INDEX

1 OBJECTIUS I ANTECEDENTS

1.1 Antecedents

La visualització científica estudia i defineix algorismes i estructures de dades que permeten fer comprensibles conjunts de dades a través d'imatges. Així doncs la visualització consisteix en la transició de representacions computacionals a representacions percentuals, és a dir, passar de dades a imatges. La representació mitjançant imatges ajuda al descobriment d'informació, a la presa de decisions, permet mostrar les dades a personal no qualificat en explicacions i representacions i un llarg etcètera. Un dels principals camps d'aplicació de la visualització és la medicina. Fins fa pocs anys aquest tipus de visualització era possible en màquines específiques de gràfics amb un cost molt elevat. L'abaratiment del hardware especialitzat i la necessitat de nous mitjans d'estudi ha fet que aquesta branca de la informàtica hagi crescut molt ens els últims anys.

En el cas de les aplicacions mèdiques les dades que cal visualitzar provenen de diferents dispositius de captació com poden ser tomografies computeritzades (TAC), ressonàncies magnètiques (RM), etc. Aquests aparells prenen mostres dels pacients corresponents a talls perpendiculars d'alguna zona del seu organisme. Cada un dels talls virtuals que fa l'aparell de captació es representa com una imatge. El conjunt de totes les imatges es representa com un volum anomenat model de vòxels (**veure Figura 1.1**).

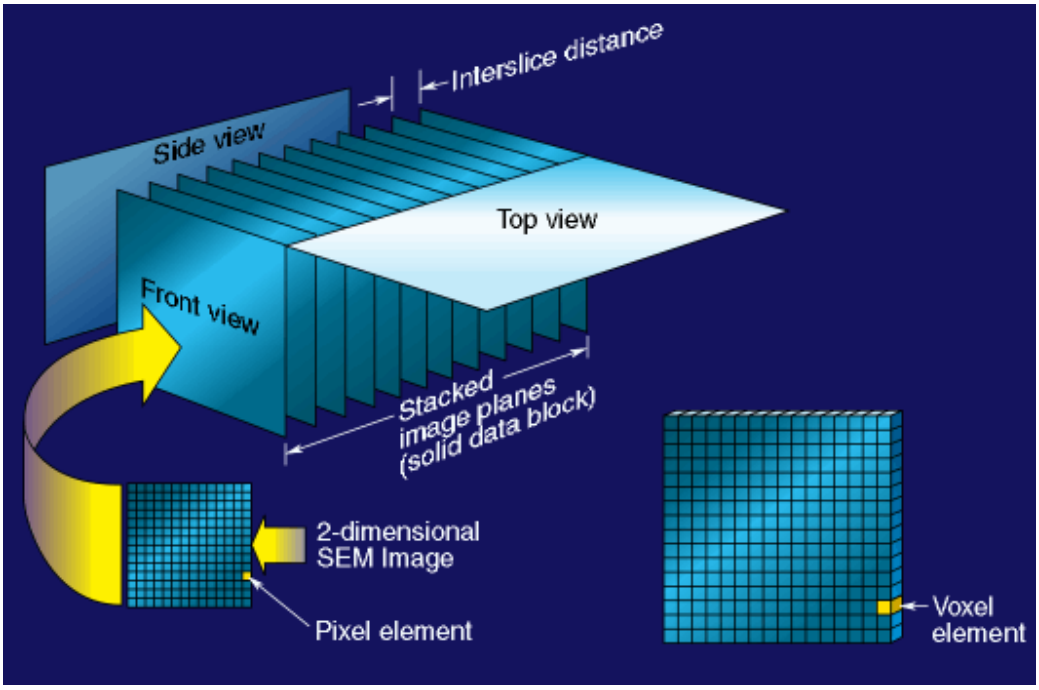


Figura 1.1. *Els aparells de captació prenen dades distribuïdes sobre plans perpendiculars. Aquests plans es representen en un model de volum anomenat model de vòxels.*

Normalment en cada vòxel només s'hi representa un sol valor de propietat provinent del dispositius de captació que s'ha usat per adquirir les dades del pacient. Hi ha situacions en les que pot interessar combinar informació de diferents models. Pensem per exemple en estudis en els que es vol veure com evoluciona una lesió al llarg del temps o bé quan volem combinar informació d'una TAC i una RM. En aquest casos el model de vòxels no representarà un sòl valor de propietat sinó que en representarà tants com models s'hagin combinat. Aquests models es coneixen amb el nom de *models fusionats* i per obtenir-los cal aplicar el que es coneix com a *procés de registre*. Aquest procés el que fa es aplicar una transformació de forma que els dos models que interessa fusionar quedin alineats.

La visualització de models fusionats no és un problema resolt ja que es poden aplicar diferents estratègies per mostrar la seva informació i no està clar quina és la millor.

En aquest projecte ens centrarem en la visualització de models fusionats. Per fer més comprensible la problemàtica d'aquest models a continuació descriurem de forma molt breu com es visualitza un model simple i posteriorment com un model fusionat.

1.1.1 Visualització d'un model simple

Definirem un model simple com el model de vòxels en el que cada vòxel manté informació d'un sol tipus de propietat. Es a dir que les dades que hi ha en el model provenen d'un únic aparell de captació (RM, TAC, etc). Per generar la imatge haurem d'aplicar un algorisme de visualització que el farà es assignar un atribut gràfic a cada valor de propietat del vòxel. Si en el vòxel tenim una propietat que representa os li assignarem un color blanc i opac, si tenim pell podem posar un color rosat i certa transparència. Un cop hem fet aquesta assignació de colors el que farem serà composar els diferents colors per determinar el color del píxel en la pantalla. La composició consisteix en dir com barrejarem els diferents colors que hem posat a cada vòxel. En la Figura mostrem un exemple del diferents passos (**veure Figura 1.2**)

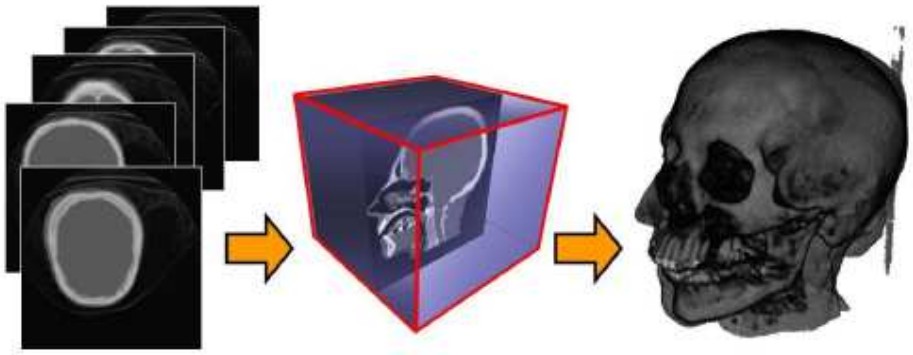


Figura 1.2. *A partir de les imatges en 2D és crea el model 3D i finalment es fa la visualització*

1.1.2 Visualització d'un model fusionat

Tot i que les diferents etapes que cal aplicar per visualitzar un model fusionat són les mateixes que per visualitzar un model simple el procés resulta més complicat. El problema està en que els algorismes de visualització només són capaços de representar un color en cada un dels píxels de la pantalla i en el cas dels models fusionats partim d'un model de vòxels en el que en cada vòxel hi ha més d'un valor de propietat. El problema està en com passar de n valors de propietat a un sol color en la pantalla. Aquest procés de fusió es pot fer en diferents parts del procés de visualització. Entre les diferents opcions que tenim hi ha: la fusió dels valors inicials de propietat en un de sol, l'assignació de colors a les diferents propietats i llavors fusionar els colors, etc.

El nostre objectiu serà estudiar diferents tècniques de visualització de volums fusionats i integrar-les en la plataforma Starviewer.

1.2 La plataforma Starviewer

La plataforma Starviewer és una plataforma de processament i visualització de dades mèdiques desenvolupada pel Laboratori de Gràfics i Imatge de la Universitat de Girona i l'Institut de Diagnòstic per la Imatge de l'hospital Josep Trueta. Actualment la plataforma pot visualitzar models simples aplicant diferents mètodes de visualització però no suporta la visualització de models fusionats.

1.3 Objectiu del Projecte

L'objectiu principal d'aquest projecte és estudiar, implementar i comparar diferents tècniques de visualització de models fusionats. Aquestes tècniques s'integraran en la plataforma de visualització i processament de dades mèdiques STARVIEWER.

Per assolir aquest objectiu es defineixen els següents objectius específics:

1- Estudiar els algoritmes de visualització de models simples i analitzar els diferents paràmetres a tenir en compte. Es partirà dels algorismes bàsics i es consideraran també tècniques avançades que integrin il·luminació global. Es seleccionarà una tècnica de visualització com a tècnica bàsica a aplicar.

2- Ampliació de la tècnica de visualització bàsica seleccionada per tal de suportar els models fusionats. S'implementaran totes les variants de fusió de la informació en les diferents etapes del procés de visualització.

3- S'avaluaran i compararan tots els mètodes implementats per poder determinar quin ofereix les millors visualitzacions.

