Títol: ***Manipulació i interacció en entorns de realitat virtual d'una aplicació mèdica***
Volum: ***1/1***

Alumne: ***Xavier de la Fuente Caballé***

Director/Ponent: ***Isabel Navazo Álvaro***

Departament: ***LSI***

Data: ***01/2009***

DADES DEL PROJECTE

Títol del Projecte: Manipulació i interacció en entorns de realitat virtual d'una aplicació mèdica

Nom de l'estudiant: Xavier de la Fuente Caballé
Titulació: Enginyeria Informàtica
Crèdits: 37.5
Director/Ponent: Isabel Navazo Álvaro
Departament: LSI

MEMBRES DEL TRIBUNAL *(nom i signatura)*

President: Carlos Antonio Andújar Gran

Vocal: Pilar Muñoz Gràcia

Secretari: Isabel Navazo Álvaro

QUALIFICACIÓ

Qualificació numèrica:
Qualificació descriptiva:

Data:

Índex

Índex de figures

1 Introducció

La major part d'aplicacions amb escenes tridimensionals (3D), com els videojocs, utilitzen monitors i dispositius d'entrada convencionals. Actualment és molt difícil imaginar la interacció amb cap aplicació informàtica sense una interfície d'usuari o GUI (Graphical User Interface), que ens permet usar l'aplicació mitjançant perifèrics d'entrada/sortida, com el teclat o el ratolí. Les interfícies més exteses són les del tipus WIMP (Windows, Icons, Mouse, Pointing device), en que el ratolí controla la posició d'un cursor, i es presenta l'informació organitzada en finestres i representada amb icones. Aquest tipus d'interfície es caracteritza per la facilitat d'ús i eficiència.

Malgrat utilitzar escenes 3D, aquestes aplicacions s'anomenen (en el marc d'aquesta memòria i en gran part de la bibliografia) 2D, ja que l'usuari veu i interacciona amb una projecció de l'escena en una pantalla 2D.

En contraposició, les aplicacions de realitat virtual utilitzen una projecció estèreo de les escenes que permeten tenir una visualització 3D com succeeix en el món real, però requereixen perifèrics específics de sortida. En aquests entorns 3D, l'usuari té més llibertat de moviment i els perifèrics tradicionals, com estan pensats per a dues dimensions, resulten poc intuïtius. Una aplicació de realitat virtual hauria de deixar que l'usuari dipositès un objecte a qualsevol lloc en un espai 3D i amb qualsevol orientació, una tasca per a la qual un ratolí 2D és inadeqüat. Per això, aquests nous sistemes necessiten noves tècniques, dispositius i metàfores d'interacció. Algún d'aquest components pot ser un simple refinament dels actuals, d'altres han de ser dissenyats des de zero. Tot això ens permetria aconseguir una interfície intuïtiva i natural entre un humà i un ambient de treball generat per una màquina. S'entén per interfície intuïtiva entre un home i una màquina aquella que requereix poc entrenament i ofereix un estil de treball més semblant a l'usat per l'èsser humà per interactuar amb ambients i objectes en la seva vida diària. En altres paraules, els humans interactuen amb els elements de l'aplicació mirant, sostenint, manipulant, parlant, escoltant i movent, fent servir les seves habilitats naturals tant com sigui possible [1]. Per això, la interacció ideal hauria de ser implícita com en el món real, sense la necessitat

d'usar perifèrics d'entrada que requereixin un aprenentatge.

L'objectiu principal del projecte consisteix en migrar una aplicació de sobretaula que permet visualitzar reconstruccions tridimensionals d'estructrues anatòmiques a un entorn de realitat virtual. S'implementaran i avaluaran diferents mètodes de manipulació i interacció que siguin intuïtius i fàcils d'usar en entorns de realitat virtual; també s'estudiarà la viabilitat i usabilitat de l'aplicació per a ser utilitzada en entorns mèdics.

1.1 Organització de la memòria

En el segon capítol es presenta una descripció teòrica dels conceptes i algorismes necessaris per enmarcar els objectius del projecte, presentats en el tercer capítol junt amb els requeriments de l'aplicació.

En el quart capítol es comenta el software que utilitza l'aplicació desenvolupada i algunes alternatives estudiades amb les mateixes funcionalitats.

En el cinquè capítol es presenta el disseny i la implementació de la nova aplicació de realitat virtual desenvolupada. S'explica l'adaptació a la realitat virtual, l'arquitectura, els algorismes de manipulació i la interfície amb l'usuari.

En el sisé capítol s'analitzen els resultats obtinguts en funció de l'usabilitat de la interacció i manipulació, mentre que al seté capítol s'exposen les conclusions extretes de la realització del projecte i es proposen objectius per a futurs desenvolupaments.

Finalment, el vuité capítol analitza el cost econòmic del projecte, i com a annexe s'afegeix el manual d'usuari de l'aplicació desenvolupada.

2 Conceptes previs

En aquesta secció s'explicaran els conceptes més importants per a enmarcar el projecte. Això inclou conèixer què és i com funciona la realitat virtual, la visió estereoscòpica, alguns conceptes relacionats amb aplicacions de suport a la visualització d'imatges mèdiques i algorismes per visualitzar models volumètrics en general.

2.1 Realitat Virtual

2.1.1 Definició

La realitat virtual es pot definir com una interfície que submergeix l'usuari complerta o parcialment en una aplicació mitjançant l'estimulació coherent d'algún dels seus sentits [4]. Es materialitza en una simulació per ordinador que varia en temps real d'acord a les accions i moviments de l'usuari. Un sistema de realitat virtual s'ha de compondre de: immersió sensorial en un món virtual, simulació interactiva en temps real i una interacció implícita en l'interfície d'usuari.

L'**immersió sensorial** és la connexió amb el món virtual a través dels sentits i la desconnexió amb el món real. Per a poder estar immers dins el món virtual hem de rebre informació coherent per els diferents sentits, especialment per la vista, ja que és el sentit que ens proporciona més informació i ens dóna la sensació de presència. Els dispositius de visualització de realitat virtual creen en l'usuari l'efecte que els objectes no estan projectats en cap superfície, sinó que es troben a diferents distàncies "flotant" en l'espai que envolta a l'observador. La clau d'aquest procés és la visió estereoscòpica. La visió estereoscòpica es basa en proporcionar dues imatges lleugerament diferents del món virtual, una per cada ull, de forma que el sistema visual humà dedueix la profunditat dels objectes a partir de les diferències en les imatges. La visió estereoscòpica és un requeriment imprescindible en tot sistema de realitat virtual, doncs és l'únic mitjà de fer que els objectes tinguin una forta presència espacial.

El fet que la **execució en temps real** sigui interactiva és el que diferència la realitat virtual d'una animació qualsevol. L'evolució de la simulació varia en temps real segons les accions del participant, es pot moure lliurement per l'escena, així que les seves accions repercuteixen en les imatges que veurà. El sistema de realitat virtual ha de projectar prou imatges per segon com per a què l'usuari noti sensació de moviment, i ha de recalcular també tot el que ha canviat degut a les accions imprevisibles de l'usuari. Això dista molt d'altres animacions com els efectes especials en les pel·lícules, on es pot calcular el renderitzat durant hores però el resultat dura pocs segons. Com a conseqüència, la potència de computació dels sistemes de realitat virtual ha de ser elevada, especialment si es manipulen models complexes.

La **interacció** amb l'escena virtual es fa principalment de manera implícita, és a dir, que l'aplicació actúa automàticament amb els moviments naturals de l'usuari. Per exemple, la posició de la càmera s'actualitza amb els moviments del cap de l'usuari, per el que l'escena es mou en concordança sense que l'usuari hagi de fer. La tecnologia actual no permet que tota la interacció es pugui realitzar implícitament en sistemes de realitat virtual, per el que es disposa de dispositius per a la interacció explícita (aquella en que s'utilitzen dispositius específics) els quals el participant n'és més conscient.

2.1.2 Aplicacions de la Realitat Virtual

La realitat virtual té potencials aplicacions en una gran varietat de camps relacionats fonamentalment amb la simulació. Alguns dels camps on és utilitzada i estudiada són:

- **Simulació**
  La realitat virtual permet simular el vehicle, avió, o qualsevol altre dispositiu que es requereixi aprendre a manejar, recreant condicions casi reals però evitant riscs i rebaixant molt el cost de la sessió de pràctiques. Els simuladors de vol i de conducció han sigut els primers en ser desenvolupats i de fet han donat un gran impuls a la realitat virtual, i són usats avui en dia en la formació de pilots. Un simulador complert inclou sistemes hidràulics per al movimient de la cabina, la reproducció de la cabina del pilot i el sistema informàtic de simulació.

- **Telepresència**
  És el resultat d'aplicar tècniques de realitat virtual a la tele-operació, que permet el control a distància de dispositius ubicats en entorns allunyats i/o perillosos. L'avenç consisteix en que l'operador se sent immers a l'entorn remot i pot executar les accions amb major naturalitat.

- **Visualització científica**
  La realitat virtual és la interfície idònea per ajudar als científics a

Figura 2.1: Cabina de simulació de vol [14]

interpretar grans volums de dades 3D (com models mol·leculars, terrenys, etc...). Moltes aplicacions es poden considerar en el camp de la medicina, desde la simulació i planificació quirúrgica fins a eines de visualització com ajuda al diagnòstic. La realitat virtual té també una evident utilitat en l'entrenament i l'ensenyament en aquest camp.

- **Disseny**
  Revisió de dissenys industrials i estudis d'ergonomia; aplicacions en indústria, arquitectura, urbanisme i interiorisme. La realitat virtual permet navegar per un edifici que encara no ha estat construït, tan sols dissenyat. Una maqueta o un prototipus en 3D es pot obtenir de manera directa a partir del programa de dibuix, i els canvis en el disseny es poden efectuar preveient el resultat.

- **Oci i cultura**
  El camp de l'oci i la cultura està treient profit de les noves tècniques de realitat virtual. La gran majoria de jocs d'ordinador utilizen motors gràfics 3D i alguns incorporen tècniques de visió estereoscòpica o interacció implícita.

2.2 Visió estereoscòpica

Un dels apartats més importants de la simulació visual és la capacitat de generar imatges estereoscòpiques per obtenir informació de profunditat de les imatges que veu l'usuari. La visió estereoscòpica és la forma que la visió humana permet detectar la profunditat, especialment per a objectes del nostre entorn més proper. La percepció estereoscòpica es basa en la propietat del cervell de combinar les dues imatges lleugerament diferents que reben els nostres ulls per a obtenir la sensació de profunditat. Si les imatges de cada retina es superposen es pot observar el que s'anomena *disparitat retinal*. La

disparitat retinal és la distància horitzontal existent entre punts homòlegs, mesurada en una retina *ideal*.

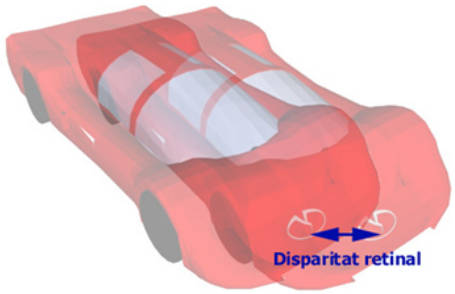
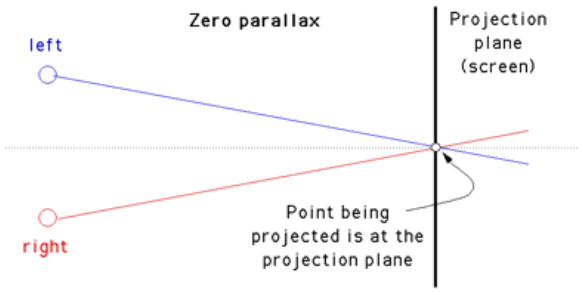


Figura 2.2: Disparitat retinal. [4]

Donat un punt de la imatge projectada a la retina de l'ull dret, i un altre punt de la imatge projectada a la retina de l'ull esquerre, es diu que aquests dos punts són homòlegs si corresponen a la projecció del mateix punt de l'escena real. Els punts homòlegs corresponents a l'objecte on convergeixen els ulls (l'objecte enfocat) tenen disparitat zero.

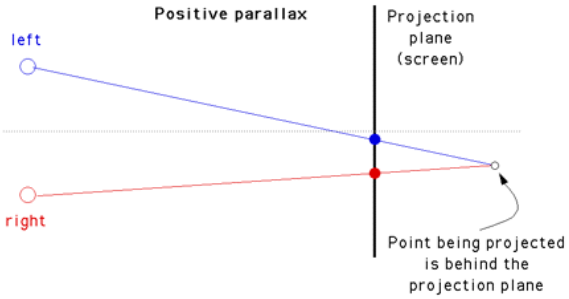
Si la disparitat retinal és la distància entre dos punts homòlegs en una retina, la *paral·laxi* és el mateix concepte mesurat sobre un monitor o pantalla de projecció. La diferència és important perquè la paral·laxi és independent de la convergència dels ulls de l'observador real (que varia contínuament segons l'objecte que concentra l'atenció de l'usuari), mentre que la disparitat retinal sí depèn de la convergència real. Els diferents tipus de paral·laxi que ens podem trobar són:

- **Paral·laxi zero:** Els objectes virtuals situats sobre el pla de la pantalla virtual tindran paral·laxi zero, la qual cosa vol dir que coincidiran les seves projeccions en pantalla. Els punts amb paral·laxi zero es perceben com si estiguessin a la mateixa profunditat que la pantalla real.

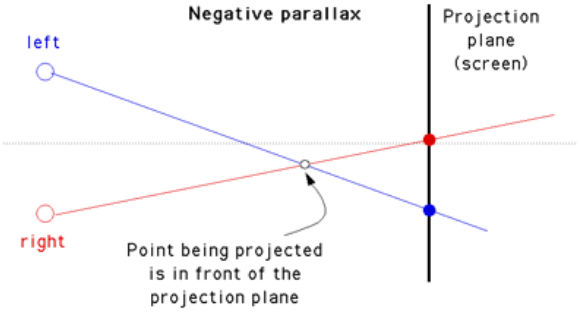


- **Paral·laxi positiu:** Els objectes virtuals situats darrera del pla de la pantalla virtual tindran paral·laxi positiu (la projecció per l'ull dret

surt més a la dreta que la de l'ull esquerre). Els objectes amb paral·laxi positiu es perceben com si estiguessin darrera del pla de la pantalla real.



- **Paral·laxi negatiu:** Els objectes virtual situats davant del pla de la pantalla virtual tindran paral·laxi negatiu (la projecció per l'ull dret surt més a l'esquerra que la de l'ull esquerre). Els objectes amb paral·laxi negatiu es perceben com si estiguessin flotant davant del pla de la pantalla real.



- **Paral·laxi divergent:** Si la paral·laxi és major que la distància interocular, els eixos oculars són divergents i es produeix fatiga ocular. Aquesta situació no es dóna en la visió normal i no es usada en estereoscòpia.

2.3 Arquitectura d'un sistema de Realitat Virtual

En un sistema de realitat virtual es poden distingir elements hardware i elements software. Els components hardware més importants són el computador, els perifèrics d'entrada, i els perifèrics de sortida. Els components software més importants són el model geomètric 3D i els programes de simulació i recollida de dades. Per aconseguir que l'usuari d'un sistema de RV tingui la sensació d'immersió és necessari que el sistema disposi de la capacitat de generar imatges estereoscòpiques, de captar la informació de l'usuari mitjançant elements posicionadors, i ha de donar la possibilitat a l'usuari d'interaccionar amb el model. Opcionalment, un sistema de realitat

virtual pot disposar d'un sistema d'àudio o d'un sistema de retroalimentació de força (que fa sentir les forces corresponents a l'ambient virtual), elements que permeten una major immersió.

2.3.1 Perifèrics d'entrada (sensors)

Els perifèrics d'entrada s'encarreguen de capturar les accions del participant i enviar aquesta informació al computador. Els perifèrics d'entrada més freqüents en realitat virtual són els posicionadors (que permeten al sistema conèixer en temps real la posició i orientació del cap, de la mà, o de tot el cos de l'usuari) i els elements d'interacció (que permeten a l'usuari interaccionar amb el model).

