EPS
Escola Politècnica
UdG Superior

Projecte/Treball Fi de Carrera

Estudi: Eng. Tècn. Informàtica de Sistemes. Pla 2001

Títol: Integració d'un visualitzador de volums en una plataforma de reconstrucció de mapes de fibres del cervell.

Document: Memòria

Alumne: Martí Reig Jacomet

Director/Tutor: Imma Boada
Departament: Informàtica i Matemàtica Aplicada
Àrea: LSI

Convocatòria Setembre 2009

Índex

Capítol 1. Introducció i Objectius.....6

1.1 Antecedents 6

1.2 La plataforma DTIWeb 7

1.3 Objectius 8

1.4 Planificació 9

1.4.1 Estudi de l'entorn de treball..... 9

1.4.2 Implementació 9

1.5 Metodologia 9

1.6 Temporalització..... 11

1.7 Estructura del document..... 14

Capítol 2. Fonaments teòrics.....15

2.1 Els models de DTI 15

2.2 Processament i visualització de les dades de DTI 17

2.3 Visualització de Volums..... 20

2.3.1 Algorismes de Visualització Directa de Volums 20

2.4 Funció de transferència..... 21

2.5 Composició de colors 23

2.6 Ray Casting 24

Capítol 3 .Fonaments Pràctics 25

3.1Eines de Programació..... 25

Java..... 25

Java 3D 26

JFC Swing 26

Eclipse..... 27

NetBeans 27

3.2 La Plataforma DTIWeb 28

Introducció 28

Estructura del DTIWeb 29

Capítol 4. Anàlisis i Disseny de l'Aplicació 31

4.1 Introducció 31

4.2 Anàlisis de Requisits 32
Casos d'Ús 32
4.3 Disseny 37
Diagrama de Classes 37
Descripció de les Classes 39
4.4 Implementació 47
Capítol 5. Interfícies 52
Capítol 6. Resultats 57
6.1 Ordinador de Proves 57
6.2 Proves i Resultats 57
Prova 1 57
Prova 2 59
Prova 3 62
Prova 4 63
Prova 5 64
Prova 6 65
6.3 Avaluacions 67
Capítol 7. Millores i Treball Futur 68
Capítol 8. Conclusions 69
Capítol 9. Webgrafia 70

Agraïments

En aquest capítol, voldria mostrar el meu afectuós agraïment a:

- Imma Boada.
- Ferran Prados.
- Ester Bonmatí.
- Alex Moreno.
- Cristina Diaz.
- Totes les persones que m'han donat suport a tirar-lo endavant.

Capítol 1. Introducció i Objectius

1.1 Antecedents

L'ús de la informàtica com una eina més en el camp de la medicina és una realitat que va en augment en els nostres dies. Els dispositius mèdics actuals permeten obtenir informació de qualsevol part de l'organisme d'un pacient i representar-la en forma d'imatges. El procés d'interpretació d'aquestes dades mèdiques és el que s'anomena diagnòstic per la imatge, i és on entre en joc l'ús de la informàtica.

El tensor per difusió (abreviat DTI de l'anglès Diffusion Tensor Imaging) és una nova modalitat de ressonància magnètica (RM) a partir de la qual es pot obtenir informació de la circulació de la matèria blanca en el cervell. A partir d'un aparell de RM es captura informació del pacient en diferents direccions. La informació que es recull en cada direcció es representa en una imatge diferent, de forma que per cada punt mostrejat tenim tantes imatges com direccions s'han tingut en compte. La informació del DTI es representa mitjançant un tensor de difusió, que representa la difusió de les molècules d'aigua en una unitat mínima de volum anomenada vòxel. A partir de la representació d'aquest tensor, s'obté informació de com són les fibres en l'interior del vòxel I es poden generar mapes de matèria blanca com el que es mostra en la Figura 1.1 i Figura 1.2.

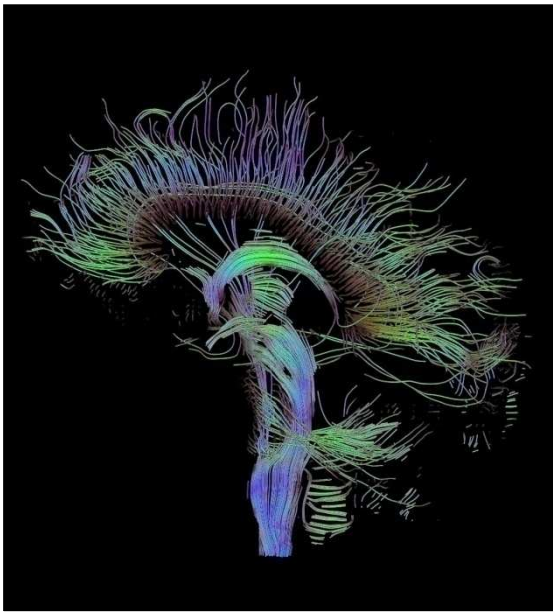

Figura 1.1. Reconstrucció tractogràfica de les connexions neuronals via DTI.

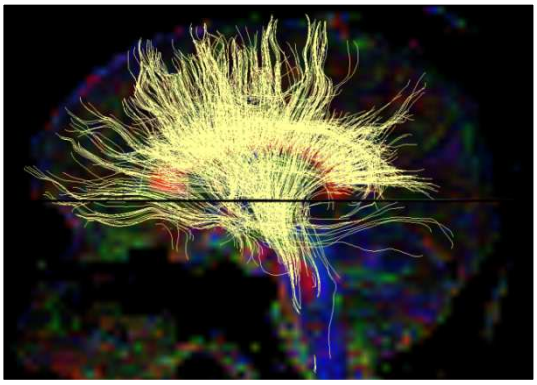

Figura 1.2. Imatge on es pot apreciar gran quantitat de fibres de cervell, imatge creada via DTIWeb.

Per poder generar els mapes de fibres s'utilitzen unes tècniques especialitzades anomenades tècniques de tracking. Aquestes tècniques el que fan es reconstruir les fibres que passen per cada un dels vòxels del model de volum que tenim. Un dels principals problemes d'aquest mètodes de tracking es la interpretació del mapes que generen. Sovint són mapes formats per milions de fibres i és fa difícil saber cada zona a quina estructura anatòmica del cervell es correspon.

Una solució a aquests problema seria combinar els mapes de fibres amb tècniques de visualització 3D que permetin representar l'estructura del cervell que s'està examinant.

1.2 La plataforma DTIWeb

La plataforma DTIWeb és una plataforma desenvolupada conjuntament entre el Laboratori de Gràfics i Imatge de la Universitat de Girona i d'Institut de Diagnòstic per la imatge de l'Hospital Josep Trueta de Girona. Aquesta plataforma integra els mètodes bàsics de reconstrucció de fibres del cervell. La principal limitació de la plataforma és que no suporta la visualització de models 3D. Aquest fet limita el seu us en la pràctica clínica habitual ja que es fa difícil la interpretació dels mapes de connectivitat que genera.

Poder combinar els mapes de fibres amb visualitzacions 3D facilitaria la interpretació de les fibres, ja que podríem establir una relació la fibra i la zona del cervell on correspon.

1.3 Objectius

L'objectiu principal d'aquest projecte és estudiar, dissenyar i implementar un algorisme de visualització de volums i integrar-lo en la plataforma DTIWeb de visualització i processament de dades de DTI.

Per poder assolir aquest objectiu hem definit els següents objectius específics:

1. *Estudiar la plataforma de visualització i processament de dades de DTI anomenada DTIWeb.*
    - La plataforma DTWeb és un dels projectes de processament de dades DTI, portat a terme pel grup d'Informàtica Gràfica de la Universitat de Girona.
    - És necessari estudiar la plataforma DTIWeb, ja que és la plataforma on serà inclòs el resultat d'aquest projecte.
    - Estudiant la plataforma, serà possible portar a terme la reutilització de codi ja escrit, i per tant seguir un millor disseny de programació.

2. *Estudiar, dissenyar i implementar diferents tècniques de visualització de volums i integrar-les en la plataforma DTIWeb.*
    - És necessari saber tractar part de les dades obtingudes d'un DTI, per poder extreure un conjunt d'imatges en 2D.
    - Seguidament, s'ha de reconstruir aquestes imatges en 2D per poder modelar un volum, i així obtenir una reconstrucció en 3D de la part explorada pels dispositius mèdics.
    - Un cop aconseguit això, tindrem un visualitzador de volums el qual haurem d'integrar dins la plataforma DTIWeb.

3. *Estudiar, dissenyar i implementar una tècnica de definició de funcions de transferència. Aquestes funcions són les que permeten pintar el volum permeten obtenir diferents efectes en la visualització.*
    - Per poder generar obtenir visualitzacions de volums es necessari un sistema que permeti modificar els colors que s'usaran per pintar el models. Aquesta assignació de colors es fa a partir de les funcions de transferència.

4. *Estudiar, dissenyar i implementar una tècnica d'il·luminació per poder obtenir efectes més realistes en el moment de fer les visualitzacions.*
    - Per poder crear un efecte de realisme en el nostre volum 3D, és necessari l'aplicació de il·luminació artificial en aquest.
    - Per poder aconseguir aquesta il·luminació artificial, s'hauran de crear tècniques d'il·luminació.

5. *Ampliar la tècnica de visualització implementada en el punt 2 per permetre que suporti la visualització simultània de volum i clúster de fibres.*
    - En el punt 2, es tractava de fer una reconstrucció 3D de tot el volum a patir de les imatges en 2D extretes del DTI.

- A més, amb els punts 3 i 4, hem aconseguit que el volum 3D reconstruït consti d'efectes visuals que permetin una millora d'aquest.
- Ara, és necessari modificar el punt 2 per tal de que també sigui capaç d'incloure en el volum clústers de fibres del cervell.

6. *Integrar tot els mètodes desenvolupats en la plataforma DTIWeb.*
    - Un cop aconseguit el nostre visualitzador 3D, amb correcte funcionament amb les tècniques d'il·luminació, i la inclusió de fibres del cervell en el mateix volum, es tracta d'integrar-lo dins la plataforma DTIWeb.
    - Un cop aconseguit això, la plataforma DTIWeb serà capaç de realitzar l'objectiu marcat en aquest projecte.

1.4 Planificació

El pla de treball que hem definit per poder assolir els nostres objectius l'hem dividit en dues etapes.

1.4.1 Estudi de l'entorn de treball.

En aquesta etapa s'estudiaran per una banda tots els aspectes teòrics relacionats amb el DTI i els algorismes de visualització i per altre s'estudiarà l'entorn de programació sobre el que s'haurà de fer la implementació.

1.4.2 Implementació

En aquesta etapa s'implementaran els mòduls necessaris per poder assolir els nostres objectius. Es farà un estudi dels requeriments i es farà el disseny i l'implementació.

1.5 Metodologia

Per portar a terme aquest projecte, s'ha utilitza la metodologia eXtremePrograming (XP).

El XP, és una metodologia de la enginyeria del software formulada per Kent Beck. Aquesta metodologia es diferencia de les tradicionals, perquè posa més atenció a la adaptabilitat que en la previsibilitat, també dit d'una altra forma, és que posa més atenció a la adaptabilitat de les persones, que no pas en la dels processos. Es tracta de ser capaç de adaptar-se als canvis de requisits en qualsevol moment de desenvolupament del projecte.

Es pot considerar que el XP com la adopció de les millors metodologia de desenvolupament d'acord al que es vol fer amb el projecte, i aplicar-lo de manera dinàmica durant el cicle de vida del software.

A la metodologia de eXtreme Programing, podríem emfatitzar cinc principis:

1. *Simplicitat.*
    - És la base del eXtreme Programing. Es tracta de simplificar el disseny per un millor desenvolupament i manteniment. A més, també es tracta de tenir una documentació simple i completa.
2. *Comunicació.*
    - Per una banda, tenim la comunicació del codi amb els programadors. Es tracta de tenir un codi autodocumentat, el qual mostri l'objectiu de les classes i mètodes sense entrar en detalls del seu desenvolupament.
    - Per altra banda, també és important la comunicació del client, fins al punt que aquest formi part del equip de treball, ja que així, és ell qui decideix les característiques de més prioritat.
3. *Retroalimentació (feedback).*
    - Es tracta de fer cicles de desenvolupament molt curts, i al finalitzar cadascun de ells es tornen a avaluar els objectius per veure si es compleixen tal i com es vol. A més, la utilització de cicles curts de desenvolupament provoca una cerca més ràpida i precisa de errors en el codi.
4. *Coratge.*
    - Es requereix per implementar las característiques el client vol ràpidament sense caure en la temptació d'optar per una perspectiva més flexible que permeti futures modificacions.
5. *Respecte.*
    - En el cas d'un projecte en equip, el membres del grup s'ha de respectar mútuament en no realitzar canvis del codi que puguin afectar a parts desenvolupades per altres membres del grup.

L'aplicació del XP en el meu projecte ha estat principalment en el punt 3 (retroalimentació) per anar comprovant que cada cicle de codi desenvolupat funcioni correctament per ell mateix, i s'adapti a les necessitats proposades o adaptar-lo per si han modificacions al respecte.

També s'ha atacat el punt 2, fent un codi autodocumentat a través dels JavaDoc de Java Sun, i el punt 1 intentant fer un disseny senzill i clar.

Els dos últims punts (4 i 5) del eXtreme Programing no s'han pogut assolir, ja que el desenvolupament del projecte ha estat individual.

1.6 Temporalització

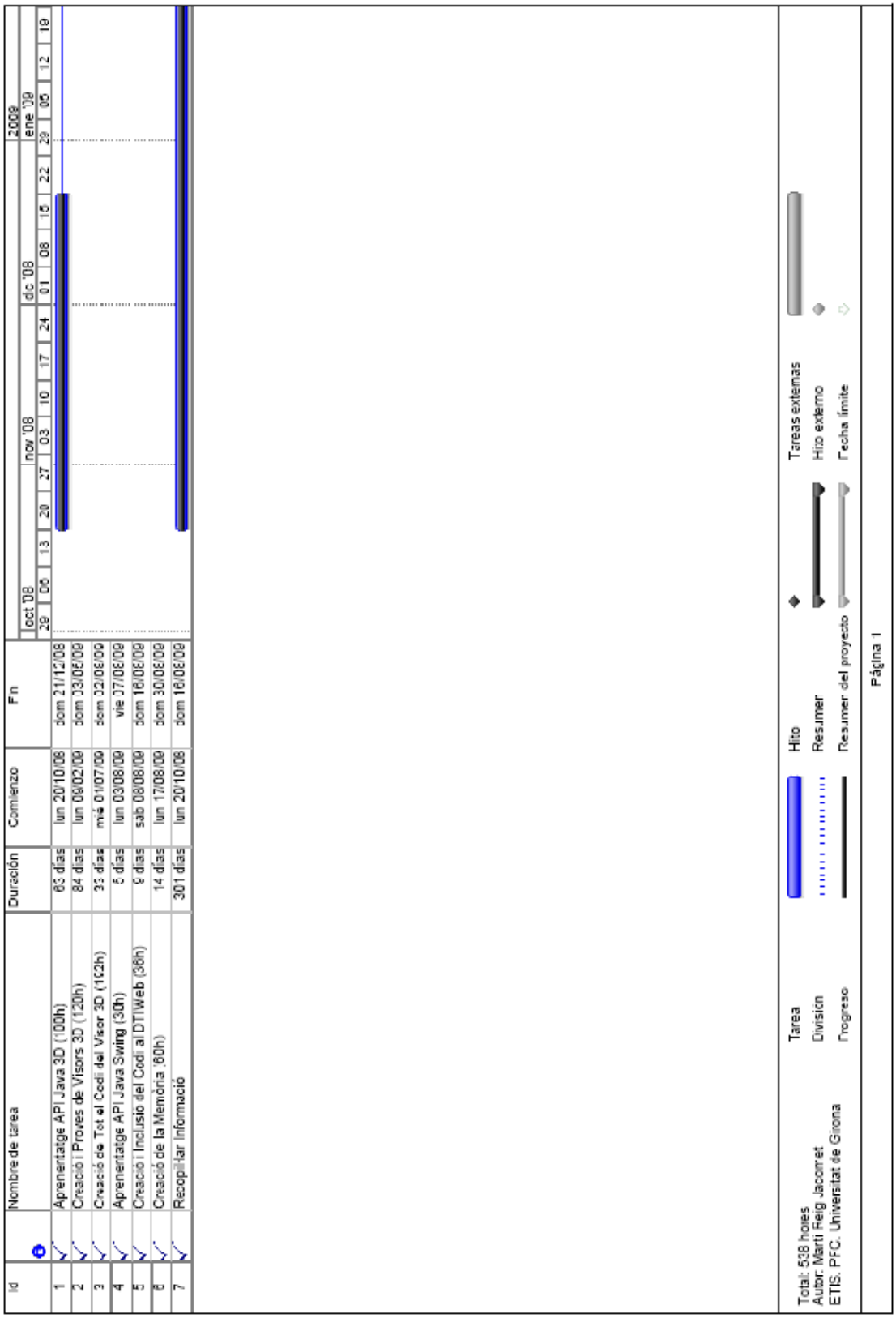

Figura 1.3. Primera part del diagrama de Gantt que mostra la temporalització portada a terme per desenvolupar aquest projecte.

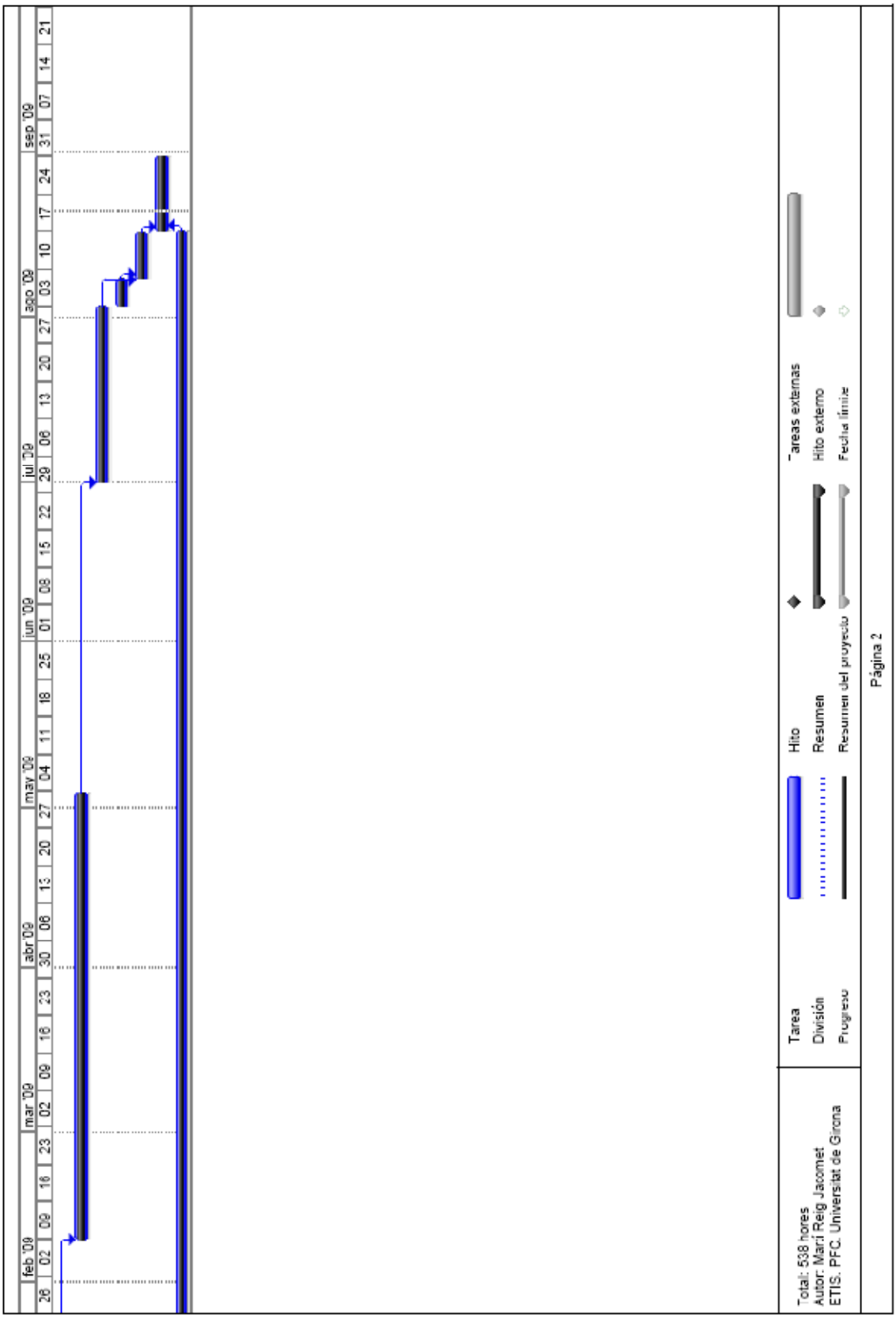

Figura 1.4. Segona part del diagrama de Gantt que mostra la temporalització portada a terme per desenvolupar aquest projecte.

En el diagrama de Gantt que es mostra en les figures 1.3 i 1.4, es presenta la temporalització que s'ha seguit durant el projecte.

Com es pot veure, vam començar fent l'estudi de la part teòrica i pràctica, aprenent l' API de Java 3D, la qual ens va comportar un temps de dos mesos. Durant aquest temps es van aprendre els conceptes principals relacionats amb el DTI i amb la Visualització de volums

Tot seguit, durant un transcurs de tres mesos vam anar construint els primers dissenys, aquests van servir com a primers prototipus de volum per representar el nostre model de vòxels, fins a trobar un disseny que complís les nostres necessitats.

Com a tercera fase, vam portar a terme el desenvolupament del disseny final sobre la capa d'aplicació, aquesta etapa va tenir la duració d'un mes.

Després, per desenvolupar la capa d'interfície es va aprendre el funcionament del Java Swing, el qual ens va portar una repercussió de cinc dies, i un cop estudiat, es va portar a terme el desenvolupament durant un transcurs de nou dies.

