES

1. INTRODUCCIÓ

1.1 MARC

La captura del moviment és un procés on la postura del cos és detectada per un sensor d'entrada. Els sistemes de captura més utilitzats necessiten marcadors al cos per fer-ho. Aquests sistemes tenen 2 grans inconvenients: és enutjós (necessita preparació) i molt costós. Moltes aplicacions de Human Computer Interaction (HCI) es podrien beneficiar de la detecció sense marcadors. Aquesta detecció està basada en visió per ordinador.

Diferents àmbits de la medicina utilitzen la captura del moviment per diferents propòsits. El *gait analysis* (anàlisi de la marxa)[1][2][3] és l'aplicació més utilitzada de la captura del moviment en medicina. Diferents tècniques permeten als metges avaluar el moviment humà a partir de factors biomètrics, sovint fent streaming en temps real de la informació a algun programari analític. En fisioteràpia, els pacients que es recuperen d'operacions quirúrgiques han de realitzar exercicis a la clínica per la seva recuperació. El millor lloc on es podria fer és a casa, on la gent pot escollir quin és el millor moment. Des de fa alguns anys la industria dels videojocs està permeten l'ús de videoconsoles per fer rehabilitació a casa. Amb la introducció del Wiimote de la Nintendo Wii, l'Eye Toy de la PlayStation 2 (PS2), l' Eye de la PlayStation 3 (PS3) Eye i més recentment amb el Kinect de Microsoft i el Move de la PS3, hi ha un ampli ventall de dispositius capaços de fer seguiment de moviments per jugar amb les videoconsoles.

Aquests dispositius són molt importants en un serious game de rehabilitació utilitzant kinesioterapia, la fisioteràpia que utilitza el moviment com a eina per a la recuperació. Durant la teràpia física convencional el fisioterapeuta ha d'ensenyar al pacient a corregir els moviments erronis. En un serious game, és el joc el que guia al pacient en aquest aspecte i per tant ja no és necessari que el fisioterapeuta estigui present en el moment que es realitzen els exercicis de recuperació.

La repetició, el feedback i la motivació també són punts claus. La repetició d'un moviment és un factor important en l'aprenentatge motriu . Per tant, la pràctica ha d'estar lligada a un èxit incremental mentre es fa el moviment de manera correcte.

Tot i la importància de la repetició, els pacients no acostumen a acabar les sessions ja que perden la motivació. Els serious games permeten entretenir i/o distreure al pacient i per tant solucionar aquest problema.

1.2 Estat de l'Art

1.2.1 Seguiment del cos sense marcadors

Hi ha diferents tècniques per capturar el moviment sense marcadors. Es poden utilitzar dispositius mecànics com giroscopis, acceleròmetres i potenciòmetres. En aquest projecte no hem contemplat l'ús d'aquest tipus de dispositius ja que són engorrosos i incòmodes per l'usuari final. En canvi, la visió per computador ens permet capturar el moviment mitjançant càmeres. D'aquesta manera l'usuari no ha de col·locar-se res al cos.

El seguiment del cos en visió per computador es pot realitzar fent servir una sola càmera, utilitzant càmeres estereogràfiques o múltiples càmeres. A continuació es farà un breu repàs d'aquestes diferents configuracions.

Visió monocular

Utilitzar una única càmera evita els problemes ocasionats en la sincronització de les dades en sistemes de múltiples càmeres. A més a més el cost computacional és menor i facilita la captura en temps real. Fer la captura amb una sola càmera també fa més accessible la captura a casa però redueix la seva eficàcia al seu camp de visió i punt de vista.

Una de les maneres de capturar consisteix en sostraure el fons de la imatge, quedar-nos amb la silueta de la persona i utilitzar la informació de color del cos per fer el seguiment dels moviments.

Chih-Chang Yu et al.[4] van aconseguir fer el seguiment del cos automàtic des d'una vista d'una càmera. Consisteix en extreure un model 2D, una silueta, del cos humà a partir d'11 punts de les articulacions incloent: cap, espatlles, malucs, colzes, genolls, mans i peus. En primer lloc, es fa la subtracció del fons per identificar el cos en moviment. El tronc i el cap són les primeres parts detectades buscant la similitud de formes. Aquestes parts es connecten dibuixant una línia i a partir d'aquí es comencen a buscar la resta d'articulacions i es van connectant. Aquest sistema funciona bé quan les persones a capturar es troben perpendiculars a la càmera, ja que les formes que el sistema busca s'adeqüen a aquesta vista. Per altra banda, el sistema és molt vulnerable a les oclusions i les ambigüitats de les profunditats.

FIGURA 1 A L'ESQUERRA, DETECCIÓ DEL CAP I EL TRONC. A LA DRETA, CONNECTIVITAT DES D'EL CAP

Nicholas et al. [5] van utilitzar probabilitat bayesiana per fer la reconstrucció del cos en 3d utilitzant una única càmera. Aquest enfoc també utilitza el coneixement previ utilitzant un conjunt d'entrenament per tal de fer el seguiment 2D de les articulacions. Es divideix el cos en 14 parts i l'algorisme calcula una mitja ponderada entre el fotograma actual i els anteriors per calcular les posicions. Un cop es fa el càlcul 2D es pot passar a la reconstrucció d'un model 3D (Figura 2). El sistema perd molta fiabilitat quan hi ha oclusions.

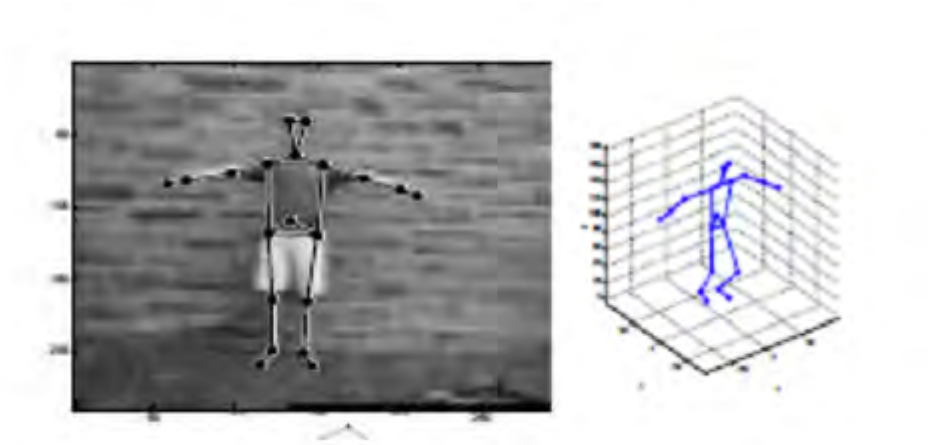

FIGURA 2 IMATGE 2D I LA SEVA RECONSTRUCCIÓ EN 3D

Múltiples càmeres

L'ús de múltiples càmeres millora la desambigüitat que produeix una sola vista. Visualitzar l'escena amb vàries càmeres evita moltes de les oclusions I permet calcular la profunditat.

Diversos projectes i tesis doctorals s'han realitzat per fer la captura de moviment amb múltiples càmeres. En [6] se'ns presenta la reconstrucció d'avatars a partir de la captura del moviment. Es calculen una seqüència de volums a partir de cada stream

de vídeo de cada càmera. Els volums es calculen fent un anàlisi espaciotemporal i a partir d'aquí s'extreuen esquelets geomètrics del subjecte capturat. A posteriori s'apliquen algorismes de cinemàtica inversa per assegurar la correcció física de les postures estimades. Aquesta reconstrucció és pot fer interactivament. Aquest procediment requereix una calibració de les càmeres i aquesta és molt sensible a posteriors errors.

A [7], el seguiment de la postura del cos sencer es fa utilitzant mostreig estocàstic. La representació volumètrica del cos s'obté a partir de siluetes de diversos fotogrames. A les diferents vistes se'ls hi aplica la subtracció de fons i de silueta. La forma 3D ve donada per la intersecció dels cons de projecció donats per la màscara de la silueta. Després les dades són aplicades a un model 3D usant SMD (stochastic meta descent). En aquest sistema es molt important la sincronització de les dades de les diverses càmeres i les transformacions dels seus sistemes de coordenades. Aquestes dues premisses dificulten les captures. Les càmeres han de posicionar-se a la posició calculada i si es varia és necessari tornar-les a calibrar.

FIGURA 3 SUBSTRACCIÓ DEL FONS I EL PRIMER PLA DE 4 VISTES

Càmeres de profunditat

Les imatges en 2 dimensions capturades per càmeres tradicionals perden la informació de profunditat de l'escena en 3 dimensions. Aquesta informació és molt important i útil a l'hora de percebre i visualitzar l'entorn del món real. Amb la immediata evolució de la industria del entreteniment i el continu desenvolupament de dispositius digitals, han sorgit en els últims anys càmeres que permeten generar models en 3 dimensions. Aquest tipus de càmeres les anomenaríem càmeres 3D. La captura de vídeo amb càmeres 3D permet fer broadcasting 3D. Aquesta característica a la vegada permet

incrementar la interactivitat i l'experiència d'usuari en entorns de realitat virtual. Diferents tipus de càmeres 3D estan disponibles en el mercat. *Zcam* és una marca de càmeres TOF per aplicacions de vídeo de 3DV Systems. Les càmeres TOF (Time-of-flight) són sensors actius que determinen el valor de la profunditat de píxel mesurant el temps que triga una llum infraroja a viatjar cap a l'objecte i tornar a càmera. Sistemes com Canesta, Mesa Image [8]i aplicacions com [9][10] van estar motivats per les càmeres TOF. Les càmeres de profunditat PrimeSensor determinen la profunditat capturant la projecció de punts infrarojos utilitzen un sensor CMOS.

FIGURA 4 A L'ESQUERRA, ZCAM. A LA DRETA, PRIMESENSOR

1.2.2 SERIOUS GAMES DE REHABILITACIÓ

Diferents dispositius han estat usats com dispositius d'entrada en serious games de rehabilitació: des de sistemes comercials de videojocs[11], dispositius específics com mesuradors de pressió[12], webcams[13], tecnologies més sofisticades com data gloves (guants de realitat virtual) i dispositius magnètics[14]. La tendència ens porta cap a la interacció amb el tot el cos. El Kinect[15] de Microsoft fa el seguiment del cos sencer del jugador i utilitza la posició jugador com a informació dins dels jocs. Les aplicacions mèdiques requereixen fiabilitat en les dades de captura. No s'ha trobat cap estudi que comprovi si Kinect és suficientment fiable o quin grau de precisió té. Aquest és un aspecte que s'afronta en aquest projecte. Superant aquesta premissa, Utilitzant la càmera de profunditat de Primesense i un framework associat com FAAST[16] podria ser possible fer serious games per a la rehabilitació.

Existeixen diferents serious games per a la rehabilitació. Per classificar-los utilitzarem el mateix criteri que [17]. En aquest article es classifiquen els serious games segons l'àrea d'aplicació, la tecnologia d'interacció, la interfície del joc, el nombre de jugadors, el gènere del joc, l'adaptabilitat del joc, donar feedback al pacient de com fa l'exercici,

el seguiment de les teràpies i la portabilitat del joc. A la Figura 5 hi ha una taula on s'especifiquen els aspectes descrits d'alguns serious games de rehabilitació.

	[12]	[14]	[18]	[19]	[20]	[13]	[21]
Àrea d'aplicació	Física	Física	Cognitiva	Cognitiva	Física i cognitiva	Física	Física
Tecnologia d'interacció	Moviment del pes corporal	Captura de moviment + HMD	Parla + Tàctil + Captura de moviment + Biosensors	Teclat	Captura del moviment	Captura del moviment	WiiMote i WiiBalance
Interfície del joc	2D	3D	3D	3D	3D	2D	2D
Número de jugadors	1	1	1	1	1	1	1/Múltiples
Gènere del joc	Memoria + simulació	Simulació	Estratègia	Simulació	-	Simulació	Laberint
Adaptabilitat	Si	Si	Si	No	Si	Si	Si
Seguiment de la teràpia	Si	Si	Si	No	Si	Si	Si
Feedback al pacient	Si	Si	Si	-	Si	Si	-
Portabilitat	Casa	Clínica	Clínica	Clínica	Clínica / Casa	Casa	-

FIGURA 5 CLASSIFICACIÓ I COMAPARITVA DE SERIOUS GAMES DE REHABILITACIÓ

Les àrees d'aplicació d'aquests serious games són la rehabilitació física i cognitiva. La captura del moviment és un tret comú que tenen totes les aplicacions de rehabilitació física, en rehabilitació cognitiva la tecnologia d'interacció no té tant de pes. En alguns casos, s'han utilitzat interfícies gràfiques 2D, en altres en 3D. El gèneres dels jocs varien des de jocs de memoritzar, simulació, estratègia i laberints. Aquesta diversitat ens indica que no hi ha relació entre el gènere i la teràpia. L'adaptabilitat, el seguiment de la teràpia i el feedback al pacient es troben en la majoria dels casos. L'adaptabilitat i el feedback són dos característiques que els jocs ja porten implícites, i el seguiment de les teràpies es valora molt positivament per part dels experts. La portabilitat de les aplicacions depèn de la tecnologia usada en cada serious game, veiem que pràcticament la meitat del jocs analitzats es poden utilitzar a casa.

Molts dels serious games de rehabilitació no han evolucionat i s'han quedat en prototips i proves amb pocs pacients. Per tant, no hi ha un workflow tancat, que hauria de començar per la calibració del pacient amb el dispositiu d'entrada, configuració del joc, jugar/ fer la rehabilitació, feedback per al pacient i avaluació mèdica. RehaCom [22] és un sistema que integra aquest workflow i ha estat implantat en varies clíniques i estudis. Malgrat això, està limitat a la rehabilitació cognitiva i dispositius d'entrada estàndards com el mouse. El nostre projecte integra aquest workflow i a més a més utilitza una càmera de profunditat com el Kinect.

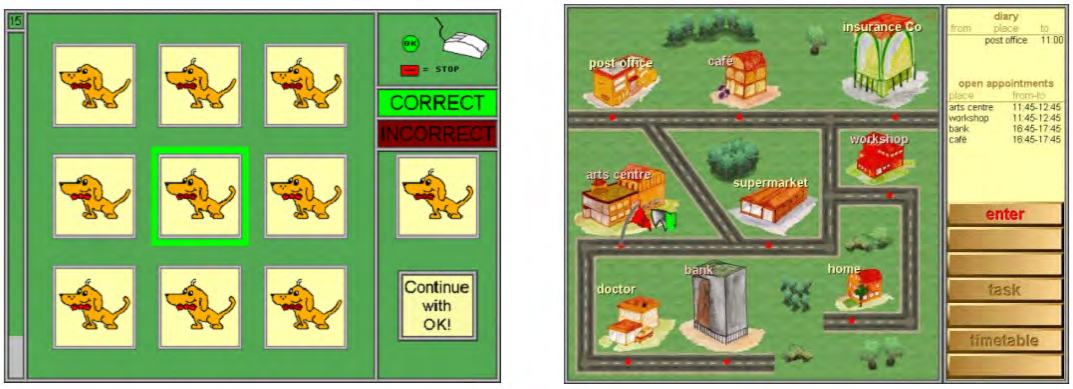

FIGURA 6 REHACOM: EXEMPLES DE JOC. A L'ESQUERRA, ATENCIÓ I CONCENTRACIÓ. A LA DRETA, PLANIFICA UN DIA

1.3 DESCRIPCIÓ DEL PROBLEMA

L'empresa col·laboradora del projecte, va presentar la proposta de projecte, i una petita introducció als serveis i projectes que oferien.

LABSID SL[23], és una empresa tècnica (R+D) especialitzada en animació per computador i processat d'imatge. El seu equip està format per matemàtics, físics, i enginyers.

Un dels seus productes principals és eBiom (easy Biomechanics), un software especialitzat que combina sistemes òptics de Motion Capture amb electromyography sensors de força, acceleròmetres que permeten mesures precises de les articulacions i l' activitat muscular. També han dirigit diverses tesis doctorals que tracten temes com simulació de roba en 3D, animació facial, animació de fluids i diverses aplicacions mèdiques.

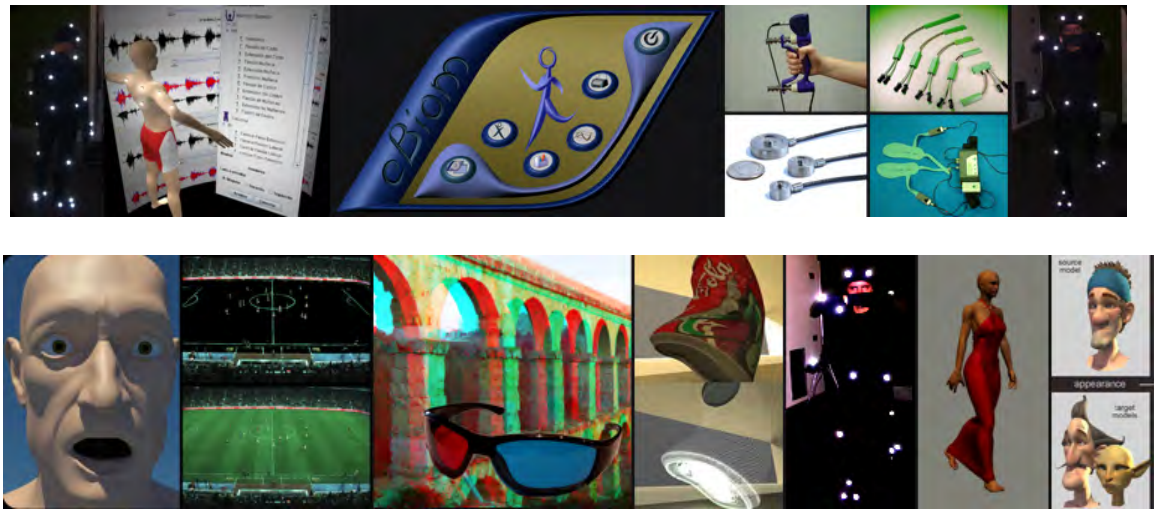

FIGURA 7 A LA PART SUPERIOR, EBIOM; A LA PART INFERIOR DIVERSOS PROJECTES DE LABSID

LABSID va fer una proposta de desenvolupament a ARtiK que incloïa la creació d'un prototip de Serious Game basat en tecnologies de l'entreteniment (de baix cost) com a recurs estratègic aplicat al context de les teràpies de rehabilitació i altres aplicacions mèdiques.

LABSID va proposar utilitzar aquests perifèrics: el Kinect de Microsoft, el WiiMote de Nintendo Wii, i la WiiBalance de Nintendo Wii.

FIGURA 8 A LA PART SUPERIOR, EL KINECT; A LA PART INFERIOR ESQUERRA, EL WIIMOTE; I A LA PART INFERIOR, EL WIIBALANCE

La part més important del projecte de cara a la seva utilització en el camp de les teràpies mèdiques era comprovar la precisió exacta que oferien els perifèrics, per a veure les limitacions que això podia tenir en segons quines aplicacions. Un cop validades les dades que poden oferir els perifèrics, l'objectiu proposat era desenvolupar una plataforma de captura de moviment low-cost que permetés als pacients realitzar la rehabilitació sota un control real.

Així doncs els objectius inicials del projecte eren els següents:

- La utilització de nous sensors que poguessin ser controlats des de l'aplicació final.
- La validació dels perifèrics low-cost, utilitzant sensors professionals.
- L'anàlisi de l'experiència d'usuari i l'eficiència podent accedir a les instal·lacions del User-Lab de La Salle.
- L'ús de guies per al disseny de l' de l'aplicació final.

1.4 SOLUCIÓ PROPOSADA

Una vegada començada la definició del projecte per part de l'equip, ARtiK, el primer mes del projecte es va dedicar a la recerca d'informació i la definició de la idea. L'empresa col·laboradora LABSID, va proposar unes tasques interessants de cara al projecte i el resultat final, que afegien un valor tècnic i científic a l'aplicació.

La solució proposada per l'empresa col·laboradora va ser la següent:

a. **Captura de dades amb 2 dispositius Kinect:** Aconseguir més robustesa de la captura utilitzant les dades de 2 perifèrics capturant al mateix instant.

b. **Utilitzar el sistema de captura per a reconstruir escenes en 3 dimensions:** Aconseguir la reconstrucció d'objectes i escenes en 3D utilitzant el dispositiu de captura a mode d'escàner 3D.

c. **Millorar la captura d'esquelet 3D que ofereix la llibreria OpenNI:** Aconseguir un esquelet de la persona capturada més precís i amb nous ossos per a poder realitzar més moviments i augmentar les possibilitats.

d. **Validació Biomecànica:** Estudiar els límits de precisió de les mesures que ofereix el dispositiu i comparar-ho amb sistemes de captura professionals com per exemple el Media Lab de La Salle.

e. **Dispositiu de captura + Realitat Augmentada en el camp de la rehabilitació**: Disseny d'aplicacions amb Realitat Augmentada per a rehabilitació, implementant un comptador de repeticions, i un pla de marketing per a centres de fisioteràpia.

L'objectiu de la proposta no era desenvolupar tots els apartats, sinó tenir una guia d'interessos de l'empresa col·laboradora, per a poder escollir i guiar l'evolució del projecte.

Es va dur a terme una investigació del mercat en aquest tipus d'aplicacions, es va contactar amb fisioterapeutes i es van escollir uns objectius, respectant les preferències del grup i de LABSID.

Els objectius del projecte marcats per l'equip són els següents:

1. **Millora de les teràpies de rehabilitació**. La majoria de les teràpies de rehabilitació es basen en la repetició d'exercicis. Això provoca que el pacient perdi la motivació al llarg del procés, i la rehabilitació es converteix en avorrida i lenta.

2. **Seguiment de les teràpies**. Les dades recollides pel dispositiu Kinect durant la teràpia dels pacients serà facilitada als professionals per a que puguin realitzar un seguiment de l'evolució del pacient.

3. **Fiabilitat**. Com a aplicació de l'àmbit mèdic, la informació facilitada ha d'estar degudament contrastada i validada. Per aquest motiu es realitzarà la validació biomecànica ja esmentades.

Com a equip vam resumir els nostres objectius en,

- **Missió:** "Millorar la qualitat de vida de les persones, utilitzant les noves tecnologies com a instrument de millora de processos quotidians".
- **Visió:** "Projecte internacional i innovador, que genera valor per a la societat i per als nostres clients i consumidors".

Per assolir aquest objectius el projecte queda dividit en tres parts:

1. Validació biomecànica.

Per a assegurar la fiabilitat de les dades, s'ha realitzat una validació biomecànica, contrastant els resultats obtinguts amb Kinect i amb un Motion Capture professional.

2. Aplicació pacient

Rehabtimals es un *serious game* per a realitzar exercicis de rehabilitació, que busca millorar la motivació dels pacients cap a les teràpies al mateix temps que proveeix als fisioterapeutes la possibilitat de realitzar un seguiment d'aquestes teràpies. *Rehabtimals* està format per un **conjunt de mini-jocs** protagonitzats per 3 tipus de **personatges animals**: els guacamayos, les tortugues marines, y els orangutans, corresponents a 3 ambients: aire, mar i terra respectivament. Aquest joc incorporà la teràpia de rehabilitació del genoll per lesions de lligaments creuats.

3. Aplicació fisioterapeuta:

Aplicació d'escriptori per gestionar els pacients i visualitzar les sessions enregistrades.

El projecte Rehabtimals es planteja com una solució completa al cicle de recuperació d'un pacient. El cicle comença amb el fisioterapeuta configurant la teràpia de recuperació pel pacient en l'aplicació fisioterapeuta. A continuació, el pacient escull el personatge amb el que la seva evolució i la seva recuperació del pacient aniran lligades. Aquest seria l'univers Rehabtimals. Posteriorment, el pacient realitza la sessió que el fisioterapeuta a personalitzat per ell, sent aquesta transferida al joc. Els exercicis realitzats dins del joc són enregistrats per l'aplicació en forma de moviments 3D i vídeos. El cicle es tanca quan el fisioterapeuta fa el seguiment dels exercicis visualitzant les dades enregistrades. La Figura 9 exemplifica aquest cicle.

FIGURA 9 WORKFLOW DEL PROJECTE

1.5 PERSPECTIVA DEL PROJECTE

Aquest treball està dividit en 6 capítols. El capítol 2 tracta de la investigació realitzada sobre la rehabilitació física per decidir quina teràpia implementar en forma de serious game. En aquest capítol es fa una introducció a la fisioteràpia i s'expliquen diferents tipus de teràpies especificant les sessions de recuperació per lesions de genoll, espatlla i cervicals.

En el tercer capítol es descriuen els sistemes de captura utilitzats per a la validació biomecànica i el programari utilitzat per fer la captura amb Kinect. També s'explica el procediment realitzat per validar les dades mostrant els resultats i exposen unes conclusions.

A continuació, en el capítol 4 s'explica l'arquitectura de l'aplicació del pacient. Es fa un repàs a les tecnologies i protocols utilitzats per a la programació. Tot seguit s'especifica l' de la solució proposada. En el següent capítol es fa el mateix per l'aplicació del fisioterapeuta.

En el capítol 6 s'extreuen unes conclusions a partir dels resultats de cadascuna de les parts descrites en el projecte i unes línies de futur per a la millora.

2. INVESTIGACIÓ SOBRE REHABILITACIÓ FÍSICA

L'elecció de quins exercicis incorporar a un serious game de rehabilitació física necessitava de la col·laboració d'experts fisioterapeutes. Bernat Pascual i Eva Pascual han estat els encarregats de transferir el coneixement sobre fisioteràpia.

En aquest capítol es farà una breu introducció sobre la fisioteràpia i a continuació s'especificaran diverses teràpies de rehabilitació facilitades pels fisioterapeutes col·laboradors. Finalment s'aclarirà quines teràpies han estat implementades per la consecució del projecte.

2.1 Introducció a la fisioteràpia

La **fisioteràpia** és un conjunt de mètodes que mitjançant l'aplicació d'agents físics, curen, preveuen i adapten a les persones afectades de disfuncions físiques i orgàniques o a les que volen tenir un nivell òptim de salut.

La **rehabilitació** és un tractament per recuperar una funció del organisme disminuïda o pèrdua a conseqüència d'una lesió o enfermetat, per exemple, per mitjà de massatges i exercicis. La rehabilitació que es basa en el moviment s'anomena **kinesiterapia**.

Els tractaments es poden dividir en els següents grups:

- **Preventiu**: Es poden prevenir patologies, lesions, o evitar que empitjorin lesions irreversibles.
- **Curatiu**: S'apliquen en el cas de lesions, bloquejos articulars, …
- **Pre-quirúrgic**: Aquests tractaments consisteixen en ensenyar al pacient una sèrie d'exercicis per que faci abans d'una intervenció.
- **Postoperatori**: Aquests tractaments es realitzen després d'una intervenció quirúrgica per tal de recuperar mobilitat, musculatura, flexibilitat, …
- **Post-inmovilització**: Després de que un pacient tingui inmobilitzada alguna part del cas, ja sigui amb guix o cèdula, necessita d'aquests tractaments per combatre l'atrofia muscular i la incalcificació del óssos.
- **Teràpia activa**

Les tècniques més usades en fisioteràpia són:

- **Mobilitzacions**: Mantenen o milloren la mobilitat articular, o normalitzen el to muscular. Aquestes poden ser actives, voluntàries, lliures, assistides, resistides, gimnàstiques, …
- **Agents físics:** Aquestes tècniques contemplen l'ús de elèctrodes, aplicació de calor (fang, infrarojos, …), aplicació de fred (gel, nitrogen líquid), aplicació d'aigua, aplicació de camps magnètics, l'ús de làsers, …
- **Poleoterapia**: Utilització de sistemes de politges.
- **Mecanoteràpia**: Aquestes tècniques usen aparells simples Figura 10 que serveixen per treballar la mobilitat, la força muscular i la propiocepció.

FIGURA 10 D'ESQUERRA A DRETA. PARAL·LELES, BICICLETA ESTÀTICA I PLAT DE BOHELER

Podríem dir que la captura de moviment amb Kinect, o amb qualsevol altra dispositiu capàs, podria ser una tècnica de fisioteràpia. De les tècniques actuals més comunes, alguns objectius de la Mecanoteràpia es podrien assolir amb la captura de moviment. El treball de la mobilitat és un clar exemple on la captura del moviment pot ser beneficiosa.

2.2 Fisioteràpia postoperatòria

La fisioteràpia postoperatòria es necessita per tractar **traumes musculars i ossis**. Els traumes musculars es poden operar, però no sempre és necessari. En canvi, els traumes ossis sempre s'operen.

En aquest tipus de fisioteràpia es necessita la monitorització del fisioterapeuta. Normalment es combinen sessions diàries als centres de rehabilitació i a casa. Aquestes sessions consisteixen en un conjunt d'exercicis que el pacient ha de realitzar i s'emmarquen dins de diferents fases de recuperació. Aquestes fases són:

- **Guany de moviment:** Aquesta fase té com a finalitat assolir un rang de moviment articulatori.
- **Guany de força**: Després d'assolir el rang desitjat és necessari guanyar força.
- **Adaptació a les activitats diàries**

La realització dels exercicis, quan aquests es fan a la consulta, es fa deixen les articulacions afectades sempre a la vista. Per exemple, si es fan exercicis per a recuperar-se d'una lesió de genoll el pacient vestirà pantalons curts.

En aquest projecte s'han escollit les teràpies de rehabilitació del **genoll**, després d'una lesió de lligaments creuats, i de rehabilitació de **l'espatlla**, després de lesions musculars de supra i manegot dels rotadors.

2.2.1 Rehabilitació del genoll

En esports com la lluita, el bàsquet, la natació de competició, el futbol americà, el futbol australià, l'esquí, el voleibol, el futbol, l'hoquei i altres esports que impliquen gran tensió del genoll, es comú d'esgarrar-se un o més lligaments o cartílags. El lligament creuat anterior es sovint desgarrat com a resultat d'un canvi ràpid de direcció mentre es corre o com a resultat d'algun altre tipus de moviment de gir violent.

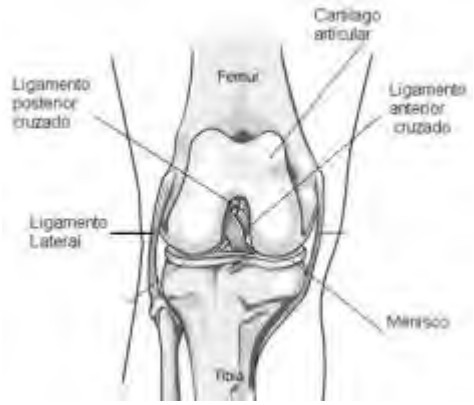

FIGURA 11 GENOLL

Normalment aquesta lesió la pateixen esportistes i persones adultes de entre 30-45 anys que realitzen algun moviment brusc, sovint quan practiquen esport. Després de la intervenció quirúrgica el pacient ha de realitzar una recuperació que contempla les 3 fases de la fisioteràpia postoperatòria.

1. Guany de moviment

L'objectiu d'aquesta fase de recuperació es guanyar flexió del genoll i no perdre la musculatura. Aquesta fase té una durada aproximada d'un mes i mig, tot i que aquesta durada pot variar depenent del pacient. Fins que el pacient no assoleixi 90º de flexió i tingui el vistiplau del fisioterapeuta, no podrà superar aquesta fase.

El pacient ha de realitzar 1 o 2 sessions al dia, depenent de les indicacions del expert. Aquestes sessions poden variar segons el fisioterapeuta i/o pacient. A continuació detallarem una possible sessió.

Sessió

1**.** 20 repeticions de flexió i extensió del genoll. Les repeticions s'han de fer amb un moviment continu sense aguantar cap posició, i el pacient es pot ajudar de les mans per moure el genoll.

FIGURA 12 FLEXIÓ I EXTENSIÓ DEL GENOLL

2. 20 repeticions de flexió i extensió del genoll. Les repeticions s'han de fer amb un moviment continu sense aguantar cap posició. Aquest exercici es fa quan el pacient ja ha fet algunes sessions i es capaç de fer el moviment sense cap ajuda.

3. Posem 5 minuts de gel.

2. Guany de força

En aquesta fase el pacient haurà de guanyar força i enfortir els músculs que intervenen en el moviment del genoll. Aquesta fase té una duració aproximada de 2 mesos i es realitzen d'1 a 2 sessions al dia.

Sessió

1. 20 repeticions de cada grup de moviments. Aquests moviments es poden realitzar assegut o de peu, i el pacient ha de aguantar la posició de flexió durant 5 segons aproximadament.

- Flexió i extensió del genoll
- Flexió i extensió de malucs
- Adducció i abducció de maluc

2. Mitja hora realitzant combinacions dels moviments anteriors. Es pot fer mitja hora seguida o fer dues parts de 15 minuts. En el cas de fer 2 parts, es farà un descans de 2-5 minuts.