- Posicionadors: Els posicionadors són sensors que tenen com a missió capturar la posició i/o orientació d'un objecte real i enviar aquesta informació al computador. Solen ser dispositius amb sis graus de llibertat (6 DoF, Degree of Freedom), tres d'ells corresponen a la posició(x,y,z) i tres més a l'orientació (α, β, γ). Hi ha posicionadors basats en diferents tecnologies:

  - **Òptics**
    Es basen en processar les imatges captades per varies càmeres amb posició coneguda. Aquest procés requereix un temps que introdueix un retard apreciable amb la posició real. Sempre ha d'haver visió directa entre l'emissor i el receptor.
  - **Magnètics**
    Utilitzen bobines per obtenir la posició i orientació basant-se en les variacions de tensió elèctrica induïdes per una font de camp magnètic, que ha d'estar pròxima als sensors. Tenen un radi d'acció de pocs metres i pot haver-hi obstacles entre l'emissor i el receptor.
  - **Acústics**
    Utilitzen ultrasons i micròfons per obtenir la posició. Es poden usar en distàncies majors que els magnètics i ha d'existir una línia de visió entre l'emissor i el receptor.
  - **Mecànics**
    Aquests posicionadors utilitzen potenciòmetres montats sobre una estructura articulada per mesurar els angles. No requereixen visió directa entre l'emissor i el receptor, però limiten la llibertat de moviments ja que solen ser bastant voluminosos.

- Elements d'interacció: Els elements d'interacció permeten al participant interactuar amb el model virtual. Podem classificar els elements d'interacció en implícits, quan la interacció del model es fa de forma

natural, o explícits quan l'interacció no es correspon amb els moviments que faríem en una escena real. A continuació veurem els més comuns.

Interacció implícita

Com a dispositiu d'interacció implícita tenim principalment el guà de dades.

- **Guants de dades**
  Els guants de dades que es fan servir en realitat virtual permeten detectar la posició dels dits de la mà, normalment expressada com els angles de flexió de cada dit. Aquesta informació es combina normalment amb la posició de la palma de la mà (proporcionada per un posicionador magnètic). Alguns poden aplicar forces sobre els dits (realimentació de força), de manera que quan l'usuari toca virtualment un objecte, els seus dits es deformen per adoptar la forma de la superfície de l'objecte.

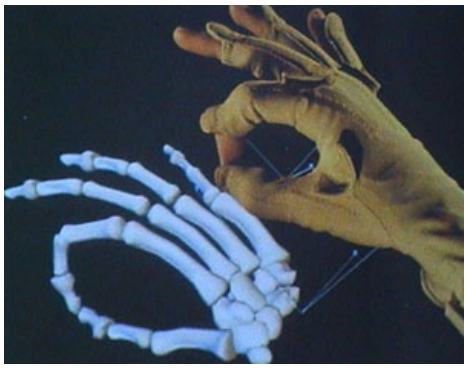


Figura 2.3: Guant de dades [4]

Interacció explícita

Alguns dels dispositius més habituals en realitat virtual són els següents:

- **Joystick**
  El joystick és un dispositiu fàcil de programar i el seu principal ús és el de permetre la navegació per escenes tridimensionals de manera intuïtiva. Són barats, robusts i es poden adquirir fàcilment.
- **Wanda (Ratolí 3D)**
  El wanda és un tipus de ratolí 3D, que disposa d'un sensor magnètic, gràcies al qual coneixem la seva posició i orientació (6DoF), a més d'incorporar tres botons i un mecanisme similar al joystick, de 2 graus de llibertat (2DoF x,y).

Figura 2.4: Wanda [13]

– **PDA**
La PDA és un petit computador que es pot programar per interactuar amb escenes virtual de la forma que es desitji. Es pot connectar a una xarxa inalàmbrica, de manera que s'allibera a l'usuari de cables que podrien dificultar els seus moviments.

2.3.2 Perifèrics de sortida (efectors)

Els perifèrics de sortida s'encarreguen de traduir els senyals d'àudio, vídeo, etc. generats pel computador en estímuls pels òrgans dels sentits (so, imatges, etc.). Els efectors es classifiquen segons el sentit al que estan adreçats: hi ha efectors visuals (cascos estereoscòpics, pantalles de projecció...), d'àudio (sistemes de so, altaveus), de força i tacte (dispositius tàctils), i del sentit de l'equilibri (plataformes mòbils). Un sistema de realitat virtual ha de disposar com a mínim d'efectors visuals, per els quals existeixen dos paradigmes bàsics de visualització: un totalment immersiu, on l'usuari només rep informació sensorial del sistema de RV; i un altre semi-immersiu, on també pot rebre informació del món real.

Cascos estereoscòpics

Un casc estereoscòpic és un dispositiu que s'adapta al cap del participant i li permet veure el món virtual en 3D. Els cascos estereoscòpics incorporen sensors que registren la posició i orientació del cap del participant per tal d'actualitzar el seu punt vista convenientment. Aquests dispositius creen una experiència totalment immersiva, al no poder veure ni escoltar res de l'exterior.

Els cascos estereoscòpics es poden classificar en dues categories segons si s'adapten i recolzen únicament al cap de l'usuari, o si s'han de subjectar amb les mans com uns binòculs:

- **Head-Mounted Displays**
  Els Head Mounted Displays (HMD) són cascos immersius que s'adapten al cap de l'usuari. Incorporen dues petites pantalles i els elements òptics necessaris per permetre l'enfocament. Cada pantalla està enfilada amb un ull i rep un senyal de vídeo diferent. També incorporen sensors que registren la posició i orientació del cap del participant per tal d'actualitzar el seu punt vista convenientment. Anul·len completament la visió de l'entorn real.

Figura 2.5: Head-Mounted Display [4]

- **Head-Coupled Displays**
  Els Head-Coupled Displays (HCD) són similars als HMD però incorporen pantalles CRT molt pesades, per la qual cosa estan muntats sobre un suport mecànic i no es recolzen directament al cap de l'usuari. Aquest suport mecànic incorpora potenciòmetres a les articulacions per tal de registrar la posició i orientació del cap de l'usuari.

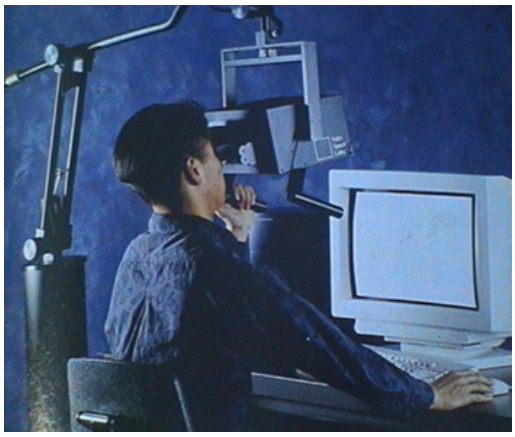


Figura 2.6: Head-Coupled Display [4]

Sistemes basats en projecció

En aquests sistemes, les imatges es projecten en una o més pantalles (normalment pantalles de projecció) les quals poden adoptar diferents configuracions segons el número, forma i disposició. Creen un entorn semi-immersiu, ja que la pantalla només és una part del què pot veure l'usuari. Les configuracions comercials més utilitzades són:

- **Habitació estereoscòpica**
  Són sistemes basats en projecció que disposen de tres o més pantalles de projecció de grans dimensions (de 2 a 4 metres). Dels sistemes de visualització de realitat virtual basats en projecció, un dels sistemes que ofereix a l'usuari un major grau d'immersió és la CAVE. Una CAVE (acrònim de Computer-Animated Virtual Environment, i alhora traducció de cova) consisteix en una habitació amb unes dimensions al voltant dels 3x3x3 m, en la qual de 2 a 5 parets laterals, el terra i/o el sostre, són pantalles on es projecten imatges estereoscòpiques. El més comú és utilitzar retroprojecció per les parets (per evitar que el participant vegi la seva ombra a la pantalla) i projecció directa pel terra.

- **Taula estereoscòpica (workbench)**
  Una taula estereoscòpica consisteix en una estructura on les imatges es projecten en una pantalla horitzontal en forma de taula. Quan les aplicacions de realitat virtual requereixen la manipulació d'objectes en l'espai virtual, és molt important el grau de coordinació entre la visió i els moviments de la mà, coordinació que és molt feble en els sistemes de visualització basats en cascos estereoscòpics, que requereixen utilitzar una representació virtual de la mà de l'usuari.

2.3.3 Software de simulació sensorial

Aquest mòdul s'encarregua de calcular la representació digital de les imatges, sons, etc. que el hardware s'encarregarà de traduir a senyals i finalment a estímuls pels sentits. Entre els mòduls de simulació sensorial, el més important és el de simulació visual, que es basa en algorismes de visualització en temps real del model geomètric. Donat que el rendiment és crític, s'utilitzen tècniques d'acceleració d'imatge per tal de reduir al mínim possible el temps de generació de cada fotograma. Respecte a la simulació auditiva, cal dir que la generació de so realista requereix tenir en compte les propietats acústiques dels objectes i que els algorismes són tan complicats com els algorismes de visualització. Respecte a la simulació tàctil, cal distingir entre els dispositius que proporcionen sensació de tacte (sovint limitada a la mà), sensació de contacte (també limitada a la mà) i

Figura 2.7: Workbench [4]

realimentació de força (impedeixen o ofereixen resistència a fer moviments amb la mà quan aquesta topa virtualment amb un objecte virtual). En qualsevol dels casos, és imprescindible que el sistema sigui capaç de detectar en temps real les col·lisions que es puguin produir entre la mà de l'usuari i els objectes de l'escena, ja que és aquest esdeveniment qui activa els dispositius hardware adients.

Per aconseguir la disparitat retinal, es necessari fer arribar dues imatges 2D, una a cada ull, de manera que cada ull rebi únicament la imatge corresponent. Això es pot fer de diverses maneres depenent del dispositiu visual emprat:

- **Estèreo actiu:** L'estèreo actiu es basa en projectar les imatges de l'ull dret i de l'ull esquerre sobre una única pantalla, de forma alternada. Per tant, es basa en fer una separació en el temps del parell estereoscòpic. Per utilitzar estèreo actiu n'hi ha prou amb que el nostre computador disposi d'un canal de vídeo en el qual es codifiquen les dues imatges de forma alterna. La majoria de computadors personals incorporen una única sortida de vídeo, a la qual es connecta el monitor o el projector, però el dispositiu ha de suportar una freqüència de refresc vertical de com a mínim 120 Hz, ja que la freqüència que rebrà cada ull serà de només la meitat, 60 Hz. El cervell humà, el percebre tan ràpidament per cada ull cadascuna de les imatges, tendeix a unir-les en una sola, amb el que s'aconsegueix la sensació d'estar veient en tres dimensions.

  De cara a que cada ull de l'usuari pugui veure només les imatges que li corresponen, l'usuari ha de dur unes ulleres d'obturació (*shutter glasses*) que porten un LCD especial davant de cada ull que obstrueix el pas de la llum de forma sincronitzada amb el senyal de vídeo. Quan

la pantalla visualitza la imatge per a un ull determinat, el LCD situat davant d'aquest ull serà transparent, mentre que el LCD de l'altre ull s'enfosquirà. Si la freqüència de refresc és prou elevada, l'obturació és imperceptible (degut a la retenció de la retina) i aconseguim de forma efectiva que cada ull rebi únicament les imatges que li pertoquen.

- **Estèreo passiu:** L'estèreo passiu es basa en la utilització de filtres polaritzadors per projectar les imatges de l'ull dret i de l'ull esquerre sobre una única pantalla de forma simultània (Figura 2.8), de forma que la llum dirigida a l'ull dret té una polarització perpendicular a la llum dirigida a l'ull esquerre. Per tant, es basa en fer una separació en la direcció de polarització de la llum. Per utilitzar estèreo passiu es requereixen dos canals de vídeo. Un canal proporcionarà la imatge per l'ull esquerre, i l'altre la imatge per l'ull dret. Això es pot aconseguir, bé utilitzant una estació gràfica amb dues sortides gràfiques, o bé combinant les sortides gràfiques de dos computadors, que hauran d'estar sincronitzats.

Figura 2.8: Estèreo passiu [4]

La solució més freqüent de visualització per l'estèreo passiu és utilitzar dos projectors, on a la sortida del projector es situa un filtre polaritzador. El filtre polaritzador actua deixant passar només la llum que oscil·la en la direcció determinada pel seu eix de polarització. Un filtre es col·loca amb l'eix de polarització vertical, i l'altre amb l'eix de polarització horitzontal. Els projectors s'ajusten perquè projectin les imatges exactament en la mateixa zona de la pantalla.

De cara a que cada ull de l'usuari pugui veure només les imatges que li corresponen, l'usuari ha de dur unes ulleres de polarització (polariser glasses) que porten filtres polaritzadors davant de cada ull, amb els eixos de polarització perpendicular l'un respecte de l'altre. D'aquesta

forma aconseguim que cada ull vegi només les imatges que li pertoquen.

- **Estèreo anàglif:** L'estèreo anàglif es basa en projectar les imatges de l'ull dret i de l'ull esquerre sobre una única pantalla, però amb colors complementaris, de forma que amb ulleres amb filtres de color es pugui dur a terme la separació del parell estereoscòpic. De cara a que cada ull de l'usuari pugui veure només les imatges que li corresponen, l'usuari ha de dur unes ulleres anaglífiques que porten un filtre vermell davant l'ull esquerre i un filtre blau davant l'ull dret. El primer filtre només deixa passar la component R de la llum; el segon filtre només deixa passar les components G i B.

Figura 2.9: Ulleres anaglífiques [4]

2.4 Entorn clínic

La relació entre metges i computadors és important perquè ofereix la possibilitat d'augmentar l'eficàcia, seguretat i eficiència dels procediments clínics existents, i de desenvolupar nous procediments que no es podrien dur a terme d'una altra manera. Els ordinadors poden ajudar als cirurgians, o metges en general, realitzant tasques com col·locar un instrument en un lloc precís, captant informació de manera molt precisa, treballant en entorns amb radiació ionitzada, etc [11].

En el cas concret d'una operació quirúrgica, un computador pot ajudar en diverses tasques per millorar el resultat. Primer de tot en obtenir i emmagatzemar tota la informació del pacient, això inclou des de resultats d'anàlisis fins la captació d'imatges d'una tomografia axial computeritzada (TAC). Després en planejar i optimitzar què es farà en l'operació, a més de poder simular-ne el resultat. Durant l'operació es pot realitzar alguna part per mitjà de manipulació robòtica, proporcionar informació actual del pacient al cirurgià o registrar les imatges de l'operació.

En un entorn clínic, una aplicació de realitat virtual pot ajudar en la visualització i manipulació de dades mèdiques, degut a la sensació de realisme en l'interacció amb l'informació. Algunes tècniques de diagnòstic com el TAC o la ressonància magnètica permeten obtenir dades de l'interior de

l'organisme que es poden representar en tres dimensions, i així poder-ne veure l'interior de l'organisme sense la necessitat d'operar el pacient. Aquestes imatges mèdiques estan compostes d'un grup d'imatges 2D adquirides a una distància regular, amb el que es té una quadrícula regular on cada píxel representa el valor de la densitat de l'àrea que ocupa. L'espai entre mostres s'acostuma a interpolar a partir de les mostres veïnes. Es pot aprofitar la immersió sensorial que ofereix la realitat virtual per a visualitzar i manipular aquestes dades mèdiques com si es tractés del mateix pacient. Els sistemes de realitat virtual es poden usar per ensenyar, entrenar, planejar i realitzar operacions, i fins i tot simular el resultat d'una operació. L'aprenentatge a través de realitat virtual té una sèrie d'avantatges respecte els vells mètodes com els llibres de text o la pràctica en cadàvers reals, com l'opció d'augmentar l'objecte per a veure'l millor o que es mostrin alhora altres tipus de dades, com la informació del pacient. Un entorn de realitat virtual es pot fer servir per a més coses que la simulació, pot controlar un sistema remot que realitza les operacions a través d'un robot, però per això es necessita una retroalimentació dels guants molt precisa ja que el cirurgià ha de determinar molt bé totes les accions.

2.5 Visualització

Per a poder visualitzar les representacions de les imatges mèdiques cal emprar tècniques de renderitzat de volum. El model de representació de dades més usual és el de vòxels, que són les unitats mínimes de volum i el seu nom prové de l'abreviació de *volume element*. Alhora de calcular el valor d'un punt interior d'un vòxel hi ha dues possibles interpretacions:

- Un vòxel és una cel·la cúbica, que té el mateix valor per a tota la regió que ocupa. Així, el valor de qualsevol punt interior de la cel·la és el mateix.
- Un vòxel és cada vèrtex de la cel·la cúbica, on cada vértex té el seu propi valor. El valor d'un punt interior qualsevol de la cel·la es calcula interpolant els valors dels vèrtexs.

