UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

Projecte final de carrera:

Sistema d'interacció basat en Kinect per a Realitat Virtual

Miguel Angel Vico Moya

Director: Pere Brunet Crosa
Ponent: Carlos Antonio Andújar Gran

Departament de Llenguatges i Sistemes Informàtics (LSI)

Dades del Projecte

Títol: **Sistema d'interacció basat en Kinect per a Realitat Virtual**

Nom de l'estudiant: **Miguel Angel Vico Moya**
Titulació: **Enginyeria en Informàtica Superior**
Crèdits: **37,5**
Director/Ponent: **Pere Brunet Crosa / Carlos Antonio Andújar Gran**
Departament: **Llenguatges i Sistemes Informàtics (LSI)**

Membres del tribunal

President: **Alvar Vinacua Pla**

Signatura:

Vocal: **Pilar Sobrevilla Frison**

Signatura:

Secretari: **Carlos Antonio Andújar Gran**

Signatura:

Qualificació

Qualificació numèrica:
Qualificació descriptiva:

Data:

Agraeixo al grup de recerca
MOVING el suport que he rebut tant
en recursos com en formació.
M'agradaria agrair al meu tutor de projecte
Pere Brunet, i a Carlos Andújar,
l'ajuda i consells que he rebut per part seva.
També vull agrair l'ajuda i ànims
rebuts per part de la meva família i amics.

Índex

Part I

Presentació del projecte

Capítol 1

Introducció al projecte

Un dels problemes comuns en el món dels gràfics per computador i, concretament, en el món de la Realitat Virtual, és la usabilitat de les interfícies de comandament. Tota bona interfície de comandament ha de ser intuïtiva i fàcil de fer funcionar, de manera que, el temps d'aprenentatge d'utilització de la mateixa sigui el més curt possible i que qualsevol persona pugui fer-la servir.

Hi ha molts tipus de dispositius i interfícies que ens permeten interactuar amb entorns de Realitat Virtual (veure secció 2.3), la majoria però tenen un preu de mercat important o obliguen a l'usuari a tenir una mínima agilitat amb l'ús de comandaments electrònics (controls de videoconsoles per exemple).

Actualment han aparegut certs dispositius, com és el cas de Kinect (veure secció 6.1), que a part de tenir un preu assequible, permet construir interfícies intuïtives i fàcils d'utilitzar per a qualsevol persona. A més, un dels avantatges d'aquest tipus d'interfície és el fet de què no es necessiten elements externs per interactuar amb l'entorn de Realitat Virtual, és a dir, no és necessari l'ús d'aparells de comandament (que normalment han d'estar endollats mitjançant cables), els quals moltes vegades fan que la interacció sigui incòmoda o problemàtica.

Aquests tipus de dispositius bàsicament funcionen obtenint un mapa de profunditat gràcies a una càmera d'infrarojos. Amb aquest mapa, és possible reconstruir una representació simplificada, o esquelet, de les persones que estan situades davant la càmera (més endavant es realitza una explicació més detallada del funcionament d'aquests dispositius: secció 6.1.3). No és que aquest procediment sigui actual, el realment revolucionari és que amb aquests dispositius és possible realitzar totes aquestes tasques en el controlador del mateix i de forma molt més ràpida. Això fa que sigui adient l'ús d'aquests dispositius per a la construcció d'interfícies per controlar entorns de Realitat Virtual, en els quals es necessita una resposta ràpida tant en les interfícies d'entrada com en les de sortida.

Aquest projecte en concret, es centra en l'ús de la càmera Kinect per tal de construir una

aplicació que utilitzi les dades proporcionades per la càmera i permeti d'una manera senzilla construir interfícies per interactuar amb un entorn de Realitat Virtual. Paral·lelament a l'esmentada aplicació, es volen construir una sèrie d'interfícies per interactuar amb entorns de Realitat Virtual construïts pel grup d'investigació ViRVIG (Visualització, Realitat Virtual i Interacció Gràfica) de la Universitat Politècnica de Catalunya, fent ús dels recursos que es disposen al Centre de Realitat Virtual del departament de Llenguatges i Sistemes Informàtics de la UPC, com poden ser els dispositius per a entorns de Realitat Virtual PowerWall (veure secció 2.2.2) o CAVE (veure secció 2.2.3) dels que allà es disposen.

Capítol 2

Sistemes de Realitat Virtual

Cap a la dècada dels 90, la Realitat Virtual va ser un tema molt revolucionari. Les expectatives eren molt altes i va suposar una gran decepció. Vint anys després, poques persones coneixen el que és possible fer en el camp de la Realitat Virtual.

La tecnologia s'ha desenvolupat silenciosament en els laboratoris d'investigació i grans empreses i ara és quan han començat a aparèixer prototips en productes comercials. Aquest és un mercat viable i de ràpid creixement.

Les indústries utilitzen els sistemes de Realitat Virtual per construir els seus productes i per entrenar el seu personal. Els investigadors els utilitzen per entendre i tractar amb les persones. El mercat està començant a redescobrir que jugar amb el cos és una experiència atractiva[1].

> **"De la mateixa forma que una imatge és millor que mil paraules, un entorn virtual és millor que mil imatges!"**[1]

En els següents apartats s'introdueixen una serie de conceptes que poden ser útils per a la correcta comprensió del projecte així com l'àmbit en el que es desenvolupa.

2.1 Definició de sistema de Realitat Virtual

Una possible definició de Realitat Virtual és [2]:

> "La Realitat Virtual és un terme que descriu un entorn tridimensional generat per ordinador el qual pot ser explorat per una persona així com a interactuar en ell. La persona forma part d'aquest entorn i es troba immersa dins el mateix."

Un sistema de Realitat Virtual, com es pot veure a la figura 2.1, es compon per una serie de dispositius de sortida (pantalla/projector, àudio...), un conjunt de dispositius d'entrada (comandaments, trackers (veure secció 2.3)...) i un ordinador o conjunt d'ordinadors per controlar tots aquests dispositius i generar les respostes del sistema (visuals, auditives, tàctils...).

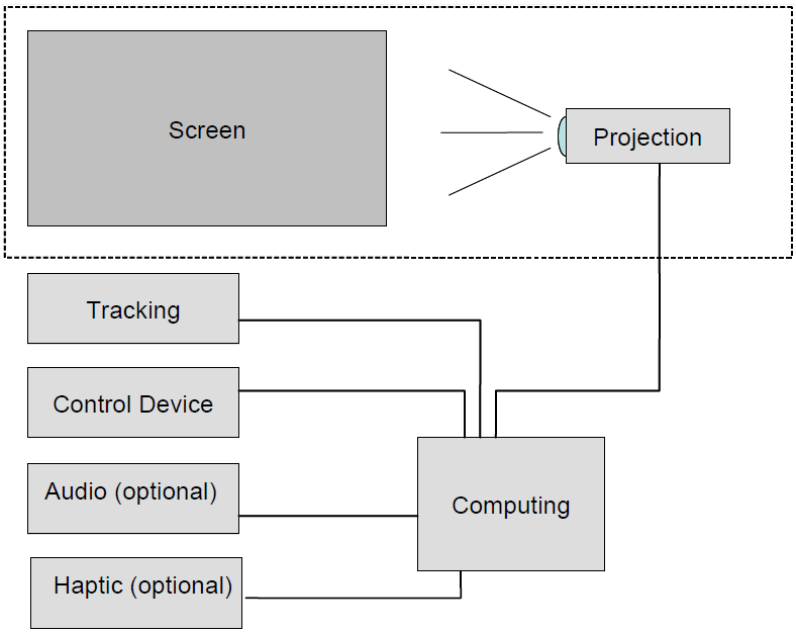


Figura 2.1: Esquema d'un sistema de Realitat Virtual basat en projecció[3].

2.2 Tipus de sistemes de Realitat Virtual

Existeixen tres grans tipus de sistemes de Realitat Virtual que s'identifiquen segons el grau d'immersió en l'entorn virtual que experimentem:

2.2.1 Sistemes d'escriptori

Els sistemes de Realitat Virtual d'escriptori són un tipus de sistema que intenta portar els beneficis de la Realitat Virtual a l'ús quotidià. Normalment disposen d'un simple monitor

d'escriptori per a la presentació de l'entorn virtual cosa que fa que la immersió sigui mínima a causa de la mida reduïda de la pantalla. En ocasions es tracta d'un monitor estereoscòpic, que juntament amb l'ús d'ulleres obturadores, permet la visualització d'imatges 3D[3].

Figura 2.2: Nvidia 3D Vision.

2.2.2 Sistemes semi-immersius (PowerWall)

El concepte darrere del terme "semi-immersiu" és un sistema de visualització fix que permet proveir d'un angle de visió major a 60 graus. La característica més important d'aquest sistema és el gran rendiment gràfic i potència dels ordinadors. S'utilitzen grans monitors o grans pantalles de projecció (corbades o planes).

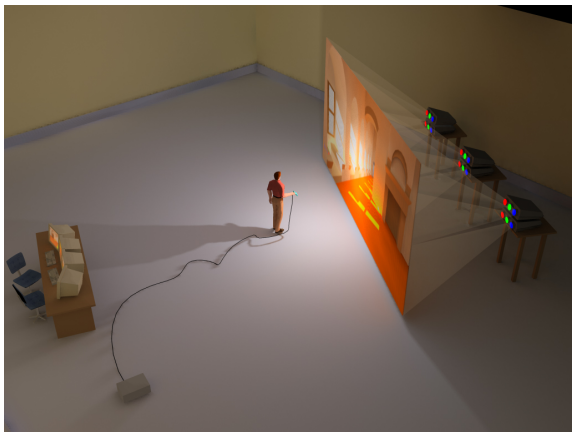


Figura 2.3: PowerWall. ©Arnaud Muthelet.

Un exemple d'aquest tipus és el sistema *PowerWall*. En aquest sistema s'utilitza una gran pantalla de projecció. Moltes vegades s'utilitzen múltiples projectors, cosa que permet obtenir una qualitat d'imatge superior[3].

2.2.3 Sistemes immersius (CAVE)

Figura 2.4: CAVE. ©Arnaud Muthelet.

Els sistemes completament immersius són aquells on l'usuari es troba totalment immers en l'entorn virtual. Aquesta sensació s'aconsegueix proveint a l'usuari amb una imatge visual a qualsevol lloc on estigui mirant[3].

Un exemple d'aquest tipus és el sistema *CAVE* (Cave Automatic Virtual Environment) que normalment és un cub (una espècie d'habitació) de com a mínim 3x3x3 metres i amb almenys quatre parets (incloent el terra) visualitzant imatges (normalment en estereoscòpic)[1].

2.3 Sistemes de seguiment

Un tracker o sistema de seguiment (de l'anglès "Tracking system"), és un sistema que proporciona la posició i orientació d'un objecte físic real de manera que es pugui fer servir en un entorn de Realitat Virtual. La posició de l'objecte es proporciona mitjançant coordenades Cartesianes x-y-z respecte a un sistema de referència donat. L'orientació es defineix mitjançant tres angles anomenats *pitch* (elevació), *roll* i *yaw* (azimut), o alguna representació equivalent (angles d'Euler, quaternió, matriu de rotació...)[3].

Existeixen diferents tipus de trackers segons la manera en la que obtenen les dades de posició i orientació (mecànics, magnètics, acústics, òptics...). A continuació es descriu el funcionament d'alguns d'ells:

2.3.1 Sistemes mecànics de posicionament

Els trackers mecànics mesuren la posició i orientació d'un objecte fixat a l'extrem d'un braç mecànic. El braç mecànic es troba unit a un punt de referència fix i està construït amb diverses seccions que es poden rotar i moure als seus punts d'unió.

Figura 2.5: Phantom OMNI com a tracker mecànic.

Aquests tipus de trackers proporcionen una gran precisió i resolució i són ràpids a l'hora de processar la informació. Aquestes propietats els fan adients per a fer seguiment de petits volums. Per contra, són pesats i necessiten ser calibrats[3].

2.3.2 Sistemes magnètics de posicionament

Els trackers magnètics són els més comuns. Aquests tipus de trackers funcionen mitjançant la mesura de la força dels camps magnètics generats per un transmissor sobre tres petites bobines de filferro, orientades perpendicularment. Aquestes tres petites bobines es troben incrustades en una petita unitat que estarà unida a l'objecte al qual es vol fer el seguiment, mentre que el transmissor es troba en posició estàtica. Mitjançant la mesura seqüencial del corrent induït pel camp magnètic sobre cadascuna de les petites bobines, és possible determinar la posició i l'orientació de l'objecte.

Figura 2.6: Transmissor magnètic Polhemus.

Aquests tipus de trackers no són excessivament cars, tenen una precisió i resolució alta i són suficientment immunes contra el soroll. El receptor sol tenir una mida petita,

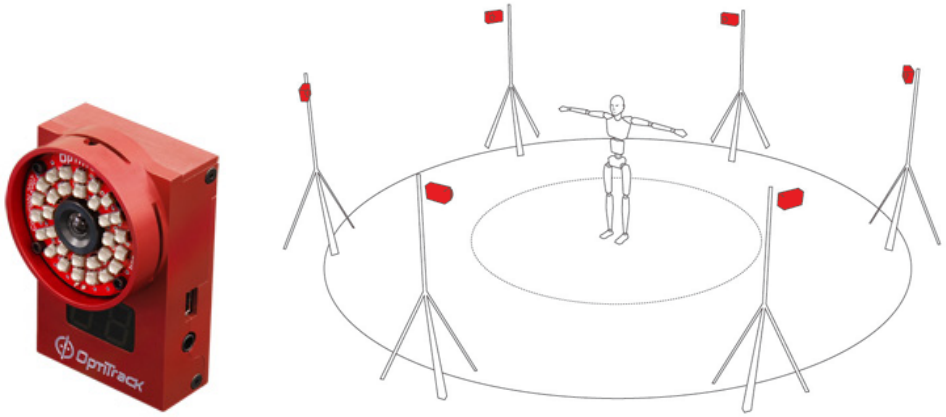
mentre que el transmissor sol ser gran. Un altre dels inconvenients és que el transmissor sol tenir cables, cosa que pot fer incòmode el seu ús[3].

Els trackers més famosos d'aquest tipus són Polhemus i Ascension.

Figura 2.7: Sistema de seguiment Ascension.

2.3.3 Sistemes òptics de posicionament

Actualment la nova tendència és la utilització de trackers òptics. Aquests tipus de trackers funcionen col·locant una sèrie de marques als objectes els quals es vol fer el seguiment. Aquestes marques són detectades des de diferents punts de vista on estan situats els detectors òptics. Mitjançant triangulació (es necessita que una marca sigui detectada per almenys dos detectors), es pot calcular la posició de l'objecte.



(a) Optitrack (b) Muntatge del sistema de seguiment

Figura 2.8: Sistema òptic de posicionament.

Aquests tipus de sistemes tenen com a avantatges que proporcionen una gran precisió i

resolució, funcionen amb un gran volum de treball i un temps de procés petit. A més, en aquests sistemes no hi ha problemes d'interferència magnètica. Per contra, tenen el problema provocat per la línia de visió (de l'anglès "line-of-sight"). Aquest fenomen provoca que aquests tipus de trackers perdin precisió a mesura que s'incrementa la distància (degut a la refracció, reflexió i absorció de les emissions de llum per part del medi de transmissió) i que sigui impossible fer el seguiment dels objectes en el cas que hi hagi obstacles ocloent-los. A més, estan limitats per la intensitat i la coherència de les fonts de llum i són més cars[3].

La càmera Kinect (dispositiu en el que es basa aquest projecte) podria considerar-se un tracker òptic. Aquesta càmera utilitza una càmera d'infrarojos per tal de fer el seguiment dels usuaris que es troben davant (veure secció 6.1).

2.3.4 Altres sistemes de seguiment

A part dels sistemes de seguiment descrits anteriorment, existeixen altres tipus de sistemes. Alguns d'ells poden ser els sistemes acústics o els sistemes d'inèrcia. En el cas dels trackers acústics, aquests funcionen mitjançant ones ultrasonores per tal d'estimar la posició de l'objecte. Els sistemes d'inèrcia obtenen l'orientació i posició de l'objecte amb l'ajuda de giroscopis i acceleròmetres.

Capítol 3

Motivacions i objectius del projecte

3.1 Motivacions del projecte

L'aparició de nous dispositius de comandament sense cables i sense cap altre element necessari a part del propi cos de l'usuari, com és el cas de la càmera Kinect, ha propiciat que molts laboratoris de recerca, i investigadors o curiosos independents, s'hagin interessat en l'ús d'aquests tipus de dispositius per a construir interfícies de comandament per tot tipus d'aplicacions informàtiques. Un altre factor importat que fa que aquests dispositius siguin atractius és el baix preu que tenen en el mercat, actualment per sota dels 100 €.

La motivació principal d'aquest projecte ha sigut doncs, la incorporació d'una interfície sense cables i intuïtiva en un sistema de Realitat Virtual, que sigui fàcil d'utilitzar i de mantenir. Tot i que el sistema Kinect no proporciona una alta precisió i resolució, proporciona certs avantatges davant altres tipus de dispositius de comandament o sistemes de seguiment:

- Té una mida reduïda que permet una instal·lació ràpida i senzilla.
- No necessita calibratge
- No necessita cables ni elements externs a part del propi cos de l'usuari, cosa que fa molt més còmode l'ús de la interfície
- No es veu afectat per interferències magnètiques
- La precisió no depèn de la intensitat o coherència de les fonts de llum ja que funciona mitjançant infrarojos
- Té un preu molt més assequible que altres sistemes de captura de moviment

La creació d'una interfície de comandament per a entorns de Realitat Virtual d'aquestes característiques representa, per tant, una motivació per a la realització d'aquest projecte.

3.2 Objectius del projecte

Com a objectiu principal del projecte tenim la construcció d'un programa que permeti capturar les dades sense tractar de diverses càmeres Kinect (una o múltiples càmeres alhora) i faci el tractament adequat a aquestes dades per fer-les accessibles a qualsevol tipus d'aplicació que vulgui fer ús d'aquesta interfície, de manera que per l'aplicació sigui transparent la manera en què es tracten les dades i la quantitat de Kinects que hi ha connectats. Un segon objectiu del projecte és demostrar el funcionament d'aquesta interfície, per tant, desenvolupar una sèrie de modes d'interacció per a una aplicació de Realitat Virtual.

Per tant, podem dividir els objectis del projecte en dos:

3.2.1 Servidor de manegament de càmeres Kinect

Com s'ha comentat, el primer i més important dels objectius és implementar una aplicació que rebi les dades de diverses càmeres Kinect, ja sigui una a una o múltiples alhora (això ens donarà versatilitat depenent de l'aplicació amb la que vulguem utilitzar la interfície). L'aplicació haurà de tractar aquestes dades pertinentment, així com permetre configurar diversos aspectes de les càmeres (inclinació de les càmeres, tipus d'informació que volem utilitzar de cadascuna de les càmeres, posició que ocupen les càmeres en el món virtual...). Un cop les dades hagin estat tractades, l'aplicació les ha de fer accessibles a altres aplicacions. El mètode més adequat en aquest cas és fer servir el protocol VRPN (veure secció 6.4) per fer accessibles les dades en rol de servidor.

Aquest primer objectiu, per tant, el podem subdividir en tres parts:

1. Recepció de les dades i configuració dels Kinect
2. Tractament i manipulació de les dades dels Kinect
3. Fer accessibles les dades dels Kinect mitjançant un servidor VRPN

3.2.2 Interfícies d'interacció per a una aplicació de Realitat Virtual

L'altre objectiu del projecte consisteix a demostrar el funcionament del Kinect en una aplicació de Realitat Virtual. Per aquest motiu, aquesta segona part consistirà en la implementació de tres plugins per a un visualitzador per a entorns de Realitat Virtual. Cadascun d'aquests plugins constituirà un mode d'interacció en el visualitzador, concretament, els modes d'interacció que es volen construir són: navegació per l'escena virtual, selecció d'objectes de l'escena i manipulació dels mateixos. A més dels plugins d'interacció, és adient tenir certa informació de retorn que ens indiqui el que estem fent i quins gestos s'estan detectant amb la càmera Kinect, per aquest motiu, també s'afegeix una petita finestra al visualitzador per tal d'obtenir aquesta informació.

Per tant, aquest segon objectiu es pot subdividir en quatre parts:

1. Creació del plugin de navegació
2. Creació del plugin de selecció
3. Creació del plugin de manipulació
4. Afegir una finestra de feedback visual

3.3 Descripció general de la solució proposada

El projecte presenta una aplicació que fa tot el manegament necessari per utilitzar una o múltiples càmeres Kinect des d'altres aplicacions externes. Més concretament, fa ús de tres tipus de dades proporcionades pel Kinect: imatge a color, mapa de profunditat i esquelet dels usuaris detectats. Aquestes dades es fan accessibles a aplicacions externes que les poden utilitzar de la manera que es vulgui. A més, l'aplicació permet modificar paràmetres de la configuració de cadascuna de les càmeres connectades. Alguns d'aquests paràmetres són:

- *Dades que es volen utilitzar (imatge, mapa de profunditat i/o esquelet)*: permet utilitzar estrictament el necessari i alliberar a l'aplicació de treball innecessari.
- *Inclinació de la càmera*: permet modificar la direcció de visió de la càmera, moltes vegades situada en llocs inaccessibles, accionant el motor de la base del Kinect.
- *Resolució de les dades*: permet modificar la resolució de la imatge i el mapa de profunditat, cosa que afecta directament a la precisió de la càmera.
- *Rotació i translació de la càmera*: permet transformar les dades del sistema de referència de la càmera a un altre sistema de referència.

Un dels punts forts de l'aplicació, és que dóna la possibilitat d'accedir a les dades d'esquelet de múltiples càmeres Kinect alhora (altres aplicacions similars no ho permeten. Veure secció 5.1).

Juntament amb l'aplicació de manegament, es presenta un conjunt de plugins d'interacció per a un visualitzador de Realitat Virtual (el visualitzador no s'ha desenvolupat dins de l'àmbit d'aquest projecte) que permetran navegar per una escena tridimensional, així com seleccionar i manipular objectes de la mateixa. Aquests plugins d'interacció venen acompanyats d'una finestra que proporcionarà a l'usuari informació sobre l'estat de la interacció, i la qual, és independent de l'aplicació que s'estigui fent funcionar en el visualitzador.

Capítol 4

Anàlisi dels reptes del projecte

Una vegada definits els objectius del projecte, s'han d'analitzar per tal de veure com assolir-los. Mitjançant aquesta anàlisi, trobarem una solució, la construcció de la qual, comportarà certs reptes. En aquest capítol es defineixen els reptes més rellevants als que ens enfrontem en aquest projecte, que són: l'obtenció de les dades de múltiples càmeres Kinect, el tractament d'aquestes dades, fer-les accessibles a altres aplicacions de forma transparent i, en quant a la creació de la interfície d'interacció, la detecció de gestos a partir d'un esquelet.

4.1 Obtenció de les dades de múltiples Kinect

La finalitat d'aquest projecte és poder construir un sistema d'interacció basat en Kinect per a entorns de Realitat Virtual. L'ús d'una càmera Kinect però, pot ser insuficient en aquestes situacions, ja que, en la majoria dels casos, es tracta d'entorns suficientment grans per a que l'usuari pugui desaparèixer del camp de visió de la càmera o, segons l'orientació que aquest adopti, pot interferir-se a si mateix i provocar una detecció errònia. Aquest motiu, doncs, afegeix la necessitat de poder utilitzar múltiples Kinect alhora.

Una càmera Kinect pot oferir tres tipus de dades: imatge de color, mapa de profunditat i/o esquelet dels usuaris detectats. Per poder obtenir aquestes dades, per tant, hem d'entendre el funcionament del controlador (de l'anglès "driver") de Kinect així com fer-lo servir correctament.

Donada la nostra necessitat d'utilitzar múltiples càmeres Kinect alhora, es fa servir el driver que Microsoft proporciona[1], que permet de forma més senzilla treballar amb múltiples

[1]Tot i que el dispositiu Kinect neix com un controlador de videojocs per a la videoconsola Microsoft Xbox 360, donat el gran interès que va despertar aquest dispositiu per fer-lo funcionar en ordinadors, i per tant, poder utilitzar-lo en altres aplicacions, Microsoft va decidir alliberar un driver i un entorn de treball oficials.

Kinect. Aquest driver però, té un problema: resulta que dins d'un mateix procés, únicament podem accedir a les dades de l'esquelet d'un dels Kinect connectats. La nostra solució, per tant, haurà d'utilitzar un procés per cadascun dels Kinect connectats per tal de poder obtenir totes les dades de tots ells.