Tots els mòduls implementats s'integraran a la plataforma STARVIEWER.

1.4 Eines

L'Starviewer està implementat fent servir les llibreries gràfiques ITK's VTK's i QT's que més endavant explicarem el seu funcionament. A més es treballarà amb el mateix entorn de programació que fan servir els desenvolupadors de la plataforma. Aquest programa és el KDEVELOP, un API de codi lliure que funciona sota el Sistema Operatiu Linux. Com a complement s'utilitzarà l'editor de formularis Qt QT4developer.

1.5 Metodologia

En aquest projecte és fa servir una programació Orientada a Objectes(OO) ja que l'estructura de les llibreries fetes servir així ho exigeix. Una metodologia que s'ajusta a aquest paradigma de programació és eXtreme Programming (XP) ja que dona molta més flexibilitat que altres metodologies alhora de fer modificacions de disseny durant la implementació.

La idea bàsica d'aquesta metodologia és la de dissenyar i implementar alhora. La millor manera de conèixer el funcionament de les llibreries és fent proves que necessiten parts implementades d'una part de l'aplicació. Així doncs dissenyarem i implementarem parts del projecte segons les nostres necessitats.

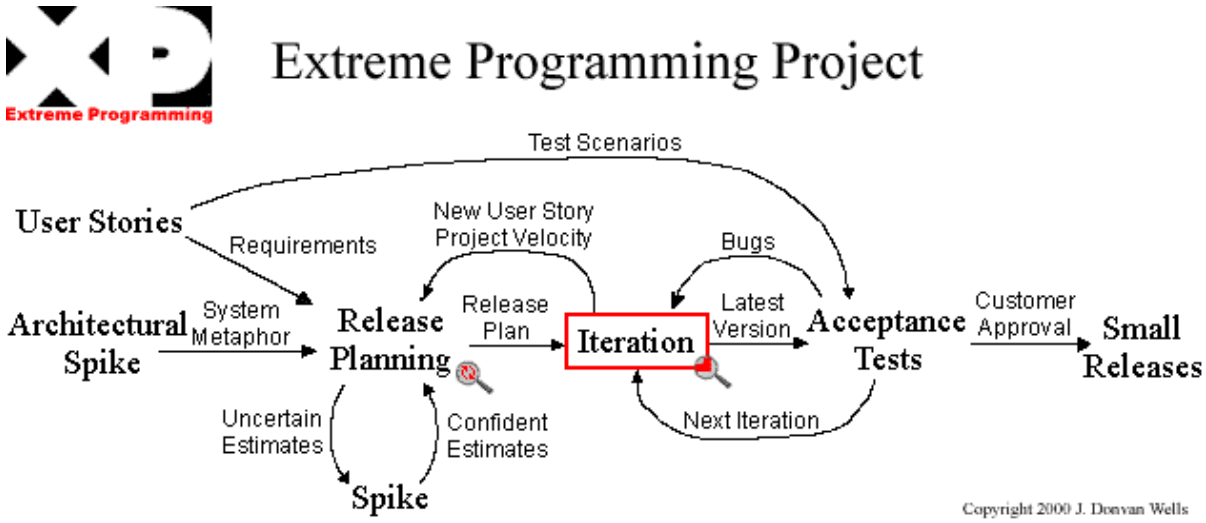


Figura 1.3. *Esquema del procés de programació fent servir eXtrem Programming.*

En aquest diagrama podem veure una sinterització dels objectius del eXtrem Programming.

Primer tenim el problema a resoldre(Architectural Spike) i per aconseguir-ho farem una planificació del problema(Release Plannig). Realitzem l'esquema i mitjançant iteracions intentem arribar als testos d'aprovació per comprovar que el que hem realitzat és correcte.

En realitzar aquest tests podem trobar diferents bugs que solucionarem tornant a

l'apartat d'iteració. Si els tests són satisfactoris, primer comprovo si hi ha més iteracions a fer, és a dir, més parts a desenvolupar segons la planificació inicial. Sinó en tinc més, puc pública petites parts del meu projecte, passant prèviament per l'aprovació del client o supervisor.

Les característiques més importants de **l'eXtrem Programming** són:

1. Una metodologia àgil ideal per projectes de curta durada amb requeriments poc clars o canviants. En el Projecte Fi de Carrera s'estableixen uns objectius que a vegades no es poden assolir o si més no varien conforme es va investigant en la matèria.

2. Es busca oferir moltes entregues del sistema, on cada una d'aquestes afegeix funcionalitats noves en un espai de temps reduït. És una bona manera de que el tutor del projecte faci un seguiment del treball realitzat pel projectista.

3. L'usuari va veient més clarament i per parts si acompleix els objectius marcats en un principi. En aquest cas l'usuari podria ser el tutor i com en el segon punt pot anar veien com evoluciona el projecte.

4. Fer servir estàndards de codificació. A l'Starviewer, com és una plataforma on hi treballa i hi ha treballant moltes persones s'intenta fer servir estàndards per tal de fer-ho més fàcil a tothom.

5. Es planifica a curt termini. El projecte Fi de Carrera és una assignatura més i per tant es planifica com un projecte de curta durada. Després segons quin projecte es desenvolupi això pot variar.

6. El resultat és el més important, és a dir, s'han d'acomplir els objectius marcats i res més.

7. KISS = Keep it simple stupid!! La solució més fàcil que acompleixi els requeriments és la més adequada. Si queda temps s'anirà millorant. En aquest projecte s'aplica aquest punt aprofitant mòduls i funcionalitats desenvolupades en altres projectes. A més els projectes realitzats no passen directament a ser part de la plataforma cosa que fa que es pugui optimitzar algunes parts.

8. Recomana programar en parelles donat que fa el codi de més qualitat ja que el que una persona no sap fer, potser l'altre li soluciona i viceversa. Lògicament en aquest projecte no s'aplica aquest punt, tot i que demanar ajuda a algun company desenvolupador de la plataforma et pot ajudar o donar una altra opinió.

9. Corregir tots els bugs o errors detectats abans d'implementar una nova funcionalitat. En aquest cas un cop s'ha aconseguit assolir algun dels objectius s'ha optat per començar el següent i deixar per últim la correcció d'errors que en la majoria dels casos són de fàcil solució.

10. Refactorització del codi, que no és més que reescriure parts del codi per tal de millorar el seu enteniment o una futura adaptació. És vital no afegir cap error en fer la refactorització. En el projecte, un cop assolits els objectius, s'han solucionat els bugs coneguts i s'han optimitzat parts.

11. Tots els programadors puguin tenir accés a tots els mòduls de l'aplicació per tal de corregir o afegir funcionalitats sempre comunicant els canvis als responsables del mòdul afectat. No s'ha aplicat aquest aspecte ja que al ser una extensió d'una plataforma molt gran no és recomanable modificar objectes ja validats per professionals. Tot i així s'ha treballat sota còpies locals amb accés únic de lectura al codi per no fer malbé cap part important de la plataforma.

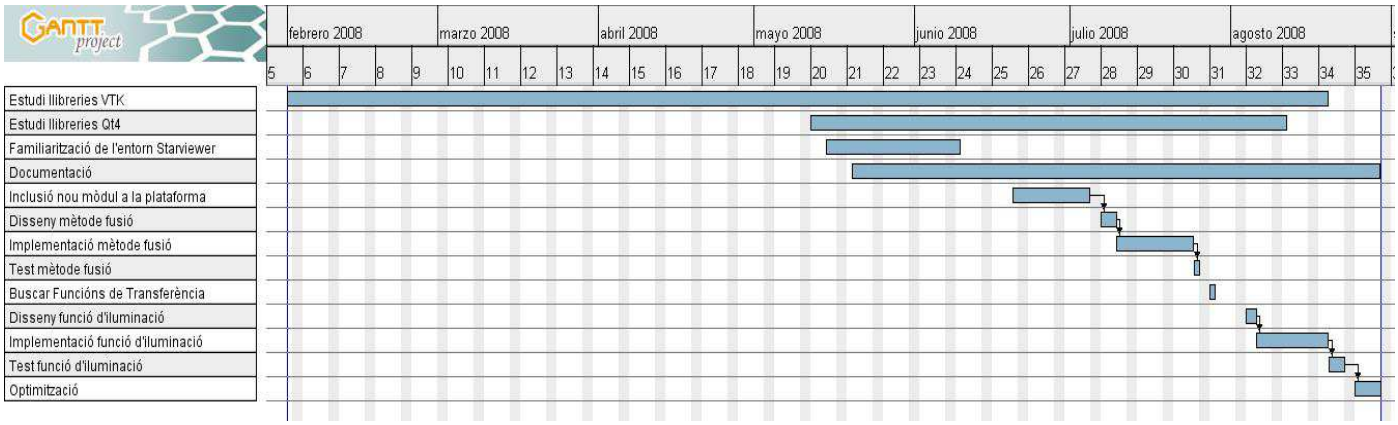
12. Com s'aplica en projectes de curta durada, aspectes com el rendiment, l'eficiència i la seguretat no són crítics tot i que sempre són importants. Es veritat que no són aspectes crítics ja que s'han desenvolupat funcionalitats que més endavant passaran uns testos més estrictes per ser afegits a la plataforma, però en tot moment s'ha intentat programar de la millor manera possiible.