Cada vòxel conté els valors numèrics associats a la funció de transferència en una regió concreta de l'espai. La funció de transferència és una representació que mapeja els valors de densitat en propietats òptiques arbitràriament, és a dir, és l'usuari qui decideix com es veuran les diferents estructures. Es poden tenir diferents regions de densitat mapejades amb diferents colors per a poder distingir diferents estructures. Per exemple, en el cas que es vulgui mostrar la densitat dels teixits d'un cos humà, cada tipus de teixit (ós, múscul, nervi...) estarà associat a un rang de valors de densitat, de tal manera que cada tipus de teixit tindrà un color diferent. Es pot veure

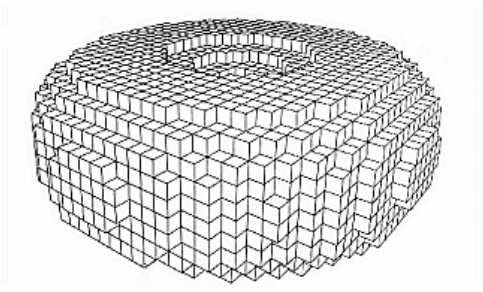


Figura 2.10: Representació d'un torus amb vòxels [12]

un exemple a la figura 2.11. Per a visualitzar aquesta representació hi ha diverses tècniques disponibles, entre elles les més exteses són les textures 3D i el ray casting. Aquestes tècniques s'han dissenyat per poder visualitzar models volumètrics en les targetes gràfiques actuals, que estan dissenyades i optimitzades per a pintar polígons.

Figura 2.11: Exemple de model segmentat on es veuen les diferents parts del cos [2]

2.5.1 Textures 3D

Una textura 2D és un patró detallat que es repeteix diversos cops per cobrir un espai multidimensional, o més generalment, una imatge que es mapeja en una superfície. El mapeig de textures és el procés d'aplicar una

textura sobre una superfície d'un objecte multidimensional. Una textura 3D és una imatge tridimensional amb alçada, amplada i profunditat, o vist des d'un altre perspectiva, una sequència d'imatges bidimensionals (plans, llesques) una darrera l'altre. Els texels són els elements que formen la textura i es caracteritzen per un color i una opacitat.

Per a la visualització d'un conjunt d'imatges mèdiques (o model de vòxeks) es genera una textura tridimensional fixant el color i la opacitat segons la funció de transferència establerta.Així, un cop definida la funció de transferència, es crea una textura 3D amb la mateixa resolució que el model de vòxels, perquè a cada vòxel li correspon un texel. Per a omplir la textura, es fa un recorregut dels vòxels, i per cada vòxel es consulta el valor de la propietat a visualitzar. A partir d'aquest valor, la funció de transferència retorna el valor de color i opacitat que té associat al valor del vòxel. La creació de la textura 3D s'ha de repetir cada cop que es modifica la funció de transferència, però no s'ha de repetir a l'aplicar transformacions geomètriques com rotacions o traslacions al volum.

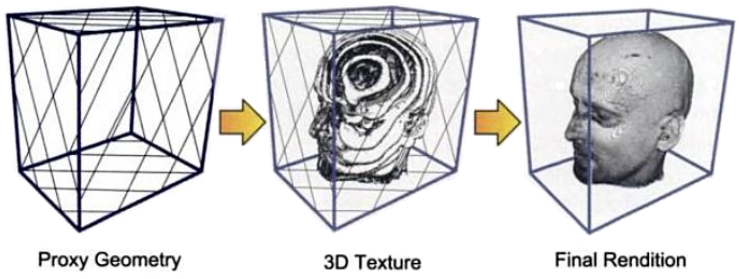


Figura 2.12: L'objecte es descompon en plans paral·lels orientats segons els viewport [2]

Per a visualitzar la textura 3D, es talla per plans ortogonals a la direcció de visió separats entre si per la mateixa distància, i així s'obtenen polígons texturats. Finalment es pinten i es componen en ordre aquests polígons des del més llunyà a l'observador fins al més proper per a obtenir una correcta sensació de volum. Quants més plans s'utilitzin per a la visualització, millor serà la qualitat de la imatges, però evidentment també augmenten el nombre de càlculs i empitjora la velocitat de visualització. Quants menys plans s'utilitzin, més augmentarà la velocitat de visualització, però apareixen nous problemes. Al utilitzar pocs plans amb relació a la resolució del model de vòxels, alguns dels vòxels poden no contribuir a la imatge final, amb el que es perden dades. També hi ha el problema que es notin els diferents plans que pintem ja que la diferència entre ells és abrupta, com en la figura 2.13. Així doncs cal arribar a un compromís entre la qualitat de l'imatge i la velocitat. El nombre ideal de llesques seria aquell que permet que cada vòxel sigui mostrejat com a mínim per una llesca.

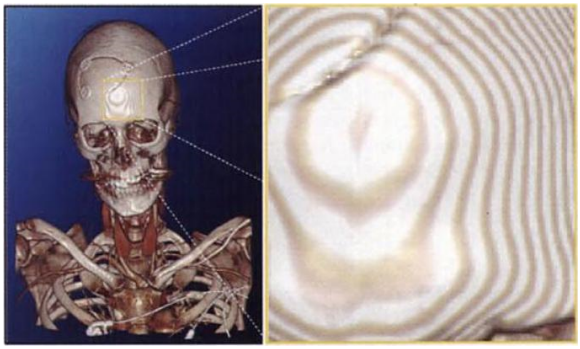


Figura 2.13: Exemple de visualització amb pocs plans [2]

El mètode de visualització d'un volum amb Textures 3D és el més emprat, ja que les targes gràfiques texturen molt ràpidament i poden renderitzar eficientment els plans texturats. Però com el nombre i posició dels polígons depén del model, és un mètode que depén molt de la complexitat de les dades.

2.5.2 Ray Casting

La tècnica de ray casting és un mètode que intenta resoldre una integral que representa la quantitat de llum emesa per un punt al llarg d'una direcció de visió donada la radiació d'entrada i les propietats òptiques. Es basa en la llei física de la conservació de l'energia, ja que la quantitat de llum de sortida (emissió i reflexió) ha de ser la mateixa que la d'entrada en totes les direccions.

Per utilitzar aquest mètode en la visualització d'un volum es genera un raig des de l'observador fins al model de vòxels per a cada pixel que es vol a la imatge resultant. S'analitzen les mostres (els valors de densitat en els vòxels creuats pel raig) al llarg de la trajectòria dins el model i s'acumulen els factors de color i opacitat que retorna la funció de transferència, i per últim es compon el color del píxel. Bàsicament els passos que segueix són:

- **Inicialització dels raigs** Primer s'ha d'inicialitzar cada raig segons els paràmetres de la càmera i les posicions dels respectius píxels. Aquest pas determina la direcció del raig i les coordenades de la primera intersecció amb el model.

- **Bucle** Dins aquest bucle s'evalua l'equació de renderitzat fins que el raig atravessa per complet el model.

  - **Accés a les dades** S'accedeix a les dades en la posició actual del raig, segons el model de vòxels pot requerir interpolació per

aconseguir el valor. S'aconsegueixen els valors de color i opacitat mitjançant la funció de tranferència.

- **Composició** S'acumulen els valors obtinguts anteriorment de color i opacitat d'acord amb una equació *front-to-back* (de davant cap a darrere).
- **Finalització** S'avança la posició actual segons la direcció del raig, i si és fora del model de volum s'acaba el bucle i es passa al següent raig.

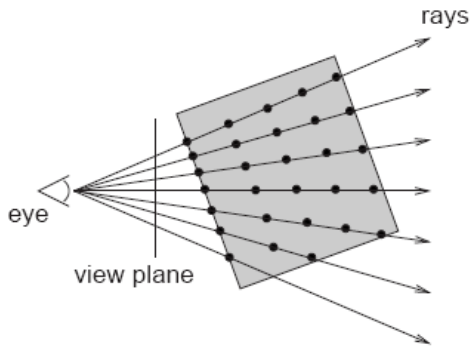


Figura 2.14: Per cada píxel es genera un raig

Aquest mètode proveeix resultats de molta qualitat a costa d'un gran cost computacional, tot i que al ser cada raig independent dels altres pot aprofitar en gran manera el paral·lelisme. Per tal d'optimitzar, es pot deixar d'analitzar un raig quan l'opacitat acumulada ja és màxima, ja que la resta no contribuirà a la imatge final. És una optimització important ja que típicament només entre el 0,2% i el 4% de tots els fragments és visible en un moment donat [2].

3 Objectius i requeriments

A continuació s'exposen els objectius finals del projecte i els requeriments necessaris per a realitzar-lo.

3.1 Objectius

En el grup de recerca MOVING [3], han desenvolupat un sistema de visualització de dades mèdiques "VRMedVolVisDicom". Es tracta d'una aplicació que permet carregar fitxers en format DICOM (Digital Imaging and Communication in Medicine), un estàndar reconegut mundialment per a l'intercanvi d'imatges mèdiques, i pensat per la seva manipulació, emmagatzematge, impressió i transmissió. L'aplicació permet manipular el model a través d'un dispositiu d'entrada 2D (el ratolí) que no seria adequat per a un entorn immersiu. L'objectiu global d'aquest projecte és adaptar l'aplicació VRMedVolVisDicom a un entorn de realitat virtual per a analitzar la seva usabilitat, avantatges i limitacions.

A tal efecte es plantegen els següents objectius concrets:

- **Adaptació de l'aplicació per a visualització estereoscòpica.** Per a poder generar la sensació de profunditat s'han de crear dues imatges lleugerament diferents de l'escena, a part de moure la càmera d'acord al moviment de l'usuari.

- **Estudi i adaptació de les interfícies gràfiques d'usuari a un entorn de Realitat Virtual (RV).** En un entorn tridimensional surgeixen nous problemes d'interacció que no són presents en els escriptoris de dues dimensions (2D). En entorns 2D s'han adoptat els mateixos principis bàsics i dissenys durant els últims deu anys o més. No es poden traslladar aquestes solucions directament a un entorn tridimensional, sinó que s'han d'utilitzar noves tècniques per 3D.

- **Anàlisi de tècniques de manipulació directa d'objectes a entorns de RV amb interacció implícita. Disseny i programació**

de nous paradigmes d'interacció. Amb la realitat virtual apareixen nous dispositius que ens permeten interactuar de noves maneres amb l'entorn, i per tant es requereixen noves tècniques i metàfores per a poder usar-los de manera fàcil i intuïtiva.

- **Anàlisi de la tècnica de visualització més apropiada en entorns de Realitat Virtual.** No totes les tècniques de visualització es comporten correctament amb les mateixes condicions, i per això s'ha d'analitzar quina és la tècnica que més s'adapta a les necessitats de la RV.

- **Anàlisi d'usabilitat de l'aplicació resultant.** Un cop finalitzada l'aplicació, s'ha de comprovar que la interacció i la manipulació siguin fàcils d'usar, comfortables i amb un comportament correcte del sistema.

3.2 Requeriments

Un cop analitzats els objectius del projecte, cal determinar els requeriments de software funcionals i no funcionals per a completar el projecte amb éxit.

3.2.1 Funcionals

- Funcionalitats de visualització similars a VRMedVolVisDicom, on es poden utilitzar dos algorismes de visualització diferents: Textures 3D i Ray Casting.

- Portabilitat. Cal que l'aplicació es pugui adaptar a diferents sistemes de RV, de manera independent, als dispositius de sortida i entrada que es disposin. A tal efecte, l'aplicació s'ha de construir utilitzant el sistema VR Juggler que permet independitzar l'aplicació dels perifèrics concrets de que es disposi.

- La interfície de control de l'aplicació ha de tenir una configuració semblant a l'aplicació de sobretaula VRMedVolVisDicom per a faciliar el seu aprenentatge per part de l'especialista clínic. A tal efecte, es proposa la utilització del sistema Qt3D per a facilitar la migració de les interfícies gràfiques dissenyades en Qt al nou entorn.

- Programació amb C++ i OpenGL [5].

3.2.2 No funcionals

Els requeriments no funcionals es basen principalment en la interacció.

- S'ha d'evitar que la interfície molesti a l'usuari quan interaccioni amb el model 3D, la interfície és un element secundari.
- Per a facilitar l'accés a l'interfície s'ha de mantenir una posició fixa respecte la pantalla.
- El sistema ha de proporcionar una resposta per a cada acció que realitza l'usuari, ja que així es pot entendre millor l'estat del sistema o el resultat d'una operació.

4 Software utilitzat

Per aconseguir l'objectiu global del projecte d'adaptar l'aplicació de sobretaula a un entorn de realitat virtual amb els requeriments comentats, s'ha analitzat el software existent que es podria utilitzar per dotar l'aplicació d'aquestes característiques. En aquesta secció s'introdieix i justifica el software escollit per a garantir la indepèndencia del sistema (VR Juggler), i per facilitar la migració de la interfície (Qt3D). També es presenten les principals característiques de l'aplicació que es vol migrar a RV (VRMedVolVisDicom).

4.1 VR Juggler

VR Juggler [6] és una plataforma que permet executar aplicacions de realitat virtual en quasi qualsevol sistema de RV. Es compon d'una sèrie de llibreries que creen una capa d'abstracció entre el hardware on l'aplicació s'està executant, els perifèrics i l'aplicació a desenvolupar (Veure figura 4.1). Cada llibreria serveix per a poder utilitzar una característica diferent: un gestor de dispositius d'entrada/sortida (Gadgeteer), un gestor d'àudio (Sonix), utilització de interfícies GUI (Tweek), un sistema de configuració basat en fitxers XML (JCCL), i una capa d'abstracció multi-plataforma (VPR).

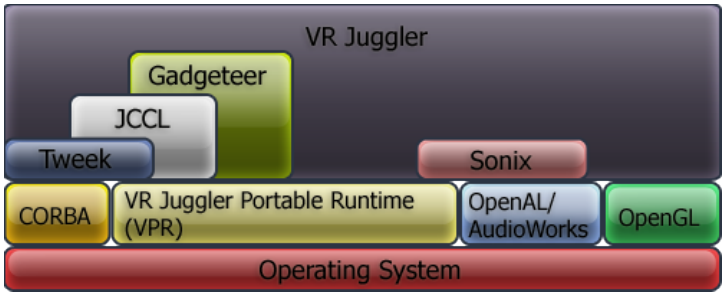


Figura 4.1: Arquitectura VR Juggler [6]

Tots aquests components doten VR Juggler d'una sèrie d'avantatges:

- **Portabilitat:** VR Juggler 2.0 suporta IRIX, Linux, Windows, FreeBSD, Solaris, i Mac OS X.
- **Escalabilitat:** Pot executar-se en gran varietat d'entorns: des d'un PC a un super-ordinador.
- **Codi lliure:** Al tindre una llicència lliure és més apropiat que altres sistemes que són més restrictius, com OpenGL Performer [15] o CaveLib [16].
- **Modular:** Cada mòdul és independent, per el que el sistema només ha d'incloure els que s'utilitzen.
- **Gestió dels perifèrics:** Funciona independentment dels perifèrics utilitzats. Gestiona la sortida independentment del nombre de pantalles, i genera la visualització en estèreo si es necessita. Per això, no es pot accedir directament al hardware dels dispositius. Aquesta decisió de disseny és el que ens permet desenvolupar aplicacions amb VR Juggler que són portables cap a qualsevol sistema, evitant dependre de dispositius específics.
- **Fitxers de configuració:** Es pot modificar la configuració dels dispositius d'entrada/sortida sense haver de tocar el codi de l'aplicació.

VR Juggler considera totes les aplicacions com a objectes utilitzats per el kernel, coneguts amb el nom de *application objects* (aplicacions objecte). Com a resultat d'això, les aplicacions no tenen un mètode *main()* sinó que han d'implementar unes interfícies pre-definides, d'aquesta manera VR Juggler controla l'aplicació cridant els mètodes de les interfícies. La interfície *vrj::App* especifica els mètodes per a la inicialització, execució i finalització de l'aplicació, per el que és necessari implementar les següents funcions:

- *init()*
  Aquest mètode és cridat per el kernel per inicialitzar qualsevol dada de l'aplicació. Quan el kernel es prepara per a començar una nova aplicació, primer crida a *init()* per avisar-la que està apunt de ser executada.
- *apiInit()*
  Aquesta funció és per si l'aplicació necessita alguna inicialització de la API gràfica, i també per les dades que s'han d'inicialitzar després de que aquesta s'hagi inicialitzat.
- *preFrame()*
  El mètode *preFrame()* és cridat quan el sistema és a punt de pintar.

És el moment en que l'aplicació objecte ha de fer les últimes actualitzacions d'informació provinents de dispositius d'entrada. És el primer mètode del bucle del kernel per obtenir els frames.

- *intraFrame()*
  Aquest mètode s'executa en paral·lel al mètode de renderitzat, o sigui, mentre es pinta el frame actual. Aquí és el lloc per a processar tot el que es pugui sobre el següent frame. S'ha d'anar amb compte de no modificar cap dada que s'estigui utilitzant en el renderitzat.