4.2 Tractament de les dades de múltiples Kinect

Com hem vist a la secció anterior, per tal d'obtenir les dades de múltiples Kinect, necessitem un procés per cadascuna de les càmeres. Això dificulta el posterior tractament de les dades degut a què no podem accedir des d'un mateix procés a les dades de tots els Kinect. Aquest problema suposarà que la solució hagi d'afegir un procés "master" que s'encarregui de tractar i escollir correctament les dades necessàries de cadascun dels Kinect, així com implementar un mecanisme de comunicació entre processos per tal de centralitzar aquestes dades en el procés "master".

Una vegada el procés "master" tingui accés a totes les dades dels Kinect, aquest ha de tractar-les de manera que s'identifiquin totes les parts dels esquelets detectats, es detecti quines parts manquen de cadascun d'ells, mantingui un control dels usuaris detectats i modifiqui aquestes dades segons la posició i orientació relativa de cada càmera respecte a un sistema de referència concret. Les dades d'imatge i mapa de profunditat, donat que són simples imatges 2D, no necessiten un tractament rellevant.

Posteriorment al tractament de les dades de cada càmera, el procés ha de combinar i/o escollir quines d'aquestes dades són les més adients i/o fiables per tal d'oferir-les a altres aplicacions.

4.3 Donar accés transparent a les dades dels Kinect

La idea principal en quant a l'accés de les dades dels Kinect es refereix, és que qualsevol aplicació pugui utilitzar-les. Aquestes aplicacions no han de tenir en compte el nombre de càmeres que s'estan utilitzant (el nombre de càmeres utilitzades només ha d'afectar a la qualitat i fiabilitat de les dades obtingudes), per tant, la nostra solució ha de donar un accés a les dades de forma totalment transparent a la quantitat de càmeres Kinect connectades, així com a la manera d'obtenir aquestes dades. Això també ens permetrà modificar la nostra solució com vulguem, sense que la resta d'aplicacions es vegin afectades.

Com ja s'ha comentat en seccions anteriors, la nostra solució farà ús del protocol VRPN (veure secció 6.4) per tal de resoldre aquest problema en l'àmbit de les dades dels esquelets detectats. La nostra tasca per tant, serà entendre com funciona aquest protocol per poder-lo integrar correctament en la nostra implementació.

4.4 Detecció de gestos a partir d'un esquelet

Per tal de construir la interfície d'interacció amb l'entorn virtual, cal utilitzar els esquelets detectats pels Kinect. Per interactuar amb l'escena virtual, l'única manera possible existent mitjançant un esquelet, és la detecció de gestos i moviments. Amb objectiu de realitzar una bona detecció de gestos, cal estudiar quins gestos són els més adients per cada tipus d'aplicació. En el nostre cas, cal detectar gestos suficientment intuïtius per tal de realitzar navegació per l'escena virtual, així com gestos per poder seleccionar i manipular objectes de la mateixa.

Donada la diversitat d'usuaris que poden acabar fent ús del sistema, la detecció de gestos ha de ser elaborada pensant en característiques que no siguin dependents de la complexió i/o alçada dels usuaris.

Altre factor que cal afrontar és que, degut a la baixa resolució que tenen les càmeres Kinect, en moltes ocasions es pot produir cert soroll en les dades dels esquelets, el qual cal saber tractar per tal de minimitzar els errors en la detecció de gestos.

4.5 Facilitat d'enteniment i adaptació a la interfície

Un problema molt comú que apareix en moltes de les interfícies d'interacció, és la falta d'enteniment del que estem fent o podem fer amb aquesta interfície. Això provoca que l'adaptació a la interfície sigui dificultosa i poc intuïtiva. Hem de pensar, que moltes de les aplicacions que es realitzen en l'àmbit de la Realitat Virtual, són utilitzades per persones poc habituades a l'ús d'ordinadors, videojocs o comandaments per als mateixos. És important, doncs, que una interfície d'interacció tingui cert feedback que ens pugui orientar en les nostres accions sobre l'entorn virtual sobre el que estem interactuant.

La nostra solució ha de proposar un mecanisme suficientment simple per tal d'indicar als usuaris les accions que estan realitzant sobre l'escena virtual i els objectes de la mateixa, de manera que es pugui actuar en conseqüència en tot moment.

Capítol 5

Treballs i investigacions prèvies relacionades

El següent capítol presenta algunes de les aplicacions existents fins al moment d'inici d'aquest projecte, així com investigacions realitzades, relacionades amb alguns dels aspectes que aquest projecte cobreix. Algunes de les investigacions esmentades, tot i que tenen algun tipus de relació amb el projecte, es focalitzen en temes que, en aquest projecte, potser no es tracten amb massa profunditat.

5.1 FAAST (Flexible Action and Articulated Skeleton Toolkit)

Probablement FAAST sigui l'aplicació més semblant que puguem trobar a la solució proposada en aquest projecte. FAAST ha sigut creada per un grup de persones del grup de recerca *MxR* pertinent al *Institute for Creative Technologies* de la *University of Southern California*. FAAST és una aplicació que facilita la integració del comandament mitjançant tot el cos amb videojocs i aplicacions de Realitat Virtual.

FAAST utilitza una càmera Kinect amb el driver OpenNI (veure secció 6.2.1) o el de Microsoft (aquest últim s'ha afegit en l'última versió, la 1.0) per fer el seguiment dels usuaris a partir de l'esquelet obtingut. FAAST inclou un servidor VRPN propi per fer accessibles a través de la xarxa, fins a quatre esquelets d'usuaris diferents, permetent així que aplicacions externes utilitzin aquestes dades a través d'un client VRPN. A més, l'aplicació permet emular esdeveniments de teclat al detectar una serie de gestos corporals predefinits.

FAAST és una eina de lliure utilització i distribució, tant per fins comercials com a no comercials.

Per cadascun dels quatre esquelets que com a molt FAAST pot fer el seguiment, s'en-

vien 24 articulacions (cadascuna amb la seva posició i orientació en coordenades respecte a un sistema fixat a la càmera Kinect). Les 24 articulacions que s'utilitzen són les següents:

ID	**Articulació**	**ID**	**Articulació**
0	Cap	12	Colze dret
1	Coll	13	Canell dret
2	Tors	14	Mà dreta
3	Cintura	15	Punta de la mà dreta
4	Coll esquerre	16	Maluc esquerre
5	Espatlla esquerra	17	Genoll esquerre
6	Colze esquerre	18	Turmell esquerre
7	Canell esquerre	19	Peu esquerre
8	Mà esquerra	20	Maluc dret
9	Punta de la mà esquerra	21	Genoll dret
10	Coll dret	22	Turmell dret
11	Espatlla dreta	23	Peu dret

Taula 5.1: Articulacions utilitzades per FAAST[4].

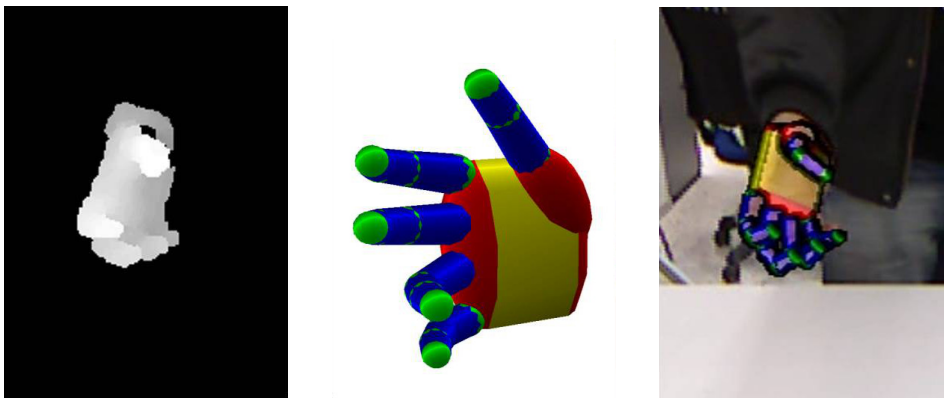
La raó principal per la qual no s'utilitza directament FAAST en aquest projecte és que no permet l'ús de múltiples càmeres Kinect. A més, existeixen altres propietats i característiques de la solució proposada en aquest projecte que FAAST no incorpora i que poden ser de gran utilitat. La següent taula mostra una petita comparació d'alguns aspectes entre FAAST i el manegador de la solució proposada:

	FAAST	**Solució proposada**
Imatge a color	✗	✓
Mapa de profunditat	✗	✓
Reconstrucció d'esquelets	✓	✓
Nombre màxim d'usuaris detectables	4	6
Suport multi-Kinect	✗	✓
Resolució configurable	✗	✓
Mode "assegut"	✓	✓
Mode "a prop"	✗	✓
Detecció de gestos predefinits	✓	✗
Utilització del motor d'inclinació	✗	✓

Taula 5.2: Comparació entre FAAST i la solució proposada.

5.2 Sistemes òptics de seguiment

Existeixen diversos treballs d'investigació que poden tenir relació amb aquest projecte en l'àmbit del posicionament òptic. Trobem un article d'un parell d'estudiants de doctorat supervisats pel professor *Antonis A. Argyros*[5] de la *University of Crete* (Grècia), en el qual, presenten una innovadora solució per fer el seguiment de les posicions i orientacions 3D de totes les articulacions d'una mà humana, utilitzant la càmera Kinect. A l'article tracten aquest problema com un problema d'optimització, en el qual, miren de trobar els paràmetres que minimitzen la discrepància entre l'estructura de diferents instàncies hipotètiques d'un model 3D d'una mà, amb les dades de profunditat obtingudes amb Kinect. La resolució d'aquest problema la fan mitjançant una variant del problema *Particle Swarm Optimization (PSO)*, un mètode d'optimització heurístic que intenta imitar el comportament d'un eixam d'abelles en la cerca de pol·len. La idea és que l'algorisme posseeix un espai de solucions en el qual es mou un eixam de partícules per tal de trobar la millor solució. Aquest eixam està guiat per les partícules que han trobat el millor valor a una funció objectiu segons la posició i velocitat actual de cada partícula. Un dels avantatges d'aquest mètode és que no necessita de cap marcador òptic per tal de fer el seguiment.



(a) Mapa de profunditat segmentat de la mà

(b) Model 3D de partida de la mà

(c) Resultat

Figura 5.1: Mètode de seguiment de la mà proposat.

Un altre treball de recerca que es va dur a terme a 2001 per *Ribo*[6], proposa un nou sistema de tracking òptic per aplicacions de realitat augmentada. La idea és que utilitzen un muntatge estereoscòpic (dues càmeres) equipat amb infrarojos que permet realitzar el seguiment d'objectes. Donen una solució amb la qual, el rendiment, la robustesa i la precisió del sistema s'assoleixen en temps real. El mètode que utilitzen es basa en la detecció de taques en imatges, predicció sobre un espai bidimensional i una reconstrucció tridimensional mitjançant les dades de les dues càmeres. El sistema proposat és capaç de fer el seguiment de fins a 25 objectes independents a una freqüència de 30 Hz.

Figura 5.2: Muntatge del tracker òptic estereoscòpic.

5.3 Estudis d'interacció en entorns virtuals

Com ja s'ha comentat, les interfícies d'interacció en entorns de Realitat Virtual han de ser prou intuïtives i fàcils d'utilitzar per a qualsevol tipus de persona. *Bowman*[7], en un article sobre navegació en entorns virtuals, presenta una classificació de tècniques de comandament en primera persona en entorns immersius de Realitat Virtual, així com un framework per poder avaluar la qualitat de diferents tasques específiques en entorns virtuals. A l'article fan la comparació de diferents tipus de tècniques de navegació directe cap a un objecte, diferents tècniques de navegació respecte un objecte de referència i diferents tècniques de navegació i la sensació de desorientació que provoca als usuaris. Aquests tipus de dades, són útils per al disseny d'interfícies d'interacció segons el tipus d'aplicació que vulguem desenvolupar.

5.4 Ajuts a la interacció en entorns virtuals

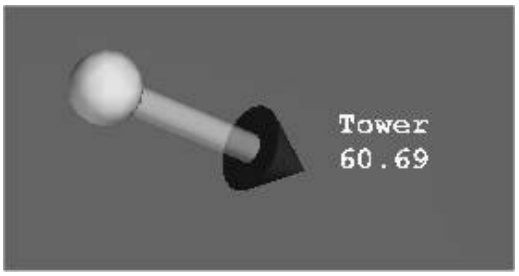


Figura 5.3: Fletxa 3D amb informació textual apuntant cap a un objecte.

En moltes ocasions, les interfícies per a entorns virtuals necessiten de cert suport o ajuda per a entendre que és el que s'està fent, i fer d'aquesta manera, que la interacció sigui més intuïtiva i senzilla. El cas és que moltes vegades aquestes ajudes són inadequades o complexes. *Chittaro* i *Burigat*[8] proposen una ajuda a la navegació per als usuaris en grans entorns virtuals. L'ajuda es basa en fletxes 3D que apunten en direcció als objectes d'interès per a l'usuari. També realitzen una serie d'avaluacions del mètode que proposen, contrastant-lo amb altres mètodes tradicionals en 2D per la mateixa tasca (fletxes 2D, ajudes tipus radar...), per tal de veure quin és el que dóna millors resultats. Les idees que aquest article dóna, serveixen per dissenyar interfícies d'interacció per a entorns virtuals més intuïtives gràcies a l'ús d'ajudes visuals 3D, les quals, ens indiquin les accions que estem duent a terme i/o les que podem dur. Aquest tipus d'ajudes també poden servir per a orientar a l'usuari en entorns de navegació lliure, indicant la direcció correcta on es troben els objectes d'interès.

Part II

Desenvolupament del projecte

Capítol 6

Eines i dispositius utilitzats

El següent capítol presenta les eines i dispositius més importants utilitzats per dur a terme el projecte. Es comenten les seves característiques més importants i rellevants, alguns dels motius de la seva elecció i algunes alternatives existents.

6.1 Microsoft Kinect

Figura 6.1: Microsoft Kinect.

6.1.1 Orígens

L'eina més important del projecte és sens dubte la càmera *Kinect*. Kinect és un dispositiu construït per l'empresa *PrimeSense*, una empresa Israeliana dedicada a la interacció natural (mitjançant gestos) amb dispositius electrònics. PrimeSense va desenvolupar un dispositiu low-cost, anomenat *PrimeSensor* (figura 6.2), per controlar televisors sense l'ús

de cap comandament a distància. Microsoft va voler portar aquest dispositiu al camp de la interacció amb videojocs en la seva videoconsola Microsoft Xbox 360.

Les propietats més rellevants de PrimeSensor són:

- Proveeix d'una interfície natural per a controlar alguns dispositius electrònics (per exemple TVs)
- Els usuaris no necessiten portar cap aparell o roba especial
- No depèn de la intensitat o coherència de les fonts de llum
- No requereix calibratge

Figura 6.2: PrimeSensor.

Una vegada Microsoft va tenir llest el dispositiu Kinect, el va posar a disposició dels consumidors a un preu molt assequible i atractiu (uns 99 €). Aquest fet va suposar que moltes persones s'interessessin pel producte, amb una finalitat però, que diferia de fer-la funcionar amb la videoconsola Xbox 360. És en aquest moment quan molts dels laboratoris d'investigació van començar a fer proves amb aquest tipus de dispositiu.

6.1.2 Hardware

Com es pot veure a la figura 6.3, Kinect disposa d'una càmera d'imatge a color, d'uns sensors de profunditat (distància a la càmera), un array de micròfons per captar só, un acceleròmetre per mesurar la inclinació de la càmera i un motor a la base que permet modificar aquesta inclinació de forma automàtica dins d'un cert interval (donat l'interès que ha provocat el dispositiu Kinect en l'ús del mateix amb ordinadors, Microsoft ha llençat una versió del hardware Kinect destinada únicament a aquest propòsit. Aquest hardware incorpora algunes millores, com poden ser, la capacitat de detectar objectes a una distància de 40cm de la càmera o una connexió USB-2 per no fer necessària l'ús de l'adaptador (secció 6.1.4)).

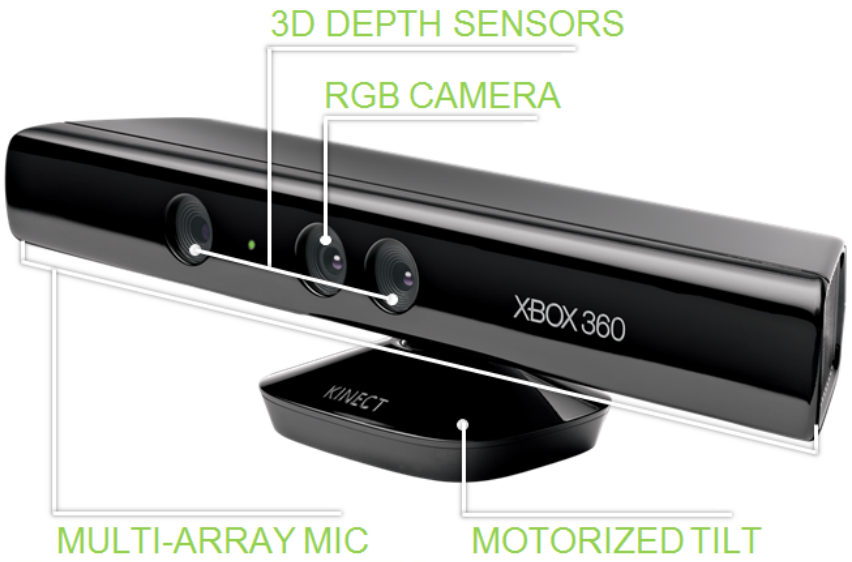


Figura 6.3: Microsoft Kinect Components.

A la taula 6.1 es poden veure algunes de les especificacions més rellevants del dispositiu.

6.1.3 Funcionament dels sensors de profunditat

La clau del funcionament de Kinect es troba en els seus sensors d'infrarojos. El funcionament dels mateixos és simple: per tal d'obtenir la profunditat dels objectes que es troben davant de la càmera Kinect, el projector d'infrarojos emet un patró irregular de rajos de diferents intensitats, repartint-los en el camp de visió de la càmera amb una distribució pseudo-aleatòria. La càmera d'infrarojos capta la distorsió del patró i, amb aquestes dades, reconstrueix el mapa de profunditat. A la figura 6.4 es pot apreciar el patró de rajos infrarojos emès per la càmera Kinect.

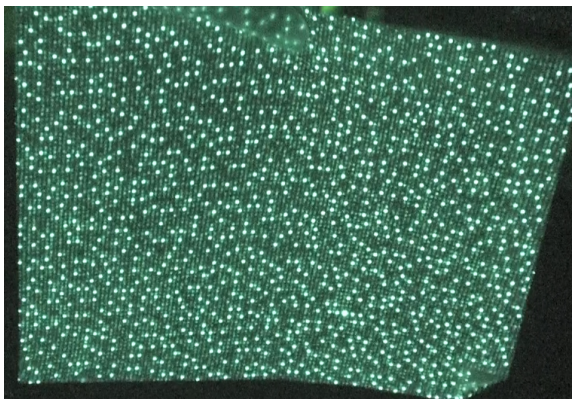


Figura 6.4: Patró d'infrarojos emès per Kinect.

Sensor de color	
Tipus de sensor	CMOS
Format	4:3
Resolució màxima	640x480
Frame rate (640x480)	30fps
Bits per píxel	32

Sensor de profunditat	
Tipus de sensor	CMOS + projector d'IR
Format	5:4
Resolució màxima	1280x1024
Frame rate (640x480)	30fps
Bits per píxel	16
Rang	1.2m - 3.5m
Resolució espacial (2m de distància)	3mm
Resolució de profunditat (2m de distància)	1cm

Field of View	
Horitzontal	57º
Vertical	43º

Sistema d'àudio	
Nombre de micròfons	4
Sistema de cancel·lació de soroll	✓

Control d'inclinació	
Composició	motor + acceleròmetre (3 eixos)
Inclinació física (motor)	±27º

Processador	
Chip	PrimeSense PS1080-A2
Memòria	64MB DDR2 SDRAM

Taula 6.1: Especificacions de la càmera Microsoft Kinect.

6.1.4 Connexió

La connexió de Kinect amb un ordinador no és directa. Kinect incorpora una connexió específica (figura 6.5a) que combina la connexió USB-2 i el suport de corrent. Aquesta connexió està pensada per realitzar la connexió amb les últimes versions de la videoconsola Microsoft Xbox 360. Tot i així, Microsoft proporciona un adaptador (figura 6.5b) amb el mateix Kinect que permet connectar el dispositiu mitjançant USB-2 a les antigues versions de la videoconsola i connectar la càmera al corrent de forma directe. Aquest adaptador també permet connectar la càmera Kinect a un ordinador.

(a) Connexió de Kinect (b) Adaptador proporcionat per Microsoft

Figura 6.5: Connexió i adaptador de la càmera Kinect.

6.2 Microsoft Kinect SDK

Per tal de construir aplicacions que utilitzin la càmera Kinect amb el driver de Microsoft, cal utilitzar l'entorn de desenvolupament que ells proporcionen. El SDK (Software Development Kit) de Microsoft Kinect està pensat per desenvolupar aplicacions en C++, C# o Visual Basic. A la taula 6.2 es poden veure algunes de les característiques més importants que proporciona el driver de Microsoft des de la versió 1.5.

Juntament amb el SDK, Microsoft proporciona un Developer Toolkit que pot ajudar als programadors a desenvolupar aplicacions que utilitzin Kinect més ràpidament. A més, presenta una sèrie d'exemples útils per entendre com fer funcionar el dispositiu.

Sistema de seguiment d'esquelets	
Esquelets màxims detectats simultàniament	6
Articulacions per esquelet	20
Dades per articulació	Posició (en metres respecte al sistema de referència de la càmera) + orientació (absoluta o relativa)
Mode assegut	Detecció d'usuaris i seguiment de les 10 articulacions de braços i cap
Mode "a prop"	Permet la detecció d'objectes a 40cm de la càmera (només en la versió de Kinect per ordinadors)

Sistema d'àudio	
Reconeixedor de veu	Suporta Anglès, Francès, Espanyol, Italià i Japonès (de diferents regions)

Altres característiques	
Reconeixedor de cares	Permet fixar una malla 3D a la cara dels usuaris i reconèixer gestos facials en temps real
Suport per a múltiples Kinect	Permet l'ús de múltiples Kinect connectades al mateix ordinador (en controladors USB diferents)

Taula 6.2: Característiques de Microsoft Kinect SDK.

6.2.1 Alternatives i elecció

Existeixen dues alternatives (ambdues de codi lliure) al driver de Microsoft per Kinect, amb les seves APIs (Application Programming Interface) o SDKs corresponents.
Vegem les seves característiques:

- ***LibFreenect***: LibFreenect és un driver creat per una comunitat d'usuaris sense ànim de lucre anomenada OpenKinect[9]. Aquest driver va ser el primer en aparèixer i va sorgir mitjançant enginyeria inversa. Aquest driver permet obtenir les dades de baix nivell de Kinect, és a dir, imatge a color i el mapa de profunditat únicament. LibFreenect també permet l'ús del motor d'inclinació de Kinect.

- ***OpenNI***: OpenNI[10] és un driver creat per una organització sense ànim de lucre amb el mateix nom. Aquest driver deriva del codi proporcionat per PrimeSense pel seu dispositiu PrimeSensor (secció 6.1.1). OpenNI, a part de proporcionar les dades de baix nivell de Kinect (imatge a color i mapa de profunditat), s'integra amb una plataforma intermèdia (anomenada NITE) que permet obtenir dades d'alt nivell, com són, els usuaris detectats i el seguiment dels seus esquelets. Aquest driver no permet l'ús del motor d'inclinació del dispositiu.

Ambdues alternatives poden funcionar en els sistemes operatius Windows, Linux o Mac OS X, però no són compatibles entre elles ni amb el driver de Microsoft.

Tot i que el suport multi-plataforma d'aquests drivers és un avantatge davant del driver de Microsoft, resulta que el driver de Microsoft té millors acabats que aquests dos. El driver de Microsoft incorpora tant les propietats d'OpenNI + NITE com les de LibFreenect. A més, la precisió i velocitat de detecció d'usuaris del driver de Microsoft és considerablement millor que en ambdues alternatives. El fet més important però, i que ha fet que ens decantéssim per aquesta opció, tot i que en OpenNI + NITE permeten l'ús de múltiples càmeres Kinect en un mateix ordinador, és que el manegament d'aquestes és considerablement més complicat que amb el driver de Microsoft. Recordem que una de les nostres necessitats en el projecte era poder connectar múltiples càmeres Kinects per millorar la interfície d'interacció.