Finalment, un cop desenvolupat tot el codi del nostre projecte, es va procedir a la creació de la memòria d'aquest.

Cal destacar, que durant el desenvolupament de tot el projecte, s'ha anat recopilant i estudiant informació, fet que també queda reflectit en el diagrama de Gantt.

Durant el desenvolupament de l'aplicació s'ha anat fent varis testeigs de l'aplicació a mesura de que s'anava avançant, finalment a la última data de desenvolupament es va portar a terme un testeig definitiu d'aquesta. Ara bé, els testeigs definitius utilitzats per aquest document s'han portat a terme durant la mateixa creació d'aquest.

1.7 Estructura del document

Aquest document s'ha estructurat de la forma que s'explicarà a continuació.

En el Capítol 2, es parlarà sobre totes les tasques teòriques portades a terme durant l'estudi previ a la realització del projecte, com són, estudiar els model de DTI, tècniques de visualització de volums, funcions de transferència, tècniques de composició de colors, i l'algorisme de ray-casting.

Al Capítol 3, continuant amb l'estudi previ, però amb una basant pràctica, es parlarà sobre les eines de programació utilitzades, i sobre la plataforma DTIWeb.

Tot seguit, en el Capítol 4, es portarà a terme l'explicació sobre d'anàlisis de requisits de l'aplicació, conjuntament amb el disseny d'aquesta i la seva implementació.

Continuant, en el Capítol 5, hi consta la interfície, on es s'explicaran totes les interfícies gràfiques dissenyades i el seu funcionament.

Al Capítol 6 s'ha efectuat les proves de la nostra aplicació i s'han avaluat el resultats obtinguts.

En el Capítol 7, s'ha estudiat les millores i treball futur que es podrien portar a terme per a la millora i optimització de l'aplicació.

Després, al Capítol 8, tenim les conclusions arribades un cop s'ha finalitzat aquest projecte.

Capítol 2. Fonaments teòrics

En aquest capítol presentarem tot l'estudi previ que ha calgut realitzar per poder assolir els conceptes necessaris per dur a terme el projecte.

2.1 Els models de DTI

En els darrers anys, s'han produït grans avenços tecnològics en el món de la imatge mèdica i en particular en els dispositius de captació d'imatges permetent obtenir de forma no invasiva informació acurada de l'organisme d'un pacient.

La captació de dades del pacient es realitza mitjançant algun dispositiu físic que permet mesurar un o més paràmetres determinats en diferents punts del cos del pacient. Les dades obtingudes dependran de les prestacions de l'aparell: la seva resolució, fiabilitat, precisió, etc.

En aquest projecte ens centrarem en les dades que s'obtenen a partir d'aparells de ressonància magnètica. La RM clàssica cada punt de l'organisme del pacient mesura la densitat i la mobilitat dels protons detectant l'energia que emeten al desplaçar-se. Aquest dispositiu permet la detecció de teixits ii obtenir imatges com la que es mostra en la figura 2.1.

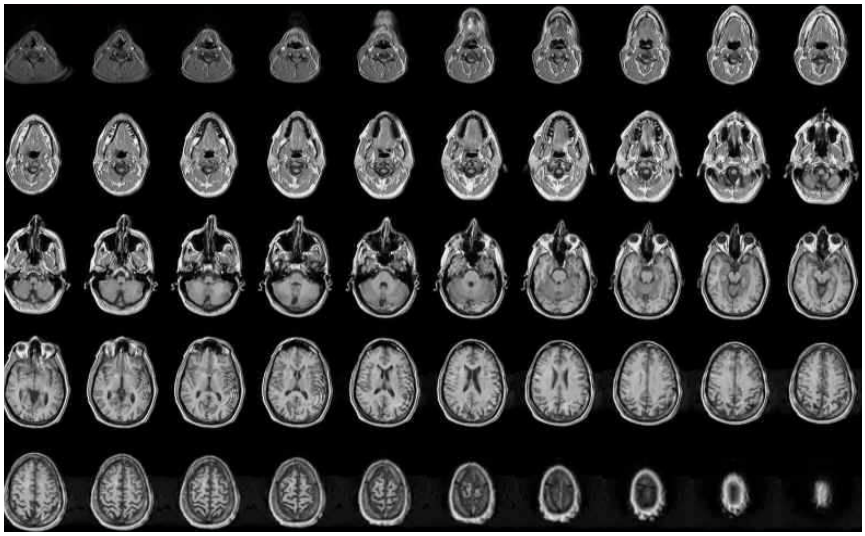

Figura 2.1. Imatges obtinguda a traves d'una MRI.

Nosaltres treballarem amb una nova modalitat d'imatge de RM l'anomenada DTI.

El DTI és una nova modalitat de RM que permet obtenir la informació necessària per determinar com circula la matèria blanca a les fibres del cervell. A través de l'aparell de RM es captura informació del pacient en diferents direccions. Cal remarcar que la RM clàssica només mesura en una direcció. La informació del DTI es representa en forma d'imatges que es codifiquen en un fitxer seguint el protocol fixat pel dispositiu de captació, normalment el DICOM. Aquests fitxers es caracteritzen per tenir un gran volum de dades ja que es pot interpretar com n vegades un model bàsic de RM on n representa el nombre de direccions que s'ha aplicat per fer l'adquisició.

Per poder treballar amb aquestes dades es necessari un sistema de representació de la informació que sigui fàcil de processar-lo des d'un punt de vista informàtic. El model que s'usa és el model de vòxels. Un vòxel és un píxel volumètric, és a dir, la versió en 3D d'un píxel. El procés de voxelització es tracta d'un procés de divisió de les imatges en un conjunt de cubs de les mateixes dimensions i repartits de manera regular formant una malla. Cada vòxel que construeix la imatge original, emmagatzema les dades obtingudes del pacient, així com la opacitat, la intensitat, etc.. d'aquella determinada regió.

Aquest seria un esquema simplificat del procés:

- *Primer Pas: Obtenció de les dades del pacient mitjançant la RM.*

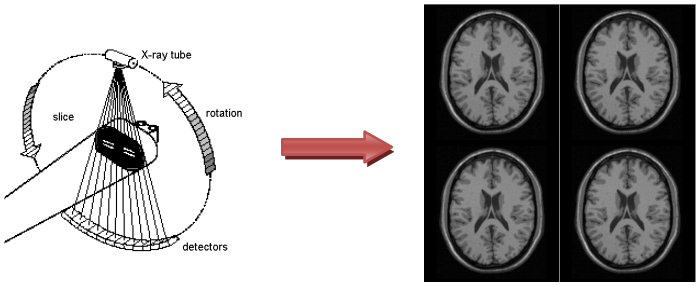

Figura 2.2. Imatges que mostren l'obtenció de dades mitjançant la ressonància magnètica.

- *Segon Pas: Els diferents plans o talls s'apilen un sobre l'altre, reconstruint de nou la part 3D del cos.*

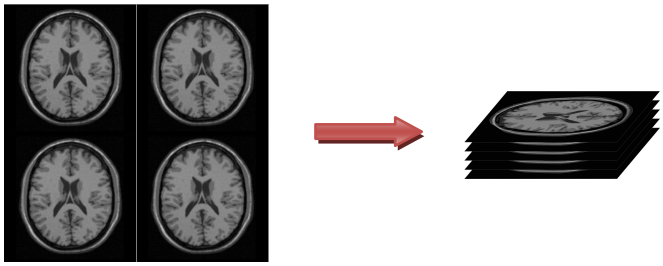

Figura 2.3. Imatges que mostren com es reconstrueix de nou el cos.

- *Tercer Pas: Finalment es definirà una malla regular 3D formada per un conjunt de cel·les sobre el vèrtexs de les quals es representarà la informació de cada tall.*

Figura 2.4. Imatges que mostren com es defineix la malla regular 3D.

Hem de tenir en compte que el nostre model serà una mica més complicat, ja que nosaltres per cada vòxel tenim informació en diferents direccions. La informació del DTI es representa mitjançant un tensor de difusió (matriu 3x3) que representa la difusió de les molècules d'aigua en un vòxel. Per obtenir aquest tensor cal aplicar un procés bastant costós que queda fora de l'abast del nostre projecte. Volem remarcar que el model de vòxels usat per mantenir el DTI es bastant costos en el moment de gestionar-lo.

2.2 Processament i visualització de les dades de DTI

Una vegada creat el model de vòxels, en el que per cada vòxel tenim un tensor que representa la informació captada pel DTI ens cal una tècnica per poder representar aquesta informació en la pantalla. Donat que mostrar tensors no es fàcil, el que s'ha proposat són tècniques que treballen amb una simplificació del tensor. El que es fa es diagnosticar el tensor i s'obtenen valors i vectors propis a partir dels quals es podem obtenir diferents paràmetres que són interpretables pels experts. Tenint en compte aquesta simplificació del tensor les diferents tècniques de visualització de dades de DTI s'han classificat en tres grans grups: mètodes 1D, 2D i 3D. En el context del nostre projecte haurem de tenir en compte les tècniques 1D i les 3D.

Mètodes 1D. Basats en calcular un valor escalar reduint la informació del tensor a un valor. Com s'ha explicat en l'apartat anterior per a cada vòxel disposem d'un tensor de difusió que pot ser representat com una matriu de 3x3. Gràcies a això, podem determinar les direccions principals de difusió de cada vòxel, a través de la diagonalització d'aquesta matriu. El que ens proporciona la diagonalització són els valors i vectors propis que ens donen com a resultat les tres direccions principals de difusió. Aquestes tres direccions es poden representar sobre una el·lipsoide. Finalment, a partir d'una fórmula que escollim, els reduïm a un únic valor que ens servirà per a representar mapes de color. A la figura 2.5 es veu un exemple d'aquest procés.

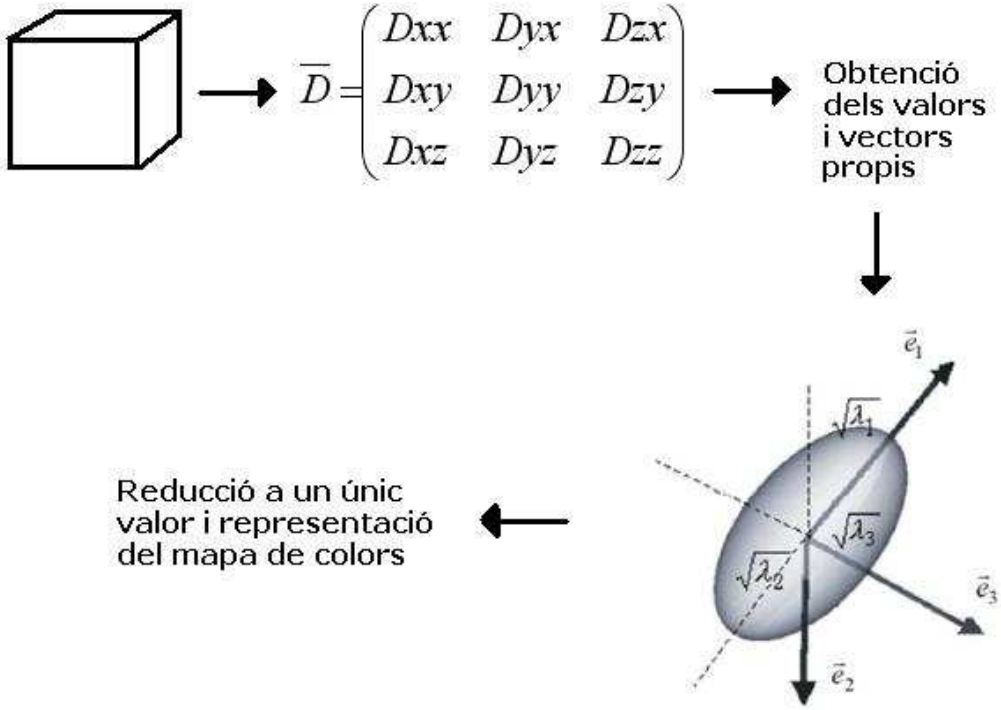

Figura 2.5. Funcionament de la tècnica de visualització 1D.

La visualització d'aquests valors es fa en escala de grisos tot i que, per a obtenir més informació, es poden reemplaçar els nivells de grisos per valors RGB fent ús de diferents tècniques. Els valors que es poden utilitzar per a obtenir aquests tipus de visualització són diversos i es calculen mitjançant equacions que s'apliquen sobre els valors de la matriu. La figura 2.6 ens mostra un exemple d'aquesta visualització sobre un model de cervell humà utilitzant diferents paràmetres.

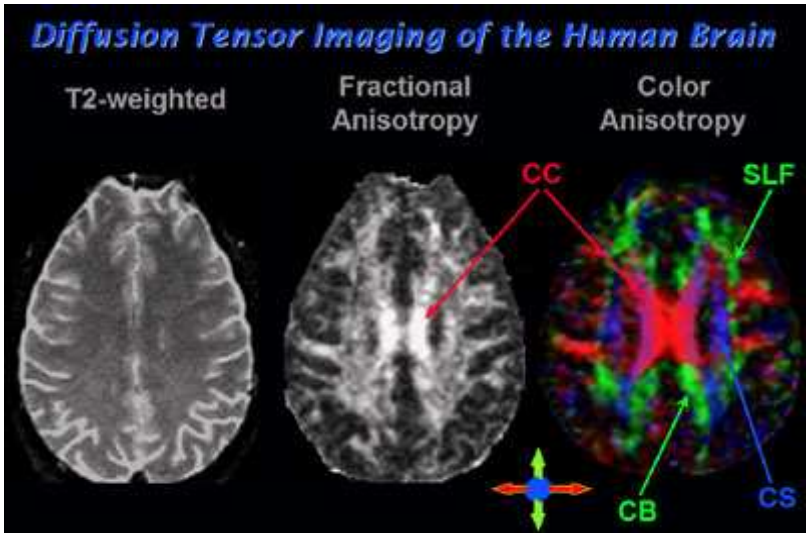

Figura 2.6 .Exemple de visualització 1D

Mètodes 3D. Aquests mètodes són també anomenats mètodes de tracking de fibres. Els mètodes 3D són els que simulen el traçat de fibres del cervell amb l'objectiu de donar una visió 3D de la fibra.

A partir d'un punt inicial reconstrueixen el camí que segueix una fibra al llarg del cervell i un cop s'aconsegueix aquest camí s'aplica un algorisme de visualització.

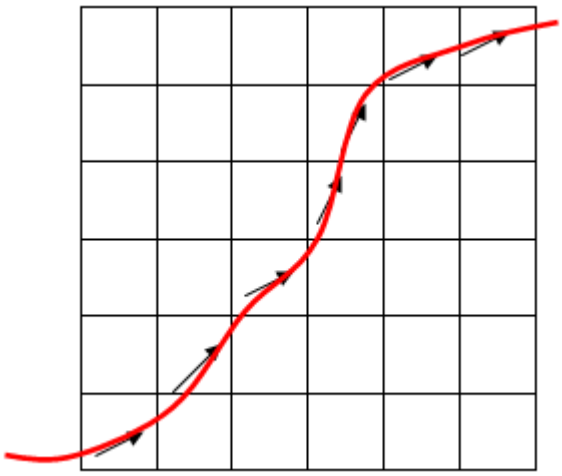

Figura 2.7. Mapa de vectors

Normalment les imatges que s'obtenen a partir dels procés de tracking es sobreposen a un mapa generat aplicant un mètode 1D. Les imatges que s'obtenen són com la que es mostra en la Figura 2.8. Com es pot veure es fa difícil determinar per on passen les fibres ja que es perd la noció del 3D. Una solució a aquest problema es l'aplicació dels algorismes de visualització de volums.

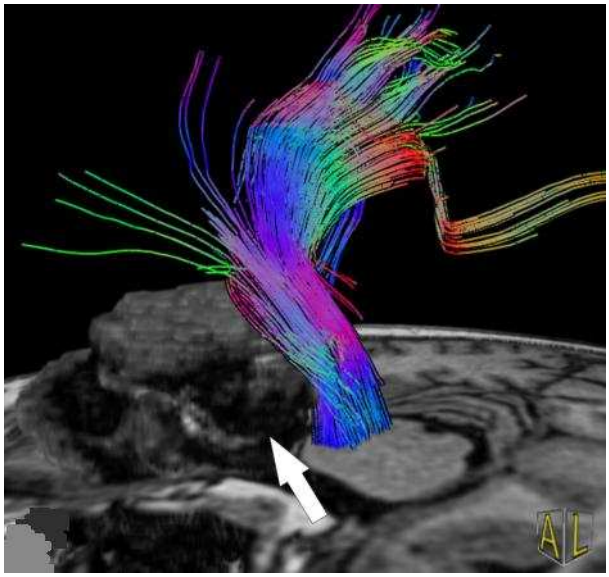

Figura 2.8. Imatge on podem apreciar, com es perd la noció del 3D.

2.3 Visualització de Volums

Per poder visualitzar tota la informació que està representada en un model de vòxels ens cal aplicar un algorisme de visualització de volums. Els diferents algorismes que s'han proposat al llarg de l'historia estan pensat per visualitzar models de vòxels en els que en cada vòxel només hi mantenim una propietat. En el nostre cas en cada vòxel i tenim un tensor a partir del qual podem obtenir diferents valors. Nosaltres considerem que el que visualitzarem serà un volum que representi una d'aquestes propietats. Això vol dir que l'usuari haurà de poder decidir quin propietat es la que vol usar per definir el seu model. Una vegada aclarit aquest punt hem d'analitzar els diferents mètodes que es poden aplicar. Podem considerar dos tipus d'algorismes:

- *Algorismes de Visualització Directe del Volum.*
  Les tècniques de visualització directa de volum treballen directament amb totes les dades representades en el model de vòxels. Per generar la imatge final s'assignen els atributs gràfics (color, transparència, textures,...) a tots els valors de propietat representats en el model de vòxels.

- *Algorismes de Visualització de Models Reconstruïts.*
  Aquests algorismes visualitzaven un model simplificat reconstruït a partir de la informació del model de vòxels. En el nostre cas, aquests mètodes no ens interessen perquè impliquen un pre-proces sobre el model de DTI per reconstruir el que volem visualitzar.

2.3.1 Algorismes de Visualització Directa de Volums

Els algorismes de visualització directa de volums es caracteritzen perquè tenen en compte tota la informació captada per un model de vòxels. Poden considerar-se dos grups d'algorismes de visualització directa de volums:

- *Algoritmes d'ordre imatge o tipus RayCasting.*
  Per obtenir la imatge es llença un raig per cada píxel de la pantalla al pla paral·lel del nostre model de vòxel i es tenen en compte totes les interseccions amb els diferents vòxels del nostre model. Segons les propietats dels vòxels travessats (material, etc) es calcula el color que tindrà el píxel des de on surt el raig inicial Per cada mostra tenim assignat un atribut gràfic que es composarà amb totes les del raig per obtenir el color final.

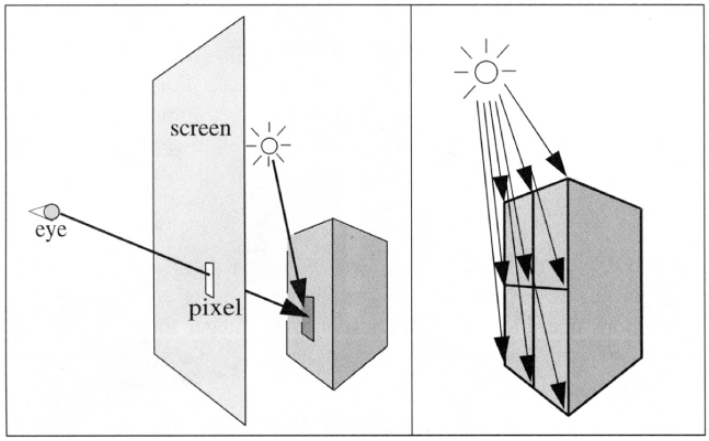

Figura 2.9. Es té en conta l'ull de l'observador i el focus de llum.

- *Algoritmes d'ordre objecte o tipus splatting:*
  En aquest cas enlloc d'anar de la pantalla al model considerem que és el model el va a la pantalla. Es fa un recorregut ordenat de tot el model i funciona com si anéssim llençant boles de neu sobre una paret (splatting).

En aquest projecte ens centrarem en els algoritmes de RayCasting. i en concret en un optimització d'aquest que s'anomena Visualització de Volums basada en textures.

Independentment de l'algorisme de visualització directa de volums que s'apliqui, tots ells tenen una part en comú: la definició de la funció de transferència i la composició de colors.

2.4 Funció de transferència

Dins de cada vòxel trobem les propietats que l'aparell de captació ha enregistrat. Aquest valors de propietats estan dins un rang de valors que defineixen el tipus de material o massa que representen. Segons quina propietat o material hi hagi en el vòxel, hem de trobar un color adequat per representar correctament la realitat o si més no, trobar la manera de representar la major informació possible. Per determinar quin color correspon a cada propietat o interval de propietats fem servir la Funció de transferència.

La funció de transferència determina el color i grau transparència (opacitat) de cada valor de propietat o interval de valors. Tradueix una propietat en un color. L'elecció del color la fa l'usuari que pot escollir veure per exemple la pell d'un to rosat semitransparent per tal de veure la part interior, o també veure els músculs de color vermellós i opacs.

Com que la funció de transferència determina el color i l'opacitat del píxel, farem servir una representació amb un valor RGBA on RGB són la component vermella, groga i blava (Red, Green, Blue) i A(alpha) el grau d'opacitat que va de 0 a 1, on 0 és transparent i 1 opac.

El problema que tenim és la dificultat de triar una funció de transferència adequada al nostre model ja que no coneixem l'estructura interna del nostre model de vòxels A més cada dispositiu de captació té uns valors de propietat diferent per a cada tipus de teixit. Així doncs no podem establir un estàndard de funció que s'adeqüi per a tots els models.