3. Activitats de la vida diària

Aquesta fase és la darrera de la recuperació. L'objectiu és adaptar la articulació a possibles situacions quotidianes del pacient. Hi ha dos tipus d'exercicis: Exercicis específics i propiocepció. Els exercicis específics són adhoc per cada pacient, ja que la seva dedicació laboral o d'oci pot ser diferent. En el cas d'un jugador de bàsquet, aquest haurà de realitzar exercicis relacionats amb el seu esport. El treball propioceptiu el realitzen tots els pacients.

El treball propioceptiu consisteix en realitzar equilibris sobre el genoll afectat. Es fan sessions d'una hora de duració que consisteixen en tandes de 10 minuts de treball i 5 minuts de descans. Les tandes es fan dret.

1. **Primera tanda**. Equilibri bipedal amb genolls semiflexionats (màxim 10 º). Es provoquen desequilibris al pacient. Per exemple, el fisioterapeuta pasa una pilota al pacient i aquest l'ha de retornar. El pacient marca la rapidesa de repetició després de retornar a la posició inicial.

2. **Segona tanda**. Equilibri monopodal. S'aguanta la posició tota l'estona que pugui fins 10 minuts com a màxim.

3. **Tercera tanda**. Equilibri monopodal. El fisioterapeuta provoca desequilibris fàcils, en els que el genoll no es mou.

4. **Quarta tanda**. Equilibri monopodal El fisioterapeuta provoca desequilibris forts. Per exemple: tocar el terra, tornar, toca el terra amb l'altra mà, tocar el peu, …

2.2.2 Rehabilitació de les espatlles

Els problemes musculars a l'espatlla, concretament al supra i al manegot dels rotadors, són molt freqüents en esports com l'handbol I el waterpolo. Aquest problemes també poden ser originats per traumatismes al húmer, l'escàpola o clavícula degut a accidents de moto o laborals. Hi ha dues franges d'edat marcades de pacients que pateixen aquesta patologia: dels 16 als 23 anys, normalment per accident de moto; i a partir del 40, per causes diverses. Després de la intervenció quirúrgica el pacient ha de realitzar una recuperació que contempla les 3 fases de la fisioteràpia postoperatòria.

FIGURA 13 ESPATLLA

1. Guany de moviment

Aquesta fase té com a objectiu aconseguir un bon rang de moviment articular, de la mateixa manera que amb el genoll. L'elevació de l'espatlla no pot superar els 90º per tal de no forçar l'articulació, aquest serà l'arc de seguretat. Al principi de la recuperació extranyament un pacient es capaç d'assolir-ho. Es realitzen dues sessions de treball al dia, recomanat una al matí i l'altre a la tarda.

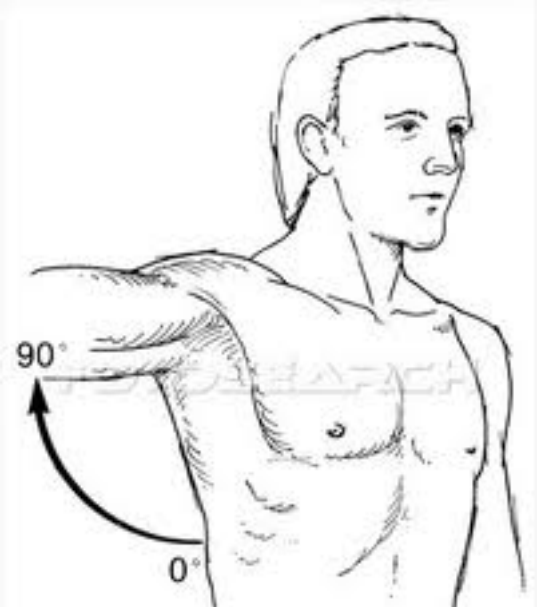

FIGURA 14 ARC DE SEGURETAT

Sessió

1. 15 repeticions de cada grup de moviments. Els moviments s'han de realitzar de manera lenta i controlada, sense aguantar cap posició i sempre tornant al punt inicial. És molt important no superar l'arc de seguretat i mantenir les espatlles alineades. És indiferent realitzar els moviments amb el braç estes o amb el colze flexionat.

- Flexió i extensió sobre el pla sagital.
- Adducció i abducció sobre el pla coronal.
- Adducció i abducció horitzontal sobre el pla transversal.

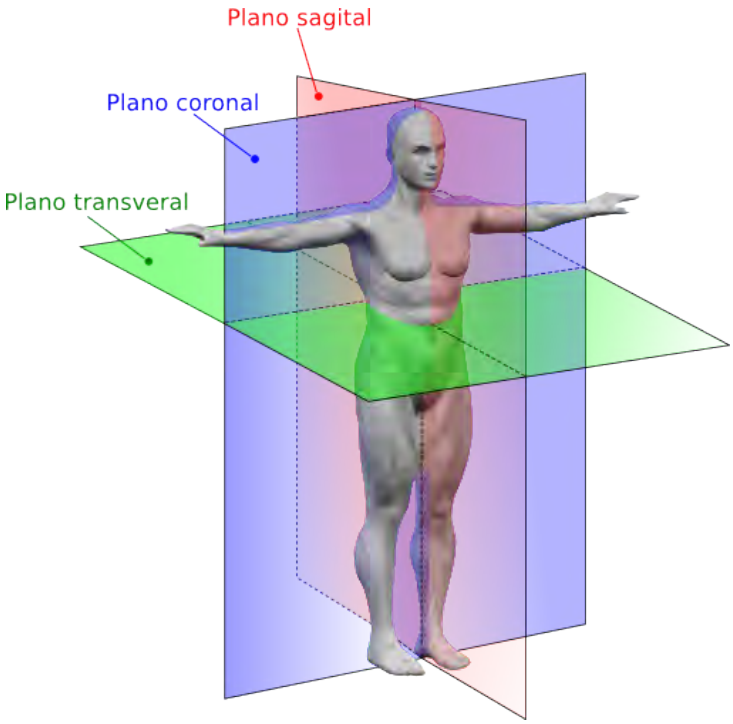

FIGURA 15 PLANS DEL COS

2. 4 minuts realitzant moviments pendulars. El moviment pendular tracta de deixar l'espatlla relaxada i fer-la girar. Es comença dibuixant un cercle petit i pot a poc es van fent cercles més grans. Un cop s'arriba a la màxima grandària del cercle s'inverteix el procediment. A la Figura 16 hi ha una imatge de com es fa aquest moviment.

FIGURA 16 EXERCICI PENDULAR

3. Posem gel a l'espatlla durant 2 o 3 minuts.

2. Guany de força

Un cop el pacient passa a aquesta fase ja no té que respectar l'arc de seguretat. Els exercicis que es realitzen són els mateixos que a la fase anterior però amb més freqüència i aguantant posicions. Es recomana fer una sessió al dia d'una hora i mitja de duració.

Sessió

1. Es realitzen els mateixos exercicis que en el punt 1 de la fase anterior però de manera més ràpida i aguantant les posicions extremes fins que el pacient claudica.

2. Mitja hora realitzant combinacions dels moviments anteriors. Es pot fer mitja hora seguida o fer dues parts de 15 minuts. En el cas de fer 2 parts, es farà un descans de 2-5 minuts.

3. Opcionalment es poden fer exercicis pendulars.

4. Posem gel a l'espatlla durant 5 minuts.

3. Activitats de la vida diària

En aquesta teràpia aquesta fase és diferent per cada pacient. Els exercicis que es realitzaran seran específics segons l'activitat diària del pacient.

2.3 TERÀPIA ACTIVA

En aquest tipus de fisioteràpia no és imprescindible un control del fisioterapeuta. Són teràpies que consisteixen en exercicis que el pacient pot aprendre i practicar-los a mode de manteniment quan ho desitgi. Una de les teràpies actives més comunes és la de cervicals i cadena posterior.

2.3.1 TERÀPIA DE CERVICALS I CADENA POSTERIOR

Aquesta teràpia té la finalitat d'alleujar la molèstia i les tensions a les cervicals. La molèstia a les cervicals és un fet molt comú que moltes persones pateixen sobretot quan comencen a treballar davant d'un ordinador. El fet de passar moltes hores treballant i mantenir una mala posició provoquen una sobrecàrrega a les cervicals. Un altre grup d'afectats per aquesta molèstia són els camioners.

La duració d'una sessió és aproximadament d'una hora. Durant la sessió el pacient ha de realitzar exercicis de cervicals i també de la cadena posterior. La cadena posterior també s'ha d'exercitar ja que les tensions en aquesta part del cos també afecten a les cervicals.

Sessió

Exercicis cervicals

1. 10 repeticions de cada grup de moviments. Els moviments s'han de realitzar aguantant les posicions finals de 2 a 5 segons.

- Inclinació lateral dreta i esquerra
- Flexió de coll endavant
- Rotació del coll dreta i esquerra
- Acostador d'espatlles a les orelles

FIGURA 17 EXERCICIS CERVICALS

Exercicis cadena posterior[24]

2. 3 exercicis de flexibilització o consciència

3. 2 exercicis de forçament muscular

4. 2 exercicis de tensió global

5. Treball propioceptiu

2.3 EXERCICIS ESCOLLITS

En els punts anteriors s'han especificat teràpies per a la recuperació del genoll, recuperació de l'espatlla i manteniment de les cervicals i cadena posterior. En un principi, el projecte contemplava la implementació de les tres teràpies. Es va establir un ordre de preferència entre elles: Genoll, cervicals i espatlla.

Es va decidir començar per implementar la teràpia de rehabilitació de genoll, ja que no era necessari fer millores en l'algorisme de captura de moviment. En segon lloc, es va escollir la teràpia de cervicals. Aquest exercici requeria d'una millor de l'algorisme de captura, explicat a [25] . La teràpia de l'espatlla presenta problemes de viabilitat. El sistema de captura amb Kinect requereix d'una posició de calibració. Aquesta calibració consisteix en aixecar les mans, moviment inadequat per lesionats de l'espatlla.

Finalment hem implementat la teràpia de genoll i no la resta de teràpies per motius de temps. La teràpia implementada tampoc s'ha fet en la seva totalitat, limitant el projecte a la fase 1 i 2 d'aquesta.

3. VALIDACIÓ BIOMECÀNICA

3.1 INTRODUCCIÓ

En aquest apartat s'explicarà com s'ha dut a terme la validació biomecànica. Aquest procediment consisteixi en comparar els valors del seguiment de l'esquelet que fa el Kinect amb els valors, considerats correctes, que ens donen els sistemes professionals de captura de moviment. Per poder dur a terme aquesta comparació es necessari realitzar simultàniament l'enregistrament de moviments amb el Kinect i un Motion Capture amb marcadors òptics.

La captura de moviment amb Kinect s'ha realitzat amb el framework OpenNi i el middleware NITE. Ambdós funcionen amb el llenguatge de programació C++. El moviment capturat es enregistrat en un fitxer de text. Per altra banda, s'ha utilitzat un sistema de captura Vicon que genera fitxers CSM (Character Studio Motion Capture). Per tant, per cada moviment es generen dos fitxers que són processats en una aplicació Matlab. Aquesta aplicació de processament s'encarrega de posar en correspondència cada parella de fitxers i calcular els errors entre ells.

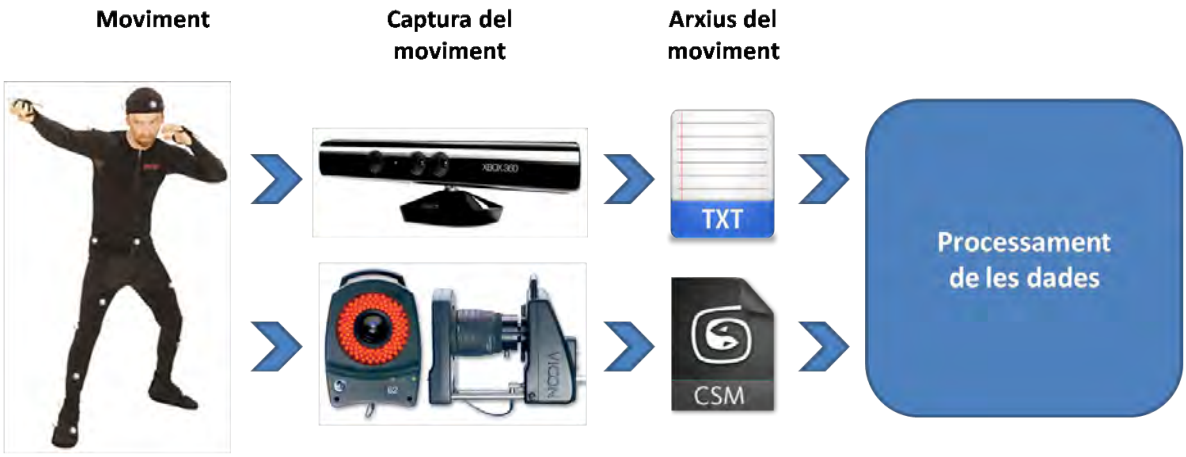

FIGURA 18 DIAGRAMA DE LA VALIDACIÓ BIOMECÀNICA

3.2 CAPTURA DEL MOVIMENT

El procés de captura del moviment tracta de que una persona, que serà denominada actor, faci una representació d'un moviment que serà capturat per algun o alguns mecanismes o dispositius per tal d'enregistrar-lo. La captura de moviment és més coneguda pel seu nom en anglès, que és motion capture, a partir d'ara es parlarà de captura de moviment, motion capture o Mocap indistintament.

Els sistemes de motion capture que es comercialitzen actualment es poden categoritzar en tres grans categories: sistemes òptics, sistemes magnètics, i sistemes mecànics. A continuació s'explicarà amb més profunditat el funcionament dels sistemes òptics, remarcant els seus avantatges i inconvenients.

3.2.1 SISTEMES ÒPTICS

Els primers sistemes de captura òptics van ser dissenyats per a aplicacions mèdiques. El primer disponible desenvolupat per a la creació d'aplicacions de gràfics per ordinador va ser el sistema Vicon 8. Un sistema òptic típic consisteix entre 4 i 32 càmeres i un ordinador que controla les càmeres. Normalment, l'actor a capturar ha de portar uns marcadors que són reflectants (passius) o emissors de llum (actius). Els marcadors passius estan fets de materials reflectants i tenen forma esfèrica, semiesfèrica o circular. Les mides i les formes dels marcadors depenent de les resolucions de les càmeres i dels subjectes a capturar (per exemple, marcadors petits són usats per a fer captures facials o de les mans). Els marcadors es col·loquen directament sobre la pell dels subjectes o amb velcro sobre un "vestit" de cos sencer fet de material elàstic , com l'espàndex. Les càmeres en els sistemes òptics passius estan equipades amb díodes emissors de llum (LEDs) i les llums que emeten els LEDs són reflectides pels marcadors.

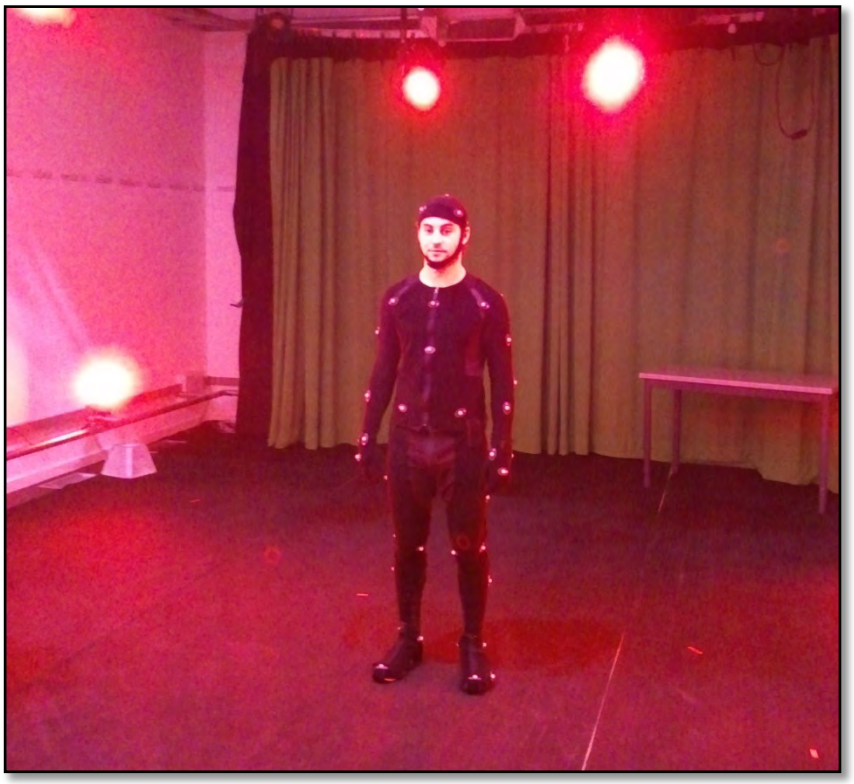

FIGURA 19 ACTOR PREPARAT PER REALITZAR UNA GRAVACIÓ EN UN LABORATORI DE CAPTURA DE MOVIMENT ÒPTIC

D'altra banda, els marcadors en els sistemes òptics actius són LEDs. Alguns d'aquests sistemes il·luminen un LED en un instant de temps, per tal de eliminar la necessitat d'identificar cada marcador. Altres il·luminen tots els LEDs a la vegada. És possible modular l'amplitud o la freqüència de cada LED per identificar els marcadors. Alguns de d'aquests sistemes més nous treballen amb condicions de llum natural, és a dir, poden capturar actors amb diferents "vestits" i a diferents localitzacions fora dels estudis; en canvi, la il·luminació ha d'estar controlada molt cuidadosament en la majoria de sistemes òptics, especialment amb els sistemes òptics passius.

Les càmeres en sistemes òptics poden capturar entre 30 i 2000 fotogrames per segon. Com a mínim dues càmeres han de visualitzar un marcador per tal de determinar la seva posició en 3 dimensions, malgrat que és preferible que siguin tres o més càmeres per a més precisió. A vegades, el subjecte que esta essent capturat, o un altre actor, o algun objecte poden cloure algun dels marcadors de l'actor. Quan hi ha aquestes oclusions, cap càmera pot veure aquests marcadors i això provoca la pèrdua de dades. Hi ha tècniques d'edició i eines per arreglar aquesta pèrdua de dades. De totes formes,

quan molts marcadors estan closos, és impossible arreglar el problema. Les dades generades per sistemes òptics són netes i precises quan no es pateixen problemes d'oclusions. A major nombre de càmeres, hi ha menys probabilitat d'haver-hi oclusions.

FIGURA 20 CÀMERA VICON

La configuració dels marcadors és flexible. Es pot fer servir la configuració de marcadors que el fabricant proporciona o es pot dissenyar una configuració pròpia que s'adeqüi a les necessitats. Un gran nombre, relativament, de marcadors es poden capturar simultàniament, per exemple, més de 200 marcadors amb 16 càmeres. El subjecte es pot moure lliurement dins del volum de captura ja que no hi ha cables que el lliguin.

La visualització en temps real de les captures normalment està limitat a figures de pal, malgrat que algun software connectat a l'equip de captura com Autodesk MotionBuilder pot renderitzar en temps real. Les dades gravades són processades per calcular les trajectòries

dels marcadors en un procés llarg de post-processat per tal d'aconseguir resultats estables. Les dades de les rotacions poden ser calculades en temps real, però normalment són calculades a partir de les dades de posicions en post-processat.

Avantatges:

- Les dades són precises.
- La velocitat de captura és elevada.
- Múltiples actors es poden capturar a la vegada.

- Es pot utilitzar un gran nombre de marcadors.
- La configuració dels marcadors es pot modificar fàcilment, depenent dels objectius.
- Els subjectes es poden moure lliurement pel volum de captura.
- El volum de captura pot ser més gran que en la majoria d'altres sistemes.
- Es poden generar dades d'esquelet.

Inconvenients:

- Es requereix un post-processat extens.
- Les dades de rotació han de ser calculades a partir de les dades de posició en post- processat.
- Els marcadors es poden cloure pels propis actors o objectes, comportant això pèrdua de dades.
- La il·luminació requereix control en la majoria d'aquests sistemes, especialment en els passius.
- La visualització en temps real es limita a figures de pal.
- Normalment el hardware és mes car que altres equipaments de motion capture.

3.2.1.1 ARXIUS CSM

Després de realitzar una gravació al Motion Capture aquesta és guardada en un arxiu de moviment. En aquest projecte hem treballat amb el format Character Studio Motion Capture (CSM). El format CSM és un arxiu escrit en ASCII que s'utilitza per importar les posicions dels marcadors de varis sistemes de captura de moviment capa a Character Studio (eina del 3D Studio Max) per animar personatges bipedals.

En la Figura 21 tenim un exemple d'arxiu CSM. Com es pot veure tenim informació del arxiu en general nom del fitxer, l'hora i la data de la captura, el nom de l'actor, comentaris, el primer frame i l'últim, la velocitat de fotograma. A continuació s'especifica l'ordre en que els valors a la part *$points* es mostren especifiquen el nom del marcador.

```
$filename fashion1.csm
$capturedate 1998/01/31
$capturetime 23:31:59
```

```
$actor Bogy

$comments
This is a Character Studio CSM File

$firstframe 1
$lastframe 3
$rate 60

$order

C7 CLAV LANK LBHD LBWT LELB LFHD LFIN LFWT LKNE LSHO LTOE LUPA LWRA LWRB
RANK RBHD RBWT RELB RFHD RFIN RFWT RKNE RSHO RTHI RTOE RWRA RWRB STRN T10

$points

1 980.6 -2365.8 1541.3 1030.6 -2239.9 1492.1 967.3 -2427.5 181.8 936.2 -
2330.1 1672.9 939.4 -2359.1 1109.3 782.9 -2263.3 1175.7 959.0 -2196.9 1753.8
762.3 -2135.1 862.6 949.3 -2187.2 1082.1 934.2 -2343.2 583.1 870.4 -2246.7
1525.4 1031.7 -2339.7 83.1 814.6 -2218.7 1318.8 799.4 -2132.3 993.7 729.2 -
2230.2 966.5 1110.1 -2060.1 197.3 1066.8 -2351.6 1670.5 1114.0 -2391.4
1116.4 1290.5 -2574.6 1396.5 1105.7 -2231.2 1740.1 1217.5 -2369.8 1149.5
1159.8 -2233.6 1093.8 1138.8 -2119.2 605.9 1136.6 -2342.8 1564.9 1081.3 -
2126.7 775.0 1065.2 -1944.5 112.2 1276.8 -2398.1 1271.2 1294.0 -2490.1
1183.6 1044.2 -2199.9 1395.9 988.1 -2404.3 1417.4

2 979.8 -2358.9 1540.1 1029.4 -2232.4 1489.5 966.5 -2426.7 181.8 936.2 -
2322.0 1671.1 940.7 -2351.6 1107.3 783.9 -2249.8 1173.7 959.4 -2189.0 1752.6
767.9 -2113.2 863.4 948.9 -2177.9 1080.7 934.4 -2338.9 583.3 871.7 -2242.9
1523.4 1031.7 -2339.5 82.9 814.6 -2207.6 1317.8 803.3 -2113.8 994.5 732.4 -
2209.8 964.1 1108.3 -2039.1 197.5 1067.2 -2343.2 1669.3 1115.0 -2382.1
1115.0 1292.6 -2563.1 1400.3 1106.5 -2223.7 1738.5 1212.0 -2367.2 1155.5
1162.2 -2222.1 1093.8 1140.2 -2119.4 601.9 1135.4 -2334.3 1563.9 1079.9 -
2109.1 774.4 1064.5 -1921.3 116.0 1278.4 -2396.2 1273.8 1284.5 -2483.4
1186.2 1043.6 -2192.5 1394.0 987.5 -2397.7 1415.2
```

FIGURA 21 EXEMPLE D'ARXIU CSM

3.2.2 CAPTURA DEL MOVIMENT AMB KINECT

3.2.2.1 KINECT

Kinect està basat en programari desenvolupat internament per Rare, una filial de Microsoft Game Studios propietat de Microsoft, i en la tecnologia de les càmeres del desenvolupador israelià PrimeSense, que interpreta la informació de l'escena 3D a partir de la llum infraroja contínuament projectada. *Light Coding* és el sistema d' escanejat 3D que utilitza Kinect. Aquest utilitza una variant de la reconstrucció 3D basada en imatge.

El sensor Kinect és una barra horitzontal connectada a una petita base amb un pivot motoritzat. Està dissenyat per posicionar-se longitudinalment sobre o sota del

dispositiu de visualització de vídeo. Les característiques del dispositiu són una càmera RGB, un sensor de de profunditat I un multi-array de micròfons funcionant amb un sistema propietari que permet la captura del moviment 3D del cos sencer, reconeixement facial I reconeixement de veu.

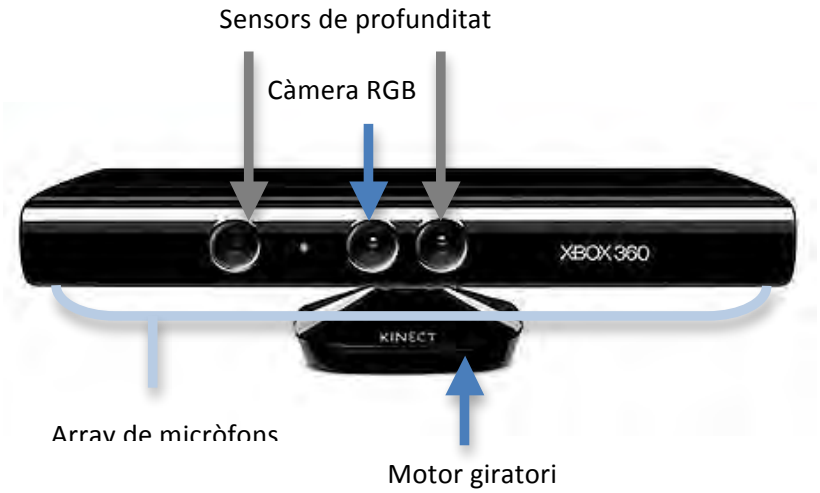

FIGURA 22 KINECT

El sensor de profunditat consisteix en un projector làser d'infrarojos combinat amb un sensor CMOS monocroma, que captura dades de vídeo en 3D sota qualsevol condició de llum ambiental. El rang de sensibilitat del sensor de profunditat és ajustable, I el programari de Kinect és capaç de calibrar el sensor basant-se amb la manera de jugar I l'entorn físic del jugador.

El Kinect és capaç de fer el seguiment de fins a 6 persones, analitzant el moviment I reconeixent 20 articulacions de dos jugadors. Tot I això, PrimeSense constata que el nombre de persones que el dispositiu pot detectar només està limitat pel camp de visió de la càmera.

S'ha determinat gràcies a enginyeria inversa que la sortida de vídeo és de 30 fotogrames per segon. El flux RGB de vídeo utilitza una resolució de 8 bits VGA (640 x 480 píxels) amb un filtre de color Bayer, mentre que el flux de vídeo del sensor monocroma de profunditat té una resolució de 11 bits VGA (640 x 480 píxels) I permet 2.048 nivells de sensibilitat. El sensor Kinect té una limitació de 1.2 fins a 3.5 metres de distància quan s'utilitza el programari de Xbox. L'àrea de joc està al voltant de 6m², tot I que el sensor pot fer el seguiment en un rang més gran que va aproximadament de 0.7 fins a 6 metres. El camp angular de visió és de 57º horitzontalment I 43º verticalment, mentre que el pivot motoritzat pot rotar fins a 27º cap amunt o cap avall. El camp de visió horitzontal del sensor té una distància mínima de visió de 0.8 metres

aproximadament I el vertical és de 63 cm aproximadament, donant una resolució per sobre de 1.3 mm per píxel. L'array de micròfons té 4 càpsules de micròfons i funciona amb 16 bits d'àudio per cada canal amb un mostreig de 16 KHZ.

Degut al sensor motoritzat el Kinect necessita més potència de la que li proporciona la Xbox 360 a través dels ports USB, s'utilitza un connector propietari que combina la comunicació USB amb potència addicional. Les noves Xbox 360 S inclouen aquest connector, per la resta s'utilitza un cable que separa la corrent de la comunicació USB.

3.2.2.2 OpenNI i NITE

OpenNI (Open Natural Interaction) és un framework multiplataforma Open Source que defineix unes API's per crear aplicacions que utilitzin Interacció Natural. El terme Interacció Natural (NI) es refereix al concepte de que la interacció home-màquina està basada en els sentits humans, i OpenNI es centra en la vista i la oïda.

OpenNI proporciona un conjunt de llibreries per ser implementades pels dispositius sensorials, i un conjunt de llibreries per ser implementades per components middleware. Trencant la dependència entre el dispositiu i el middleware, OpenNI permet la portabilitat de les aplicacions sense cap esforç addicional per a que funcionin per sobre de qualsevol altre mòdul.

PrimeSense's Natural Interaction Technology for End-user (NITE) és un middleware que treballa amb el món 3D, basat en les imatges de profunditat, i traslladant aquestes percepcions en dades útils de la mateixa manera que les persones fem. El middleware NITE inclou algorismes de visió per ordinador que permeten identificar usuaris i fer el seguiment dels seus moviments. També un conjunt de llibreries per implementar interfícies d'usuari basades en gestos dels usuaris.

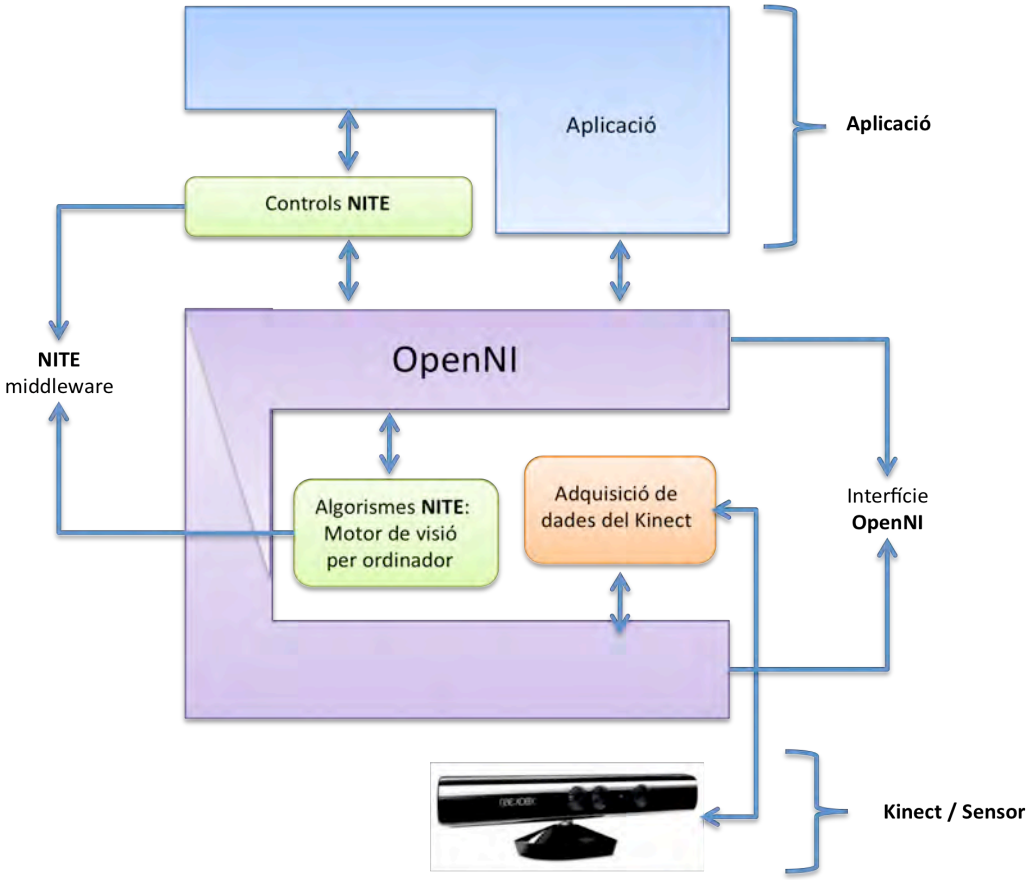

FIGURA 23 VISTA PER CAPES

OPENNI I NITE FUNCIONEN CONJUNTAMENT PER FER APLICACIONS D'INTERACCIÓ NATURAL. LA

Figura 23 mostra una vista de les capes que produeixen, adquireixen i processen les dades de profunditat, fins al nivell més al on l'aplicació utilitza aquestes dades per crear interacció natural. Mostrant així la conjunció de la llibreria i el middleware. A la capa més baixa es troba el sensor. Aquesta capa adquireix els valors del sensor directament, en forma de stream d'imatges de profunditat. A la següent capa, la que té forma de C, representa OpenNI. OpenNI fa d'interfície de comunicació entre el sensor i els components middleware, que són els que analitzen les dades del sensor. Els controls de NITE i els algorismes de NITE són middlewares. El primer és un framework per identificar gestos i a conté elements per fer interfícies. El segon és un motor de visió per ordinador. En la capa superior trobem l'aplicació final. L'aplicació final pot utilitzar els controls de NITE, pot accedir a OpenNI per accedir a les dades generades pels algorismes de NITE, o utilitzar les dades no tractades del sensor.