- *postFrame()*
  Finalment, aquesta funció està disponible al final del bucle dels frames, per el que és un bon lloc per a posar qualsevol actualització que no depengui dels dispositius d'entrada o que no es pugui solapar amb el renderitzat.

El mòdul que gestiona la part de dispositius dintre de VR Juggler és el *Gadgeteer Input Manager*, per a l'informació d'entrada del dispositiu cap a l'aplicació. La manera de poder utilitzar un dispositiu és conectar-se a ell via proxy, que no és més que un intermediari entre dos parts. En aquest cas, una de les parts és el dispositiu i l'altre l'aplicació VR Juggler. Per tant, l'aplicació fa peticions al dispositiu mitjançant el proxy. Per ajudar a l'ús de proxies per dispositius, Gadgeteer proveeix el que es coneix com interfícies de dispositius, que oculten com l'Input Manager adquireix la informació del proxy. Per a cada tipus de dispositiu, ja sigui digital, analògic, teclat o ratolí existeix una classe de proxy. Conèixer una classe proxy per tal de gestionar un dispositiu no requereix gran esforç, només cal conèixer la interfície per al tipus de dispositiu que es vulgui gestionar.

Abans d'usar una interfície d'un dispositiu, s'han de declarar i inicialitzar alguns objectes. Un cop escollits el tipus d'interfícies més adequats per als dispositius de l'aplicació (GloveInterface, KeyboardMouseInterface, PositionInterface...), s'han d'inicialitzar a la funció *vrj::App::init()*. Per a fer-ho, n'hi ha prou amb passar el nom del proxy a que es connectarà com a argument de la funció *gadget::DeviceInterface<T>::init()*. Un cop inicialitzat es pot aconseguir fàcilment la informació del dispositiu amb el mètode *getData()*, que en el cas d'un dispoditiu posicionador retorna una matriu de 4 files i 4 columnes que indica la transformació geomètrica realitzada. Aquesta transformació indica el canvi de posició realitzat, que s'afegeix a OpenGL per actualitzar la posició dins el món virtual.

4.1.1 Alternatives

VR Juggler és un entorn per desenvolupar aplicacions de realitat virtual, però no és l'únic que existeix. Una d'aquestes alternatives és XVR [7] que comentem a continuació.

XVR (eXtreme Virtual Reality) està basat en un script semblant a C++, i es pot desenvolupar sense la necessitat de compiladors externs, generant codi multiplataforma que es pot desplegar tant en aplicacions professionals de realitat virtual com en pàgines web. Incorpora tecnologia com la detecció de col·lisions, simulació física en temps real, comunicació per xarxa o gestió de dispositius de realitat virtual.

Una de les avantatges d'utilitzar XVR és que es pot executar en qualsevol ordinador que disposi del navegador *Internet Explorer*. S'ha d'instalar un plugin d'*ActiveX* el primer cop que s'executa, que no és res més que el controlador de XVR que incorpora una màquina virtual. A l'executar-se en un navegador té les mateixes limitacions que aquest per qüestions de seguretat, però es pot ampliar mitjançant llibreries *dll* que incorporen noves funcions. Així, s'hauria de descarregar aquesta llibreria, a part del programa en si mateix i el controlador.

En principi aquesta solució no compleix tots els requeriments, ja que requereix l'ús d'un navegador en concret no present en tots els SO. No obstant, s'estan desenvolupant plugins per a altres navegadors com *Mozilla Firefox*, el que permetria més portabilitat. Tot i això, no es va utilitzar en el projecte, principalment perquè VR Juggler és el que s'utilitza dins el grup MOVING i és té una experiència positiva.

4.2 Qt3D

Qt3D [8] és una llibreria que permet integrar interfícies gràfiques d'usuari (GUI) creades amb Qt en qualsevol entorn de visualització. La idea és tenir el GUI dissenyat en Qt executant-se en un thread diferent del que renderitza, i comunicant-se entre ells amb el mecanisme de Signal/Slot típic de Qt. Qt3D mapeja les finestres GUI de Qt en textures 2D i les incorpora al món virtual, permetent dissenyar la interfície de l'aplicació en Qt, que dóna una imatge ja coneguda en les aplicacions de sobretaula. Per a poder interaccionar amb les finestres, proporciona una representació del cursor 3D, que indica la posició del dispositiu i la direcció on apunta. El bucle principal comprova si l'usuari està apuntant a alguna finestra mitjançant una tècnica de ray casting i mapeja la interacció, comportant-se com si es tractés de la finestra real.

Els principals avantatges d'usar finestres 2D en un entorn 3D a través de Qt3D són:

- Delegació del manegament de les funcionalitats de l'interfície a un component independent.
- Els usuaris poden treballar amb interfícies ja conegudes.
- Es poden usar diferents tècniques de selecció 3D (Com el ray-casting).

- Una gran part de la interfície es pot desenvolupar i testar en una estació de treball convencional.
- Accés a eines generals de desenvolupament d'interfícies (QtDesigner).
- Fàcil i ràpida adaptació d'aplicacions existents a entorns de realitat virtual.

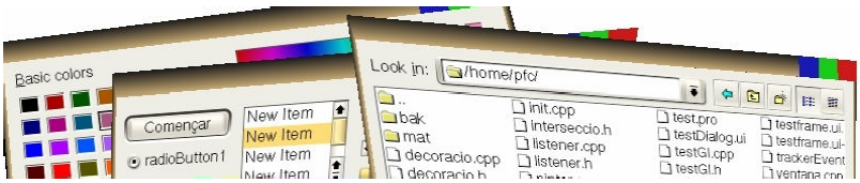


Figura 4.2: Finestres de Qt3D

Utilitzar Qt3D en una aplicació és bastant semblant a fer servir Qt, en comptes de declarar una aplicació QApplication declarem una QApplication3D. Després s'han de fer algunes inicialitzacions pròpies de Qt3d, com la profunditat a la que es col·locaran les finestres o el seu factor d'escalat. Només queda pintar les finestres i activar el fil principal amb *app.exec()*.

4.2.1 Alternatives

A part de Qt3D es podien haver emprat altres solucions existents per integrar les interfícies gràfiques en l'aplicació. Una d'elles s'anomena Tweek, que permet a una interfície Java communicar-se amb una aplicació en C++. Tweek forma part de l'arquitectura de VR Juggler (com es veu a la figura 4.1) i utilitza l'arquitectura CORBA, que proveeix una plataforma on es poden communicar objectes distribuïts independentment del llenguatge de programació. Això permet que les GUI puguin ser dissenyades en qualsevol llenguatge inclós Qt, no només Java. Tot i això, el principal inconvenient de Tweek és que no es poden incorporar les GUI a l'entorn virtual, sinó que s'hauria d'usar un altre dispositiu manual (Tablet PC, PDA...) només per a mostrar l'interfície.

4.3 VRMedVolVisDicom

L'aplicació VRMedVolVisDicom és un sistema de visualització de dades mèdiques desenvolupat per el grup de recerca MOVING [3] de la Universitat Politècnica de Catalunya. L'objectiu del projecte és adaptar aquest programa a un entorn de realitat virtual, per el que s'analitzarà detalladament. Es tracta d'una aplicació de sobretaula que permet carregar fitxers en format DICOM (Digital Imaging and Communication in Medicine), un

estàndar reconegut per a l'intercanvi d'imatges mèdiques, i pensat per la seva manipulació, emmagatzematge, impressió i transmissió.

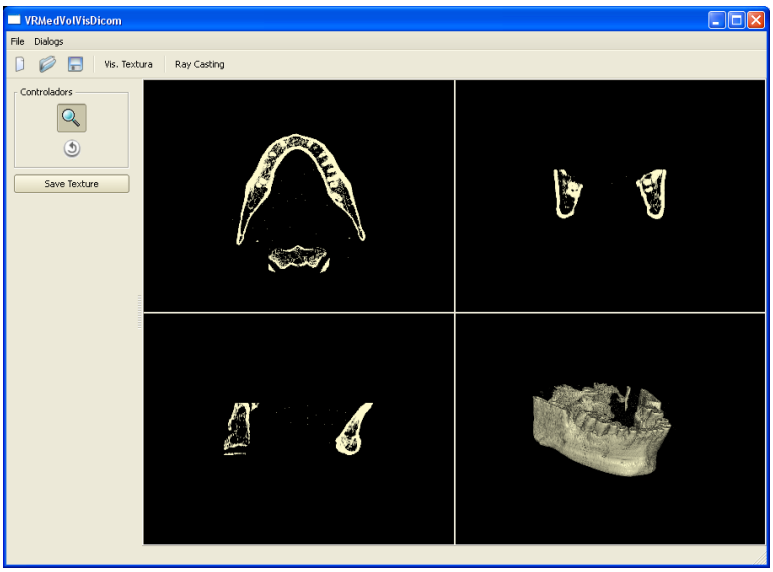


Figura 4.3: Finestra de VRMedVolVisDicom

Un cop carregat un model, es visualitza en quatre vistes (Figura 4.3), tres de les quals són plans de tall perpendiculars, i una amb la vista en 3D. A la vista en 3D es pot escollir entre l'algorisme de textures 3D o el de ray casting, podent comparar les diferents visualitzacions i modificar paràmetres avançats. Incorpora una eina per a poder visualitzar l'interior dels models, o per acolorir-ne diferents parts per a poder reconeixer-les millor podem utilitzar, anomenada paleta. La paleta (Figura 4.4) serveix per a modificar la funció de transferència, que es basa en una representació 2D on la x és la densitat dels punts del model, i la y és l'alpha, o grau de transparència.

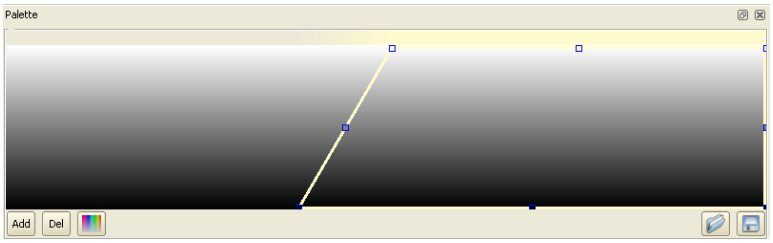


Figura 4.4: Eina de la paleta

Les principals eines per interactuar amb el model són la seva rotació i translació, que permeten explorar el model en tres dimensions des de diferents orientacions. També es disposa del retall de la caixa contenidora del

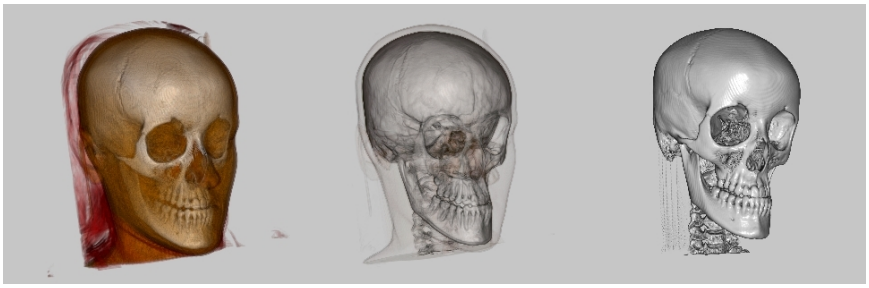
model, on movent les seves cares es pot veure l'interior del model, i també del retall del model per un pla amb qualsevol inclinació, anomenat pla de *clipping*.

L'aplicació està basada en llibreries, ja que així es poden utilitzar independentment en diferents projectes. VRMedVolVisDicom consta de les següents llibreries:

- **LibCaptacio** Defineix un món de vòxels (classe *Captacio*). Aquesta classe emmagatzema les dades necessàries per poder treballar amb el conjunt d'imatges que formen un estudi mèdic.
- **LibCore** Engloba el conjunt d'objectes bàsics de l'aplicació. Tots els objectes d'aquesta llibreria es defieneixen dins el *namespace* MGS. Entre d'altres proporciona objectes de: Càmera, geometria i TADs (Tipus Abstractes de Dades).
- **LibDicom** S'encarrega de totes les operacions amb els fitxers DICOM.
- **LibGUI** Ofereix principalment una especialització del widget *QGLWidget*.
- **LibRayCasting** Ofereix diverses tècniques de visualització, totes elles basades en l'algorisme de Ray Casting.
- **LibRenderObject** Proporciona objectes genèrics que es poden pintar.
- **LibTexture3d** Definició de l'interfície que han de complir tots els algorismes basats en Textures 3D.
- **LibVolumeRenderObj** Defineix un objecte volumètric renderitzable (classe *VolumeRenderObject*). Tots els algorismes de visualització de models volumètrics han d'heretar d'aquesta classe.

La interacció de l'aplicació està dissenyada per utilitzar un ratolí, i es divideix principalment en els menús de les finestres de Qt i l'ús del cursor dins la visualització 3D. La interacció amb els menús de les finestres és l'habitual, on es poden sel·leccionar accions i opcions. Dins la finestra de la paleta, podem manipular les diferents àrees representades per trapezoides movent els seus punts exteriors, que ens permeten veure diferents parts del model. Si una zona de densitats del model no està dins d'una d'aquestes àrees no es visualitzarà. Les inclinacions als extrems dret i esquerra dels intervals defineixen un gradient d'opacitat per als valors en aquests rangs.

Definir una bona funció de transferència és vital per a una bona visualització, especialment en visualització mèdica, ja que determina la qualitat de les imatges i ajuda a distingir les diferents seccions del cos. Podem veure diferents resultats del mateix model i diferents paletes a la figura 4.5.



(a) (b) (c)

Figura 4.5: El mateix model vist amb diferents paletes

La interacció dins la visualització 3D utilitza la posició en 2D del ratolí per moure l'objecte tridimensional. En la translació, l'objecte segueix la posició del cursor, per el que si volem fer zoom s'utilitza un altre botó i es mou el ratolí amunt o avall. El moviment és similar en la rotació, on només cal moure el ratolí per fer rotar el model. Dins el mode del retall de la caixa contenidora, s'ha de col·locar el cursor sobre el pla que es vol traslladar, i moure el ratolí amunt o avall per moure el pla. Per moure el pla de clipping el procés és una mica més complexe, ja que s'ha de moure el ratolí amunt i avall per fer girar el pla en l'eix X, i cap als costats per girar-lo en l'eix Y.

4.3.1 Fitxers d'entrada

Un cop l'usuari ha decidit les millors opcions de paleta i paràmetres de visualització per un Dicom, es poden guardar en un fitxer XML amb extensió *.vrmed* per a posteriors visualitzacions (Hi ha un exemple d'un arxiu *.vrmed* al codi 4.1). En especial, en aquest projecte, aquests arxius només es poden llegir per a obrir un model, ja que per crear-los es necessita una altra aplicació com VRMedVolVisDicom. Per crear aquest arxiu s'ha de sel·leccionar la opció de crear un nou projecte dins l'aplicació, on sel·leccionem tots els arxius Dicom que composen el model. Així, l'arxiu *.vrmed* conté la informació per trobar el fitxer de la paleta, la configuració i tots els arxius Dicom que composen el projecte.

L'arxiu de la paleta, amb una extensió *.plt*, conté la informació per a definir la funció de transferència. Quan modifiquem la paleta a través de la interfície podem salvar-la en un arxiu amb aquesta extensió. La primera linea conté el nombre de rangs definits, i a continuació trobem grups de 8 valors, un per a cada rang. El significat d'aquests valors és:

- Els quatre primers valors defineixen els vertexs del trapezoide, en ordre: esquerra inferior, esquerra superior, dreta superior i dreta inferior.
- Els altres quatre valors defineixen els components R,G,B i A.

Codi 4.1 Exemple de fitxer *vrmed*

```
<!DOCTYPE VRMED>
<Info>
 <Estudi Nom="name" Id="1" >
  <Captacion files="Dicom\obis\obis40, Dicom\obis\obis39,
  Dicom\obis\obis38, Dicom\obis\obis37 [...]
  Dicom\obis\obis1" type="0" />
  <Textura Config="configs\jaw.vrmed.conf"
  Tipus="0" Paleta="plt\jaw-bone.vrmed.plt" Seleccio="">
  </Textura>
 </Estudi>
```

El fitxer de configuració (extensió *.conf*) conté en les dues primeres línies el nombre de plans que s'utilitzaran en la visualització del model en baixa i alta resolució. A continuació conté la informació de la il·luminació del model: ambient i difusa.

5 Disseny i implementació

En aquesta secció ens centrarem en l'aplicació desenvolupada en el marc del projecte, VRJVolVisDicom, en les seves funcionalitats i arquitectura. La interacció de l'aplicació es divideix en dos apartats, la manipulació directa de l'objecte, i el control de l'aplicació a través de la interfície.