Per trobar la funció de transferència adequada podem fer-ho de dues maneres:

- Buscar automàticament la funció de transferència del model: És força complicat intentar que l'ordinador busqui la funció adequada al nostra informació. S'hauria de fer un recorregut exhaustiu mirant les estructures internes del model i enregistrar els seus rangs de propietat. El que seria molt més complicat és la selecció dels colors RGBA per a cada rang de valors. Aquesta part quedaria igualment a càrrec de l'usuari.

- L'altre possibilitat que és la que farem servir és la d'assaig i error. L'usuari s'encarrega de provar quins intervals són més adients per al nostre model. A més alhora anirà escollint els colors dels rangs per tal de poder visualitzar els canvis fets. El problema d'aquest mètode és que no sempre és ràpid trobar la funció adequada.

Un cop tenim la funció de transferència definida necessitem escollir la tècnica de composició de colors que farem servir.

2.5 Composició de colors

Ja definida la funció de transferència adaptada al nostre model de dades amb els colors RGBA assignats, apliquem ray-casting. Aquesta tècnica que veurem més endavant amb més detall consisteix en llençar raigs de llum des de cada píxel de la pantalla cap al model de dades. Aquest raig anirà travessant l'estructura interceptant amb diferents vòxels Aquests vòxels contenen ara, després de definir la funció de transferència un color RGBA. Hem de tenir en compte que el raig es va atenuant conforme va penetrant per superfícies més opaques. En el píxel volem representar la combinació de color que s'obté al considerar totes les mostres o vòxels que ha entravessat el raig.

Per obtenir aquesta combinació s'usa una equació simplificada de la teoria del transport de la llum. Bàsicament considerem el raig no com una línia continua sinó com una línia discreta.

Els colors i opacitats dels píxels del darrera són atenuats per les opacitats dels píxels del davant:

$$rgb = rgb_{darrera} \cdot \alpha(1-\alpha_{davant}) + rgb_{davant}t \cdot \alpha_{davant}t$$

$$\alpha = \alpha_{darrera}(1-\alpha_{davant}) + \alpha_{davant}$$

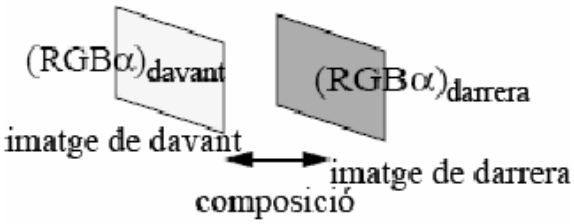

Figura 2.10. Imatge on es mostra el mètode de composició.

- Fent servir:

$$rgb_{darrera} = rgb_{darrera} \cdot \alpha_{darrera}$$

$$rgb_{davant} = rgb_{davant}t \cdot \alpha_{davant}$$

obtenim 2 equacions recursives que es poden fer servir per compondre qualsevol nombre d'objectes de davant cap endarrere:

$$rgb = rgb_{darrera} \cdot \alpha(1-\alpha_{davant}) + rgb_{davant}$$

$$\alpha = \alpha_{darrera}(1-\alpha_{davant}) + \alpha_{davant}$$

La visualització directa de volums utilitza aquesta expressió recursiva per combinar (compondre) les mostres preses al llarg del raig.

Aplicant aquesta equació de forma recursiva obtenim la imatge final.

2.6 Ray Casting

En el món real, els rajos de llum viatgen per l'aire des d'un punt de llum i interaccionen amb els objectes provocant col·lisions. En aquestes col·lisions els rajos es comporten de diferents maneres segon la tipologia de l'objecte amb que interaccionen. L'objecte pot agafar tot el color o reflectir-ne els rajos dels colors que no capta.

En visualització 3D s'intenta representar el món real i per tant s'ha de simular el comportament de les llums en una escena amb un objecte. Així va sorgir l'algoritme de rai casting.

Aquesta tècnica consisteix en llençar un raig per a cada un dels píxels de la imatge que es mostrarà per pantalla. Aquests rajos són perpendiculars al pla de visió, que representaria a l'ull de l'observador. Tots els raigs són paral·lels entre sí i distribuïts en intervals regulars.

En un principi la tècnica estava dissenyada per visualitzar superfícies. Amb un model superficial quan el raig xoca amb una superfície enregistra el color que ha de tenir segons les propietats del material, les propietats de les llums, i els altres objectes que hi ha a l'escena.

En RayCasting Volumètric l'idea bàsica és la mateixa tot i que s'ha de tenir en compte que al tractar-se d'un volum, l'objecte té contingut a l'interior i si les capes més externes deixen passar la llum, és a dir, no són opaques, el raig ha de continuar travessant el volum fins que l'opacitat acumulada és màxima(Figura 2.11). La composició dels colors es fa aplicant l'equació de la figura 16.

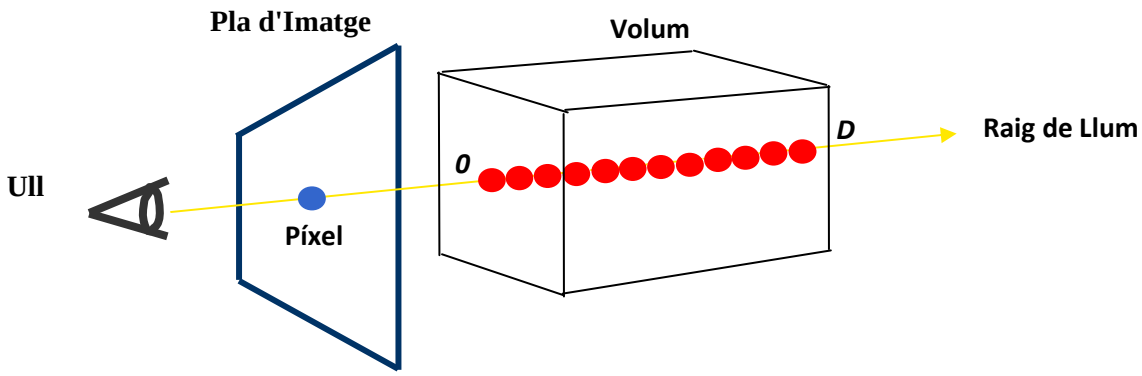

Figura 2.11. El raig entra pel punt O i a cada col·lisió dins el volum calcula el color i la transparència. Un cop calculades totes les col·lisions es computarà el color final D que serà el que es mostra al pla d'imatge(pantalla).

$$C = \sum_{i=0}^{N} \alpha_i c_i \prod_{j=0}^{i-1} (1 - \alpha_j)$$

Figura 16. Equació de la composició de colors.

Capítol 3 .Fonaments Pràctics

En aquest capítol presentarem les eines que hem utilitzat en el desenvolupament del projecte, així com una breu explicació de cadascuna d'elles, tot mostrant les principals característiques, i els motius que ens han portat a la seva utilització. També presentarem la plataforma DTIWeb en la que s'ha d'integrar tot el que desenvolupem.

3.1Eines de Programació

Java

Java és un llenguatge de programació orientat a objecte, desenvolupat per Sun Microsystems a principis dels anys noranta.

La sintaxis del llenguatge va estar molt influïda per C i C++, però posseeix un model més simple, com pot ser la eliminació de punters i accessos a memòria.

Figura 3.1. Logotip de l'empresa Sun Microsystems.

Com hem dit, una de les seves principals característiques és que és un llenguatge orientat a objecte, tot i això, aquest no ho és el 100%, ja que les classes primitives no ho són.

Independència de la plataforma, la segona característica principal del llenguatge Java ha estat el axioma de Java "write once, run everywhere" ("escriu una vegada, executa a qualsevol lloc"), és a dir, és un llenguatge capaç d'executar-se sobre qualsevol hardware. Tot i això, actualment s'han trobats problemes, els quals no han pogut portar aquesta portabilitat de plataforma al 100%.

La portabilitat del llenguatge, ha estat gràcies a la JVM (Java Virtual Machine, o Màquina Virtual de Java). La JVM és un programa específic de cada plataforma, capaç de interpretar llenguatge Java i adaptar-lo a la plataforma que s'està executant.

Les aplicacions de Java estan compilades en un bytecode (java byte code) el qual és interpretat per la JVM, i aquesta fa de "pont" entre el codi compilat i el hardware.

S'ha escollit el llenguatge Java per portar a terme el projecte, ja que és el llenguatge en què està creat la plataforma DTIweb.

Concretament, per desenvolupar aqueta part, s'ha utilitzat el Java SE Devolepment Kit (JDK) versió 6 Update 16, la qual és la versió més actual fins a la data.

Figura 3.2. Logotip de Java.

Java 3D

La API Java 3D, és una extensió estàndard del JDK2 de Java. És una API de gràfics 3D, que permet poder crear entorns tridimensionals en Llenguatge Java.

Figura 3.3. Escenificació del logotip de la API de Java 3D.

Inicialment, aquesta API només corria sobre OpenGL, però actualment pot ser executada sobre OpenGL O DirectX.

La API de Java 3D crea un arbre de classes de Java que serveixen com interfície per sistemes de renderitzat de gràfics 3D i un sistema de so.

De cara al desenvolupador, aquest treballar amb constructors de alt nivell per crear i manipular geometries tridimensionals. Aquestes geometries residiran en un univers virtual que posteriorment serà renderitzat. La API de Java 3D està preparada per poder treballar a diferents escales de dimensions, des de tamanys astronòmics fins a nivells subatòmics.

A més, Java 3D està pensat per poder ser utilitzat conjuntament amb els Threads de Java, així es pot aconseguir un procés de renderitzat en paral·lel per obtenir un rendiment molt més òptim.

S'ha escollit la API de Java3D, ja que necessitàvem la creació de universos en 3D, i com que la plataforma DTIweb està feta sobre el Llenguatge Java, la millor forma era continuar utilitzant el mateix llenguatge, i fer ús de les llibreries que aquest ens proporciona.

JFC Swing

Java Foundation Classes (JFC) i Java Swing són unes biblioteques gràfiques per Java, les quals tenen un conjunt de característiques per ajudar a construir Interfícies Gràfiques d'Usuari (GUI's). A grans termes, podríem dir que serveix per a la creació de botons, caixes de text, desplegables, taules, widgets,…

La biblioteca neix de a causa de dos antigues biblioteques de Java, la IFC i la AWT.

La IFC (Internet Foundation Classes) era una biblioteca per Java, desenvolupada inicialmente r Netscape, publicada al 1996. Aquest botons eren controlats íntegrament pel Java, i eren totalment independents de la plataforma, cosa que va permetre una completa portabilitat.

Figura 3.4. Un dels logotips de Java.

AWT (Abstract Window Toolkit), era una biblioteca concebuda com una API, la qual principalment permetia poder utilitzar els components natius de

cada sistema operatiu. Per tant, una aplicació Java amb AWT, funcionant en un sistema operatiu Windows utilitzaria els botons estàndards d'aquest, i en canvi funcionant sobre un sistema operatiu Linux distribució Ubuntu, utilitzaria els botons de Gnome. A la pràctica, no va funcionar perfectament.

Al 1997, Sun Microsystems i Netscape Communications Corporations, va anunciar la seva intenció de combinar IFC i altres tecnologia d les Java Fondation Classes, on va sorgir la biblioteca Swing. Aquest, permetria que l'aspecte de cada component d'una aplicació pogués canviar sense introduir canvis substancials en el codi de l'aplicació, a més permetia emular la aparença dels components natius de la plataforma en que s'estigués desenvolupant.

S'ha utilitzat aquesta llibreria, ja que permet un fàcil maneig al moment de voler crear una interfície gràfica per poder incloure el nostre codi a una plataforma gràfica. A més, inicialment la plataforma DTIweb també està desenvolupada sobre aquesta llibreria.

Eclipse

Eclipse és un Entorn de Desenvolupament Integrat (IDE) de codi obert i multiplataforma.

Eclipse va ser desenvolupat originàriament per IBM, com a successor de la seva família d'eines per VisualAge. Actualment, es troba desenvolupat per la *Foundación Eclipse*, una organització independent, sense ànim de lucre, que fomenta una comunitat de codi obert i un conjunt de productes complementaris, capacitats, i serveis.

De cara, al nostre projecte, la major part del codi ha estat desenvolupat a través d'Eclipse, a causa de la familiarització amb aquesta IDE, en excepció de la part final de la creació de la interfície gràfica.

En un inici, es va utilitzar al versió Ganymede (versió 3.4 d' Eclipse), i a partir del 24 de Juny de 2009 es va utilitzar la versió Galileo (versió 3.5 d'Eclipse).

Figura 3.5. Logotip d'Eclipse.

NetBeans

NetBeans es una plataforma pel desenvolupament d'aplicacions d'escriptori utilitzant Java i a un Entorn de Desenvolupament Integrat (IDE) desenvolupat utilitzant la Plataforma NetBeans.

NetBeans és un projecte de codi obert de gran èxit amb una gran base d'usuaris, una comunitat constant en creixement, amb molts socis arreu del mon. Sun Microsystem va fundar el projecte de codi obert NetBeans al junt del 2000, y continua essent el patrocinador principal del projectes.

S'ha escollit el NetBeans, perquè pel desenvolupament d'aplicacions d'escriptori (utilitzant Java Swing) és molt més manejable que l'Eclipse, per aquest motiu, podem dir que la major part del codi ha estat escrit en Eclipse, i després, quan aquest s'ha hagut d'incloure a la plataforma DTIweb de forma gràfica, s'ha utilitzat el NetBeans.

Figura 3.6. Logotip de NetBeans.

S'ha utilitzat la versió NetBeans 6.7.1. la qual és la més actual fins a la data.

3.2 La Plataforma DTIWeb

Introducció

El Projecte Final de Carrera s'ha desenvolupat per a crear una extensió més a la plataforma DTIWeb, i dotar aquest de més eines.

Aquesta plataforma sestà desenvolupant conjuntament amb el Laboratori de Gràfics i Imatge de la Universitat de Girona, i l'Hospital Dr. Josep Trueta de Girona.

El DTIweb, figura 3.7, és una plataforma creada per a donar suport a l'especialista en el moment de donar un diagnòstic a un pacient. Principalment el DTIWeb integra diferents tècniques de traçat de fibres, tècniques d'interrogació que doten d'informació de gran importància al moment d'efectuar un diagnòstic, i també està dotat de varies tècniques de visualització.

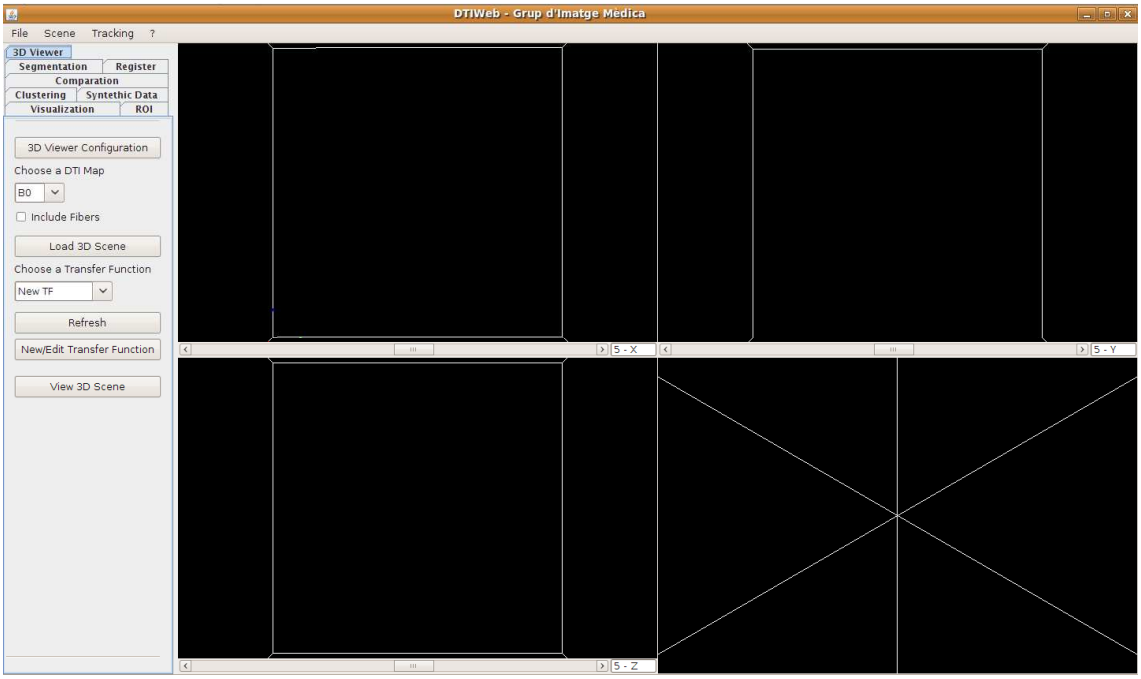

Figura 3.7. Imatge que mostra la plataforma DTIWeb amb la pestanya 3D Viewer activada.

Estructura del DTIWeb

El DTIWeb ha seguit des del seu inici un disseny per capes. Aquestes capes són, la capa de dades, la capa d'aplicació, i la capa de presentació. Aquestes capes són mostrades en el diagrama que mostra la figura 3.7

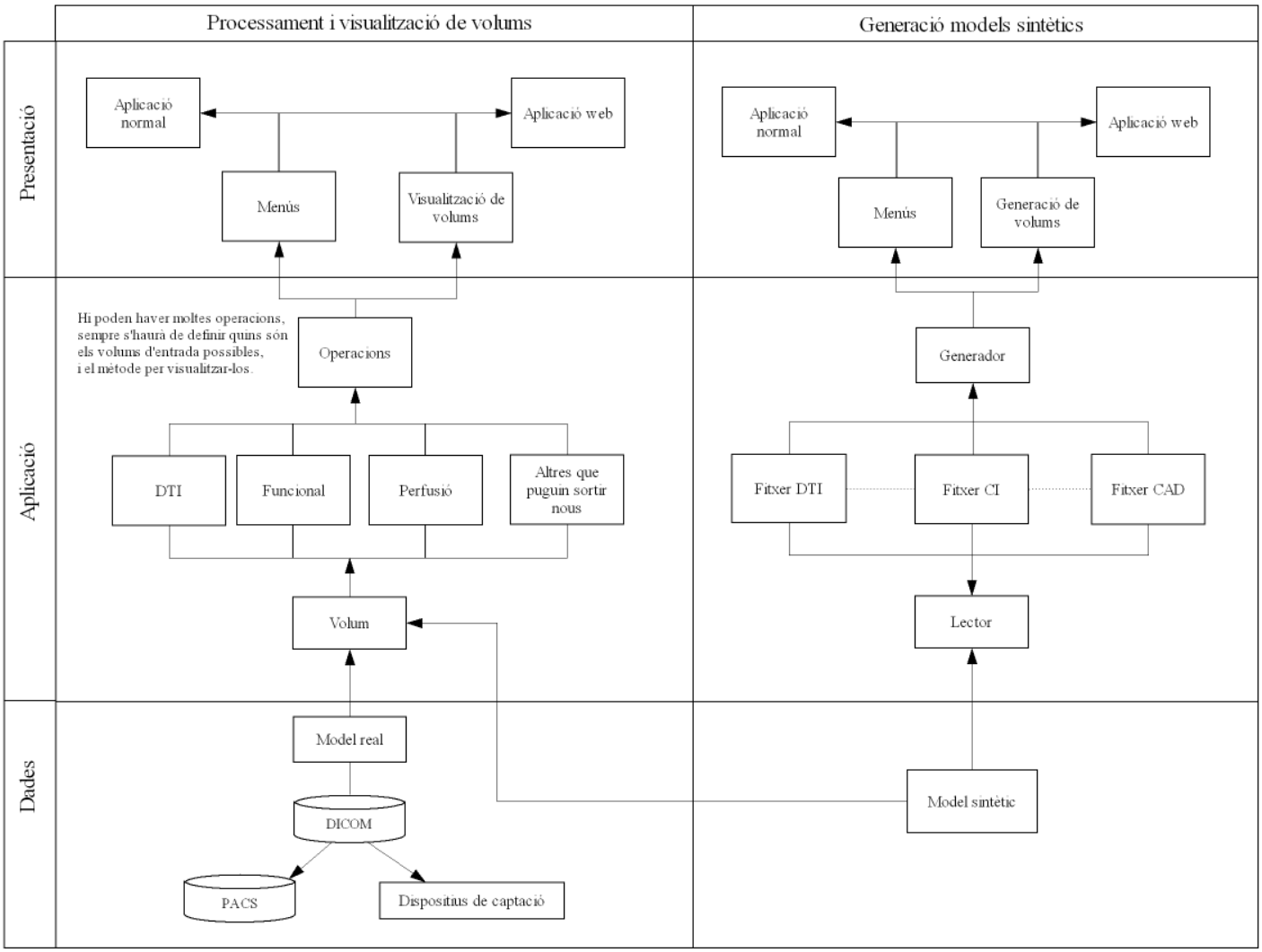

Figura 3.7. Diagrama que mostra els mòduls del DTIWeb.

Capa de Dades del DTIWeb

Com hem dit, el DTIWeb té un capa de dades, aquesta és la capa més inferior de la plataforma.

Aquesta capa conté tots el mòduls creats per a la lectura i tractament de tots els fitxers d'entrada.

L'origen dels fitxers que se li donen com entrada en el DTIWeb són directament la informació proporcionada per els aparells mèdics de captació de dades d'un pacient, sense haver rebut cap tractament entremig.

A més, la capa de dades també disposa dels mòduls necessaris per a guardar informació, com podrien ser models creats o modificat durant l'execució de la plataforma.

Capa d'Aplicació del DTIWeb

El DTIWeb, conté també una capa d'aplicació, la qual és la capa intermitja de la plataforma.

En aquesta capa si desenvolupen totes les operacions de tractament, manipulació, i creació de dades. Aquí hi podem trobar la creació del Model de Vòxels, models sintètics, mètodes de clústering de fibres, etc. És en aquesta capa on nosaltres hem creat tots els mètodes necessaris per a la creació del nostre Visor 3D, des de tractar el Model de Vòxels de la forma que ens pot interessar, fins a preparar tota la informació per a ser mostrada finalment per pantalla.