3.2.2.3 Algorismes de NITE

En aquest punt s'explicaran els algorismes de NITE de segmentació d'usuaris i seguiment de l'esquelet. Ambdós algorismes són els que s'utilitzen per fer la captura de moviment amb Kinect.

Segmentació d'usuaris

La segmentació d'usuaris s'utilitza per identificar i fer el seguiment dels usuaris que es troben a l'escena. Cada usuari a l'escena té una ID única i persistent. El procés de segmentació genera un mapa de l'escena on assigna IDs dels usuaris a cada píxel. El seguiment de l'esquelet utilitza aquest mapa etiquetat per generar esquelets. Cada error de precisió de l'algorisme de segmentació afecta al seguiment de l'esquelet. A la Figura 24 es pot veure el resultat de la segmentació.

FIGURA 24 SEGMENTACIÓ D'USUARIS

Hi ha certes situacions en que aquest algorisme té un rendiment més baix. L'usuari ha d'evitar situar-se molt a prop d'objectes i evitar moure's mentre toca a un altra usuari. Si dos usuaris són closos per un tercer usuari, aquest se li poden assignar IDs dels altres. També s'ha d'intentar no moure el sensor mentre aquest algorisme està actiu.

La ID és fiable mentre l'usuari no abandona el camp de visió. Si ho fa o l'algorisme deixa de segmentar-lo la seva ID es pot assignar a un altra usuari. Si un usuari no és visible en més de 10 segons es considera que s'ha perdut. La seva ID serà alliberada i se li assignarà una altra en cas de tornar a ser detectat. El fet de que les IDs siguin intercanviables fa important capturar els events de nou usuari i pèrdua d'usuari per evitar confusions en el seguiment de l'esquelet.

SEGUIMENT DE L'ESQUELET

Una de les funcionalitats de NITE que utilitzarem en aquest projecte és el seguiment de l'esquelet. Perquè el seguiment de l'esquelet sigui possible és necessari que com a mínim la part superior del cos es trobi en el camp de visó de la càmera. Segons els autors dels algorismes la distància ideal entre l'usuari i el Kinect és de 2 metres i mig. És recomanable que l'usuari no vesteixi roba molt ample i els cabells llargs poden baixar la qualitat del seguiment.

Per que l'algorisme estimi la posició de l'usuari es realitza una calibració. La calibració s'utilitza per ajustar les proporcions del cos al model d'esquelet. Aquest procés dura uns 3 segons on l'usuari a de romandre de cara a la càmera i en la posició de la Figura 25. El cap i els braços durant la calibració han d'estar 100% visibles i l'usuari no pot flexionar els genolls ni doblegar l'esquena.

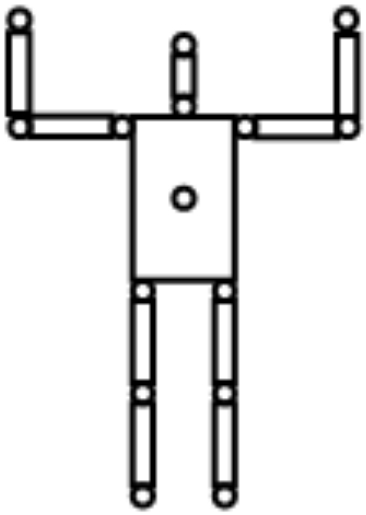

FIGURA 25 POSICIÓ DE CALIBRACIÓ

La API ens retorna la posició i la rotació de les articulacions que es mostren a la Figura 26. Les longituds de les articulacions, com per exemple la distància entre el *elbow*

(colze) i el *shoulder* (espatlla), poden variar al llarg del temps. Segons la documentació de la llibreria les posicions de les articulacions són més precises que les rotacions. Això es degut a que les longituds de les articulacions pot variar i provoca més soroll en el càlcul de les rotacions. També tenim accés a un valor de confidència per cada articulació que és binari: 0 si el seguiment és incert, i 1 si pot fer el seguiment.

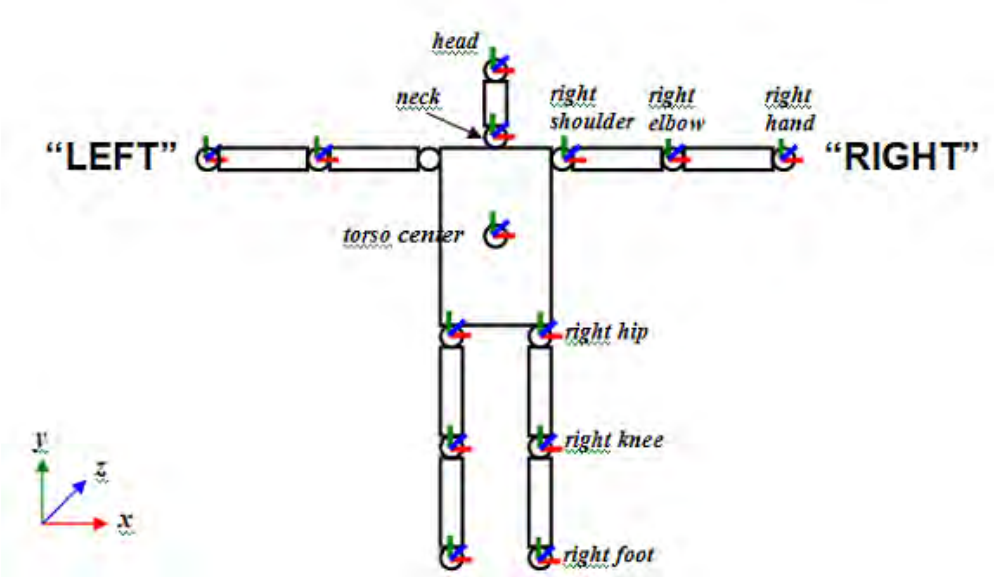

FIGURA 26 DEFINICIÓ DE LES ARTICULACIONS

3.3 IMPLEMENTACIÓ DE LA CAPTURA DE MOVIMENT AMB KINECT

Per realitzar la captura de moviment amb Kinect s'ha implementat una aplicació OpenGL i GLUT en C++ sobre la plataforma Windows. Aquesta aplicació utilitza les llibreries OpenNI i NITE per dur a terme el seguiment de l'esquelet. S'ha utilitzat Microsoft Visual Studio com a entorn de desenvolupament, ja que facilita la creació de projectes C++.

L'aplicació conté un objecte xn::Context. Aquest objecte és qui governa qualsevol aplicació OpenNI ja que s'encarrega de captar les dades del dispositiu Kinect. Les classes xn::DepthGenerator i xn::ImageGenerator capten la imatge de profunditat i la imatge RGB del sensor. Aquests objectes han de ser inclosos dins del objecte xn::Context, d'aquesta manera tindrem accés a la imatge RGB i de profunditat del Kinect. Per executar els algorismes de visió per computador de segmentació d'usuaris i d'estimació de la postura hem d'utilitzar la classe xn::UserGenerator. De la mateixa manera que les anteriors, també ha de ser inclosa al context. També hem implementat la classe Capture per guardar la informació de les posicions de les articulacions estimades i enmagatzemar aquesta informació en fitxers.

A continuació, s'especificarà com s'ha inicialitzat el sistema, s'ha dut a terme la captura del moviment i s'han guardat les dades en arxius de text.

3.3.1 INICIALITZAR OPENGL, OPENNI I NITE

Tota aplicació C++, té una funció *main* que es crida automàticament al iniciar l'execució del programa, aquí és on es fan les inicialitzacions del sistema. Primerament hem d'inicialitzar GLUT i definir com volem que el motor gràfic renderitzi la nostra aplicació. També definim les mides de la finestra i el nom. Després d'inicialitzar el sistema hem d'informar al subsistema gràfic quina és la funció que ha de cridar cada cop que s'hagi de dibuixar de nou a la pantalla. Les funcions de teclat i de IDLE també s'han d'especificar quines són. L'últim pas és iniciar el bucle infinit que domina qualsevol aplicació OpenGL. El fragment de codi següent mostra aquestes inicialitzacions.

```
glutInit(pargc, argv);
glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
glutInitWindowSize(GL_WIN_SIZE_X, GL_WIN_SIZE_Y);
glutCreateWindow ("Captura de moviment amb Kinect");

glutDisplayFunc(glutDisplay);
```

```
glutKeyboardFunc(glutKeyboard);
glutIdleFunc(glutIdle);

glutMainLoop();
```

Un cop ja tenim l'aplicació gràfica inicialitzada hem de configurar les variables d'OpenNI i NITE necessàries per a fer el seguiment del cos. El primer que farem es crear l'objecte g_Context dels tipus xn::Context i carregar-li la informació que conté un arxiu XML sobre les funcionalitats que volem utilitzar. El contingut d'aquest arxiu ha de coincidir amb les variables que es volen inicialitzar. En el nostre cas, hem utilitzat un XML de configuració que especifica l'ús de la imatge 2D i 3D del Kinect. Aquesta informació es carrega a les variables g_ImageGenerator i g_DepthGenerator respectivament i s'afegeixen a g_Context.

```
// Inicialització a partir del XML

xn::EnumerationErrors errors;
nRetVal = g_Context.InitFromXmlFile(SAMPLE_XML_PATH, &errors);
if (nRetVal == XN_STATUS_NO_NODE_PRESENT)
{
        XnChar strError[1024];
        errors.ToString(strError, 1024);
        printf("%s\n", strError);
        return (nRetVal);
}
elseif (nRetVal != XN_STATUS_OK)
{
        printf("Open failed (initFromXml): %s\n", xnGetStatusString(nRetVal));
        return (nRetVal);
}

nRetVal = g_Context.FindExistingNode(XN_NODE_TYPE_DEPTH, g_DepthGenerator);
CHECK_RC(nRetVal, "Find depth generator");

nRetVal = g_Context.FindExistingNode(XN_NODE_TYPE_IMAGE, g_ImageGenerator);
CHECK_RC(nRetVal, "Find image generator");

g_DepthGenerator.GetMetaData(g_depthMD);
g_ImageGenerator.GetMetaData(g_imageMD);
```

Tot seguit inicialitzem l'objecte g_UserGenerator. Aquest objecte és el que farà la segmentació d'usuaris i posteriorment el seguiment de l'esquelet. Després fem el registre dels callbacks associats a la generació d'usuaris. Aquests callbacks es divideixen en dos tipus: callbacks associats a la detecció d'usuaris i callbacks associats al seguiment de l'esquelet. Els primers són cridats quan el sistema detecta un nou usuari o perd a un usuari. Els segons s'executen quan es detecta la posició de calibració, comença la calibració i quan aquesta s'acaba. També hem de definir quin model d'esquelet es vol utilitzar. Hem utilitzat el model que ens facilita la llibreria. Per acabar, cridem al mètode StartGeneratingAll() de l'objecte g_Context. A partir d'aquest moment activem la recepció de dades de Kinect.

```
// Inicialització del UserGenerator

nRetVal = g_Context.FindExistingNode(XN_NODE_TYPE_USER, g_UserGenerator);
if (nRetVal != XN_STATUS_OK)
{
	nRetVal = g_UserGenerator.Create(g_Context);
	CHECK_RC(nRetVal, "Find user generator");
}

// Registre de callbacks per la detecció d'usuaris

XnCallbackHandle hUserCallbacks, hCalibrationCallbacks, hPoseCallbacks;
if (!g_UserGenerator.IsCapabilitySupported(XN_CAPABILITY_SKELETON))
{
	printf("Supplied user generator doesn't support skeleton\n");
	return 1;
}
g_UserGenerator.RegisterUserCallbacks(User_NewUser, User_LostUser, NULL,
hUserCallbacks);

// Registre de callbacks per al seguiment d'usuaris

g_UserGenerator.GetSkeletonCap().RegisterCalibrationCallbacks(UserCalibration_Calibratio
nStart, UserCalibration_CalibrationEnd, NULL, hCalibrationCallbacks);

if (g_UserGenerator.GetSkeletonCap().NeedPoseForCalibration())
{
	g_bNeedPose = TRUE;
	if (!g_UserGenerator.IsCapabilitySupported(XN_CAPABILITY_POSE_DETECTION))
	{
		printf("Pose required, but not supported\n");
		return 1;
	}
	g_UserGenerator.GetPoseDetectionCap().RegisterToPoseCallbacks(UserPose_PoseDetect
ed, NULL, NULL, hPoseCallbacks);
	g_UserGenerator.GetSkeletonCap().GetCalibrationPose(g_strPose);
}

g_UserGenerator.GetSkeletonCap().SetSkeletonProfile(XN_SKEL_PROFILE_ALL);


nRetVal = g_Context.StartGeneratingAll();
CHECK_RC(nRetVal, "StartGenerating");
```

3.3.2 CAPTURA DEL MOVIMENT

Com ja hem explicat anteriorment, la captura del moviment de l'usuari l'executa l'objecte g_UserGenerator. La captura del moviment està dirigida pels callbacks definits a la inicialització. El funcionament d'aquests callbacks es mostra al diagrama d'estats de la Figura 27. El sistema està contínuament buscant a un nou usuari, i quan aquest detecta un se li assigna una ID. El sistema activa la detecció de la captura de calibració per cada usuari que ha detectat. L'usuari es comença a calibrar quan es detecta la posició de calibració. Si el procés la calibració es completa s'inicia el seguiment de l'esquelet. En cas contrari, el sistema continua buscant la posició de calibració. Si per alguna raó el seguiment de l'esquelet deixa de funcionar l'usuari haurà de tornar-se a calibrar. També es possible que el sistema perdi a l'usuari si

aquest surt del camp de visió de la càmera. Llavors el sistema estarà buscant nous usuaris.

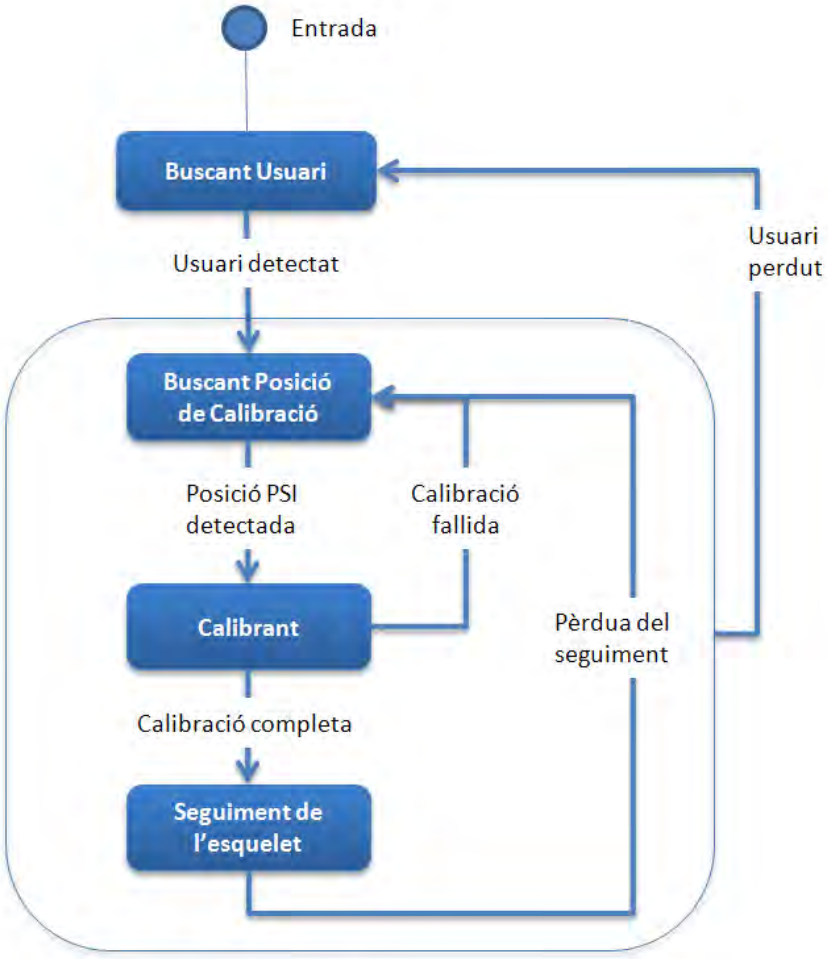

FIGURA 27 DIAGRAMA D'ESTATS DE LA SEGMENTACIÓ D'USUARI I POSTERIOR SEGUIMENT DE L'ESQUELET

A cada crida de la funció de pintat s'executa el codi següent per comprovar si el sistema està fent el seguiment d'algun usuari, i en cas positiu, s'accedeix a la postura del usuari que el sistema ha estimat.

```
// Read next available data
g_Context.WaitAndUpdateAll();
xn::SkeletonCapability sc = g_UserGenerator.GetSkeletonCap();
XnUserID aUsers[15];
XnUInt16 nUsers = 15;

g_UserGenerator.GetUsers(aUsers, nUsers);

for (int i = 0; i < nUsers; ++i)
{
	XnUserID player = aUsers[i];
```

```
        XnPoint3D com;
        g_UserGenerator.GetCoM(player, com);

        sc.GetSkeletonJointPosition(player,XN_SKEL_HEAD, captura->GetJoint(0));
        sc.GetSkeletonJointPosition(player,XN_SKEL_NECK, captura->GetJoint(1));
        sc.GetSkeletonJointPosition(player,XN_SKEL_TORSO, captura->GetJoint(2));
        sc.GetSkeletonJointPosition(player,XN_SKEL_LEFT_SHOULDER, captura->GetJoint(3));
        sc.GetSkeletonJointPosition(player,XN_SKEL_LEFT_ELBOW, captura->GetJoint(4));
        sc.GetSkeletonJointPosition(player,XN_SKEL_LEFT_HAND, captura->GetJoint(5));
        sc.GetSkeletonJointPosition(player,XN_SKEL_RIGHT_SHOULDER, captura-
>GetJoint(6));
        sc.GetSkeletonJointPosition(player,XN_SKEL_RIGHT_ELBOW, captura->GetJoint(7));
        sc.GetSkeletonJointPosition(player,XN_SKEL_RIGHT_HAND, captura->GetJoint(8));
        sc.GetSkeletonJointPosition(player,XN_SKEL_LEFT_HIP, captura->GetJoint(9));
        sc.GetSkeletonJointPosition(player,XN_SKEL_LEFT_KNEE, captura->GetJoint(10));
        sc.GetSkeletonJointPosition(player,XN_SKEL_LEFT_FOOT, captura->GetJoint(11));
        sc.GetSkeletonJointPosition(player,XN_SKEL_RIGHT_HIP, captura->GetJoint(12));
        sc.GetSkeletonJointPosition(player,XN_SKEL_RIGHT_KNEE, captura->GetJoint(13));
        sc.GetSkeletonJointPosition(player,XN_SKEL_RIGHT_FOOT, captura->GetJoint(14));

}

PintaEsquelet();
```

El primer que fem és cridar el mètode GetSkeletonCap del g_UserGenerator per guardar-nos la informació de la captura a la variable sc. A continuació fem un bucle pels usuaris que l'aplicació esta detecten i utilitzem les funcions GetCoM iGetSkeletonJointPosition per obtenir la posició del centre de masses de l'usuari i de l'articulació que s'especifiqui.

La informació de les posicions de les articulacions es guarda a la classe Capture. Aquesta classe conté els mètodes descrits a la Figura 28. Posteriorment es crida a la funció PintaEsquelet() que s'encarrega de pintar al FrameBuffer la representació de l'esquelet estimat. A la Figura 29 es pot veure l'esquelet estimat pel sistema.

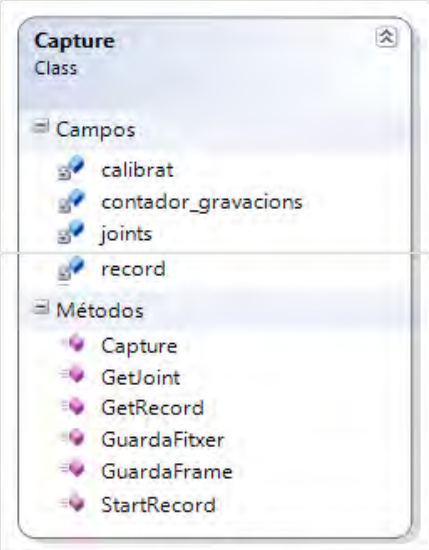

FIGURA 28 CLASSE CAPTURE

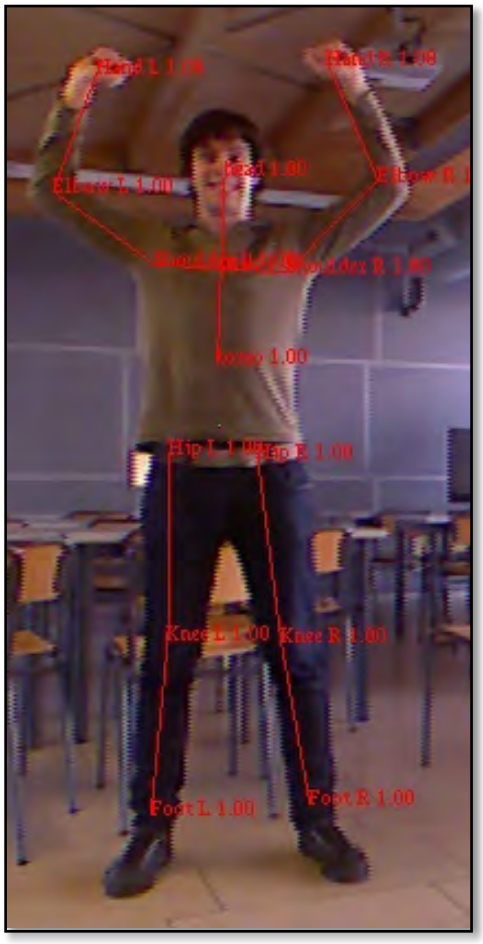

FIGURA 29 CAPTURA DE L'ESQUELET AMB EL KINECT

3.3.3 GUARDAR MOVIMENTS

Per tal de poder fer la validació biomecànica és necessari guardar les posicions estimades al llarg del temps en un fitxer. Hem utilitzat la funció *clock()*de la llibreria <*time.h*>per guardar en una variable el temps inicial a l'inici d'un moviment. Aquesta variable és start_time. Per saber quan de temps passa d'una iteració a l'altre utilitzem el fragment de codi següent.

```
current_time = clock();
accum_time = (current_time - start_time)* CLOCKS_PER_SEC;
```

En la variable accum_time ens guardem el temps acumulat de la iteració actual. Tenint aquesta informació el que hem de fer es guardar a cada iteració el temps acumulat i la posició de les articulacions. Hem programat l'inici i el fi de la gravació amb la tecla *r*

utilitzant el callback de teclat de GLUT. Cada cop que premem aquesta tecla s'executa el codi següent:

```
if (captura->GetRecord() == false) {
	start_time = clock();
	accum_time = 0.00;

	printf("Start recording\n");
	captura->StartRecord();
}else{
	captura->GuardaFitxer();
}
```

El mètode StartRecord() activa el mode de gravació. Si tenim el mode de gravació activat a cada iteració de la detecció de la postura fem una crida a GuardaFrame(accum_time,com). Aquest mètode s'encarrega d'anar acumulant cadenes de caràcters en un vector. Finalment, tota aquesta informació es bolcada en un fitxer de text quan es crida al mètode GuardaFitxer(). El format del fitxer de sortida es pot veure a la Figura 30.

```
Time COMX COMY COMZ HeadX HeadY HeadZ HeadC NeckX NeckY NeckZ NeckC
TorsoX TorsoY TorsoZ TorsoC ShoulderLX ShoulderLY ShoulderLZ
ShoulderLC ElbowLX ElbowLY ElbowLZ ElbowLC HandLX HandLY HandLZ
HandLC ShoulderRX ShoulderRY ShoulderRZ ShoulderRC ElbowRX ElbowRY
ElbowRZ ElbowRC HandRX HandRY HandRZ HandRC HipLX HipLY HipLZ HipLC
KneeLX KneeLY KneeLZ KneeLC FootLX FootLY FootLZ FootLC HipRX HipRY
HipRZ HipRC KneeRX KneeRY KneeRZ KneeRC FootRX FootRY FootRZ FootRC

0 -103.6 146.3 2504 -98.14 698 2452 1 -94.88 494.3 2474 1 -89.55 289
2499 1 -243.2 489 2462 1 -592.3 518.7 2485 1 -900.8 550.3 2507 1 53.44
499.6 2486 1 363.9 480.5 2514 1 654.2 456 2591 1 -180.3 83.58 2515 1 -
208.9 -308.8 2643 1 -246.2 -674.1 2643 1 10.09 89.41 2527 1 3.481 -302
2661 1 42.18 -676.7 2635 1

36000 -103 142.2 2505 -101.1 700 2458 1 -95.68 494.9 2477 1 -88.9
289.4 2499 1 -243.8 488.6 2463 1 -601.7 519.2 2488 1 -911.8 548.5 2517
1 52.46 501.3 2490 1 362.4 481.5 2515 1 654.4 454.5 2592 1 -179.3
80.32 2515 1 -208.3 -312.4 2642 1 -246.5 -674.6 2643 1 10.86 87.12
2531 1 3.609 -305.1 2662 1 40.82 -674.4 2636 1

74000 -104.3 144.5 2505 -103.5 701.6 2462 1 -96.58 496.2 2478 1 -88.8
290.6 2499 1 -244.5 489.1 2463 1 -609.8 518.5 2491 1 -920.1 543.5 2512
1 51.38 503.3 2493 1 360.8 484.1 2511 1 652.6 452.3 2587 1 -177.1
79.81 2513 1 -206.5 -309.6 2649 1 -245.4 -679 2644 1 12.84 87.95 2530
1 4.762 -303.2 2664 1 41.25 -674.6 2637 1
```

FIGURA 30 EXEMPLE FITXER KINECT

3.4 CAPTURES AL MEDIALAB

Les dades dels moviments que s'han utilitzat per dur a terme aquest projecte han estat capturades en un laboratori de captura de moviment , concretament en el MediaLab de la Universitat Ramon Llull - La Salle. En aquest laboratori s'hi troba una instal·lació d'última generació de captura de moviment, concretament un sistema òptic passiu de Vicon amb un total de 24 càmeres d'alta velocitat i resolució. El volum de captura és de 45 m2, amb la disponibilitat de capturar salts des d'una plataforma situada a 4 metres d'alçada.

Com ja s'ha explicat anteriorment, la captura dels moviments era necessària realitzar-la amb el Motion Capture i el Kinect a la vegada. Per tant, es va col·locar el Kinect dins de la sala i es va procedir amb el protocol de gravació habitual. A la Figura 31 es pot observar el muntatge esmentat.

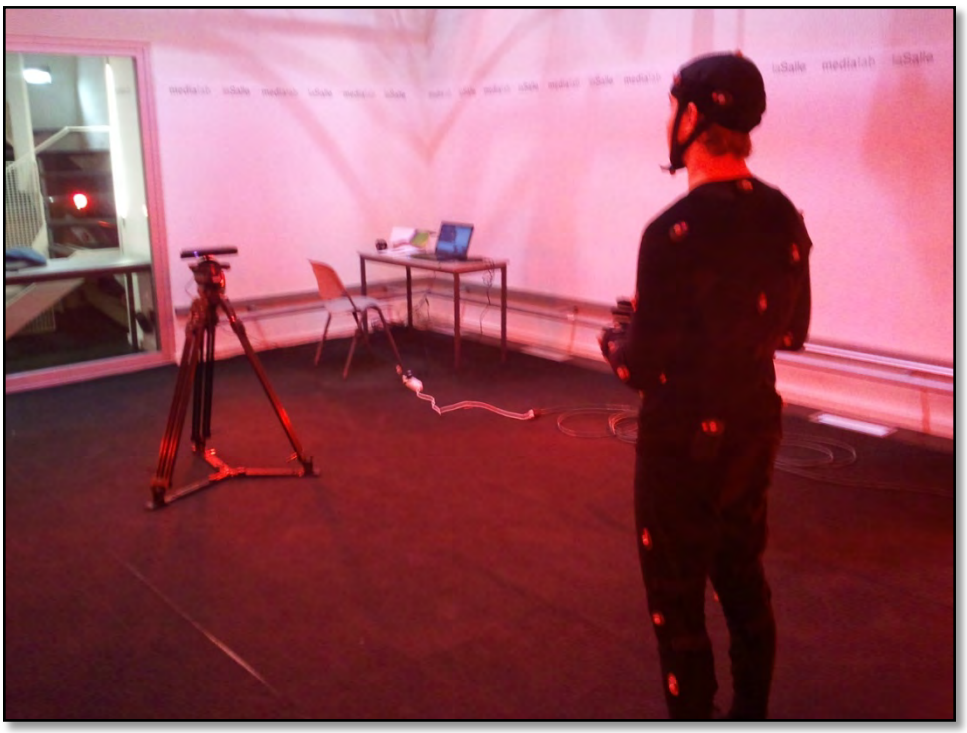

FIGURA 31 CAPTURES AL MEDIALAB

Les gravacions les ha realitzat el fisioterapeuta Bernat Pascual, expert en fisioteràpia física del projecte. D'aquesta manera es maximitza la correcte realització dels exercicis de rehabilitació. Les captures han estat enregistrades a una velocitat de gravació de 120 fotogrames per segon. Els exercicis gravats s'especifiquen en a Figura 32.

Articulació	Descripció	Arxiu Kinect	Arxiu Mocap
Cervicals	4 moviments cervicals	gravacio_1	cervicals001
	4 moviments cervicals	gravacio_2	cervicals002
	4 moviments cervicals	gravacio_3	cervicals003
	4 moviments cervicals	gravacio_4	cervicals004
Genoll	Fase 1.1	gravacio_3	Genoll001
	Fase 1.1	gravacio_4	Genoll002
	Fase 1.2	gravacio_5	Genoll003
	Fase 1.2	gravacio_6	Genoll004
	Fase 2.1.1 Flexió i extensió genoll	gravacio_7	Genoll005
	Fase 2.1.1 Flexió i extensió maluc	gravacio_8	Genoll006
	Fase 2.1.1 Add i abd maluc	gravacio_9	Genoll007
	Fase 2.1.2	gravacio_10	Genoll008
	Fase 2.2.2	gravacio_11	Genoll009
	Propio. Primera tanda	gravacio_12	Genoll010
	Propio. Primera tanda	gravacio_13	Genoll011
	Propio. Segona tanda	gravacio_14	Genoll012
	Propio. Tercera tanda	gravacio_15	Genoll013
	Propio. Quarta tanda	gravacio_16	Genoll014
Espatlla	Fase 1.1 Flexió i extensió	gravacio_1	Espatlla001
	Fase 1.1 Add i abd	gravacio_2	Espatlla002
	Fase 1.1 Add i abd horitzontal	gravacio_3	Espatlla003
	Exercicis pendulars	gravacio_4	Espatlla004
	Exercicis pendulars	gravacio_5	Espatlla005
	Exercicis pendulars	gravacio_6	Espatlla006
	Fase 2.1	gravacio_7	Espatlla007
	Fase 2.2	gravacio_8	Espatlla008
	Exercicis pendulars	gravacio_9	Espatlla009
	Fase 1.1 Flexió i extensió	gravacio_10	Espatlla010
	Fase 1.1	gravacio_11	Espatlla011
	Fase 1.1	gravacio_12	Espatlla012
	Fase 2.2	gravacio_13	Espatlla013
	Fase 2 Rotació	gravacio_14	Espatlla014

FIGURA 32 TAULA DE MOVIMENTS

Per tal de realitzar els moviments es necessari un protocol de gravació degut a que els dos sistemes funcionen per separat. En el cas del Motion Capture, cada cop que s'inicia una gravació l'actor ha de començar en T-Pose (Figura 33) i els tècnics del laboratori activen i paren el sistema a l'hora de gravar. En el cas de la captura amb Kinect, després de realitzar la calibració al principi, també es necessari activar i parar la gravació utilitzen la tecla "r" del teclat. Aquest desajust temporal s'ha corregit en la

fase de processat de les dades detectant en quin instant l'actor deixa d'estar en T-Pose (3.5.3 Sincronització).