6.3 Boost C++ library

Boost és un conjunt de llibreries lliures per a C++ que han estat revisades i testejades per experts. Aquestes llibreries proporcionen una gran quantitat d'implementacions de tot tipus: implementacions per treballar amb processos i threads, diferents tipus d'algorismes, diferents implementacions d'estructures de dades, funcions d'entrada i sortida... Totes les implementacions estan pensades per treballar amb les estructures de la STL (Standard Template Library) de C++. Boost és un gran recurs per a molts dels problemes que normalment apareixen a l'hora de construir qualsevol aplicació mitjanament gran. De fet, deu de les llibreries de Boost s'han inclòs al *C++ Standards Committee's Library Technical Report* (TR1) i en el nou estàndard de C++11 i, altres de les llibreries de Boost, estan proposades per ser incloses al TR2.

En el cas d'aquest projecte, s'ha utilitzat el mòdul de memòria compartida que proporciona Boost. La idea és la de poder compartir dades entre processos, ja que, com s'ha comentat anteriorment (secció 4.2), la nostra solució necessita la construcció d'un mecanisme de comunicació entre processos.

6.4 VRPN (Virtual Reality Peripheral Network)

VRPN és un conjunt de llibreries i servidors designats a implementar una interfície de xarxa transparent entre aplicacions i dispositius físics d'entrada (trackers, comandaments, haptics, dispositius analògics...). VRPN proporciona connexions entre dispositius i aplicacions utilitzant un tipus de model apropiat per a cada tipus de dispositiu. L'aplicació no és conscient, per tant, de la topologia de la xarxa ni del funcionament de cadascun dels diferents dispositius. De fet, VRPN proporciona una abstracció entre tots els dispositius de la mateixa classe, de forma que, l'aplicació només ha de tenir en compte les dades que rep i de quin tipus són, i no pas, de quin dispositiu les està produint i com es tracten o s'extreuen aquestes dades. Per exemple, tots els dispositius que siguin trackers proporcionaran el mateix tipus de dades (posicions i orientacions), independentment de com computin aquestes dades cadascun dels trackers. Cada tracker estarà governat per un servidor VRPN que enviarà les dades en el format adequat, i així, l'aplicació que vulgui fer-les servir només ha d'implementar un client VRPN per tal d'accedir a aquestes dades. Això fa que les aplicacions resultants siguin més portables i fàcils de mantenir i que, fàcilment, es puguin modificar el sistema o dispositius d'entrada sense que l'aplicació sigui conscient[11][12].

El protocol VRPN va ser implementat pel *Department of Computer Science* de la *University of North Carolina at Chapel Hill* a l'any 2001. Es tracta d'un protocol de domini públic, per tant, es pot utilitzar tant amb fins comercials com a no comercials.

L'elecció d'aquest tipus d'interfície per fer accessibles les dades de la càmera Kinect, és

degut a què es tracta d'un protocol extens i summament pràctic en el món de la Realitat Virtual. A més, permet que tant dispositius com aplicacions, s'executin en ordinadors diferents i entorns diferents, per exemple, podem tenir el servidor VRPN funcionant sobre Windows i el client funcionant sobre Linux. Un altre motiu d'aquesta elecció és que, el visualitzador destinat a funcionar amb Kinect utilitzat en aquest projecte, ja implementava el protocol VRPN.

6.5 wxWidgets

wxWidgets és una llibreria en C++ que permet crear interfícies gràfiques per Windows, Linux i Mac OS X, tant en versions de 32 com 64 bits. També permet crear interfícies gràfiques per a plataformes mòbils com pot ser Windows Mobile, iPhone SDK o GTK+. wxWidgets proveeix a les aplicacions d'un *"look and feel"*[1] natiu, ja que utilitza les APIs dependents de cada plataforma per crear la interfície gràfica. A més, wxWidgets proporciona altres recursos, a part dels relacionats amb interfície gràfica, que ajuden i faciliten el treball del programador. Alguns d'aquests recursos són: timers, estructures de dades contenidores, interfícies de xarxa (sockets per exemple) i funcionalitat amb OpenGL(secció 6.7)[13].

wxWidgets és totalment de codi lliure, per tant, es pot utilitzar tant amb fins comercials com a no comercials. S'ha construït per la comunitat i millora dia a dia gràcies a aquesta.

6.5.1 Alternatives i elecció

L'alternativa més important que podem trobar a wxWigdets, és possiblement, Qt[14]. Qt és un framework multi-plataforma per crear interfícies gràfiques. És també de codi lliure i es pot utilitzar tant amb fins comercials com no comercials. Com wxWidgets, proporciona recursos no gràfics (timers, contenidors...). A més, Qt proporciona un editor visual d'interfícies gràfiques que facilita considerablement el seu ús.

Tot i que ambdues plataformes (wxWidgets i Qt) són molt similars, i que Qt possiblement estigui en un estat més madur, s'ha escollit wxWidgets. L'elecció es basa en raons purament subjectives de l'autor i no té res a veure amb les característiques d'ambdues plataformes.

Altres alternatives que no s'han tingut en compte perquè no són tant completes com les anteriors, o perquè abasten camps poc útils per aquest projecte, són:

[1] "Look and feel" és un terme utilitzat en el disseny d'interfícies gràfiques per tal de referir-se al disseny, colors, forma, etc. de la mateixa.

- FLTK (`http://www.fltk.org/`): Té un disseny poc madur en l'orientació a objectes i no disposa de recursos addicionals (xarxa, timers...).

- FOX (`http://www.fox-toolkit.org/`): No disposa de recursos addicionals (xarxa, timers...) i no utilitza l'estil natiu dels widgets.

- SDL (`http://www.libsdl.org`): Només permet tenir una finestra en la interfície gràfica i proporciona altres recursos que són més adients en la creació de videojocs (per exemple, utilització i manegament d'arxius de so).

- SFML (`http://sfml.sourceforge.net/index.php`): Disposa de molts recursos que són més adients en la creació de videojocs.

- GTK+ (`http://www.gtk.org`): Tot i que s'ha portat a Windows, és més adient per a sistemes Unix.

6.6 VRJuggler

VR Juggler és una plataforma per desenvolupar aplicacions de Realitat Virtual. Aquesta plataforma permet als usuaris crear aplicacions per a quasi qualsevol sistema de Realitat Virtual. VR Juggler actua com una capa d'abstracció entre els components del sistema de Realitat Virtual (dispositius i/o interfícies d'entrada i sortida) i l'aplicació. Això permet que la mateixa aplicació es pugui fer funcionar en diferents entorns virtuals simplement modificant certs fitxers de configuració, sense tenir la necessitat de modificar l'aplicació o recompilar la mateixa. VR Juggler, per tant, permet utilitzar la mateixa aplicació en ordinadors personals, sistemes multi-pantalla, sistemes PowerWall (secció 2.2.2), sistemes CAVE (secció 2.2.3)...

VR Juggler es va començar a desenvolupar l'any 1997 per la Doctora *Carolina Cruz-Neira* i un equip d'estudiants al *Iowa State University's Virtual Reality Applications Center*. Es tracta d'una plataforma de codi lliure i es troba sota llicència GNU LGPL, per tant, pot ser utilitzat lliurement i de forma gratuïta per propòsits comercials i no comercials[15].

Aquest projecte fa ús directe dels fitxers de configuració de VR Juggler per tal d'integrar la interacció amb Kinect amb el visualitzador per al qual s'han desenvolupat els plugins de navegació, selecció i manipulació, ja que aquest, està construït amb aquesta plataforma.

6.7 OpenGL

OpenGL (Open Graphics Library) és una especificació estàndard que defineix un API multi-llenguatge i multi-plataforma per a escriure aplicacions que produeixen gràfics 3D i que va ser desenvolupada originalment per *Silicon Graphics Incorporated* (SGI). Entre les seves característiques podem destacar que la gestió de la generació de gràfics 2D i 3D per ordinador ofereix al programador una API senzilla, estable i compacta. De fet, la interfície consisteix en més de 250 funcions diferents que poden utilitzar-se per dibuixar escenes tridimensionals complexes a partir de primitives geomètriques simples, com poden ser, punts, línies o triangles. A més, la seva escalabilitat ha permès que no s'hagi estancat el seu desenvolupament i també la creació d'extensions per tal d'aprofitar les creixents evolucions tecnològiques. El seu ús està molt estès en els camps de CAD, Realitat Virtual, representació científica, visualització d'informació i simulació de vol. També s'utilitza pel desenvolupament de videojocs, encara que, sota el sistema operatiu Windows, té un gran competidor: *Direct3D*.

El fet de què OpenGL sigui multi-plataforma i multi-llenguatge, fa que aquesta alternativa sigui més adient per a entorns acadèmics i d'investigació que no pas Direct3D.

OpenGL és utilitzat pel manegador de càmeres Kinect per visualitzar i donar una idea global de la disposició dels esquelets, proporcionats per cada Kinect connectat, en una representació virtual de l'habitació on es troben instal·lades. També s'utilitza en el visualitzador per a entorns de Realitat Virtual per al qual s'ha desenvolupat la interfície d'interacció amb Kinect. S'utilitza per realitzar la visualització de les escenes 3D. Així mateix, els plugins d'interacció desenvolupats en aquest projecte, també utilitzen OpenGL per a implementar una finestra de feedback que proporciona informació sobre la interacció a l'usuari.

6.8 MGSoft

El visualitzador per a entorns de Realitat Virtual per al qual s'ha desenvolupat la interfície d'interacció amb Kinect, està basat en les classes proporcionades per la llibreria *MGSoft*. MGSoft (Moving Group Software) és una llibreria desenvolupada pel grup de recerca *MOVING* (ara *ViRVIG*[16]) de la *Universitat Politècnica de Barcelona*, grup que ofereix els seus recursos en el desenvolupament d'aquest projecte. Aquesta llibreria és suficientment extensa i proporciona classes i mètodes matemàtics i geomètrics, procediments d'entrada i sortida, gestió d'escenes i objectes 3D, funcionalitats de visualització 2D i 3D, entre d'altres.

Capítol 7

Arquitectura del projecte

El següent capítol mostra l'arquitectura i estructura del projecte. Primerament, es descriu l'arquitectura general del sistema desenvolupat, és a dir, la relació entre el visualitzador amb els plugins d'interacció i el manegador de càmeres Kinect, així com amb les eines utilitzades (descrites al capítol 6). Posteriorment, es descriu l'arquitectura de la solució proposada pel manegador de càmeres Kinect, així com una descripció de cadascuna de les classes implementades. Finalment, es presenta l'estructura i arquitectura que segueixen els plugins d'interacció del visualitzador que han sigut desenvolupats.

7.1 Arquitectura general del sistema

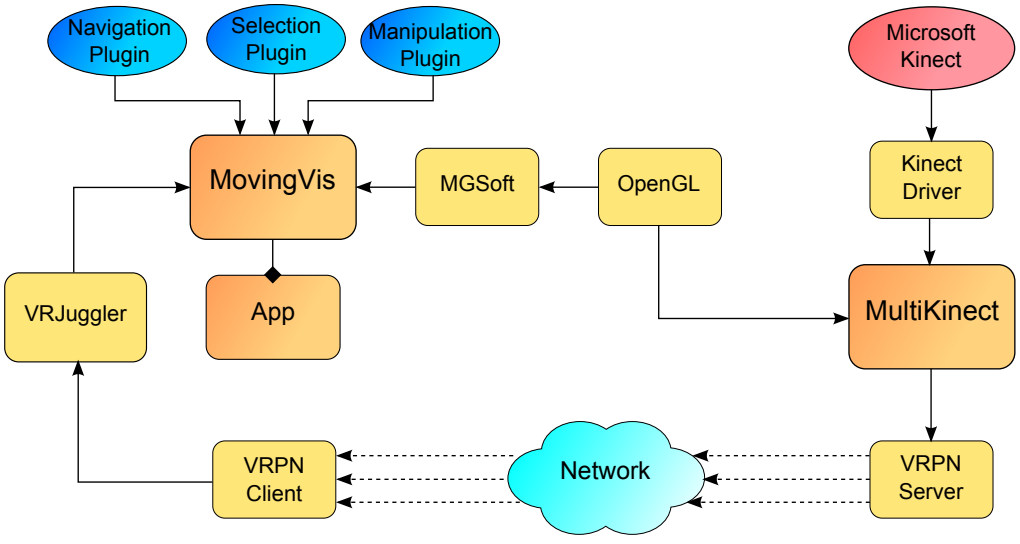


Figura 7.1: Diagrama de l'arquitectura general del sistema.

La figura 7.1 mostra de forma visual l'estructura general del sistema desenvolupat. A la part de la dreta podem veure el subsistema format pel manegador de càmeres Kinect, anomenat *MultiKinect*. A la part de l'esquerra es pot observar el subsistema format pel visualitzador, anomenat *MovingVis*, i els plugins d'interacció.

Primerament, les càmeres Kinect connectades generen les dades de baix nivell (imatge a color i mapa de profunditat) mitjançant els sensors de color i d'infrarojos dels quals disposen (veure secció 6.1). El driver de Kinect és l'encarregat d'agafar aquestes dades de baix nivell i calcular les dades d'alt nivell. D'aquesta manera, el driver de Kinect estableix la quantitat d'usuaris que es troben davant de la càmera, els assigna un identificador i calcula l'esquelet de cadascun d'ells (posició i orientació de les 20 articulacions). Aquestes dades es recullen per l'aplicació MultiKinect. MultiKinect disposa d'un procés per a cadascuna de les càmeres connectades. Cadascun d'aquests processos recull les dades d'alt nivell del driver de Kinect i, mitjançant una interfície de memòria compartida, les fa accessibles a un procés "master" que fa el tractament i selecció de les dades per tal de fer-les accessibles a aplicacions externes. Aquest procés "master", implementa un servidor VRPN (veure secció 6.4) que utilitza per fer accessibles les dades mitjançant la xarxa. La idea principal és que aquest servidor crea una connexió per a cadascun dels esquelets detectats amb Kinect i, per cadascuna d'aquestes connexions, envia la posició i orientació de cadascuna de les articulacions de cada esquelet. També realitza, mitjançant OpenGL (veure secció 6.7), la visualització d'una representació virtual d'una habitació on es pot veure la disposició de les càmeres Kinect i els esquelets detectats dins aquesta habitació.

L'aplicació de Realitat Virtual està construïda sota el visualitzador MovingVis. VR Juggler disposa d'un mòdul que implementa un client VRPN. Mitjançant el client VRPN, VR Juggler pot obtenir les dades dels esquelets proporcionats per MultiKinect. Seguidament, VR Juggler proporciona aquestes dades a MovingVis. Els plugins d'interacció es connecten amb MovingVis i, d'aquesta manera, poden accedir a les dades dels esquelets que MovingVis disposa. Cadascun dels plugins realitza el seguiment dels esquelets i la detecció de gestos, els quals, permeten interactuar amb l'escena. Els plugins realitzen els canvis necessaris sobre l'escena (moviment de la càmera, selecció i modificació d'objectes...), la qual finalment es renderitzada mitjançant OpenGL, accessible per MovingVis a través de la llibreria MGSoft.

7.2 Arquitectura del manegador MultiKinect

Donada la quantitat de classes de les que es compon MultiKinect, s'ha dividit en tres parts el diagrama que mostra l'estructura principal de l'aplicació i la relació de les classes amb les llibreries i eines utilitzades. En els següents diagrames es poden veure quatre tipus de classes:

- *Blocs de llibreries/eines*: Amb color taronja
- *Classes principals d'una única instància (singletons)*: Amb color blau
- *Classes principals de múltiples instàncies*: Amb color groc
- *Estructures de dades útils*: Amb color verd

Posteriorment es realitza una breu descripció de cadascuna de les classes que conformen MultiKinect per tal de fer-se una idea del funcionament de cadascuna.

El diagrama de la figura 7.2 mostra les relacions de les classes de major importància de l'aplicació.

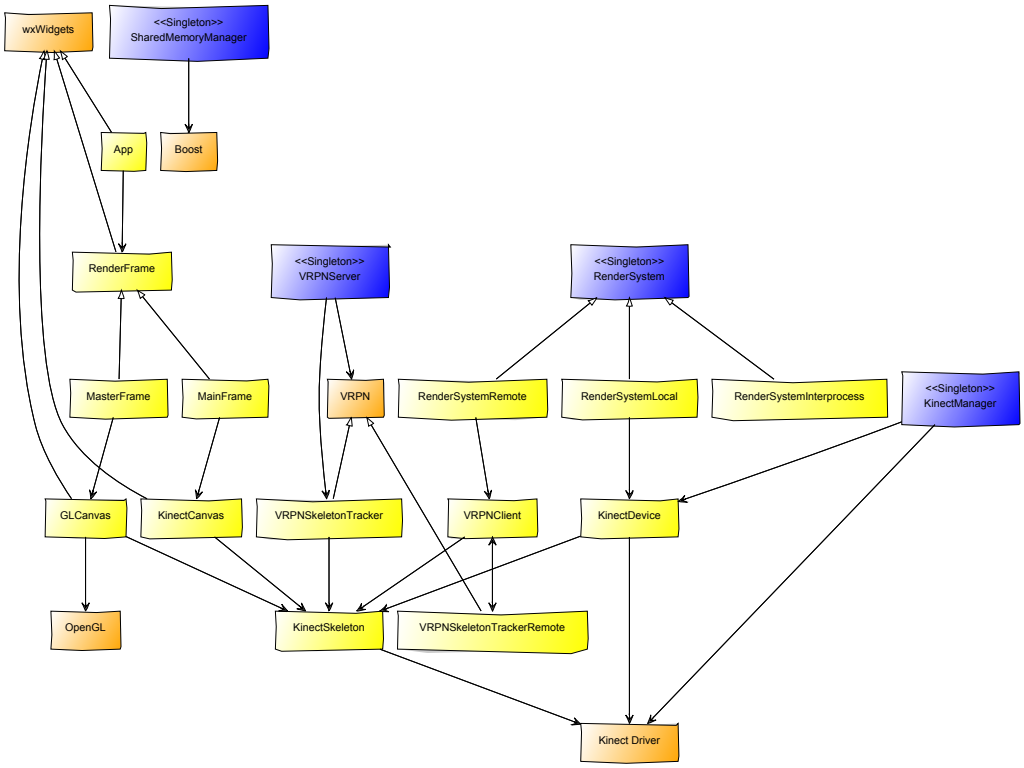


Figura 7.2: Diagrama de l'arquitectura principal de MultiKinect.

A la figura 7.3 es presenta el diagrama de les classes de suport de l'aplicació. El propòsit d'aquestes classes és ajudar tant en l'etapa de depuració del projecte com en proporcionar a l'usuari informació i una plataforma de configuració de l'aplicació.

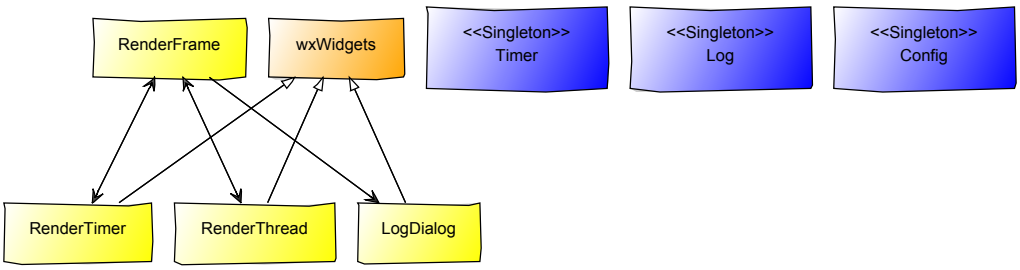


Figura 7.3: Diagrama de les classes de suport de MultiKinect.

Al diagrama de la figura 7.4 es poden veure les classes menys rellevants de l'aplicació. Aquestes classes implementen algunes estructures de dades d'alguns elements matemàtics i geomètrics necessaris per la representació i tractament de les dades proporcionades per Kinect i per la visualització amb OpenGL (veure secció 6.7).

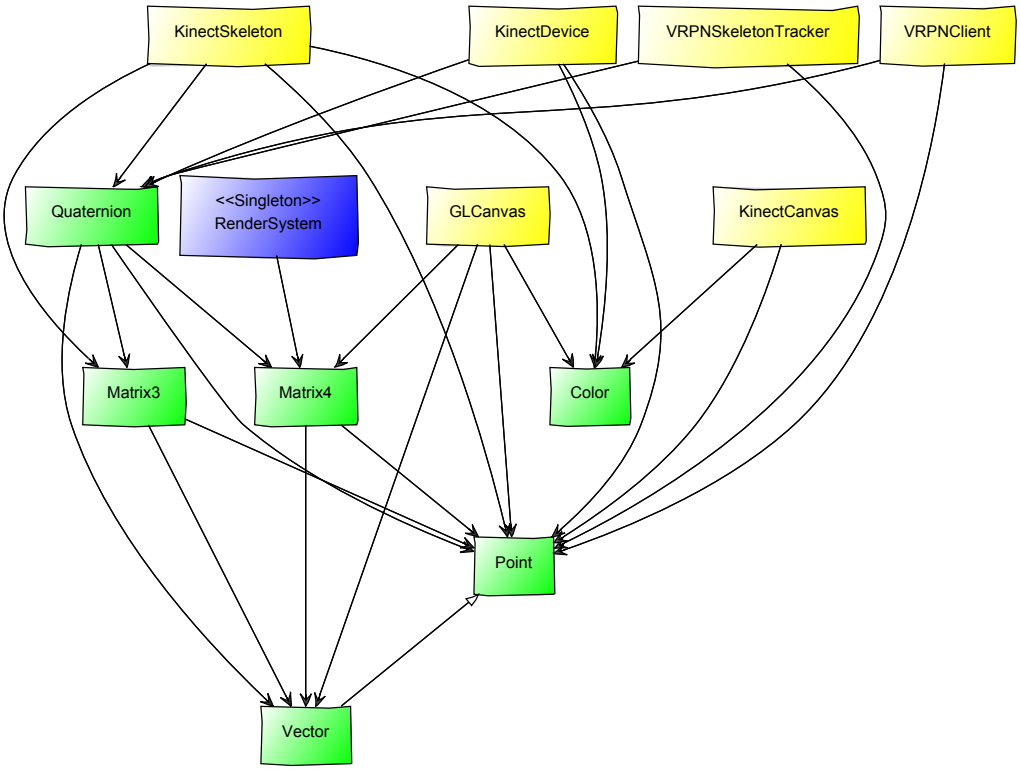


Figura 7.4: Diagrama de les estructures simples de MultiKinect.

7.2.1 Classes principals

Classe App

App és la classe que proporciona el punt d'entrada a l'aplicació. És una classe de deriva de *wxApp*, pertinent a la llibreria de wxWidgets (secció 6.5). Aquesta classe realitza les tasques d'inicialització i finalització de l'aplicació. Les tasques d'inicialització són les següents:

1. Establir el mode d'execució i obtenir l'adreça del servidor VRPN (veure secció 6.4) segons els arguments de l'aplicació
2. Inicialitzar totes les classes singleton de l'aplicació
3. Crear el marc de la interfície gràfica segons si es tracta del procés "master" o no

I les tasques de finalització les següents:

1. Finalitzar totes les classes singleton de l'aplicació
2. En cas del procés "master", enviar un missatge de finalització a la resta de processos
3. Reiniciar l'aplicació si l'usuari ha donat aquesta instrucció

Classe GLCanvas

GLCanvas és una classe que implementa una finestra que utilitza OpenGL (veure secció 6.7) per tal de visualitzar una representació virtual d'una habitació, així com la disposició, dins de l'habitació, tant de les càmeres Kinect connectades com dels esquelets detectats. Aquesta classe és utilitzada pel procés "master" de l'aplicació i deriva de la classe *wxGLCanvas* pertinent a la llibreria de wxWidgets (secció 6.5).

A part de visualitzar dades en 3D, GLCanvas permet certa interacció amb el que s'està visualitzant. Concretament, permet moure la càmera de l'entorn 3D per tal de poder veure l'habitació virtual des de diferents punts de vista, o fer zoom per tal d'apreciar amb més detall algun dels elements dibuixats.