13. Per altra banda, S'ha fet servir l'estàndard UML(Unified Modeling Language) per representar els diagrames de l'aplicació en la fase d'anàlisi i disseny. Tots els esquemes representats a la part de disseny d'aquesta memòria fan servir aquest estàndard i han sigut elaborats amb les pautes pertinents.

1.6 PLANIFICACIÓ

L'estructuració del treball realitzat en el projecte per tal d'assolir els objectius fixats es divideix en dues parts:

- **Fase1:** En aquesta fase es realitzarà l'estudi previ que ens permeti:
  - Assolir el nivell necessari de coneixements per poder desenvolupar el disseny i la implementació, és a dir, l'estudi teòric previ al disseny.
  - En aquesta fase entra l'estudi de les llibreries fetes servir
  - Adaptació a l'entorn STARVIEWER.
- **Fase 2:** Aquesta fase engloba tot el disseny, implementació i integració de tots els mòduls i funcionalitats dins la plataforma, tot assolint els objectius marcats.



El document que presentem l'hem estructurat seguit aquests fases.

1.7 Temporització

Nom	Inici	Fi
Estudi llibreries VTK	1/02/08	20/08/08
Estudi llibreries Qt4	12/05/08	12/08/08
Familiarització de l'entorn Starviewer	15/05/08	10/06/08
Documentació	20/05/08	30/08/08
Inclusió nou mòdul a la plataforma	20/06/08	5/07/08
Disseny mètode fusió	7/07/08	10/07/08
Implementació mètode fusió	10/07/08	25/07/08

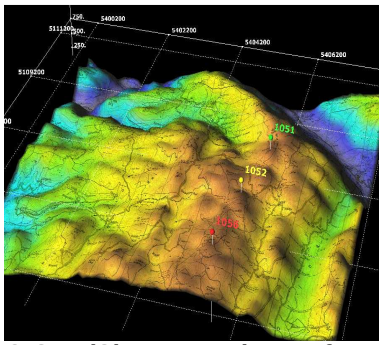
Nom	Inici	Fi
Test mètode fusió	25/07/08	26/07/08
Buscar Funcións de Transferència	28/07/08	29/07/08
Disseny funció d'iluminació	4/08/08	6/08/08
Implementació funció d'iluminació	6/08/08	20/08/08

2 FONAMENTS TEÒRICS

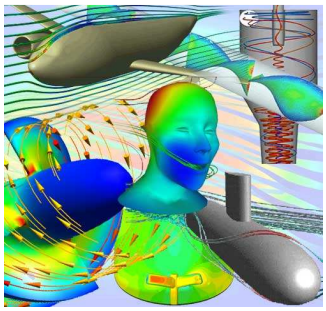
2.1 Introducció

La visualització científica és una de les moltes aplicacions de la informàtica. L'objectiu és definir els processos i algoritmes necessaris per poder representar gràficament grans volums d'informació per tal de fer-la més intel·ligible. Trobem diferents especialitats científiques on la visualització és una eina molt utilitzada, com podria ser la geologia, la cartografia, paleontologia, i sobretot la medicina (veure **Figura 2.1**).

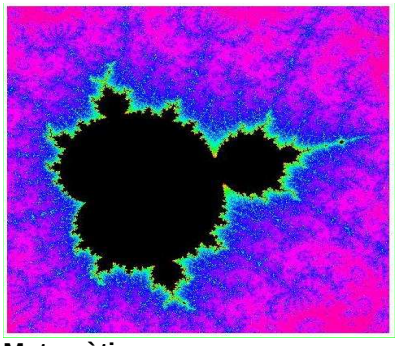
En aquest projecte ens centrarem en la visualització de dades captades per dispositius mèdics com podrien ser els TACs o les RMs.. Els continus avenços de les tecnologies implicades en la visualització mèdica com podrien ser, els aparells de mesura, estudi i demés, i les pròpies màquines informàtiques fan que aquesta area de la informàtica estigui en continua evolució apareixen tècniques cada cop més realistes i òptimes. Darrera aquestes tècniques hi ha el que es coneix com a procés de visualització.



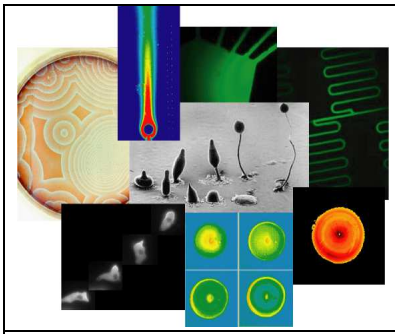
GIS (Sistema de Informació Geogràfica):
Representació virtual de dades geogràfiques, estudis cartogràfics.



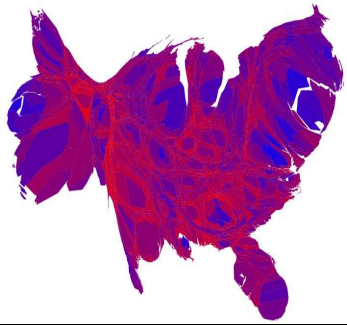
Dinàmica de sistemes de fluids:
S'estudia el comportament dels fluids en diferents situacions pel disseny de peces mecàniques**.**



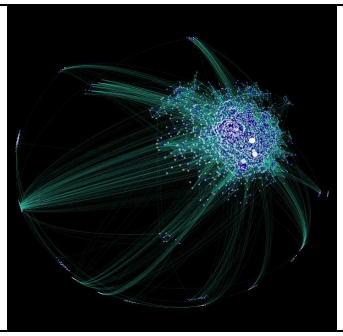
Matemàtiques:
Visualització de models matemàtics per explicar comportament de funcions o per presentar resultats.



Física teòrica i experimental:
S'observen els resultats d'estudis físics mitjançant reproduccions gràfiques.



Economia i estadística:
Agrupació de grans quantitats de dades estadístiques



Data mining:
Estudi del flux de dades d'una estructura interconnectada

Figura 2.1. *Visualització de dades de diferents àrees científiques*

2.2 EL PROCÉS DE VISUALITZACIÓ

Aquest procés és el que fa la transició des de l'obtenció de les dades a interpretar fins la representació gràfica per pantalla. Aquest procés està format per les 4 etapes que expliquem a continuació:

2.2.1 Adquisició de dades

La primera etapa consisteix en l'adquisició de les dades. Aquestes poden provenir d'un model real o d'una simulació.

En imatge mèdica normalment obtenim les dades d'un pacient a partir d'un dispositiu escanejador o captador de dades. Aquest dispositiu, depenent de la seva funcionalitat, serà capaç d'enregistrar unes propietats o unes altres, amb més o menys resolució.(Veure dispositius de captació a la ***figura 2.1***) La mesura que es pren en cada punt del pacient és un valor numèric anomenat valor de propietat. Aquest valor es pot representar com un valor d'intensitat en una imatge. Cada dispositiu de captació tradueix el teixit mesurat a un valor de propietat, però no sempre aquest teixit tindrà el mateix valor. Depèn de moltes variables com el tipus de dispositiu, les condicions de l'entorn, etc.