Capa de Presentació del DTIWeb

Finalment, tenim la capa més superior del DTIWeb, la capa de presentació.

En aquesta capa, hi ha tot el referent a la visualització de la plataforma, per una banda tenim tot el referent a la interfície gràfica, com podrien ser menús, i pestanyes, i per l'altra banda tenim tota la part que mostra els resultats visuals del calculat, com podrien ser textures, i clústers de fibres del cervell.

En aquesta part, és son nosaltres hem fet la pestanya corresponent al tractament del Visor 3D, i també é son hem preparat la nostra escena tridimensional per a poder mostrar les dades calculades a la capa d'aplicació.

Capítol 4. Anàlisis i Disseny de l'Aplicació

4.1 Introducció

Com ja hem explicat en anterioritat, el principal objectiu del projecte era integrar, a la plataforma de visualització d'imatges mèdiques DTIWeb, una aplicació capaç de reconstruir i visualitzar volums en tres dimensions, i incloure-hi els clústers de fibres.

En el moment de fer l'anàlisi de requisits, la plataforma ja estava dotada d'unes funcions per a poder obrir models de vòxels, i també permetia fer els anàlisis de fibres. Per tant, hem suposat que ja tenim un model de vòxels carregat a memòria, en una variable del programa inicial. També hem suposat tenir una variable que conté els clústers de fibres inicialment.

Tenint en compte aquests fets, haurem d'implementar un mòdul que s'encarregui de d'implementar totes les característiques referents al visor 3D, Ens aquest apartat explicarem els models de disseny i implementació del sistema que resolen el domini de la solució.

Per poder crear els models de disseny, utilitzarem UML, el qual és un llenguatge per visualitzar, especificar, construir, i documentar els elements d'un sistema de programari des d'una perspectiva orientada a objectes.

En primera instancia, d'anàlisis de requisits, s'estudiaran les noves funcionalitats dins la plataforma DTIWeb, amb inclusió de les explicacions detalladament. Això ho farem mitjançant un diagrama unificat de cas d'ús, i una fitxa de cas d'ús per cadascun.

En segona instància, i a través d'un diagrama de classes, podrem complementar la informació, mostrant com aniran col·locades i dissenyades les noves classes desenvolupades per a la nostra aplicació. Aquestes classes aniran acompanyades d'una breu explicació de cadascuna, i una explicació de cadascuns dels seus mètodes.

En tercera instància, a l'apartat d'implementació s'estudiarà els fluxes d'execució dels casos d'ús més importants, acompanyant-los del diagrama de seqüència corresponent i una breu explicació.

4.2 Anàlisis de Requisits

Casos d'Ús

A través dels casos d'ús descriurem les funcionalitats dels sistema quan aquest interactua amb un usuari extern, i això es definirà des del punt de vista de l'usuari extern.

Nosaltres només explicarem els casos d'ús referents a la nostra aplicació desenvolupada, i no tots els casos d'ús de la plataforma DTIWeb.

Inicialment, mostrarem un diagrama de cas d'ús, figura 4.1, en el qual es mostren tots els casos d'ús, i posteriorment estudiarem cada cas d'ús per separant mitjançant la seva fitxa.

Diagrama de Cas d'Ús

Figura 4.1. Diagrama de Cas d'Ús on es mostren els casos d'ús desenvolupats en aquest projecte.

Fitxes de Cas d'Ús

CAS D'ÚS	Carregar Escena
Descripció	Es vol carregar la geometria que representarà el model de vòxels.
Actors	Usuari
Precondició	• El model de vòxels ha estat carregat prèviament. • S'ha configurat el visor prèviament.
Flux Principal	1. Seleccionar (o no) la inclusió de fíbres (usuari) 2. Escollir un Mapa de DTI 3. Seleccionar Carregar Escena (usuari) 4. Crear volum (sistema) 5. Assignar Funció de Transferència transparent (sistema) 6. Afegir volum a l'escena (sistema).
PostCondició	La geometria que representa el model de vòxels està carregada a memòria, no està pintada, i no es mostra per pantalla.
Comentaris	• El model està pintat transparent, i no es mostra per pantalla. • És necessari ser cridat com a mínim una vegada perquè funcioni la resta de l'aplicació.

CAS D'ÚS	Seleccionar Funció de Transferència
Descripció	Es vol seleccionar una funció de transferència que es troba a una base de dades
Actors	Usuari
Precondició	• S'ha carregat un model prèviament. • Existeixen Funcions de Transferència a la Base de Dades.
Flux Principal	1. S'obre el directori que conté les funcions de transferència (sistema) 2. Es llegeixen totes les funcions de transferència (sistema) 3. Es mostren les funcions de transferència(sistema) 4. Seleccionar Funció de Transferència (usuari) 5. Si Funció de transferència no seleccionada (sistema) 5.1 Avisar usuari.(sistema)
PostCondició	La Funció de Transferència escollida queda guardada en el sistema.
Comentaris	• El model continua sense visualitzar-se.

CAS D'ÚS	Crear Funció de Transferència
Descripció	Es vol crear una nova funció de transferència.
Actors	Usuari
Precondició	• S'ha carregat un model prèviament.
Flux Principal	1. Seleccionar nova FT (usuari) 2. Introduir nom de la funció de transferència (usuari) 3. Introduir descripció de la funció de transferència (usuari) 4. Introduir els rangs de colors (usuari) 5. Si algun paràmetre incorrecte (sistema) 5.1 Avisar Usuari (sistema) 6. Seleccionar guardar tots els canvis (usuari) 7. Crear una nova funció de transferència en un fitxer, i guardar-lo al directori de les funcions de transferència amb tots els paràmetres introduïts per l'usuari (sistema)
PostCondició	S'ha creat una nova funció de transferència a la Base de Dades.
Comentaris	El model continua sense visualitzar-se.

CAS D'ÚS	Editar Funció de Transferència
Descripció	Es vol editar una funció de transferència.
Actors	Usuari
Precondició	• S'ha carregat un model prèviament. • La funció de transferència que es vol editar, existeix en la Base de Dades.
Flux Principal	1. Seleccionar la FT a editar (usuari) 2. Es mostra la funció de transferència (sistema) 3. S'introdueixen els nous valors. (usuari) 4. Si paràmetres incorrectes (sistema) 4.1 Avisar usuari (sistema 5. Es guarden les noves modificacions al fitxers que guarda la funció de transferència.
PostCondició	S'ha modificat una funció de transferència.
Comentaris	El model continua sense visualitzar-se.

CAS D'ÚS	Definir Model d'Il·luminació
Descripció	Es vol editar els elements que fan referencia al model d'il·luminació del visor 3D.
Actors	Usuari
Precondició	• No existeix.
Flux Principal	1. Seleccionar color de difusió. (usuari) 2. Seleccionar color especular. (usuari) 3. Seleccionar color de les fonts de llum. (usuari) 4. Introduir valor de brill. (usuari) 5. Si valor de brill incorrecte (sistema) 8. 1 Avisar usuari. (sistema) 6. Escollir la funció de composició (usuari) 7. Seleccionar color fíbres (usuari) 8. Seleccionar tancar l'editor. (usuari)
Flux Alternatiu	En cas de no introduir-se cap valor, el sistema assigna els colors per defecte.
PostCondició	S'han recollit nous valors per a la configuració del visor 3D.
Comentaris	El model continua sense visualitzar-se.

CAS D'ÚS	Definir Funció de Composició
Descripció	Es vol definir una nova funció de Composició
Actors	Usuari
Precondició	• No existeix.
Flux Principal	1. L'usuari escull una nova funció de Composició (usuari)
Flux Alternatiu	En cas de no introduïr-se cap valor, el sistema assigna la funció de 1-α.
PostCondició	S'ha recollit un nou valor per a la configuració de la Funció de Composició.
Comentaris	El model continua sense visualitzar-se.

CAS D'ÚS	Inlusió de Fibres
Descripció	Es vol incloïr els clústers de fibres en el volum 3D
Actors	Usuari
Precondició	• S'han carregat les fibres prèviament.
Flux Principal	1. Seleccionar inclusió de fibres (usuari)
Flux Alternatiu	En cas de no ser activat, les fibres no seran incloses en el model.
PostCondició	S'han inclòs les fibres en el nostre model.
Comentaris	El model continua sense visualitzar-se.

CAS D'ÚS	Interacció amb el Visor 3D
Descripció	Es vol poder girar el volum, aplicar-li translacions, i/o zoom.
Actors	Usuari
Precondició	• Visualització Activa.
Flux Principal	1. L'usuari fa una interacció de rotació, translació, o zoom. (usuari) 2. El volum es modificat segons la interacció.(sistema)
Flux Alternatiu	En cas de no haver interacció, el volum romandrà immòbil.
PostCondició	El volum ha estat girat, traslladat, o se li ha aplicat un zoom.
Comentaris	La visualització estarà d'acord amb la interacció.

CAS D'ÚS	Activar el Visor 3D
Descripció	Es vol activar la visualització del model.
Actors	Usuari
Precondició	• El model ha d'estar creat • S'ha escollit una Funció de Transferència
Flux Principal	1. Seleccionar 3D Reconstruction (usuari) 2. S'aplica la funció de Transferència seleccionada (sistema). 3. Es visualitza l'escena 3D (sistema)
PostCondició	El volum es mostra per pantalla.
Comentaris	La visualització estarà d'acord amb el volum creat.

4.3 Disseny

En aquest apartat ens centrarem en com s'ha portat a terme el disseny per al desenvolupament del projecte.

Per explicar correctament el disseny final, utilitzarem els diagrames de classes corresponents. En aquests diagrames únicament es mostrarà les classes desenvolupades en aquest projecte i si tenen alguna relació amb alguna classe del DTIWeb, també es mostraran les relacions entre elles. Després de cada diagrama, cada classes serà estudiada, tot explicant els seus mètodes.

Diagrama de Classes

A la figura 4.2 de a continuació, es mostra el diagrama de classes que forma la nostra aplicació, i que explicarem a continuació.

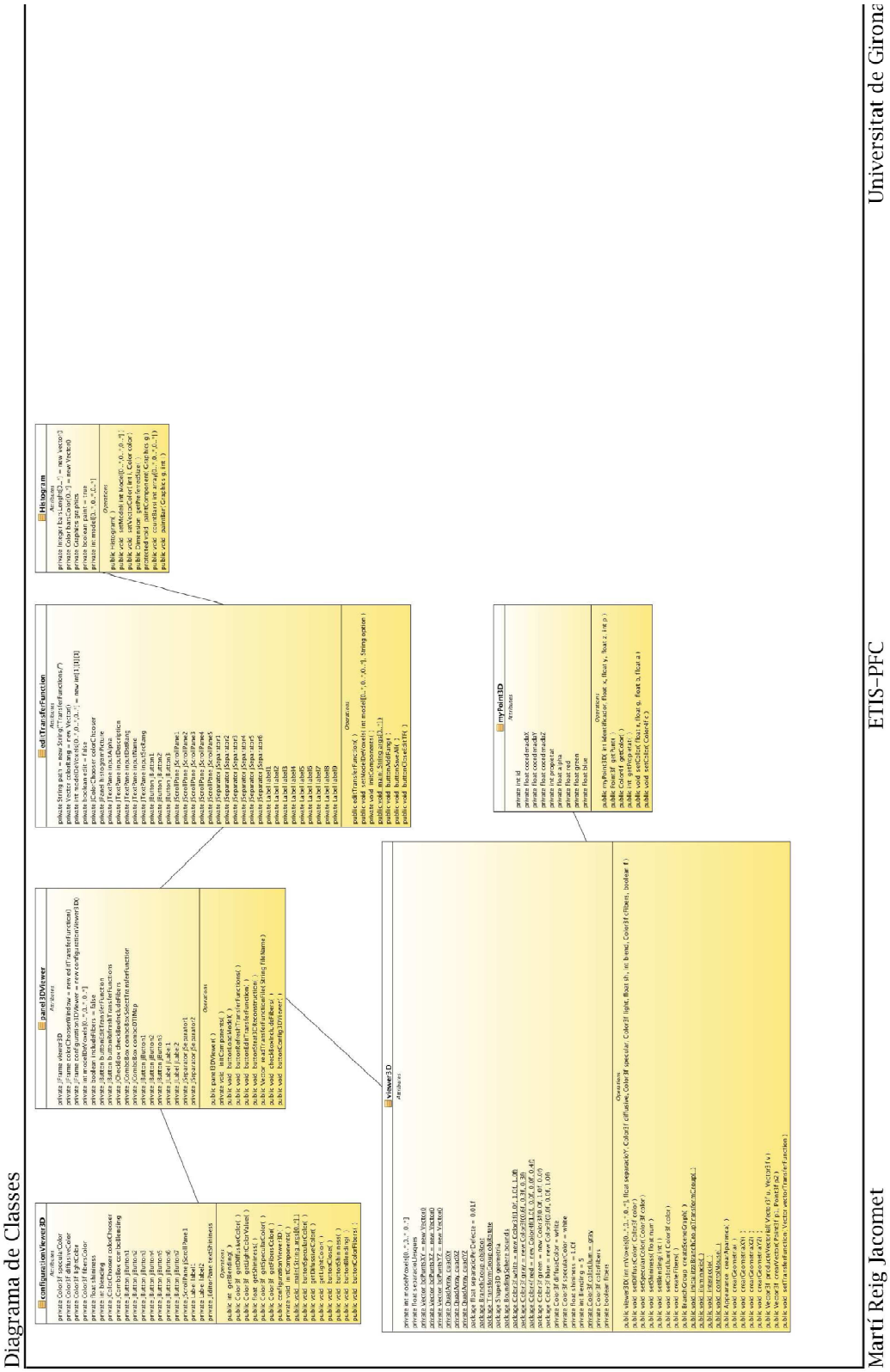

Figura 4.2. Diagrama de Classes de l'aplicació desenvolupada.

Descripció de les Classes

En aquest apartat estudiarem cada classe, fent una breu descripció de cadascuna, i una explicació de cada mètode propi.

La nostra aplicació esta formada per dues classes a la capa d'aplicació, la classe Viewer3D.java, i la classe myPoint3D.java. A la capa d'interfícies tenim 3 classes més, Panel3DViewer.java, configurationViewer3D.java, i editTransferFunction.java.

- **Viewer3D.java**

  Aquesta classe extends de JFrame.

  S'encarrega de crear tot l'univers virtual que formarà la nostra escena.

  També portarà a terme els mètodes necessaris per a la construcció del volum que representarà el nostre model de vòxels.

  - *public viewer3D(int[][][] mVoxels, float separacioY,Color3f diffusive,Color3f specular,Color3f ligt,float sh,int blend)*
    - És el constructor.
    - Primerament inicialitzem les mides inicials en què es mostrarà el JFrame. guardem el Model de Vòxels entrat per paràmetres, i la separació entre llesques.
    - Recull tots els paràmetres necessaris per a la configuració del Visor 3D.
    - Fem les inicialitzacions bàsiques per crear una escena en Java3D, on realment es mostra el codi de reconstrucció es en el mètode createSceneGraph.
  - *createSceneGraph() retorna BranchGroup*
    - Mètode que crea la escena.
    - Aquest mètode porta a terme la creació de tota la escena en 3D. El que fa es fer cridar a diferents mètodes separats pel seus objectius. Finalment compilem el BranchGroup, i el retornem.
    - Concretament, crida a crear la Geometria i les fibres (si es necessari), crear el Bounds, la il·luminació, i el control del mouse, tot seguit compila el BG i el retorna.
  - *inicialitzaBranchGroupTransformGroup ()*
    - Aquest mètode inicialitza el BranchGroup i el TransformGroup. Inicialitzem el BG (objRoot) i el TF (objRotate) que es trobem com a Atributs privats de la classe per més comoditat.
    - També, activem les Capabilities necessàries en el TF.
    - Finalment fem que el TF pengi de l'arbre del BG, perquè així aquest el contengui.
  - *setDiffuseColor(Color3f color)*
    - Inicialitza el color de difusió pel introduït per paràmetres.
  - *setSpecularColor(Color3f color)*
    - Inicialitza el color de especular pel introduït per paràmetres
  - *setShininess(float num)*
    - Inicialitza el valor de brill, pel introduït per paràmetres.
  - *setBlending(int i)*
    - Inicialitza el valor que representa la funció de composició.

- *setColorLlum(Color3f color)*
  - Inicialitza el valor del color de la llum, pel introduït per paràmetres.
- *createFibers()*
  - Aquest mètode s'encarrega de la reconstrucció de les fibres en 3D.
  - Recollim les fibres recollides de les tècniques de tracking de la plataforma DTIWeb, i tot seguit les guardem les fem penjar del transformGroup de l'escena perquè es mostrin correctament, i s'hi pugui interaccionar d'una forma correcta amb la resta de l'escena.

- *iluminacio()*
  - Aquest mètode controla el sistema d'il·luminació de la nostra escena.
  - S'ha seguit la recepta bàsica del tutorial de Java3D de Sun Microsystems.
  - Les llums utilitzades han estat una llum ambiental i varies llums Spot.
  - El color de la llum és agafat des dels atributs de la classe.
- *controlMouse()*
  - Aquest mètode controla el Mouse. Rotació, Translació, i Zoom.
  - Aquesta classe, utilitza les llibreries incloses en el Java3D, per permetre una interacció directe entre el ratolí del Usuari i la geometria mostrada.
  - Amb el botó primari del ratolí, l'usuari pot fer qualsevol rotació a la geometria.
  - Amb el botó secundari del ratolí, l'usuari pot fer qualsevol translació a la geometria.
  - Amb el botó terciari (o rodeta) del ratolí, l'usuari pot fer qualsevol zoom a la figura.
- *crearAparenca() retorna Appearance*
  - Aquest mètode crea la Aparença de les Geometries.
  - El primer que fem es crear un material, aquest te uns colors de difusió i especulació, que en el nostre cas seran blanc per defecte, a no ser que s'hagi canviat per part de l'usuari.
  - També té una coeficient de brillantor, que en el nostre cas també serà escollit per l'usuari, i en cas contrari el valor per defecte serà 1.0, per aconseguir uns efectes de il·luminació més verdaders.
  - Amb el serColorTArget, activem la forma de activar els blancs, el nostre cas hem escollit de forma ambient i difosa, perquè tot quedi millor il·luminat.
  - A continuació, creem un objecte ColoringAttributes, el qual li activem el model de ombra Gouraud.
  - Posteriorment, fem un objecte TransparencyAttributes, on activem la opció de utilitzar les funcions de composició (Blending) de les llibreries de OpenGL, també escollida per l'usuari.
  - També, hem creat un objecte PolygonAttributes, on hem activat el setCullFace, el qual fa que s'activi la visualització de la "cara del darrera" de la geometria.
  - Finalment, cada objecte creat, ha estat inclòs en l'objecte final Appearance.
- *crearGeometria()*

- Creem la Geometria.
- En aquest cas crearem la representació de un model de Vòxels. Per crear el model de Vòxels hem optat per la opció de visualització basada en textures. Es tracta de crear primer tots els plans XY multiplicat per Z vegades, tot seguit, crear els plans XZ multiplicat per Y vegades, finalment, crear els plans YZ multiplicat per X vegades, d'aquesta format, si tots els plans estan en correlació en un mateix origen de coordenades, aconseguirem que es sobreposin, i obtindrem una figura en 3D.
- Tot seguit estudiarem la visualització de volums basada en textures.

Anteriorment hem explicat que en el nostre projecte hem interpretat la visualització de vòxels com a cel·les.

Tot seguit, hem de portar a terme algun algorismes per a la visualització de totes les cel·les del nostre volum, en el nostre projecte hem optat per a la visualització basada en textures.

Aquest algorisme, interpreta el model de vòxels com a un conjunt de plans, anomenats textures, que es distribueixen de forma paral·lela, recorrent tot el model. En un sistema ortogonal, podríem dir que tindríem *z* plans (x,y). Això però ens presenta un problema, si girem $\frac{\pi}{2}$ en l'eix de *y*, veiem que els plans (al no tenir gruix) desapareixen, per tant, introduïm més plans, però amb el gir indicat, en conseqüència obtenim *x* plans (y,z). Tot i això, el problema es repeteix al girar $\frac{-\pi}{2}$ sobre l'eix de *x*, on la solució es tracta de fer *y* plans (x,z). Finalment veiem que tenim el volum per acabat, en el qual hem fet tot una col·lecció de plans per a cada eix de coordenades, tal i com es pot veure a la figura 4.3.

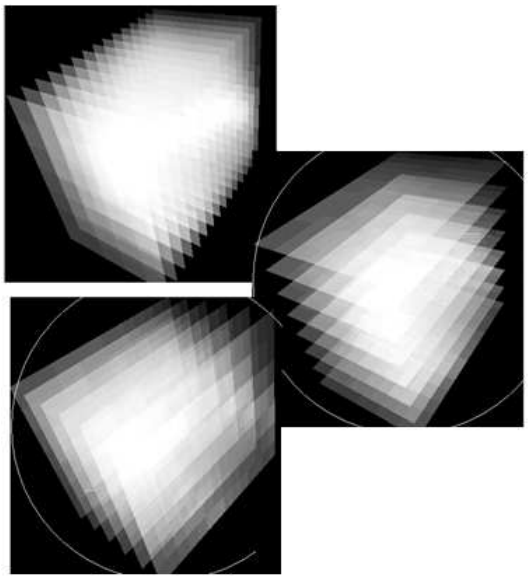

Figura 4.3. Imatge on es mostren les repeticions de plans per cada eix, on les tres superposades farien el volum complet.