Classe KinectCanvas

La classe *KinectCanvas* implementa una finestra on es mostren les dades de Kinect de forma visual. Deriva de la classe *wxWindow*, pertinent a la llibreria de wxWidgets (secció 6.5). Existeixen tres tipus de classe KinectCanvas, segons la informació que mostra. Aquests tipus són:

- COLOR_CANVAS: Mostra la imatge a color proporcionada pel sensor de color de Kinect

- DEPTH_CANVAS: Mostra el mapa de profunditat imatge proporcionat pels sensors d'infrarojos de Kinect
- SKELETON_CANVAS: Mostra els esquelets (en format de línies i punts) dels usuaris detectats per la càmera Kinect

Les dades mostrades per la classe KinectCanvas s'obtenen a través de la classe RenderSystem.

Classe KinectDevice

La classe *KinectDevice* és l'encarregada de realitzar tot el manegament de l'API de Microsoft proporcionada per treballar amb el driver de Kinect. Aquesta classe permet inicialitzar, configurar i finalitzar un dispositiu Kinect connectat a l'ordinador. També permet obtenir dades relacionades amb cada dispositiu. Algunes de les tasques que podem realitzar sobre un dispositiu Kinect mitjançant aquesta classe són:

- Definir de forma independent si volem que el dispositiu obtingui la imatge a color, mapa de profunditat i/o faci el seguiment dels esquelets
- Obtenir l'identificador únic i l'estat del dispositiu Kinect
- Configurar els modes "assegut" i "a prop" que proporciona el driver de Kinect
- Obtenir la imatge a color i/o mapa de profunditat proporcionats per Kinect
- Obtenir el nombre d'esquelets detectats i/o les dades dels mateixos
- Consultar i/o establir l'angle d'inclinació de la càmera Kinect
- Consultar i/o establir la matriu homogènia de transformació que defineix la posició i orientació de la càmera Kinect dins d'un sistema de referència donat

Aquesta classe té un fil d'execució independent. Aquest fil d'execució es troba constantment esperant a què es produeixi algun dels següents esdeveniments definits:

- *Processar color*: Produït pel driver de Kinect, indica que hi ha disponible nova informació del sensor de color
- *Processar profunditat*: Produït pel driver de Kinect, indica que hi ha disponible nova informació dels sensors de profunditat
- *Processar esquelet*: Produït pel driver de Kinect, indica que hi ha disponible nova informació sobre el seguiment dels esquelets detectats
- *Finalitzar*: Produït al tancar l'aplicació, indica al fil d'execució que ha d'acabar

Classe KinectManager

KinectManager és la classe que realitza el manegament dels dispositius Kinect connectats a l'ordinador. És l'encarregada de detectar la quantitat de càmeres Kinect connectades a l'ordinador, obtenir els seus identificadors i proporcionar, a les classes que ho necessitin, l'estructura KinectDevice corresponent per poder treballar amb el dispositiu Kinect desitjat. Es tracta d'una classe singleton, per tant, únicament existeix una instància d'aquesta.

Classe KinectSkeleton

La classe *KinectSkeleton* implementa la representació de l'esquelet proporcionat per Kinect. Permet modificar i obtenir les dades de posició i orientació de cadascuna de les articulacions de les que es forma l'esquelet. També permet obtenir l'identificador d'usuari al qual correspon l'esquelet en qüestió i obtenir les dades de conjunts d'articulacions que formen les diferents parts de l'esquelet (per exemple, el braç dret, la columna vertebral o la cama esquerra).

Aquesta classe disposa d'un mètode que permet obtenir la projecció en 2D de cadascuna de les articulacions de l'esquelet i, un altre mètode, per convertir l'identificador d'usuari a un color en format RGB. Això ens permet facilitar la tasca de mostrar els esquelets dels usuaris a les classes GLCanvas i KinectCanvas.

Classe MainFrame

La classe *MainFrame* defineix el marc de la interfície gràfica de l'aplicació en el procés que controla un dispositiu Kinect. Aquesta classe disposa d'una barra de menú que permet interactuar amb la interfície. També disposa de tres finestres de tipus KinectCanvas (una de cada tipus) per poder visualitzar les dades proporcionades per la càmera Kinect i d'una secció amb diverses opcions que permeten configurar el dispositiu Kinect i guardar aquesta configuració per a pròximes execucions.

MainFrame deriva de la classe RenderFrame, que a la seva vegada, deriva de la classe *wxFrame* pertinent a la llibreria de wxWidgets (secció 6.5).

Classe MasterFrame

La classe *MasterFrame* defineix el marc de la interfície gràfica de l'aplicació en el procés "master". Aquesta classe disposa d'una barra de menú que permet interactuar amb la interfície. També disposa d'una finestra de tipus GLCanvas per poder visualitzar en 3D les dades dels esquelets detectats per les càmeres Kinect connectades, així com una representació de les mateixes, en una habitació virtual, les mides de la qual, es poden configurar. Les dades dels esquelets i de les càmeres Kinect són proporcionades al procés "master" per la resta de processos. També disposa d'una secció amb diverses opcions que permeten configurar, a part de les mides de l'habitació virtual, altres aspectes com poden ser la resolució amb la que volem visualitzar la finestra GLCanvas.

MasterFrame, de la mateixa forma que MainFrame, deriva de la classe RenderFrame.

Classe RenderFrame

Donat que l'aplicació pot tenir dos interfícies gràfiques diferents segons el tipus de procés ("master" o no), per tal d'aprofitar el polimorfisme que ofereix C++, i simplificar algunes de les relacions de MainFrame i MasterFrame amb la resta de classes, apareix *RenderFrame*, una classe abstracta que deriva de la classe *wxFrame* pertinent a la llibreria de wxWidgets (secció 6.5).

Classe RenderSystem

RenderSystem és una classe singleton (una única instància) que realitza el manegament de les dades proporcionades per Kinect que es poden visualitzar. Segons el tipus de procés de l'aplicació, les dades de Kinect poden provenir de fonts diferents, és a dir, en el cas dels processos que controlen dispositius Kinect, aquestes dades provenen de la classe KinectDevice però, en el procés "master", aquestes dades provenen de la resta de processos, i per tant, de la interfície de memòria compartida. Per aquest motiu, RenderSystem implementa una classe abstracta per tal de fer ús del polimorfisme de C++, en la qual, s'implementen certs procediments generals per simplificar el funcionament de les subclasses específiques, que són les encarregades de definir els mètodes per obtenir les dades de les diferents fonts disponibles. Això permet que les classes GLCanvas i KinectCanvas, que fan ús de les dades de Kinect, no hagin de tenir en compte d'on provenen aquestes dades, i per tant, el tractament de les dades sigui totalment transparent a la font d'aquestes.

Classe RenderSystemInterprocess

RenderSystemInterprocess és una de les subclasses específiques de RenderSystem. Aquesta implementació és l'encarregada d'obtenir les dades proporcionades per Kinect a través del sistema de memòria compartida (implementat a la classe SharedMemoryManager). Aquesta classe, per tant, és la que fa possible accedir des del procés "master" a les dades dels diferents Kinect connectats.

Donat que RenderSystemInterprocess és capaç d'accedir a les dades de totes les càmeres Kinect connectades, altra tasca que ha de realitzar és la de seleccionar i combinar aquestes dades.

Classe RenderSystemLocal

La classe *RenderSystemLocal* és una altra de les subclasses de RenderSystem. Aquesta classe implementa els mètodes encarregats de proporcionar les dades de Kinect directament del dispositiu KinectDevice. RenderSystemLocal, és per tant, la classe utilitzada en cadascun dels processos que controlen les càmeres Kinect connectades.

Classe RenderSystemRemote

RenderSystemRemote és l'última de les subclasses de RenderSystem. Per tal d'efectuar tasques de depuració durant la realització del projecte, el procés "master" de l'aplicació disposa d'un altre mode de funcionament, el qual, obté les dades de Kinect a través d'un client VRPN (implementat a la classe VRPNClient). RenderSystemRemote, per tant, accedeix a les dades de Kinect a través de la xarxa mitjançant VRPN (veure secció 6.4).

Classe SharedMemoryManager

La classe *SharedMemoryManager* és una classe que proporciona mètodes senzills per tal d'utilitzar memòria compartida entre processos. Es tracta d'una classe singleton que utilitza la classe *managed_shared_memory* de la llibreria Boost (secció 6.3). Aquesta classe de Boost permet crear segments de memòria mapejats en disc i crear objectes de C++ en aquests segments. Això permet utilitzar aquests objectes des de diversos processos de la mateixa forma que si estiguessin en la memòria de procés.

SharedMemoryManager simplifica l'ús de les classes de Boost, oferint mètodes per les següents tasques:

- Crear i destruir segments de memòria compartida
- Crear i destruir objectes (de qualsevol tipus) en memòria compartida
- Obtenir un objecte que es troba en memòria compartida

La idea principal és que un procés crea el segment de memòria compartida, assignant-hi un identificador, i els objectes dins d'aquest segment (també amb un identificador). Des de la resta de processos es pot obtenir un apuntador a aquests objectes a partir de l'identificador de segment i l'identificador de l'objecte. A partir d'aquí, ambdós processos poden llegir i escriure en aquest objecte, el qual, és el mateix per a tots dos.

Classe VRPNClient

La classe *VRPNClient* implementa un client de VRPN (veure secció 6.4) que obté les dades dels esquelets detectats per l'aplicació a la banda del servidor VRPN. Per obtenir les dades dels esquelets, fa ús de diverses instàncies de VRPNSkeletonTrackerRemote (una per cada esquelet que permet detectar Kinect). Aquesta classe proporciona mètodes per obtenir la quantitat d'esquelets detectats/rebuts i la representació d'aquests.

VRPNClient té un fil d'execució independent que actualitza constantment cadascun dels trackers remots per tal d'observar si s'han rebut noves dades d'esquelets. Al tancar l'aplicació, s'envia un esdeveniment de finalització que fa que el fil acabi.

Classe VRPNServer

VRPNServer és una classe singleton que implementa un servidor VRPN (veure secció 6.4). VRPNServer utilitza la classe *vrpn_Connection_IP* de la llibreria VRPN i diverses instàncies de VRPNSkeletonTracker (una per cada esquelet que permet detectar Kinect) per tal d'enviar, a través d'una connexió IP, les dades dels esquelets detectats per Kinect.

Aquesta classe disposa d'un fil d'execució independent que actualitza constantment cadascun dels trackers locals per tal d'obtenir i preparar les dades dels esquelets detectats. Seguidament actualitza la connexió IP per realitzar l'enviament de les dades a través de la xarxa. Al tancar l'aplicació, s'envia un esdeveniment de finalització que fa que el fil acabi.

Classe VRPNSkeletonTracker

La finalitat de la classe *VRPNSkeletonTracker* és preparar les dades dels esquelets detectats per tal d'enviar-les amb el servidor VRPN (veure secció 6.4). Aquesta classe deriva de *vrpn_Tracker* de la llibreria VRPN. Disposa d'un únic mètode d'actualització, el qual, obté les dades d'un dels esquelets que Kinect és capaç de detectar a través de RenderSystem i, en el cas que siguin vàlides, les prepara pel posterior enviament a través d'una connexió IP. L'esquelet assignat a VRPNSkeletonTracker es defineix mitjançant el mètode de creació de la classe, conjuntament amb l'adreça de xarxa (única) que tindrà el tracker.

Classe VRPNSkeletonTrackerRemote

La classe *VRPNSkeletonTrackerRemote* permet obtenir les dades d'un dels esquelets detectats per Kinect a la banda del servidor VRPN (veure secció 6.4). Aquesta classe deriva de *vrpn_Tracker_Remote* de la llibreria VRPN. Implementa una funció callback que es crida cada cop que es rep nova informació a través d'una connexió IP. Aquesta funció crida a la classe VRPNClient per enviar les dades rebudes. L'adreça de xarxa a la qual es connecta el tracker remot s'especifica en el mètode de creació de la classe.

7.2.2 Classes de suport

Classe Config

Config és una classe singleton que permet el manegament de la configuració de l'aplicació. Existeixen dos tipus de preferències: de sistema i d'usuari. Les preferències de sistema permeten establir els paràmetres que defineixen el funcionament de l'aplicació (mode d'execució, l'ús de memòria compartida, l'adreça del servidor VRPN...) i no són modificables durant el temps d'execució. Les preferències d'usuari permeten definir opcions modificables mentre l'aplicació s'està executant (ús de la imatge a color/mapa de profunditat/seguiment d'esquelets, resolució de les finestres de tipus GLCanvas i KinectCanvas...).

Classe Log

La classe *Log* proporciona un mecanisme de depuració i de sortida d'informació rellevant sobre l'execució de l'aplicació. Concretament, Log implementa un historial de missatges de text. Això permet escriure en un fitxer la informació que desitgem (en format de text pla) durant l'execució del programa. D'aquesta manera pot ajudar a detectar errors en l'etapa de desenvolupament i depuració o, simplement, informar a l'usuari de certes dades rellevants, com per exemple errors esdevinguts a l'aplicació.

Aquesta classe és una classe singleton i utilitza un únic historial durant una mateixa execució del programa.

Classe LogDialog

LogDialog és una classe pertinent a la interfície gràfica de l'aplicació. És una classe simple que deriva de *wxDialog* pertinent a la llibreria de wxWidgets (secció 6.5). Aquesta classe permet a l'usuari veure, en qualsevol moment i de forma senzilla durant l'execució de l'aplicació, l'historial de Log.

Classe RenderThread

La classe *RenderThread* neix al realitzar proves per trobar la millor manera d'implementar el bucle de visualització de l'aplicació. Aquesta classe proporciona un mecanisme de rendering basat en un fil d'execució que dóna l'ordre de pintat de les dades de Kinect de forma indefinida. Aquesta classe permet una actualització més ràpida de les finestres de visualització (GLCanvas i KinectCanvas) tot i que no proporciona un framerate fix.

RenderThread deriva de la classe *wxThread* pertinent a la llibreria de wxWidgets (secció 6.5).

Classe RenderTimer

De la mateixa manera que RenderThread, la classe *RenderTimer* apareix a raó de fer proves per trobar el tipus de bucle de visualització adient. RenderTimer implementa un mecanisme de rendering basat en esdeveniments provocats per un comptador de temps. El comptador es configura amb un interval de temps concret, i cada cop que passa el temps especificat, es produeix un esdeveniment que envia l'ordre de pintat.

RenderTimer deriva de la classe *wxTimer* pertinent a la llibreria de wxWidgets (secció 6.5).

Classe Timer

Timer és una classe singleton que proporciona un mecanisme de mesura de temps. Aquesta classe permet al programador definir diferents marques (identificades mitjançant una

cadena de text) per tal de mesurar el temps transcorregut per aquesta marca. Ofereix procediments per iniciar una mesura, resetejar-la, finalitzar-la o obtenir la quantitat de temps transcorregut. Timer ajuda a poder crear funcionalitats dependents del temps, depurar l'aplicació, o fins i tot, mesurar el rendiment i eficiència del programa.

7.2.3 Estructures matemàtiques i geomètriques

Classe Color

La classe *Color* implementa una estructura de dades que representa un color en el format RGBA. Aquesta classe ofereix mètodes per realitzar diferents tipus d'operacions entre diferents instàncies (acumulació i sostracció de colors, multiplicació per constants...). La necessitat d'aquesta classe resideix en la visualització dels esquelets detectats per Kinect i proporciona un mecanisme per dibuixar-los amb diferents colors segons l'identificador d'usuari.

Classe Matrix3

La classe *Matrix3* implementa una representació d'una matriu quadrada 3x3. Matrix3 ofereix mètodes que implementen les operacions més importants que es poden fer amb una matriu quadrada (multiplicació de matrius i de matriu per punt o vector, invertir i transposar matrius...). La idea d'utilitzar aquesta classe és la de representar rotacions en l'espai tridimensional. Aquestes rotacions es poden utilitzar per definir l'orientació d'una articulació d'un esquelet, o per aplicar una rotació a un punt o vector.

Aquesta classe també disposa d'un mètode per crear una matriu de rotació a partir d'un eix i un angle de rotació especificats.

Classe Matrix4

La classe *Matrix4*, de forma similar a Matriu3, implementa una representació d'una matriu quadrada 4x4. L'objectiu d'aquesta classe però, és representar matrius homogènies de transformació, de forma que puguin contenir rotacions i translacions en l'espai tridimensional. Aquestes transformacions es poden utilitzar, per exemple, a l'hora de definir la transformació del sistema de coordenades de la càmera Kinect a un altre sistema de coordenades desitjat.

A part dels mateixos mètodes que proporciona Matriu3, aquesta classe defineix mètodes per crear una matriu de translació o d'escalat segons un vector de translació o un factor d'escalat en un dels eixos cartesians respectivament.

Classe Point

Point defineix una estructura de dades que representa un punt en l'espai tridimensional. Aquesta classe ofereix procediments per tal de realitzar operacions bàsiques amb punts 3D.

Classe Quaternion

Quaternion és una classe que implementa la representació d'un quaternió. Un quaternió és bàsicament un nombre complex (part real i part imaginària) que conceptualment representa una matriu de rotació. Aquesta classe, per tant, permet representar orientacions i/o rotacions en l'espai tridimensional. Quaternion ofereix mètodes similars a Matriu3: composició de quaternions, multiplicació de quaternió per punt o vector... També ofereix altres procediments més complexes que permeten conjugar, invertir, calcular el logaritme o l'exponencial, etc. d'un quaternió o interpolar diversos quaternions de diverses formes (linealment, quadràticament, mitjançant corbes de Bezier...).

Classe Vector

La classe *Vector* és similar a la classe Point. Implementa la representació d'un vector en l'espai tridimensional. La diferència entre aquesta classe i Point resideix en els mètodes que ofereix. Vector permet obtenir la longitud (o norma) del vector, normalitzar-lo, calcular productes escalars i vectorials...

7.3 Arquitectura dels plugins d'interacció

Tot i que l'arquitectura de tot el visualitzador MovingVis és considerablement gran, donat que el desenvolupament d'aquest no entra en l'àmbit del projecte, en aquesta secció únicament es presenta l'arquitectura dels plugins d'interacció. Els codis de color que segueix el diagrama de la figura 7.5 és el mateix que en els anteriors diagrames (secció 7.2) i trobem el següent:

- *Blocs de llibreries/eines*: Amb color taronja
- *Classes principals de múltiples instàncies*: Amb color groc

Seguidament es realitza una breu descripció de cadascuna de les classes que conformen els plugins d'interacció per tal de fer-se una idea general del funcionament.

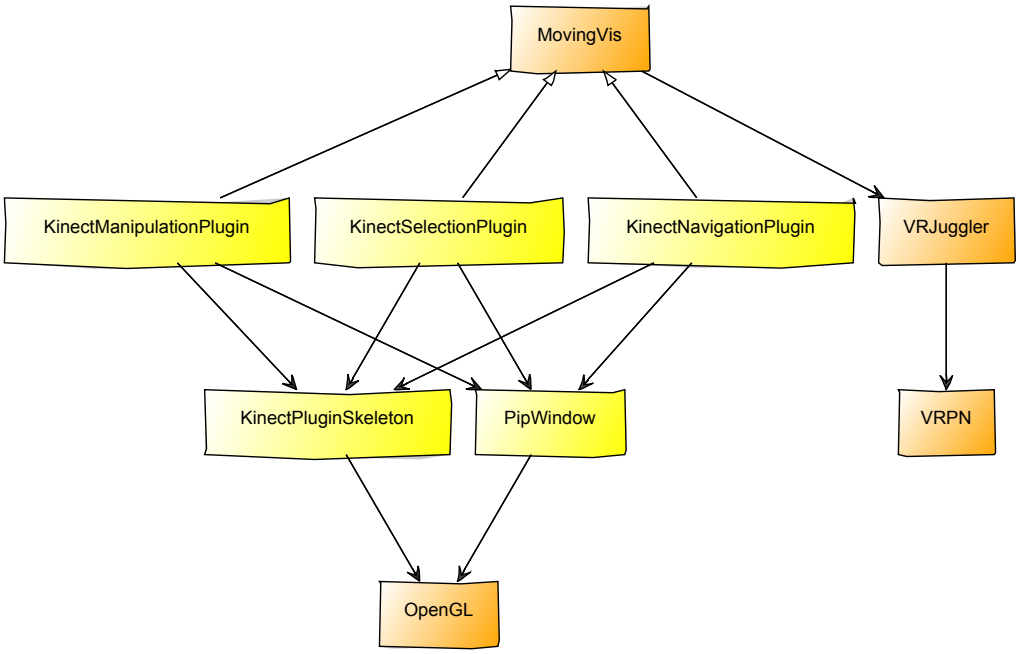


Figura 7.5: Diagrama de l'arquitectura dels plugins d'interacció amb Kinect.

7.3.1 Classes principals

Classe KinectManipulationPlugin

La classe *KinectManipulationPlugin* implementa el plugin d'interacció que realitza la manipulació dels objectes que es troben a l'escena del visualitzador MovingVis. Deriva de la classe *ManipulationPlugin* pertinent a MovingVis. Aquesta classe utilitza una instància de la classe KinectPluginSkeleton per tal de manegar i accedir a la posició de les articulacions dels esquelets detectats per Kinect. La idea és que MovingVis envia les posicions de les articulacions, rebudes a través de VRPN (veure secció 6.4), cap als plugins mitjançant esdeveniments. De la mateixa forma, si es tracta d'un esdeveniment de posició, els plugins el passen a la classe KinectPluginSkeleton per tal que aquesta faci el manegament oportú.

A partir de la informació de les articulacions que rep, aquesta classe realitza el reconeixement de gestos similars als de "Pinch-In" i "Pinch-Out" (zoom en iPhone) per tal de modificar la mida dels objectes seleccionats de l'escena. El reconeixement dels gestos i l'actualització de l'escena es realitza en el moment en què MovingVis envia un esdeveniment d'actualització al plugin (veure secció 8.2.3).

KinectManipulationPlugin també disposa d'un mètode de render el qual es crida des de MovingVis per tal que el plugin pugui pintar certa informació que pugui ser rellevant per informar a l'usuari. És en aquest mètode on es fa ús de la classe PipWindow per

tal de pintar una petita finestra en la pantalla. En aquesta finestra es mostra l'esquelet rebut i informació sobre l'acció que s'està realitzant (fer més gran/petit) sobre l'objecte seleccionat.

Classe KinectNavigationPlugin

La classe *KinectNavigationPlugin* implementa el plugin d'interacció que realitza navegació per l'escena 3D del visualitzador MovingVis. Deriva de la classe *NavigationPlugin* pertinent a MovingVis. El funcionament d'aquesta classe és totalment anàleg a la classe KinectManipulationPlugin, la diferència però, és que els gestos reconeguts són específics per tal de realitzar la navegació.

En aquest cas, s'utilitzen ambdues mans per tal de definir una direcció de navegació. A partir de la posició de les dues mans podrem controlar si el moviment de la càmera és cap endavant o cap enrere, així com la velocitat de navegació (veure secció 8.2.3).

Aquesta classe, a més de l'esquelet, mostra de forma intuïtiva, sobre la PipWindow, la direcció en la que la càmera està girant i indica si el moviment és estacionari, cap endavant o cap enrere. Això dóna a l'usuari un ajut constant dels gestos i accions que s'estan detectant.

Classe KinectPluginSkeleton

KinectPluginSkeleton és una classe que implementa una estructura per manegar els esquelets que MovingVis (i per conseqüent, els plugins) rep a través de VRPN (veure secció 6.4). Aquesta classe emmagatzema les dades de cadascuna de les articulacions que conformen un esquelet. A més, realitza activament el seguiment de les articulacions al llarg del temps per tal de detectar articulacions errònies o perdudes (han desaparegut del radi d'acció de Kinect).

Aquesta classe proporciona mètodes per accedir i modificar les articulacions i/o segments (braços, cames, columna vertebral...) així com per pintar l'esquelet complet mitjançant OpenGL (veure secció 6.7).

Classe KinectSelectionPlugin

La classe *KinectSelectionPlugin* implementa el plugin d'interacció que realitza la selecció dels objectes que es troben a l'escena del visualitzador MovingVis. Deriva de la classe *SelectionPlugin* pertinent a MovingVis. El funcionament d'aquesta classe és totalment anàleg a la classe KinectManipulationPlugin, la diferència però, és que els gestos reconeguts són diferents, així com l'actualització de l'escena que es realitza.

En aquest cas, utilitzant una única mà, es defineix un raig de selecció que permet escollir un objecte de l'escena (veure secció 8.2.3).

Aquesta classe, a més de mostrar l'esquelet i un indicador per mostrar que s'està en mode selecció en la PipWindow, dibuixa el raig de selecció, cosa que permet a l'usuari veure quin objecte està escollint.