5.1 Arquitectura de l'aplicació

L'aplicació a desenvolupar en el marc del projecte ha de tenir unes funcionalitats similars a VRMedVolVisDicom, i ha d'utilitzar VR Juggler i Qt3D. Per desenvolupar-lo, es compta amb un sistema de *tracking* per posicionar l'observador, un *wanda* com a dispositiu d'entrada, i una pantalla de projecció que disposa de dos projectors per poder fer l'estéreo passiu.

L'esquema de la figura 5.1 mostra els elements bàsics de l'arquitectura de l'aplicació. VR Juggler rep tota la informació que envien els dispositius d'entrada, en aquest cas el *tracker* i el *wanda*. La nostra aplicació (VRApp) pot accedir a aquesta informació a través de la classe IOHandler de VR Juggler, i després realitzar tots els càlculs que necessiti. Quan VRApp envia a pintar l'escena, és VR Juggler qui ho rep, ja que també és ell qui controla els dispositius de sortida. Un cop VR Juggler té l'escena a pintar, genera les dues imatges que es necessiten per fer l'estèreo passiu i les envia als dispositius de sortida. La classe VRApp també ha de pintar les finestres de Qt3D dins el món virtual.

L'arquitectura es basa en un esquema multi-thread, un thread amb VR Juggler controlant tota l'entrada/sortida dels perifèrics, i un altre amb l'aplicació que controla la lògica i Qt3D, comunicant-se entre ells a través del mecanisme de signals/slots.

Les funcionalitats de visualització de l'aplicació de realitat virtual són les mateixes que en l'aplicació de PC, però evidentment canvia com s'interacciona amb l'aplicació ja que no tenim teclat ni ratolí, sinó un perifèric amb diversos botons anomenat *wanda*, i un *tracker* que segueix la nostra posició. També necessitem unes ulleres especials amb els vidres polaritzats,

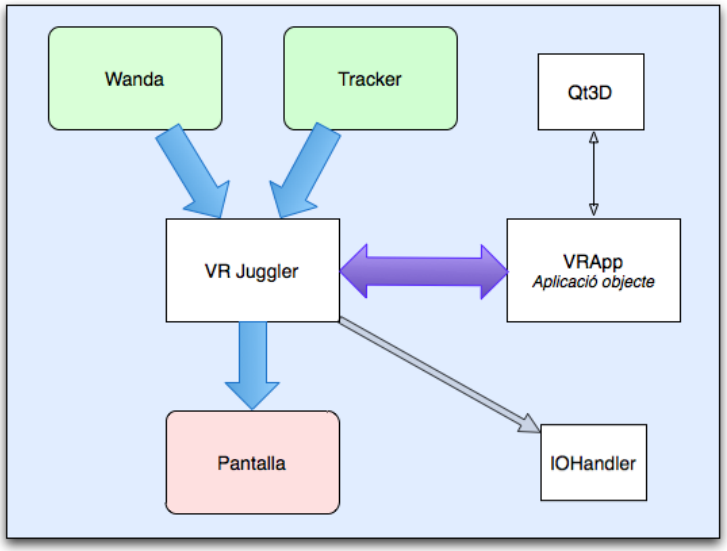


Figura 5.1: Arquitectura de VRJVolVisDicom

per a tenir una visió en estèreo; veure diferents imatges a cada ull i donar sensació de profunditat quan mirem l'objecte a la pantalla. D'aquesta manera, ens podem moure lliurement per l'habitació i l'objecte a la pantalla es mourà coordinadament segons el que fem, deixant-nos veure diferents parts de l'objecte sense utilitzar directament cap dispositiu d'entrada. A part de l'objecte, a la pantalla tenim la representació de la posició del nostre *wanda*, amb un avatar (una esfera) marcant la seva posició, i un raig assenyalant a on apuntem. Gràcies a això, podem interaccionar amb les finestres de l'aplicació, podent carregar l'objecte, canviant d'opcions o el que ens faci falta.

El diagrama de la figura 5.2 mostra les principals classes que intervenen en la interacció de l'aplicació. *VRApp* és la classe que fa d'aplicació objecte de VR Juggler, ja que hereda de *vrj::GlApp* i implementa totes les funcions comentades a l'apartat 4.1. Dins la funció *draw()* pinta l'escena de *VRJScene* i les finestres de Qt3D. *VRJScene* conté els objectes a pintar, *RayCastingGPU* i *Texture3dRender*, que hereden de la classe *VolumeRenderObject*. La classe *IOHandler* obté la informació dels dispositius d'entrada a través dels proxies que disposa VR Juggler, que arriba a *VolVisInteractionManager* a través de la interfície *InteractionManager*. La classe *VolVisInteractionManager* prioritza qui rep la interacció, primer es fa la manipulació i després la interacció de les finestres. També actualitza la informació dels dispositius a *VolumeRenderObjManipulation*, que manega l'estat de l'aplicació segons el botó que s'està utilitzant, i alhora dóna la informació dels dispositius a *VolumeRenderObjManipulator*. Aquesta classe conté un objecte *Volume-*

RenderObject i realitza la manipulació sobre ell en funció del moviment del dispositiu d'entrada i l'estat actualitzat anteriorment. L'objecte que manipula pot ser del tipus *RayCastingGPU* o *Texture3dRender*, però només es diferencien alhora de pintar-los. La classe *DicomLoader* és la que carrega els arxius en format Dicom i crea un objecte *Texture3dRenderDicom* amb tota la informació, i a partir d'ell es crea l'objecte *RayCastingGPU*.

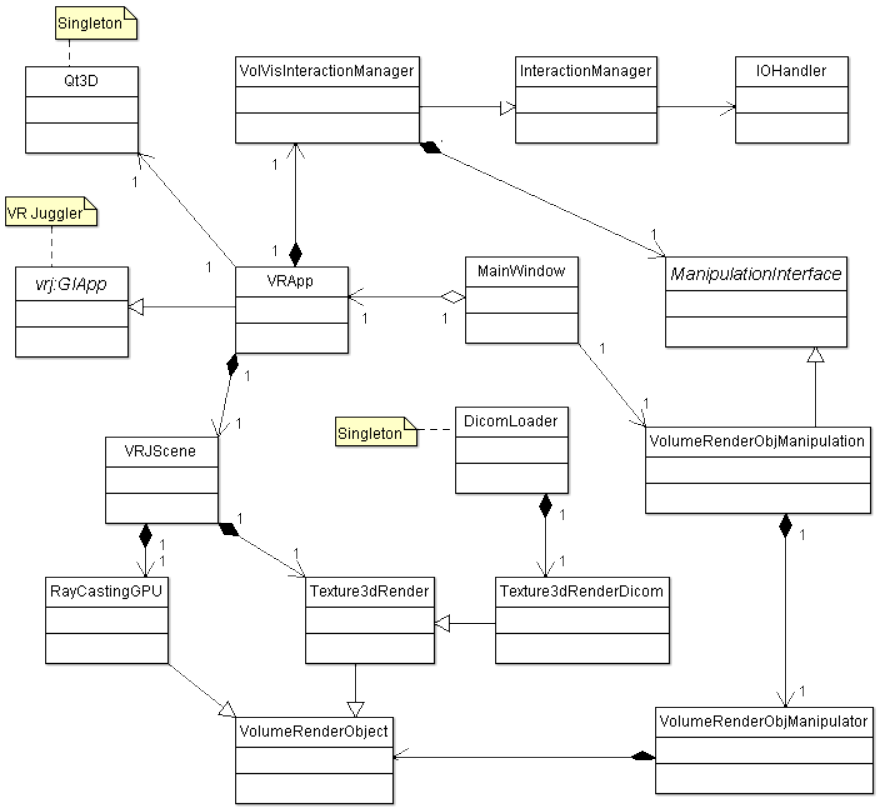


Figura 5.2: Diagrama de classes d'interacció

L'aplicació desenvolupada s'ha basat en un altre en què es disposava d'una estructura bàsica dissenyada, tanmateix no tenia programades les funcions d'interacció i es podria considerar com una proposta d'esquelet. Només estava preparat per visualitzar i manipular objectes de tipus textures 3D, pel que es va adaptar per poder-ho fer amb objectes *VolumeRenderObject* (S'explica com a la secció 5.4).

5.2 Control de l'aplicació

L'ús d'una interfície gràfica en un entorn de realitat virtual pot aportar tantes opcions com es vulgui, però és més lent que la manipulació directa. Les finestres s'han d'adaptar a les característiques del nou entorn, fent botons grans, utilitzant textos clars, etc. La interfície controlarà el mode de manipulació, els paràmetres de visualització i la funció de transferència.

En un principi les finestres estaven organitzades com en la figura 5.3. La finestra principal servia bàsicament per obrir altres finestres secundàries que sí tenien funcionalitats. La finestra on es podia canviar el mode de manipulació també servia per canviar el mode de visualització, i alhora obrir també altres finestres.

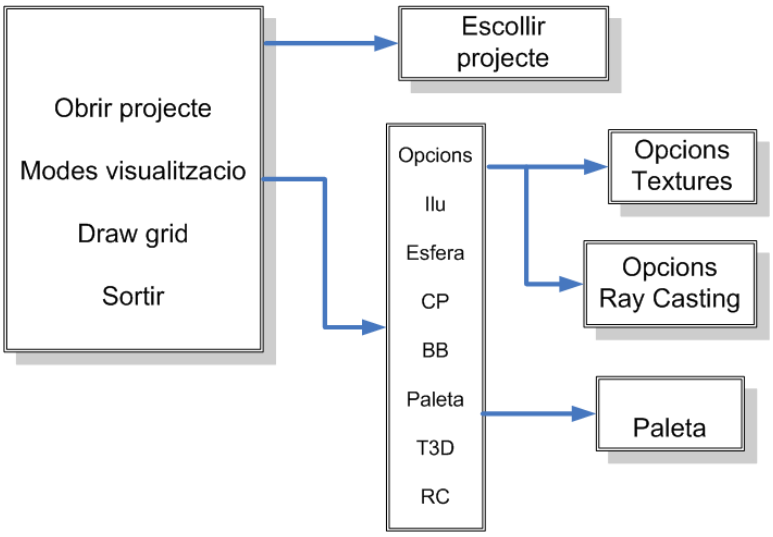


Figura 5.3: Organització inicial de finestres

Amb aquesta configuració van aparèixer diversos problemes, en particular es va veure que es necessitaven millorar els següents aspectes i crear noves metàfores per:

- **Tancar finestres**
- **Visualització de les eines**
- **Inconitzar finestres**

5.2.1 Tancar finestres

Per tal de tancar o minimitzar una finestra en concret s'havia de prèmer el botó corresponent, com en les finestres habituals. Una de les opcions per solucionar-ho era augmentar el tamany d'aquests botons, però ocuparien

massa espai i arribarien a molestar. Així que la opció que es va decidir va ser eliminar del tot la barra superior i tancar les finestres d'alguna altre manera. Per això, es va haver de crear una nova decoració de Qt3D que no envia a pintar res i, per tant, no es pot intersectar. Es van aprofitar els límits de la pantalla per fer desaparèixer les finestres, en quant es trasllada fora dels límits es tanca sense haver de prèmer cap altre botó. Així, s'aprofita més l'espai i es crea una metàfora com si l'exterior de la pantalla fós un altre món.

5.2.2 Visualització de les eines

Una altre característica que es va decidir afegir per aprofitar l'espai va ser la desaparició de l'objecte. Quan volem interactuar només amb la interfície, especialment amb les opcions i eines avançades, el model pot arribar a molestar. Per això es va usar un dels botons per poder fer desaparèixer el model en qualsevol mode de manipulació que fós. Així, si volem veure només finestres polsem el botó, i si volem tornar a veure el model només cal tornar-lo a prèmer. En relació amb això, quan s'obre la finestra d'opcions avançades ja es fa desaparèixer automàticament el model, en previsió de que es voldrà interactuar primer amb la finestra.

5.2.3 Iconitzar finestres

En contraposició a la desaparició del model per poder interactuar amb les finestres, també es va considerar la visualització únicament del model sense altres elements que estorbin (les finestres). Com hi pot haver diverses finestres obertes en un moment determinat no s'ha de deixar que l'usuari les tanqui una per una, sinó que s'ha de fer desaparèixer totes les finestres alhora sense arribar a tancar-les. Utilitzar un altre botó del wanda augmentava massa el nombre de botons a utilitzar, així que es va optar per un altre tipus d'interacció: un moviment de l'usuari. Es va intentar escollir un moviment que no fós difícil de realitzar ni de recordar, però que tampoc es fés de forma involuntària. Com en el tancament de finestres es va optar per utilitzar els límits de la pantalla per fer desaparèixer-les totes, així que només cal moure el raig fora de la pantalla per fer aparèixer o desaparèixer les finestres. Però no tots els límits de la pantalla es poden utilitzar, ja que per exemple si reposem la mà el raig s'envà més enllà del límit inferior. Per això es va utilitzar només el límit superior de la pantalla per fer la desaparició de les finestres, considerant els límits laterals també poc convenients. La resposta del sistema a aquest moviment, a part de la aparició/desaparició de les finestres, és incorporar a la pantalla un petit senyal que indica que les finestres han desaparegut. És una senyal senzilla per no molestar a l'usuari, però visible, una petita esfera que es pinta al límit superior esquerra de la pantalla. Quan apareixen les finestres, aquesta senyal desapareix.

En resum, s'han afegit les següents característiques:

- Es fan servir els límits de la pantalla per tancar les finestres.
- Amb un moviment cap al límit superior desapareixen totes les finestres.
- El botó verd del wanda serveix per fer aparèixer/desaparèixer el model.

5.2.4 Configuració de les finestres

Tot i les característiques afegides, la configuració de les finestres no era la millor possible. La finestra principal sempre havia d'estar oberta per poder tancar l'aplicació, tot i que la finestra de les opcions era més utilitzada. Per això es va canviar a una nova configuració. La finestra principal faria de barra d'eines amb les principals funcionalitats que a priori s'han d'utilitzar: obrir un projecte, escollir entre els tres modes de manipulació, obrir una finestra amb opcions avançades i sortir del programa. Aquesta finestra té forma horitzontal i està col·locada a la part baixa de la pantalla per ocupar un espai poc utilitzat però accessible. A la finestra d'opcions avançades trobem que podem escollir entre els algorismes de visualització, obrir les opcions d'aquests (segons quin estiguem utilitzant) i obrir la paleta. Es pot veure la nova configuració a la figura 5.4 i un exemple en funcionament a la figura 6.1.

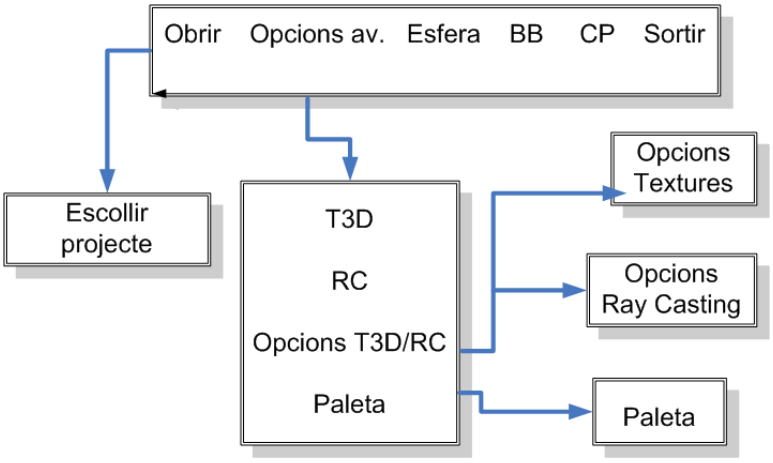


Figura 5.4: Configuració de les finestres

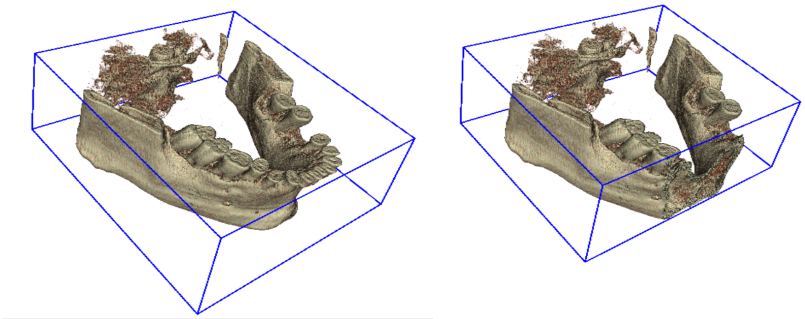
5.3 Manipulació directa

La manipulació directa compren el conjunt d'accions que es realitzen per inspeccionar els models actuant directament sobre ells. No intervé la interfície del programa, ha de ser el més senzilla i intuïtiva possible, per el que

ha d'utilitzar el mínim nombre de perifèrics que es pugui. La manipulació directa es basa en una interacció ràpida i intuïtiva, limitada a poques operacions i molt dependent del dispositiu d'entrada utilitzat i el seu nombre de botons i graus de llibertat. En aquest apartat incloem el posicionament de l'objecte, i el control del pla de clipping i la caixa contenidora. Per activar o canviar entre aquests modes de manipulació, s'ha d'accedir a la interfície, no poden estar tots activats alhora perquè no es podria diferenciar quin es vol usar en cada moment.