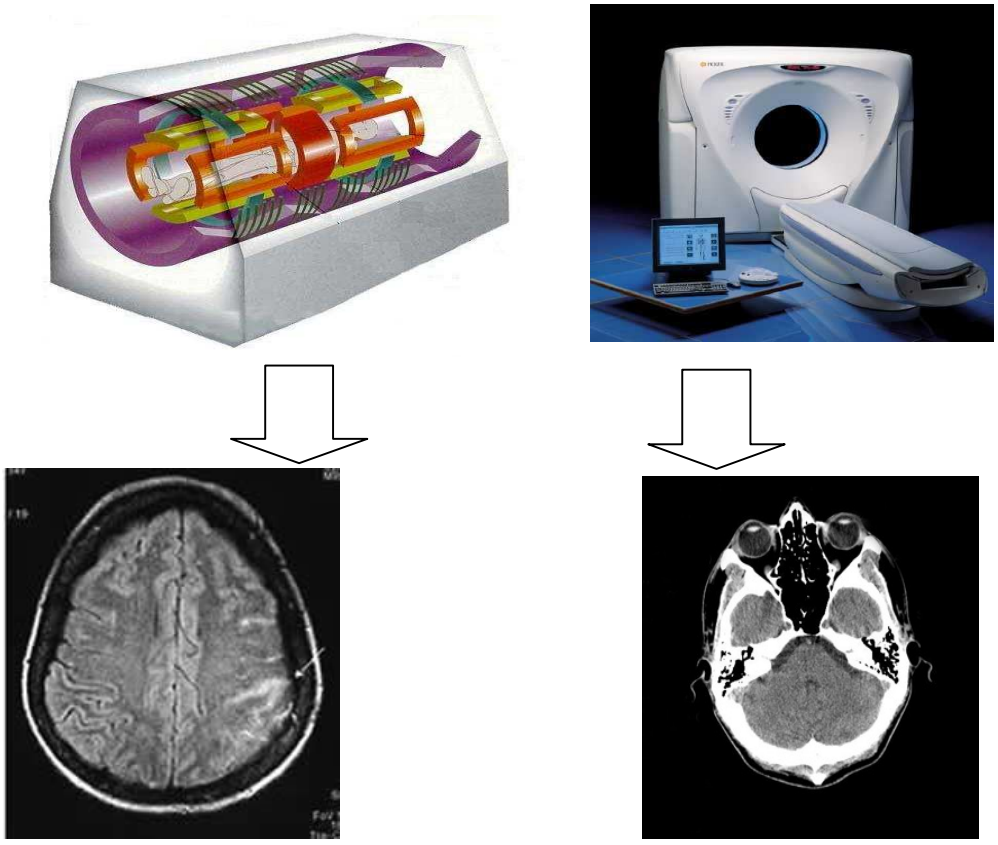


Figura.2.2: *Imatges adquirides per dispositius de captació.*

2.2.2 Definició del model

Les imatges que hem obtingut a l'etapa anterior s'han de representar d'alguna manera per tal de facilitar-ne el tractament computacional. En imatge mèdica el més usual és fer servir el model de vòxels. Aquest model representa la informació que s'obté en l'etapa anterior com un cub de cubs de mida idèntica on a cada un hi guardem els valors de propietat mesurats pel dispositiu de captació. Aquests cubs iguals s'anomenen vòxels i el model obtingut model de vòxels. Un vòxel podria entendre's com un píxel de 3 dimensions.

Les dades captades anteriorment es guarden distribuïdes en plans paral·lels ordenats. Per generar el nostre model, es van apilant aquest plans per tal d'aconseguir una quadrícula tridimensional. ***Figura 2.3***

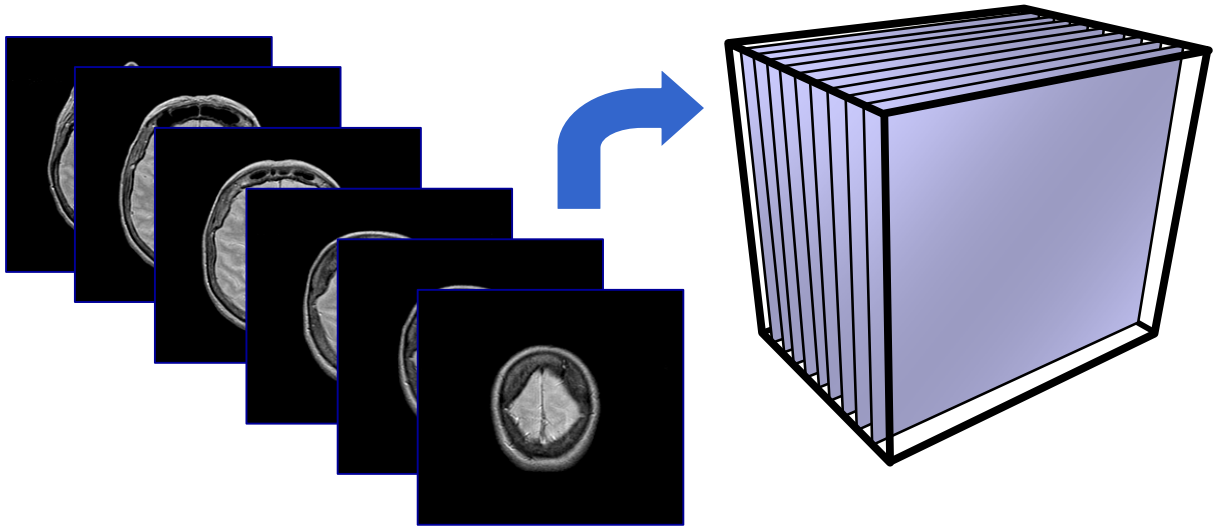


Figura 2.3. *Amb imatges de diferents profunditats d'un pacient es reconstrueix el model 3D de vòxels.*

2.2.3 Reconstrucció

En aquesta fase s'elimina del model de vòxels tota la informació suplementària que no necessitem per estudiar el nostre pacient. S'apliquen diferents filtres que no son més que algoritmes per tal d'escollir la informació realment important.

Després de fer el filtratge, es selecciona la part del models de vòxels que voldrem representar en la pantalla. Aquesta procés es coneix com a reconstrucció. L'algorisme de reconstrucció més conegut i més aplicat en l'entorn mèdic és el marching cubes.

Marching Cubes

Aquest algoritme crea models triangulars de superfícies a partir d'un conjunt de dades 3D.Es crea una taula de definició de la topologia triangular fent servir una estratègia de divideix i venç. Després es processen les dades per capes i es calculen els vèrtex fent servir interpolació lineal. Finalment es troba el gradient de les dades originals, es normalitza i es fa servir de base per aplicar algoritmes d'iluminació al model.

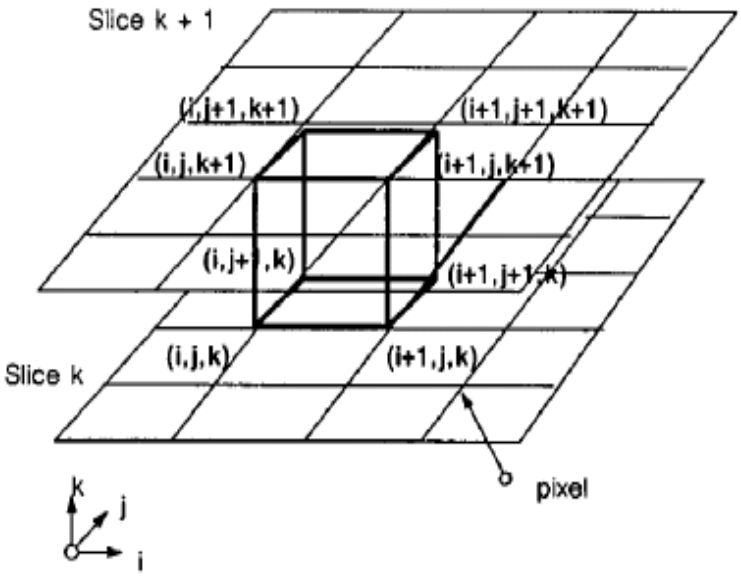


Figura 2.4: *cub lògic entre 2 capes.*

En la ***figura 2.4*** es pot observar com s'estudien els vòxels dins el model. Es crea un cub lògic entre dues capes contigües. Tot seguit es comparen les buit arestes amb un llindar escollit. Si com a mínim un vèrtex del cub està per sota i un altre per sobre el llindar, el cub formarà part de la isosuperfície. D'altra manera no s'inclourà.

Com que es determina dins del cub per on passarà la superfície es poden generar triangles unint ells punts d'intersecció. Si s'uneixen tots els triangles del cub s'obtindrà la superfície buscada.

Podem trobar 256 maneres diferents d'intersecció de la superfície. Això comporta molts càlculs per cada vòxel i per tant s'han de buscar solucions més eficients. Per complement es redueixen a 128 possibilitats i si rotant el cub es poden reduir fins a 15 possibilitats d'intersecció com veiem a la ***figura 2.5.***

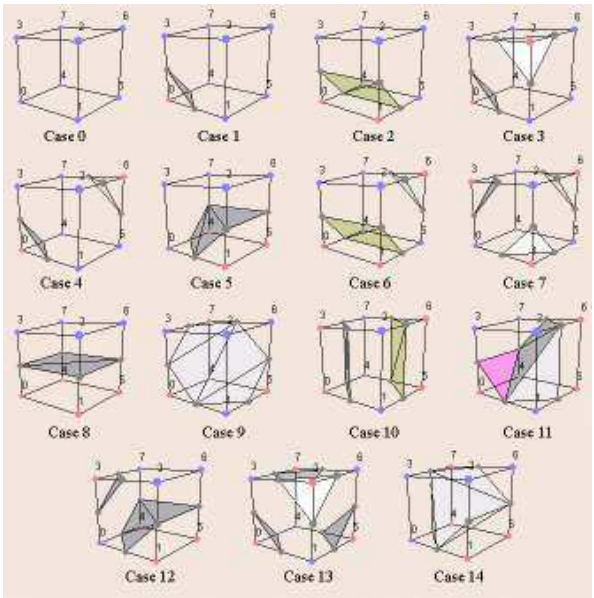


Figura 2.5.: *15 maneres possibles d'intersecció.*

Si es visualitza l'isosuperfície, depenent del valor del llindar i de la mida i nombre de vòxels, es pot veure la imatge amb un efecte aliasing(dents de serra), és a dir, es veu una imatge generada a partir de cubs(*Figura 2.6*).