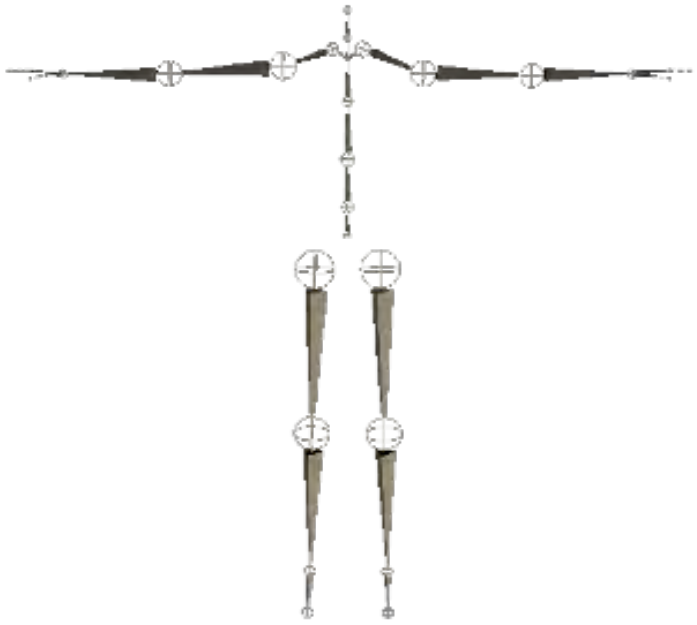

FIGURA 33 T-POSE

3.4.1 CONFIGURACIÓ DELS MARCADORS

Com ja s'ha explicat anteriorment, per realitzar la capturar de moviments es necessari que l'actor es col·loqui uns marcadors al cos. L'objectiu d'aquesta part del projecte és contrastar les dades del Mocap amb el Kinect, però per fer-ho hem de ser capaços de detectar la posició de les articulacions que detecta Kinect a partir dels marcadors col·locats sobre l'actor. Això ha fet necessari crear una nova configuració de marcadors. El que s'ha fet és col·locar dos marcadors a cada articulació que volem estudiar, aconseguint la seva posició fent el punt mig entre els dos marcadors. A la Figura 34 i Figura 35 podem veure imatges de la col·locació dels marcadors.

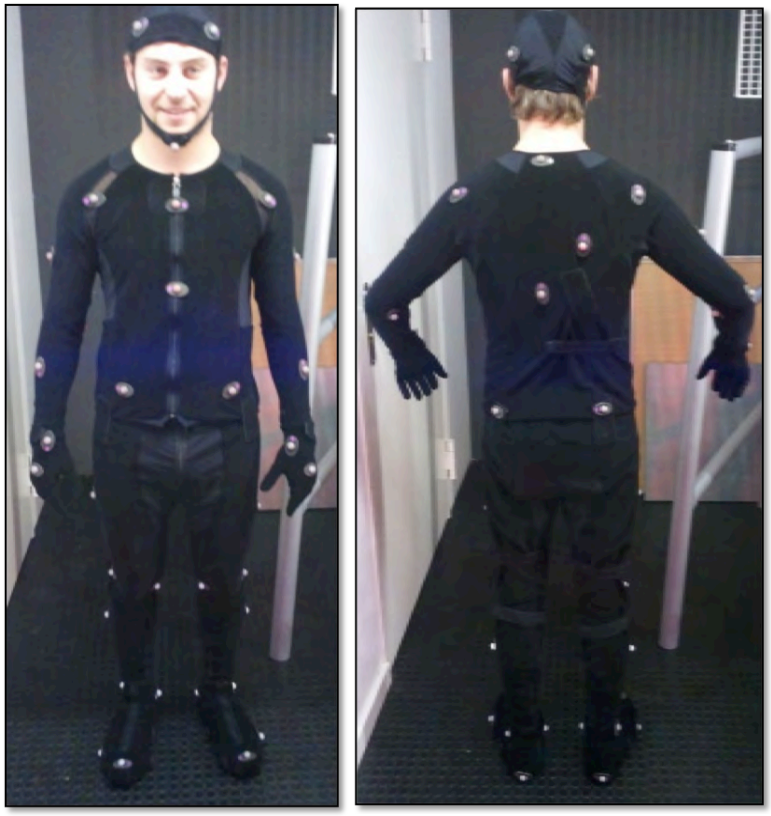

FIGURA 34 COL·LOCACIÓ DELS MARCADORS ÒPTICS

FIGURA 35 DETALL DE LA COL·LOCACIÓ DELS MARCADORS ÒPTICS

Aquesta configuració fa que l'arxiu de moviment generat de cada gravació contingui nous marcadors. A la Figura 36 hi ha un exemple de fitxer CSM que s'ha obtingut amb la configuració de marcadors mostrada a les figures anteriors. En negreta estan marcats els noms dels marcadors que hem hagut d'afegir.

```
$Filename Genoll001.csm
$Date 2011/03/22
$Time 13:56:39
$Actor kinect_Adso

$Comments
General Capture

$FirstFrame 1
$LastFrame 8604
$Rate 120.000

$Order
LFHD RFHD LBHD RBHD TOP BARB C7 T10 CLAV STRN RBAC LSHO LSHFLUPA LELB
LELA LFRM LWRA LWRB LFIN RSHO RSHFRUPA RELB RELA RFRM RWRA RWRB RFIN
LFWT RFWT LBWT RBWT LTHI LKNE LKNF LSHN LANK LANI LHEE LTOE LMT5 RTHI
RKNE RKNF RSHN RANK RANI RHEE RTOE RMT5

$Points
1     -136.044647  -2293.646729  727.491943     -0.505111  -2301.416992
729.946533  -144.749832 -2410.188965 654.616699  9.252156 -2407.347656
654.622498     -71.978172   -2411.481445   747.195679     -65.748276   -
2204.831543 540.394104  -91.066322 -2416.068848 492.846161  -93.168724
-2468.724365  245.104706     -67.564377   -2204.552979  248.540268     -
65.571144 -2240.105225 413.451477  -11.753707 -2481.898438 339.064392
-236.187759   -2427.231934   465.654236      -245.046982   -2293.756836
465.846405     -384.188690  -2395.769531  453.468964     -522.519104  -
2420.630371  408.216217     -509.180389  -2377.678467  336.317322     -
622.679382   -2382.923096   443.885620      -791.984985   -2353.643555
420.520294     -781.681335  -2434.825684  404.396515     -845.189209  -
2400.119873 432.301788  75.532555 -2449.333252 448.871002  87.292717 -
2293.563232 449.275818  233.448593 -2429.919189 446.621521  360.836761
-2467.774414    383.984497       363.410828    -2399.557373    337.836548
464.724976 -2441.245850 439.901215  630.615295 -2431.384766 431.432800
622.755798 -2496.235352 407.036407  688.222839 -2474.204346 427.134705
-181.580521  -2240.513184  10.806158   38.818012  -2246.929199  13.831056
-198.522537 -2458.853027 -23.463684  38.973145 -2476.672119 -11.709346
-262.952423  -2348.287598  -320.492249     -240.085876  -2378.631592  -
515.640015     -135.201675  -2368.154297  -543.781494     -261.834106  -
2437.059570  -659.268066     -240.542496  -2460.808350  -914.092957     -
143.642593  -2447.774658  -916.506042     -205.480820  -2529.282715  -
993.611694     -237.514954  -2252.000977  -959.845459     -290.095947  -
2334.667969   -981.362610      126.741318   -2351.777344   -274.773132
105.277145   -2398.052734   -505.325500      -0.906531   -2371.976807   -
545.090210     124.945358   -2449.522705   -654.358337     107.566116   -
2463.667725 -888.987366  18.947880 -2434.490967 -911.252014  65.848480
-2532.266602   -987.905823      177.341629   -2350.198975   -999.401367
131.898438 -2264.779053 -964.322449
```

FIGURA 36 EXEMPLE FITXER MOCAP

3.5 PROCESSAMENT DE LES DADES

Per tal de poder comparar els valors del Kinect i del Mocap hem tenir en compte varis factors. Aquests factors són:

- Aconseguir la posició de les mateixes articulacions: Per una banda tenim les posicions de les articulacions, i per l'altre la posició dels marcadors.
- Coherència temporal: Les animacions del Kinect estan gravades a 30 fotogrames per segon i les del Mocap a 120 fotogrames per segon. A més a més l'inici i el fi d'aquestes no està sincronitzat.
- Coherència espaial: Els sistemes de coordenades dels dos sistemes de captura són diferents.

Per aconseguir aquests factors s'han seguit el procediments esmentats en la Figura 37, diferents per les dades del Kinect i les del Mocap. L'implementació dels mòduls s'ha dut a terme amb el software Matlab, ja que incorpora moltes funcionalitats matemàtiques per treballar amb vectors i matrius que són de gran ús per a realitzar les tasques descrites.

Per carregar els arxius en memòria s'ha utilitzat un arxiu Excel on està emmagatzemada la informació de les gravacions especificant el nom i el path. Per fer-ho s'ha utilitzat la crida següent:

```
[NUMERIC,TXT,RAW]=xlsread('Motions.xls','Hoja1','a1:e34');
```

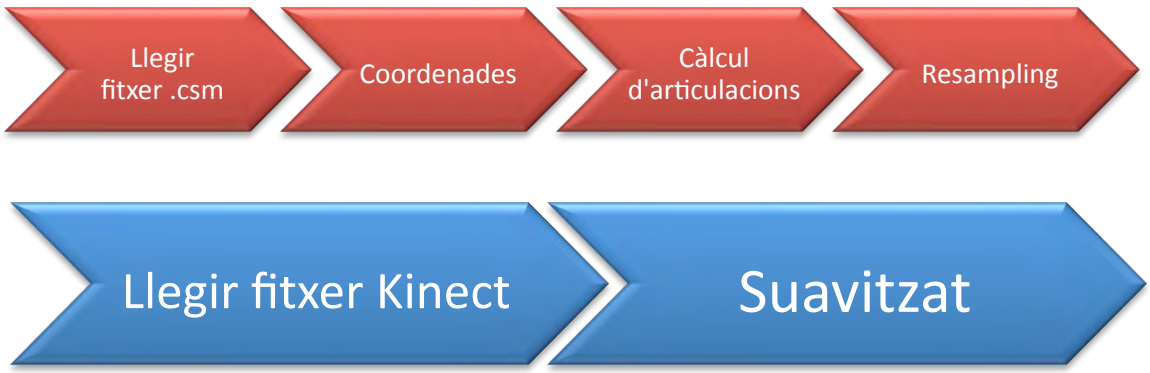

FIGURA 37 PROCÉS DELS ARXIUS DE MOCAP I KINECT

Després dels procediments que se'ls hi aplica a cada parella d'animacions és necessari fer una sincronització temporal perquè els moviments es realitzin al mateix moment. Un cop ja tinguem les dues animacions en el mateix moment temporal ja podrem calcular els errors.

3.5.1 PROCÉS ARXIUS CSM

Com ja s'ha explicat al punt anterior, els dos sistemes de captura tenen diferents sistemes de coordenades. Aquests sistemes són els que es poden veure a la Figura 38.

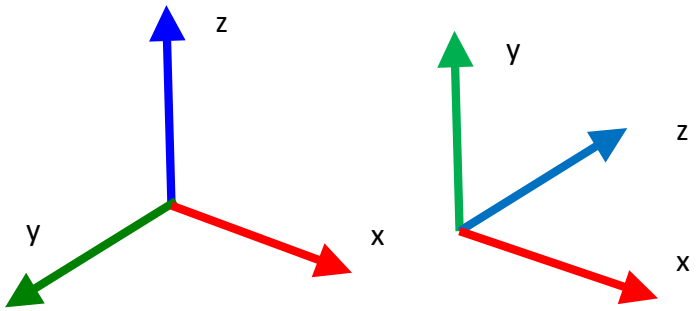

FIGURA 38 A LA DRETA, EL SISTEMA DE COORDENADES DEL MOCAP; I A L'ESQUERRA, EL SISTEMA DE COORDENADES DEL KINECT

S'ha modificat el sistema de coordenades de les animacions Mocap intercanviant els valors de les coordenades amb el següent codi:

```
matrixMocapEixos = matrixMocap;

for i=2:3:size(matrixMocapEixos,2)
     matrixMocapEixos(:,i)   = -matrixMocap(:,i+1); %x -> -y
     matrixMocapEixos(:,i+1) = matrixMocap(:,i+2); %y -> z
    matrixMocapEixos(:,i+2) = matrixMocap(:,i); %z -> x
end
```

A continuació calcularem la posició de les articulacions que ens dona el Kinect a partir de les dades del Mocap. Com ja hem explicat en el

3.4.1 Configuració dels marcadors, s'ha utilitzat una configuració especial de marcadors on les articulacions de l'esquelet del Kinect es poden calcular fent el punt mig entre parelles de marcadors. Les posicions de les articulacions han estat calculades amb les relacions següents.

Marcador A	Marcador B	Articulació Kinect
TOP	BARB	Head
CLAV	CLAV	Neck
STRN	STRN	Torso
RSHO	RSHF	ShoulderR
RELB	RELA	ElbowR
RFIN	RFIN	HandR
LSHO	LSHF	ShoulderL
LELB	LELA	ElbowL
LFIN	LFIN	HandL
RFWT	RBWT	HipR
RKNE	RKNF	KneeR
RANK	RANI	FootR
LRWT	LBWT	HipL
LKNE	LKNF	KneeL
LANK	LANI	FootL

FIGURA 39 TAULA DE LA RELACIÓ ENTRE ELS MARCADORS A I B AMB LES ARTICULACIONS DE KINECT

El següent pas és baixar el framerate de l'animació de 120 fotogrames per segon a 30. Hem utilitzat interpolació lineal i hem reduït el framerate en un factor de 0.25. El codi següent mostra aquest procediment.

```
factor = 0.25;
t = 1:1:size(motion_curve,1);
increment = 1/(ceil(size(motion_curve,1)*factor) - 1);
t2 = 1:increment:1;
motion_curve_warped = interp1(t,motion_curve,t2);
motion_curve_warped = motion_curve_warped';
```

Arribats a aquest punt tenim l'animació del Mocap representada de la mateixa manera que l'animació del Kinect. Les dues animacions realitzen el mateix moviment però estan desfasades en temps. Aquest retard és el que s'ha de corregir a la sincronització dels fitxers.

3.5.2 Procés arxius Kinect

Les dades dels arxiu Kinect després de carregar-se en memòria se'ls hi ha aplicat un suavitzat per eliminar el soroll local. Per tal de suavitzar les dades s'ha utilitzat el mètode Lowess. El nom Lowess ve del terme en anglès "locally weighted scatter plot smooth", que significa suavitzat d'un gràfic dispersat a nivell local i ponderat. El procés de suavitzat es considera local perquè cada valor suavitzat s'obté a partir de punts veïns. El nombre de punts veïns està determinat per un valor que és l'extensió (en anglès, span). El procés és ponderat perquè els punts que se'ls hi aplicarà la regressió tenen pesos assignats, per tant, alguns tenen més importància que d'altres. Aquest mètode utilitza polinomis de primer grau per fer la regressió.

Per implementar aquest mètode he utilitzat la funció smooth fent la crida següent:

```
curve = smooth(curve,15,'rlowess');
```

Els paràmetres seleccionats fan que el mètode tingui una extensió de 15 valors, utilitzi polinomis de primer grau i una funció de pes bicúbica. La versió utilitzada és la robusta, que el que fa es posar uns pesos de 0 a les dades que superen una desviació mitjana absoluta de 6. A la Figura 40 tenim un exemple del resultat del suavitzat de les dades.

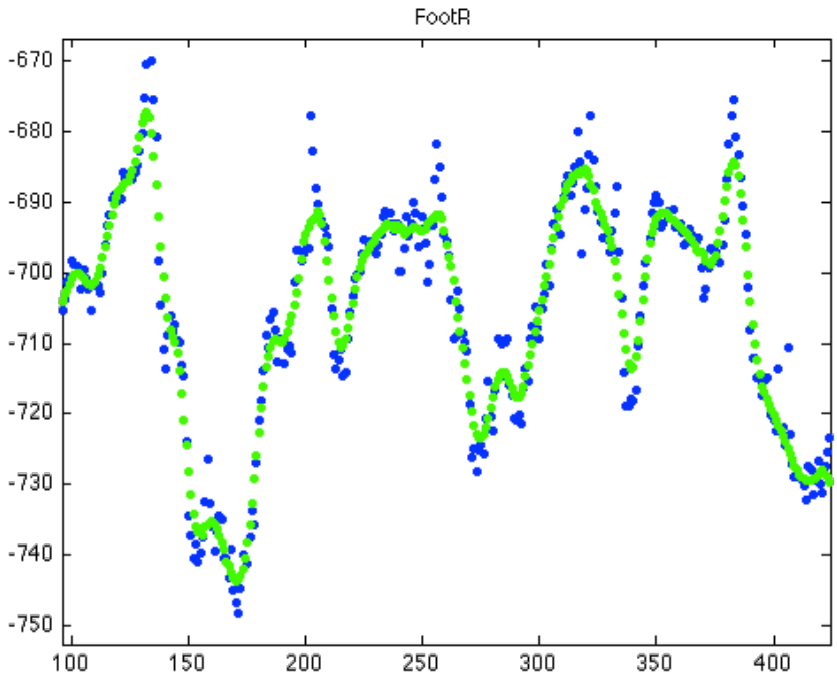

FIGURA 40 GRÀFICA DE LA POSICIÓ Y DEL PEU DRET, DE COLOR BLAU ELS VALORS ORIGINALS I DE COLOR VERD ELS VALORS SUAVITZATS

3.5.3 SINCRONITZACIÓ

El procés de captura genera dos tipus d'arxius, els que provenen del Mocap i els que provenen de la captura amb Kinect. Cada arxiu Mocap conté la mateixa sessió de captura que un arxiu Kinect. Tot i així, els arxius no comencen exactament al mateix instant de temps, i per tant estan desincronitzats temporalment. Les animacions estan relacionades per parelles, però la realització dels moviments no està sincronitzada. La sincronització entre les dues animacions és molt important per poder comparar les dades. Per fer-ho s'ha tret benefici de la posició T-Pose (Figura 33) que es fa a l'inici de cada gravació.

A la Figura 41 podem veure les posicions i velocitats Y de la mà esquerra en ambdós animacions. Visiblement ja podem observar que estan desincronitzades. Si ens fixem en les dues imatges es pot veure que hi ha un increment de velocitat quan hi ha un descens de la posició Y. Aquest és el moment en que l'actor deixa de estar quiet en T-Pose i comença l'acció.

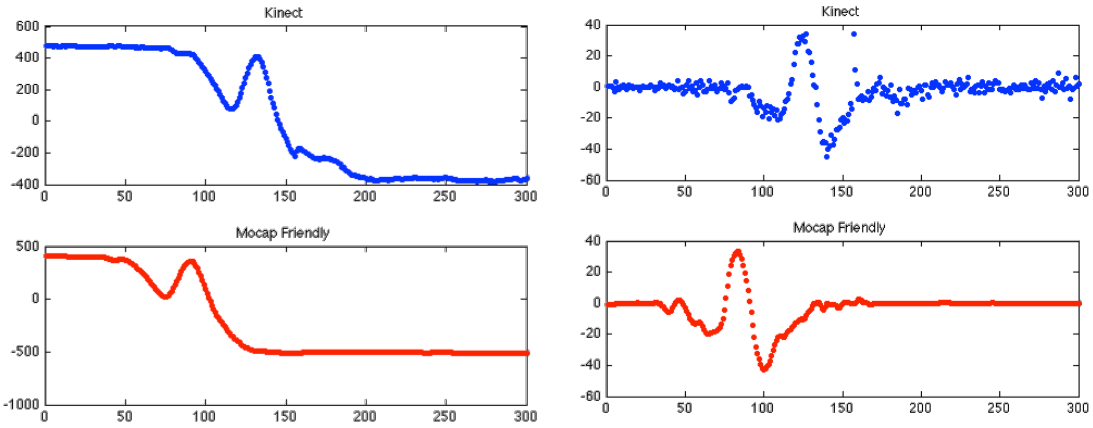

FIGURA 41 GRÀFIQUES DE LA POSICIÓ (ESQIERRA) I VELOCITAT (DRETA) Y DE LA MÀ ESQUERRA

Per buscar el moment en que hi ha aquest increment de velocitat s'ha buscat el primer frame en que la velocitat és superior a 10. Aquest nombre ha estat calculat empíricament. A la Figura 42 podem veure el resultat d'aquest procediment. Cal dir que no totes les parelles d'animacions han estat sincronitzades amb èxit i ha calgut un ajust manual.

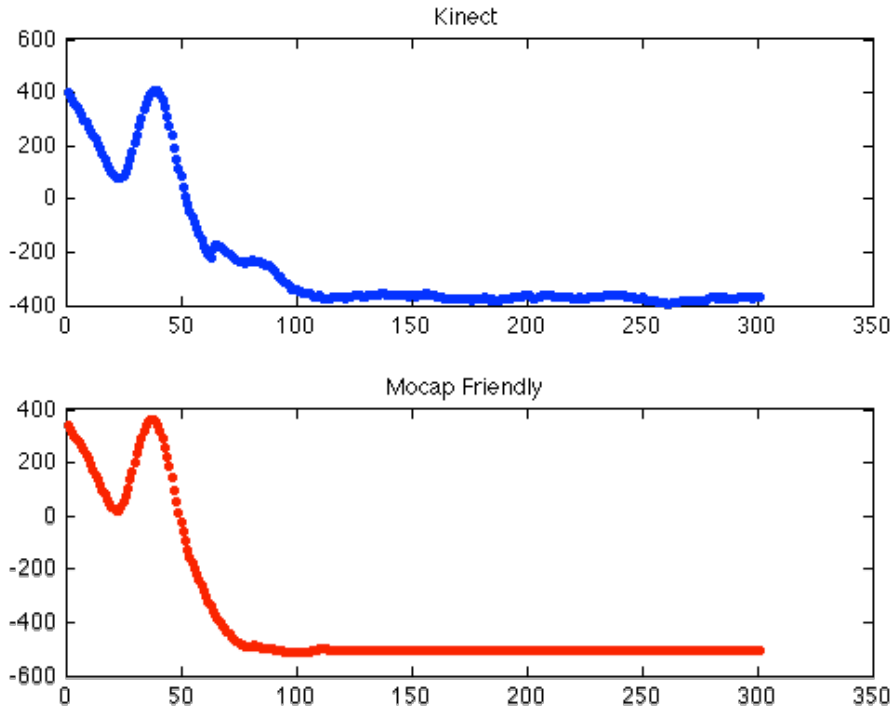

FIGURA 42 GRÀFIQUES SINCRONITZADES DE LA POSICIO Y DE LA MÀ ESQUERRA

3.5.4 CÀLCUL DE LES ROTACIONS

Ara ja tenim les posicions de les articulacions donades pels dos sistemes de captura, però el que volem comparar són les rotacions. Calcularem les rotacions de les articulacions respecte la seva articulació pare. D'aquesta manera evitarem tenir que posar en correspondència els dos sistemes de coordenades dels dos sistemes de captura.

Les rotacions que volem comparar en aquest projecte són las del genoll i el maluc. Compararem les rotacions en exercicis on aquestes siguin sobre un pla del cos (Figura 15), és a dir, tinguin un sol grau de llibertad. Assumint aquests requisits podem utilitzar la fórmula següent:

$$\cos\theta = \frac{|<a,b>|}{||a||.||b||}$$

FÒRMULA 1 ANGLE ENTRE DOS VECTORS

Aquesta fórmula ens diu l'angle θ que hi ha entre dos vectors (a i b). A la Figura 43 hi ha una representació dels vectors que hem d'utilitzar per calcular els angles buscats.

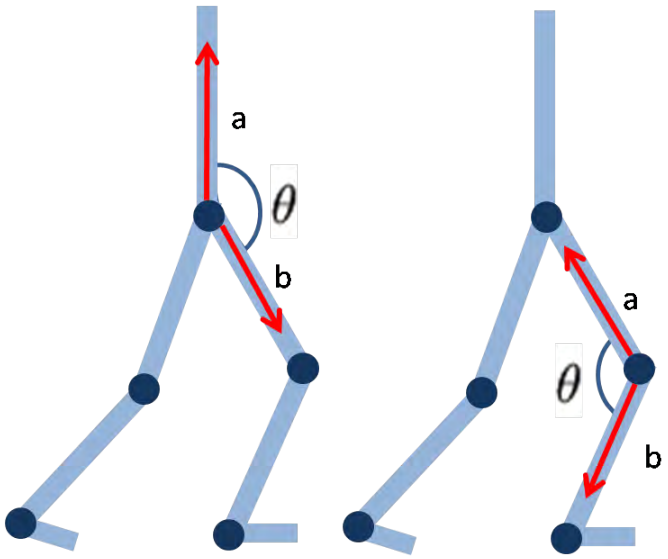

FIGURA 43 CÀLCUL DE LES ROTACIONS DEL GENOLL I EL MALUC

3.6 VISOR D'ANIMACIONS I DADES

Per tal de poder fer una avaluació subjectiva dels resultats i poder comparar a cop de vista les animacions aparellades s'ha implementat un visor. Com es pot veure a la Figura 44, aquest visor es divideix en dues parts: Visor de l'animació i visor de les dades.

El visor de l'animació consta de la representació de les posicions de les articulacions en un espai 3D i d'un conjunt de botons. La representació de les posicions es fa distingint les dues animacions que s'estan mostrant amb diferents colors. De color vermell tenim les animacions del Mocap i de color blau les animacions del Kinect. Els botons ens permeten interactuar amb les animacions, permetent-nos fer *play* (reproduir), *stop* (aturar), *forward* (avançar un fotograma), *previous* (retrocedir un fotograma). També hi ha un botó de sortir per abandonar la visualització.

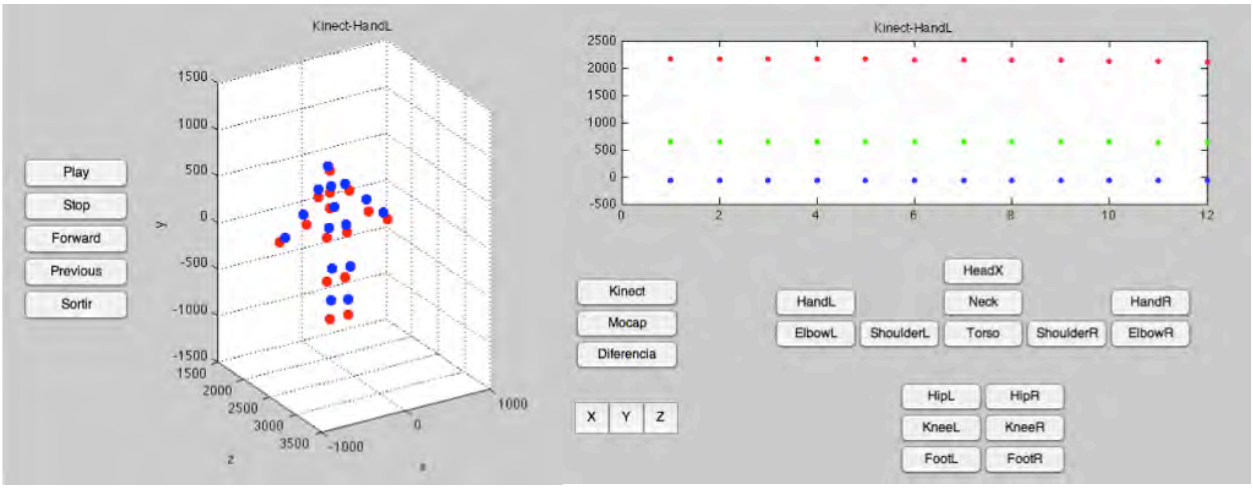

FIGURA 44 A LA PART ESQUERRA, EL VISOR D'ANIMACIONS I A LA PART DRETA, EL VISOR DE DADES

En la part de visualització de les dades es poden veure gràfiques amb les coordenades de posició de les articulacions capturades. Es pot seleccionar si es vol veure les dades de l'animació Kinect o de Mocap, inclús la diferencia entre ambdues dades. També es pot seleccionar i desseleccionar les coordenades que es vulguin. Per poder escollir de quina articulació volem visualitzar les dades hi ha uns botons en forma de cos.

3.7 RESULTATS

Per cada animació i articulació a analitzar hem definit els següents errors entre els angles calculats.

- **Error absolut:** Correspon a la diferència (en valor absolut) entre el valor del Mocap i el valor del Kinect. Aquest error es calcula per cada fotograma.
- **Error absolut respecte el rang (EAR):** Correspon al quocient entre l'error absolut i el rang de valors Mocap de l'articulació en concret. Aquest error es calcula per cada fotograma.
- **Error absolut mig respecte el rang (EAMR):** Correspon a la mitjana dels errors absoluts respecte el rang de cada fotograma. Aquest error es calcula per tota l'animació.

A la Figura 45 hi ha un exemple dels errors esmentats. A la cinquena fila de la gràfica hi ha el factors de confidència de cada l'articulació en concret, l'articulació pare i filla. Aquesta dada s'ha utilitzat per eliminar les dades on algun d'aquests factors val 0.

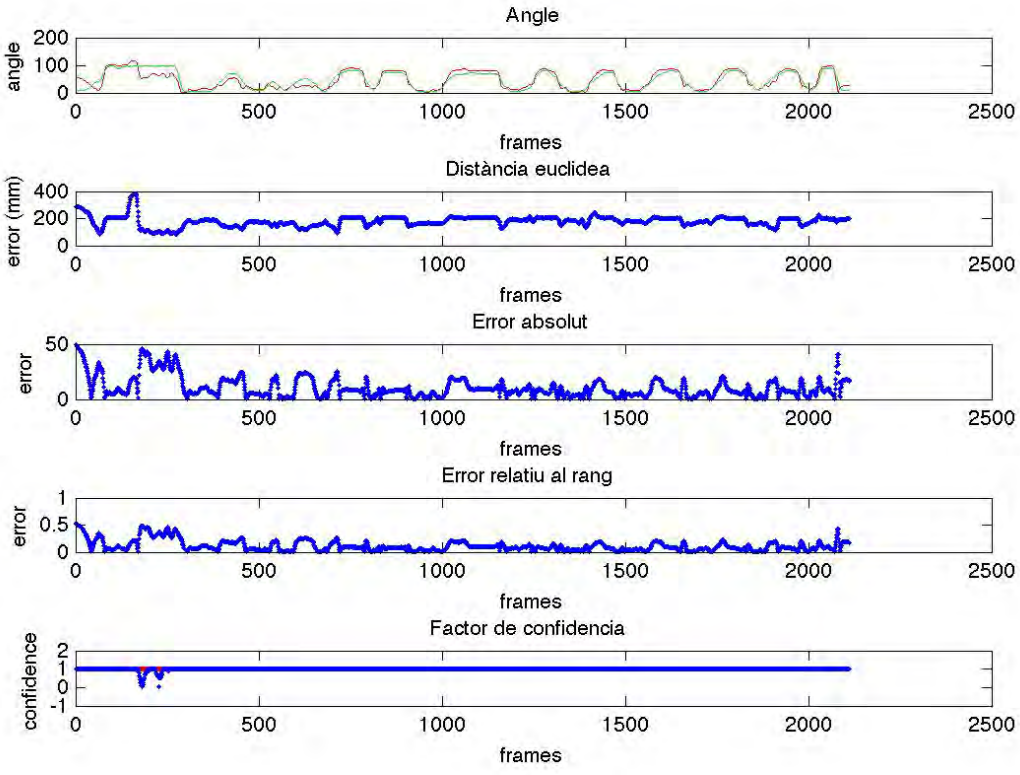

FIGURA 45 GRÀFIQUES DEL GENOLL DRET EN LA GRAVACIO_3

A la Figura 46 i Figura 47 estan els resultats del genoll i del maluc. En aquestes taules es mostra el rang articulatori, l'error absolut mig respecte el rang, l'error absolut mig respecte el rang eliminant les mostres on el factor de confidència és igual a 0; i el tant per cent de confidència de les mostres de cada animació. En l'última columna es mostra l'error mig en graus de cada animació. Aquest valor s'ha calculat multiplicant el EAMR i el rang articulatori.

Animació (nom Kinect)	Rang articulatori	EAMR	EAMR (conf. = 1)	% Confidència	Error Mig
Gravacio_3	94.01°	11.375 %	11.375 %	100 %	10,69°
Gravacio_4	93.40°	8.5054 %	8.5054 %	100 %	7,94°
Gravacio_5	93.57°	7.487 %	7.5434 %	98.72 %	7,01°
Gravacio_6	87.58°	18.186 %	18.182 %	99.93 %	15,95°
Gravacio_7	115.07°	11.978 %	8.5943 %	84.83 %	13,77°
Gravacio_8	29.34°	22.505 %	22.505 %	100 %	6,60°
Gravacio_9	20.76°	25.393 %	25.393 %	100 %	5,07°

FIGURA 46 TAULA DE RESULTATS DEL GENOLL

Els rangs articulatoris del genoll varien entre 20,76° i 115,06°. Les primeres 5 animacions analitzades es mouen en la franja 87,58° i 115,07, sent l'error mig màxim de 15,95° i l'error mig mínim de 7,94°. En les dues últimes animacions el rang articulatori es troba entre els valors 20,76° i 29,34°, donant com a resultat errors mig de 6,60° i 5,07°. Si analitzem els resultats, ens adonem que en la majoria d'animacions l'erro mig és inferior a 10°.