- *crearGeometriaXY()*
    - En aquest mètode ens dedicarem a crear els plans XY per Z vegades.
    - El procediment a seguir, és el següent:
    - Obtenim les resolucions.
    - Creem un objecte QuadArray, amb totes les Capabilities que necessitem.
    - Anirem recorrent cada pla del model de Vòxels i anirem creant petits quadrats. Fem incís, que per crear un quadrat l'ordre dels vèrtex ha de ser antihorari.
    - A cada quadrat, li assignarem les Normals corresponents, perquè les rajos de llum quan hi apliquem la il·luminació reflecteixin de forma correcta. Per calcular les normals, es tracta de agafar 2 vèrtex oposats, i per cada vèrtex crear 2 vectors amb origen a ells i destí ens els seus vèrtex dels costats. Després es calcula el producte vectorial d'aquests 2, i seguint la Llei de la Mà Dreta per aconseguir el sentit desitjat, després els vectors són normalitzats i aplicats a tots els vèrtexs del quadrat. També, cada punt creat és guardat a un Vector que ens fa de Base de Dades dels punts, on hi guardem un objecte myPoint3D, i aquest ens guarda tot el referent a les característiques necessàries d'aquest, això posteriorment ens servirà per poder re-pintar sense problemes la geometria.
    - Un cop creats, tots els quadrats de tots els plans, i tinguem la geometria completa, li assignarem la seva Aparença amb el mètode corresponent.
    - Finalment farem penjar el resultat del TransformGroup ojbRotate.
- *crearGeometriaXZ()*
    - En aquest mètode ens dedicarem a crear els plans XZ per Y vegades.
    - El procediment a seguir, és el següent:
    - Obtenim les resolucions.
    - Creem un objecte QuadArray, amb totes les Capabilities que necessitem.
    - Anirem recorrent cada pla del model de Vòxels i anirem creant petits quadrats. Fem incís, que per crear un quadrat l'ordre dels vèrtex ha de ser antihorari.
    - A cada quadrat, li assignarem les Normals corresponents, perquè les rajos de llum quan hi apliquem la il·luminació reflecteixin de forma correcta. Per calcular les normals, es tracta de agafar 2 vèrtex oposats, i per cada vèrtex crear 2 vectors amb origen a ells i destí ens els seus vèrtex dels costats. Després es calcula el producte vectorial d'aquests 2, i seguint la Llei de la Mà Dreta per aconseguir el sentit desitjat, després els vectors són normalitzats i aplicats a tots els vèrtexs del quadrat. També, cada punt creat és guardat a un Vector que ens fa de Base de Dades dels punts, on hi guardem un objecte myPoint3D, i aquest ens guarda tot el referent a les característiques necessàries d'aquest, això posteriorment ens servirà per poder re-pintar sense problemes la geometria.
    - Un cop creats, tots els quadrats de tots els plans, i tinguem la geometria completa, li assignarem la seva Aparença amb el mètode corresponent.
    - Finalment farem penjar el resultat del TransformGroup ojbRotate.

- *crearGeometriaYZ()*
  - En aquest mètode ens dedicarem a crear els plans YZ per X vegades.
  - El procediment a seguir, és el següent:
  - Obtenim les resolucions.
  - Creem un objecte QuadArray, amb totes les Capabilities que necessitem.
  - Anirem recorrent cada pla del model de Vòxels i anirem creant petits quadrats. Fem incís, que per crear un quadrat l'ordre dels vèrtex ha de ser antihorari.
  - A cada quadrat, li assignarem les Normals corresponents, perquè les rajos de llum quan hi apliquem la il·luminació reflecteixin de forma correcta. Per calcular les normals, es tracta de agafar 2 vèrtex oposats, i per cada vèrtex crear 2 vectors amb origen a ells i destí ens els seus vèrtex dels costats. Després es calcula el producte vectorial d'aquests 2, i seguint la Llei de la Mà Dreta per aconseguir el sentit desitjat, després els vectors són normalitzats i aplicats a tots els vèrtexs del quadrat. També, cada punt creat és guardat a un Vector que ens fa de Base de Dades dels punts, on hi guardem un objecte myPoint3D, i aquest ens guarda tot el referent a les característiques necessàries d'aquest, això posteriorment ens servirà per poder re-pintar sense problemes la geometria.
  - Un cop creats, tots els quadrats de tots els plans, i tinguem la geometria completa, li assignarem la seva Aparença amb el mètode corresponent.
  - Finalment farem penjar el resultat del TransformGroup ojbRotate.
- *producteVectorial(Vector3f u, Vector3f v) retorna Vector3f*
  - Aquest mètode ens fa el producte Vectorial de uxv el producte Vectorial. Recordar, que el sentit del resultat del nou Vector, va en relació a la Llei de la Mà Dreta.
- *crearVector(Point3f p1, Point3f p2) retorna Vector3f*
  - Aquest mètode ens crea un vector amb origen el Punt p1, i destí el Punt p2. Per fer el Vector, es resten les components de cada punt.
- *setTransferFunction(Vector vectorTransferFunction)*
  - Aquest mètode serveix per re-pintar la nostra escena 3D.
  - El que fem és, a través del Vector rebut per paràmetres, que conté els valors de la nova funció de transferència, actualitzem les 3 Bases de Dades dels punts (una base de dades per cada tipus de pla: XY*Z, XZ*Y, i YZ*X). Un cop actualitzats el punts, fem que els QuadArray's s'actualitzin cada punt amb el seu nou color accedint a la base de dades.

- **myPoint3D.java**

  Aquesta classe crea un objecte myPoint3D, un objecte myPoint3D, ens crea un punt amb les seves coordenades a l'espai 3D, i contenen una propietat, posteriorment (a través d'una funció de transferència) aquestes propietats són canviades per colors RGB.

  - *myPoint3D(int identificador, float x, float y, float z, int p)*
    - Constructor.
    - Crea un Objecte myPoint3D, amb l'identificador corresponent, les seves coordenades, i el valor de la propietat (que serà entrat per el valor d'algun Model de Vòxels).
  - *getPunt()retorna Point3f*
    - Retorna un Objecte Point3f, amb les coordenades del myPoint3D.
  - *getColor() retorna Colo4f*
    - Retorna un Objecte Color4f, que és el color del myPoint3D.
  - *getPropietat() retorna int*
    - Retorna el valor de la Propietat.
  - *setColor(float r, float g, float b, float a)*
    - Canvia el Color del myPoint3D, pels valors introduïts per paràmetres.
  - *setColor(Color4f c)*
    - Canvia el Color del myPoint3D pel Color introduït per paràmetres.
- **Panel3DViewer.java**

  Aquesta classe extends de JPanel. Aquesta classe crea la interfície Gràfica, del panell "3D Viewer" del JavaViewer de la Plataforma DTIweb.

  Només farem constància dels mètodes creats purament per a nosaltres, els mètodes creats automàticament per la API de Java Swing no en farem constància.

  - *panel3DViewer()*
    - Constructor.
    - Incialitza tot l'entorn gràfic per a poder veure el JPanel.
  - *buttonLoadModel()*
    - Porta a terme l'execució del botó "Load Model" del panell.
    - Aquest botó recull tota la informació referent a la configuració del Visor 3D. Posteriorment, carrega el model de vòxels que es troba a la plataforma DTIWeb,.
    - Finalment fa una crida al Viewer3D per a crear el volum que representi el model, aquest volum es creat de color "transparent".
  - *buttonRefreshTransferFunctions()*
    - Porta a terme l'execució del boto "Refresh Transfer Functions" del panell.
    - Aquest mètode llegeix el directori que conté els fitxers de les funcions de transferència, i actualitza la llista desplegable de Funcions de Transferència.
  - *buttonEditTransferFunction()*
    - Porta a terme l'execució del botó "Edit Transfer Function" del panell.
    - Obra una nova finestra, on permet la creació, per part de l'usuari, d'una nova funció de transferència.
  - *buttonStrat3DReconstruction()*

        - Porta a terme l'execució del botó "Start 3D Reconstruction" del panell.
        - Aplica la funció de transferència seleccionada a la llista desplegable en el volum que representa el model de vòxels, tot seguit activa la visualització del volum, i aquest es mostrat per pantalla a través d'una nova finestra.
    - *readTransferFunctionFile(String fileName) retun Vector*
        - S'encarrega de buscar la funció de transferència amb el nom que sigui el mateix introduït per paràmetres.
        - Un cop localitzat el fitxer que conté la funció de transferència, l'interpreta, i la carrega tota en un vector el qual retorna
    - *buttonConfig3DViewer()*
        - Aquest mètode porta a terme l'execució del botó "Configuration 3D Viewer".
        - S'encarrega d'obrir una nova finestreta, la qual permet la edició dels paràmetres de configuració del visor 3D.

- **configurationViewer3D.java**

  Aquesta classe extends de JFrame.

  Mostra una nova finestreta, que permet configurar el visor3D, per part de l'usuari.
    - *configurationViewer3D()*
        - Constructor.
        - Inicialitza tots els paràmetres necessaris per a mostrar la finestreta de configuració del visor 3D.
    - *buttonBlending()*
        - Obté la funció de composició seleccionada per l'usuari.
    - *getDiffusiveColor()*
        - Obté el color seleccionat per l'usuari i l'identifica com a color de diffusió del volum.
    - *getLightColor()*
        - Obté el color seleccionat per l'usuari i l'identifica com a color de les fonts d'il·luminació.
    - *buttonShininess()*
        - Obté el valor entrat per teclat, i l'identifica com a valor de brill del material del volum.
    - *buttonSpecularColor()*
        - Obté el color seleccionat per l'usuari, i l'identifica com a color especular del material que conformarà el volum.
    - *buttonClose()*
        - Guarda tots els valors seleccionats per l'usuari.
    - *getSpecularColor() retorna Color3f*
        - Retorna el color especular.
    - *getDiffusiveColor() retorna Color3f*
        - Retorna el color de Difusió.
    - *getShininess() retorna float*
        - Retorna el valor de brill.
    - *getLightColorValue() retorna Color3f*
        - Retorna el color de la llum.

- *getBleding() retorna int*
  - Retorna un enter que identifica la funció de composició escollida.
- *buttonColorFibers()*
  - Obté el color seleccionat per l'usuari, i l'identifica com a color de les fibres.
- *getFibersColor() retorna Color3f*
  - Retorna el color de les fibres.

- **editTransferFunction.java**

  Aquesta classe extends de JFrame. Aquesta classe es de suport a la classe panel3DViewer. La funció de la classe es permetre la creació/edició de Funcions de Transferència (Transfer Functions). Per crear una classe el procediment a seguir és el següent: _escrivim el nom de la Funció de Transferència. _escrivim la descripció de la Funció de Transferència. _Tenim un histograma que ens mostra el numero de píxels per cada propietat del Model De Vòxels. _Tenim una paleta de color per escollir el color. _Tenim per escollir un valor de Transparència (alpha). _Hem de escollir rangs de propietats i assignar-li el color i la transparència. _Recordar: que el rang màxim és 0-176, el qual tot es mostraria igual. _Un cop fet tot, Save All, i després Close.

  - *setModelDeVoxels(int[][][]model)*
    - Aquest mètode ens serveix per poder actualitzar el Model de Voxels des de l'exterior de la classe.
    - També, repintem el histograma amb el nou Model de Vòxels Rebut.
  - *Main*
    - Aquest mètode porta a terme la execució principal del JFrame.
    - El que fa, es crear un nou Thread (veure com crea el Runnable) on passa per paràmetres el seu codi, i es posa com a visible.
  - *buttonAddRang()*
    - Aquest mètode llegeix el color RGB-α introduït per l'usuari, i l'aplica al rang seleccionat per l'usuari.
    - A més col·lorejar d'histograma per saber la zona que hem pintat.
    - Tots els rangs de color queden guardats a un vector.
  - *buttonSaveAll()*
    - Aquest mètode s'encarrega de crear un arxiu amb el nom i la descripció de la funció de transferència, i li assigna tots els colors emmagatzemats pel vector.
    - Aquest arxiu serà guardat al directori de les funcions de transferència.

  - *buttonCloseEditTF()*
    - Aquest botó inhabilita la visualització de la actual finestra de editar funcions de transferència.

4.4 Implementació

En aquest apartat en centrarem en el punt de vista del flux de l'execució.

Per fer-ho, farem els diagrames de seqüència corresponents als casos d'ús que hem explicat anteriorment. Tot i això no mostrarem els diagrames de seqüència de tots els casos d'ús, sinó que ens centrarem en els més importants.

En els diagrames de seqüència ens trobarem, que algunes crides són fetes amb pseudocodi i engloben altres crides, això ha estat fet així per a la millor comprensió per part del lector, ja que sinó, el diagrama s'hauria allargat molt i podria provocar confusió.

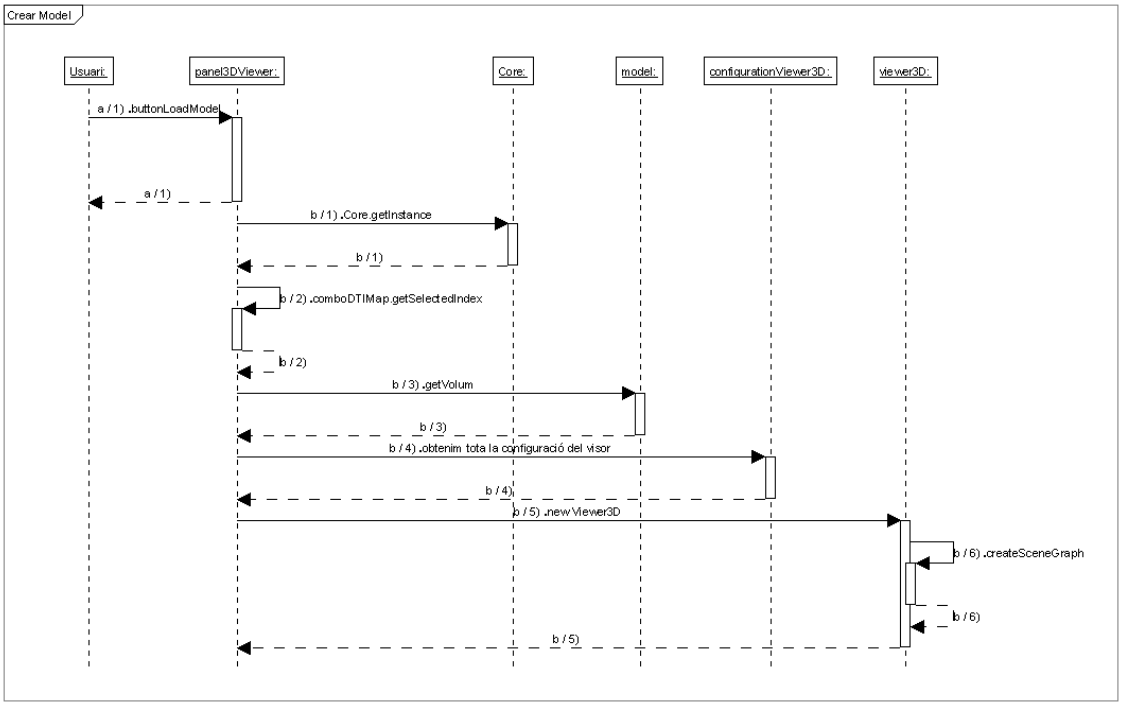

Figura 4.3 Diagrama de Seqüència Crear Model

A la figura 4.3 podem veure el diagrama de seqüència del cas d'ús crear model. Quan l'usuari fa la crida a crear el model, resumidament el que es fa és el que explicarem a continuació. Es recull tota la informació a la configuració 3D que ha introduït prèviament l'usuari, com el color de difusió, el color especular, i la brillantor del material de la geometria, el color de les fonts de llum, el color de les fibres (si es volen representar), i la funció de composició.

Un cop recollida tota aquesta informació, es recull el Mapa de DTI escollit per l'usuari. Tot seguit, es procedeix a la crida d'un objecte Viewer3D, el qual amb tota la informació abans esmentada crearà la escena amb el model representat, però aquest es trobarà pintat amb color de fons (negre) amb el 0% d'opacitat, cosa que el farà transparent.

El visor estarà desactivat, i per tant, l'usuari no podrà veure el resultat.

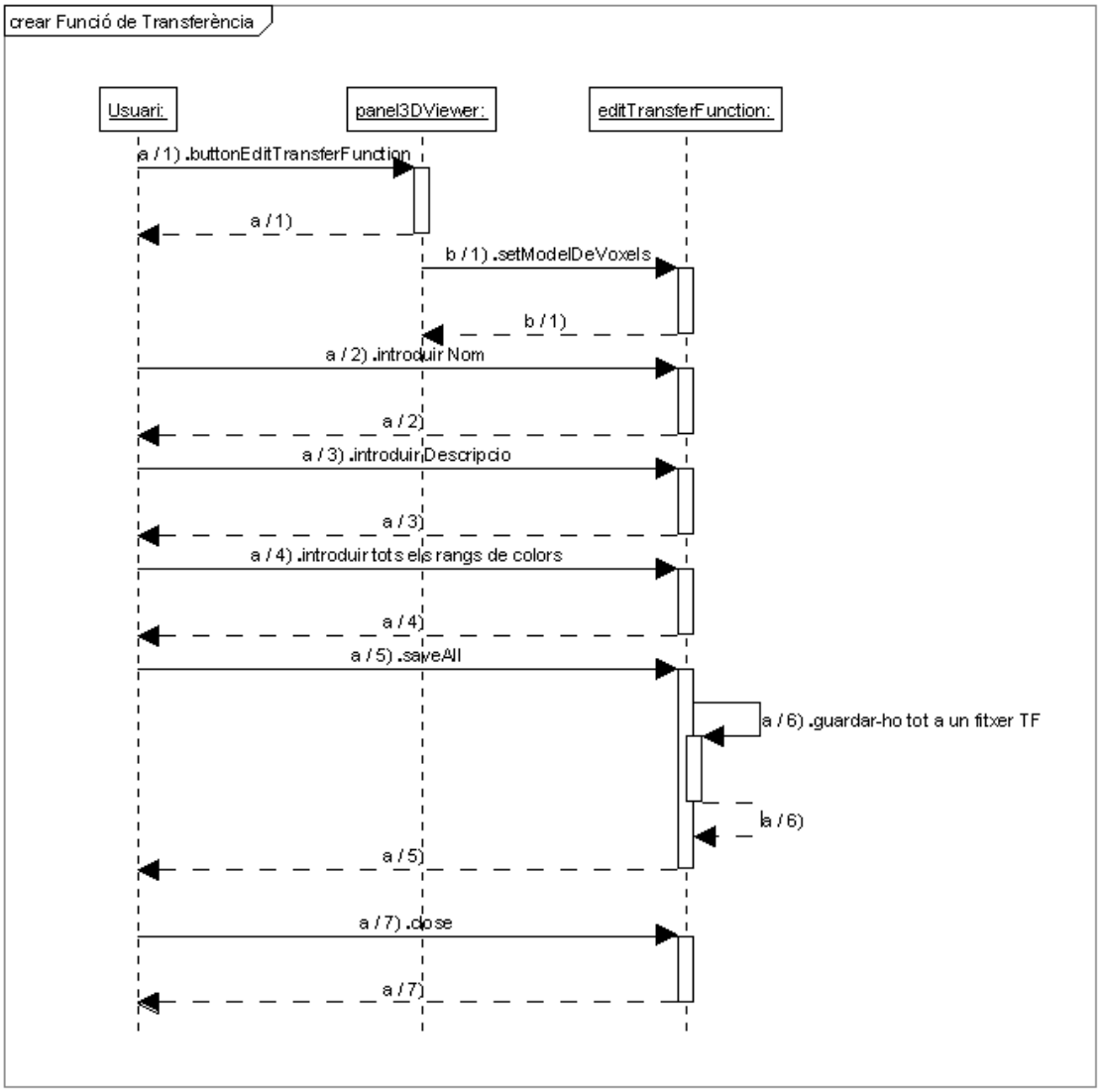

Figura 4.4 Diagrama de Seqüència de Crear la Funció de Transferència

En aquest diagrama (figura 4.4) es pot observar que el procediment es el següent. L'usuari fa obrir el panel de edició de funcions de transferència, tot seguit introdueix tots els paràmetres necessaris, i finalment quan l'usuari permet que tot es guardi, es crea un fitxer amb tots els valors.

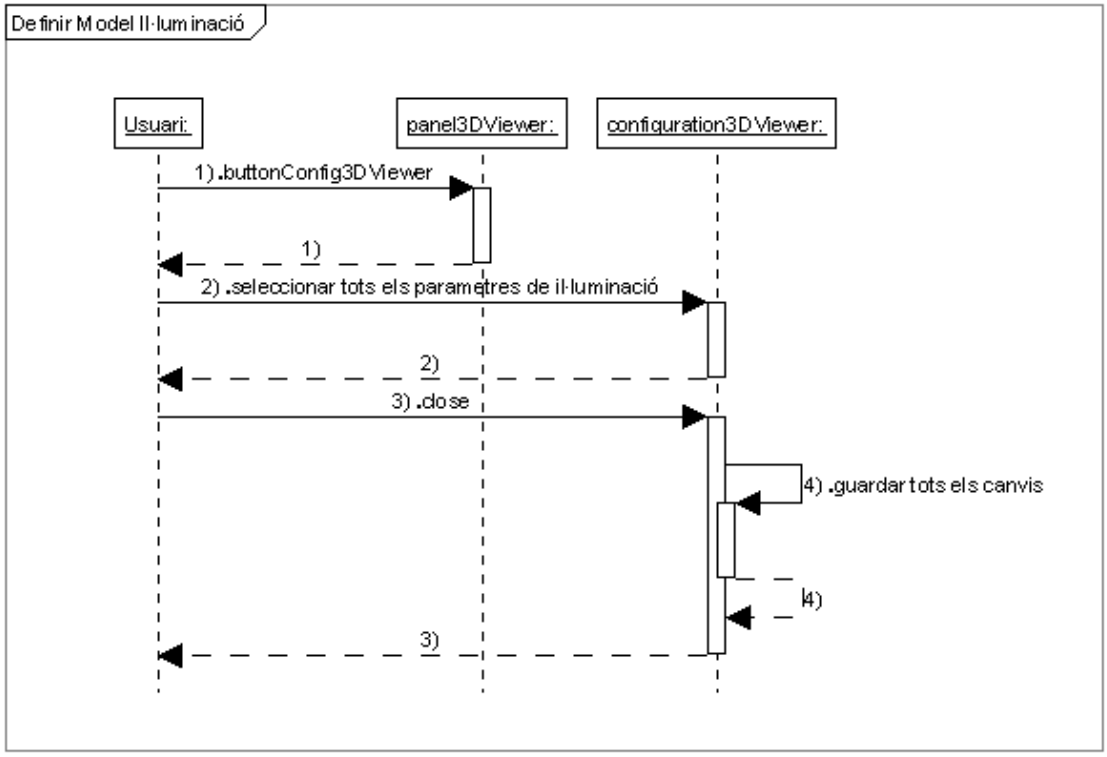

Figura 4.5 Diagrama de Seqüència de Definir el Model de Il·luminació.