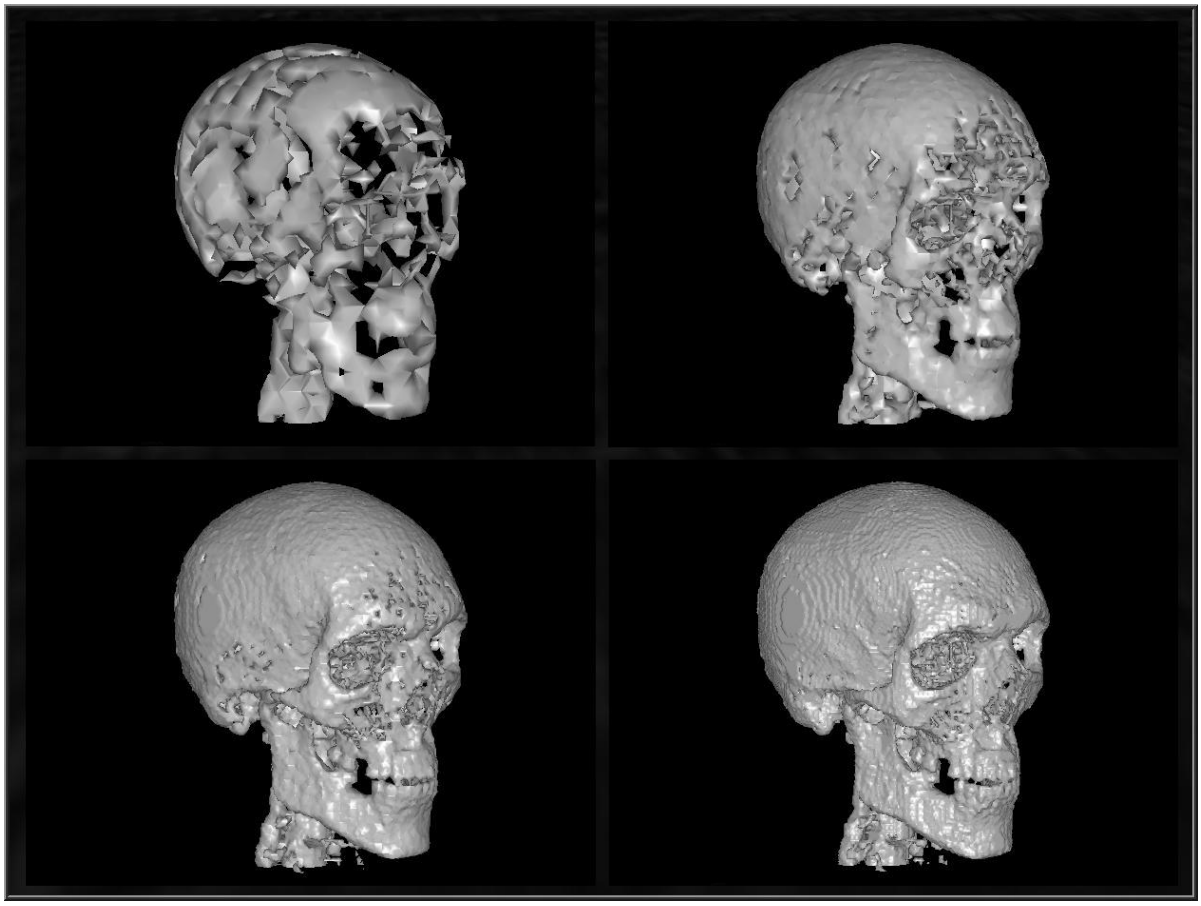


Figura 2.6: *4 visualitzacions d'isosuperfícies amb diferents número i mida de vòxels.*

Aquest algoritme té la avantatja de produir imatges amb bona qualitat i resolució amb poc cost computacional. A més fa que la visualització sigui molt més senzilla. El problema pot ser quan treballem amb models amb molta informació ja que poden sorgir alguns forats degut a informació ambigua alhora d'agrupar les interseccions(***Figura 2.7***):

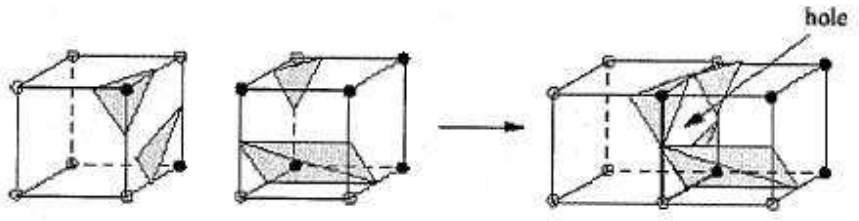


Figura2.7: *En reconstruir el model es poden trobar situacions on la fusió no sigui del tot clara i es generin forats, és a dir, zones buides dins d'una superfície.*

2.2.4 Visualització

En el cas de que partim d'un model reconstruït en aquesta fase el pintarem en la pantalla. Normalment els models reconstruïts es representen com a malles de triangles, per tant en aquest cas la visualització consisteix en pintar aquests triangles.

En el cas de que no partim d'un model reconstruït, és a dir que volem pintar tot el model de vòxels aplicarem el que es coneixen com algorismes de visualització directa de volums.

En aquest projecte s'aplicaran algoritmes de visualització directa de volums.

Poden considerar-se dos grups d'algorismes de visualització directa de volums:

- Algoritmes d'ordre imatge o tipus Ray Casting:

  Per obtenir la imatge es llença un raig per cada pixel 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 d'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.

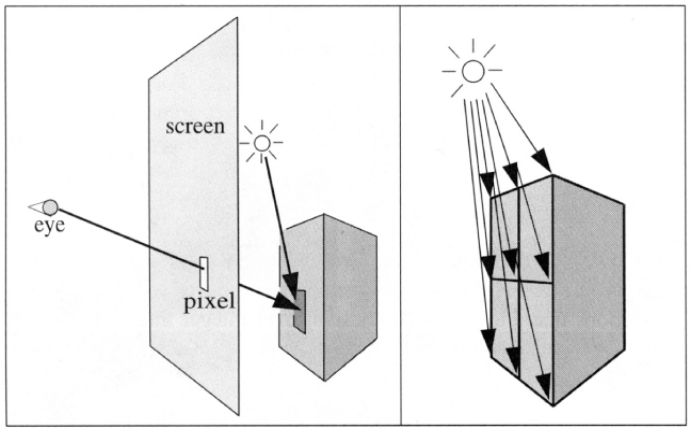


Figura 2.8. Es té en conta l'ull de l'observador i el focus de llum

- 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 Ray Casting.

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.

2.2.5 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:

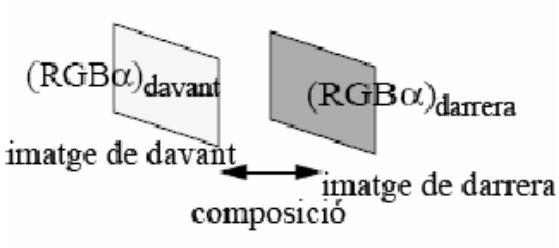
2.2.6 Composició de colors

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 intersectant 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:

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



- 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.

2.2.7 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 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 2.9**).

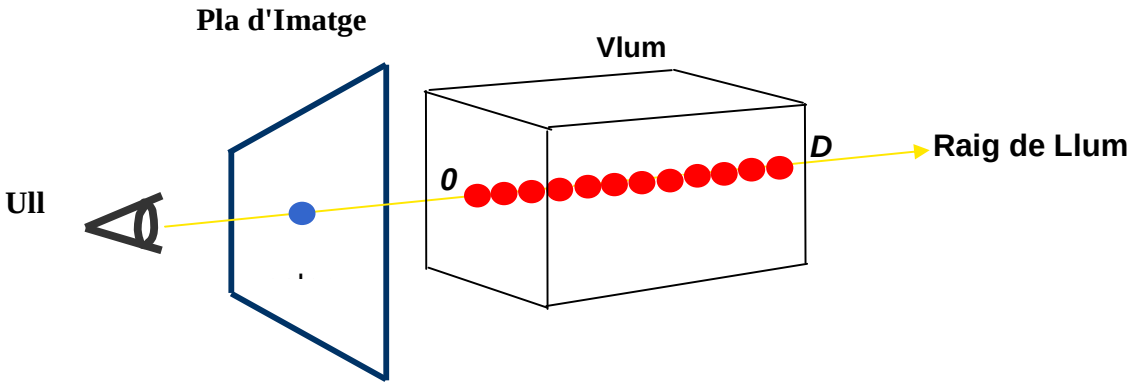
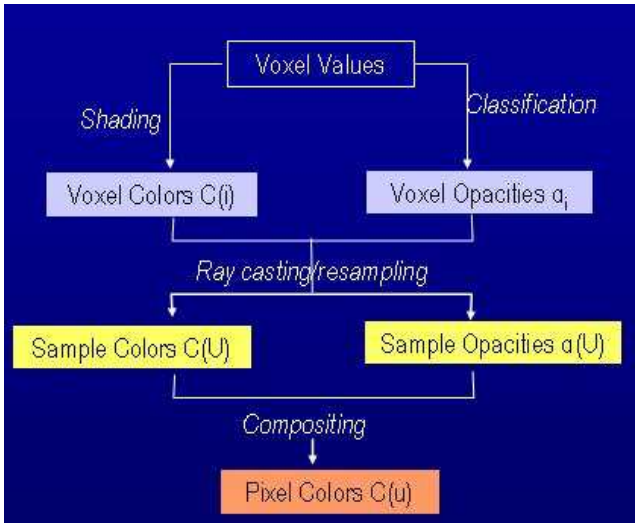


Figura 2.9: *El raig entra pel punt O i a cada col·lisió dins el volum calcula el color i la transparència. Un cop calculades totes les col·lisions es computarà el color final D que serà el que es mostra al pla d'imatge(pantalla).*

De fet el terme col·lisió no és del tot correcte ja que continua travessant l'objecte. En podríem dir interacció. A cada llançament s'observa el color i l'opacitat del volum travessat, que després d'una composició ens donarà el color final del píxel de la imatge que es mostrarà per pantalla. El color i l'opacitat és guarden en un objecte RGBx on RGB fa referència al color i x a l'opacitat. El color final que apareix en la pantalla depèn de la funció de transferència assignada en el procés de computació.