Animació (nom Kinect)	Rang articulatori	EAMR	EAMR (conf. = 1)	% Confidència	Error Mig
Gravacio_3	77.52°	10.131 %	10.131 %	100 %	7,85°
Gravacio_4	78.55°	10.958 %	10.958 %	100 %	8,27°
Gravacio_5	86.18°	6.3585 %	6.3588 %	100 %	5,89°
Gravacio_6	89.96°	6.8417 %	6.8424 %	99.92 %	6,15°
Gravacio_7	32.28°	22.327 %	22.655 %	84.83 %	7,20°
Gravacio_8	44.61°	20.672 %	20.672 %	100 %	9,22°
Gravacio_9	38.63°	16.804 %	16.804 %	100 %	6,49°

FIGURA 47 TAULA DE RESULTATS DEL MALUC

Pel que fa el maluc els rangs articulatoris oscil·len entre 32,28° i 89,96°. Els errors mig per les animacions analitzades es troben situats entre 5,59 ° i 9,22°.

Considerem bons els resultats obtinguts ja que els fisioterapeutes tenen un error d'apreciació que es troba al voltant dels 10°. En la majoria de casos, els valors obtinguts per la captura de moviment amb Kinect són millors.

4. APLICACIÓ PACIENT

4.1 INTRODUCCIÓ

L'aplicació pacient és un serious game que té com a objectiu facilitar la rehabilitació del genoll. Aquesta aplicació consta de dues parts: Servidor i Client. Aquestes parts s'executen en local en un mateix ordinador sota la plataforma Windows. Dividint l'aplicació així separem detecció de gestos i moviments mitjançant el Kinect de la lògica i imatge del serious game, a això ens permet programar les dues parts en diferents llenguatges de programació.

La part del Servidor és l'encarregada de gestionar les dades d'entrada que ens facilita Kinect. Hem utilitzat OpenNi i NITE per capturar els gestos de l'usuari i la seva postura quan aquest realitza un exercici. La programació d'aquesta part s'ha fet en C++. El Servidor es comunica amb el Client per informar dels events generats per l'usuari i enviar les dades de la captura de moviment. Aquesta comunicació es realitza mitjançant el protocol UDP. Aquesta aplicació també comparteix la imatge de color del Kinect mitjançant memòria compartida.

La part del Client és on s'executa el serious game. El serious game ha estat programat en Python utilitzant Panda3D, un game engine molt utilitzat per al prototipatge ràpid. L'aplicació rep un fitxer de configuració, en format XML, que defineix i configura els jocs/exercicis que el pacient haurà de realitzar. L'informació relativa a les sessions, com per exemple, el número de repeticions o la data de la sessió, s'emmagatzema a una base de dades implementada en SQLlite. El Client es qui controla l'inici i el fi dels exercicis ha realitzar. Per tant, envia trames UDP cap al Servidor per que aquest generi fitxers de text del moviment 3D capturat i vídeos. La generació de vídeos s'ha fet utilitzant la llibreria FFMPEG.

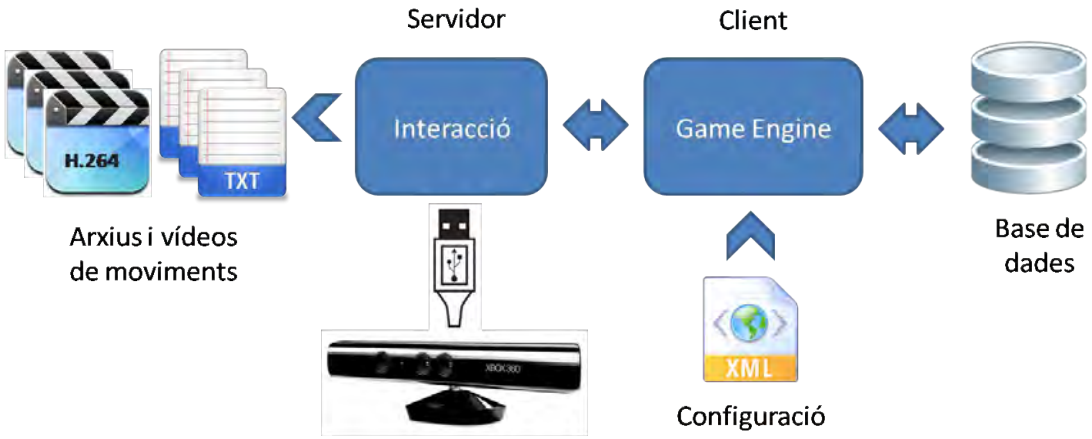

FIGURA 48 DIAGRAMA DE L'APLICACIÓ PACIENT

4.2 CONTROL GESTUAL AMB NITE

El NITE middleware té com a una de les seves funcionalitats més rellevant el control per gestos. El control gestual es fa fent el seguiment del moviment de la mà. El sistema funciona correctament mentre la mà roman dins del camp de visió de la càmera i no està closa per altres usuaris i/o objectes. Marxar del camp de visió significa perdre el control. Perquè aquest control funcioni de forma correcte es recomana situar-se entre 1 i 3,5 metres de distància a la càmera.

4.2.1 SESSION MANAGER

El control gestual es gestiona amb sessions. L'usuari inicia una sessió quan el sistema detecta un *focus gesture* (gest d'enfocament). N'hi ha dos tipus: *click* i *wave*. El gest de *click* consisteix en mostrar el palmell de la mà a càmera i moure la mà en direcció al sensor, i justament després retirar-la cap a un mateix. El gest *wave* consisteix en mostrar el palmell i moure la mà diversos cops d'esquerra a dreta i vicebersa. Durant la sessió, es fa el seguiment del punt de la mà assignant-li un identificador únic. Mentre un usuari està en sessió, cap altre usuari el pot agafar fins que la aplicació perd la mà, hi ha una oclusió, marxem del camp de visió o hem programat una manera de sortida. A la Figura 49 es mostra aquest procés.

Hi ha 3 possibles estats de la sessió:

- **Fora de sessió**: En aquest estat no hi ha cap sessió activa. El sistema esta contínuament escanejant l'escena per detectar un focus gesture. Quan el sistema detecta aquest gest,'estat canvia cap a en sessió.
- **En sessió**: En aquest estat el sistema esta fent el seguiment de la mà que controla l'aplicació.
- **Quick refocus**: Aquest és un estat de transició. Quan estem en sessió i el sistema per la posició de la mà hi ha un període de temps en que no tenim el control. En aquest temps el sistema pot reactivar la sessió si detecta un quick refocus gesture. Els focus gestures normals també poden reactivar la sessió. Un cop a transcorregut el temps l'estat canvia a Fora de sessió. L'estat Quick Refocus és opcional.

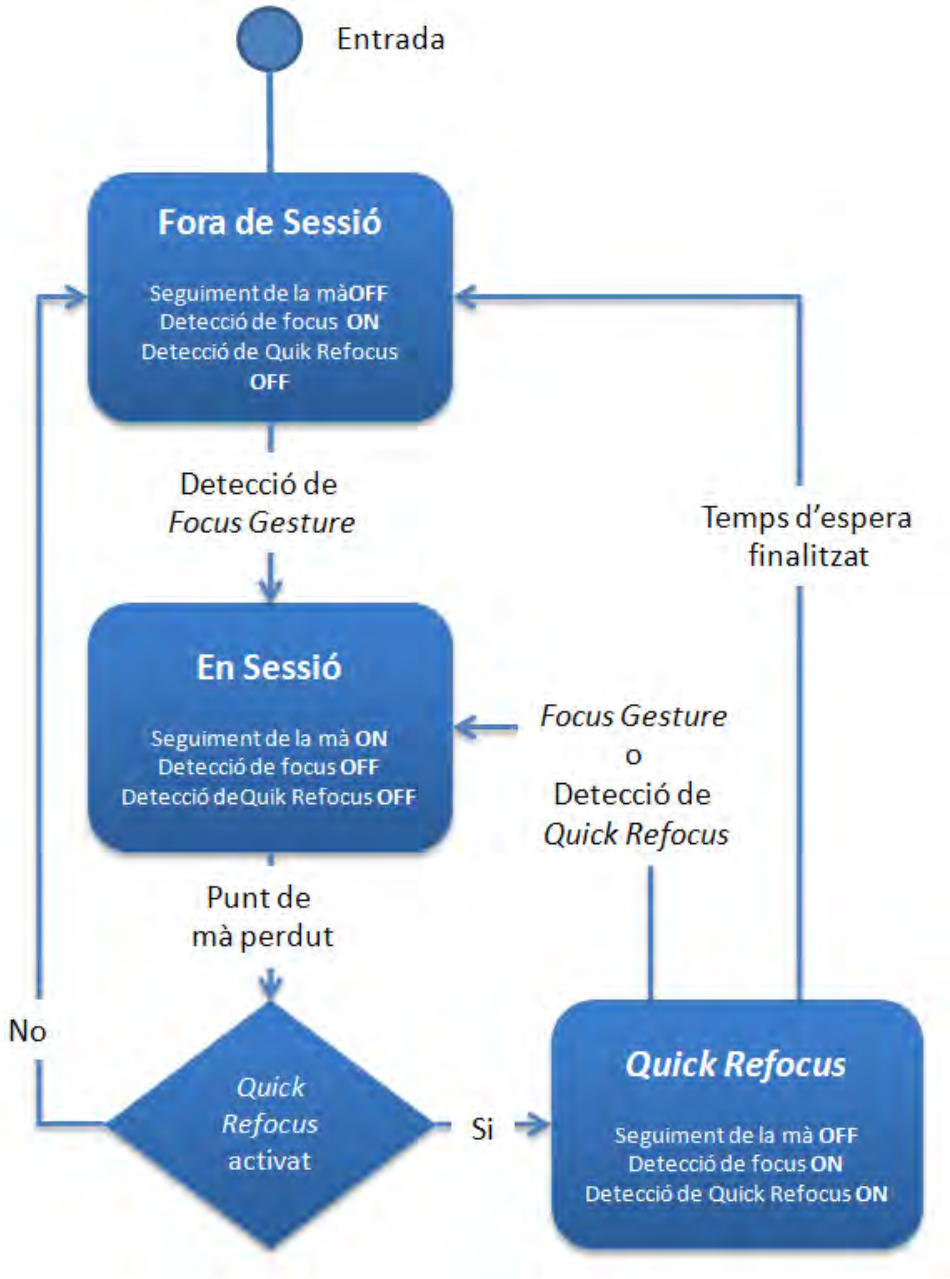

FIGURA 49 ESTAT DE LES SESSIONS

4.2.2 POINT CONTROLS

Els *Point Controls* són un conjunt d'objectes que utilitzen els punts de la mà detectats pels algorismes de NITE. A cada fotograma reben un flux dels punts de la mà actius d'alguna font, l'analitzen i realitzen alguna acció. Els *Point Controls* intenten extreure informació rellevant respecte el comportament dels punts de la mà. En la implementació actual, tots els *Point Controls* funcionen amb un únic punt, apart del *Steady Detector*.

4.2.2.1 PUNT PRIMARI

Tots els *Point Controls* del NITE s'alimenten amb punts que pertanyen a una mà en concret. Aquesta mà està definida com a mà activa, o punt primari. El punt primari és el primer punt de la mà reconeguda (possiblement la que ha fet el focus gesture). Si el punt primari no està disponible, i existeixen altres punts, un d'aquests punts passa a ser el punt primari. El punt primari és definit pel *Session Manager*.

4.2.2.2 EVENTS

Cada *Point Control* implementa un conjunt d'events dels quals s'han de registrar callbacks. A cada fotograma, tots els events registrats són processats. Es poden registrar els següents events per tots els punts de control.

1. Nou punt creat
2. Punt existent en moviment
3. Punt desaparegut
4. Punt primari creat
5. Punt primari en moviment
6. Punt primari intercanviat. El punt primari no està disponible, però un altra punt existeix i es converteix en el punt primari.
7. Punt primari destruït. Això vol dir que no hi ha més punts.
8. Sense punts

4.2.2.3 GESTOS

Quan realitzem un gest podem saber l'estat d'aquest, el tipus de gest que és i la localització del mateix. L'estat del gest pot ser *gesture started* (gest començat) o *gesture completed* (gest complet). Els tipus gestos que es poden detectar són:

- *Push* : Aquest punt de control reconeix el gest d'apretar fet per la mà que està essent seguida. Un push es detecta quan una certa velocitat es assolida i un angle proper a l'eix de la z per un cert període de temps. Aquest gest és típicament usat per seleccionar opcions.
- *Swipe left/right/up/down*: Aquest punt de control reconeix el moviment d'escombrat de la mà, ja sigui cap amunt, avall, dreta o esquerra. El moviment d'escombrat és un moviment curt en una direcció específica seguit d'una suspensió de la mà.
- *Steady*: Aquest punt de control detecta quan el punt de la mà està suspès en una posició durant un temps. La detecció té en compte el petit moviment de la mà quan l'usuari vol mantenir-la quieta. Steady és molt útil per separar gestos ja que una suspensió de la mà és potencialment l'inici i el fi d'un gest.

- *Wave:* Aquest punt de control intenta detectar el moviment d'ona. Aquest moviment és un nombre de canvis de direcció en un temps finit. Per defecte, 4 canvis de direcció són necessaris per identificar un *wave.*
- Hi ha altres punts de control com el detector de cercles i detectors relacionats amb elements d'interfície d'usuaris que ofereix NITE.

4.2.3 Objectes *Flow* i l'arbre de NITE

Els controls de NITE permeten definir el flux de l'aplicació, gestionar la concurrència entre controls i definir condicions per activar-los. Si creem *Point Controls* i els assignem tots al *Session Manager,* aquests funcionaran tots a la vegada. El flux d'informació (els punts de la mà) entre els punts de control es pot organitzar en forma de graph utilitzen els objectes *Flow*. El més comú és crear un graph en forma d'arbre. L'arbre de NITE descriu els estats possibles del flux d'informació. Els tipus d'objectes *Flow* són:

- *Router*: Envia totes les dades que rep d'un dels objectes que té connectat. Aquest és l'objecte que està actiu. L'objecte actiu és únic i es pot anar intercanviant segons es vulgui.
- *Broadcaster*: Envia totes les dades dels objectes que té connectats.

4.3 GAME ENGINES: PANDA3D

Per el desenvolupament d'un serious game o qualsevol aplicació que sigui del tipus videojoc és molt recomanable utilitzar un Game Engine. Hi ha diferents games engines disponibles en el mercat, alguns de comercials i d'altres de codi lliure. Alguns dels més populars actualment són Unity 3D, Ogre 3D, Panda3D i Irrlicht. Cadascun té unes característiques i funcionalitats que hem tingut en compte alhora de decidir quin utilitzar. A la Figura 50 hi ha una comparativa entres els games engines esmentats.

	unity	OGRE	PANDA3D	IRRLICHT LIGHTNING FAST REALTIME 3D ENGINE
Llicència	Unity Free Unity Pro	Open Source	Open Source	OpenSource
Llenguatge	JavaScript, C#, Boo (Python)	C++	C++/Python	C++
Plataformes	Windows, Linux, OSX, Iphone/Ipad, Android	Windows, Linux i Mac OSX	Windows, Linux i Mac OSX	Windows, Linux, Mac OSX, Solaris, (SDL)
Editor	Si	No	No	No
Gràfics	OpenGL	DirectX i OpenGL	DirectX i OpenGL	DirectX i OpenGL
Interfície 2D	Si	No	Si	Si
Il·luminació	Si	Si	Si	Si
Física	si	No	Si	No
Àudio	Si	No	Si	Si
Animació Esqueletal	Si	Si	Si	Si
Xarxa	Si	No	Si	No

FIGURA 50 COMPARATIVA DE GAME ENGINES

Els games engines comparats es poden utilitzar de manera gratuïta. Tant Ogre3D, com Panda3D i Irrlicht són OpenSource. En canvi, Unity té dos tipus de llicències: una de pagament i l'altre gratuïta. La versió de pagament té més funcionalitats. Ogre3D, Panda3D i Irrlicht coincideixen amb el llenguatge de programació C++. Tot i que Panda3d també es pot programar en Python. Unity es programa utilitzant Javascript, C# i Boo. Tots es distribueixen per plataformes Windows, Linux i MacOSX. A Unity es pot exportar a plataformes Iphone/Ipad i Android; i Irrlicht a Solaris. Només Unity3D ens facilita un editar per facilitar la programació. El render de gràfics el poden fer tant en OpenGL com en DirectX, menys Unity que només permet DirectX. Per la resta de característiques veiem que només Unity i Panda3D les implementen totes.

Després d'analitzar les opcions ens hem decidit per Panda3D degut al seu gran nombre de funcionalitats i sobretot al seu ràpid aprenentatge i velocitat de desenvolupament. Panda3D es pot programar en C++ i Python. El fet de que es pugui programar en Python és el que permet un ràpid desenvolupament. Python és un llenguatge d'alt nivell i amb una sintaxi molt neta ja que és necessari tabular el codi. Es tracta d'un llenguatge de programació multiparadigma ja que suporta programació orientada a objectes, programació imperativa i, en menor mesura, programació funcional. És un llenguatge interpretat, utilitza tipat dinàmic (una variable pot prendre valors de diferents tipus en diferents moments) i és multiplataforma.

4.3.1 PANDA3D: SCENE GRAPH

Molts motors simples de gràfics en 3 dimensions tenen una llista de models 3D a renderitzar a cada fotograma. En aquest motors, el programador ha de carregar cada model 3D i inserir-lo en aquesta llista de models per renderitzar-lo. El model no és visible pel motor fins que no està dins d'aquesta llista. Panda3D és lleugerament més sofisticat. En comtes de tenir una llista d'objectes per renderitzar, té un arbre d'objectes per renderitzar. Els objectes s'han d'inserir en aquest arbre perquè el motor hi tingui accés.

Aquest arbre conté objectes de la classe PandaNode. Aquesta classe és una superclasse de la qual hereten objectes del tipus: ModelNode, GeomNode, LightNode i d'altres. Aquests objectes són nodes de l'arbre. L'arrel se'n diu render. Hi ha diferents arrels per diferents propòsits. Com per exemple render2d per objectes en dues dimensions. L'arbre de Panda3d s'anomena *scene graph* (graf de l'escena). Aquest arbre és jeràrquic, això ens permet situar objectes relatius a altres objectes i agrupar-los segons el nostre criteri.

Quan els models s'afegeixen a l'arbre, qualsevol atribut de renderitzat es propaga pels nodes fills. Panda3D generar bounding boxes per cada node, per tant, una bona organització dels objectes incrementa la velocitat dels algorismes de frustum i oclusió de parts no visibles.

4.4 ALTRES TECNOLOGIES USADES

4.4.1 XML

XML, sigles angleses d'*Extensible Markup Language* (llenguatge de marques extensible), és un metallenguatge extensible d'etiquetes desenvolupat pel *World Wide Web Consortium* (W3C). És un sistema per a l'organització i etiquetat de documents, i permet definir la gramàtica de llenguatges específics. Aquest fet fa que no sigui realment un llenguatge específic, sinó una forma de definir llenguatges.

El seu origen està en un llenguatge inventat per IBM durant els anys setanta i anomenat GML. Posteriorment, aquest va derivar en SGML, un llenguatge normalitzat per la ISO al 1.986, i definit a la norma ISO 8879. Al 1.996, i de la mà de Sun Microsystems, sorgeix el desenvolupament d'XML com a intenció desimplificar i adaptar SGML, que era molt efectiu, però en ocasions massa complex. El W3C també va contribuir activament en la creació i desenvolupament d'aquest estàndard, arribant l'any 1.998 a XML 1.0, que acomplia les fites proposades pel grup de treball d'usabilitat en quant a compatibilitat amb SGML, llegibilitat, formalitat, facilitat d'autorització i facilitar el desenvolupament de processar software.

Després, amb el temps XML 1.0 ha anat patint petites variacions i ja es troba per la seva quarta edició. Per altra banda, es va crear un nou format, XML 1.1 , que conté caràcters que intenten fer XML més senzill d'usar en certs casos, principalment habilitant l'ús de caràcters de fi de línia i l'ús d'scripts i caràcters absents d'Unicode 2.0. Aquest estàndard està menys estès i només es recomana el seu ús per a qui necessiti característiques concretes.

Perquè un document XML sigui correcte ha de ser un Well-formed Document.Un document well-formed és aquell que es cenyeix a totes les normes de sintaxis dels XML. Un document que no és well-formed no es considera XML. Les normes bàsiques que ha de complir un Well-Formed Document per assolir aquest nivell de validesa són:

- Els elements no buits han d'estar delimitats per una etiqueta d'inici i una final.
- Els elements buits poden ser indicats amb una etiqueta que es tanqui a si mateixa.
- Tots els valors d'atribut s'han de posar entre cometes, ja siguin simples o dobles.
- Les etiquetes poden ser niuades però no sobreposar-se. Cada element no pertanyent a l'arrel directament ha d'estar contingut en un altre element.

- El document ha de tenir en compte la declaració de la codificació de caràcters. Aquesta codificació es pot declarar externament o internament. Si no hi ha cap declaració s'assumeix una codificació Unicode.

També s'ha de tenir en compte que es distingeix entre majúscules i minúscules en els noms dels elements (*case-sensitive*) i es recomana que aquests estiguin posats per a una correcta comprensió sense necessitat de documentació addicional.

4.4.2 Protocol Udp

User Datagram Protocol (UDP) és un protocol del nivell de transport del model OSI basat en l'intercanvi de datagrames (paquets de dades). Permet l'enviament de datagrames a través de la xarxa sense que s'hagi establert connexió prèvia, ja que el propi datagrama incorpora suficient informació de direccionament a la seva capçalera. Tampoc té confirmació ni control de flux, pel que els paquets es poden adelantar els uns als altres; i tampoc es pot saber si han arribat correctament ja que no hi ha confirmació d'entrega o recepció.

El seu ús principal és per a protocols com DHCP, BOOTP, DNS i altres protocols en els que l'intercanvi de paquets de la connexió/desconnexió són majors, o no són rentables respecte a la informació transmesa, així com per la transmissió de àudio i vídeo en temps real, on no és possible realitzar retransmissions pels estrictes requisits de retard que es té en aquests casos.

4.4.3 Memòria compartida

La forma més eficaç que tenen els processos per comunicar-se consisteix en compartir una zona de memòria, d'aquesta manera per enviar dades d'un procés a un altre, només s'ha d'escriure informació en aquesta memòria i automàticament les dades estaran disponibles per qualsevol procés. La utilització d'aquest espai de memòria comú evita la duplicació de dades i la lenta transferència d'informació entre processos.

4.4.4 FFMPEG

FFMPEG és una col·lecció de programari lliure que pot gravar, convertir i fer streaming d'àudio i vídeo. Inclou libavcodec, una biblioteca de còdexs. FFMPEG està desenvolupant en GNU/Linux, però pot ser compilat en la majoria de sistemes

operatius, incloent Windows. El projecte el va començar Gerard Lantau, un pseudònim de Fabrice Bellard, i ara es mantingut per Michael Niedermayer.

Aquesta llibreria implementa còdexs com: MPEG-1, MPEG-2, MPEG-4 Part 2 (el format utilitzat pels còdexs DivX i Xvid), H.261 , H.263, H.264/MPEG-4 AVC (únicament la descodificació), WMV versió 7, 8 y 9 (únicament la descodificació), Sorenson codec, Cinepak, MJPEG, Huffyuv, Snow, Theora (únicament la descodificació), VP3 / VP5 / VP6 (únicament la descodificació), ...

4.4.5 SQLLITE

SQLITE és una llibreria en C que permet l'ús de bases de dades lleugeres locals que no necessiten un procés a cap servidor per separat i permet accedir a la base de dades utilitzant una variant no estàndard del llenguatge de consultes SQL. Algunes aplicacions poden usar SQLite per emmagatzematge intern. També es possible prototipar una aplicació utilitzant SQLite i per després fer la portabilitat del codi a una base de dades més gran com podria ser PostgreSQL o Oracle.

Per aquest projecte s'ha utilitzat Sqlite3. Sqlite3 és una implementació en python escrita per Gerhard Häring que prové d'una interfície SQL amb l'especificació DB-API 2.0 descrita per PEP 249 [26].

4.8 Implementació del Servidor

L'aplicació servidor és una evolució de la aplicació explicada en el punt 3.3 Implementació de la captura de moviment amb kinect. A aquesta aplicació se li han afegit més funcionalitats. Les 4 funcionalitats principals són:

- Detecció de gestos i seguiment de la mà
- Detecció de la postura del usuari
- Guardar els vídeos i els moviments fets per l'usuari.
- Comunicar-se amb l'aplicació client.

La detecció de la postura de l'usuari i guardar els seus moviments ja estaven implementats en l'aplicació predecessora. Per afegir les noves funcionalitats s'han creat diverses classes que queden encabides en el diagrama de classes de la Figura 51.

De la mateixa manera que l'aplicació anterior, aquesta conté un objecte xn::Context. Que a la vegada inclou xn::DepthGenerator, xn::ImageGenerator i un xn::UserGenerator. També utilitzem un objecte del tipus xn::HandsGenerator. Aquest és el que s'encarrega de fer el seguiment de la posició de la mà. Per poder capturar els gestos que l'usuari realitza s'ha creat un objecte SessionManager que conté un objecte Flow Router. Dins d'aquest objecte FlowRouter és on trobem la classe Interfície que hereta de XnvPointControl, on es detecten els gestos. Com s'ha explicat en la part teòrica, els objectes FlowRouter només treballen amb les dades d'un objecte Point Control que dels que estan connectats. Per aquest motiu s'ha connectat la classe Interfície a un FlowRouter, permetent canviar la detecció de gestos creant altres classes i connectant-les al FlowRouter.

La classe Capture s'ha ampliat afegint variables membres i mètodes. S'ha implementat un filtre de soroll per les dades capturades i mètodes per calcular angles entre vectors definits per les posicions de les articulacions. S'ha creat la classe Socket per implementar la comunicació UDP entre aquesta aplicació i l'aplicació pacient. Per tal de poder enregistrar els vídeos capturats per la càmera de Kinect s'ha programat la classe VideoEncoderFFMPEG. Aquesta classe conté mètodes per començar la codificació de fotogrames i generar fitxers de vídeo. Per acabar, hem utilitzat un objecte del tipus SharedBuffer per implementar la memòria compartida entre les dues aplicacions esmentades.

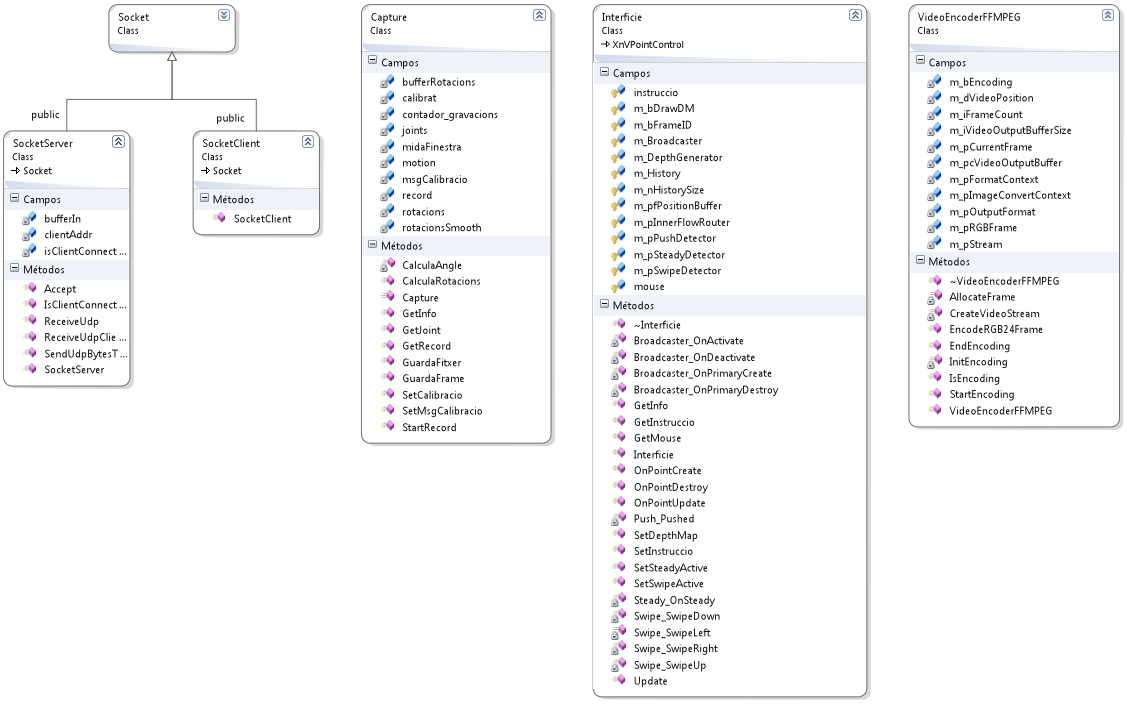

FIGURA 51 DIAGRAMA DE CLASSES DE L'APLICACIÓ SERVIDOR

El funcionament global de l'aplicació és pot resumir amb el diagrama d'estats de la Figura 52. L'aplicació comença fent diverses inicialitzacions: inicialització de l' objecte Context, inicialització d'OpenGL, inicialització de l'objecte captura i socket, creació del framebuffer, inicialització de FFMPEG i creació de l'objecte SessionManager. Un cop el sistema està inicialitzat s'executaran dos fils d'execució a la vegada. L'un es tracta d'un thread que s'encarrega d'escoltar les trames rebudes per l'aplicació client. L'altre és la rutina habitual d'OpenGL. En la funció IDLE es fa l'escriptura de la imatge de la càmera Kinect al FrameBuffer i es codificant els fotogrames en cas d'estar gravant el vídeo. També es fa l'enviament de trames cap a l'aplicació client, especificant l'estat de la sessió, els gestos detectats, la posició de la mà i dades referents a la captura de l'esquelet. A continuació el programa executa el mètode Display. Aquest mètode s'encarrega d'actualitzar les dades rebudes del dispositiu Kinect i actualitzar els processos referents a l'arbre de NITE. També s'actualitzen les posicions de les articulacions en la detecció de la postura de l'usuari i es guarden aquestes posicions en un fitxer quan l'aplicació client ho sol·licita. Finalment es pinta per pantalla tant les imatges de la càmera de Kinect com la posició de les articulacions estimades. El bucle infinit d'OpenGL continua executant-se igual que el thread EscoltaClient fins la finalització del programa.

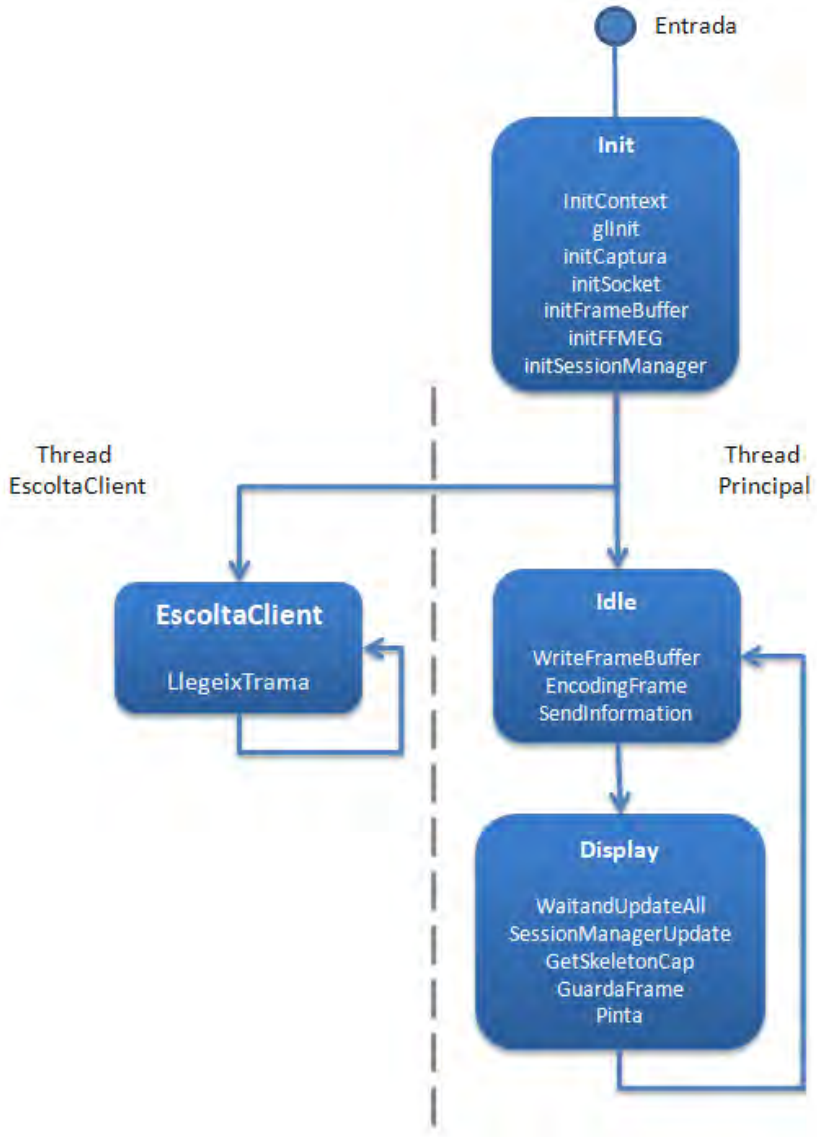

FIGURA 52 DIAGRAMA D'ESTATS DE L'APLICACIÓ SERVIDOR

4.8.1 INICIALITZAR OPENGL, OPENNI I NITE

L'inicialització de l'aplicació gràfica i la interacció natural és fa de la mateixa manera que en el

3.3.1 Inicialitzar OpenGL, OpenNI i NITE. En aquesta aplicació afegim la gestió de sessions, seguiment de la mà i detecció de gestos que ens permet OpenNI i NITE. Per fer-ho hem utilitzat el fragment de codi següent:

```
// Create NITE objects

nRetVal = g_Context.FindExistingNode(XN_NODE_TYPE_HANDS, g_HandsGenerator);
CHECK_RC(nRetVal, "Find hands generator");

g_pSessionManager = new XnVSessionManager;
nRetVal = g_pSessionManager->Initialize(&g_Context, "Click,Wave", "RaiseHand");
CHECK_RC(nRetVal, "SessionManager::Initialize");
```

```
g_pSessionManager->RegisterSession(NULL, SessionStarting, SessionEnding,
FocusProgress);

g_pDrawer = new Interficie(20, g_DepthGenerator);
g_pFlowRouter = new XnVFlowRouter;
g_pFlowRouter->SetActive(g_pDrawer);

g_pSessionManager->AddListener(g_pFlowRouter);
```

A la variable g_Context hi afegim l'objecte HandsGenerator. On resideixen els algorismes referents als moviments amb la mà. A continuació es crea l'objecte Interfície, que és el Punt de Control actiu de la variable global g_pFlowRouter del tipus XnVFlowRouter. Aquest Flow Router (4.2.3 Objectes *Flow* i l'arbre de NITE) actua com a Listener de la variable g_pSessionManager, la Session Manager (4.2.1 Session manager) de l'aplicació.