5.3.1 Caixa contenidora

La caixa contenidora, o *bounding box* en anglés, es refereix a la caixa mínima orientada segons els eixos que envolta qualsevol objecte, incloent tots els vòxels que són visibles. Moure una de les seves cares implica retallar el model, com es pot veure a la figura 5.5. El retallat de la caixa contenidora modifica els vòxels visibles i per tant ens deixa veure l'interior d'un model al llarg del pla de la caixa que estiguem movent. Quan s'activa aquest mode, es pinten només les arestes de la caixa contenidora al voltant de l'objecte perquè no interfereixi en la visualització. Es pot sel·leccionar quina cara es vol moure amb el wanda, només cal intersecar la cara que es desitja i prèmer un botó per a moure-la.



(a) Caixa original (b) Caixa retallada

Figura 5.5: Retall de la caixa contenidora

A l'aplicació de PC el moviment de les cares de la caixa es fa a partir d'un vector 2D amb el moviment del ratolí. Aquest vector es projecta al món virtual per aconseguir-lo en tres dimensions, i es fa el producte escalar entre ell i la normal de la cara. Amb això s'aconsegueix saber la direcció en què es vol moure la cara, si és positiu l'angle és inferior a 90°, el que vol dir que es vol augmentar el tamany de la caixa, i a l'inrevés si és negatiu. Aquesta tècnica per decidir la direcció de moviment de la cara és aplicable a l'aplicació en 3D, ja que es té el vector directament en 3D sense haver de transformar-lo.

S'han realitzat diferents implementacions i avaluacions d'usabilitat per veure quin tipus de manipulació va millor, el que ha permés millorar alguns aspectes. Al principi, es podien moure lliurement els plans endavant i enrere a velocitat constant, això és, que a qualsevol moviment del wanda el pla sempre es movia la mateixa distància. Això causava poca precisió i naturalitat, per el que es va decidir canviar fent-ho depenent de la velocitat a la que es mou el wanda. D'aquesta manera es pot controlar millor on es vol col·locar el pla. També es va millorar el control dels plans, ja que en un principi es podien moure més enllà de la caixa contenidora inicial o fins i tot atravessar l'altre pla paral·lel del cub. Per a poder veure quin és el pla que s'està sel·leccionant, es va implementar un sistema que pintava d'un color diferent la cara que s'estava interseccionant en qualsevol moment. L'algorisme bàsic es pot resumir en els següents passos:

- Comprovar si hi ha intersecció entre alguna cara i la direcció on apunta l'usuari.
- Si hi ha intersecció, pintar la cara d'un altre color. Si s'intersecta més d'una cara, la que ens interessa és la més propera a l'usuari.
- Si es clica un botó, mirar en quina direcció es mou el wanda. Calcular en quina direcció es vol moure el pla intersecat.
- Moure el pla en la direcció assignada comprovant les dimensions de la caixa.

Així es com van quedar mapejats els botons en el mode de la caixa contenidora:

- Botó vermell - Sense funció
- Botó groc - Moviment del pla
- Botó verd - Sense funció

5.3.2 Retall del pla de clipping

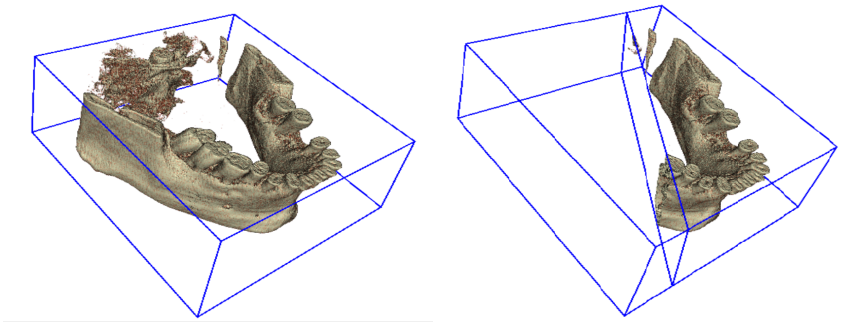
El pla de clipping és un pla qualsevol que atravessa el model i el talla en dues parts, només una de les quals serà visible si el *clipping mapping* està activat. Per tal de no pintar un pla infinit, i com tampoc tindria sentit, es retalla el pla dins el contingut de la caixa contenidora. El sentit del pla de clipping és poder veure l'interior del model tallat en qualsevol angle, no només segons els plans coordenats. Dins aquest mode, es pinta la caixa contenidora com en l'anterior, i dins es pinta el pla de clipping retallat, que es modifica segons el botó que es premi.

Proves prèvies

A l'aplicació de PC s'ha d'intersecar primer el pla de clipping i moure el ratolí. Si es mou en l'eix X, el pla gira en aquest eix sobre el seu baricentre, i igualment en l'eix Y. És un mètode poc intuïtiu i difícil d'usar, però és una de les poques solucions viables en una aplicació 2D. Es va portar aquesta tècnica directament a la nova aplicació, havent de moure el wanda en el pla XY. Es va veure que aquesta manipulació del pla no era gens natural, així que es va canviar per un pla que tingués com a normal la direcció on apunta el wanda. D'aquesta manera és fàcil preveure com acabarà el pla segons els nostres moviments. L'únic problema és que quan es prem el botó el pla fa un salt brusc a la nova posició si aquesta és molt diferent a la que ja tenia, però es un lleu problema inherent a aquesta tècnica.

Solució final

Per tal de millorar la sel·lecció del pla es va canviar que no s'hagués d'intersecar el pla de clipping, sinó la caixa contenidora. Així, es pinta d'un color diferent el pla quan es sel·lecciona la caixa, ja que anteriorment era difícil intersecar el pla segons en la posició que estava. També es va afegir i provar el trasllat del pla, ja que fins aleshores només es podia modificar la normal del pla i sempre creuava el centre de l'objecte. Es va afegir la funcionalitat en un altre botó del wanda, de manera que amb un botó es podia escollir la direcció del pla i amb l'altre la posició. A l'afegir la possibilitat de traslladar el pla es pot arribar a sortir de la caixa contenidora, cosa que no ens interessa, per el que es va retringir que el pla sempre fós dins la caixa amb una superfície mínima del 5%.



(a) Model original (b) Model retallat

Figura 5.6: Retall del pla de clipping

També es va comprovar que activar la opció del clipping mapping a través del GUI prenia massa temps, per el que es va optar per afegir un botó que quan s'intersecta la caixa contenidora activa o desactiva el clipping mapping.

Com és previsible que quan s'entri dins el mode del pla de clipping es vulgui activar el clipping mapping, aquest s'activa per defecte. Es pot veure el mode en funcionament a la figura 5.6. Així es com van quedar mapejats els botons en el mode del pla de clipping:

- Botó vermell - Translació del pla
- Botó groc - Orientació del pla
- Botó verd - Activar/Desactivar el clipping mapping

5.3.3 Posicionament de l'objecte

El posicionament de l'objecte comporta poder ubicar-lo lliurement respecte la càmera (el punt de vista de l'usuari), per el que s'han utilitzar transformacions geomètriques. Per transformacions geomètriques ens referim a translacions i rotacions, bàsics per a poder explorar completament un model tridimensional. Quan s'activa aquest mode apareix una esfera representada per filferros envoltant l'objecte per tal de saber en quin mode som. No obstant, com aquest mode serà previsiblement el més emprat, ens hem decidit per sel·leccionar-lo de manera predefinida, i quan no h hagi cap mode sel·leccionat, aquest serà el que funcionarà. A més, quan es sel·leccioni vía interfície el mode de transformacions geomètriques, sempre es dibuixarà l'esfera, i quan estigui sel·leccionat implícitament (per defecte) no es veurà.

Rotació

Per a poder rotar intuïtivament un objecte, aquest ha de girar sobre el seu centre independentment de la posició en què es trobi. Només es necessiten dos valors per a rotar qualsevol objecte en tres dimensions, que són l'eix i l'angle de gir. L'eix de gir el determina la perpendicular de la direcció en què l'usuari mou el dispositiu d'entrada, i l'angle depèn de quant l'hagi mogut.

Per calcular l'eix de gir es calcula el pla que l'usuari està considerant (pla α, veure figura 5.7), és a dir, el pla que tingui per normal el raig del wanda i passi pel centre de l'objecte. Amb aquest pla es pot obtenir la matriu de transformació A que el converteix al pla XY, on sabem que la perpendicular de qualsevol vector *(a,b)* és *(-b,a)*, ja que la component Z és zero i no afecta. Els passos bàsics a fer per calcular l'eix de gir són:

1. Calcular el vector direcció al pla α. S'han de restar les interseccions del raig del wanda i el pla α en dos moments diferents.
2. Calcular la transformació d'aquest vector al pla XY (Multiplicar per A).

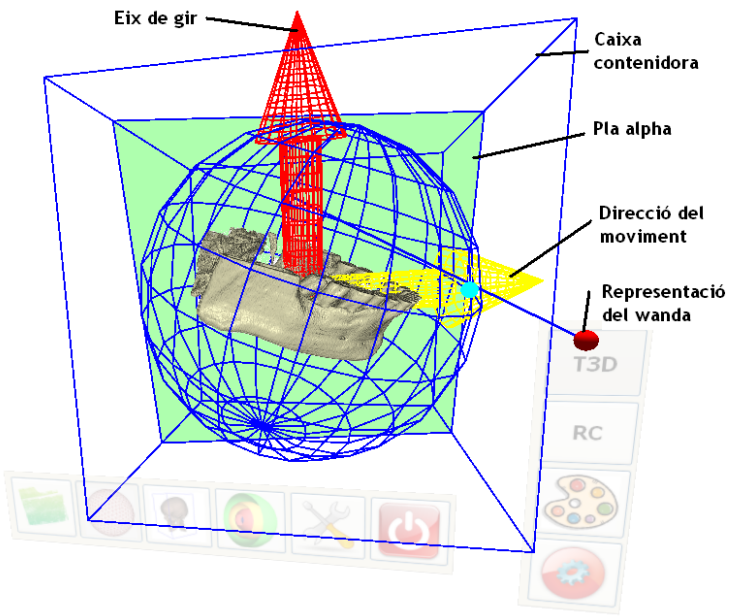


Figura 5.7: Diagrama del càlcul de la rotació

3. Obtenir el vector perpendicular dins el pla XY.
4. Calcular el valor del vector perpendicular en el pla A (Multiplicar per la inversa d'A).

Es pot veure la funció en C++ al codi 5.1.

Codi 5.1 Funció de rotació

```
void VolumeRenderObjManipulator::updateRotate(const
        MGS::LGPt3d &proj, const MGS::LGPt3d &projPrev){

    LGMat4x4 mat,rot;
    LGPt3d pXY,prevXY,pPerp,pMov;
    LGPt3d center = mTexture->getBoundingBox(true).Center();

    //traslladem al centre de la caixa contenidora
    mat.MatTranslation(-center.x(),-center.y(),-center.z());

    //Projectem els punts al pla XY.
    //mMatPlane és la matriu que transforma
    //el pla XY al pla alpha
    pXY = proj * !mMatPlane;
    prevXY = projPrev * !mMatPlane;
```

```
//Calculem el vector moviment i després la perpendicular
    LGVec3d moviment(prevXY,pXY);
    pPerp = LGPt3d(-moviment.y(),moviment.x(),0);

    //Tornem el vector al pla alpha
    pPerp = pPerp * mMatPlane;
    LGPt3d pZero(0,0,0);
    pZero = pZero * mMatPlane;
    perp = LGVec3d(pZero,pPerp);
    perp.Normalize();
    //Calculem l'angle de rotació segons la longitud del vector
    LGReal dist = proj.Distance(projPrev);

    //Rotem l'objecte sobre el seu centre
    rot.MatRotation(perp, dist * mOffset);
    mat = mat * rot;

    //El traslladem on era
    rot.MatTranslation(center);
    mat = mat * rot;
    mat = mTexture->getMatrixTransformation() * mat;
    mTexture->setMatrixTransformation(mat);
}
```

La rotació calculada s'afegeix a les anteriors a través de la concatenació de transformacions. Si està activada l'opció de visualitzar l'esfera de filferros, es veurà un punt que es mou conjuntament amb l'esfera que assenyala on s'ha fet la primera intersecció, per tal d'ajudar a l'usuari d'entendre millor com es mou l'objecte.

Una de les millores incorporades a aquest algorisme és que es continúi rotant encara que es deixi d'intersecar el raig amb l'objecte. En un principi es va fer al contrari però es va veure que s'havia de tornar a prèmer el botó masses cops per girar qualsevol model. En relació amb això, es va incorporar una última característica per fer girar automàticament el model. Per activar-lo s'ha de deixar de prèmer el botó del wanda mentre s'està en moviment, d'aquesta manera l'objecte es segueix movent en la mateixa direcció que ho feia. Es necessita definir una velocitat mínima a la que l'objecte comença a girar sol, ja que si no es desitja activar aquesta característica resulta molt molest. Per això, s'ha de definir una velocitat que no sigui molt baixa per no activar-la involuntàriament, ni molt alta ja que sinó costarà molt fer-la servir. Es van provar dues configuracions diferents:

- En la primera l'objecte seguia girant amb la mateixa direcció i velocitat

indefinidament fins que l'usuari tornava a començar una nova rotació.

- En la segona es va disminuïr progressivament la velocitat fins que l'objecte s'acabava aturant del tot, com si l'objecte hagués adquirit inèrcia (Codi 5.2).

És convenient definir una velocitat a la que s'aturarà, sino la velocitat pot ser molt propera a zero sense que es percebin els resultats, el que faria perdre eficiència.

Codi 5.2 Funció que fa rotar inercialment el model

```
void VolumeRenderObjManipulator::move(){
    switch (mMode)
    {
    case ROTATE:
        if(mLastMove.Distance(LGPt3d(0,0,0)) < 0.01)
            return;
        if(mInertia)
        {
            mCurTime = clock();
            double time = (mCurTime - mTime)/CLOCKS_PER_SEC;
            mLastMove = mLastMove * (1 - (time / 3));
        }
        mTime = mCurTime;
        updateRotate(mLastMove,LGPt3d(0,0,0));
        break;
    }
}
```

Finalment la segona opció va resultar millor, ja que amb la primera opció sempre s'havia d'aturar manualment el model cada vegada. Gràcies a aquesta opció, girar un objecte pot requerir menys moviments de l'usuari i pot ajudar a crear la sensació d'objecte real.

Translació

L'objecte s'ha de poder situar a qualsevol lloc que l'usuari vulgui, ja sigui per col·locar-lo on li vagi millor o per apropar-lo i veure'n més detalls. En un principi es va mapejar la posició del wanda amb la de l'objecte, de tal manera que si l'usuari el movia cap amunt l'objecte també es movia amunt. Aviat es va comprovar que no resultava òptim, ja que era bastant molest l'haver de moure continuament la mà. El més natural era no moure la mà de posició, sinó apuntar on es volia traslladar l'objecte, semblant a com es fa la rotació. D'aquesta manera es va aprofitar el càlcul del pla α de la rotació per

obtenir el vector de moviment. Aquest vector ens indica la direcció en què es vol moure l'objecte, amb el que només cal fer una transformació. Amb això s'aconsegueix moure el model en el pla de la pantalla, i per apropar-lo s'ha d'*estirar* de la pantalla. Aquest moviment d'apropar l'objecte pot ser intuitïu, però que no resulta gaire fàcil de realitzar correctament, ja que s'ha d'estirar en línia recta per evitar que el model es surti de les dimensions de la pantalla. Per això, es va decidir realitzar una nova funcionalitat que actués com a zoom per apropar i allunyar l'objecte. Es va aprofitar el joystick que incorpora el wanda per a realitzar aquesta funció. Per a utilitzar-lo només cal moure el joystick endavant per allunyar, i enrere per apropar.