La composició dels colors consisteix en simular el procés de visualització humana. De tots els colors i opacitats enregistrats d'un dels raigs llençats a través del volum fa una suma balancejada. No es ben ve el procés real, ja que en realitat és una integral des de 0 fins INFINIT , és a dir, amb infinites mostres. En el Volume Rendering és fa una suma discreta tal i com explicàvem en la secció 2.4

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

2.3 MODELS FUSIONATS

Una vegada explicat el procés de visualització en centrarem en la visualització dels models fusionats. Per començar explicarem el procés de registre, és a dir el procés que cal dur a terme per poder tenir un model fusionat. Hem de dir que en la projecte considerarem que ja ens donen el model fusionat i per tant no ens haurem de preocupar de fer el registre.

2.3.1 Registre

Els diferents aparells de captació de dades del pacient sovint ens dona molta informació complementària la qual pot ser molt interessant pels especialistes que hagin de fer un diagnòstic. Per poder comparar i tractar aquesta informació de forma conjunta es poden aplicar estratègies que permetin representar tot la informació en un sol model de volum. Evidentment també es podria considerar diferents models de volum tractar-los de forma independent però sempre resulta millor considerar-los com un de sòl. Tot això s'ha d'enregistrar de tal manera que es puguin tractar la informació adequadament. El principal problema que tenim per crear un únic model de representació és el fet de cada aparell recull les dades usant un protocol diferent i per tant el models poden tenir diferents resolucions, diferents orígens de coordenades, o correspondre a zones que no són totalment iguals. També es pot donar el cas de que el pacient no estigui col·locat en la mateixa posició.

El registre és el procés d'unificació de la informació obtinguda de diferents models d'imatge en un esquema de representació, mantenint tota la informació dels models inicials.

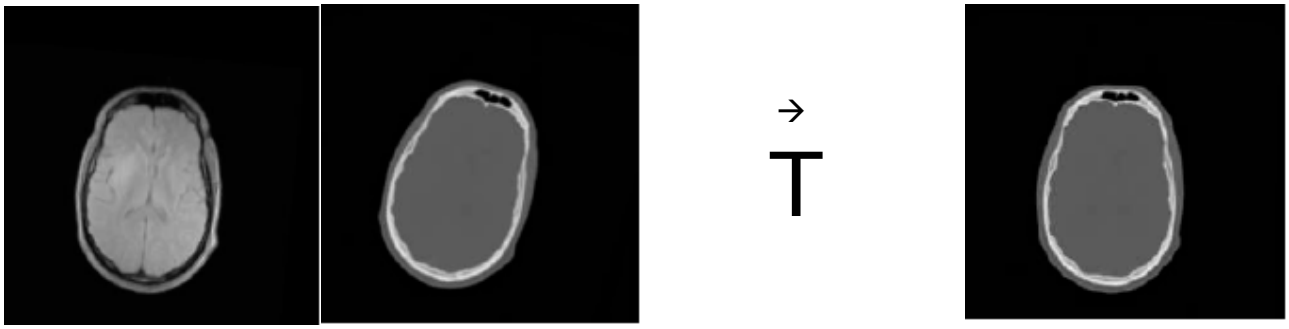


***Figura 2.9**. Amb la funció de transformació s'aconsegueix alinear els models.*

El procés de registre podem dividir-ho en dues fases:

- **Fase d'alineament:** Com diu el nom, s'intenten alinear els models de dades per tal d'obtenir un únic model de dades anomenat model registrat.
  Per aconseguir aquest model es fan servir diferents operacions d'escalat, rotació o translació per tal alinear-los.

- **Fase de fusió:** En aquesta fase es busca obtenir, de les dades que ens proporciona el model registrat, una representació gràfica òptima que ens doni el màxim d'informació possible per tal de facilitar el treball als professionals.

Si tenim dos models no alineats el que hem de fer es agafar un de referència, normalment serà el model de menys definició degut a que podem discretitzar informació però no en podem crear de nova sense tornar a fer un escaneig, i a l'altre li aplicarem una funció de transformació per tal de solapar perfectament els dos models.

El model fusionat o registrat es representa amb un model de vòxels. En cada vòxel es guarden les diferents propietats que ens especifiquen els models que s'han fusionat. A diferència d'un model simple, que només guardava 1 propietat en cada vòxel, ara en guardarem n, tants com models fusionem

2.3.2 Visualització models fusionats:

Per visualitzar models fusionats primer hem d'entendre els passos necessaris per fer la visualització (veure apartats 2.3 i 2.4). Aquests passos els hem representat en la figura 2.8:

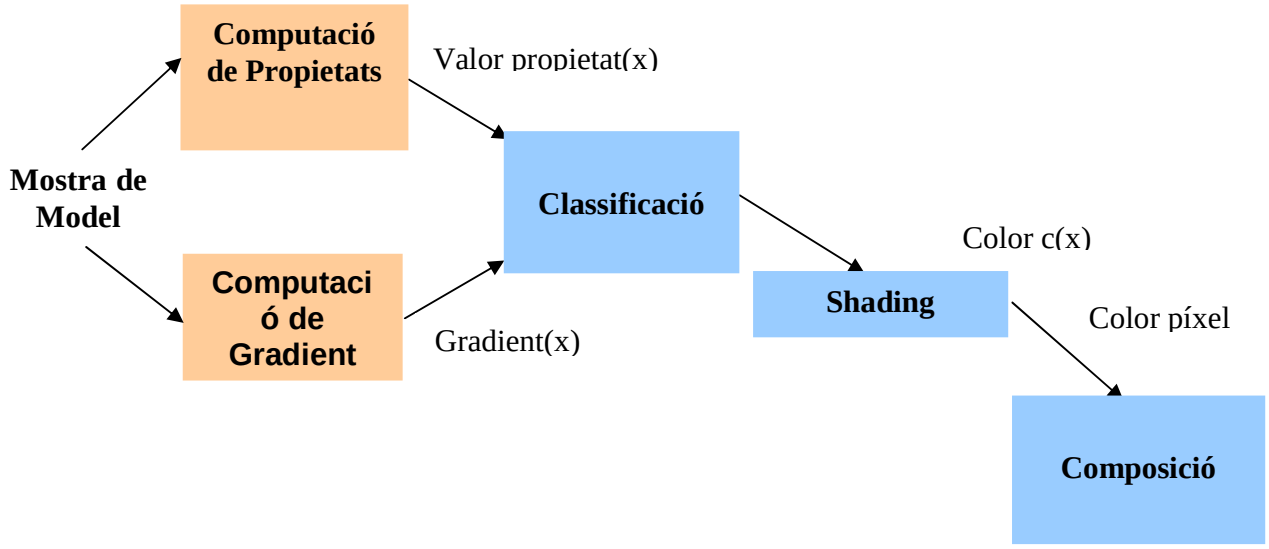


Figura 2.10: *Els diferents passos en el procés de visualització d'un model simple. Per fusionar dos models podem escollir en quina d'aquestes fases fer-ho.*

Per poder visualitzar un model fusionat hem de veure en quin punt de tot el procés de visualització fem la fusió de la informació. De fet les diferents estratègies que podrien aplicar-se es podrien classificar en funció de si es fusionen inicialment les propietats en el primer pas i llavors s'aplica el procés clàssic de visualització, o bé si primer es classifiquen els valors de forma independent i a continuació es fa la fusió de la informació que retorna la classificació, etc.

Els aspectes principals que cal tenir en compte en la visualització multimodal són

- Tipus d'integració visual, com es realitza la fusió
- Flexibilitat al canviar paràmetres visuals, és a dir com poden modificar-se les diferents propietats gràfiques que afecten a cada model.
- Estabilitat del rendering al canviar paràmetres del procés de visualització.

Tenim dos tipus de visualització de models fusionats: Una propietat per model(OPS) i Multiples propietats per model(MPS).

Una propietat per model

- La imatge generada és semblant a la generada amb un model simple, el que passa és que en el procés de visualització s'han tingut en compte varies propietats.

- Modalitat RM i CT. Totes dues enregistren informació anatòmica però representen diferents teixits.

Múltiples propietats per model

- Cada propietat representa un material diferent i la fusió d'aquests es realitza durant el procés de visualització.