4.8.1 Classe Interfície

Hem implementat una classe que hereta de XnVPointControl que es diu Interficie. Aquesta classe és l'encarregada de detectar els gestos push, swipe i steady. Per poder agrupar aquest controls es necessari utilitzar l'arbre de NITE. L'organització dels punts de control està descrita a la Figura 53. A la capa superior, tenim un objecte Broadcaster connectat al Detector de Push i a un objecte Router. El router està connectat a un Detector Steady i un Detector Swipe. Per tant, el *Session Manager* gestionarà al mateix temps el Push Detector i el detector actiu del Router. El router intercanviarà el detector actiu cada cop que detecta un swipe o un steady.

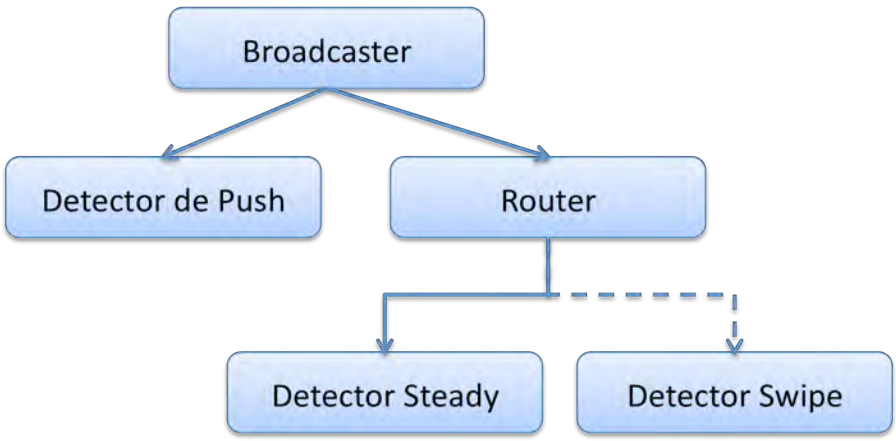

FIGURA 53 CONTROL INTERFÍCIE

Per actualitzar el *Session Manager* s'utilitza la comanda següent:

```
g_pSessionManager->Update(&g_Context);
```

Aquesta comanda es crida a la funció IDLE del programa. Cada cop es detecta un gest s'activa un callback de la classe. Aquest callbacks guarden la informació referent al gest en una cadena de caràcters, com per exemple, "push". D'aquesta manera, quan l'aplicació servidor notifiqui els events a l'aplicació client, es pugui enviar aquesta cadena dins d'una trama, juntament amb informació d'altres mòduls.

4.8.2 CAPTURA DEL MOVIMENT

Per fer la captura del moviment més robusta hem implementat un filtre de suavitzat. D'aquesta manera evitem les fluctuacions en la detecció de les posicions de les articulacions. El filtre de suavitzat consisteix en fer la ponderació de la posició en l'instant actual calculada i les mostres anteriors. S'ha fet servir aquesta fórmula,

$$y(n) = w * x(n) + (1 - w) * \left(\frac{x(n-1) + x(n-2) + \ldots + x(n-N)}{N}\right)$$

FÒRMULA 2 FILTRE DE SUAVITZAT

On n, és l'instant de temps actual; w, és el pes de la mostra actual i N és la mida de la finestra utilitzada.

Els paràmetres que s'han escollit han estat w = 0.5 i N = 10. Això significa que la mostra actual té tant de pes com la mitja de les 10 anteriors mostres, per tant, que es té en compta els valors de 0,3 segons anteriors.

Per implementar-ho s'han utilitzat vectors de 10 posicions per guardar els valors de les mostres anteriors de cada articulació detectada. El següent fragment de codi mostra el càlcul d'un valor suavitzat d'una articulació en concret.

```
rotacions.at(3) =
CalculaAngle(joints.at(3),joints.at(4),joints.at(5)); //ElbowL

for (int i=0;i<bufferRotacions.at(3).size();i++){
```

```
      valorSmooth = valorSmooth + bufferRotacions.at(3).at(i);

}
valorSmooth = valorSmooth / bufferRotacions.at(3).size();
rotacionsSmooth.at(3) = w1 * (rotacions.at(3)) + w2 * (valorSmooth);


nouValor = rotacions.at(3);
if  (bufferRotacions.at(3).size()< midaFinestra){
      bufferRotacions.at(3).push_back(nouValor);
}else{
      bufferRotacions.at(3).pop_front();
      bufferRotacions.at(3).push_back(nouValor);
}
```

Els procediments explicats en aquest punt han estat afegits en la classe Capture. L'altre mètode rellevant d'aquesta classe és CalculaAngle. Aquest mètode s'encarrega de calcular l'angle format per la configuració de 3 articulacions. Rep com a paràmetres la posició en tres dimensions de 3 extremitats. Calcula un vector des de l'extremitat 1 i 2, i un altre des de 2 i 3. Després es calcula l'angle de format utilitzant la Fòrmula 1.

4.8.3 COMUNICACIÓ UDP

La comunicació UDP s'ha implementat utilitzant la classe Socket. En aquest programa s'ha creat una instància SocketServer. Aquest objecte té l'objectiu d'enviar i rebre datagrames UDP. Després d'inicialitzar l'objecte especificant el port, s'ha utilitzat el mètode ReceiveUdpClientRegistration() que paral·litza l'aplicació fins que rep una trama del Client.

Per llegir les trames que envia el client hem creat un thread utilitzant la llibreria *process.h*. Aquest thread executa un bucle infinit que va llegint contínuament les trames d'entrada.

```
unsigned ret;
_beginthreadex(0,0,EscoltaClientRender,(void*)udpSocket,0,&ret);
```

Mitjançant les trames "*Inici gravacio"* I "*Fi gravacio*", l'aplicació gestiona l'inici I el fi de les gravacions dels moviments.

Al encarregar-se el client de la interacció de l'aplicació pacient és necessari estar enviant ininterrompudament informació sobre la detecció de gestos i la posició de la mà quan l'aplicació client es trobi en els menús. Quan el pacient està jugant també es necessari enviar contínuament les posicions de les articulacions. Per fer-ho s'ha creat

la funció sendInformation() que es crida a la funció Idle de GLUT. El següent fragment de codi pertany a aquesta funció.

```
sendInformation() {
	string estat;
	switch (g_SessionState) {
		case IN_SESSION:
			estat = "In Session";
			break;
		case NOT_IN_SESSION:
			estat = "Not in Session";
			break;
		case QUICK_REFOCUS:
			estat = "Quick refocus";
			break;
	}

	udpSocket->SendUdpBytesToClient(estat + g_pDrawer->GetInfo() +
	captura->GetInfo());

	g_pDrawer->SetInstruccio("nothing");
}
```

Les trames que s'envien constantment estan formades per l'estat de l'usuari respecte l'aplicació, la informació de la interfície I la informació de la captura. Seguidament la informació de la interfície torna al estat inicial per evitar que el client rebi els events per duplicat.

4.8.5 SHAREDBUFFER

Per fer l'enviament de la imatge de color capturada pel dispositiu Kinect des de l'aplicació servidor a l'aplicació servidor utilitzem memòria compartida. Per crear l'espai de memòria compartida hem utilitzat la variable sharedBuffer. Aquesta variable, que és del tipus LPCTSTR, li assignem un fitxer que serà el compartit. Per fer-ho creem la variable sharedFile de tipus HANDLE i cridem a la funció MapViewOfFile. Per definir el fitxer de compartició és necessari assignar un nom a l'espai compartit i la seva mida. Això ho fem amb la variable sharedName i la constant FRAME_SIZE. Per calcular la mida de l'arxiu hem multiplicat la mida de la imatge capturada per kinect (640x480) pel nombre de canals de la imatge (3: vermell, blau i verd). El següent fragment de codi mostra les comandes utilitzades.

```
#define FRAME_SIZE 921600 //640*480*3
TCHAR sharedName[]=TEXT("Global\\SharedKinectFrameBuffer");
LPCTSTR sharedBuffer;
```

```
HANDLE sharedFile;

void initSharedFrameBuffer()
{
   sharedFile = CreateFileMapping(
                  INVALID_HANDLE_VALUE,    // use paging file
                  NULL,                    // default security
                  PAGE_READWRITE,          // read/write access
                  0,        // maximum object size (high-order DWORD)
                  FRAME_SIZE,// maximum object size (low-order DWORD)
                  sharedName);             // name of mapping object
   if (sharedFile == NULL) {
      tprintf(TEXT("Could not create file mapping object (%d).\n"),
GetLastError());

   }

   sharedBuffer = (LPTSTR)MapViewOfFile(sharedFile,   // handle to map
object
                        FILE_MAP_ALL_ACCESS, // read/write permission
                        0,0, FRAME_SIZE);

   if (sharedBuffer == NULL) {
            _tprintf(TEXT("Could not map view of file (%d).\n"),
GetLastError());
      CloseHandle(sharedFile);
   }
}
```

Quan el programa ja es troba en el seu bucle d'execució, copiem la imatge de la càmera del Kinect a l'espai de memòria compartit com es mostra al fragment de codi següent.

```
g_ImageGenerator.GetMetaData(g_imageMD);
unsigned char* data= (unsigned char*)g_imageMD.Data();
unsigned int size= g_imageMD.DataSize();

CopyMemory((PVOID)sharedBuffer, data, FRAME_SIZE);
```

Per accedir a la imatge del Kinect utilitzem el mètode GetMetaData de la classe Image Generator i la copiem utilitzant la funció CopyMemory.

4.8.6 Classe VideoEncoderFFMPEG

La classe VideoEncoderFFMPEG és la que s'ocupa de la generació d'arxius de vídeo a partir de la càmera de Kinect. Aquesta classe utilitza els mètodes StartEncoding i

EndEncoding per començar i acabar la codificació de fotogrames. Aquest mètodes es criden en el mateix moment que els mètodes StartRecord i GuardaFitxer de la classe Capture.

Seguidament de copiar la imatge de la càmera a l'espai de memòria compartida fem la codificació dels fotogrames fent crides al mètode EncodeRGB24Frame. Només codifiquem fotogrames quan l'aplicació ha fent anteriorment la crida a StartEncoding. Per comprovar-ho utilitzem el mètode isEncoding. Aquest és el fragment de codi usat.

```
if(g_videoEncoder.IsEncoding())
{
      unsigned char* pcEncodedData;
      g_videoEncoder.EncodeRGB24Frame(data,   &pcEncodedData,   size);

}
```

La codificació dels fotogrames en un stream de vídeo s'ha fet amb el còdec H264, generant arxius amb la extensió MP4.

4.9 IMPLEMENTACIÓ DEL CLIENT

L'aplicació Client és la que es mostra a l'usuari final. S'encarrega del render i de la lògica del joc. Aquesta aplicació ha estat programada amb el game engine Panda3D i en el llenguatge de programació Python.

L'usuari navega per l'aplicació amb els events d'usuari capturats per la part del servidor, que informa constantment a l'aplicació client. La navegació entre pantalles es mostra a la Figura 54. Abans d'iniciar la navegació, l'aplicació carrega el fitxer de configuració. Aquesta informació servirà al programa per seleccionar els jocs. L'usuari quan inicia l'aplicació es troba a la pantalla de Benvinguda. En aquesta pantalla l'usuari ha de realitzar un Push amb la mà, capturat pel servidor, per poder controlar el mouse amb la mà. A continuació, l'usuari pot accedir al menú de personatges on n'escollirà un i passarà a la introducció d'un joc relacionat amb l'exercici que ha de fer. Després de la introducció del joc es passa a la pantalla de calibració, en cas de que l'usuari no estigui calibrat. Seguidament es mostraran les instruccions de l'exercici i es procedirà amb el joc. Un cop finalitzat el joc es mostrarà la puntuació obtinguda. Després de la puntuació es tornarà a la introducció del joc següent, sempre i quan no sigui l'últim a realitzar. En el cas d'estar en l'últim exercici es passarà a la pantalla resum. En aquesta darrera pantalla es mostra un resum dels exercicis realitzats.

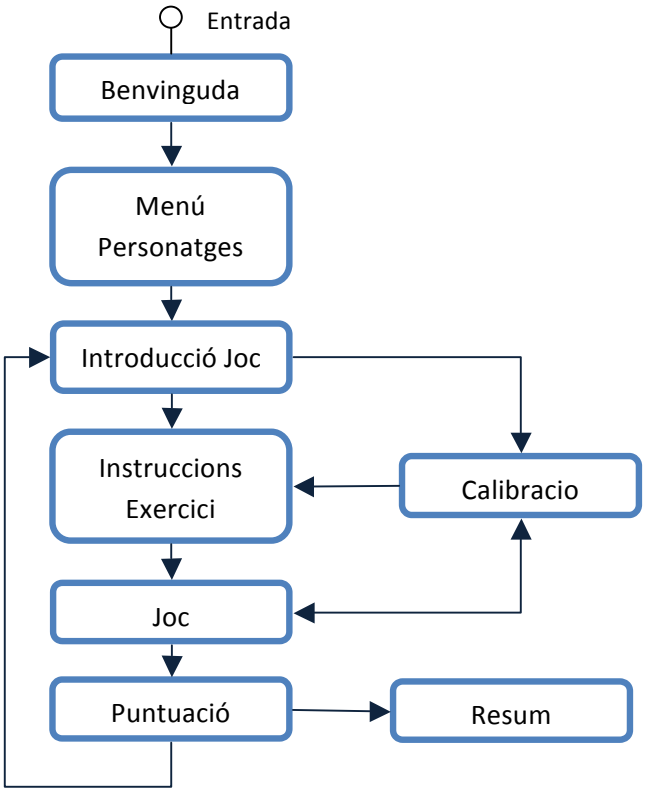

FIGURA 54 NAVEGACIÓ ENTRE PANTALLES DE L'APLICACIÓ PACIENT

Per implementar aquesta aplicació s'han utilitzat les classes descrites a la Figura 55. L'aplicació té una classe principal que és la que s'encarrega d'inicialitzar el sistema, gestionar els events I les pantalles. La resta de classes s'encarreguen de llegir la configuració, comunicar-se amb l'aplicació servidor, mostrar les pantalles I tractor amb la base de dades.

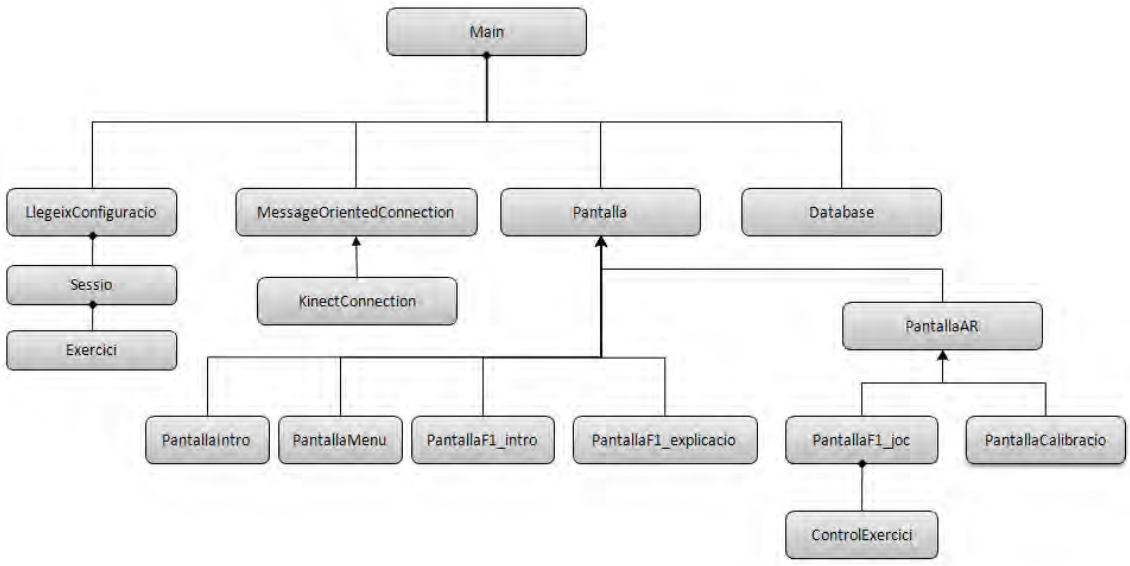

FIGURA 55 DIAGRAMA DE CLASSES DE L'APLICACIÓ CLIENT

4.9.1 Classe Main

La classe Main és la classe principal de l'aplicació. Aquesta classe hereta de DirectObject. Aquesta classe és l'encarregada d'inicialitzar el sistema. Per inicialitzar-lo s'executa el fragment de codi següent:

```
self.initCamara()
self.initCartoon()
self.initUserEvents()
self.initCursor()

self.config = llegeixConfiguracio("../config/configuracio1.xml")
self.config.parseja()

self.kinect= KinectConnection(2000)
self.kinect.enableSkeletonReception()

self.db = Database(“rehabtimals.db”)

self.pantallaActual= PantallaIntro(self)
```

El primer que fem es inicialitzar la càmera del món 3D cridant a la funció initCamara(). Tot seguit cridem a la funció initCartoon() per aplicar a l'aplicació un shader de renderitzat gràfic del tipus Cartoon. A continuació, inicialitzem els events d'usuaris I el mouse de l'aplicació.

El següent que fem és instanciar la classe llegeixConfiguracio I cridar al seu mètode parseja. D'aquesta manera carreguem la informació de l'arxiu que li passem com a paràmetre. Per poder comunicar-nos amb l'aplicació servidor creem la variable kinect del tipus KinectConnection I activem la recepció de dades amb el mètode enableSkeletonReception(). Inicialitzem també la base de dades utilitzen la classe Database. Per acabar creem la variable pantallaActual. Aquesta variable contindrà la pantalla que es mostra a l'aplicació. La inicialitzem amb la pantalla d'introducció. Cal notar que la pantalla rep com a paràmetre la classe principal. Això es degut a que Panda3D només permet crear events a classes DirectObject, I així podem crear events des de qualsevol pantalla.

La gestió de pantalles també es duu a terme en la classe principal. El mètode nextPantalla(idPantalla) és l'encarregat de fer-ho. Aquest mètode el cridem des de les pantalles I selecciona la nova pantalla en funció de idPantalla.

4.9.2 Càrrega de la configuració de la sessió

El flux de l'aplicació està definit per un arxiu de configuració. Aquest arxiu és un arxiu XML que defineix la sessió de recuperació que el pacient ha de fer. La configuració defineix el nom del pacient i la teràpia. La teràpia ve definida per l'articulació a recuperar, la lateralitat de la recuperació (dreta o Esquerra) i la fase en la que el pacient es troba. A continuació es defineix la sessió, indicant els cops que el pacient l'ha de realitzar al dia. La sessió inclou un conjunt d'exercicis. Els exercicis de la sessió venen definits per la seva descripció i els paràmetres de control escollits pel fisioterapeuta. Aquest paràmetres són el nombre de repeticions, l'angle mínim i màxim que el pacient ha d'assolir i el temps d'espera a les posicions extremes. El format d'aquest arxiu es mostra a continuació.

```
<configuracio>
	<pacient> Adso Fernandez </pacient>
	<teràpia articulacio="Genoll" lateralitat="Dreta" fase="F1">
		<sessio frecuencia="1">
			<exercici descripcio="flexio_extensio genoll">
				<repeticions> 12 </repeticions>
				<temps_espera> 0 </temps_espera>
				<angle_minim> 30 </angle_minim>
				<angle_maxim> 90 </angle_maxim>
			</exercici>
			<exercici descripcio="flexio_extensio genoll amb ajuda">
				<repeticions> 15 </repeticions>
				<temps_espera> 0 </temps_espera>
				<angle_minim> 30 </angle_minim>
				<angle_maxim> 90 </angle_maxim>
			</exercici>
			<exercici descripcio="gel">
				<temps> 5 </temps>
			</exercici>
		</sessio>
	</teràpia>
</configuracio>
```

Per carregar l' informació del arxiu en memòria s'ha creat la classe llegeixConfiguracio. Aquesta classe utilitza la llibreria de python *xml.dom.minidom* per parsejar l'arxiu XML. Aquesta classe té una instancia de la classe *Sessio*, que a la vegada té una llista d'objectes *Exercici*. En aquestes classes s'emmagatzema la informació de la configuració.

4.9.3 Comunicació UDP

La comunicació entre aquesta aplicació i l'aplicació Servidor s'ha dut a terme amb la classe KinectConnection. Aquesta classe hereta de MessageOrientedConnection que

implementa els mètodes necessàries per crear un socket UDP, connectar-se amb el Servidor, rebre I enviar trames UDP.

La classe KinectConnection té un bucle infinit que rep les trames que el servidor envia. Les trames són parsejades per extreure informació sobre la sessió, la posició de la mà (que es convertirà en posició del cursor) i els angles de rotació de les articulacions. Aquesta informació es guarda en variables de la classe.

4.9.4 PANTALLES

Les pantalles de l'aplicació hereten totes de la classe Pantalla. La classe Pantalla té com a variables membres main, contenedor3d, contenedor2d i llistaBotons. La variable main és inicialitzada amb la pantalla, ja que aquesta rep la classe principal a la seva inicialització. Contenedor3D és una variable on tots els elements 3D de la pantalla seran fills. Les variables contenedor2d i llistaBotons funcionen de la mateixa manera.

Els mètodes destruye() i acabaPantalla() són els més importants. El primer es crida quan la pantalla actual es dona per finalitzada. S'encarrega d' esborrar la informació de les variables membres, esborran així els elements que surten per pantalla. El segon es crida quan volem passar a la següent pantalla. Aquest fa una crida al mètode nextPantalla() de la classe main.

Per poder visualitzar la imatge que capta la càmera de Kinect hem implementat la classe PantallaAR, que hereta de Pantalla. A la seva inicialització, es crea un objecte cardMaker que texturitzarem amb imatges rebudes. Aquesta classe incorpora dos nous mètodes: startVideoStream i updateSharedFrameBuffer. El mètode startVideoStream crea una tasca que és updateSharedFrameBuffer. Aquesta llegeix la informació de l'espai de memòria compartida i canvia la textura a mostrar. Per implementar aquesta funcionalitat hem usat la llibreria mmap de Python. El codi d'aquests mètodes es mostra a continuació.

```
def startVideoStream(self):
      taskMgr.add(self.updateSharedFrameBuffer,"updateSharedFrameBuffe
r")

def updateSharedFrameBuffer(self,task):
        sharedFB = mmap.mmap(0, 921600,
"Global\\SharedKinectFrameBuffer", mmap.ACCESS_READ)
        frameBuffer= PTAUchar.emptyArray(921600)
        frameBuffer.setData(sharedFB)
        self.tex.setRamImage(CPTAUchar(frameBuffer),Texture.CMOff)
        return Task.cont
```

4.9.5 Classe ControlaExercici

Quan l'usuari es troba en la pantalla de joc, l'aplicació ha d'analitzar els angles rebuts pel Servidor per poder dirigir el flux del joc i donar feedback a l'usuari. Els exercicis de recuperació implementats estan basats en la repetició. Aquesta repetició està definida en el fitxer de configuració indicant l'angle mínim i màxim que el pacient ha d'assolir. A la Figura 56 hi ha una representació d'aquests angles en una repetició de flexió del genoll, l'exercici que hem implementat. De color vermell tenim les zones on el pacient ha superat tant l'angle mínim com l'angle màxim.

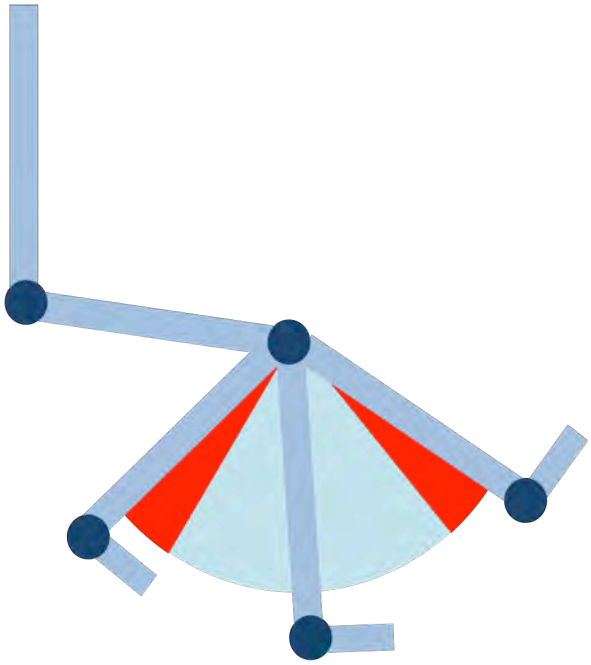

FIGURA 56 REPRESENTACIÓ DELS ANGLES MÍNIMS I MÀXIMS D'UNA REPETICIÓ DE FLEXIÓ DE GENOLL

Per poder controlar si el pacient realitza la repetició i esbrinar si aquesta és bona o no hem implementat la classe ControlExercici. Aquesta classe l'utilitzarem en cada pantalla de joc on vulguem controlar les repeticions. El procés de detecció de repetició està representat pel algorismes descrits a la Figura 57. El procés consta de dos bucles que funcionant en paral·lel. Un és encarregat de detectar les repeticions controlant els angles mínims i màxims. L'altre és un contador de passos pel punt entremig entre els dos angles extrems. El que es vol aconseguir és contar com a repeticions bones quan s'assoleix l'angle mínim i màxim, i com a repetició dolenta dos passos per l'angle mig sense sobrepassar els límits extrems marcats per l'angle mínim i màxim.

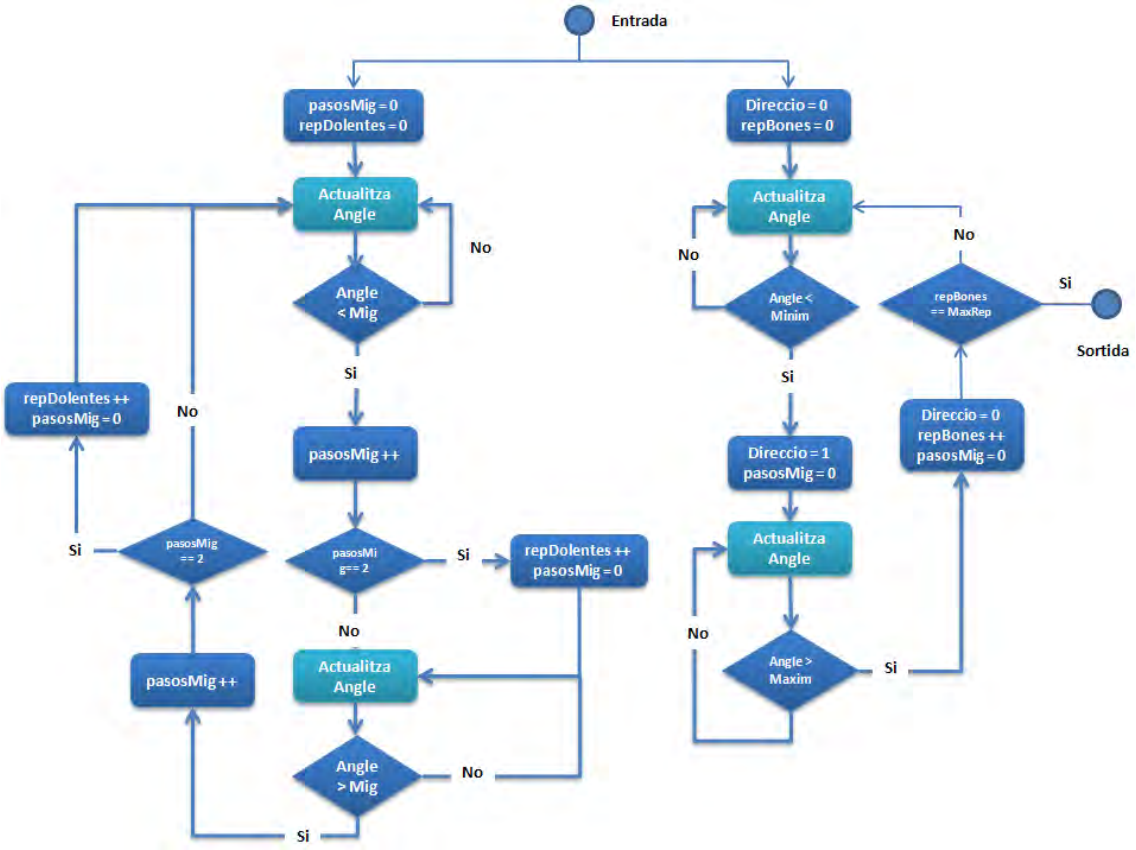

FIGURA 57 DETECCIÓ DE REPETICIÓ

El bucle de l'esquerra és el contador de passos per l'angle mig. L'angle mig és calculat fent la resta en valor absolut entre els angles i restant el resultat a l'angle màxim. El procés s'inicia amb les variables pasosMig i repDolentes a 0. L'algorisme rep l'angle de rotació i comprova si aquest és inferior a l'angle mig. En cas negatiu, es repeteix el pas anterior i en cas positiu, s'incrementa la variable pasosMig. Després d'incrementar la variable es comprova si aquesta és igual a 2. Si ho és, s'incrementa repDolentes i es posa a 0 pasosMig. A continuació s'actualitza l'angle i es comprova si és superior a l'angle mig. Mentre no ho sigui anirà repetint el pas anterior. Quan la resposta és afirmativa s'incrementa el contador pasosMig i es torna a comprovar si és igual a 2. En cas positiu s'incrementa repDolentes i es posa a 0 pasosMig i es torna a començar. En cas negatiu es torna a començar directament.

El bucle descrit en l'anterior paràgraf només es una part de la detecció de repeticions. A la part dreta de la Figura 57 tenim l'altra bucle. Aquest s'inicia amb les variables direcció i repBones igual a 0. S'actualitza l'angle d'entrada i comprovem si aquest és més petit que l'angle mínim. Continuem fent el pas anterior fins que la resposta sigui

positiva. Quan ho sigui la variable direccio la posarem a 1 i la variable pasosMig la posarem a 0. Actualitzem l'angle d'entrada i comprovem si aquest és més gran que l'angle màxim. Continuem fent el pas anterior fins que la resposta sigui positiva. Quan ho sigui la variable direccio i pasosMig pasan a valdre 0 i augmentem repBones. Seguim repetint el procediment fins que assolim el màxim de repeticions fixat.