En una de les proves es va detectar que el moviment de translació no quedava del tot natural perquè al moure's per el pla de la pantalla hi havia zones que es deixaven de veure. Això era perquè es movia sense tenir en compte l'usuari, de manera que la cara que mira a l'usuari no és sempre la mateixa. Per a solucionar-ho s'ha de fer rotar l'objecte amb el punt de gir sobre la posició de l'usuari, no al seu interior. D'aquesta manera es té una sensació més real de que el model està realment traslladant-se segons els nostres moviments.

Així es com van quedar mapejats els botons en el mode de transformacions geomètriques:

- Botó vermell - Translació de l'objecte, principalment en el pla de la pantalla
- Botó groc - Rotació de l'objecte
- Joystick - Apropar/Allunyar l'objecte

5.4 Visualització

En el camp de la visualització mèdica és molt important el resultat final de la reconstrucció de les dades, ja que ha de ser el més exacte possible. A més d'això, un entorn de RV fa que aquesta visualització s'hagi de realitzar d'una manera fluida i a temps real. Per això, a part de la visualització per textures 3d, s'afegeix al possibilitat de visualització amb ray casting i així poder comprovar quina tècnica funciona millor en aquestes condicions.

Per a poder visualitzar les dues tècniques es va fer un canvi en les classes que tractaven amb les textures 3d. *RayCastingGPU* és la classe que implementa l'algorisme de ray casting. La classe que manipula els objectes *Texture3dRender* (*Texture3dRenderManipulator*, ara renombrada a *VolumeRenderObjManipulator*) no podia tractar objectes del tipus *RayCastingGPU*, així que va passar a tractar objectes *VolumeRenderObject*, del qual hereten els dos. Amb aquest canvi n'hi ha prou de manipular un sol objecte, ja que aleshores per mitjà de l'herència ja s'utilitzarà l'algorisme de

pintat corresponent a l'objecte. Això es pot fer gràcies a que tant la classe *Texture3dRender* com *RayCastingGPU* sobreescriuen el mètode *drawGL()* de *VolumeRenderObject*, que es crida quan l'objecte s'ha de pintar.

Quan s'inicialitza el programa l'algorisme de visualització que s'utilitza és el de les textures 3d, però es pot canviar en qualsevol moment a través de la interfície. Dins les opcions avançades, s'han afegit dos botons auto-exclusius que permeten escollir entre les textures i el ray casting. Així, quan es canvia el mode de visualització es pot veure el mateix objecte en la posició que l'havíem deixat, ja que cada vegada s'envia la transformació geomètrica actual a l'altre objecte. També s'actualitza tota la informació referent a la bounding box i el pla de clipping, i així obtenir l'objecte en les mateixes condicions.

6 Anàlisi de resultats

En aquesta secció presentarem el funcionament final de l'aplicació, i una primera valoració experimental del resultat en funció de proves realitzades a alguns usuaris, que no arriba al nivell d'un test d'usabilitat. No hi ha hagut temps de realitzar proves amb metges, sinó que s'han realitzat un conjunt de rpoves als memebres del grup MOVING al llarg del projecte i en la seva etapa final.

6.1 Interacció

S'han intentat adaptar les interfícies gràfiques d'usuari a l'entorn de realitat virtual, tenint en compte els moviments més fàcils i usuals. La distribució de finestres ha col·locat les funcions més usades el més accessibles possible en una barra d'eines. Es pot veure el disseny final de les finestres a la figura 6.1, on es pot comprovar que no ocupen espai a l'hora de manipular el model. S'ha vist que quant menys s'hagi d'interaccionar amb les finestres millor, i que quan es faci no s'hagi d'apuntar a un lloc molt precís. En aquest sentit, s'han col·locat a les opcions avançades aquelles que requerien més coneixements com les que són més difícils d'usar, com la paleta. La distribució i col·locació de les finestres no varen donar cap problema durant les proves realitzades.

Un dels principals problemes que encara presenta la usabilitat de la interfície ve del widget per escollir el model a obrir. És un diàleg on s'ha de llegir el nom dels projectes existents i sel·leccionar el desitjat, el què pot arribar a ser difícil. Si s'augmenta el tamany de la lletra disminueix el nombre de projectes visibles, per el que no és una solució vàlida. S'hauria de redissenyar tota la finestra per adaptar-la a les necessitats de la realitat virtual, com un widget amb botons grans que mostri el nom del projecte i una imatge del mateix a mode de pre-visualització.

Es va comprovar que el moviment per fer desaparèixer totes les finestres pot resultar útil tot i que s'activa involuntàriament més del previst. Si s'està usant únicament la barra d'eines, o fins i tot les opcions avançades,

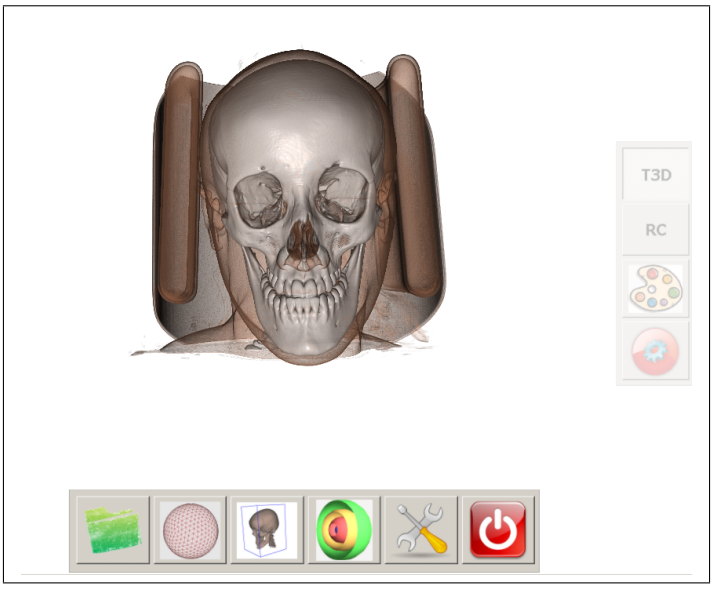


Figura 6.1: VRJVolVisicom amb la finestra principal i d'opcions avançades

pot arribar a ser més molest que útil. Per això, donada la configuració de finestres actual potser es podria solucionar eliminant aquesta funcionalitat, o realitzar una altra implementació que s'asseguri que l'usuari vol realment fer desaparèixer totes les finestres.

El tancament individual de les finestres resulta del tot intuïtiu, encara que presenta el problema de que es pot eliminar qualsevol finestra, per el què si es tanca la finestra principal no podrem obrir cap altre. És un problema poc important que depèn del comportament de l'usuari i que a priori no tindria perquè realitzar. No obstant, es podria realitzar una funcionalitat de *reset* de les finestres que les recol·loqués a la posició inicial.

6.2 Manipulació

Un dels principals objectius en quant a la manipulació era que fós intuïtiva i fàcil d'usar. Veiem els resultats dels diferents modes en què hem dividit la manipulació:

- Caixa contenidora: El retall de la *bounding box* resulta intuïtiu, ja que s'ha de moure la mà en el mateix sentit que el pla que es vol retallar i es mou a la mateixa velocitat. És fàcil saber quina cara estem sel·leccionant perquè es mostra de diferent color abans de tocar cap botó. Un dels principals problemes d'aquest mode és que no es pot orientar el model mentre estem en ell, de manera que s'ha d'entrar al mode de transformacions geomètriques cada vegada que es vol veure

d'un angle diferent. Això és perquè no es podria diferenciar quina acció vol fer l'usuari si estan els dos modes actius, per el què les possibles solucions comportarien tenir els dos modes actius i utilitzar un o l'altre en funció d'on es clica dins el model. Però això seria contraproduent, ja que no es pot clicar fàcilment en una zona petita.

- Pla de clipping: El pla és fàcil d'interseccionar en qualsevol posició, donat que només cal intersecar l'àrea de la caixa contenidora. També resulta fàcil col·locar el pla en la posició desitjada perquè només cal apuntar perpendicularment a ell. El botó que activa el clipping mapping resulta molt útil i es pot activar en qualsevol moment, ja sigui abans o després de posicionar el pla. Hi ha el mateix problema que en la caixa contenidora, el de no poder girar l'objecte dins aquest mateix mode, però no és tan necessari perquè aquí podem veure més angles interiors.

- Transformacions geomètriques: La rotació és intuïtiva, tot i que de vegades costa una mica col·locar el model en la posició exacte que es vol. És a causa del tipus de dispositiu d'entrada, ja que si s'utilitzés per exemple un guant de dades seria més fàcil d'usar. Si en primera instància no resulta fàcil d'entendre, es pot activar el dibuix de l'esfera que ajuda a veure el moviment que fa l'objecte. L'activació del moviment inercial pot resultar complicat, s'hauria d'acabar d'ajustar la velocitat a la que s'inicia o es podria deixar escollir-la a l'usuari. També, donat el poc profit que es pot treure d'aquesta funcionalitat, es podria simplement eliminar.

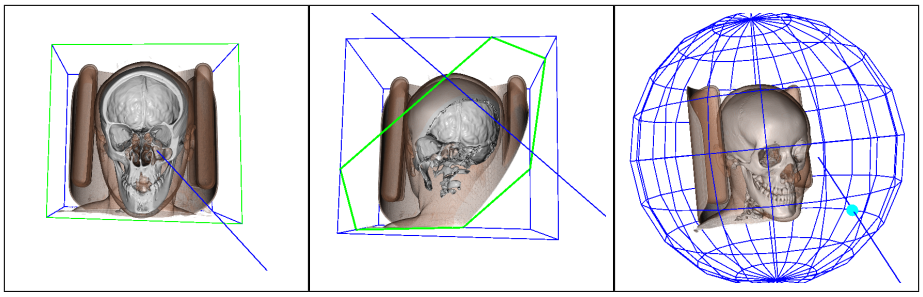
  En quant a la translació el principal problema és poder moure l'objecte fora de l'espai visible, de manera que es podria perdre i no recuperar. Per resoldre-ho es pot utilitzar la funció de la tecla Q, que inicialitza tots els paràmetres i dibuixa l'objecte al centre de la pantalla. Pot passar el mateix amb la funcionalitat del joystick, el zoom, perquè pot col·locar l'objecte massa lluny o massa aprop.

Es poden veure els diferents modes de manipulació en funcionament a la figura A.3.

6.3 Visualització

En quant a la visualització dels models mèdics hem de comparar els dos algorismes presents a l'aplicació:

- Textures 3D: En la visualització amb textures 3d apareixen problemes en models relativament grans en dades. La latència entre frames augmenta de tal manera que fa difícil interaccionar amb el model, ja



(a) Caixa contenidora (b) Pla de clipping (c) Esfera de la rotació

Figura 6.2: Modes de manipulació

que l'objecte es pot trobar en un lloc diferent del què ens mostra la pantalla. A part d'això, apareixen problemes propis de l'algorisme, com és que es notin els diferents plans que es pinten, com veiem a la figura 2.13. Això es pot millorar augmentant el nombre de plans, però això fa augmentar encara més la latència. Els plans generen un altre tipus d'efecte negatiu, i és que a l'estar orientats perpendicularment a la direcció de visió, es mouen sempre que es mou l'usuari. Això genera un efecte d'*aigües*, ja que el tracker detecta qualsevol mínim moviment encara que l'usuari intenti estar quiet. Aquest defecte és abstant perceptible i genera una mala visualització del model.

- Ray Casting: En l'algorisme de ray casting també apareix una gran latència en models amb grans dades. No obstant, no apareixen els defectes en la visualització de les textures 3d degut a que no utilitza plans, així que es pot considerar una correcta visualització.

D'aquesta manera, l'algorisme de visualització que resulta millor per les condicions de la realitat virtual resulta ser el de ray casting, gràcies a no generar defectes. Tot i això, si es millorés l'algorisme de les textures 3d per evitar aquests defectes en la visualització, també es podria considerar un algorisme apte.

7 Conclusions

A la finalització del projecte s'ha complert l'objectiu principal de portar una aplicació de visualització mèdica (VRMedVolVisDicom) a un entorn de realitat virtual. S'ha utilitzat VR Juggler com a plataforma per a facilitar la migració. Això independitza l'aplicació dels perifèrics disponibles, i facilita la seva portabilitat i execució en diferents sistemes de RV.

Per tant, s'han assolit els objectius proposats a l'inici del projecte:

- S'han dissenyat les tècniques d'interacció pensant en les necessitats que tindria un metge a l'utilitzar l'aplicació, de manera que les accions més comunes es puguin realitzar fàcilment usant una manipulació directa.

- S'han usat moviments naturals de la mà per fer-ho més intuïtiu i així minimitzar el temps d'aprenentatge. Les funcionalitats més específiques i menys usades s'han inclós dins la interfície gràfica del món virtual usant la llibreria Qt3D. Degut a que està desenvolupada en Qt seria molt senzill afegir noves funcionalitats en un futur, i permetent un canvi de disseny en l'interacció.

- S'ha adaptat l'aplicació per a poder utilitzar diferents tècniques de visualització, analitzant quina és més apropiada per a la RV. S'ha desenvolupat una aplicació que compleix els requeriments descrits anteriorment.

- S'ha dut a terme un petit anàlisi d'usabilitat per a comprovar els resultats. S'han vist els problemes que presenta l'aplicació i es proposen algunes solucions.

Com a treball futur, primer s'han de resoldre els problemes que s'han trobat a l'aplicació:

- Per evitar la latència que es genera amb alguns models, es podria usar un mode de visualització de baixa resolució que entraria en funcionament en determinats moments, com quan es fa la visualització interactiva dels objectes.

- El moviment que fa desaparèixer totes les finestres no és adeqüat. No n'hi ha prou amb que el raig traspassi les fronteres de la pantalla, s'ha d'implementar de manera que comprovi tot el moviment.
- El moviment automàtic inercial pot generar més confusió que avantatges. S'hauria de deixar escollir a l'usuari si el vol o no.
- S'ha de realitzar un anàlisi d'usabilitat més precís, i amb metges.

8 Anàlisi econòmic

En aquest apartat analitzarem el cost econòmic associat al desenvolupament d'aquest projecte. S'ha tingut en compte quin seria el cost de personal desglossat en les diferents tasques a realitzar. A més a més s'han tingut en compte les infrastructures necessàries per dur a terme el projecte.

8.1 Cost del personal

S'ha considerat que en el desenvolupament del projecte han intervingut un programador i un analista. L'estimació del cost per hora de cadascú és de 40 €/hora per al programador i 90 €/hora per a l'analista.

A la següent taula s'assignen les diferents tasques al programador o l'analista i es calcula el cost del personal.

Taula 8.1: Cost del personal

Tasca	Hores	Responsable	Cost
Anàlisi	40	Analista	1600€
Disseny	40	Analista	1600€
Implementació manipulació	200	Programador	8000€
Implementació interfície	50	Programador	2000€
Implementació visualització	40	Programador	1600€
Proves	80	Programador	3200€
Documentació	150	Programador	6000€
Total	600		18600€

8.2 Cost de l'infraestructura

Com a costs de hardware es té en compte el PC on s'ha desenvolupat l'aplicació i el sistema de realitat virtual on s'ha provat principalment el fun-

cionament. No es compta el preu del software amb el què s'ha desenvolupat el projecte, que inclou: Microsoft Windows XP, Microsoft Visual Studio, Qt i TortoiseSVN.

S'ha utilitzat un PC amb aproximadament un preu de 600€, amb les següents característiques: Pentium 4 a 3.4 GHz, 1.50GB de RAM i una targeta gràfica NVidia GeForce 6800 Ultra de 256MB. El lloguer del sistema de realitat virtual és de 80 hores, amb un preu per hora de 25€/hora, el que fa 2000€.

Es càlcul total de l'infraestructura és a la taula següent:

Taula 8.2: Cost en infraestructura

Component	Cost
PC	600€
Lloguer sistema RV	2000€
Total	2600€

8.3 Cost total

El cost total del projecte és de 21200€.

Taula 8.3: Cost total

Concepte	Cost
Cost del personal	18600€
Cost de l'infraestructura	2600€
Total	21200€

8.4 Planificació

La duració total del projecte ha estat de 4 mesos, de mitjans de Setembre 2008 a mitjans de Gener 2009. L'horari de treball considerat és de 8 hores al dia. La planificació de les diferents etapes del projecte és a la figura 8.1.