2.3.3 Tipus de fusió

Els algoritmes de visualització de models fusionats difereixen en el pas on es realitza la fusió:

1- Fusió de propietats

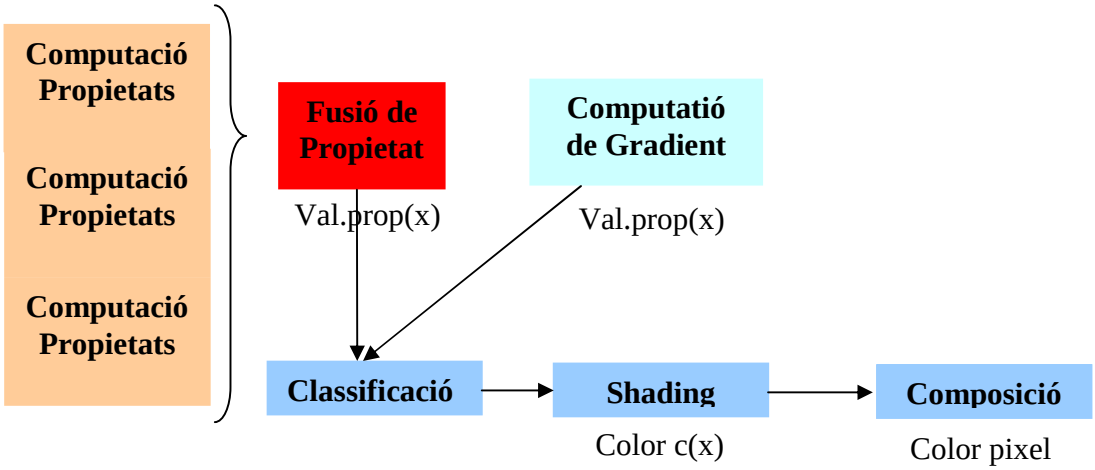
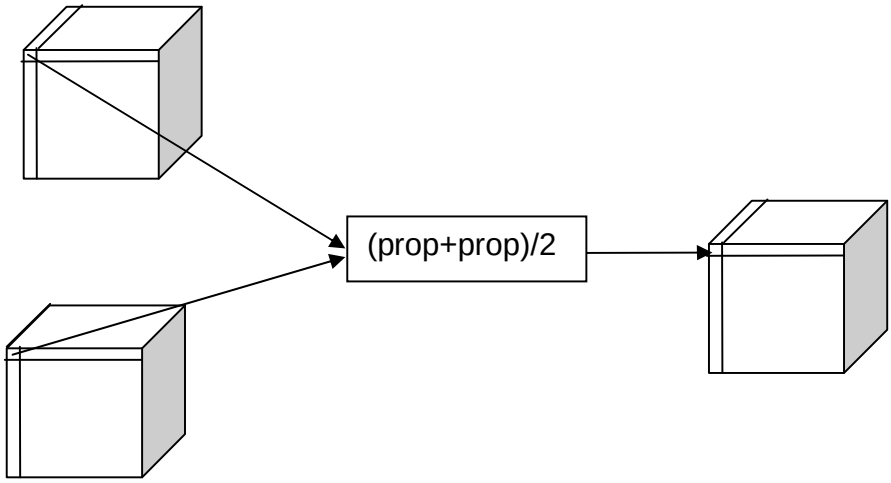


***Figura 2.11**: La fusió es fa al computar les propietats.*

Aquest algoritme agafa les propietats dels models i les fusiona mitjançant una interpolació o algun altre càlcul per generar un nou volum ja fusionat.

Aquesta tècnica genera una visualització estable a la modificació de paràmetres com pot ser la càmera, el model de llums o la funció de transferència. Si fem un canvi en el model de dades s'ha de fer una nova execució. S'ha de tenir en compte que hem de trobar una funció de transferència nova ja que la que fèiem servir per visualitzar els models anterior no té perquè servir-nos pel model fusionat.



2- Property and Gradient fusion

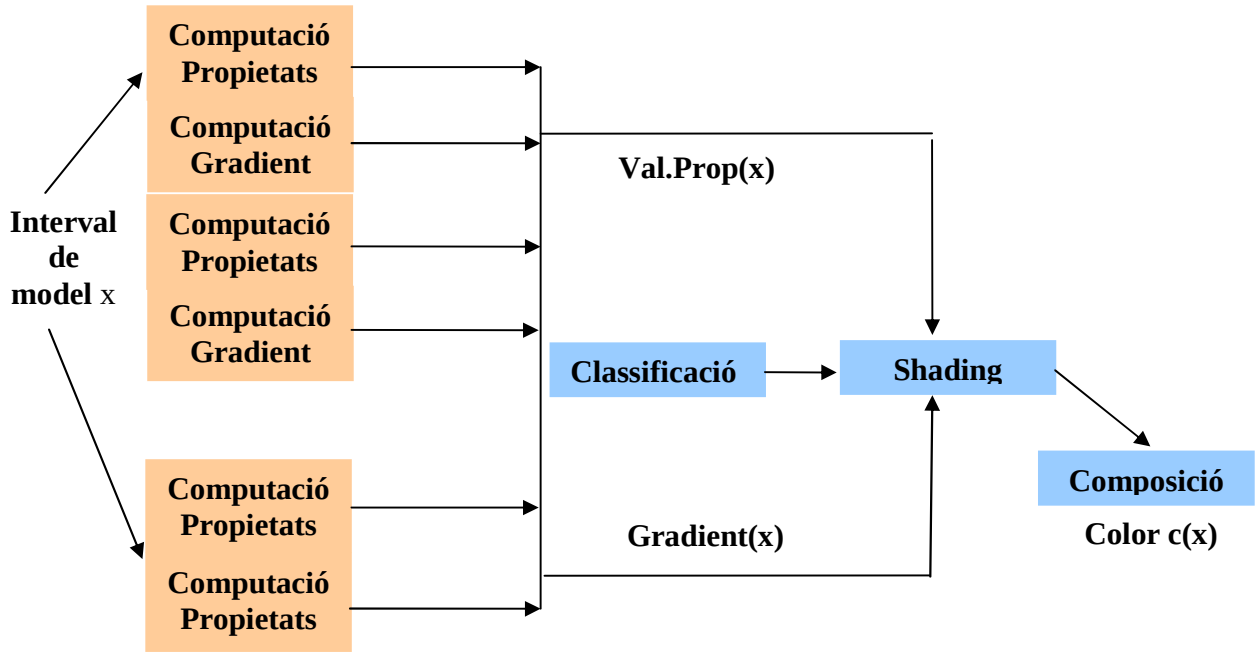
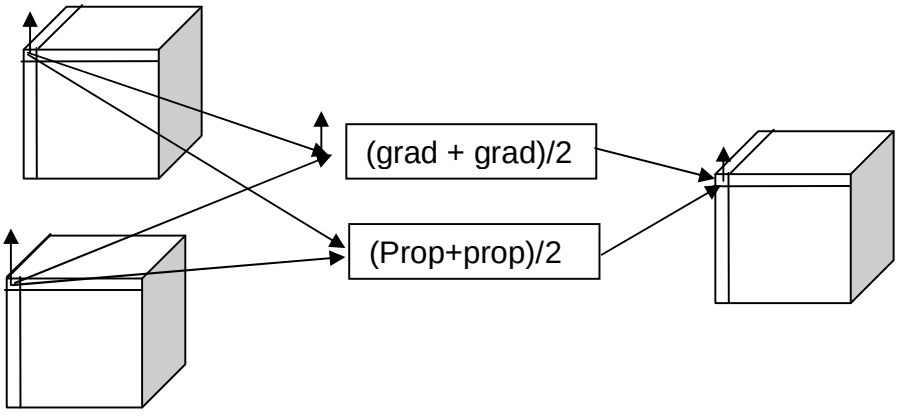


***Figura 2.12**: La fusió es fa al classificar les propietats amb el gradient.*

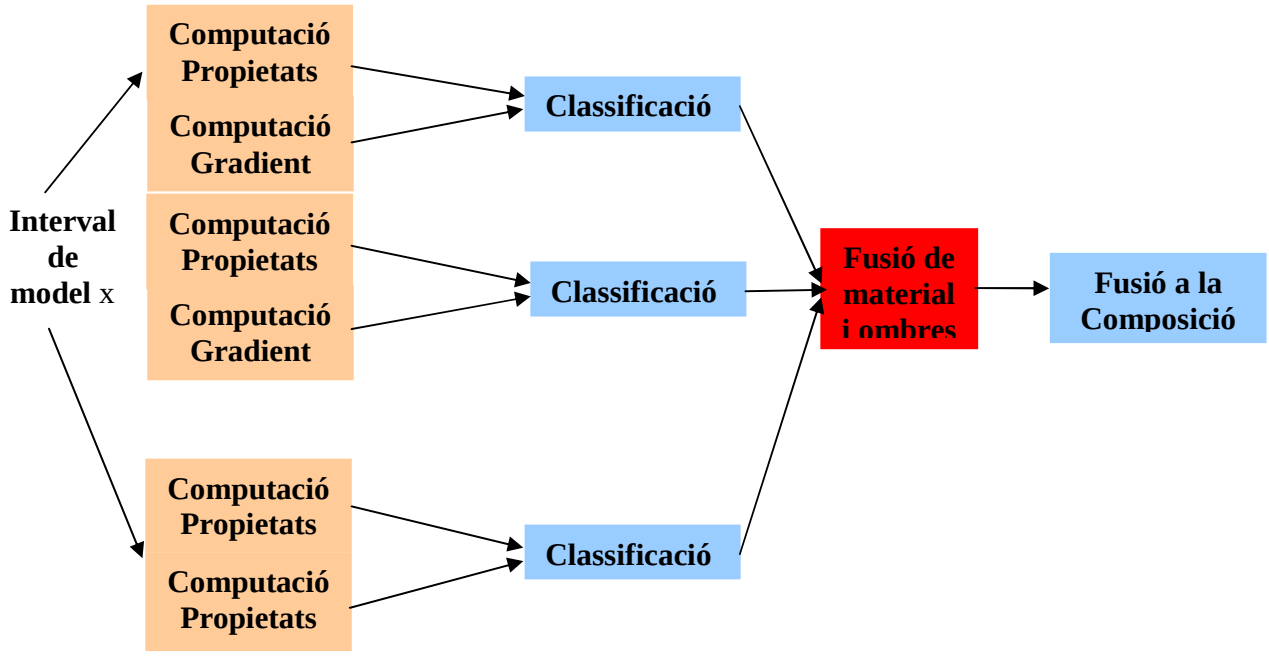
El gradient calcula la intensitat màxima en cada punt de la imatge. Es fa servir per detectar fronteres en imatges, ja siguin 2D o 3D.