4.9.6 BASE DE DADES

Cada cop que el pacient utilitza l'aplicació, fa una sessió de recuperació, es guarden una sèrie de dades a una base de dades. La base de dades que s'ha implementat està descrita en el punt 5.5.1 Base de dades. Les dades a guardar les trobem a la classe controlExercici. Aquesta classe, com ja hem explicat, només s'utilitza en les pantalles de joc. Per tant, cada cop que l'usuari finalitza el joc, la informació de la classe controlExercici es bolcada a la base de dades inserint un nou exercici a la taula Exercicis. Si l'exercici que el pacient ha realitzat és el primer de la sessió, s'insereix una nova sessió a la taula Sessions.

4.10 Execució de l'aplicació pacient

L'execució de l'aplicació pacient requereix inicialitzar les aplicacions servidor i client. Per automatitzar aquest procediment, des de l'aplicació servidor es fa la inicialització de l'aplicació client.

L'aplicació client s'inicia quan executem l'arxiu *executa.bat*. Aquest és un arxiu de pila de comandes on hi ha la instrucció ppython *main_rehabtimals.py*. Per tant, des de l'aplicació servidor hem utilitzat el fragment de codi següent per executar l'arxiu .bat.

```
SHELLEXECUTEINFO ExecuteInfo;

memset(&ExecuteInfo, 0, sizeof(ExecuteInfo));

ExecuteInfo.cbSize       = sizeof(ExecuteInfo);
ExecuteInfo.fMask        = 0;
ExecuteInfo.hwnd         = 0;
ExecuteInfo.lpVerb       = "open"; // Operation to perform
ExecuteInfo.lpFile       =
"c:\\Users\\Adso\\Desktop\\Panda3D\\src_artik\\executa.bat";  //
Application name
ExecuteInfo.lpParameters = ""; // Additional parameters
ExecuteInfo.lpDirectory  =
"c:\\Users\\Adso\\Desktop\\Panda3D\\src_artik"; // Default directory
ExecuteInfo.nShow        = SW_SHOW;
ExecuteInfo.hInstApp     = 0;

if(ShellExecuteEx(&ExecuteInfo) == FALSE) cout <<"No s'ha pogut
inicialitzar l'aplicació client";
```

4.11 RESULTATS

En aquest punt es mostren captures de pantalla de l'aplicació pacient. Aquesta aplicació està desenvolupada completament a nivell funcional, en canvi, falten alguns elements visuals i podria patir algun canvi a nivell gràfic. La Figura 58 mostra la primera pantalla que se li mostra a l'usuari final. Podem veure que hi ha un icona on apareix una mà. Aquest icona desapareix en el moment que l'usuari es fa amb el control de l'aplicació, és a dir, es comença a realizar el seguiment de la seva mà.

FIGURA 58 PANTALLA D'INTRODUCCIÓ

La captura de la Figura 59 es tracta de la següent pantalla. En aquesta pantalla el pacient ha d'escollir amb quin animal vol fer la teràpia. El pacient pot rotar el cub de personatges fent gestos de swipe. A la Figura 60 podem veure aquest cub rotant.

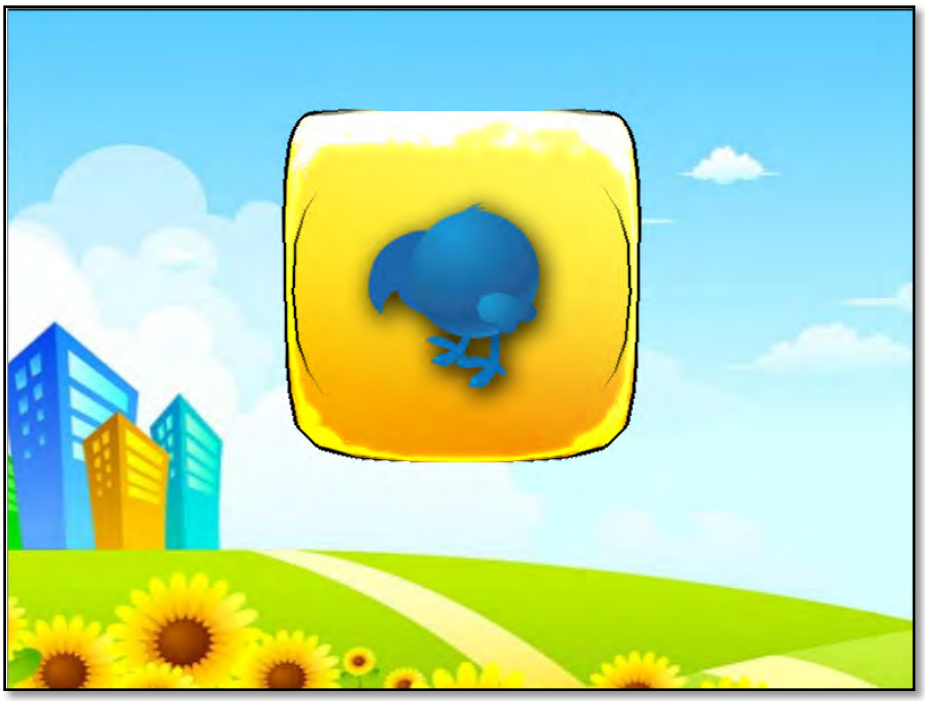

FIGURA 59 PANTALLA MENÚ DE PERSONATGES

FIGURA 60 PANTALLA MENÚ DE PERSONATGES. CUB GIRANT.

Després d'escollir el personatge de l'univers Rehabtimals es mostra la introducció del joc que inclou la teràpia corresponent. La Figura 61 mostra el resultat de la pantalla d'introducció de la fase 1 de la teràpia implementada.

FIGURA 61 PANTALLA INTRODUCCIÓ DE LA FASE 1. SELECCIÓ DE NIUS

Abans de realitzar l'exercici de recuperació, és necessàri que l'usuari es calibri per que el sistema de captura comenci a fer el seguiment. Hem implementat la pantalla de calibració amb el resultat que es mostra a la Figura 62.

FIGURA 62 PANTALLA DE CALIBRACIÓ

En aquest moments el sistema està estimant la posició del pacient i només resta donar-li les instruccions pertinents per que faci correctament l'exercici que li toca. La Figura 63 mostra aquesta pantalla d'instruccions. Podem veure que al centre de la pantalla tenim un model 3D que realitza el moviment que l'usuari ha d'imitar.

FIGURA 63 PANTALLA INSTRUCCIONS DE LA FASE 1

Finalment l'usuari es disposa a jugar. La Figura 64 mostra l'aspecte del joc. En aquesta pantalla podem observar que hi ha 3 etiquetes de text on s'indiquen els erros comesos, l'angle de rotació actual i el nombre de repeticions fetes. Aquestes etiquetes seran substituïdes per icones o imatges més estètiques. A la part central de la imatge podem veure un ou esquerdat i un missatge d'"Aaaau!!". Ambdues coses són encarregades de donar feedback a l'usuari mentre realitza l'exercici. Per acabar, tenim la imatge de vídeo capturada per la càmera de Kinect a la part inferior dreta de la pantalla.

/

FIGURA 64 PANTALLA DE JOC DE LA FASE 1

5. APLICACIÓ FISIOTERAPEUTA

5.1 INTRODUCCIÓ

L'aplicació Fisioterapeuta és una aplicació que té com a objectiu configurar les sessions de rehabilitació pels pacients i fer el seguiment d'aquestes. Aquesta aplicació ha estat implementada en python utilitzant Panda3D.

La configuració de les sessions de recuperació es fa generant un arxiu XML a partir de les teràpies que es troben a una base de dades. El fisioterapeuta disposa d'una interfície per per seleccionar la teràpia que desitgi, seleccionar els exercicis per configurar una sessió i parametritzar-los.

El seguiment de les teràpies es fa visualitzen vídeos, moviments en tres dimensions i gràfiques dels angles calculats. El fisioterapeuta selecciona les sessions que es troben a la base de dades. Aquestes són generades per l'aplicació pacient. Un cop a seleccionat la sessió a visualitzar es carrega la informació per fer el seguiment.

FIGURA 65 DIAGRAMA DE L'APLICACIÓ FISIOTERAPEUTA

5.2 Interfícies amb panda3D: DirectGUI

Panda3D té un conjunt d'eines per la creació d'interfícies gràfiques. Aquestes eines les podem englobar en el sistema **DirectGUI**. DirectGUI ens permet la creació de botons, etiquetes, entrades de text i frames dins d'un programa. Aquests elements es poden personalitzar usant text, imatges i gràfics en 3D. Per poder interactuar amb ells cal associar mètodes als events generats pels elements. Aquests objectes estan emparentats per defecte amb el node *aspect2d*. *Aspect2d* ens permet col·locar elements sobre la pantalla, per tant, la seva situació no es veu afectada quan l'usuari navega pel món 3D.

Tots els objectes de DirectGUI es construeixen d'una manera similar:

```
from direct.gui.DirectGui import *
myObject = Directxxxxxx(keyword=value, keyword=value, ...)
```

Cada objecte pot contenir alguna de les 4 peces fonamentals que determinen la seva aparença. Pot contenir un text opcional, una geometria opcional, una imatge opcional i un frame opcional. Per utilitzar una etiqueta de text s'utilitza la keyword *text*. Per poder personalitzar la geometria d'un objecte es poden utilitzar arxius *egg* generats a partir d'imatges. S'utilitza la keyword *geom* per especificar l'arxiu. Si volem afegir una imatge a l'objecte utilitzarem la keyword image. Sempre que creem un objecte DirectGUI ens cal un frame on dibuixar-lo, normalment aquest és un rectangle gris. Podem personalitzar aquest frame i dotar-lo d'un estil. Per seleccionar un estil hem d'utilitzar la keyword *relief*. Els estils disponibles són *sunken* (enfonsat), *raised* (elevat), *groove* (foradat) o *ridge* (arrugat).

El elements DirectGUI que hem utilitzat en aquest projecte són els següents:

1. **DirectButton** és un objecte que respon a la interacció amb el mouse i executa la funció que li assignem quan l'usuari clica l'objecte. El següent fragment de codi mostra com crear aquest tipus d'objecte.

```
b = DirectButton(text = ("OK", "click!", "rolling over", "disabled"),
command=click)
```

2. **DirectEntry** crea un camp on l'usuari pot introduir text. El cursor parpadeja quan es troba dins d'un DirectEntry i se'ns permet utilitzar les fletxes i esborrar

amb el teclat. Suporta una línia única de text amb longitud fixa o múltiples línies. El següent fragment de codi mostra com crear aquest tipus d'objecte.

```
b = DirectEntry(text = "" ,scale=.05,command=setText,
initialText="Type Something", numLines =
2,focus=1,focusInCommand=clearText)
```

3. Un **DirectFrame** és un contenidor per múltiples objectes DirectGUI. Això ens permet controlar varis objectes a la vegada emparentant-los al mateix frame. Quan els objectes estan emparentats amb un frame, es poden posicionar relatius a ell. El següent fragment de codi mostra com crear aquest tipus d'objecte.

```
myFrame = DirectFrame(frameColor=(0, 0, 0, 1),   frameSize=(-1, 1, -1,
1), pos=(1, -1, -1))
```

Per emparentar objectes a un DirectFrame s'utilitza el paràmetre parent. Aquest paràmetre s'inclou en la definició de l'objecte que volen emparentar especificant el frame pare.

4. **DirectOptionMenu** és una classe que implementa un menú desplegable amb un nombre arbitrari d'elements. Està format per la barra del menú, el marcador desplegable i el propi menú desplegable. El menú desplegable apareix quan l'objecte es clicat i desapareix quan l'usuari torna a clicar.

```
menu = DirectOptionMenu(text="options",
scale=0.1,items=["item1","item2","item3"],initialitem=2,
highlightColor=(0.65,0.65,0.65,1),command=itemSel)
```

5. **DirectScrolledLists** crea una llista d'objectes gràfics DirectGUI. Cada objecte es crea individualment i es pot afegir a la llista.

```
myScrolledList = DirectScrolledList(incButton_propertyName = value,
decButton_propertyName = value)
```

6. **DirectSlider** ens permet crear sliders. Un slider és un elemnt gràfic que permet a l'usuari seleccionar un valor dins d'un interval limitat. Un DirectSlider consisteix en una llarga barra, que té un botó especial que l'usuari pot moure d'esquerra a dreta.

```
slider = DirectSlider(range=(0,100), value=50, pageSize=3,
command=showValue)
```

5.3 DISPLAY REGIONS

Les finestres de Panda3D no poden renderitzar res per pantalla si no hi ha almenys una DisplayRegion. Una DisplayRegion és una zona rectangular de la pantalla que conté una escena, vista des d'una càmera. Quan creem una finestra aquesta internament crea una DisplayRegion que la ompla. Per tant, si volem dividir la pantalla en diferents vistes de càmeres hem de crear DisplayRegions i associar-les a les càmeres. A la Figura 66 tenim diferents exemples de finestres utilitzen DisplayRegions.

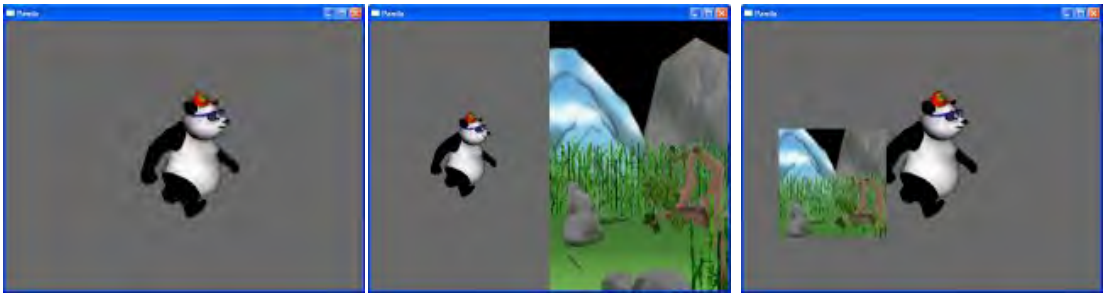

FIGURA 66 A L'ESQUERRA, UNA FINESTRA AMB UNA DISPLAY REGION; AL MIG, UNA FINESTRA AMB DUES DISPLAYREGIONS; I A LA DRETA, UNA FINESTRA AMB DUES DISPLAYREGIONS AMB UNA DISTRIBUCIÓ DIFERENT A L'ANTERIOR

Per crear una DisplayRegion hem de fer el següent:

```
win.makeDisplayRegion()
win.makeDisplayRegion(left, right, bottom, top)
```

Els paràmetres d'entrada left, right, bottom, top van de 0 a 1, on 0 està a l'esquerra i part inferior de la finestra, i 1 es troba a la dreta i part superior de la finestra.

Una nova DisplayRegion no renderitzarà fins que no li associem una càmera. Cada DisplayRegion ha de tenir només una càmera associada. El següent fragment de codi ens permet crear una càmera.

```
camNode = Camera('cam')
camNP = NodePath(camNode)
displayRegion.setCamera(camNP)
```

Un cop tenim una càmera, aquesta s'ha d'afegir a l'scene graph. Si el que volem és obtenir una nova vista d'una escena diferent, hem de crear un nou scene graph i enparentar la càmera amb aquest. El codi següent mostra com fer-ho.

```
render2 = NodePath('render2')
camNP.reparentTo(render2)
env = loader.loadModel('environment.egg')
env.reparentTo(render2)
```

5.4 Reproducció de vídeo amb Panda3D

Panda3D permet la reproducció de vídeos utilitzant-los com a textures de models 3D. El subsistema de reproducció de vídeo està implementat usant FFMPEG. Per tant, tots els formats que FFMPEG suporta estan suportats.

Per assignar una textura de vídeo a un objecte de l'escena 3D utilitzarem el codi següent:

```
myMovieTexture=loader.loadTexture('myMovie.avi')
myObject.setTexture(myMovieTexture)
```

Un cop tenim el fitxer de video carregat a una textura podem utilitzar diferents comandes per visualitzar el video. Aquestes comandes ens permeten les accions habitual de començar a reproduir I parar la reproducció. També podem mostrar la imatge correspondent a un temps concret I saber en quin instant de temps esta la reproducció. La repetició del video també esta permesa. Es pot fixar el nombre de repeticions I saber quantes repeticions s'han visualitzat. Una altra acció molt útil és canviar la velocitat de reproducció I saber si el video s'esta reproduint. Les crides a aquestes accions es troben a continuació.

```
myMovieTexture.play()
myMovieTexture.stop()
myMovieTexture.setTime(t)
myMovieTexture.getTime()
myMovieTexture.setLoopCount(n)
myMovieTexture.getLoopCount()
myMovieTexture.setPlayRate(speed)
myMovieTexture.getPlayRate()
myMovieTexture.isPlaying()
```

Panda3D també ens permet escoltir l'audio del video. Per fer-ho és necessary carregar l'arxiu de video dos cops: un com a textura I l'altre com a so:

```
mySound=loader.loadSfx("myMovie.avi")
```

Tot seguit, es necessàri sincronitzar les dues variables.

```
myMovieTexture.synchronizeTo(mySound)
```

5.5 IMPLEMENTACIÓ DE L'APLICACIÓ

Aquesta aplicació ens permet fer principalment dues tasques: configurar una terapia i visionar les sessions. Per tant, al iniciar-se l'usuari es trobarà en la pantalla Menú Inicial on podrà escollir entre Configurador de Terapies i Buscador de Sessions. L'usuari pot retornar al Menú inicial des de qualsevol d'aquestes pantalles. Si l'usuari es troba en el Buscador de Sessions, podrà accedir a la pantalla Sessió. Aquesta vindrà donada per la sessió seleccionada en el buscador. La Figura 67 mostra aquesta navegació.

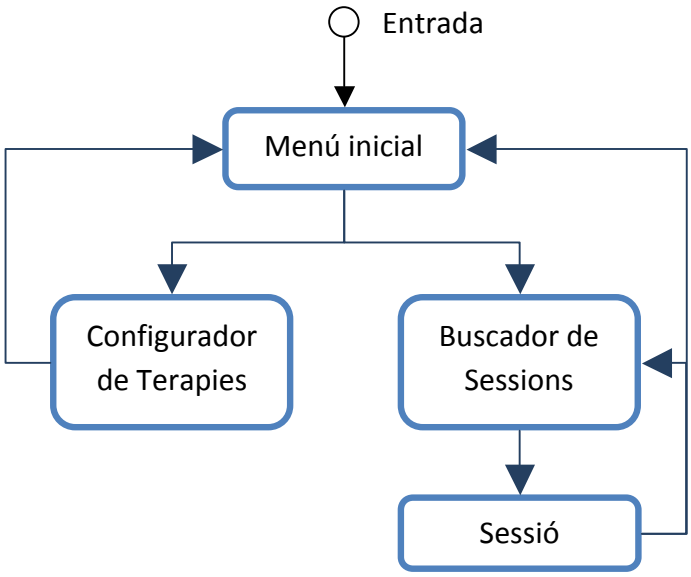

FIGURA 67 NAVEGACIÓ DE L'APLICACIÓ FISIOTERAPEUTA

De la mateixa manera que el client de l'aplicació Pacient, l'aplicació fisioterapeuta s'ha implementat amb Panda3D. Per tant, la classe principal inicialitza el sistema, gestiona els events i les pantalles. També s'ha implementat la classe Database per consultar a una base de dades. La classe CreaConfiguracio és l'encarregada de generar els arxius XML de configuració. Les classes Kinect2Matrix, Matrix i dataDrawer s'utilitzen a l'hora de visionar les sessions. El diagrama de classes de la Figura 68 descriu les relacions entre les classes esmentades.

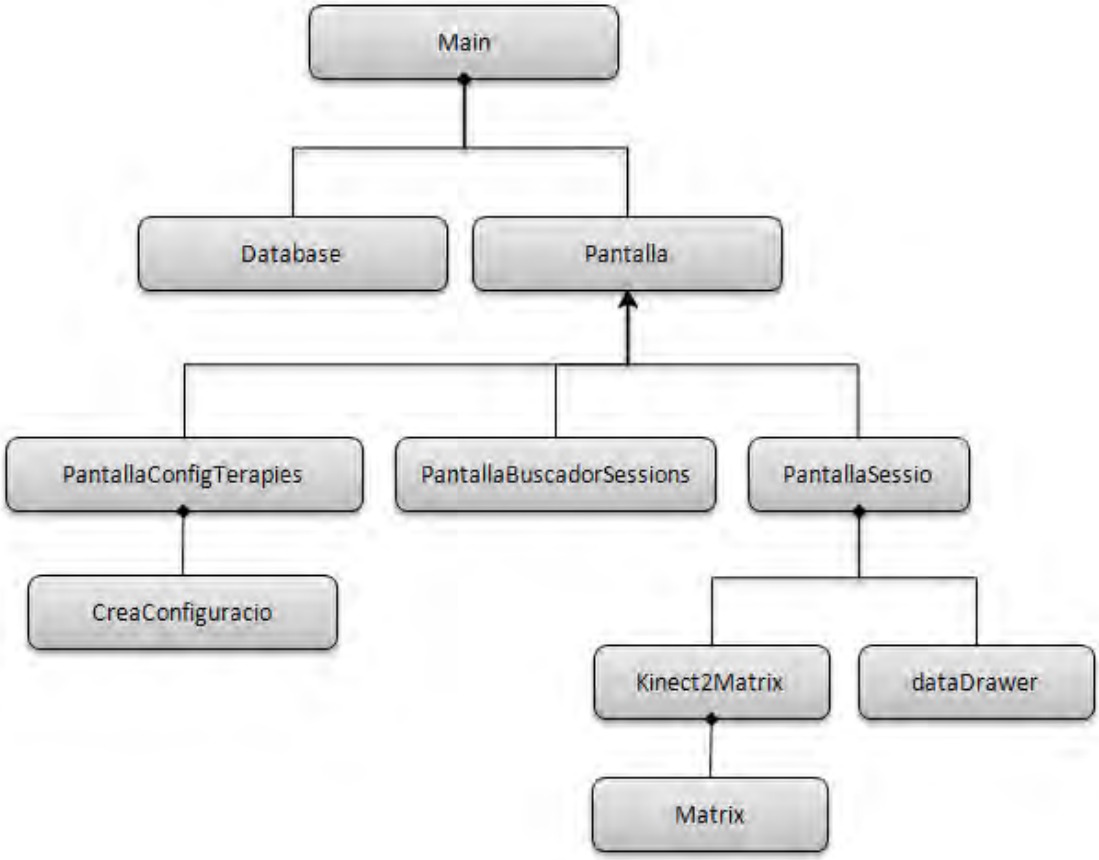

FIGURA 68 DIAGRAMA DE CLASSES DE L'APLICACIÓ FISIOTERAPEUTA

5.5.1 Classe Main

La classe Main, que hereta de DirectObject, és la classe principal de l'aplicació. Aquesta classe inicialitza dos objectes: un de tipus Database i l'altra de tipus Pantalla. També s'inicialitzen els events d'usuari I es gestionen les pantalles. La gestió de pantalles es fa de la mateixa manera que 4.9.1 Classe Main.

5.5.1 Base de dades

La aplicació Fisioterapeuta tracta amb la informació d'una base de dades. Aquesta base de dades conté informació sobre les terapies. Les terapies estan caracteritzades per fases, que a la vegada contenen exercicis. Aquesta informació es guarda en les taules TipusTerapia, Fase i TipusExercici. L'aplicació també consulta les sessions que els

pacients realitzen. Els pacients són guardats en la taula Pacients, dels quals s'especifica el nom i els dos cognoms. Els pacients un cop entrats al sistema se'ls hi assigna una terapia nova. Aquesta nova terapia, d'un tipus de les terapies de la taula TipusTerapia, s'introduiex a la taula Terapia. Com ja hem explicat anteriorment, el fisioterapeuta pot configurar les sessions pel seu pacient i aquestes s'han de registrar quan el pacient les efectui. El registre de sessions es fa a la taula Sessio. Cada sessió està relacionada amb una terapia. Ademés, les terapies contenen exercicis que es guarden a la taula Exercici. Aquest exercicis són d'un dels tipus de la taula TipusExercici. El model entitat-relació de la base de dades es mostra a la Figura 69.

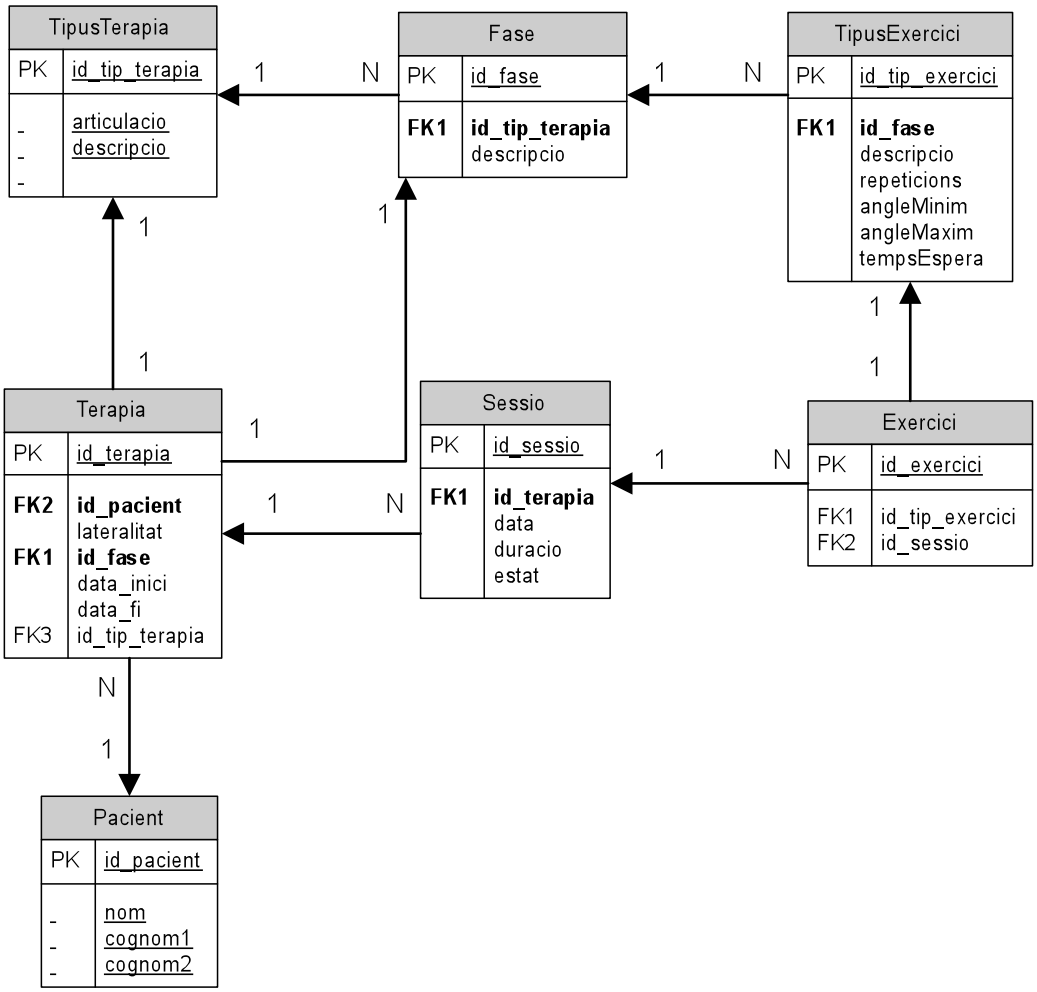

FIGURA 69 MODEL ENTITAT-RELACIÓ DE LA BASE DE DADES

L'implementació d'aquesta base de dades s'ha fet dins la classe Database utilitzant la llibreria Sqlite3. S'han implementat mètodes per la creació de la base de dades, inserció d'elements a les taules i mètodes per fer consultes.

5.5.2 PANTALLA INICIAL

Per l'implementació de la pantalla inicial, com la resta de pantalles, s'han utilitzat els elements de DirectGUI. Aquesta pantalla inclou dos DirectButtons que ens permeten accedir a les pantalles de Configuració de terapies i Buscador de Sessions. La Figura 70 mostra la distribució d'aquesta pantalla i descriu els objectes DirectGUI usats.

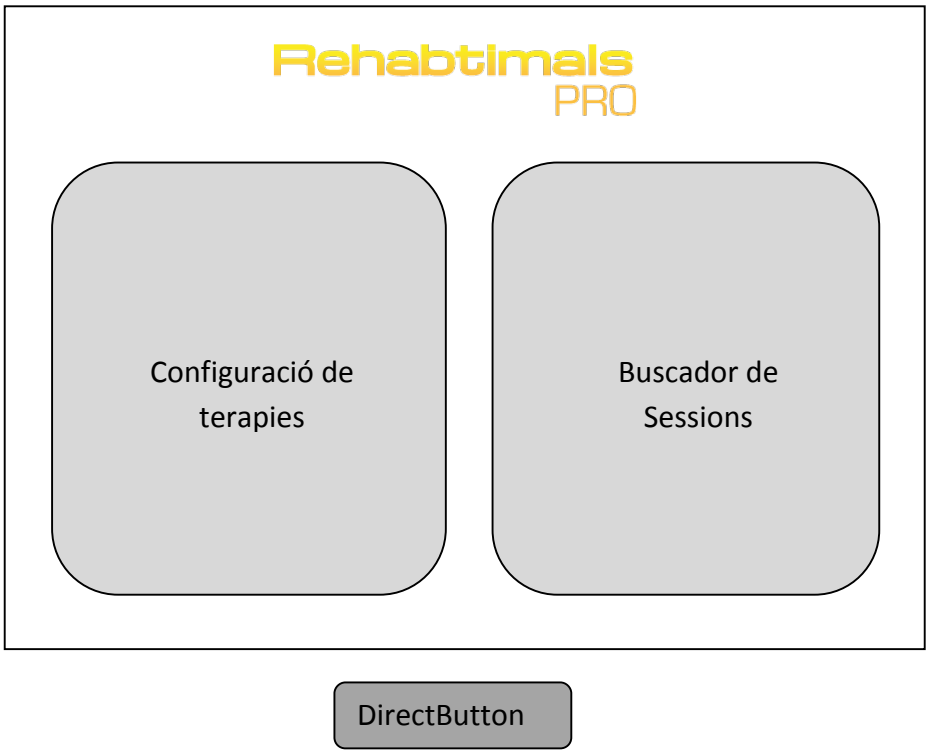

FIGURA 70 PANTALLA INICIAL

5.5.3 CONFIGURACIÓ DE TERAPIES

En aquesta pantalla és on el fisioterapeuta configura les terapies. La pantalla està dividida dues parts diferenciades: buscador i contingut. En el buscador hi ha una camp de text on escriure el pacient del qual volem configurar la terapia. Per buscar hem d'apretar a la tecla intro o al botó buscar. Aquest event farà una consulta a la base de dades i l'informació es mostrarà en el contingut. D'altra banda, si volem crear un pacient nou li donarem al botó crear. A la Figura 71 es poden veure els elements que composen l'interfície gràfica d'aquesta pantalla.

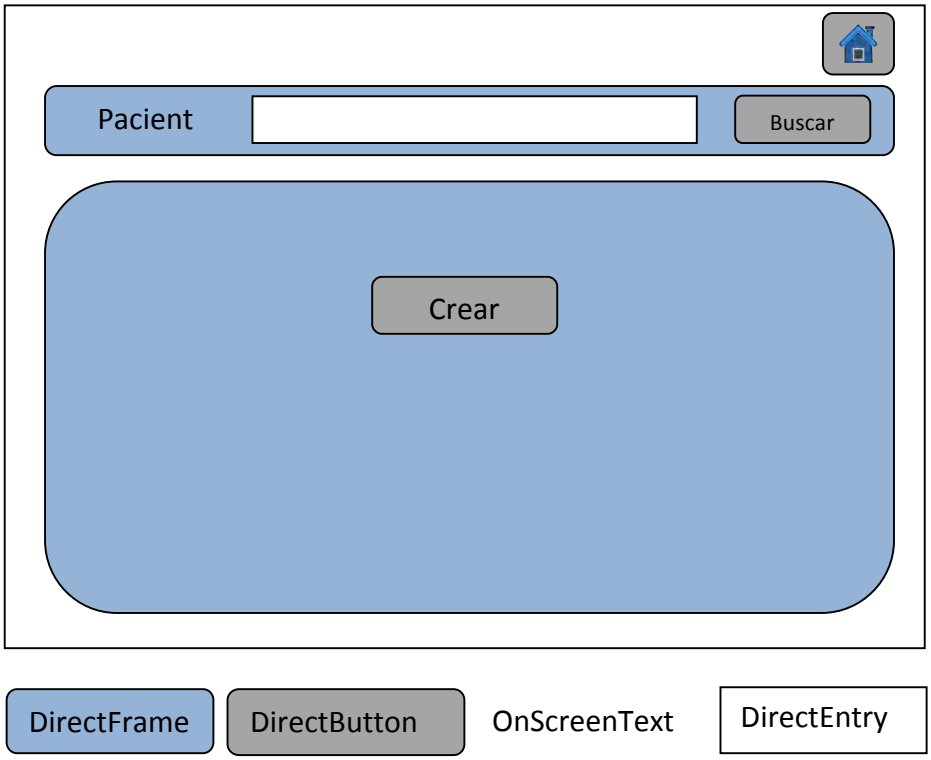

FIGURA 71 PANTALLA CONFIGURADOR DE TERAPIES I

Si decidim crear un pacient nou, se'ns mostrarà la pantalla de la Figura 72. Com podem observar, hi ha un formulari que el fisioterapeuta ha d'anar omplint per configurar la terapia. Un cop ja està tot omplert, si apretem al botó Crear es realitzaran una sèrie d'operacions. El primer que es farà, es introduir un nou element a les taules Pacients i Terapies de la base de dades. I a continuació, es crearà l'objecte CreaConfiguracio. Aquest objecte serà l'encarregat de volcar tota la informació del formulari a un arxiu XML. Aquest serà l'arxiu de configuració que el client de l'aplicació Pacient llegirà. Per fer-ho s'ha utilitzat la llibreria *xml.dom.minidom* de Python.