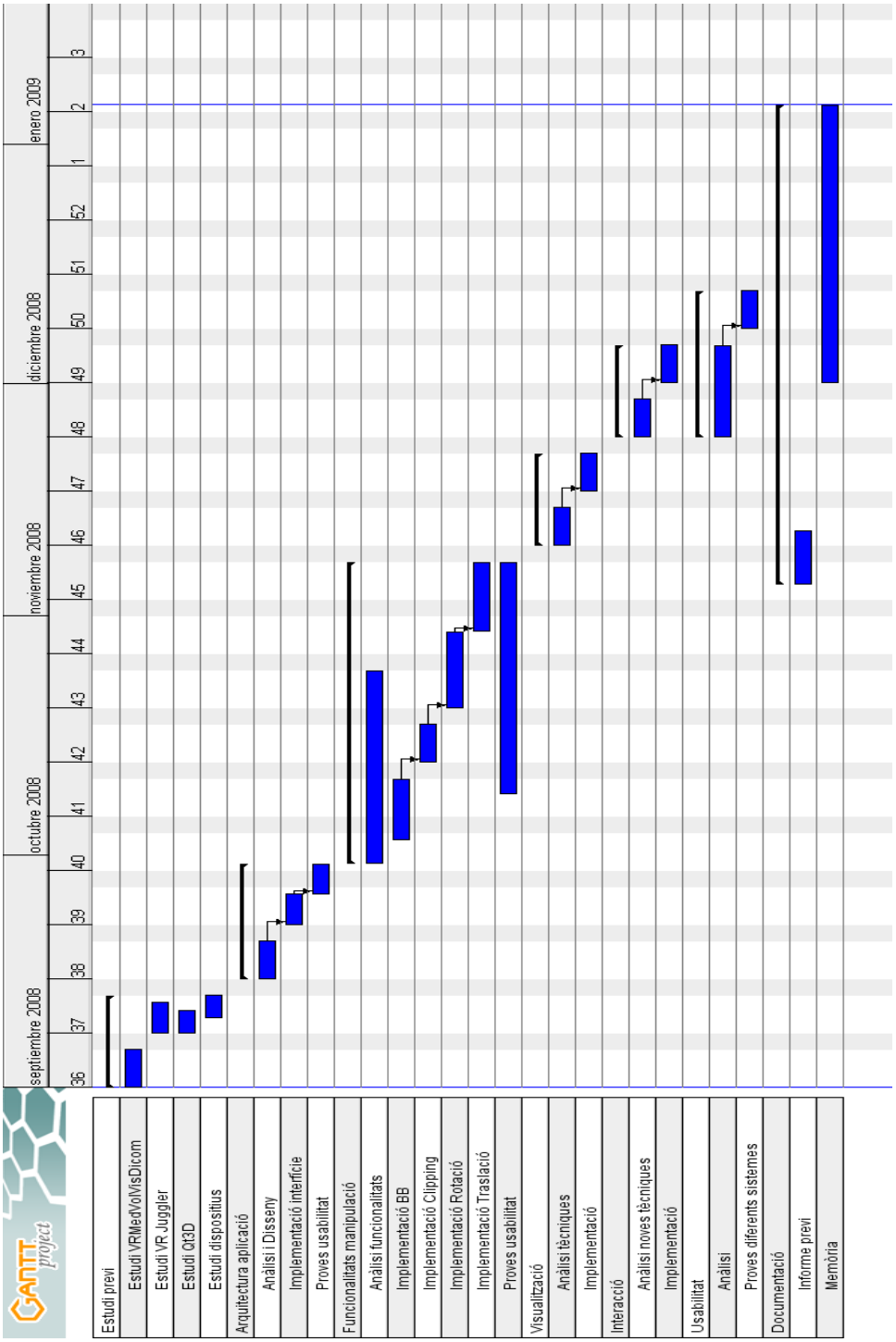


Figura 8.1: Planificació del projecte

Bibliografia

[1] Stone, R. J. 1993. Virtual reality systems. Edited by R.A. Earnshaw, M.A. Gigante, and H. Jones. Virtual Reality: A tool for telepresence and human factors research. *London: Academic Press.*

[2] K. Engel, M. Hadwiger, J. Kniss, C. Rezk-Salama i D. Weiskopf. Real-Time Volume Graphics. 2006

[3] The Modeling, Visualization, Interaction and Virtual Reality Research Group. *http://www.lsi.upc.edu/∼moving*

[4] CD Visualització i Interfícies Gràfiques (VIG). C. Andújar, P. Brunet, M. Fairén, I. Navazo, A. Vinacua.

[5] OpenGL *http://www.opengl.org*

[6] VR Juggler *http://www.vrjuggler.org*

[7] XVR, VRMedia *http://www.vrmedia.it*

[8] C. Andujar, M. Fairén i F. Argelaguet. A Cost-effective Approach for Developing Application-control GUIs for Virtual Environments. *Proceedings of the 2006 IEEE Symposium on 3D User Interfaces.*

[9] D. A. Bowman, E. Kruijff, J. J. LaViola i I. Poupyrev. 3D User Interfaces: Theory and Practice. 2004

[10] Jens Krüger and Rüdiger Westermann. Acceleration Techniques for GPU-based Volume Rendering. *Proceedings IEEE Visualization 2003*

[11] R. Taylor, S. Llavallée i G. Burdea. Computed-Integrated Surgery. The MIT Press, 1996.

[12] Foley, Van Dam and alt. Computer Graphics. Principles and Practice. Addison-Wesley,1991.

[13] *http://www.wandavr.com/*

[14] *http://www.elrincondejavier.net/*

[15] *http://www.sgi.com/products/software/performer/*

[16] *http://www.inition.co.uk/*

Annexes

Manual d'usuari

L'aplicació VRJVolVisDicom permet visualitzar i manipular fitxers en format Dicom (Digital Imaging and Communication in Medicine), però necessita un fitxer d'entrada *.vrmed* per a carregar tot el projecte. Aquesta aplicació no permet crear projectes *.vrmed*, així que s'ha d'utilitzar un altre programa per crear-los (com per exemple, VRMedVolVisDicom).

L'aplicació està dissenyada per utilitzar-se en un sistema de realitat virtual, amb un wanda com a dispositiu d'entrada i un sistema de *tracking* de la posició de l'observador. Tot i això, es pot utilitzar amb qualsevol altre configuració. Al llarg del manual es considera que s'està utilitzant un wanda com a dispositiu d'entrada amb 3 botons i un joystick. La representació del wanda dins el món virtual apareix com una esfera vermella indicant la posició de la nostra mà, i un raig blau que apunta on nosaltres apuntem. Els botons del wanda estan mapejats de la següent forma:

- Primer botó - Botó vermell
- Segon botó - Botó groc
- Tercer botó - Botó verd

A continuació explicarem com s'interacciona amb la interfície, per després començar a explicar les funcionalitats de l'aplicació. Això inclou la càrrega d'un projecte per a poder visualitzar-lo, les opcions bàsiques de manipulació directa, i finalment com utilitzar les opcions avançades.

A.1 Interacció amb la interfície

La interfície és molt semblant a la usada habitualment en d'altres aplicacions, per el que el funcionament general és el mateix. La principal diferència és que la barra superior amb les opcions de tancar la finestra i d'altres no existeix. No es pot minimitzar ni maximitzar cap finestra, però si es pot tancar.

Per interaccionar amb les finestres s'utilitzen només dos botons del wanda: amb el botó vermell premut controlem la posició de la finestra fins que l'alliberem, i el botó groc serveix per clicar dins la finestra, com si es tractés del botó dret del ratolí.

Per tancar una finestra només cal traslladar-la (botó vermell) fora de la pantalla cap amunt o els costats. Si volem fer desaparèixer momentàniament totes les finestres hem d'apuntar cap amunt amb el wanda sense prèmer cap botó. Aleshores apareixerà una petita senyal vermella a la cantonada superior esquerra indicant que les finestres són invisibles. Per tornar a visualitzar les finestres, s'ha de repetir el mateix moviment cap amunt.

A.2 Càrrega d'un projecte

Quan obrim l'aplicació només veiem la barra d'eines de la figura A.1. Per a carregar un projecte existent hem de clicar al primer botó i apareixerà un diàleg (Figura A.2) on podem escollir el fitxer *vrmed* que desitjem.

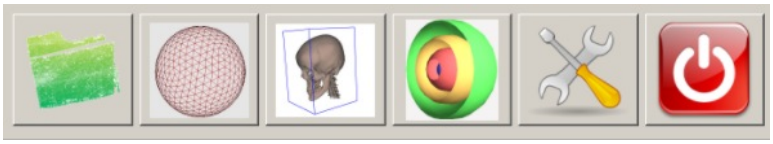


Figura A.1: Barra d'eines

Sel·leccionem l'arxiu que volem obrir i cliquem el botó "Open". Al cap d'un temps apareixerà el model al centre de la pantalla.

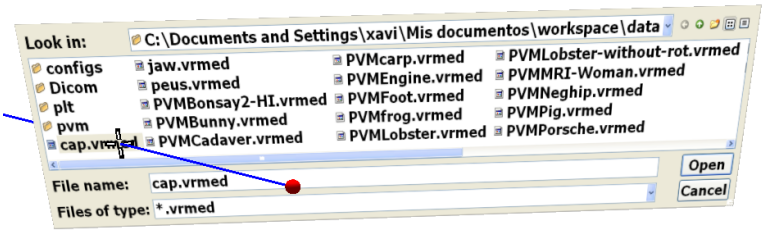


Figura A.2: Diàleg per obrir un projecte

A.3 Manipulació del model

Per manipular directament el model tenim tres modes diferents: posicionament de l'objecte, retall de la caixa contenidora, i retall del pla de clipping. Els tres modes es poden activar a través de la barra d'eines, segons els botons de la figura A.3. Per desactivar un mode es torna a clicar sobre

el seu botó, o si el que volem és activar un altre mode el podem sel·leccionar directament i l'altre s'elimina sol.

(a) Posicionament (b) Caixa contenidora (c) Pla de clipping

Figura A.3: Botons dels modes de manipulació

A.3.1 Posicionament de l'objecte

Per sel·leccionar aquest mode no cal prémer el botó de la barra d'eines ja que està sel·leccionat per defecte quan no n'hi ha cap en ús. Només quan es sel·lecciona el botó apareix una esfera al voltant del model que ajuda a compendre el moviment que fa l'objecte. Quan aquest mode està en funcionament podem realitzar dos tipus de moviments sobre el model. Abans de clicar cap botó per moure el model sempre hem d'apuntar cap a ell, fent intersecció entre ell i el raig que representa el wanda. Per fer rotar el model s'ha de clicar el botó groc, i per traslladar-lo el vermell. Mentre mantinguem el botó clicat, estarem movent el model.

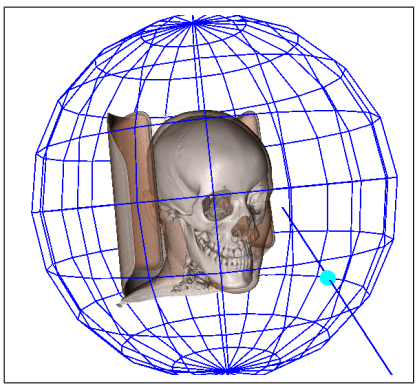


Figura A.4: Mode de posicionament

A.3.2 Caixa contenidora

Quan activem aquest mode es dibuixa la caixa al voltant de l'objecte. La cara que estem intersectant es pinta d'un color diferent, de manera que és la que controlarem si premem el botó groc. Mentre el mantinguem apretat podem controlar amb el nostre moviment la posició d'aquesta cara. Només podrem controlar les cares que tinguem mirant cap a nosaltres, les més properes, si volem moure una altre cara hem de rotar l'objecte amb el mode de posicionament.

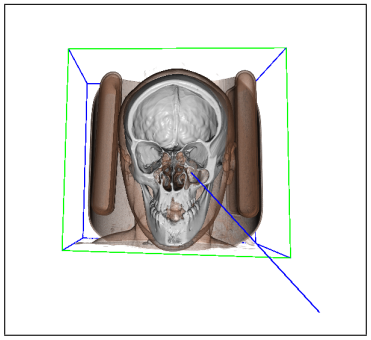


Figura A.5: Mode de la caixa contenidora

A.3.3 Pla de clipping

Dins el mode del pla de clipping es veurà la mateixa caixa contenidora que a l'apartat anterior, a més d'un pla a l'interior que talla el model. Quan activem el mode, la meitat del model desapareix per aquest pla, permetent veure'n l'interior. Podem fer aparèixer/desaparèixer aquesta meitat alternativament amb el botó verd mentre apuntem al model. Per modificar el pla tenim els altres dos botons, amb el botó groc podem controlar la direcció del pla, que serà perpendicular a la direcció on apuntem. Amb el botó vermell podem moure paral·lelament el pla dins els limits de la caixa.

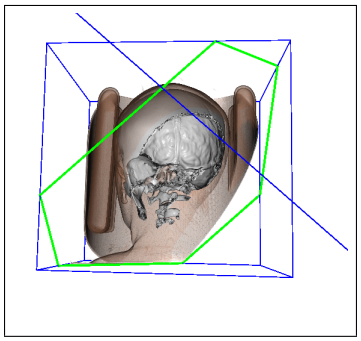


Figura A.6: Mode del pla de clipping

A.4 Opcions avançades

Quan premem el cinqué botó de la barra d'eines apareix un altre menú a la part dreta de la pantalla, les opcions avançades (Figura A.7). En ell podem escollir entre dos algorismes de visualització, les textures 3d (T3D) i el ray casting (RC). Només cal prémer el botó corresponent per veure el model amb la visualització corresponent. Els altres dos botons son per obrir la funcionalitat de la paleta i les opcions específiques de cada algorisme de visualització.

Figura A.7: Opcions avançades

A.4.1 La paleta

Per facilitar el reconeixement de les diferents regions que formen el model, s'utilitza una funció de transferència que determina el color i opacitat d'un vòxel donat el seu valor de densitat. El diàleg de la paleta (Figura A.8) permet definir una funció de transferència amb diversos rangs, permeten definir el color i opacitat per a cada rang. També permet carregar i guardar funcions de transferència al disc.

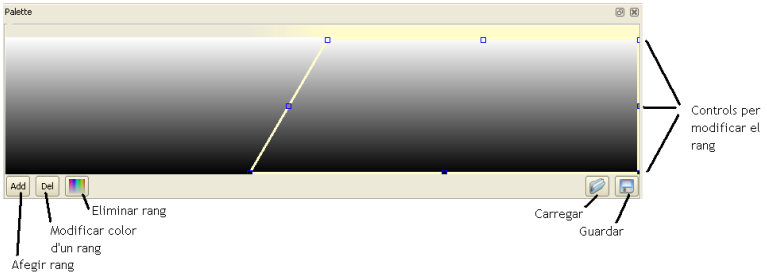


Figura A.8: Diàleg de la paleta

Amb el botó groc del wanda podem moure lliurement els punts de control

de cada rang.

A.4.2 Opcions de visualització

La variació d'aquestes opcions modifica la visualització del model. Les opcions que apareixen dins la finestra canvien segons l'algorisme que s'estigui emprant en el moment d'obrir-la.

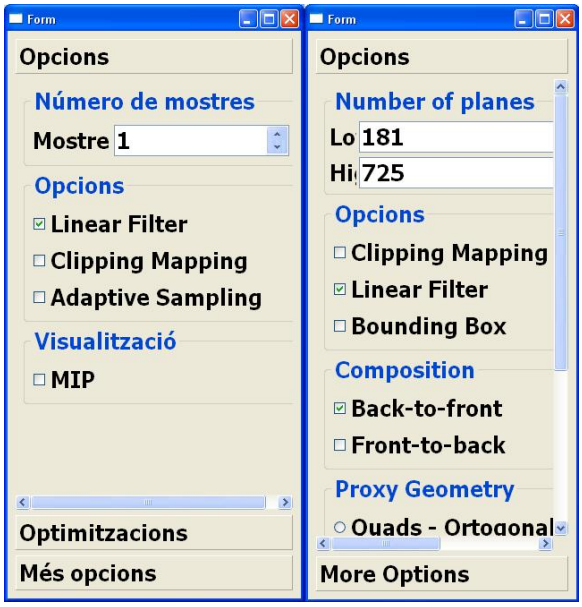


Figura A.9: Opcions de ray casting i textures 3d

Algunes de les opcions que apareixen en la finestra de ray casting son:

- **Número de mostres** que es prenen per cada vòxel. A més mostres millor visualització, pero l'aplicació tindrà un rendiment pitjor.
- **Linear Filter**. Activa la interpolació trilineal per obtenir visualitzacions més suaus. Si està desactivada s'aprecien els vòxels.
- **Clipping mapping**. Activa o desactiva el retallat per el pla de clipping.
- **Iluminació**. Es poden carregar diferents tipus d'iluminació per augmentar la calitat i realisme del model.

Algunes opcions de visualització del ray casting també apareixen en les textures 3d, d'altres són pròpies d'aquest menú:

- **Número de plans** utilitzats per a la visualització del volum. A més plans millor visualització, pero l'aplicació tindrà un rendiment pitjor.

- **Proxy geometry**. Modifica la geometria en que es col·loquen els plans de visualització.