En el diagrama de la figura 4.5, podem veure que per definir el model d'il·luminació, s'obre el panel de edició del visor 3D, i tot seguit l'usuari introdueix tots els paràmetres necessaris com són, el color de difusió del material, el color especular, el valor de brill, el color de les fonts d'il·luminació. Finalment, quan l'usuari decideix tancar el panel, aquest guarda tota la configuració.

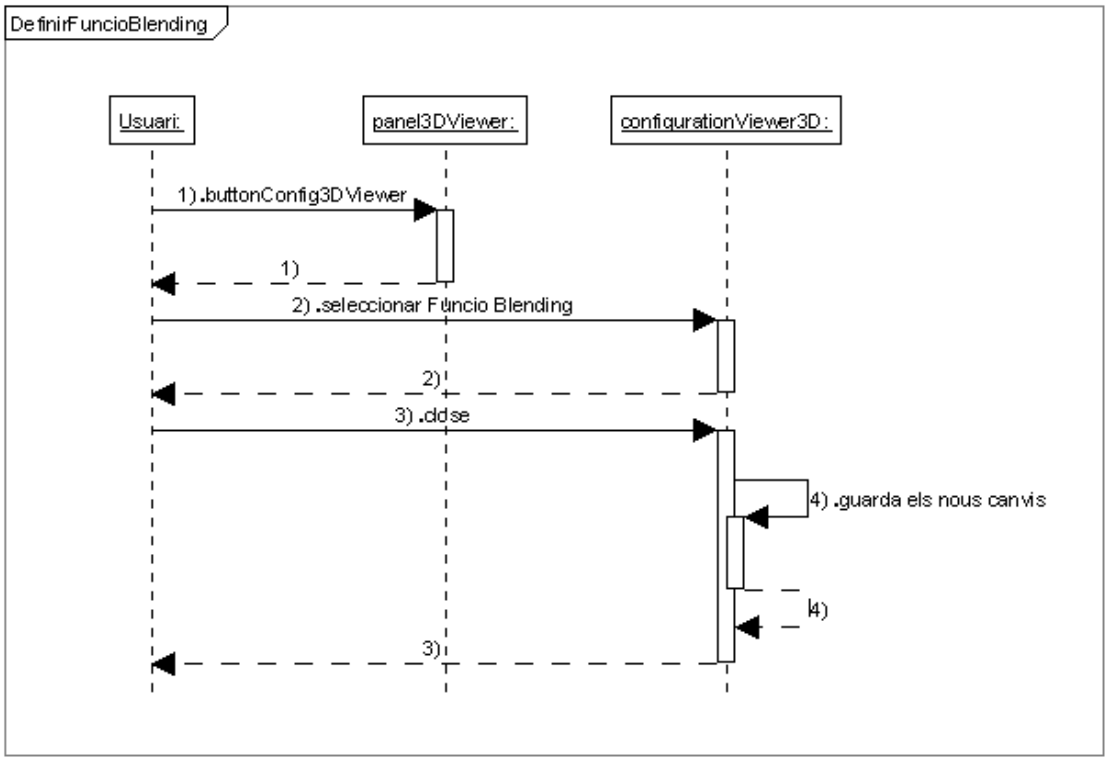

Figura 4.6 Diagrama de Seqüència de Definir Funció de composició

En el diagrama mostrat a la figura 4.6, podem veure el diagrama de seqüència representatiu de Definir una Funció de composició (Blending), On l'usuari ha d'obrir el panel de configuració del visor 3D, i escollir l'equació de la funció de composició que més li correspongui.

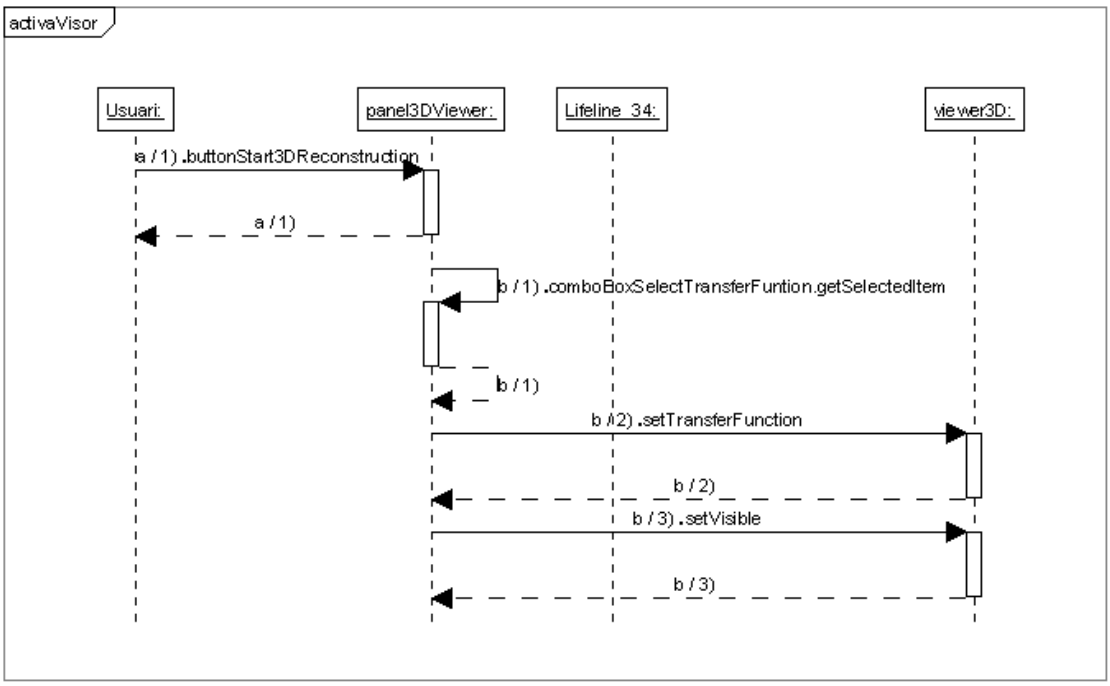

Figura 4.7 Diagrama de Seqüència de Activar Visor

A la figura 4.7 es mostra el diagrama de seqüència de activar el visor. L'usuari un cop decideix activar-lo, aquest fa una crida a re-pintar el volum amb la funció de transferència escollida per l'usuari, això ens permet poder canviar de funció de transferència sense haver de tornar a carregar el model. Un cop el model ha estat re-pintat, s'activa a visible, i aquest es mostra per pantalla.

Capítol 5. Interfícies

En aquest capítol, mostrarem la interfície gràfica que hem utilitzat per a la nostra aplicació, on també explicarem que fan cadascuna de les opcions en els menús que ens puguem trobar.

Inicialment, quan la plataforma DTIWeb ha estat oberta, la podem visualitzar de la següent forma, tal i com es mostra a la figura 5.1.

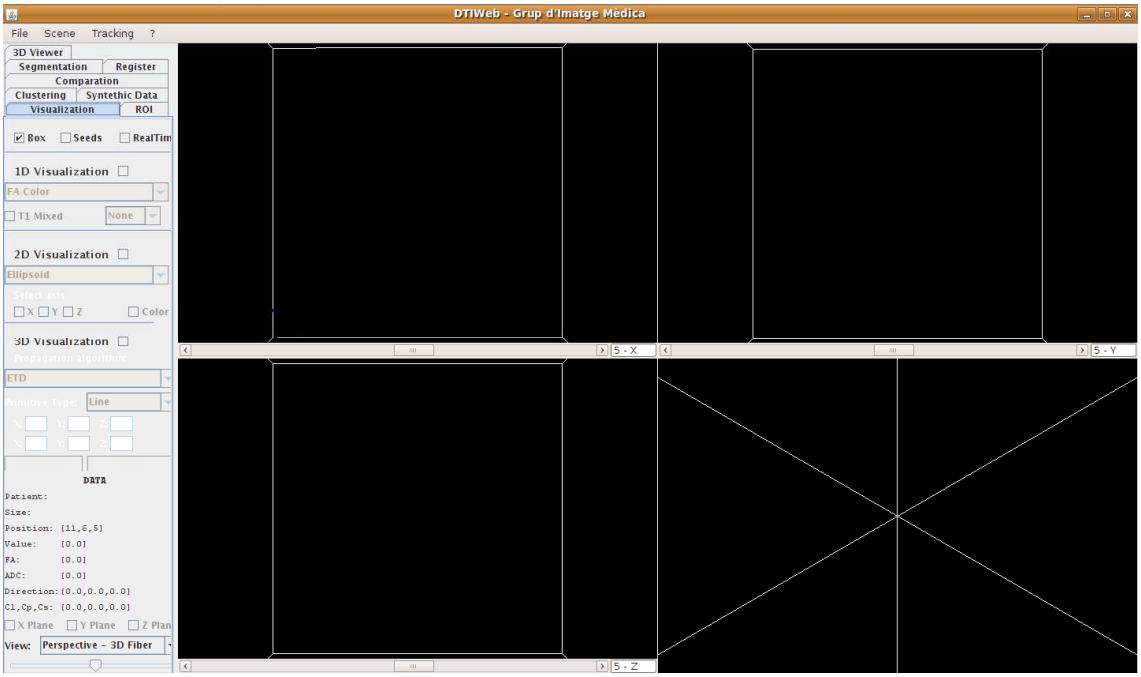

Figura 5.1. Imatge que mostra la plataforma DTIWeb on cop està en execució.

Un cop la plataforma està carregada, per poder carregar un fitxer d'un pacient hem de seguir el següent procediment *File → Open* on seleccionarem el fitxer ha carregar.

Un cop el fitxer ha estat carregat, per poder tenir els clústers de fibres calculats, hem de seleccionar les opcions que es mostren a la figura 5.2

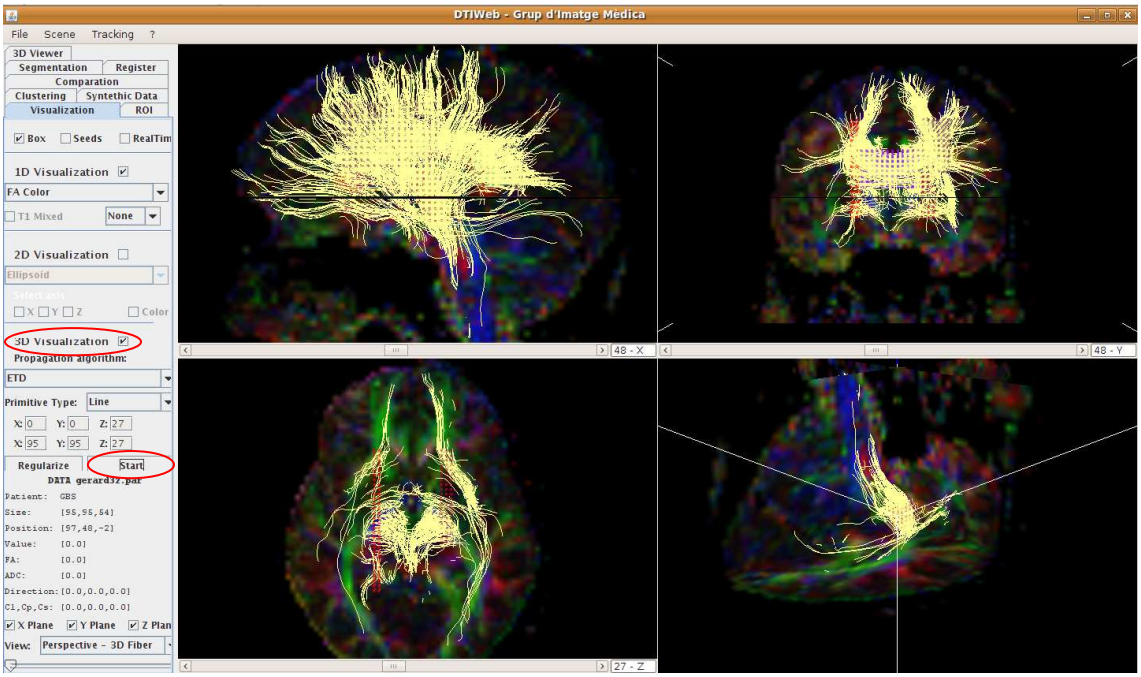

Figura 5.2. Imatge que mostra com es carreguen els clústers de fibres.

Tot seguit, podem dirigir-nos a la pestanya de la nostra aplicació *3D Viewer* on el menú canviarà i es mostrarà com es veu a la figura 5.3.

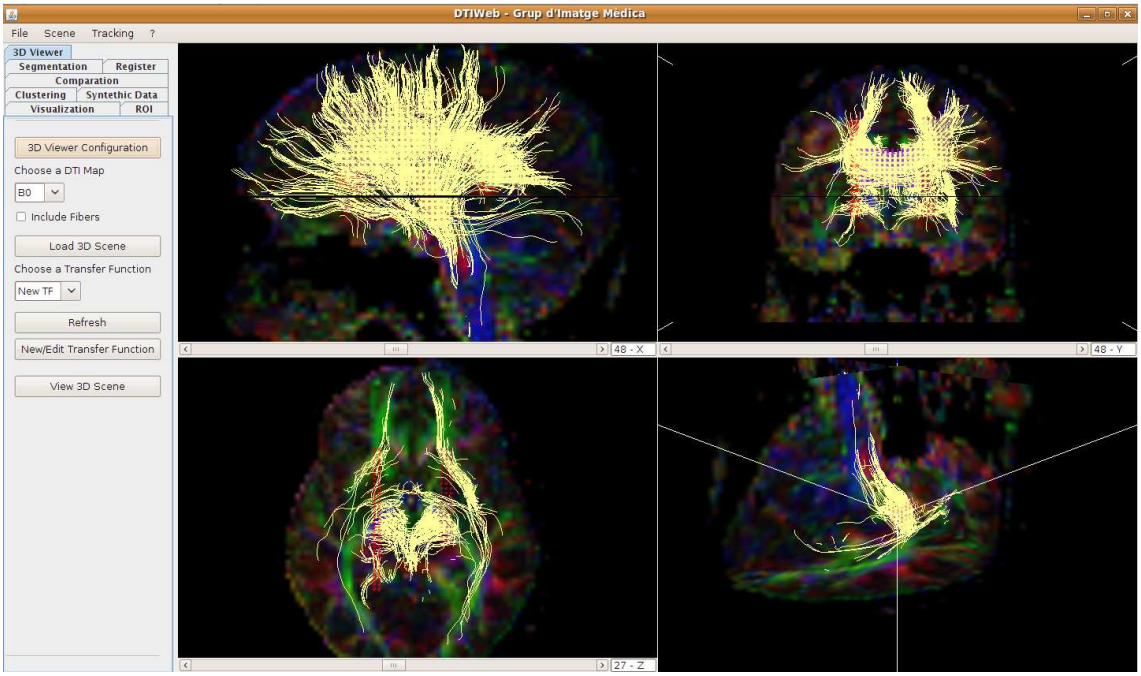

Figura 5.3. Imatge que mostra la pestanya *3D Viewer*, la qual é son es troba la nostra aplicació.

Seguint un sentit descendent, el primer boto que trobem *3D Viewer Configuration* ens permet l'obertura d'un editor per configurar el visor 3D (figura 5.4). L'editor està format per un mòdul el qual permet la selecció d'un color per part de l'usuari, i varis botons per assignar el color escollit a on prefereixi l'usuari, aquest pot ser assignat com a color especular del material, color de difusió, color de les fonts de llum, o bé com a color per a pintar les fibres. A més, l'editor ens permet a través de l'entrada per teclat l'elecció d'un valor de brill per al nostre

volum, el qual ha de ser entre 0,0 i 128,0. També esta dotat d'una llista desplegable la qual permet a l'usuari escollir una equació per a la funció de composició. Finalment, quan la configuració ha estat escollida, es disposa d'un botó *Close* el qual finalitza l'editor. Cal remarcar, que si algun d'aquests valors no ha estat escollit, l'aplicació els dotarà dels seus valors per defecte.

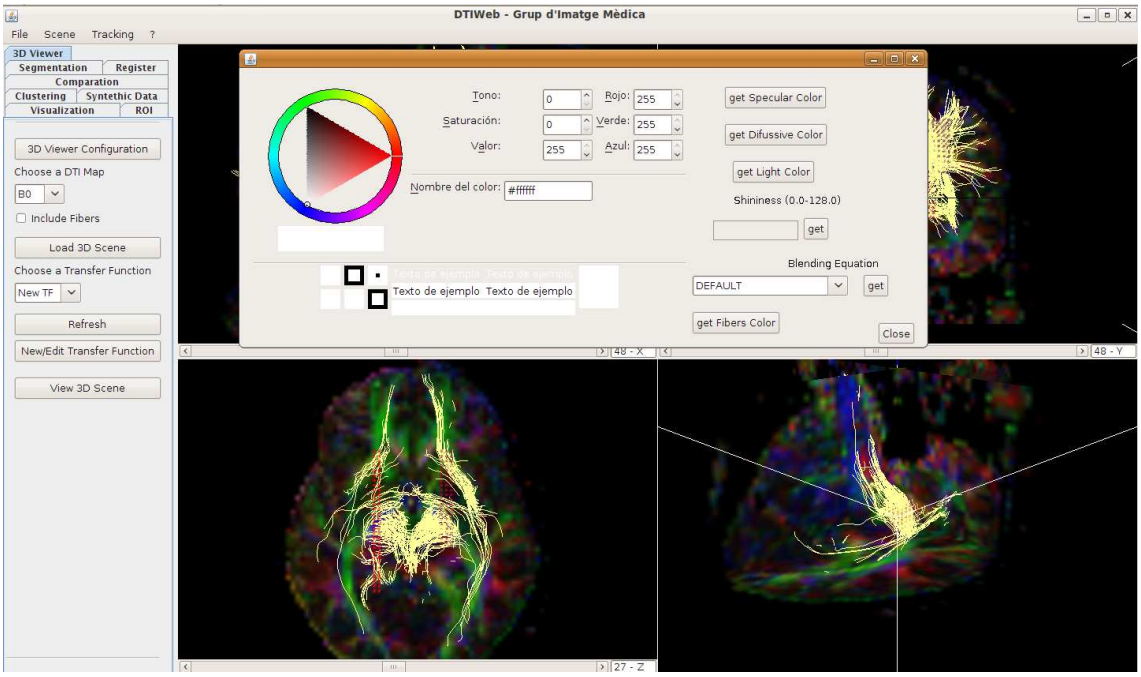

Figura 5.4 Imatge que mostra l'editor per configurar el Visor 3D.

En segon lloc, i seguint la figura 5.3, en el menú de la pestanya de la nostra aplicació, trobem una llista desplegable la qual ens permet la elecció del mapa de DTI que volem carregar, per exemple, podem optar per la B0, la ADC, la VR, entre d'altres.

Tot seguit, es mostra una casella, la qual permet a l'usuari l'elecció d'escollir si vols que es vegin les fibres incloses en el volum o no. En el cas que l'usuari sàpiga prèviament que no vol les fibres, aquestes no haurien estat necessàries de carregar a la pestanya *Visualization*.

Després, ens trobem amb un botó *Load 3D Scene*, el qual permet la construcció de l'escena 3D, amb els valors decidits i explicats anteriorment. Cal destacar que, quan l'escena estigui carregada, aquesta no es mostrarà per pantalla.

Una vegada, la nostra escena ha estat carregada a memòria, tenim una llista desplegable on es mostren totes les Funcions de Transferència que hi ha en el directori de l'aplicació. A sota de la llista, es disposa d'un botó *Refresh* que permet l'actualització de la llista en cas que s'hagin creat o editat noves Funcions de Transferència.

Per crear o editar una nova Funció de Transferència es disposa un botó *New/Edit Transfer Function*, el qual obra un editor de Funcions de Transferència (figura 5.5). Si a la llista desplegable ha estat escollida la opció *New TF* es procedirà a la creació d'una nova Funció de Transferència, mentre que si pel contrari s'ha escollit una Funció de Transferència, l'editor l'obrirà i permetrà la edició d'aquesta.

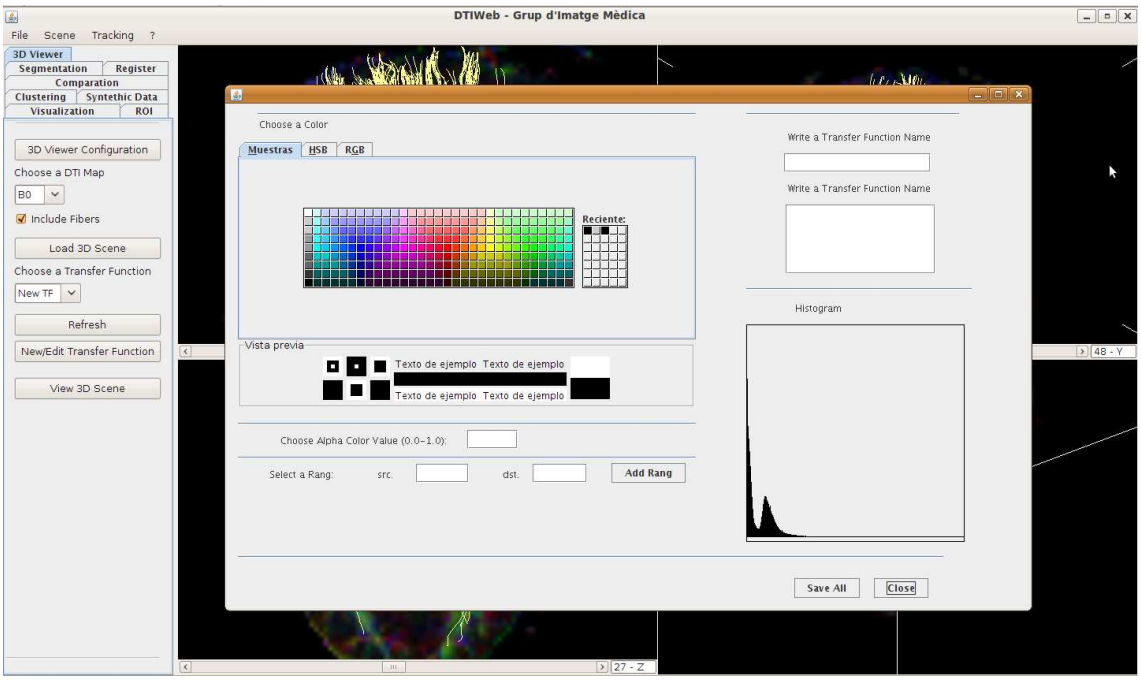

Figura 5.5 Imatge que mostra l'editor de Funcions de Transferència.