Classe PipWindow

PipWindow (Picture In Picture Window) proporciona un mecanisme que permet definir una petita finestra dins la finestra original del visualitzador. Aquesta petita finestra és totalment independent de l'aplicació que s'executi en MovingVis i el programador pot definir, de forma senzilla, la mida i posició de la mateixa. L'ús d'aquesta finestra és simple, només cal definir certs paràmetres de la finestra (mida, posició, color d'esborrat, color i mida de la frontera del requadre...) i realitzar dues crides: activar i desactivar la finestra. Entre aquestes dues crides, es poden utilitzar qualsevol de les funcions d'OpenGL (veure secció 6.7), de la mateixa forma que ho faríem amb la finestra original.

El format de les crides de les funcions que proporciona PipWindow segueix l'estil d'OpenGL, això fa que el codi resultant sigui simple, fàcilment comprensible i elegant.

Capítol 8

Implementació del projecte

Per tal de mostrar amb una mica més de detall el desenvolupament d'aquest projecte, aquest capítol presenta el funcionament del manegador de càmeres Kinect (MultiKinect) i dels plugins d'interacció del visualitzador. Concretament, es presenta la interfície gràfica d'ambdues parts, els modes d'execució de MultiKinect, la configuració de VRJuggler (veure secció 6.6) per als plugins d'interacció i les parts de programació més rellevants tant de MultiKinect com dels plugins del visualitzador (obtenció i tractament de les dades de Kinect i detecció de gestos).

8.1 Implementació del manegador MultiKinect

En aquesta secció s'explica el funcionament més rellevant de MultiKinect. Primerament es mostra la disposició de la interfície gràfica i es dóna una breu explicació de les opcions que aquesta permet dur a terme. Posteriorment, es realitza una explicació de cadascun dels modes de funcionament del manegador de càmeres Kinect així com el seu propòsit. La part final d'aquesta secció explica d'una forma més detallada com funciona internament el programa, és a dir, el procediment que es segueix per tal d'obtenir i manipular les dades provinents de Kinect i fer-les accessibles a altres aplicacions.

8.1.1 Interfície gràfica

Com ja s'ha comentat en capítols i seccions anteriors, MultiKinect té diversos tipus de processos i modes d'execució. Dels dos tipus de processos que existeixen, anomenarem procés "slave" a aquell que s'encarrega de manegar i obtenir directament les dades d'una càmera Kinect. El procés encarregat de centralitzar les dades proporcionades pels processos "slave" és el procés "master".

La figura 8.1 mostra el marc inicial de la interfície gràfica del procés "slave". Com es pot observar, disposa de tres panells de tipus *KinectCanvas*, una per mostrar la imatge a color, altre per mostrar el mapa de profunditat i altre per dibuixar els esquelets detectats.

També disposa d'una barra de menú a la part superior del marc, on trobem els menús *File* i *View*, i una barra d'estat a la part inferior, la qual, mostra contínuament els frames per segon de Kinect i de l'aplicació. El framerate de Kinect es correspon a la quantitat de dades de Kinect (imatge a color, mapa de profunditat i/o esquelets) que la càmera proporciona per segon. El framerate de l'aplicació fa referència a la freqüència de pintat, és a dir, el nombre d'ordres de pintat que s'envien per segon.

Figura 8.1: Interfície gràfica principal del procés "slave" de MultiKinect.

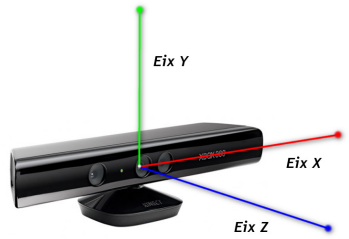
A la figura 8.3 també es pot observar la interfície del procés "slave" però, en aquest cas, es mostren els menús *File* i *View* desplegats, i també, el panell d'opcions d'aquest procés. Al menú *File* trobem les opcions *Restart* i *Exit* per tal de reiniciar o finalitzar, respectivament, el procés "slave". Al menú *View* es troben dues opcions més, *View options* per mostrar i amagar el panell d'opcions i *View log...* que permet obrir una finestra on es mostra el contingut del fitxer de log del procés.

Com es pot veure, el panell d'opcions disposa de diversos camps per tal de poder modificar algunes parts de la configuració del procés:

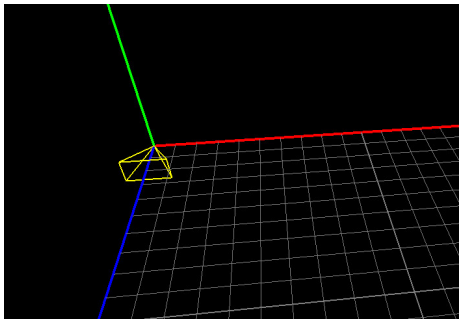
- *Enable color image*, *Enable depth map* i *Enable skeleton tracking* permeten activar/desactivar l'obtenció de la imatge a color, el mapa de profunditat o la detecció d'esquelets respectivament, així com mostrar/amagar el panell KinectCanvas corresponent.
- *Enable near mode* i *Enable seated mod* permeten activar/desactivar els modes "a prop" i "assegut" afegits en la versió 1.5 del driver de Kinect (veure secció 6.2).
- *Stream resolution* és un desplegable que permet seleccionar la resolució dels panells KinectCanvas. Les resolucions disponibles són 80x60, 320x240 i 640x480.
- *Rotation X*, *Rotation Y* i *Rotation Z* permeten definir la rotació dels eixos del sistema de referència de la càmera Kinect (que es pot observar a la figura 8.2a) respecte al sistema de referència d'una habitació virtual definida al procés "master" (veure figura 8.4). Inicialment es fa coincidir el sistema de referència de la càmera

Kinect amb el sistema de referència de l'habitació virtual. La rotació resultant s'obté segons l'equació $Rot = Rot_z(\phi_z) * Rot_y(\phi_y) * Rot_x(\phi_x)$ on ϕ_x, ϕ_y i ϕ_z són els angles dels tres camps respectivament. Aquests camps reben angles en unitats de graus. La rotació a l'eix X no es permet editar per l'usuari donat que aquest angle es mesura automàticament mitjançant l'acceleròmetre que Kinect disposa.

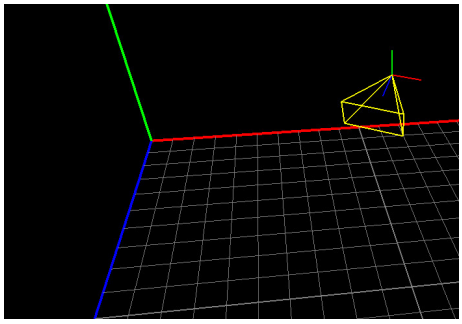
- *Translation X*, *Translation Y* i *Translation Z* permeten definir, de forma similar als camps de rotació, la translació de l'origen de coordenades del sistema de referència de Kinect respecte al sistema de referència de l'habitació virtual. Aquests camps reben distàncies en unitats de centímetres. La figura 8.2 il·lustra la transformació del sistema de referència de la càmera Kinect respecte al sistema de referència de l'habitació virtual.



(a) Sistema de referència de Kinect



(b) Sistema de referència de Kinect coincidint amb el de l'habitació



(c) Sistema de referència de Kinect transformat

Figura 8.2: Transformació del sistema de referència de Kinect.

- *Elevation angle* es compon d'un "slider" i un camp de text (fix) per tal de modificar i mostrar, respectivament, l'angle d'inclinació de la càmera Kinect mitjançant el motor del que aquesta disposa a la seva base (veure secció 6.1).

A la figura 8.4 es mostra el marc inicial de la interfície gràfica del procés "master". Trobem que disposa d'un panell de tipus *GLCanvas* on es visualitza la representació virtual

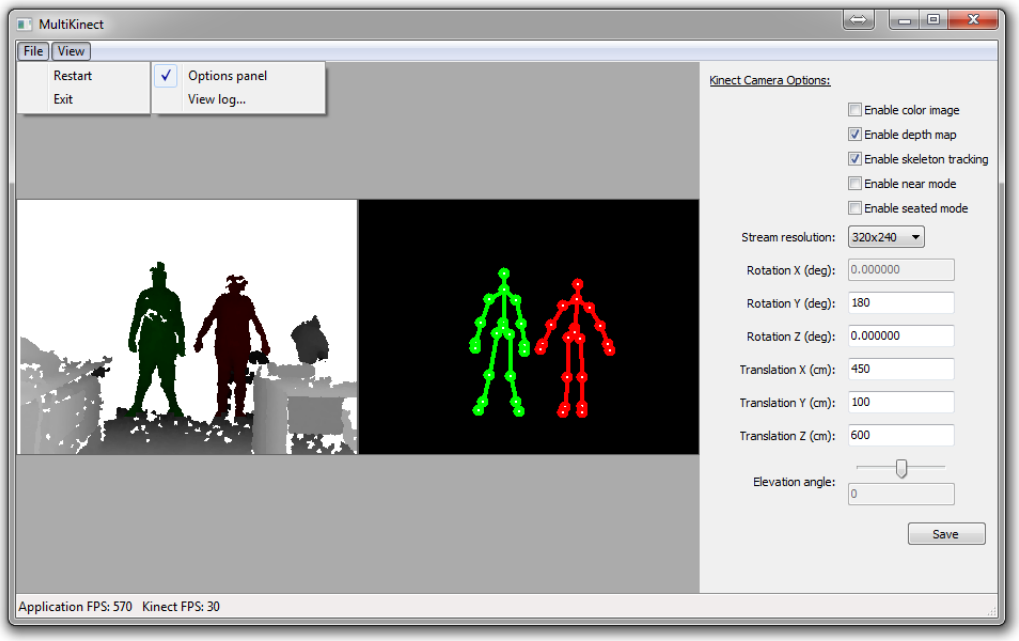


Figura 8.3: Menús i opcions del procés "slave" de MultiKinect.

d'una habitació (sistema de referència i una quadrícula per representar el terra), amb les càmeres Kinect connectades (piràmides de color groc i sistema de referència de la càmera) i els esquelets detectats. De la mateixa forma que el procés "slave", el marc del procés "master" disposa d'una barra de menú amb els menús *File* i *View* i d'una barra d'estat on es mostra contínuament el framerate de l'aplicació (freqüència de pintat de GLCanvas).

La figura 8.5 mostra els menús *File* i *View* desplegats així com el panell d'opcions de la interfície del procés "master". Com es pot veure, els menús *File* i *View* disposen de les mateixes opcions que en el cas del procés "slave", tot i que en aquest cas, les opcions de *Restart* i *Exit* no reinicien o finalitzen únicament el procés "master", sinó que també reinicien o finalitzen tots els processos "slave" que "master" hagi llençat. Les opcions *Options panel* i *View log...* serveixen pel mateix que en cas del procés "slave" (mostrar/amagar el panell d'opcions i mostrar una finestra amb el log del procés).

Els camps del panell d'opcions permeten modificar els següents aspectes de l'aplicació:

- El desplegable *Stream resolution* permet modificar la resolució del panell GLCanvas. Les resolucions disponibles són 320x240, 640x480, 800x600 i 1024x768.
- *Combine skeletons* serveix per combinar els esquelets detectats per diversos Kinect en un únic esquelet més precís. En cas de tenir desactivada aquesta opció es mostren els esquelets dels diferents Kinect per separat.
- *Room origin X*, *Room origin Y* i *Room origin Z* tenen la funció de modificar la coor-

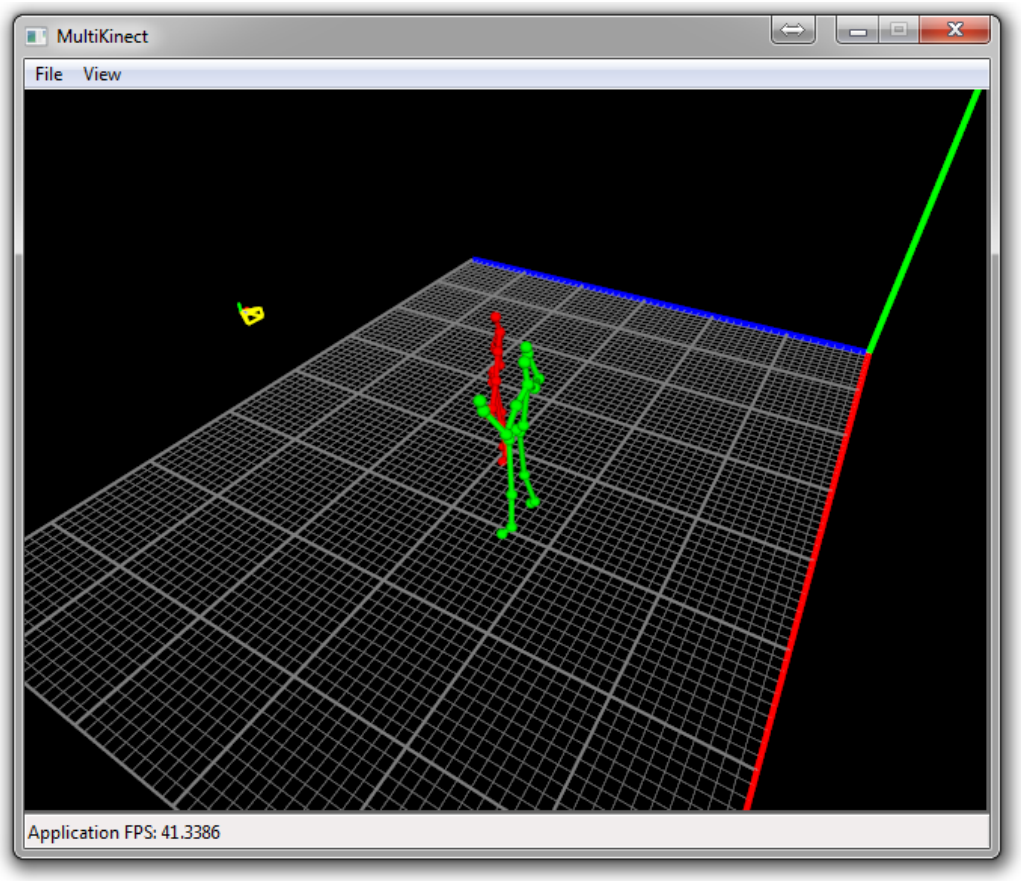


Figura 8.4: Interfície gràfica principal del procés “master” de MultiKinect.

denada 3D de l'origen de l'habitació respecte el sistema de referència fixat. Aquests camps utilitzen centímetres com a unitat de mesura.

- *Room width*, *Room height* i *Room depth* permeten modificar les dimensions de l'habitació virtual que s'utilitza. Aquests camps reben distàncies en unitats de centímetres.

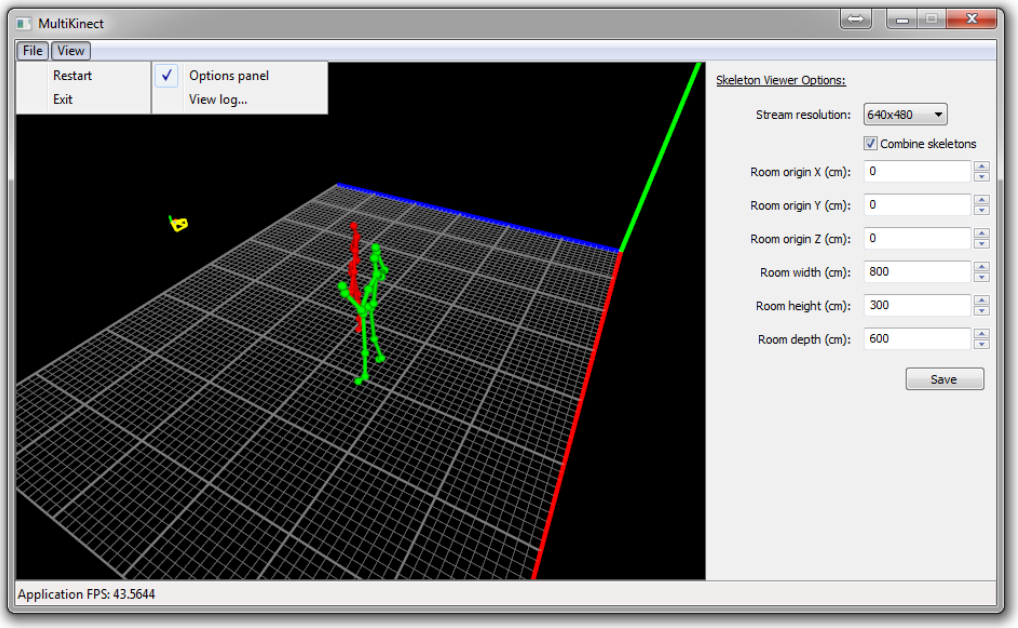
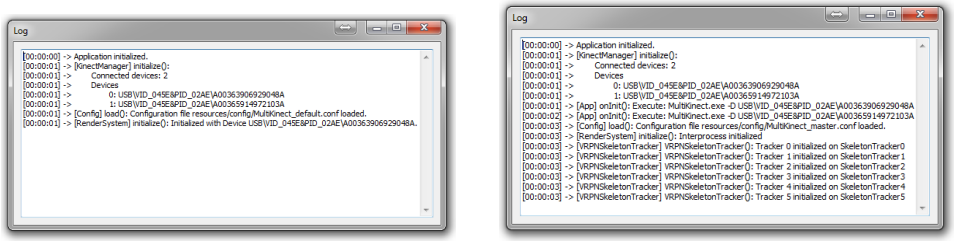


Figura 8.5: Menús i opcions del procés "master" de MultiKinect.

Com es pot veure a la figura 8.6, tant la interfície del procés "slave" com la del procés "master" utilitzen la mateixa finestra per tal de mostrar el fitxer de log. Es tracta d'una finestra molt simple que conté un panell que permet mostrar text.



(a) Log del procés "slave" (b) Log del procés "master"

Figura 8.6: Finestra de Log dels processos "master" i "slave" de MultiKinect.

Tal i com s'explica a la secció 8.1.2, existeix un mode d'execució de MultiKinect que no utilitza totes les càmeres Kinect connectades alhora, sinó que permet utilitzar una única

càmera de les disponibles. En aquest cas, el procés "master" i el procés "slave" són el mateix. La seva interfície gràfica és la mateixa que la del procés "slave" amb una única diferencia: al panell d'opcions es disposa d'un desplegable que permet seleccionar quina càmera Kinect volem utilitzar. La figura 8.7 mostra la interfície gràfica amb el selector de Kinect en el panel d'opcions.

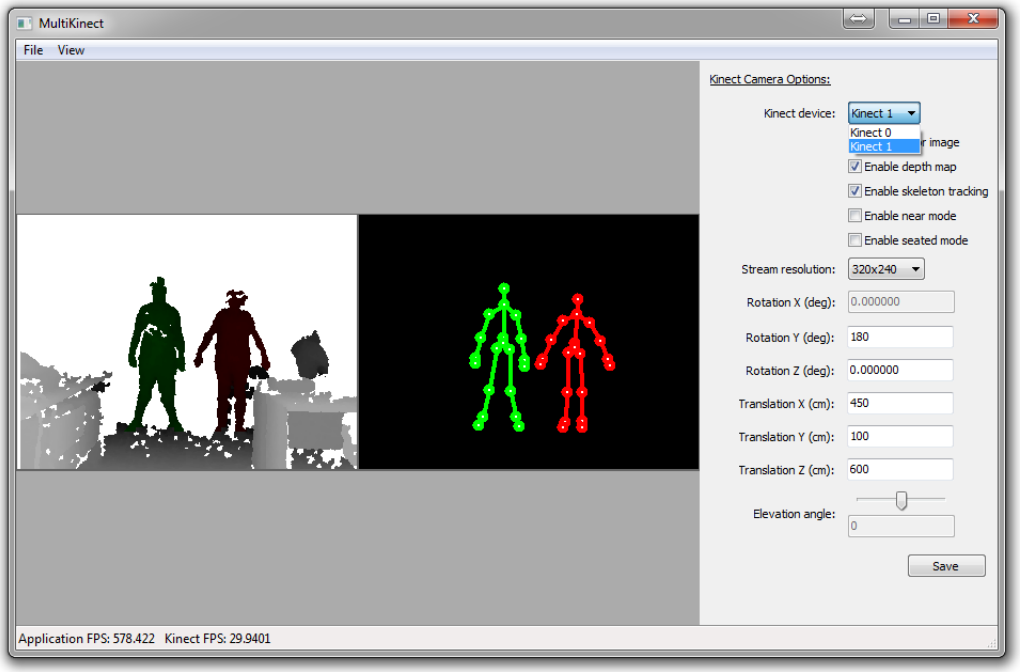


Figura 8.7: Interfície gràfica de MultiKinect en mode "selector de Kinect".

8.1.2 Modes d'execució

Per tal de satisfer diversos tipus de funcionament per part de les aplicacions que vulguin utilitzar MultiKinect, s'han implementat dos modes d'execució diferents, un d'ells utilitza les dades de les càmeres Kinect connectades alhora, i l'altre, utilitza les dades d'una única càmera Kinect de les disponibles. A més, per tal de fer tasques de depuració durant el desenvolupament del projecte, s'ha implementat un tercer mode d'execució el qual implementa un client VRPN (veure secció 6.4) per tal de visualitzar les dades emeses per algun dels altres dos modes d'execució.

Per tal d'executar un mode o un altre, s'utilitzen els arguments que es passen al executable de l'aplicació. Segons el mode que s'utilitzi, l'aplicació necessita altres arguments per definir alguns dels paràmetres del programa, com per exemple, l'adreça del servidor VRPN o el nom que identifica els trackers per accedir remotament als esquelets.

Seguidament es realitza una breu descripció del funcionament de cadascun d'aquests tres modes d'execució.

Mode MultiKinect bàsic

El mode d'execució bàsic de MultiKinect utilitza les dades de totes les càmeres Kinect connectades alhora. El funcionament de l'aplicació en aquest mode d'execució es basa en l'ús d'un procés (anomenat "slave") per cadascun dels Kinect, donat que, el driver de Microsoft no permet accedir a les dades de detecció d'esquelets de diversos Kinect alhora en un mateix procés.

Per tal de centralitzar les dades obtingudes per cadascun d'aquests processos "slave", al executar l'aplicació, es crea un procés "master" (que és l'encarregat de llençar cadascun dels processos "slave") capaç d'accedir a les dades mitjançant l'ús d'un mòdul de memòria compartida. El procés "master" és l'encarregat de crear un segment de memòria compartida, mitjançant la classe *SharedMemoryManager* (veure secció 7.2.1), per a cadascun dels processos "slave". Això permet als processos "slave" crear objectes en aquests segments de manera que siguin accessibles, tant per lectura com per escriptura, per ambdós processos.

El procés de tipus "master" és l'encarregat de tractar, escollir i combinar les dades de totes les càmeres Kinect connectades. Aquest procés disposa d'un servidor VRPN que és l'encarregat d'enviar a través de la xarxa les dades resultants dels esquelets detectats, i d'aquesta manera, fer-les accessibles per a qualsevol aplicació externa.

Per tal d'executar MultiKinect en aquest mode, cal cridar a l'executable amb l'argument *-M*. Addicionalment, cal indicar el nom que identificarà els trackers locals per poder accedir a les dades dels esquelets a través de VRPN. Una comanda d'execució vàlida seria:

```
MultiKinect.exe -M SkeletonTracker
```

Mode MultiKinect per selecció

El mode d'execució de MultiKinect per selecció utilitza les dades d'una única càmera Kinect de les connectades. Aquest mode utilitza un únic procés que fa tant les funcions de "master" com les de "slave". El funcionament d'aquest mode es basa en obtenir les dades d'un dels Kinect, tractar aquestes i, a través d'un servidor VRPN, fer accessibles les dades dels esquelets a altres aplicacions externes.

La interfície gràfica d'aquest mode d'execució és la corresponent al procés de tipus "slave" del mode bàsic i disposa d'un selector per tal de poder canviar quina de les càmeres connectades es vol utilitzar (veure figura 8.7).

Per executar MultiKinect en aquest mode d'execució cal cridar l'executable amb l'argument *-S*. Al igual que en el mode bàsic, cal indicar el nom que identificarà els trackers

locals per poder accedir a les dades dels esquelets a través de VRPN. Un exemple de comanda vàlida seria:

```
MultiKinect.exe -S SkeletonTracker
```

Mode client VRPN

L'ultim mode d'execució de MultiKinect apareix amb l'objectiu de facilitar alguna de les tasques de depuració durant el desenvolupament del projecte. Es tracta d'un client VRPN per tal de visualitzar les dades emeses per qualsevol dels altres dos modes d'execució. En aquest mode, l'aplicació executa un únic procés que inicialitza la classe *RenderSystem* de forma que rebi les dades a través del client VRPN (veure secció 7.2.1).