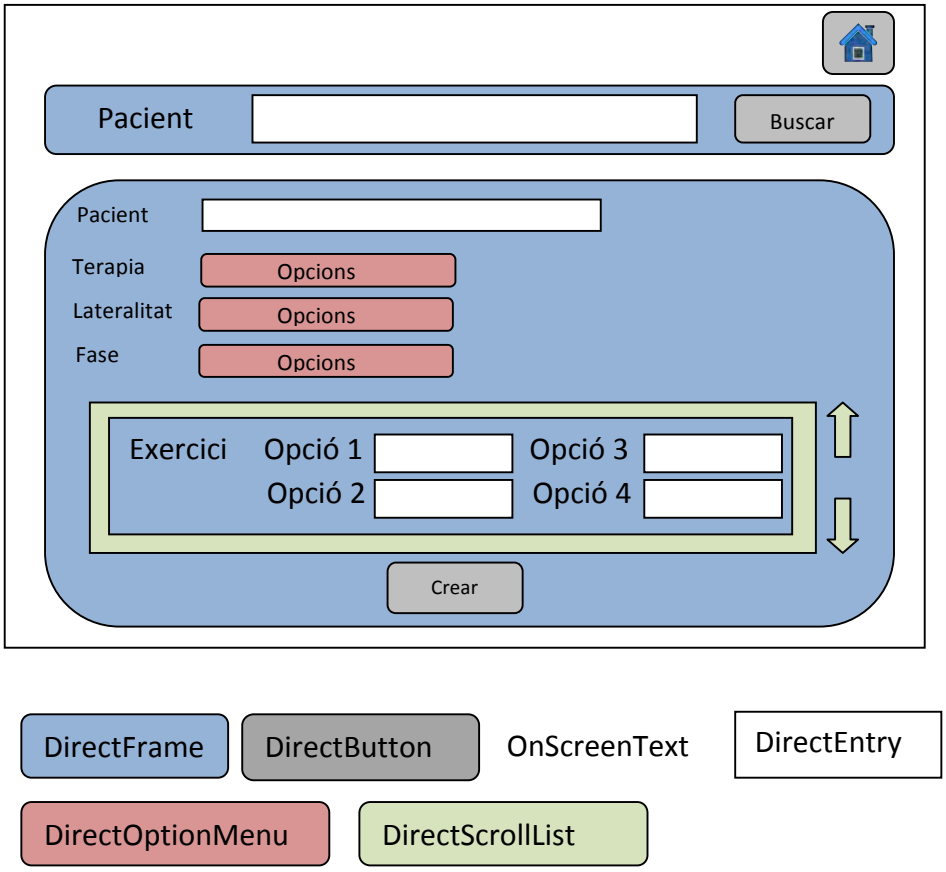

FIGURA 72 PANTALLA CONFIGURADOR DE TERAPIES II

5.5.4 BUSCADOR DE SESSIONS

La pantalla Buscador de Sessions té un camp d'entrada per introduir el nom del pacient que volem buscar. La cerca, que s'efectua clican al botó buscar o la tecla intro, fa una consulta a la base de dades buscant les sessions efectuades pel pacient que tinguin l'estat seleccionat del menú desplegable. A la part del contingut de la pantalla, es llisten totes les sessions trobades amb els requisits de cerca. Les sessions contenen una informació que les descriu i un botó per accedir a elles. La Figura 73 mostra la distribució d'aquesta pantalla i els tipus d'objectes utilitzats per mostrar la informació descrita.

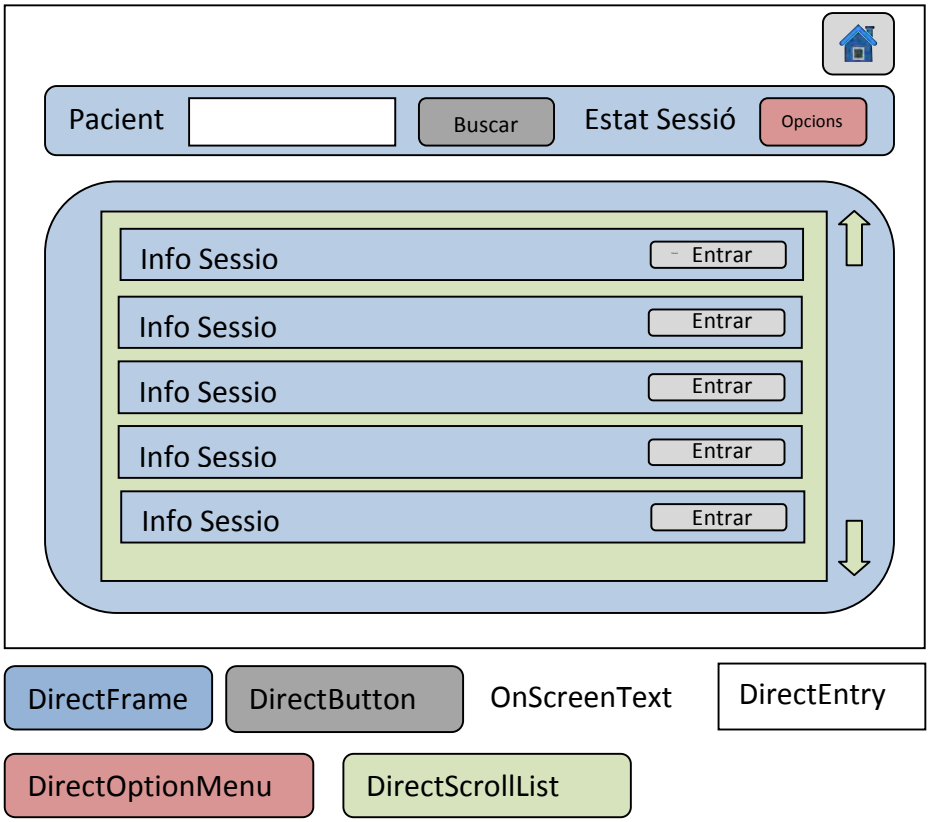

FIGURA 73 PANTALLA BUSCADOR DE SESSIONS AMB SESSIONS LLISTADES

5.5.3 PANTALLA SESSIÓ

Un cop el fisioterapeuta ha seleccionat la sessió que vol visualitzar, l'aplicació mostra la PantallaSessio. Aquesta pantalla conté la informació de la sessió i tres regions on es mostra la informació de la sessió en diferents formats (Figura 74). A la part superior, trobem un resum de la sessió on s'indica el nom del pacient, la terapia, la fase on es troba i la data de la sessió. Justament a sota, hi ha un visor que mostra les gràfiques de l'evolució de l'angle de l'articulació a recuperar. Sota de la gràfica, hi ha dues regions l'una al costat de l'altre. A la part esquerra, és on es mostra el vídeo i la part dreta, és un visor de moviments 3D.

Sobre de la gràfica a la part dreta, hi ha botons per reproduir, parar, avançar i retrocedir. I a la part inferior de la pantalla hi trobem un slider. Aquest controls serveixen per reproduir la informació de la sessió de manera sincronitzada a les tres regions. Per marcar el fotograma a la gràfica s'ha dibuixat una línia vertical que actua com a marca de temps.

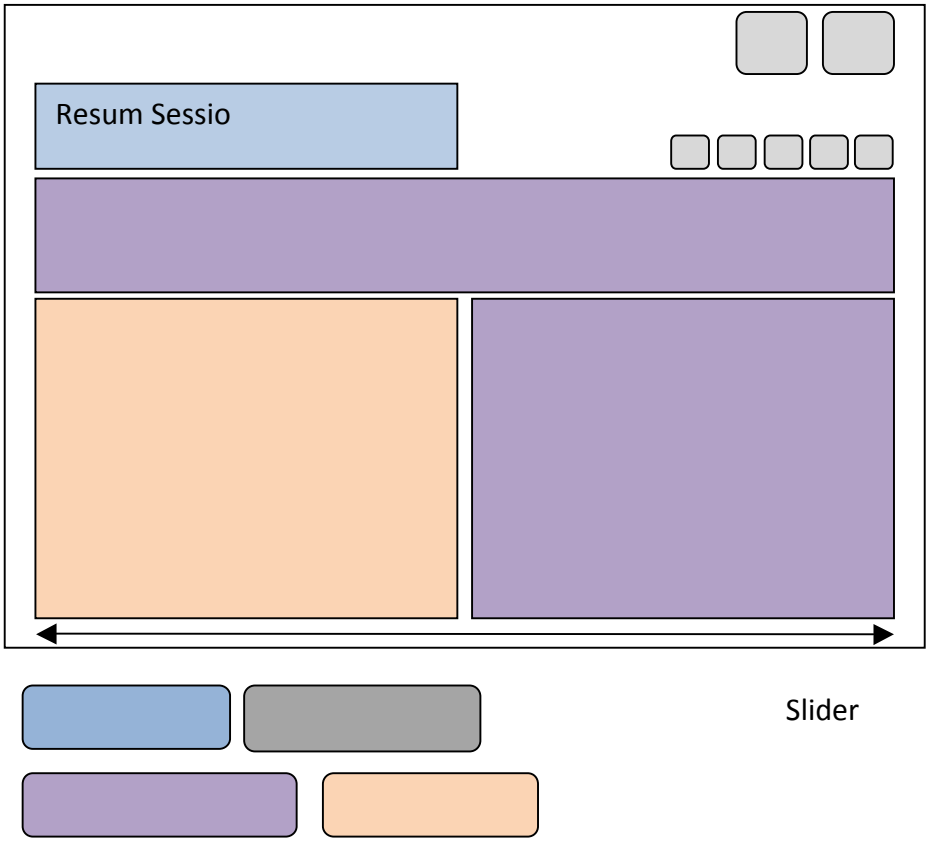

FIGURA 74 PANTALLA SESSIÓ

Per poder separar la finestra en diferents regions i mostrar diferents escenes s'han utilitzat DisplayRegions. Al tractar-se de tres escenes diferents la navegació entre elles ha de ser independent. Per aquest motiu hem creat un objecte MouseWatcher per cada DisplayRegion i els hem relacionat. El fragment de codi següent mostra com fer-ho.

```
self.myMouseWatcher2D = MouseWatcher()
base.mouseWatcherNode.getParent(0).addChild(self.myMouseWatcher2D)
self.myMouseWatcher2D.setDisplayRegion(self.dr3)
```

Per poder accedir a la posició x i y del mouse hem utilitzat,

```
x=self.myMouseWatcher2D.getMouseX()
y=self.myMouseWatcher2D.getMouseY()
self.mousePos = [x,y]
```

5.5.3.1 CÀRREGA DE L'ARXIU DE MOVIMENT

Al iniciar-se la PantallaSessio el primer que fem és carregar l'arxiu de moviment de la sessió seleccionada. Hem implementat la classe Kinect2Matrix. Aquesta classe s'encarrega de parsejar la informació de l'arxiu de moviment corresponent. Recordem que el format d'aques arxiu és l'indicat a Figura 30. Per emmagatzemar el moviment s'ha utlitzat la classe Matrix, que crea una matriu i té mètodes d'inserció i consulta d'elements. Per tant, el moviment queda guardat dins d'una matriu. On les files són els fotogrames i les columnes les posicions de les diferents articulacions.

5.5.3.2 VISOR 3D

El visor de moviments 3D s'ha implementat col·locant esferes 3D a les posicions llegides del l'arxiu de moviment. S'ha utilitzat la variable frame per saber en quin fotograma del moviment es troba la reproducció. Sabent en quin fotograma ens trobem, només tenim que accedir a les posicions de les articulacions per col·locar les esferes on toquen. Aixo és molt senzill ja que només hem de llegir el valor de la matriu en la fila pertanyent al fotograma i la columna a la coordenada pertinent.

Una altra funcionalitat que té aquest visor és que podem remarcar articulacions fent clic sobre d'elles per fixar-nos en el seu comportament. Al seleccionar-les canviarem el seu color pel color vermell. Quan les cliquem també se'ns informa de quina articulació es tracta. El mètode de la classe encarregat de fer aquesta funció és picking3D().

Per fer encara més útil aquest visor hem implementat el zoom, el moviment de la càmera dins del món 3D i la rotació de la càmara. D'aquesta manera podem navegar i obtenir els punts de vista desitjats. Les funcions encarregades d'això són Zoom(), dragNdrop() i rotateWorld().

5.5.3.3 VISOR DE VÍDEO

Per poder mostrar el vídeo s'han utilitzat les comandes descrites en 5.4 Reproducció de vídeo amb Panda3D. S'han utilitzat els mètodes getTime() i setTime() per poder sincronitzar la reproducció del vídeo amb els moviments 3D, guiant-nos sempre per la variable frame que es la que marca el fotograma actual.

5.5.3.4 VISOR DE GRÀFIQUES

Per implementar el visor de gràfiques ha estat necessàri dibuixar línies en l'escena mostrada. Per fer-ho s'han utilitzat les comandes que hi ha a continuació:

```
segs = LineSegs( )
segs.setThickness( 2 )
segs.setColor( Vec4(0.8,0.8,0.8,1) )
segs.moveTo( punt1)
segs.drawTo( punt2)
```

S'ha implementat la classe dataDrawer per dibuixar tots els elements gràfics necessaris. Aquesta classe incorpara mètodes per pintar la graella i els valors dels angles de rotació. També s'ha implementat el mètode dibuixaMarca(). Aquest mètode dibuixa una línia horitzontal de color vermell que ens permet situar la gràfica en el fotograma actual.

En aquest visor també s'han implementat mètodes per poder fer zoom i moure'ns per la gràfica utilitzant el mouse.

5.6 RESULTATS

En aquest punt es mostren captures de pantalla de l'aplicació fisioterapeuta. L'aspecte d'aquesta aplicació no és el definitiu a falta de la implementació del disseny gràfic. La Figura 75 mostra una captura de la pantalla incial de l'aplicació. Com podem observar, aquesta pantalla consta de dos botons: un per accedir a la configuració i l'altre a les teràpies realitzades.

FIGURA 75 PANTALLA INICIAL DE L'APLICACIÓ FISIOTERAPEUTA

Si decidim accedir a la configuració de les teràpies l'aplicació ens portarà cap a Figura 76. En aquesta figura podem veure que hi ha un buscador i una zona on es mostra el contingut. En el contingut veiem un formulari que ens permet crear la configuració de la teràpia per un pacient.

FIGURA 76 PANTALLA DE CONFIGURACIÓ DE TERÀPIES

Si ens trobem en la pantalla de llistat de sessions el resultat serà el de la Figura 77. Aquesta pantalla ens mostra el resultat de la cerca de sessions indicant el pacient que les ha realitzat. Clicant al botó veure podem accedir a visualitzar la sessió en concret.

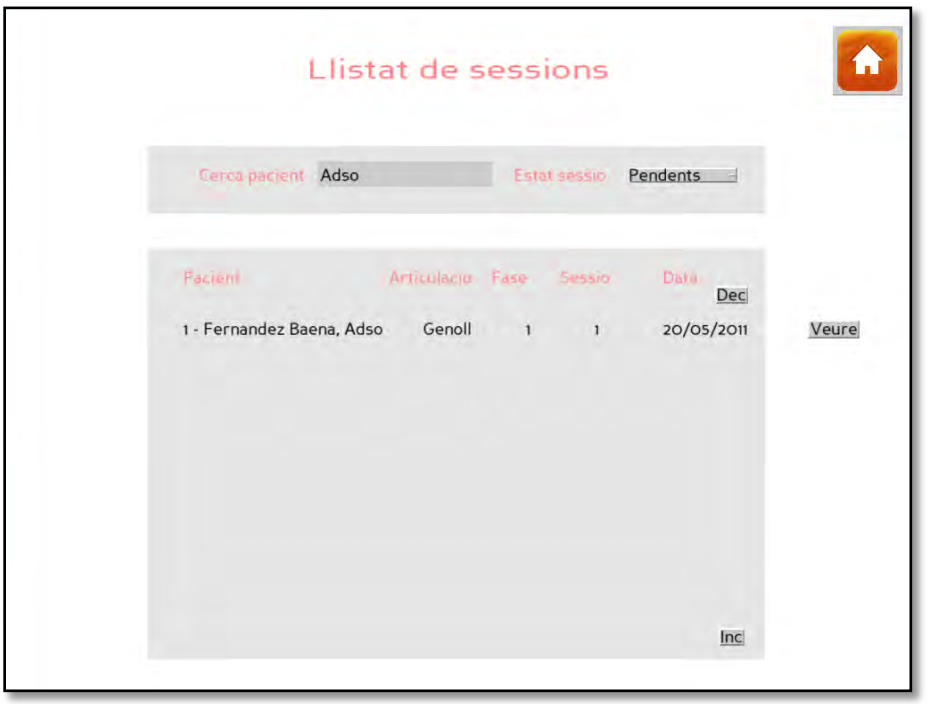

FIGURA 77 PANTALLA DE LLISTAT DE SESSIONS

La Figura 78 mostra la pantalla de visualització de sessions. Com podem observar a la part superior tenim una gràfica que ens diu l'angle de rotació de l'articulació implicada en l'exercici. A la part inferior, trobem dos visualitzadors: un on veiem el vídeo capturat de la càmera del Kinect, i l'altre on trobem una representació de l'esquelet estimat pel sistema de captura. També podem veure que hi ha uns botons i un slider per controlar la reproducció de les dades.

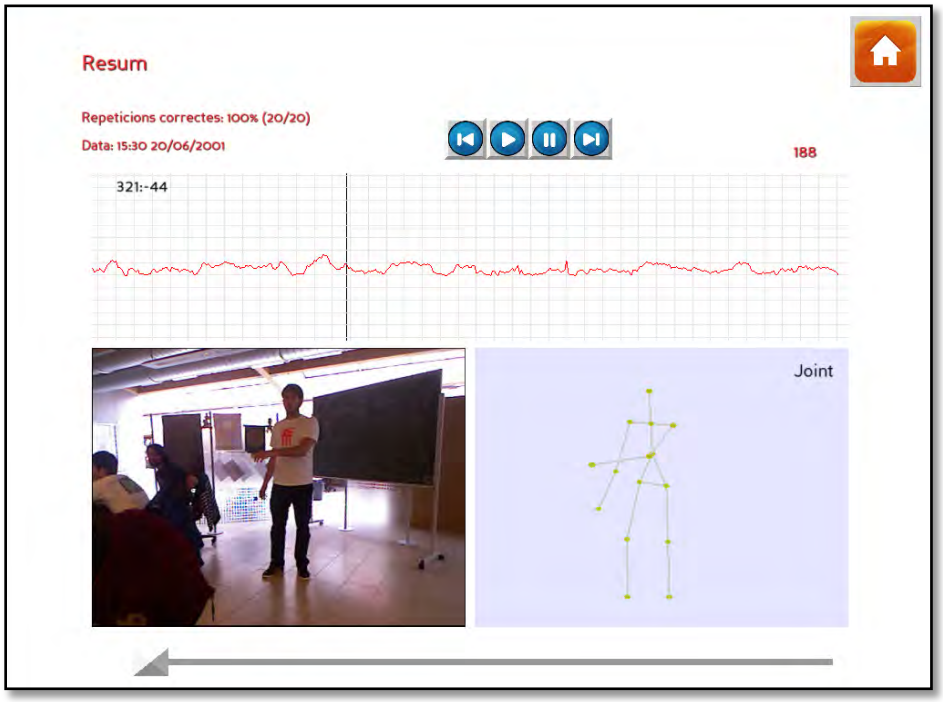

FIGURA 78 PANTALLA DE VISUALITZACIÓ DE LES SESSIONS

6. CONCLUSIONS

Aquest projecte s'ha composat de diverses parts ben diferenciades. Per tant, farem una avaluació dels resultats explicant i justificant els punts assolits en cadascuna d'elles.

Els resultats obtinguts en la validació biomecànica mostren la fiabilitat de les dades capturades per Kinect. En aquest projecte s'han avaluat les rotacions del genoll i del maluc. En el cas del genoll s'han obtingut errors que varien d'entre 5,07° i 15,95°. Tot i això, en la majoria de les animacions analitzades aquest error és inferior a 10°. Pel que fa a les rotacions del maluc els resultats són millors. L'error màxim obtingut és de 9,22° i el mínim de 5,89°. Hem donat com a bons aquests resultats ja que la precisió de l'ull humà en calcular la rotació està més o menys en 10°, per tant, la captura que realitza Kinect és molt propera i en la majoria de casos millor que la visual.

S'ha desenvolupat un serious game per la rehabilitació de genoll, tal com es va dir. Es va plantejar la possibilitat d'expandir aquesta eina per tal de fer rehabilitació d'espatlla i cervicals. Finalment es van desestimar aquestes teràpies per falta de temps i es va reduir el serious game a les fases 1 i 2 de la rehabilitació de genoll. El serious game plantejat contava amb un univers de personatges dividit en tres ambients. Finalment s'ha implementat només un personatge. Tot i això, aquesta aplicació ha estat implementada en la seva totalitat i per tant, les dues primeres fases de la rehabilitació del genoll es poden realitzar utilitzant-la.

Un dels requisits més importants del serious game era que aquest fos capaç d'identificar les repeticions de moviments fetes per l'usuari. Per aconseguir-ho s'ha implementat amb èxit un algorisme. Aquest algorisme es pot utilitzar per fer la detecció de repeticions per qualsevol moviment on una articulació es mogui amb un grau de llibertat. Per fer-ho és necessari especificar els graus màxims i mínims que limiten la repetició.

D'altra banda, també és destacable que la navegació per l'aplicació Pacient es faci amb interacció natural. Implementar una aplicació amb captura de moviment i utilitzar una interfície gràfica controlada per un mouse o teclat, fa que l'experiència d'usuari no sigui tant bona. Per tant, l'aplicació implementada és coherent i no trenca l'experiència d'usuari ni la immersió.

En un principi, es va plantejar fer la configuració de les teràpies de manera manual modificant un arxiu XML. Finalment s'ha desenvolupat l'aplicació fisioterapeuta incorporant aquesta funcionalitat i d'altres. Un dels punts més crítics d'aquesta

aplicació ha estat la sincronització dels moviments 3D amb el vídeo. Per aconseguir una sincronització exacta s'ha guardat la informació tant del fotograma en imatge com de la postura en el mateix procediment. Per tant, podríem dir que la presentació de les dades de les sessions s'ha assolit de manera satisfactòria.

La gran complexitat del projecte i les diverses parts que el composen han fet que la unió d'aquestes sigui la tasca més complexa que hem realitzat. Fer que totes les parts siguin coherents les unes amb les altres no ha estat fàcil. Finalment, s'ha aconseguit donar-li un sentit a tot el projecte tancant així el tot el procés de rehabilitació. Això és el que fa més valuós aquest projecte.

7. LÍNIES DE FUTUR

Com tot projecte, hi ha molts aspectes a millorar. En el procés de captura de moviment amb OpenNI i NITE utilitzant el Kinect se li podrien afegir algorismes per millorar els resultats de les postures estimades. Com ve diuen els autors de les llibreries, les longituds de les articulacions estimades pel sistema varien. Aquest fet fa que les dades no s'ajustin al model d'esquelet humà ja que aquest té óssos de longitud fixa. Es podrien aplicar algorismes de cinemàtica inversa i retargeting per assegurar almenys la correcció física de les postures. També es podria utilitzar un filtre més sofisticat que l'utilitzat en aquest projecte per eliminar el soroll de les dades capturades. Un altre aspecte a millorar és la col·locació de les articulacions estimades, que no s'ajusta a la situació real. Per fer-ho es podria optimitzar el procés ajudant-nos de la silueta i el mapa de profunditats també calculat per les eines utilitzades. Seguint el fil de millorar la captura es podrien utilitzar varis Kinects per capturar millor moviments complexos i aquells que produeixen oclusions entre les articulacions. Per encara fer millor aquest sistema de captura es podrien implementar mòduls per estimar les posicions de les articulacions que no es calculen actualment pel sistema.

Pel que fa a la validació biomecànica les línies de futur són previsibles. Validar totes les articulacions estimades pel Kinect seria el següent pas, ja que en aquest projecte només hem contrastat les articulacions implicades amb els exercicis escollits. Millorar l'algorisme de sincronització o gravar amb els dos sistemes de captura de manera sincronitzada també es un aspecte a millorar. Per acabar, també es podria utilitzar un procés de calibració entre els dos sistemes de coordenades per així poder contrastar les posicions directament, sense necessitat de calcular les rotacions relatives.

L'aplicació Pacient conté la teràpia de rehabilitació del genoll. Aquesta teràpia no s'ha implementat en la seva totalitat, restant per incloure en el joc la Fase 3. Afegir aquesta fase i més teràpies de rehabilitació física farien el producte més valuós. Pel que fa la història del serious game està dissenyat amb varis personatges que guien les mateixes rehabilitacions, per no avorrir als pacients. Actualment l'aplicació s'ha implementat amb un sol personatge, per tant, afegir més personatges a l'aplicació també seria una millora substancial. Una altra aspecte per millorar l'experiència d'usuari seria eliminar el procés de calibració. Per altra banda, aquesta aplicació ha estat implementada per executar-se en local. Es podria modificar la implementació d'aquesta per que es pogués executar en un servidor i ser visionada a través d'un explorador d'internet.

Si ens fixem en l'aplicació fisioterapeuta veiem que el seu potencial també és molt gran. Afegir noves teràpies a la base de dades o més funcionalitats a l'aplicació serien

una millora. Es podrien implementar noves pantalles per poder editar les configuracions creades o veure en un calendari les sessions realitzades pel pacient. També es podria implementar la transmissió de dades en viu per què el fisioterapeuta pogués fer el seguiment en directe. Inclòs, es podria dotar d'un algorisme intel·ligent a l'aplicació perquè avalues les dades enregistrades. D'aquesta manera el fisioterapeuta estalviaria temps.

BIBLIOGRAFIA

[1] Michael W. Whittle, *An Introduction to Gait Analysis*, 4th ed., Michael W. Whittle, Ed.: Butterworth Heinnemann, 2007.

[2] DH Sutherland, "The evolution of clinical gait analysis: Part II Kinematics," *Gait & Posture*, vol. 16, pp. 159-179, 2002.

[3] Russell Best and Rezaul Begg, "Overview of Movement Analysis and Gait Features," *Computational Intelligence for Movement Sciences: Neural Networks and Other Emerging Technique*, pp. 11–18, 2006.

[4] Chih-Chang Yu, Jenq-Neng Hwang, Gang-Feng Ho, and Chaur-Heh Hsieh, "Automatic Human Body Tracking and Modeling from Monocular Video Sequences," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007*, April 2007, pp. IEEE International Conference on , vol.1, no., pp.I-917-I-920, 15-20.

[5] Nicholas R. Howe and Michael E. Leventon and William T. Freeman, "Bayesian reconstruction of 3d human motion from single-camera video," in *Advances in Neural Information Processing Systems*, 1999, pp. 820--826.

[6] Maria Cruz Villa Uriol, "Video-Based Avatar Reconstruction and Motion Capture," University of California, Irvine, Phd Thesis 2005.

[7] R. Kehl, M. Bray, and L. Van Gool, "Full body tracking from multiple views using stochastic sampling," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference* , 20-25 June 2005, pp. pp. 129- 136 vol. 2.

[8] Mesa imaging. (2011, Juliol) [Online]. http://www.mesa-imaging.ch

[9] L. Wang, R. Yang, and J. Davis. J. Zhu, "Fusion of time-of-flight and stereo for high accuracy depth maps.," *IEE Conference on Computer Vision and Pattern Recognition*, 2008.

[10] C. Zhang, R. Yang, and C. Zhang L. Wang, "Towards robust real-time foreground extraction using time-of-flight camera," *3DPVT*, 2010.

[11] H. Li, D. Losowyj, and V. Prakash J.Decker, "Wiihabilitation: rehabilitation and

wrist flexion and extension using a wiimote-based game system.," *Governor's Shool of Engineering and Technology Research Journal*, 2009.

[12] A. Desai, C. Nett, N. Kapadia, and T.Szturm A. L. Betker, "Game-based exercises for dynamic short-sitting balance rehabilitation of people with crhonic spinal cord and traumatic brain injuries," in *Physical Therapy*, 2007, pp. 87 (10): 1389.

[13] M.D.J. McNeill, D.K. Charles, P.J. Morrow, J.H. Crosbie, and S.M. McDonough J.W. Burke, "Optimising engagement for stroke rehabilitation using serious games.," in *The Visual Computer*, 2009, pp. 25 (12): 1085-1099.

[14] M. Ma and K. Bechhoum, "Serious games for movement therapy after stroke.," in *IEEE International Conference Systems, Man and Cybernetics, 2008.* , 2009, pp. 1872-1877.

[15] Microsoft. (2010) Kinect full body interaction. [Online]. http://www.xbox.com/kinect

[16] D. Krum, B. Lange, S.Rizzo, and M. Bolas E. Suma, "Faast: The flexible action and articulated skeleton toolkit," *IEE Virtual Reality*, 2011.

[17] P. Rego, P.M Moreira, and L.P. Reis, "Serious games for rehabilitation: A survey and a classification towards a taxonomy," in *Information Systems and Technologies (CISTI), 5th Iberian Conference*, Juny 2010, pp. 1-6,16-19.

[18] T. Ganchev, O. Kocsis, G. Papadopoulos, F. Fernández-Aranda, and S. Jiménez-Murcia A. Conconi, "PlayMancer: A Serious Gaming 3D Environment," in *International Conference on Automated Solutions for Cross Media Content and Multi-channel Distribution*, 2008, pp. 111-117.

[19] L. Latini-Corazzini, F. D'Agata, F. Cauda, K. Sacco, S. Monteverdi, M. Zettin, S. Duca, G. Geminiani M. Caglio, "“Video game play changes spatial and verbal memory: Rehabilitation of a single case with traumatic brain injury," in *Journal of Cognitive Processing*, 2009, pp. S195--S197.

[20] S. B. Badia, L. Zimmerli, E. D. Oller, P. F. M. J. Verschure M. S. Cameirão, "The Rehabilitation Gaming System: a Review," in *Studies in Health Technology and Informatics*, 2009, pp. vol. 145, pp. 65-83.

[21] S. Smith, B. Chung, S. Cossell, N. Jackman, J. Kong, O. Mak, i V. Ong. M. Ryan. (2011, Juliol) Viviang Ong. [Online]. http://vivianong.com/work/balance-rehabilitation/

[22] RehaCom. (2011, Juny) Rehacom cognitive rehabilitation. [Online]. http://www.rehacom.com

[23] Labsid SL. (2011, Juliol) Labsid. [Online]. http://www.labsid.com

[24] Lina Marcelina Zapata Zapata. (2011, Juny) Ejercicios básicos para la zona central del cuerpo. [Online]. http://viref.udea.edu.co/contenido/pdf/182-ejercicios.pdf

[25] Marc Rodríguez, "Rehabtimals: Creació d'un nou òs i pla de marketing," La Salle - Universitat Ramon Llull, Barcelona, Treball final de màster (TFM) 2011.

[26] Python. (2011, Juliol) PEP 249. [Online]. http://www.python.org/dev/peps/pep-0249/

[27] Peter Norgaard, *Plotting in Matlab - A quick tutorial*.

[28] Entertainment Technology Center, Carnegie Mellon University, *Marker placement guide*.: http://www.etc.cmu.edu/projects/mastermotion/Documents/markerPlacementGuide.doc, 2005.

[29] C. Wren, A. Azarbayejani, T. Darrell, and A Pentland, "Pfinder: real-time tracking of the human body," in *Automatic Face and Gesture Recognition, Proceedings of the Second International Conference*, 14-16 Oct 1996, pp. Vol., no., pp.51-56.

[30] Nicholas R. Howe and Michael E. Leventon and William T. Freeman, "Bayesian reconstruction of 3d human motion from single-camera video," in *Advances in Neural Information Processing Systems*, 1999, pp. 820-826.

[31] Panda3D. (2011, Juny) Panda3D Download. [Online]. http://www.panda3d.org/download.php