En l'editor de les Funcions de Transferència (figura 5.5) podem observar varies opcions, es mostren dos camps per introduir el nom i la descripció d'aquesta respectivament. L'editor porta incorporat d'un histograma que mostra la quantitat de punts que hi ha per valor de propietat. També conté un mòdul on l'usuari pot escollir un color, i el valor de transparència α, aquest color podrà ser assignat a un rang de colors mitjançant el boto *add Rang*. Finalment quan l'usuari ha escollit tot el rang de colors pot guardar la Funció de Transferència *Save All*, i tanca l'editor *Close*.

L'últim botó que es mostra a la pestanya és *View 3D Scene*, aquest recull la Funció de Transferència escollida per l'usuari i l'aplica a l'escena creada anteriorment, un cop ha estat completat el procediment, activa la visualització (figura 5.6) i permet la interacció de rotacions, translacions i zoom, per part de l'usuari.

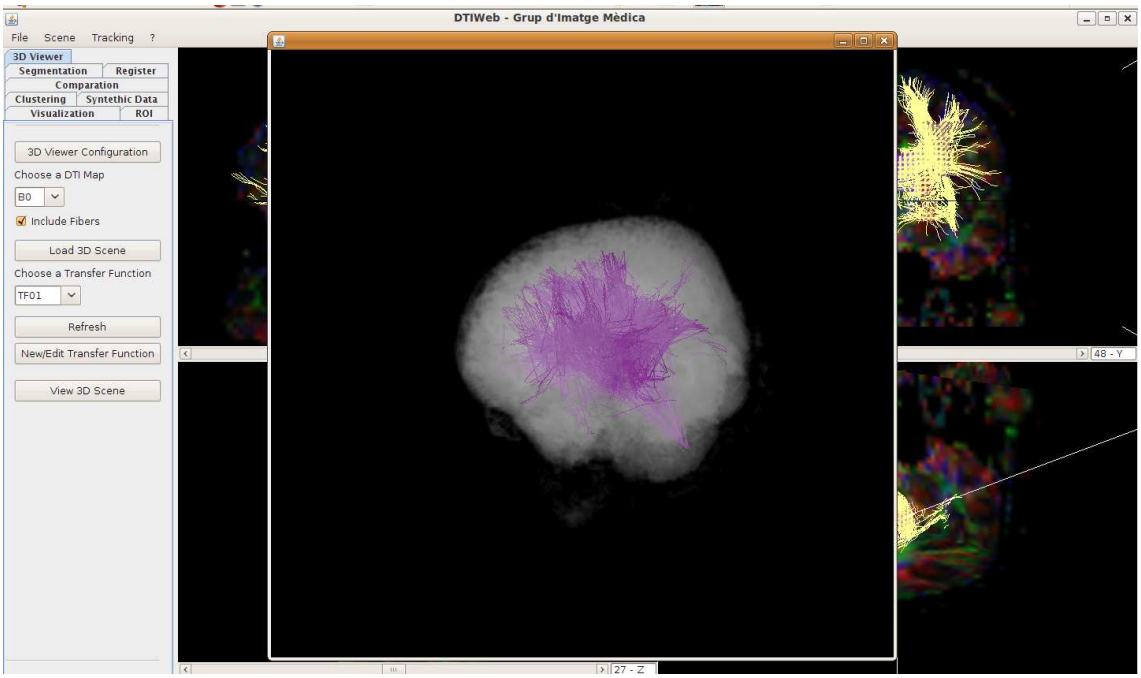

Figura 5.6. Imatge que mostra l'escena 3D.

Capítol 6. Resultats

6.1 Ordinador de Proves

Per al desenvolupament d'aquest projecte hem utilitzat un portàtil, però per fer les últimes proves i retocs per a l'aplicació vam necessitar una màquina amb un hardware més potent. Les característiques importants per a la nostra aplicació de l'ordinador utilitzat per a les proves són les que es mostren a la taula 5.1.

Característica	Component
Microprocessador	Intel® Core(TM) 2 Duo E6750@2,66GHz
Memòria Cau	4Mb
Memòria RAM	3,00GB DDRII
Targeta Gràfica	nVidia GeForce 8500GT 512MB
Sistema Operatiu	Ubuntu 9.04 Jaunty Jackalope 64bits
Entorn Gràfic	Gnome 2.26
Java	JDK 6 Update 16
API Java 3D	API Java 3D 1.5.1

6.2 Proves i Resultats

En aquest apartat mostrarem els resultats obtinguts en varies proves diferents, les quals a cadascuna explicarem les seves característiques. Els temps estaran calculats manualment amb un cronòmetre, per tant i pot haver un cert error.

En totes les proves utilitzarem el mateix model, el qual té unes dimensions de [95,95,54]. També la configuració del visor ha estat la que hi ha configurada per defecte a l'aplicació.

Prova 1

- En aquesta prova visualitza el model sense fibres.
- El Mapa de DTI carregat ha estat: B0.
- La funció de transferència ha estat la que es mostra a la taula de continuació.

Rang	Color
[0,15]	(0,0,0,0.0)
[16,25]	(102,102,102,0.1)
[26,35]	(153,153,153,0.1)
[36,45]	(204,204,204,0.1)
[46,55]	(255,255,255,0.1)
[56,300]	(0,0,0,0.0)

- Temps per carregar l'escena: 11 segons 92 centèsimes
- Temps per pintar el volum i visualitzar-lo: 15 segons 89 centèsimes.
- Seguidament mostrem les captures de pantalla a les figures 6.1, 6.1, i 6.3.

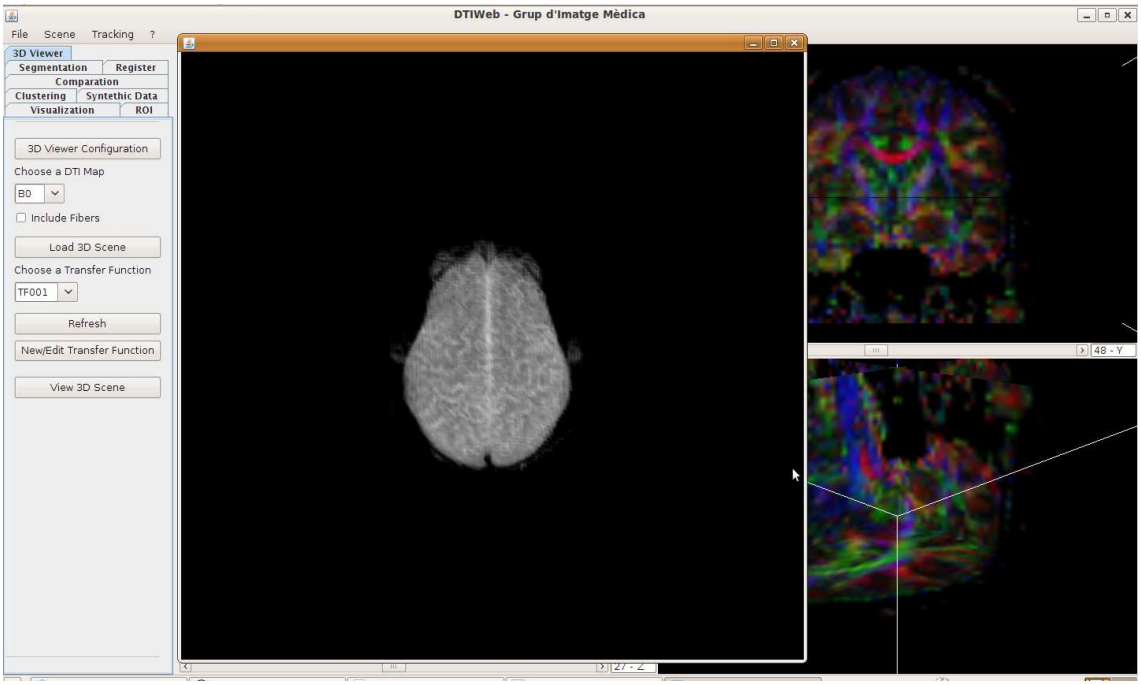

Figura 6.1. Imatge on es el volum de la prova 1 des de dalt.

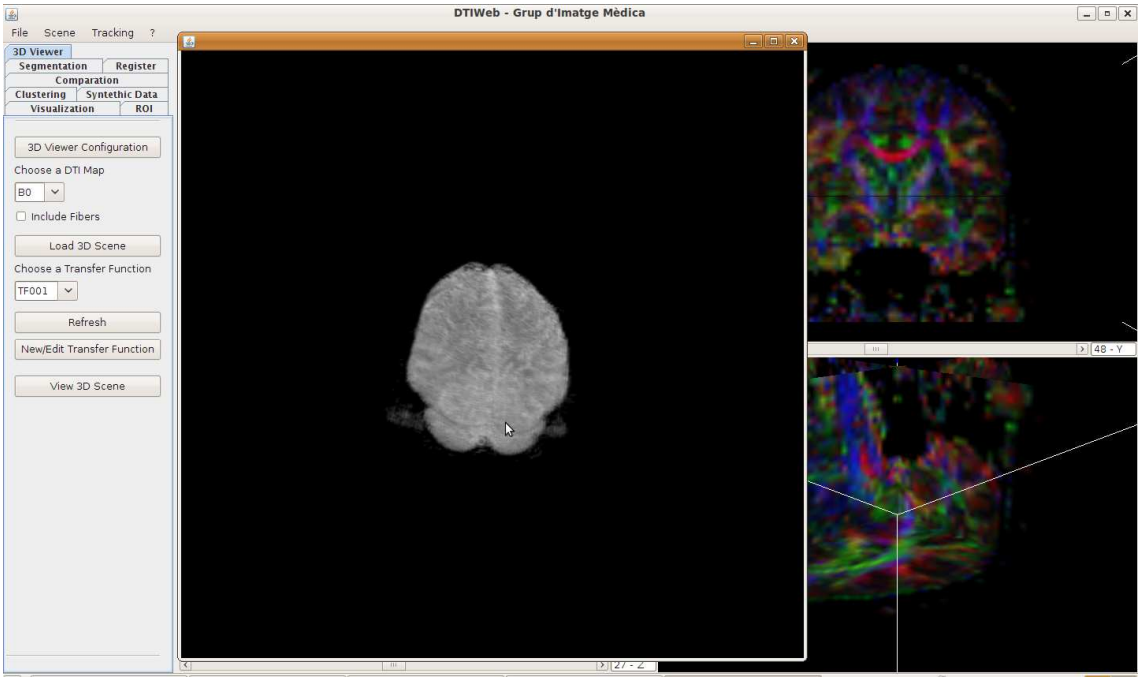

Figura 6.2. Imatge on es el volum de la prova 1 des de darrera

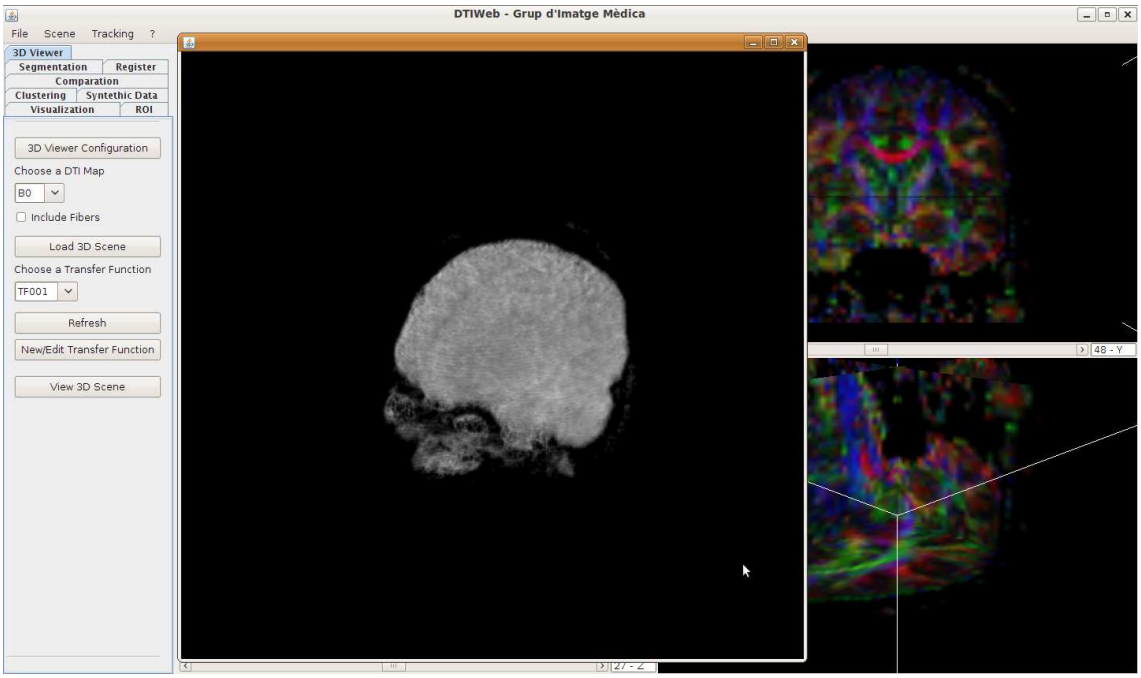

Figura 6.3. Imatge on es el volum de la prova 1 des de davant i la seva esquerre dreta, i amb zoom.

Prova 2

- En aquesta prova visualitza el model amb fibres.
- El Mapa de DTI carregat ha estat: B0.
- La funció de transferència ha estat la mateixa que l'anterior, però amb més transparència.

Rang	Color
[0,15]	(0,0,0,0.0)
[16,25]	(102,102,102,0.05)
[26,35]	(153,153,153,0.05)
[36,45]	(204,204,204,0.05)
[46,55]	(255,255,255,0.05)
[56,300]	(0,0,0,0.0)

- Temps per carregar l'escena: 16 segons 68 centèsimes
- Temps per pintar el volum i visualitzar-lo: 21 segons 14 centèsimes.
- Seguidament mostrem les captures de pantalla a les figures 6.4, 6.5, i 6.6.

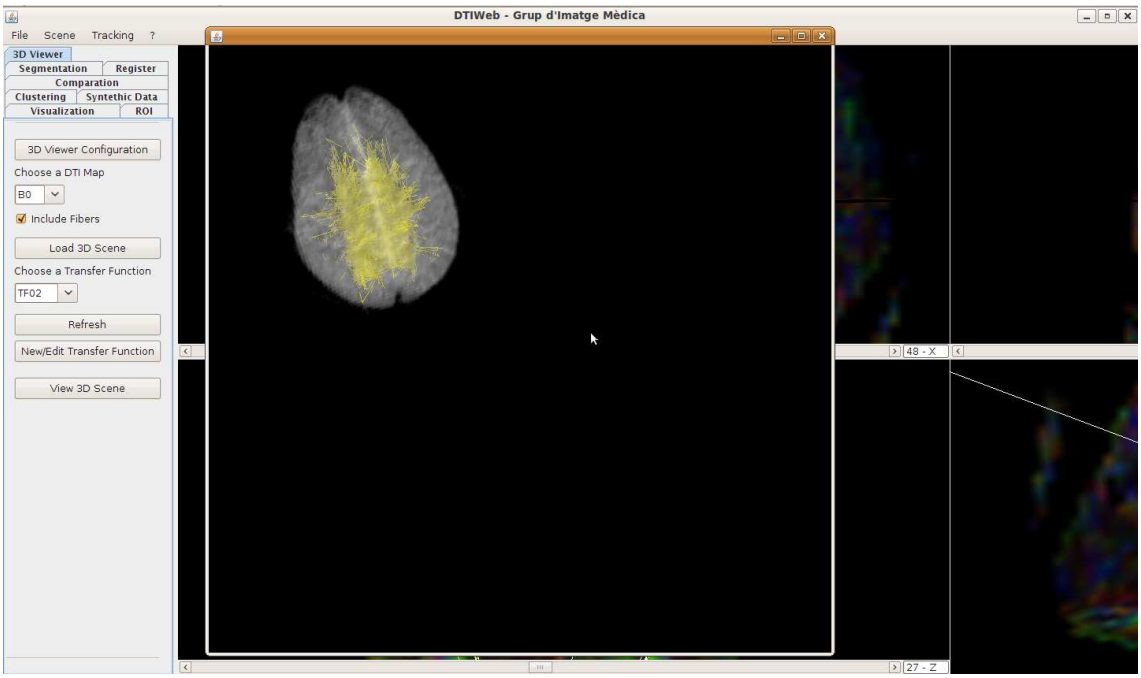

Figura 6.4. Imatge on es el volum de la prova 2, amb una translació

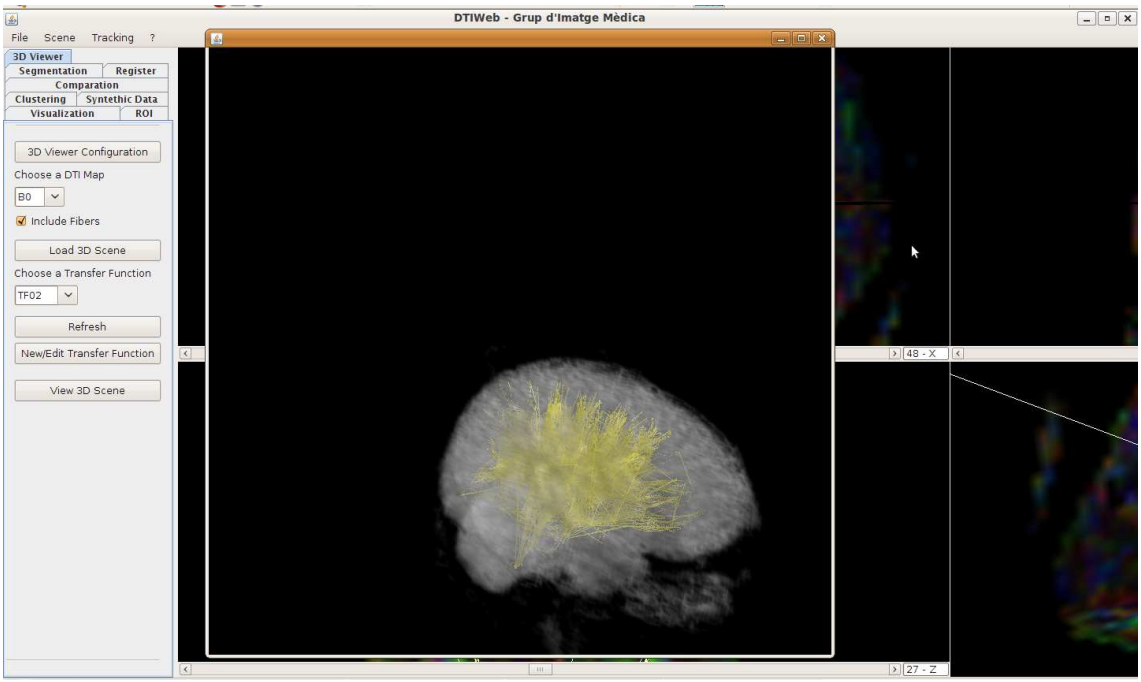

Figura 6.5. Imatge on es el volum de la prova 2, amb una translació, veient la seva dreta, i amb zoom.

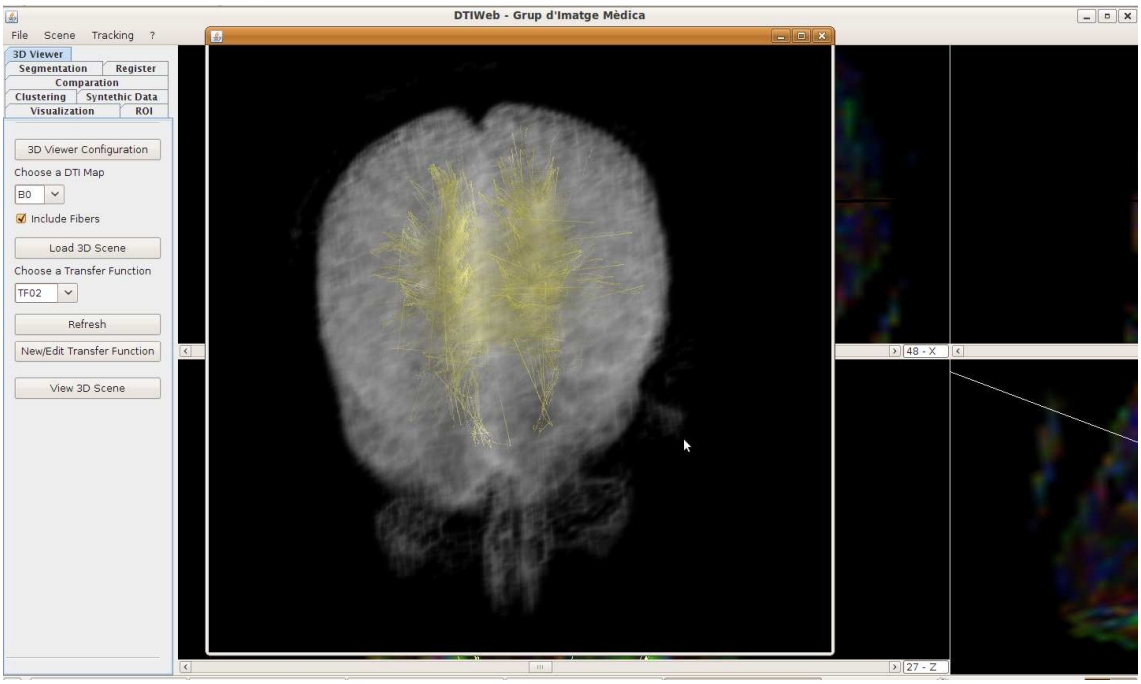

Figura 6.6. Imatge on es el volum de la prova 2 des de davant i dalt, i amb un considerable zoom.