Per tal de mostrar les dades rebudes a través de VRPN d'una forma senzilla i clara, aquest mode utilitza la mateixa interfície gràfica que el procés de tipus "master" del mode bàsic. D'aquesta manera es poden visualitzar en 3D els esquelets detectats en qualsevol dels altres dos modes i la representació virtual de l'habitació, les mides de la qual són modificables d'igual forma a través del panell d'opcions (veure figura 8.5).

Per executar l'aplicació MultiKinect en aquest mode, cal cridar l'executable amb l'argument *-C*. A més, cal afegir dos arguments, un per indicar el nom dels trackers remots que emeten les dades dels esquelets a través del servidor VRPN i un altre per definir l'adreça de xarxa d'aquest servidor. Una comanda vàlida per executar MultiKinect en aquest mode seria la següent:

```
MultiKinect.exe -C SkeletonTracker localhost
```

8.1.3 Obtenció i tractament de les dades de Kinect

Per tal d'explicar el procediment d'obtenció i tractament de les dades de Kinect, es mostra el funcionament de MultiKinect en el mode d'execució bàsic. El mode d'execució per selecció funciona de la mateixa manera exceptuant la part d'elecció/combinació d'esquelets.

Primerament, cadascun dels processos "slave" de l'aplicació obtenen, a través del driver de Kinect, les dades d'imatge a color, mapa de profunditat i/o esquelets detectats.

Les dades d'imatge a color proporcionades pel driver no han de ser tractades ja que, per cada píxel de la imatge, proporciona tres bytes que codifiquen el seu color RGB. Es tracta doncs d'un mapa de bits RGB que es pot visualitzar directament.

El mapa de profunditat es representa mitjançant un valor de dos bytes per cada píxel, el qual, codifica l'índex d'usuari als bits 0-2 i el valor de profunditat a la resta de bits (3-15). El tractament d'aquestes dades es basa en convertir cadascun dels valors del mapa de profunditat en un color RGB. L'algorisme que transforma un valor de profunditat (*depthValue* al codi) a RGB és el següent:

```
realDepth   := depthValue & 0xFFF8          (Bits 3-15)
realDepth   := 255 - 256*realDepth/4095     (Profunditat entre 0 i 255)
playerIndex := depthValue & 0x0007          (Bits 0-2)

if (playerIndex = 0) then playerIndex := 0x0007 endif

if ((playerIndex & 0x0001) = 1) then colorR := realDepth/255 endif
if ((playerIndex & 0x0002) = 1) then colorG := realDepth/255 endif
if ((playerIndex & 0x0004) = 1) then colorB := realDepth/255 endif
```

D'aquesta manera, si no hi ha cap usuari identificat, el color resultant és gris, per contra, s'utilitza un dels color primaris o barreja d'aquests amb igual proporció (recordem que com a màxim es poden detectar sis usuaris, per tant, el gris mai s'utilitza per identificar un usuari). La intensitat del color resultant queda determinada pel valor real de profunditat. Les constants per convertir *realDepth* en un valor entre 0 i 255 s'han extret dels exemples proporcionats per Microsoft.

El driver de Kinect proporciona, per cada articulació de cada esquelet detectat, un punt 3D i un quaternió (o matriu de rotació) representant la posició i orientació de cadascuna dins del sistema de referència de la càmera Kinect. L'únic tractament inicial que es realitza a aquestes dades és la inserció de les mateixes en una estructura de tipus KinectSkeleton (veure secció 7.2.1).

Posteriorment a l'obtenció i tractament inicial de les dades de Kinect, cadascun dels processos "slave" escriuen aquestes dades en els segments de memòria compartida dels que disposen. El procés "master" de l'aplicació accedeix a les dades dels esquelets de totes les Kinect a través d'aquests segments. Cadascuna de les càmeres Kinect disposa d'una matriu homogènia de transformació (definida per l'usuari) que converteix el sistema de referència de la càmera en un altre sistema de referència desitjat. Aquestes matrius són utilitzades pel procés "master" per transformar les dades dels esquelets i poder així combinar-les. Els únics esquelets que es poden combinar entre si, són els que tenen el mateix identificador d'usuari, i les regles que es segueixen per tal de fer-ho, són les següents:

1. Si una articulació és vàlida en més d'un dels esquelets, s'utilitza una mitjana aritmètica ponderada (cada articulació té un pes específic) de la posició i orientació de totes elles

2. Si una articulació no és vàlida en algun dels esquelets, aquesta s'ignora i s'utilitzen la resta segons la primera regla

3. Una articulació és vàlida si la càmera Kinect la està detectant

4. El pes d'una articulació ve definit per dos factors: el temps de validesa de la mateixa i l'orientació de l'esquelet respecte al seu Kinect (les articulacions d'un esquelet de front a la càmera tenen més pes que les d'un que estigui de costat)

Finalment, un cop s'han combinat els esquelets dels diferents Kinect, les articulacions resultants són enviades a través del servidor VRPN implementat a la classe VRPNServer (veure secció 7.2.1), per fer-les accessibles a aplicacions externes. La següent taula indica totes les articulacions que s'envien així com l'identificador de cadascuna:

ID	Articulació	ID	Articulació
0	Cap	10	Columna vertebral
1	Espatlla esquerra	11	Maluc esquerre
2	Espatlla centre	12	Maluc centre
3	Espatlla dreta	13	Maluc dret
4	Colze esquerre	14	Genoll esquerre
5	Colze dret	15	Genoll dret
6	Canell esquerre	16	Turmell esquerre
7	Canell dret	17	Turmell dret
8	Mà esquerra	18	Peu esquerre
9	Mà dreta	19	Peu dret

Taula 8.1: Articulacions utilitzades per MultiKinect.

8.2 Implementació dels plugins d'interacció

Aquesta secció mostra les parts més rellevants de la implementació dels plugins d'interacció per al visualitzador MovingVis. En primer lloc es presenta la interfície gràfica que s'ha dissenyat per la finestra d'ajuda dels plugins. Seguidament s'explica la part de configuració de VRJuggler que cal realitzar per obtenir correctament les dades dels esquelets de MultiKinect a través de VRPN. La part final presenta els gestos que s'han definit per poder interaccionar amb l'entorn virtual en cadascun dels plugins.

8.2.1 Interfície gràfica

Anteriorment s'ha comentat que la finestra d'ajuda utilitzada en els plugins d'interacció del visualitzador MovingVis, és independent de l'aplicació que s'estigui executant. La figura 8.8 mostra una aplicació executant-se sota el visualitzador MovingVis. A la cantonada d'abaix a la dreta es pot apreciar la finestra d'ajuda d'un dels plugins d'interacció. La posició i mida de la finestra es pot configurar mitjançant els mètodes que proporciona la classe PipWindow (veure secció 7.3.1), que és l'encarregada d'implementar aquesta finestra.

La informació mostrada en la finestra d'ajuda és similar en els tres plugins d'interacció. Com es pot veure en la figura 8.9, en tots tres plugins es divideix la finestra en dues parts. A la part de l'esquerra es dibuixa en color vermell l'esquelet rebut a través de VRPN.

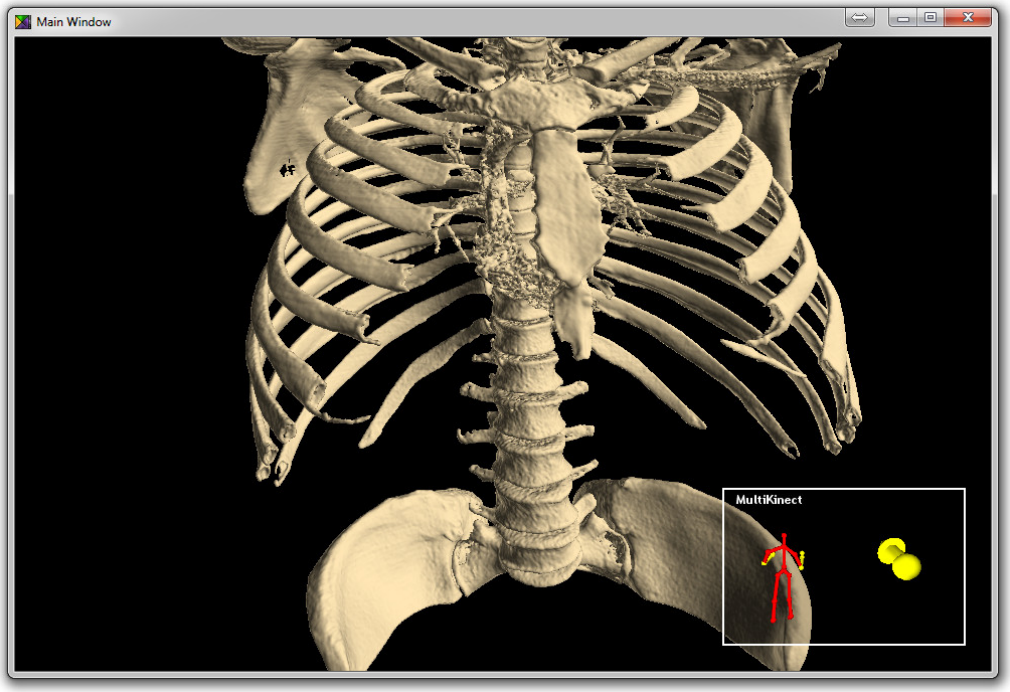


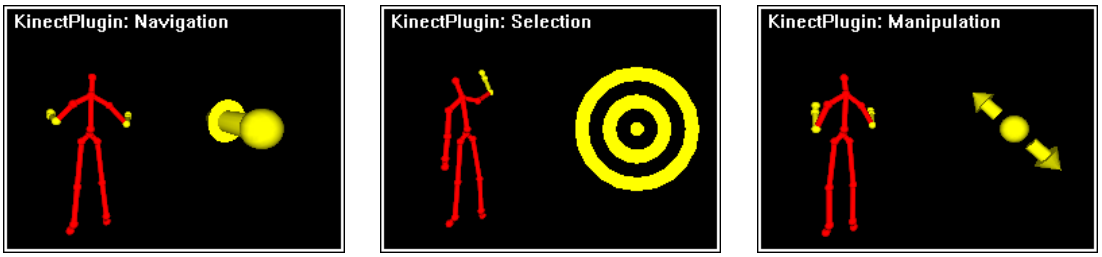
Figura 8.8: Interfície gràfica d'una aplicació executant-se en el visualitzador MovingVis.

Segons l'acció que s'està realitzant en els plugins, els braços de l'esquelet canvien de color per indicar si es troben o no actius. A la part de la dreta es mostra informació que depèn de cadascun dels plugins. En el cas del plugin de navegació, es dibuixa una fletxa 3D que indica la direcció en la que està girant la càmera i, a més, es mou cap endavant o cap enrere segons si la càmera està avançant o retrocedint. Al plugin de selecció es mostra una espècie de diana per tal d'indicar que ens trobem en aquest mode. Per últim, la finestra del plugin de manipulació mostra dues petites fletxes en 3D per indicar si s'està realitzat el gest de "Pinch-In" o "Pinch-Out" per empetitir o fer més gran, respectivament, l'objecte seleccionat. El canvi de visualitzar la finestra d'un plugin o d'un altre és inapreciable per l'usuari.

8.2.2 Configuració de VRJuggler per la recepció d'esquelets via VRPN

La llibreria VRJuggler (veure secció 6.6) proveeix d'un framework que conté una gran quantitat d'opcions i mòduls per realitzar tot tipus de tasques referents a l'entrada i sortida de l'aplicació. Per tal de manegar totes aquestes opcions i mòduls, VRJuggler proporciona un mecanisme de fitxers de configuració codificats en XML[1]. Donat que per l'ús dels plugins d'interacció cal utilitzar VRPN, s'ha d'indicar a VRJugler, mitjançant

[1] *Extensible Markup Language* (XML) és un llenguatge de marques (similar a HTML) que defineix un conjunt de regles per codificar documents en un format que sigui llegible tant per les persones com per les màquines (o programes).



(a) PipWindow de navegació (b) PipWindow de selecció (c) PipWindow de manipulació

Figura 8.9: Finestres d'ajuda dels tres plugins d'interacció.

els fitxers de configuració, la direcció de xarxa del servidor VRPN i com tractar les dades que es rebran, que en aquest cas es tracta d'indicar l'identificador de cadascuna de les articulacions i definir un àlies per tal d'accedir a aquestes dades des de l'aplicació.

Tots els fitxers de configuració de VRJuggler tenen la mateixa estructura. Existeix l'etiqueta <*elements*> dins de la qual es defineixen la resta d'opcions.

Per indicar a VRJuggler que ha d'utilitzar el mòdul de VRPN, s'han d'afegir les següents línies:

```
<input_manager name="VRPNInputManager" version="2">
    <driver_path>${VJ_BASE_DIR}/lib/gadgeteer/drivers</driver_path>
    <driver>VRPN_drv</driver>
</input_manager>
```

Cal notar que el nom de l'element <*input_manager*> i el valor de <*driver_path*> poden ser diferents.

Una vegada feta la declaració del mòdul, cal definir un dispositiu VRPN. En la definició han de constar obligatòriament la direcció de xarxa del tracker que emetrà les dades d'un dels esquelets i la quantitat de dades diferents que es poden rebre (en el nostre cas, 20 articulacions). Opcionalment es pot declarar un filtre on es defineixen dues rotacions i translacions que es poden aplicar a les dades rebudes. Per tal de definir un dispositiu VRPN cal afegir, per exemple, el següent:

```
<vrpn name="VRPN_Kinect" version="2">
    <tracker_server>SkeletonTracker0@localhost</tracker_server>
    <button_server />
    <analog_server />
    <tracker_count>20</tracker_count>
    <button_count>0</button_count>
    <analog_count>0</analog_count>
    <min>-1.0</min>
    <max>1.0</max>
```

```
    <position_filters>
        <position_transform_filter name="Googles" version="1">
            <pre_translate>0.0</pre_translate>
            <pre_translate>0.0</pre_translate>
            <pre_translate>0.0</pre_translate>
            <pre_rotation>0.0</pre_rotation>
            <pre_rotation>0.0</pre_rotation>
            <pre_rotation>0.0</pre_rotation>
            <custom_scale>1.0</custom_scale>
            <device_units>1.0</device_units>
            <post_translate>0.0</post_translate>
            <post_translate>0.0</post_translate>
            <post_translate>0.0</post_translate>
            <post_rotation>0.0</post_rotation>
            <post_rotation>0.0</post_rotation>
            <post_rotation>0.0</post_rotation>
        </position_transform_filter>
    </position_filters>
</vrpn>
```

Cal notar que tant el nom de l'element *<vrpn>* com el de *<position_transform_filter>* poden ser diferents.

Posteriorment, cal definir un proxy de posició per cadascuna de les articulacions que es vulguin rebre. En la definició del proxy caldrà indicar el dispositiu VRPN, l'identificador de l'articulació i, opcionalment, un filtre per indicar dues rotacions i translacions aplicables a l'articulació (com en el cas del dispositiu VRPN). Com a exemple, es mostra la definició de l'articulació del cap:

```
<position_proxy name="Head Proxy" version="1">
    <device>VRPN_Kinect</device>
    <unit>0</unit>
    <position_filters>
        <position_transform_filter name="Googles" version="1">
            <pre_translate>0.0</pre_translate>
            <pre_translate>0.0</pre_translate>
            <pre_translate>0.0</pre_translate>
            <pre_rotation>0.0</pre_rotation>
            <pre_rotation>0.0</pre_rotation>
            <pre_rotation>0.0</pre_rotation>
            <custom_scale>1.0</custom_scale>
            <device_units>1.0</device_units>
            <post_translate>0.0</post_translate>
            <post_translate>0.0</post_translate>
```

```
            <post_translate>0.0</post_translate>
            <post_rotation>0.0</post_rotation>
            <post_rotation>0.0</post_rotation>
            <post_rotation>0.0</post_rotation>
        </position_transform_filter>
    </position_filters>
</position_proxy>
```

De forma similar al cas anterior, tant el nom de l'element <*position_proxy*> com el de <*position_transform_filter*> poden ser diferents.

VRJuggler implementa el seguiment del cap de l'usuari (en anglès "head tracking") de manera que el programador no hagi de realitzar aquesta tasca. L'únic que cal fer per activar el "head tracking" és definir un proxy per la càmera que s'utilitzarà en l'entorn virtual. El codi que cal afegir, per exemple, és el següent:

```
<position_proxy name="SimCamera Proxy" version="1">
    <device>VRPN_Kinect</device>
    <unit>0</unit>
    <position_filters>
        <position_transform_filter name="Position Filters" version="1">
            <pre_translate>0.0</pre_translate>
            <pre_translate>0.0</pre_translate>
            <pre_translate>0.0</pre_translate>
            <pre_rotation>0.0</pre_rotation>
            <pre_rotation>0.0</pre_rotation>
            <pre_rotation>0.0</pre_rotation>
            <custom_scale>1.0</custom_scale>
            <device_units>1.0</device_units>
            <post_translate>0.0</post_translate>
            <post_translate>0.0</post_translate>
            <post_translate>0.0</post_translate>
            <post_rotation>0.0</post_rotation>
            <post_rotation>0.0</post_rotation>
            <post_rotation>0.0</post_rotation>
        </position_transform_filter>
    </position_filters>
</position_proxy>
```

A diferència de les articulacions, el nom de l'element <*position_proxy*> de la càmera ha de ser *SimCamera Proxy* per tal d'activar el "head tracking" correctament.

Per últim, per tal de definir els àlies de cadascuna de les articulacions i fer-les accessibles a l'aplicació, cal afegir les línies següents (utilitzant com exemple el cas del cap de l'esquelet):

```
<alias name="VJHead" version="1">
    <proxy>Head Proxy</proxy>
</alias>
```

Altre cop, el nom de l'element *<alias>* pot ser el que el programador desitgi. Cal tenir en compte però, que s'ha d'utilitzar el mateix nom quan es vulgui accedir a les dades des de l'aplicació.

8.2.3 Detecció de gestos

El procediment que s'ha dut a terme per implementar la detecció de gestos als plugins d'interacció, es basa en el control de distàncies relatives i direccions que s'estableixen entre dos o més articulacions de l'esquelet detectat. El desavantatge que tenim amb Kinect, respecte altres tipus d'interfícies de comandament, és la falta d'un element activador que ens permeti indicar quan volem o no realitzar una acció determinada. Per aquest motiu, en tots tres plugins d'interacció, s'han definit dos gestos simples per tal d'utilitzar-los com a activador. Concretament, s'han definit els gestos anomenats "Push" i "Pull" que consisteixen en el següent:

- *Push* és el gest produït al tenir el braç cap endavant, és a dir, quan la mà està allunyada, superant un cert llindar, del cos de l'esquelet en direcció perpendicular a la recta establerta entre l'espatlla esquera i dreta.
- *Pull* té el mateix funcionament que Push però utilitzant un llindar molt més petit.

El llindar utilitzat en ambdós gestos és relatiu a la distància que existeix entre l'espatlla esquerra i dreta de l'esquelet. D'aquesta manera es fa que la detecció d'aquests gestos no depengui de l'estatura i/o complexió de l'usuari. A més, s'estableix una zona intermèdia entre els dos llindars en la qual no s'identifica cap dels dos gestos. Aquesta zona intermèdia s'ha anomenat "zona neutra".

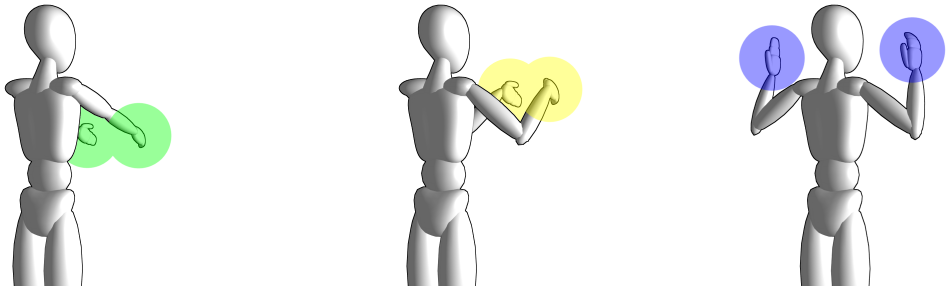
A part dels gestos bàsics esmentats, existeix la posició de relax. Mentre l'usuari es troba en aquesta posició, no es detecta cap gest i no es produeix cap interacció amb l'entorn virtual. La posició de relax és la posició natural de l'usuari amb els braços rectes cap abaix.

Cal notar que tant la posició de relax com els gestos esmentats es detecten de forma independent per cadascun dels dos braços de l'esquelet.

Utilitzant els gestos de "Push", "Pull" i la "zona neutra", cadascun dels plugins implementa altres gestos que permeten realitzar la interacció amb l'escena virtual. El funcionament i gestos dependents són els següents:

- *Plugin de navegació*: Per tal de controlar el moviment de la càmera de l'escena virtual, s'utilitza el gest de "Push" i "Pull" (amb les dues mans alhora) per moure's cap endavant o cap enrere, respectivament, en la direcció de visió. A més, si ens trobem en la "zona neutra" o en el gest de "Push" (sempre amb les dues mans),

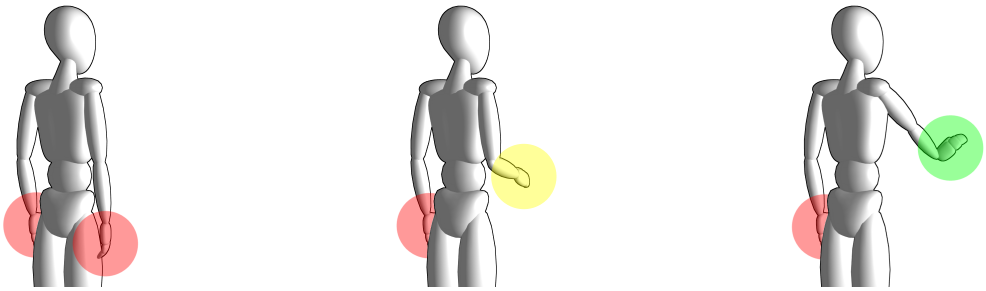
es modifica la direcció de visió de la càmera segons el vector definit entre les dues mans i l'espatlla esquerra i dreta de l'esquelet. La navegació resultant amb aquests gestos és semblant al vol lliure d'un avió (afegint però la capacitat d'anar cap enrere). Mitjançant la distància entre les dues mans es pot controlar la velocitat de moviment de la càmera. A la figura 8.10 es pot observar una representació visual dels gestos de navegació.



(a) Navegació cap endevant (b) Rotació de la càmera (c) Navegació cap enrere

Figura 8.10: Gestos de navegació.

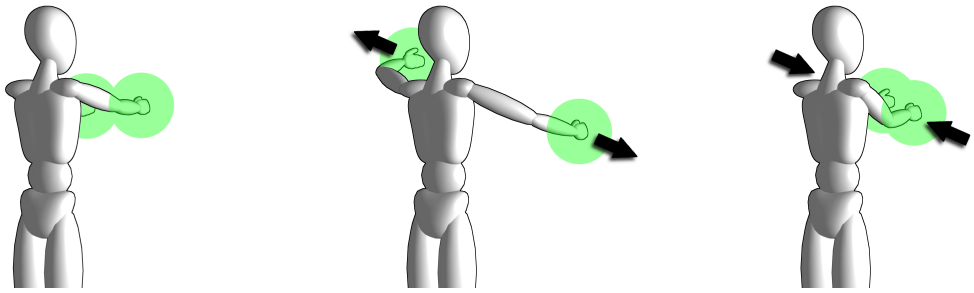
- *Plugin de selecció*: Per tal de fer ús del plugin de selecció, s'utilitza un únic braç de l'esquelet (esquerre o dret indistintament) mentre l'altre es troba en posició de relax. La idea en la que es basa el plugin és l'ús d'un raig de selecció definit mitjançant el vector establert entre la mà i l'espatlla del braç en qüestió. Gràcies a aquest raig, el visualitzador MovingVis, pot establir quin dels objectes de l'escena s'està seleccionant. Per tal de realitzar la confirmació de selecció, s'utilitza el gest de "Push". En cas de voler descartar la selecció es pot tornar a realitzar el gest de "Push" havent passat abans per la "zona neutra". La figura 8.11 mostra una representació visual dels gestos de selecció.



(a) Posició de relax (b) Seleccionar (c) Confirmar selecció

Figura 8.11: Gestos de selecció.

- *Plugin de manipulació*: Una vegada es troba seleccionat un objecte de l'escena, es pot utilitzar el plugin de manipulació. Aquest plugin utilitza les dues mans únicament quan es detecta el gest de "Push" en ambdues. Mentre ens trobem en aquesta situació, mitjançant la distància entre la mà esquerra i la mà dreta, es pot modificar la mida de l'objecte seleccionat (a més distància més gran i viceversa), simulant així, els gestos de "Pinch-In" i "Pinch-Out" utilitzats en molts dispositius (per exemple iPhone) per implementar el zoom. A la figura 8.12 es poden veure els gestos de "Push", "Pinch-In" i "Pinch-Out".



(a) "Push" (b) "Pinch-Out" (c) "Pinch-In"

Figura 8.12: Gestos de manipulació.

Part III

Resultats, planificació i conclusions

Capítol 9

Prova d'usabilitat de la interfície d'interacció

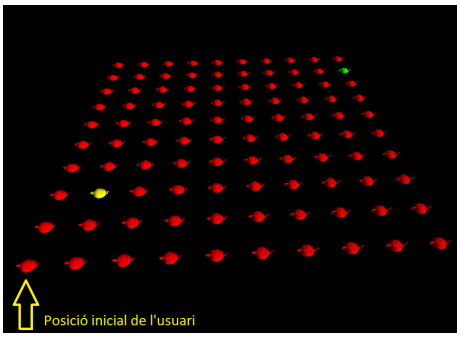
Donat que aquest projecte es basa en la construcció d'un sistema d'interacció basat en Kinect per a entorns de Realitat Virtual, la millor manera d'avaluar els resultats obtinguts és mesurant la usabilitat d'aquest sistema. Per tal de realitzar aquesta mesura, s'ha decidit fer una petita prova d'usabilitat del nostre sistema i comparar els resultats amb un altre sistema d'interacció diferent. Les següents seccions descriuen la prova i les valoracions obtingudes.

9.1 Descripció de l'experiment

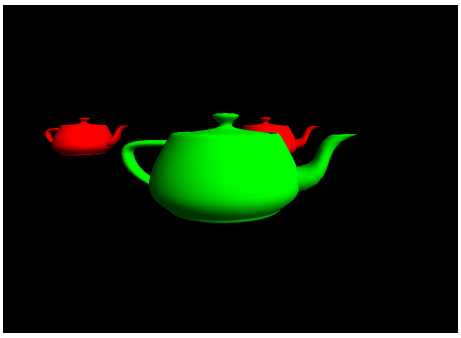
En aquest projecte s'ha dissenyat i desenvolupat un sistema d'interacció amb Kinect. Aquest sistema permet connectar una o múltiples càmeres Kinect de forma totalment transparent a les aplicacions externes que hi facin ús del mateix. L'interés d'aquesta prova d'usabilitat és veure si realment aquest sistema és més còmode i intuïtiu que d'altres ja existents.

Aquest experiment es realitza dins d'una escena virtual (3D) de proves formada per cent teteres distribuïdes en una quadrícula de 10x10 teteres. S'ha triat una escena 3D simple per tal d'evitar distreure a l'usuari. La tasca que el participant ha de fer dins d'aquesta escena és navegar des del punt d'inici fins a una tetera marcada de color verd, adoptar una vista específica d'aquesta tetera i navegar novament fins a una altra tetera marcada de color groc, a la qual, ha d'aproximar-se des d'amunt. A la figura 9.1 es pot observar la distribució de l'escena amb les teteres marcades i amb la posició inicial de l'usuari dins d'aquesta, la vista aproximada que cal adoptar de la tetera verda i l'aproximació final a la tetera groga.

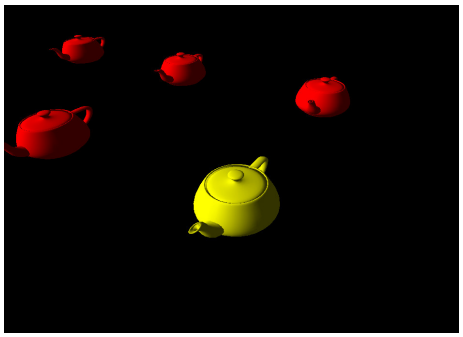
L'experiment es realitza mitjançant tres tècniques diferents:



(a) Distribució de l'escena de proves



(b) Vista a adoptar de la tetera verda



(c) Aproximació final a la tetera groga

Figura 9.1: Descripció visual de la tasca de l'experiment.

- Utilitzant un sistema d'interacció d'InterSense[17] basat en un sistema de seguiment per ultrasò (amb el receptor instal·lat sobre unes ulleres estereoscòpiques polaritzades) i un comandament de tipus Wanda (veure secció 9.2).
- Utilitzant el sistema d'interacció Kinect amb una única càmera.
- Utilitzant també el sistema d'interacció Kinect però, en aquest cas, amb dues càmeres.

El fet de comparar el sistema d'interacció amb Kinect amb el sistema de seguiment per ultrasò i comandament Wanda és que, aquest últim, és el mètode que s'utilitza actualment amb el sistema de Realitat Virtual semi-immersiu que es troba al Centre de Realitat Virtual de la UPC (on s'ha desenvolupat aquest projecte).

9.2 Muntatge de l'experiment

L'experiment es realitza sota un sistema de Realitat Virtual semi-immersiu basat en projecció (veure secció 2.2.2). S'utilitzen dos projectors i gràcies a dos filtres es polaritza la llum emesa que es projecta sobre una pantalla especial que manté aquesta polarització,

produint així, una visió estereoscòpica. Les mesures de la pantalla són de 2,7 metres d'amplada per 2 metres d'alçada i la imatge projectada té una resolució de 1024x768 píxels. Davant de la pantalla es disposa d'una àrea d'actuació de 2,7 per 3 metres.

La figura 9.2a mostra el receptor del sistema de seguiment per ultrasò instal·lat sobre unes ulleres estereoscòpiques polaritzades i el comandament de tipus Wanda, un comandament per a entorns de realitat virtual que disposa de diversos botons, un joystick i està equipat amb un acceleròmetre par captar el moviment i direcció de la mà de l'usuari. A l'experiment, s'utilitzen tres dels botons i l'acceleròmetre del comandament Wanda. El participant ha d'equipar-se amb ambdós elements (ulleres amb receptor i Wanda) per tal d'utilitzar aquest sistema.

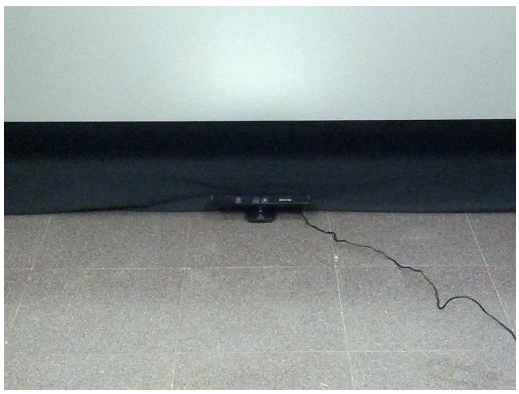
Pel que fa a la disposició de les càmeres del sistema d'interacció amb Kinect, a la prova amb una sola càmera, aquesta es col·loca sobre el terra i centrada en la pantalla, mentre que a la prova amb dues, aquestes es col·loquen una en cada cantó de la pantalla i a una alçada d'1,75 metres del terra gràcies a dos trípodes de càmeres fotogràfiques. Amb aquest sistema, l'usuari únicament ha d'equipar-se amb unes ulleres estereoscòpiques polaritzades. A la figura 9.2 es pot observar el muntatge efectuat amb una i amb dues càmeres Kinect.

Tot aquest muntatge està governat per tres ordinadors. De l'aplicació de Realitat Virtual i dels projectors s'encarrega un ordinador que disposa d'un processador Intel Core i7-3820 a 3,8 GHz de freqüència, 16 GB de memòria RAM i una tarja gràfica GeForce GTX 590. Un segon ordinador que controla el sistema de seguiment per ultrasò, el comandament Wanda i proporciona les dades mitjançant un servidor VRPN (veure secció 6.4), disposa d'un processador Intel Pentium IV a 2,5 GHz de freqüència, 512 MB de memòria RAM i una tarja gràfica Quadro4 900 XGL (que pel que fa a la tasca d'aquest ordinador, és irrellevant). Per últim, encara que en un muntatge definitiu no caldria, per tema de drivers s'ha utilitzat un altre ordinador per tal de manegar les càmeres Kinect i el servidor VRPN per fer accessibles les dades d'aquestes. Aquest ordinador és un portàtil amb un processador Intel Core i7-2637M a 2,8 GHz de freqüència, 6 GB de memòria RAM i una tarja gràfica GeForce GT 640M.

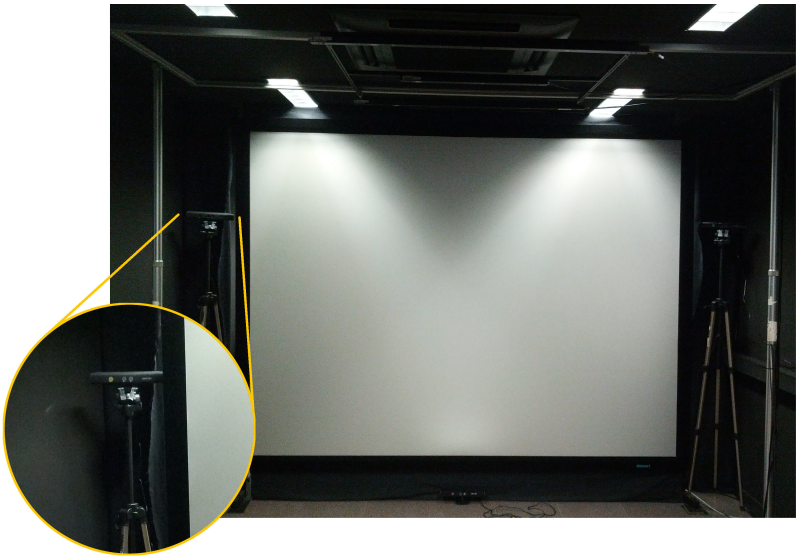
9.3 Participants

L'experiment ha estat realitzat per dotze voluntaris d'entre 20 i 35 anys. Nou dels participants (set homes i dues dones) són treballadors i investigadors del grup de recerca *MOVING* (ara *ViRVIG*[16]) on s'ha desenvolupat l'experiment. Aquests participants tenien experiència prèvia amb l'ús d'interfícies d'interacció en entorns de Realitat Virtual, sobretot amb el sistema de seguiment per ultrasò i comandament Wanda. Tot i així, cap d'ells havia provat encara el mètode de navegació de l'experiment.

(a) Receptor de seguiment per ultrasò i Wanda



(b) Muntatge amb un únic Kinect



(c) Muntatge amb dos Kinect

Figura 9.2: Dispositius d'interacció utilitzats a l'experiment.

Per tal de tenir una mostra més representativa de participants, els altres tres restants (un home i dues dones) no tenien cap experiència prèvia en sistemes de Realitat Virtual ni estaven habituats amb l'ús de comandaments per a videojocs.

9.4 Disseny de l'experiment

Per tal de realitzar una comparació de les tres tècniques d'interacció, cadascun dels participants realitza la mateixa tasca de navegació amb cadascuna de les tres. A fi d'anular l'efecte d'aprenentatge que es produiria seguint un mateix ordre, sis dels participants realitzen l'experiment utilitzant en primer lloc el sistema de seguiment per ultrasò i comandament Wanda i després el sistema d'interacció amb Kinect (primer amb una i després amb dues càmeres). Els altres sis participants realitzen l'experiment a l'inrevés, primer utilitzant el sistema d'interacció amb Kinect (primer amb una i després amb dues càmeres) i després el sistema de seguiment per ultrasò i comandament Wanda. L'elecció dels participants d'ambdós grups és totalment aleatòria.

Cada participant té un únic intent per realitzar la navegació sobre l'escena de proves. Abans però, se li permet jugar amb el sistema d'interacció (un màxim de tres minuts i amb cap tetera marcada) per tal que s'adapti i aprengui el seu funcionament.

Com a variables dependents de l'experiment, es mesura el temps que cada participant triga en realitzar la tasca de navegació i, al acabar l'experiment amb les tres tècniques d'interacció, se li fan una serie de preguntes per tal d'obtenir una valoració en escala Likert d'1 a 7 de les següents variables:

- *Comoditat*: La comoditat que sent el participant amb els elements físics que ha de manegar per interactuar.
- *Facilitat d'ús*: Com de fàcil li resulta al participant utilitzar el sistema d'interacció.
- *Facilitat d'aprenentatge*: Com de fàcil/ràpid aprèn el participant a utilitzar el sistema d'interacció.
- *Precisió*: La precisió que percep el participant per part del sistema d'interacció a l'hora de respondre a les seves accions.
- *Sensació de fatiga*: Grau de fatiga/mareig que sent el participant a l'hora de navegar per l'escena.
- *Sentit de l'orientació*: L'orientació que té el participant a l'hora de navegar per l'escena (si en tot moment sap la direcció que ha de seguir i on es troben les teteres marcades).

9.5 Hipòtesis

L'adaptació al sistema d'interacció i la manera de navegar per l'escena dependrà de cadascun dels participants, tot i així, donat que s'espera que el sistema amb Kinect (tant amb una càmera com amb dos) sigui més intuïtiu i senzill de fer servir, el temps mig del recorregut s'espera que sigui més petit amb el sistema amb Kinect que no pas amb el sistema de seguiment per ultrasò i comandament Wanda.

Pel que respecta a les variables dependents en escala Likert, s'han realitzat les següents hipòtesis:

- *Comoditat*: Donat que el sistema amb Kinect (tant amb una càmera com amb dos) funciona sense cables, s'espera que aquest sigui més còmode que el sistema de seguiment per ultrasò i comandament Wanda.
- *Facilitat d'ús*: El sistema amb Kinect (tant amb una càmera com amb dos) fa ús del propi cos de la persona i dels seus gestos, per aquest motiu s'espera que sigui més fàcil d'usar.
- *Facilitat d'aprenentatge*: Ja que ambdós sistemes no són massa complexes, no s'espera gaire diferència en aquest cas. Tot i així, per les mateixes raons que amb la variable anterior, pot ser que el sistema amb Kinect sigui més fàcil d'aprendre.
- *Precisió*: La precisió del sistema de seguiment per ultrasò i comandament Wanda s'espera que sigui màxima ja que funciona mitjançant cables i no es produeix pèrdua de senyal ni soroll. Pel que fa al sistema amb dues càmeres Kinect, s'espera que tingui una precisió suficientment alta, mentre que el sistema amb una sola càmera, s'espera una precisió baixa.
- *Sensació de fatiga*: Aquesta variable depèn molt del participant. S'espera que la sensació de fatiga sigui baixa amb les tres tècniques de l'experiment.
- *Sentit de l'orientació*: Igual que amb la variable anterior, el sentit de l'orientació depèn molt del participant. S'espera que amb les tres tècniques l'orientació del participant sigui alta.

9.6 Resultats

A la taula 9.1 es pot observar la mitjana aritmètica dels resultats obtinguts en cadascuna de les variables dependents (X) de l'experiment amb les tres tècniques utilitzades (on $\bar{X}_1$ es refereix al sistema de seguiment per ultrasò i comandament Wanda, $\bar{X}_2$ al sistema amb un únic Kinect i $\bar{X}_3$ al sistema amb dos Kinect).

X	$\bar{X}_1$	$\bar{X}_2$	$\bar{X}_3$
Temps de tasca (segons)	65,99	83,45	48,1
Comoditat (Likert 1-7)	3,25	6,33	6,75
Facilitat d'ús (Likert 1-7)	4,75	5,08	6,5
Facilitat d'aprenentatge (Likert 1-7)	4,5	6	6,08
Precisió (Likert 1-7)	5,92	2,83	6,08
Sensació de fatiga (Likert 1-7)	1,42	1,75	1,08
Sentit de l'orientació (Likert 1-7)	5,33	5,25	6,25

Taula 9.1: Mitjana aritmètica dels resultats de l'experiment.

Per tal de tenir una idea més detallada dels resultats de temps, la figura 9.3 mostra un diagrama de caixes, basat en quartils, on es pot observar la mediana i la distribució dels temps de tasca de cadascuna de les tres tècniques (on 1 es refereix al sistema de seguiment per ultrasò i comandament Wanda, 2 al sistema amb un únic Kinect i 2 al sistema amb dos Kinect).

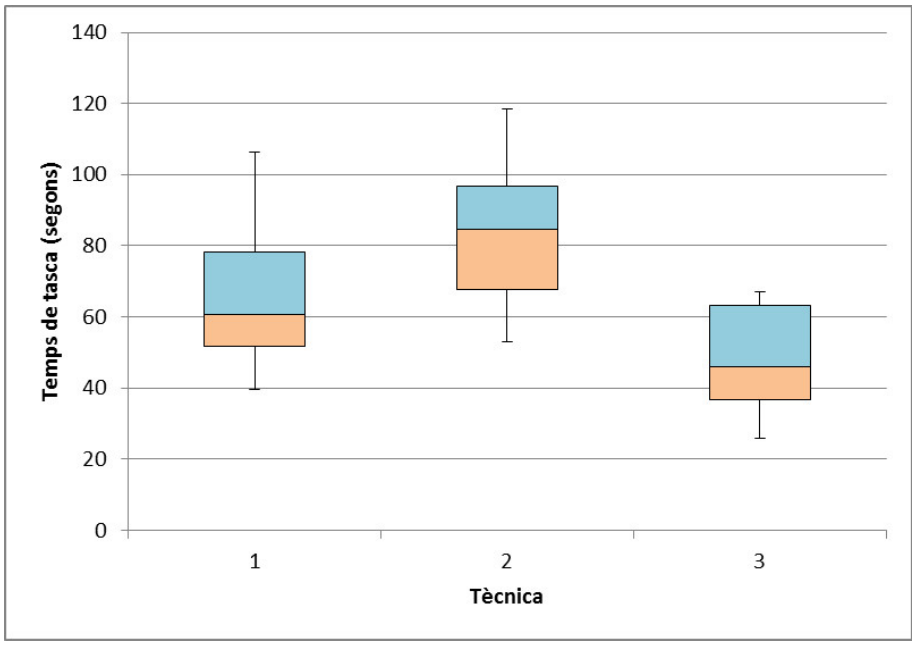


Figura 9.3: Diagrama de caixes dels temps de tasca de l'experiment.

Donada la mida reduïda de la mostra de participants, no es pot assegurar que les variacions obtingudes en les mitjanes aritmètiques de cadascuna de les variables siguin estadísticament significatives, és a dir, poden ser degudes simplement a la aleatorietat de la mostra de participants i no pas a la tècnica utilitzada. Per tal d'esbrinar quines de

les diferències entre mitjanes aritmètiques es deuen a la tècnica utilitzada, es realitza una anàlisi de variància per cadascuna de les variables dependents de l'experiment. Per les dades de temps s'aplica un ANOVA clàssic i per les variables tipus Likert s'utilitza el test de Friedman (típic en aquests tipus de variables ja què l'escala no és uniforme i, a diferència d'ANOVA, no té les assumpcions de què les dades han de seguir una distribució normal i que els grups han de tenir iguals variàncies).

La taula 9.2 mostra els resultats de l'ANOVA aplicada a la variable de *Temps de tasca*. Com es pot observar, amb un interval de confiança del 95%, el resultat d'ANOVA ($P - value$) és menor que 0,05, i per tant, es pot establir que les diferències entre les mitjanes de temps de les tres tècniques són estadísticament significatives (almenys dos d'elles). Els resultats del test Tukey HSD ens indiquen si aquestes diferències són estadísticament significatives per a cada parella de tècniques ($P - value < 0,05$). En aquest cas s'estableix que les diferències obtingudes entre les tres tècniques utilitzades sí són estadísticament significatives (T_1, T_2 i T_3 es refereixen al sistema de seguiment per ultrasò i comandament Wanda, sistema amb un únic Kinect i sistema amb dos Kinect respectivament).

	$\boldsymbol{F - value}$	$\boldsymbol{P - value}$
ANOVA	22,64	0,0001

Tukey HSD ($\boldsymbol{P - value}$)	$\boldsymbol{T_2}$	$\boldsymbol{T_3}$
$\boldsymbol{T_1}$	0,01	0,01
$\boldsymbol{T_2}$		0,01

Taula 9.2: Resultats de l'ANOVA del Temps de tasca.

A la taula 9.3 es poden observar els resultats ($P - value$) obtinguts al realitzar el test de Friedman amb un interval de confiança del 95% en cadascuna de les variables dependents de tipus Likert (X). A més, en els casos que aquest test indica que la diferència entre mitjanes és estadísticament significativa ($P - value < 0,05$), es realitza una comparació múltiple entre totes les parelles de tècniques per tal d'establir la rellevància de la diferència entre ambdues. A la taula es pot veure el resultat de la comparació múltiple en forma d'afirmació (✓) o negació (✗) segons si la diferència és o no estadísticament significativa (de la mateixa forma que la taula anterior, T_1, T_2 i T_3 es refereixen al sistema de seguiment per ultrasò i comandament Wanda, sistema amb un únic Kinect i sistema amb dos Kinect respectivament).

Veient totes les dades recollides i el diagrama de caixes del temps de tasca, podem concloure que:

- Els participants han completat la tasca de navegació amb el sistema amb dos Kinect considerablement més ràpid que amb el sistema de seguiment per ultrasò i comandament Wanda. Així, la hipòtesi sobre aquest aspecte es pot considerar correcta

X	$P-value$	$T_1 \sim T_2$	$T_1 \sim T_3$	$T_2 \sim T_3$
Comoditat	0.000021	✓	✓	✗
Facilitat d'ús	0.001805	✗	✓	✗
Facilitat d'aprenentatge	0.001312	✓	✓	✗
Precisió	0.000031	✓	✗	✓
Sensació de fatiga	0.165300	✗	✗	✗
Sentit de l'orientació	0.041730	✗	✗	✗

Taula 9.3: Resultats del test de Friedman amb comparació múltiple.

encara que, en el cas d'un únic Kinect, els resultats hagin estat pitjors. Aquest fet és degut a la falta de precisió que es té amb una única càmera ja que l'usuari, a mesura que realitza gestos, provoca oclusions amb si mateix i la càmera perd de vista algunes articulacions de l'esquelet. L'avantatge del sistema amb dos càmeres és que s'augmenta la precisió, donat que, el que no és capaç de veure una càmera, ho veu l'altre.

- Com es suposava en les hipòtesis, la comoditat que ofereix el sistema amb Kinect és considerablement superior que amb el sistema de seguiment per ultrasò i comandament Wanda. Mentre que amb el primer sistema l'usuari únicament ha d'equipar-se amb unes ulleres estereoscòpiques polaritzades, amb el segon, l'usuari ha de suportar a més els cables tant del receptor de seguiment com del comandament Wanda.

- A diferència del sistema amb dos Kinect, que té una facilitat d'ús molt elevada i compleix la hipòtesi inicial, el sistema amb un únic Kinect dóna uns resultats similars als obtinguts amb el sistema de seguiment per ultrasò i comandament Wanda. Això és degut a la poca precisió que ofereix una càmera, fet que dificulta la interacció de l'usuari amb l'escena considerablement. Alguns dels participants han trobat que la facilitat d'ús del sistema amb Kinect és superior gràcies a què els gestos són prou intuïtius i que es disposa de la finestra d'ajuda, que en tot moment proporciona a l'usuari informació del que està passant.

- A tots els participants els ha sigut relativament fàcil aprendre el funcionament de tots dos sistemes. La diferència obtinguda entre el sistema de seguiment per ultrasò i comandament Wanda respecte al sistema amb Kinect, es deu a què alguns dels participants han trobat una mica més fàcil d'aprendre el funcionament amb aquest últim, ja fos per no estar habituat als comandaments de videojocs o per què simplement trobaven més intuïtiu el mètode d'interacció basat en gestos.

- Com es podia preveure en les hipòtesis, el sistema de seguiment per ultrasò i comandament Wanda proporciona una precisió molt elevada, mentre que la que proporciona el sistema amb un únic Kinect, és molt baixa. Tot i que el resultat amb el sistema amb dues càmeres és lleugerament superior a l'obtingut amb la primera tècnica, veient els resultats del test de Friedman s'estableix que aquesta diferència

no és estadísticament significativa, i per tant, es podria dir que ambdós sistemes proporcionen una precisió similar.