Prova 3

- En aquesta prova visualitza el model sense fibres.
- El Mapa de DTI carregat ha estat: VR.
- La funció de transferència ha estat la mateixa que l'anterior, però amb més transparència.

Rang	Color
[0,20]	(0,0,0,0.0)
[21,30]	(255,51,51,0.2)
[31,40]	(255,102, 102,0.2)
[41,50]	(255,153,153,0.2)
[51,60]	(255,204,204,0.2)
[61,300]	(0,0,0,0.0)

- Temps per carregar l'escena: 14 segons 86 centèsimes
- Temps per pintar el volum i visualitzar-lo: 21 segons 14 centèsimes.
- Seguidament mostrem les captures de pantalla a les figures 6.7.

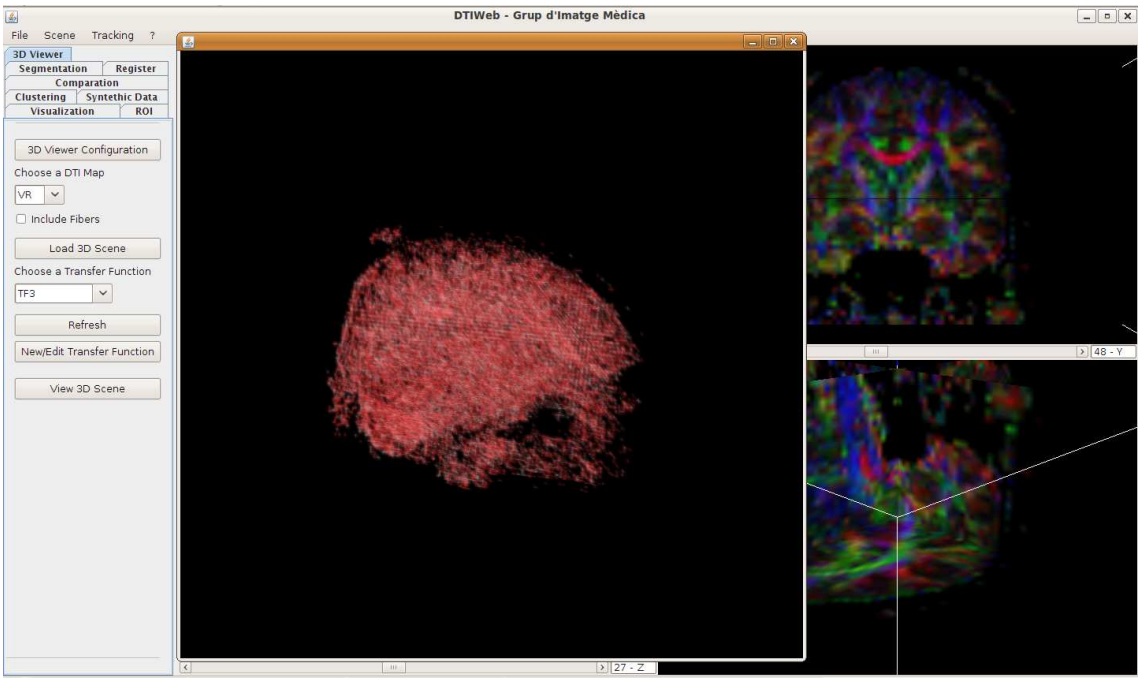

Figura 6.7. Imatge on es el volum de la prova 3, vista des de la dreta del volum.

Prova 4

- En aquesta prova visualitza el model amb fibres.
- El Mapa de DTI carregat ha estat: CL.
- La funció de transferència ha estat la mateixa que l'anterior, però amb més transparència.

Rang	Color
[0,20]	(0,0,0,0.0)
[21,30]	(255, 153, 153,0.3)
[31,50]	(255,204, 204,0.3)
[51,300]	(0,0,0,0.0)

- Temps per carregar l'escena: 17 segons 70 centèsimes
- Temps per pintar el volum i visualitzar-lo: 21 segons 14 centèsimes.
- Seguidament mostrem les captures de pantalla a les figures 6.8.

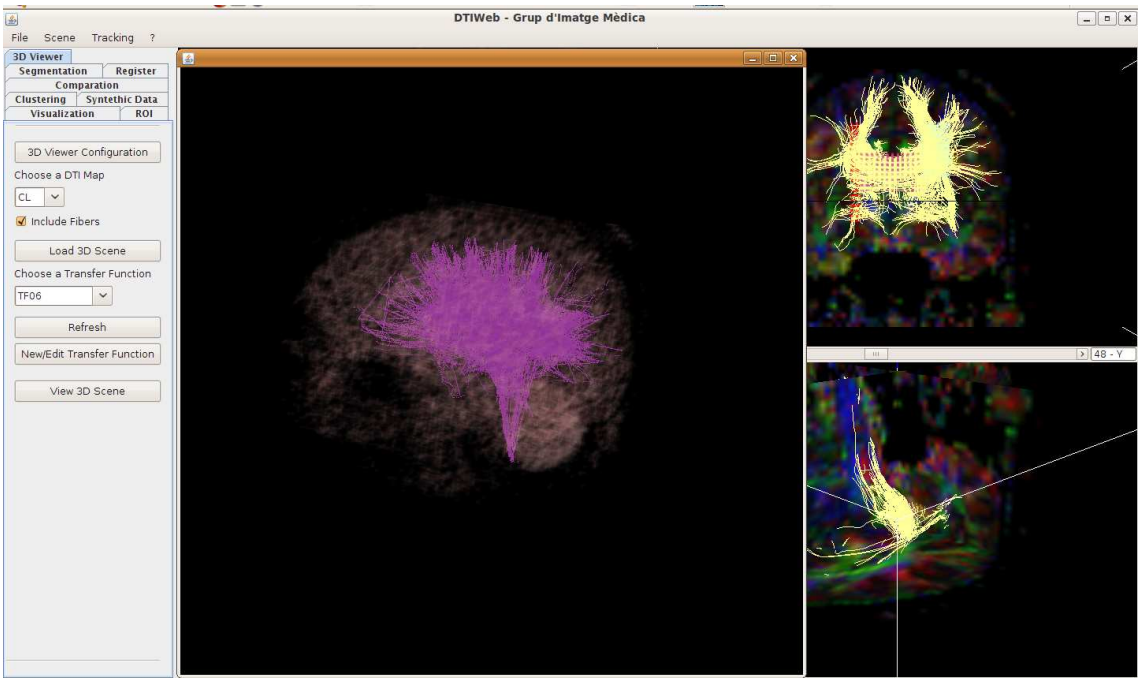

Figura 6.8. Imatge on es el volum de la prova 4, vista des de l'esquerre del volum.

Prova 5

- En aquesta prova visualitza el model únicament fibres.
- El Mapa de DTI carregat ha estat: FA.
- La funció de transferència ha estat la mateixa que l'anterior, però amb més transparència.

Rang	Color
[0,300]	(0,0,0,0.0)

- Temps per carregar l'escena: 18 segons 43 centèsimes
- Temps per pintar el volum i visualitzar-lo: 24 segons 48 centèsimes.
- Seguidament mostrem les captures de pantalla a les figures 6.9.

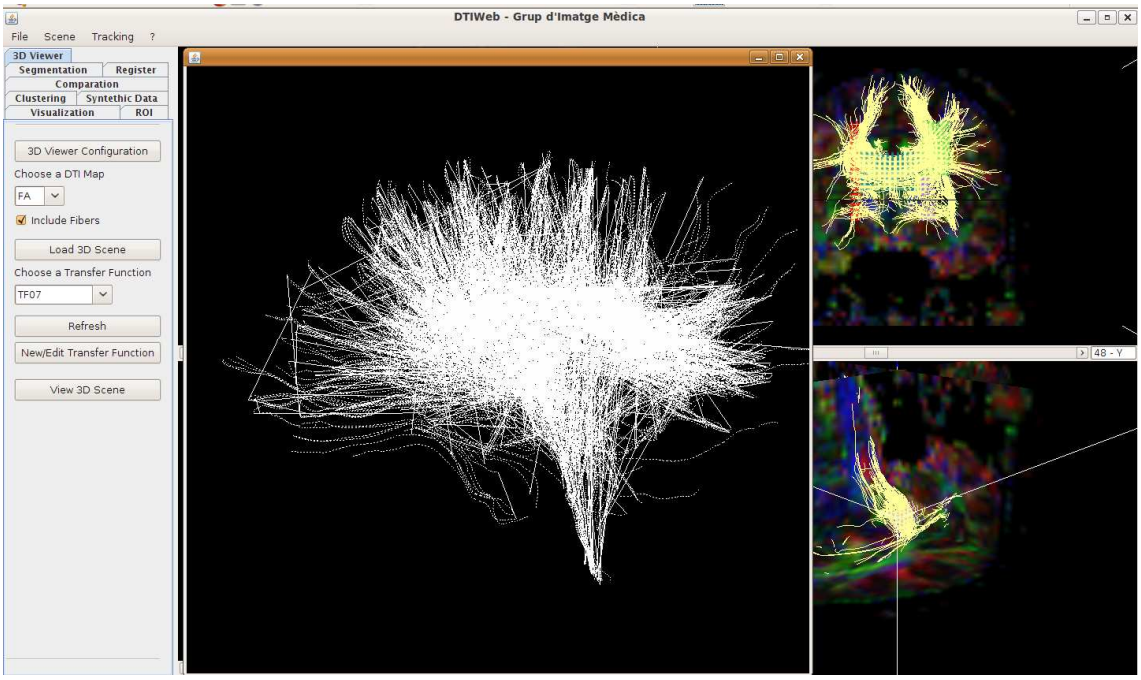

Figura 6.9 Imatge on es el volum de la prova 5, vista des de la dreta del volum.

Prova 6

- Com hem explicat, la nostra aplicació està preparada per detectar errors causats per l'usuari.
- En aquesta prova, hem intentat introduir valors incorrectes, i fer accions que no es puguin efectuar depenén la situació. En cap cas hem aconseguit que saltés un error d'execució. A la figura 6.10, podem veure intentant de visualitzar una escena la qual no estat carregada, on el missatge d'error avisa l'usuari. A la figura 6.11, es pot veure l'usuari intentant entrar una valor incorrecte al valor alpha de la funció de transferència, on l'aplicació també ha avisat l'usuari.

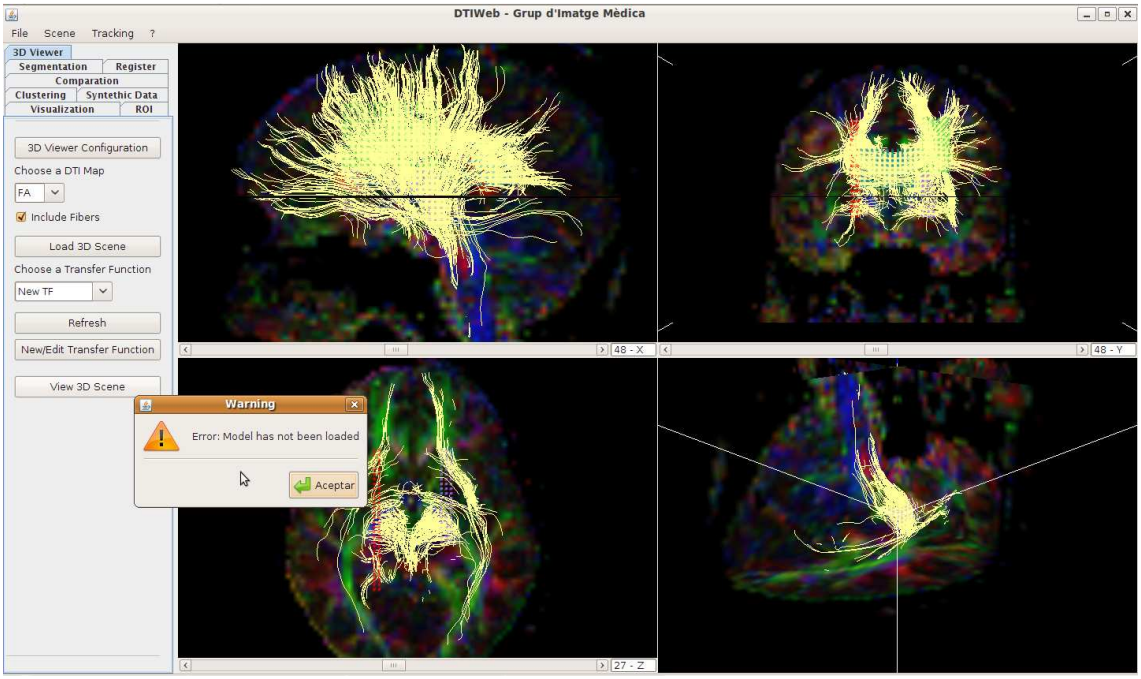

Figura 6.10. Imatge on es veu el missatge d'error per haver volgut visualitzar l'escena abans de ser carregada.

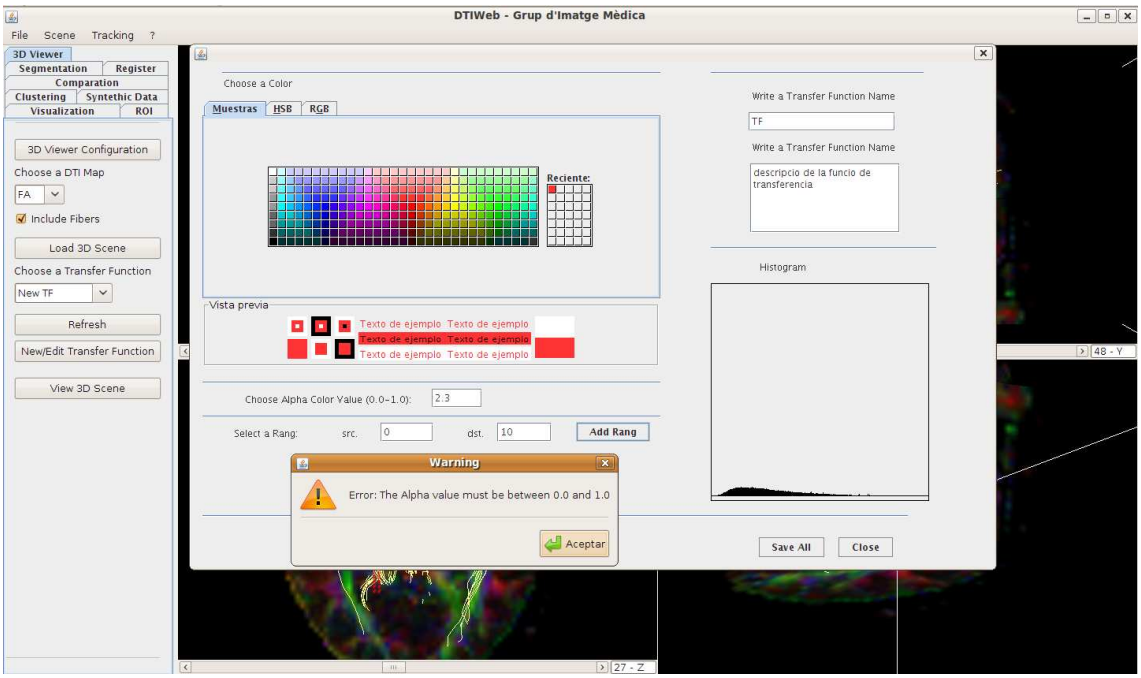

Figura 6.11. Imatge on veu el missatge d'error per introduir un valor incorrecte a l'alpha de la funció de transferència.

6.3 Avaluacions

En aquest apartat ens centrarem en estudiar els valors obtinguts en els resultat de l'apartat anterior.

Si observem el temps, en totes les proves efectuades el temps són molt semblants, ja que en totes les proves el model utilitzat és el mateix, on l'únic que varia és la configuració la qual no consumeix recursos de més en funció de la que sigui. En el cas de la inclusió de fibres, si que es veu un lleuger augment del temps, però no es un augment considerable, això es degut a que els clústers de fibres ja estan calculats a memòria prèviament, i l'únic que es fa és la construcció de la geometria que representa les fibres.

Deixant de banda el temps de càrrega i visualització de l'escena tridimensional, s'ha de tenir en compte el temps d'interacció amb l'usuari un cop el model ha estat carregat, aquest ha de ser petit, ja que en cas contrari, la interacció seria molt lenta i provocaria un mal maneig. A l'ordinador de proves, un cop el model ha estat carregat a memòria, i la visualització ha estat activada, les interaccions de rotació, translació, i zoom, són gairebé instantànies.

Els resultat de les imatges han estat els millors que he pogut aconseguir segons el meu punt de vista. Els que han de fer les proves i avaluar els resultats obtingut són els usuaris finals, els quals ells tenen la informació sobre el tipus de material que correspon a cada valor de les propietats d'un mapa de DTI, i per tant, poder dissenyar Funcions de Transferència més correctes.

Capítol 7. Millores i Treball Futur

Les tècniques i mètodes estudiats durant la memòria han permès assolir tots els objectius que nosaltres havíem marcat. Tot i això som crítics que moltes d'aquestes tècniques podrien ser millorades per obtenir un millor rendiment de l'aplicació. En aquest capítol ens centrarem a recollir de forma molt esquemàtica algunes de les possibles millores i que deixem obertes per a un treball futur.

- *Millorar el sistema d'il·luminació.*
  Dissenyar unes fronteres a l'escena, tal i com hem explicat a l'apartat anterior, ens servirien per poder millorar la il·luminació, ja que podríem estudiar una millor col·locació de les fonts de llum per darrera de la frontera, sabent que aquestes sempre influiran a la il·luminació del volum. D'aquesta manera controlaríem més la il·luminació i aconseguiríem un millor realisme del volum.
- *Mostrar una imatge d'exemple a cada funció de transferència*
  Si a la llista desplegable de funcions de transferència es pogués veure una imatge dels resultats que dona questa sobre un model d'exemple facilitaria molt més la elecció per part de l'especialista quina funció de transferència necessita de més per a visualitzar el volum.
- *Millora de la interfície.*
  Si els usuaris finals no estan satisfet amb la interfície sempre aquesta es pot millorar i canviar totalment, si es mantenen les mateixes crides a les classes de la capa d'aplicació.
- *Suggeriments dels usuaris finals*
  Sempre es poden fer millores suggerides pels usuaris finals.

Capítol 8. Conclusions

Els objectius del projecte eren estudiar la plataforma de processament de dades DTIWeb, estudiar tècniques de visualització de volums i integrar-les al DTIWeb, estudiar i dissenyar tècniques de definició de funcions de transferència, tècniques d'il·luminació per dotar de més realisme als volums, permetre la integració de clústers de fibres, i finalment que tot estigués inclòs al DTIWeb.

Doncs bé, arribat a aquest punt, podem dir que hem complert els objectius.

S'ha implementat una aplicació de reconstrucció de volums en 3D, amb inclusió (opcional) de les clústers de fibres. La majoria de paràmetres del procés de visualització són configurables per l'usuari, com poden ser propietats del material del volum, com les fonts d'il·luminació, i color de les fibres.

A més, l'usuari té un editor de funcions de transferència on aquestes es poden crear i editar quan es vulgui. També esta dotat d'un apartat on poder definir la funció de composició (Blending).

Com a complement l'usuari pot rotar, traslladar i fer zoom al model quan vulgui mitjançant simplement el ratolí.

S'ha intentat fer una interfície gràfica amigable i intuïtiva la qual permeti una senzilla manejabilitat, a més, s'ha dotat l'aplicació d'un sistema de seguretat perquè l'usuari no pugui produir errors d'execució de forma involuntària, de tal forma que si l'usuari fa alguna acció no permesa, l'aplicació no l'efectuarà i avisarà del que està succeint.

El disseny de l'aplicació s'ha centrat en dos mòduls del DTIWeb, la capa de dades i la capa d'interfície, d'aquesta manera s'ha permès la independència entre mòduls, cosa que permet que l'aplicació pugui ser fàcilment modificable i ampliable.

L'execució a l'ordinador de proves ha estat ràpida, tot i que en el moment de càrrega pot durar uns instants de més, la durada és totalment acceptable, a més, un cop carregat i visualitzat el model, l'usuari i pot interactuar amb una resposta instantània.

El fet de que tot el DTIWeb, i inclòs la nostra aplicació, han estat desenvolupats en Java, no ens hem de preocupar en el sistema operatiu a compilar-ho o executar-lo, ja que hi ha total comptabilitat.

Per a desenvolupar l'aplicació, s'han utilitzat eines de domini públic.

Finalment, tot i que s'han assolit els objectius, sempre és possible fer noves millors i ampliacions per dotar el DTIWeb de més possibilitats i un rendiment molt més òptim.

Capítol 9. Webgrafia

- *wikipedia.org*
- *www.sun.com*
- *javadabaduuu.blogspot.com*
- *www.adva.com.ar*
- *books.google.es*
- *javaejemplos.com*
- *www.javamexico.org*
- *www.esi.uclm.es*
- *www.chuidiang.com*
- *www.it.uc3m.es*
- *www.programacion.com*
- *www.itapizaco.edu.mx*
- *www.java2s.com*
- *trevinca.ei.uvigo.es*
- *zarza.usal.es*
- *lineadecodigo.com*