- La sensació de fatiga experimentada pels participants és nul·la o gairebé nul·la amb les tres tècniques. El fet de què sigui lleugerament superior amb el sistema d'un únic Kinect es pot deure a la poca precisió que aporta. Aquest fet provocava en alguns participants una lleugera sensació de mareig degut a què el sistema no responia correctament amb les seves accions o que el moviment de navegació era a batzegades.

- En quant al sentit de l'orientació que tenien els participants, no es pot concloure que hi hagi un sistema millor que l'altre. En tots els casos els participants sabien cap on havien d'anar i on es trobaven les teteres marcades. La petita variació que s'obté amb el sistema amb dos Kinect es pot deure simplement a la comoditat i facilitat d'ús que els participants experimentaven amb aquest.

Com a valoració addicional, se'ls va preguntar als participants quin dels sistemes els hi agradava més. Dels dotze participants, onze van respondre que preferien el sistema amb dues càmeres Kinect, mentre que únicament un, va respondre que preferia el sistema de seguiment per ultrasò i comandament Wanda.

Capítol 10

Planificació i anàlisi econòmica

El següent capítol presenta la planificació que s'ha dut a terme durant la realització del projecte i una anàlisi econòmica del cost que hauria suposat la realització d'aquest fora de l'àmbit d'un projecte final de carrera.

10.1 Planificació

Tot i que la idea del projecte va sorgir al Juny o Juliol de 2011, no va ser fins finals de 2011 i principis de 2012 quan es va començar amb la seva realització. A més, durant aquest període, l'autor ha estat acabant algunes de les assignatures finals de la carrera i treballant com a becari en projectes de recerca al grup *MOVING* (ara *ViRVIG*[16]). Aquest motiu ha propiciat que la realització del projecte hagi sigut lenta i hagi patit algunes pauses.

La figura 10.1 mostra un diagrama de Gantt en el qual es pot veure la planificació que s'ha dut a terme. Seguidament, per tal d'entendre millor les tasques que s'han realitzat en cada moment, es fa una breu descripció per mes:

- ***Gener de 2012***: Es comença la cerca d'informació sobre la millor manera d'utilitzar múltiples càmeres Kinect en un mateix ordinador i sobre treballs previs similars. Es realitza l'elecció del driver de Kinect a utilitzar en el desenvolupament del projecte (i per tant, el sistema operatiu que s'utilitzarà).

- ***Febrer de 2012***: A la primera meitat del mes s'estableix l'esquema, títol i principals objectius (a grans trets) del projecte. Posteriorment, es fa una petita anàlisi i disseny de l'arquitectura general del sistema. Durant la segona meitat es comença l'aprenentatge de l'estructura i funcionament del SDK de Microsoft per treballar amb Kinect, es crea un primer esquelet de l'aplicació MultiKinect per tal de començar a fer proves amb el driver de Kinect i s'obté una primera versió funcional amb un únic Kinect.

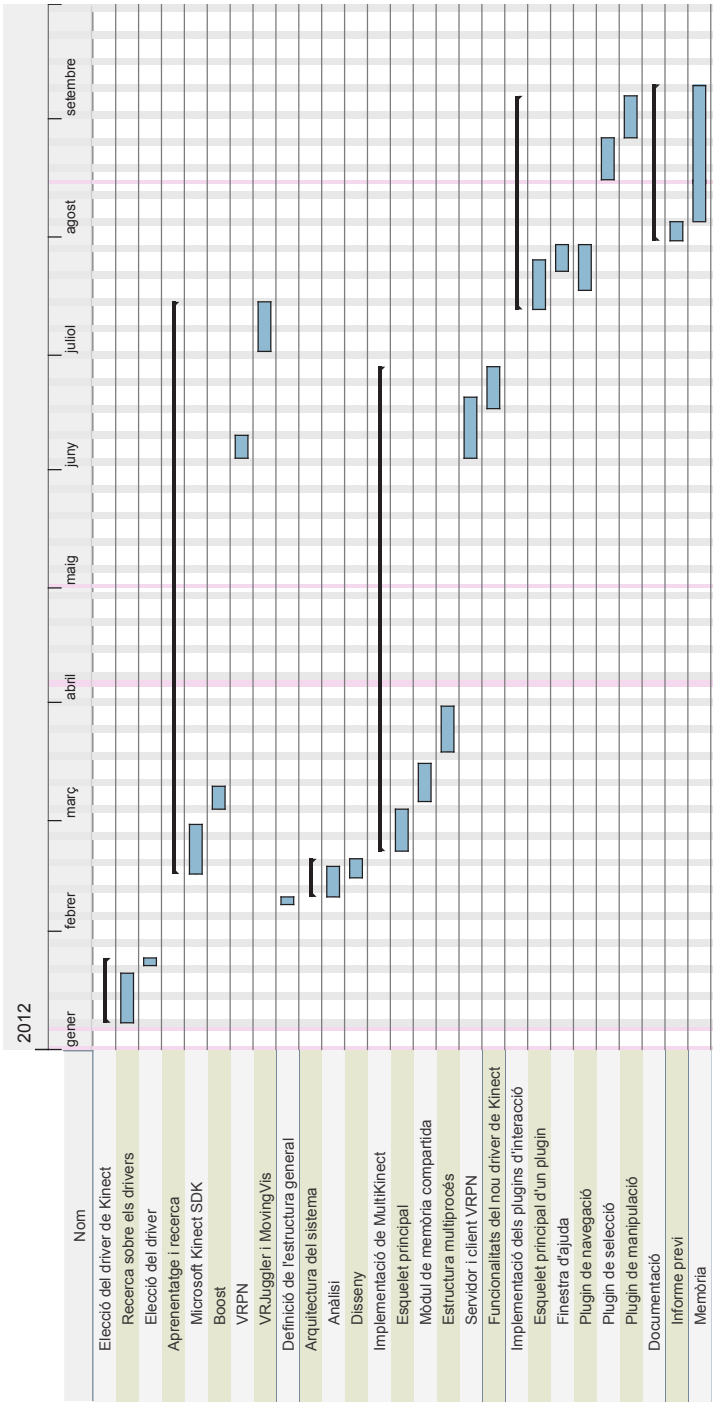


Figura 10.1: Diagrama de Gantt de la planificació del projecte.

- ***Març de 2012***: Es comença a afrontar l'objectiu d'utilitzar múltiples càmeres Kinect. Durant la primera quinzena es realitza la cerca de la millor manera d'implementar el mòdul de memòria compartida i es decideix utilitzar la llibreria Boost. S'aconsegueix una versió funcional del mòdul de memòria compartida entre processos. La segona meitat del mes es dedica a definir l'estructura de MultiKinect per tal de tenir processos de tipus "master" i de tipus "slave". S'obté una segona versió funcional de MultiKinect amb múltiples càmeres.

- ***Juny de 2012***: Durant la primera meitat d'aquest mes s'integra la llibreria VRPN i s'elaboren el servidor i client VRPN utilitzats per l'enviament i recepció dels esquelets detectats per Kinect. A la segona meitat s'integra la nova versió del driver de Kinect (versió 1.5) i s'afegeixen a l'aplicació les noves funcionalitats que aquest permet dur a terme (modes "a prop" i "assegut"). S'obté una tercera versió de MultiKinect que utilitza VRPN.

- ***Juliol de 2012***: Durant aquest mes es procedeix amb la creació dels esquelets dels plugins d'interacció per a MovingVis. S'aprèn el funcionament mínim necessari tant dels fitxers de configuració de VRJuggler com del visualitzador. S'obté una versió funcional del plugin de navegació.

- ***Agost de 2012***: Al mateix temps que es continua amb la implementació dels plugins de selecció i manipulació de MovingVis, es confecciona la documentació del projecte.

- ***Setembre de 2012***: Durant les dues primeres setmanes d'aquest mes es finalitza la documentació del projecte i es realitzen els últims retocs de la implementació dels plugins d'interacció del visualitzador. S'obtenen les versions definitives (en l'àmbit d'aquest projecte) tant de MultiKinect com dels plugins.

10.2 Anàlisi econòmica

Aquesta secció presenta una estimació econòmica del cost que hauria tingut aquest projecte si no hagués estat desenvolupat en l'àmbit d'un projecte final de carrera. Aquesta anàlisi té en compte tant el cost de personal com el cost d'equipaments necessaris i suposa que cal fer la instal·lació d'un sistema de Realitat Virtual.

10.2.1 Cost de personal

Per estimar el cost de personal, s'han distingit els rols d'analista i de programador. S'ha suposat que l'analista és extern (una empresa externa o autònom) i per tant, el cost de la contractació depèn del nombre d'hores treballades. El programador es suposa que és un treballador en nòmina amb una jornada de 4 hores diàries. El preu estimat per hora treballada per part de l'analista, ha estat de 50 € i en el cas del programador, de 20 €.

Tasca desenvolupada	**Nombre d'hores**	**Encarregat**	**Cost**
Anàlisi del sistema	32	Analista	1.600 €
Disseny del sistema	20	Analista	1.000 €
MultiKinect: Esquelet principal	44	Programador	880 €
MultiKinect: Mòdul de memòria compartida	40	Programador	800 €
MultiKinect: Estructura multi-procés	48	Programador	960 €
MultiKinect: Servidor i client VRPN	64	Programador	1.280 €
MultiKinect: Funcionalitats del nou driver	44	Programador	880 €
Plugins: Esquelet principal	52	Programador	1.040 €
Plugins: Finestra d'ajuda	28	Programador	560 €
Plugins: Plugin de navegació	48	Programador	960 €
Plugins: Plugin de selecció	44	Programador	880 €
Plugins: Plugin de manipulació	44	Programador	880 €
Documentació	160	Programador	3.200 €
Total	*668*		*14.920 €*

Rol	**Cost**
Analista	2.600 €
Programador	12.320 €
Total	*14.920 €*

Taula 10.1: Cost de personal.

10.2.2 Cost d'equipaments

L'equipament hardware necessari per dur a terme aquest projecte es suposa que consta d'un ordinador de gama mitjana (perquè el programador pugui dur a terme les tasques de desenvolupament i documentació del projecte), dues càmeres Kinect i un sistema de Realitat Virtual (es considera que és un sistema CAVE similar al disposat al Centre de Realitat Virtual del departament de Llenguatges i Sistemes Informàtics de la UPC). En quant al software, seria necessari pagar la llicència de *Microsoft Windows 7*, sistema operatiu on s'ha desenvolupat el projecte. La resta d'eines poden ser utilitzades de forma gratuïta i per tant, no suposen un càrrec addicional al cost del projecte. A més, s'ha de tenir un lloc on el programador pugui treballar, la qual cosa suposa uns costos de lloguer d'oficina. Per calcular el temps del lloguer s'ha suposat una jornada laboral de 8 hores i 5 dies laborables a la setmana. Això resulta en 4 mesos i uns quants dies més, per tant, 5 mesos de lloguer.

Hardware	**Quantitat**	**Preu per unitat**	**Cost**
Ordinador gama mitjana	1	600 €	600 €
Kinect	2	100 €	200 €
CAVE: Pantalles	1	32.100 €	32.100 €
CAVE: Estructura	1	1.200 €	1.200 €
CAVE: Ordinador gama alta	10	2.600 €	26.000 €
CAVE: Projector	40	590 €	23.600 €
CAVE: Parell de filtres	20	670 €	13.400 €
CAVE: Càmera de calibrat	4	430 €	1.720 €
CAVE: Cablejat i altres	1	500 €	500 €
Total			*99.320 €*

Software	**Quantitat**	**Preu per unitat**	**Cost**
Microsoft Windows 7 Professional	1	285 €	285 €
Total			*285 €*

Lloguer	**Mesos**	**Preu per mes**	**Cost**
Oficina	5	600 €	3.000 €
Total			*3.000 €*

Equipament	**Cost**
Hardware	99.320 €
Software	285 €
Oficina	3.000 €
Total	*102.605 €*

Taula 10.2: Cost d'equipaments.

10.2.3 Cost total del projecte

Fent la suma del cost de personal més el cost d'equipaments, resulta que el cost total del projecte és aproximadament de 118.000 €. En cas que es disposés prèviament del sistema de Realitat Virtual, i únicament volguéssim instal·lar el sistema d'interacció mitjançant càmeres Kinect, el cost es reduiria fins arribar a uns 20.000 € de cost total.

Concepte	**Cost**
Personal	14.920 €
Equipaments	102.605 €
Total	*117.525 €*

Taula 10.3: Cost total del projecte.

Capítol 11

Conclusions

En aquest capítol es relaten les conclusions generals del projecte i es fa una petita anàlisi dels aspectes i conceptes de la carrera que s'han aplicat durant la realització del mateix.

11.1 Conclusions generals del projecte

El primer objectiu del projecte era la creació d'una aplicació per manegar una o múltiples càmeres Kinect per tal d'utilitzar les seves dades de detecció d'esquelets en altres aplicacions. Aquest objectiu s'ha assolit amb el disseny i la implementació d'un programa que és capaç d'obtenir les dades d'imatge a color, mapa de profunditat i detecció d'usuaris, de múltiples càmeres Kinect alhora i fer accessibles, mitjançant VRPN (veure secció 6.4), una representació en forma d'esquelet de cadascun d'aquests usuaris. Per tal de resoldre els problemes relacionats amb aquest objectiu (veure el capítol 4) s'ha hagut de procedir amb una solució consistent en l'ús de diversos processos per manegar les dades (un per cada Kinect i un més per centralitzar les dades) i d'un mòdul de memòria compartida per tal que aquests es comuniquin (veure secció 8.1.3). A més, l'aplicació proporciona a l'usuari una interfície fàcil i pràctica tant per configurar diversos paràmetres de les càmeres com per visualitzar les dades proporcionades per aquestes (veure secció 8.1.1).

El segon objectiu del projecte era la construcció d'una interfície d'interacció basada en les dades dels Kinect per a un visualitzador d'entorns virtuals. Per tal de complir aquest objectiu, s'ha realitzat el disseny i la implementació de tres plugins d'interacció: un de navegació, un de selecció i un altre de manipulació. El funcionament conjunt d'aquests tres plugins proporciona a l'usuari la capacitat de moure la càmera, seleccionar objectes i modificar la seva mida, dins d'una escena 3D virtual. Els problemes de com utilitzar les dades dels Kinect per a realitzar la interacció, s'han resolt gràcies a un mecanisme de detecció de gestos simples a partir de la representació de l'esquelet de l'usuari (veure secció 8.2.3). A més, per tal de fer més comprensible la interacció, s'ha afegit a tots els plugins una finestra d'ajuda la qual mostra l'esquelet detectat i l'acció que s'està realitzant sobre

l'escena virtual (veure secció 8.2.1).

En definitiva, es pot afirmar que s'ha construït un sistema d'interacció basat en Kinect per a un sistema de Realitat Virtual. Tot i així, l'aplicació de manegament de càmeres Kinect pot ser utilitzada per qualsevol tipus d'aplicació externa, per tant, es podrien construir altres tipus de sistemes d'interacció o utilitzar les dades de Kinect amb propòsits diferents.

11.2 Conceptes de la carrera aplicats

A la realització d'aquest projecte han sigut aplicats diferents aspectes i conceptes apresos durant la carrera. Evidentment, la base de coneixements sobre programació, s'ha obtingut de les diferents assignatures de programació i algorísmia de la carrera. Alguns dels conceptes adquirits en les assignatures d'enginyeria del software han estat útils per realitzar l'anàlisi i disseny de la solució. Els coneixements obtinguts de les següents assignatures però, poden ser més rellevants que la resta en l'àmbit d'aquest projecte:

- *Visualització i Interacció Gràfica (VIG) i Visualització Avançada (VA)*: Aplicades en el disseny de les interfícies gràfiques i la visualització 2D i 3D.
- *Geometria Computacional (GEOC) i Física orientada a la Modelització i l'Animació Realistes (FMAR)*: Aplicades en el manegament i utilització d'algunes estructures geomètriques.
- *Sistemes Operatius (SO)*: Aplicada en l'ús de processos i fils d'execució, així com, en el manegament d'estructures de memòria compartida.

Capítol 12

Treball futur

Encara que els objectius del projecte s'hagin assolit, sempre hi ha elements o funcionalitats millorables. Aquest capítol presenta algunes idees que es podrien dur a terme per tal de millorar les funcionalitats de l'aplicació MultiKinect o la interfície d'interacció construïda per al visualitzador MovingVis.

12.1 Afegir accessibilitat a les dades de color i profunditat de Kinect per part d'aplicacions externes

En la realització d'aquest projecte s'han fet accessibles a través de la xarxa les dades dels esquelets reconstruïts per Kinect. D'aquesta manera qualsevol aplicació externa pot utilitzar aquestes dades de forma senzilla per diferents tipus de tasques. Per exemple en aquest cas, construir una interfície d'interacció basada en la detecció de gestos dels esquelets detectats.

Una de les millores de l'aplicació MultiKinect seria la d'utilitzar la mateixa política amb les dades d'imatge a color i mapa de profunditat. Donat que MultiKinect ja obté aquestes dades de les càmeres Kinect, només caldria afegir un mecanisme per tal de fer accessibles aquestes dades a través de la xarxa.

Una solució possible a aquesta millora seria l'ús d'un servidor TCP que obrís una connexió IP accessible des de qualsevol aplicació. Les dades d'imatge a color i mapa de profunditat es podrien enviar frame a frame a través de la connexió mitjançant la utilització d'un format de compressió d'imatges adequat, per exemple JPEG o MPEG o qualsevol de les seves variants.

Afegint aquesta millora a MultiKinect es donaria més versatilitat a l'ús de les dades de Kinect. D'aquesta manera cadascuna de les aplicacions externes tindrien la capacitat de millorar les seves funcionalitats gràcies a aquestes dades.

12.2 Utilitzar el reconeixedor de veu de Kinect com a comandament

Una de les deficiències d'utilitzar un sistema d'interacció basat en Kinect és la mancança d'un element activador que ens permetés indicar al sistema l'activació, desactivació o canvi d'una funcionalitat.

Una possible millora de l'aplicació MultiKinect podria ser l'ús del sistema reconeixedor de veu del que disposa Kinect. Mitjançant aquest sistema es podrien detectar certes comandes i utilitzar-les a les aplicacions externes. Es podria aprofitar el protocol VRPN per tal de proporcionar aquestes comandes a través de la xarxa simulant esdeveniments o diferents estats d'un botó o comandament.

Aquesta millora afegiria versatilitat a l'hora d'implementar certes funcionalitats. Un exemple aplicat al cas d'interacció en entorns virtuals podria ser permetre a l'usuari canviar el mode d'interacció de forma senzilla o activar i desactivar la detecció de gestos.

12.3 Millorar la interfície gràfica de MultiKinect

Ara per ara, la interfície gràfica de MultiKinect és diferent segons si el procés és de tipus "master" o "slave". A més, cadascun dels processos "slave" té la seva pròpia interfície.

Una millora en aquest camp seria la unió de totes aquestes interfícies en una única, de manera que, es permetés a l'usuari mostrar i/o amagar els elements que desitgés. També hauria de permetre a l'usuari activar i desactivar de forma independent qualsevol de les càmeres Kinect connectades (ara per ara s'utilitzen totes les que estiguin connectades a no ser que es tanqui el procés "slave" que les controla) i modificar la configuració de qualsevol d'elles de forma totalment centralitzada. A més, seria adient permetre modificar des de la interfície alguna de les opcions de configuració del sistema, com per exemple, la direcció de xarxa dels trackers dels esquelets o el mode d'execució.

Aquesta millora proporcionaria una interfície unificada que faria més còmode la configuració de les càmeres Kinect i la visualització de les seves dades.

12.4 Millorar la detecció de gestos

Una de les parts més importants dels plugins d'interacció de MovingVis, és sens dubte la detecció de gestos a partir de l'esquelet de l'usuari. Els gestos definits als plugins són simples i es basen en la comparació de distancies i en les direccions definides pels vectors

compresos entre dues articulacions.

Una possible millora d'aquest aspecte podria consistir en fer més amplia la gama de gestos i més robusta la seva detecció. Això influiria de forma directa en la facilitat i comoditat de la interfície. Per fer efectiva aquesta millora s'haurien d'estudiar gestos més complexes i la forma de detectar-los. Un exemple més complex de detecció de gestos podria ser el control de les trajectòries definides en el temps per part d'algunes articulacions i comparant, aquestes trajectòries, amb altres precalculades que definissin certs gestos.

A més, l'aplicació MultiKinect es podria millorar per tal que detectés alguns gestos simples i els fes accessibles a les aplicacions externes a través de VRPN. De forma similar al cas del reconeixement de veu, aquests gestos es podrien fer accessibles simulant esdeveniments o estats d'un botó o comandament. D'aquesta manera es simplificaria la feina a fer per part d'aquestes aplicacions.

Bibliografia

[1] Sebastien Kunz. State of Virtual Reality, December 2007. URL `http://cb.nowan.net/blog/state-of-vr/`. Citat a les pàgines 15 (2) i 18.

[2] Virtual reality. URL `http://www.vrs.org.uk/virtual-reality/what-is-virtual-reality.html`. Citat a la pàgina 16.

[3] Terrence Fernando, Norman Murray, Gilles Gautier, Sas Mihindu (USAL) Konstantinos Loupos (ICCS), Philippe Gravez (CEA), Hilko Hoffmann (FhG-IAO), Johan Blondelle (Barco), Sergio Di Marca (IDG), Marco Fontana (PERCO), Wojciech Cellary (PUE). State-of-the-art in VR. Technical report, INTUITION Virtual Reality, 2006. Citat a les pàgines 16, 17 (2) , 18 (2) , 19, 20, 21, i 109.

[4] Evan A. Suma, Belinda Lange, Skip Rizzo, David Krum, Mark Bolas. Flexible Action and Articulated Skeleton Toolkit (FAAST). University of Southern California, 2011. URL `http://projects.ict.usc.edu/mxr/faast/`. Citat a les pàgines 32 i 111.

[5] Iason Oikonomidis, Nikolaos Kyriazis, Antonis A. Argyros. Efficient Model-based 3D Tracking of Hand Articulations using Kinect. 2011. Citat a la pàgina 33.

[6] Miguel Ribo, Axel Pinz, Anton L. Fuhrmann. A new optical tracking system for virtual and augmented reality applications. 2001. Citat a la pàgina 33.

[7] Doug A. Bowman, David Koller, Larry F. Hodges. Travel in Immersive Virtual Environments: An Evaluation of Viewpoint Motion Control Techniques. 1997. Citat a la pàgina 34.

[8] Luca Chittaro, Stefano Burigat. 3D location-pointing as navigation aid in Virtual Environments. 2004. Citat a la pàgina 34.

[9] OpenKinect. OpenKinect. URL `http://openkinect.org/wiki/Main_Page`. Citat a la pàgina 43.

[10] OpenNI TM. OpenNI. URL `http://openni.org/`. Citat a la pàgina 43.

[11] Russell M. Taylor II, Thomas C. Hudson, Adam Seeger, Hans Weber, Jeffrey Juliano, Aron T. Helser. VRPN: A Device-Independent, Network-Transparent VR Peripheral System. *VRST 2001 Conference*, 2001. Citat a la pàgina 44.

[12] Department of Computer Science, University of North Carolina. VRPN: Virtual Reality Peripheral Network. URL http://www.cs.unc.edu/Research/vrpn/. Citat a la pàgina 44.

[13] wxWidgets. wxWidgets. Cross-Platform GUI Library. URL http://www.wxwidgets.org/. Citat a la pàgina 45.

[14] Nokia. Qt. URL http://qt.nokia.com/. Citat a la pàgina 45.

[15] VRJuggler. VRJuggler. URL http://vrjuggler.org/. Citat a la pàgina 46.

[16] ViRVIG. ViRVIG. Visualització, Realitat Virtual i interacció Gràfica. URL http://www.virvig.eu/. Citat a les pàgines 47, 87, i 95.

[17] InterSense, LLC. InterSense. URL http://www.intersense.com/. Citat a la pàgina 86.

[18] Microsoft Corporation. Chapter 3 Developing Phase: Process and Thread Management, May 2006. URL http://technet.microsoft.com/en-us/library/bb497007.aspx. No citat.

[19] Microsoft Research. *Kinect for Windows SDK beta. Programing Guide.* Microsoft Corporation, Beta 1 Draft Version 1.1 edition, July 2011. No citat.

[20] Rob Relyea. Kinect for Windows. Code Migration from Beta2 to v1.0, February 2012. URL http://robrelyea.wordpress.com/2012/02/01/k4w-code-migration-from-beta2-to-v1-0-managed/. No citat.

Índex de figures

Índex de taules