En la visualització 3D, consisteix en calcular els valors de propietat i els vectors de les normals en aquell punt. Aquesta tipus d'algoritme té en compte els valors de propietat dels veïns. Si el valor del gradient és molt alt vol dir que ens trobem amb una frontera. Si és petit vol dir que no hi ha canvis respecte els seus veïns. En el cas de visualització de models, el gradient ajuda a determinar el canvi de propietats, és a dir, a delimitar cada teixit(propietat) d'una manera més remarcada i precisa. S'utilitza sobretot per aplicar ombres a la il·luminació.

El comportament és semblant a la FP i també és estable als canvis de llum i càmera. El problema és que té un comportament molt més lent.



3- Material Fusion



Aquest tipus de fusió es realitza en temps d'ombreig. En aquesta fase es tenen en compte les propietats dels materials que componen el volum i el tipus de llums de l'escena. Es fa servir per obtenir imatges amb molt realisme i definició.

Cada material té unes propietats gràfiques diferents. A part del color, tot material té unes capacitats per absorbir determinat tipus de llum. Aquestes propietats són la component difusa, ambient i especular. A més les llums que intervenen en l'escena emeten un tipus de raig. Aquest feix de llum té una component difusa, una especular i una ambient. Segons les propietats del material i la llum, el material tindrà una intensitat de llum o una altra i per tant un color o un altre.

En aquest projecte, al tractar-se d'una aplicació mèdica, no s'ha cregut necessari implementar una tècnica de visualització amb tant novell de detall. A més, el visualitzador de models simples de la plataforma no incorpora aquestes tècniques encara.

4- Fusió de Colors

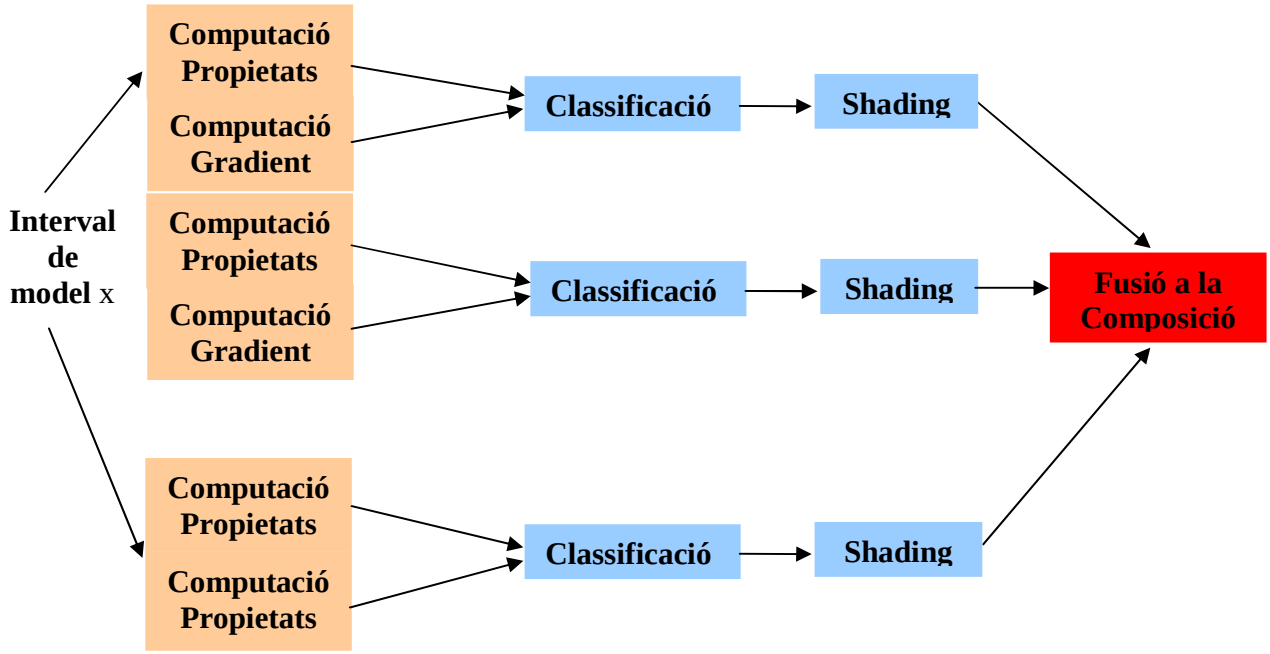
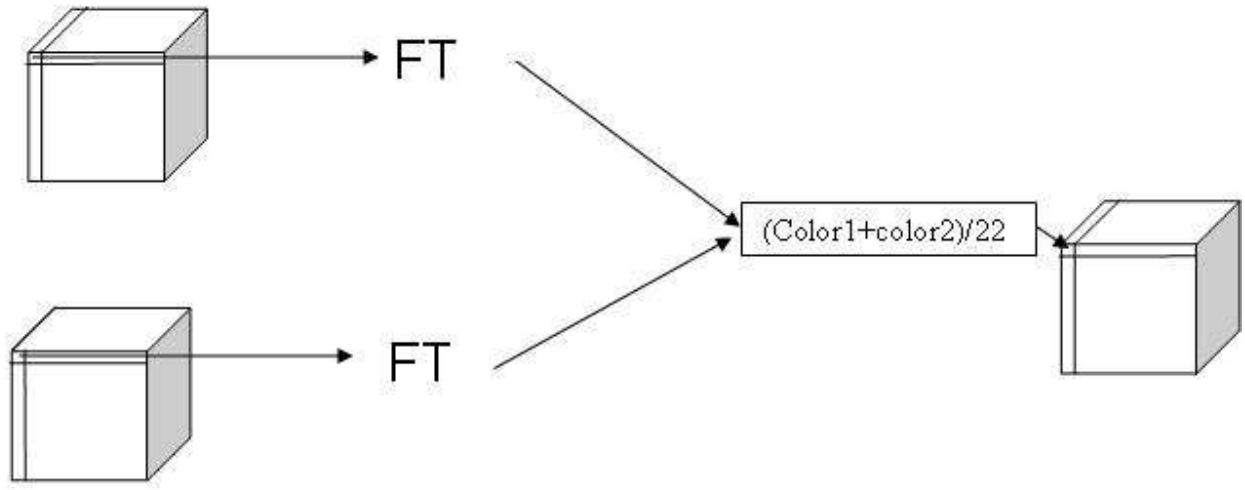


***Figura 2.13**: La fusió es fa al composar la imatge final.*

Aquest algorisme s'aplica un cop assignada la funció de transferència,és a dir, un cop seleccionat el color que tindrà un cert valor de propietat, es fusionen els colors dels dos models. Així obtindrem la informació ja tractada només haurem de visualitzar-la.



3 FONAMENTS PRÀCTICS

Dins d'aquest apartat s'explicarà quines eines i llibreries s'han fer servir en la implementació del projecte. La plataforma sobre la que s'implementa aquest projecte és l'Starviewer. Bàsicament s'han fet servir les llibreries Qt, Vtk i Itk i pel que fa a eines de desenvolupament s'han fet servir diferents aplicacions i entorns IDE com Kdevelop, Qt Designer, etc.

3.1 ESTUDI DE L'APLICACIÓ

L'Starviewer està desenvolupat conjuntament amb el Laboratori de Gràfics i Imatge de la Universitat de Girona i l'Institut de Diagnòstic per la Imatge de l'hospital Josep Trueta. Aquesta plataforma proporciona funcionalitats per l'àmbit d'imatge mèdica per manipular, tractar i visualitzar dades mèdiques.

L'Starviewer incorpora funcionalitats tant de tractament d'imatge 2D com de visualització de models 3D. La plataforma te un disseny modular que li permet integrar noves funcionalitats i ampliar les ja existents. La integració de noves funcionalitats implica el disseny de noves barres d'eines específiques de les noves funcions que es realitzen.

Esquema de l'Straviewer

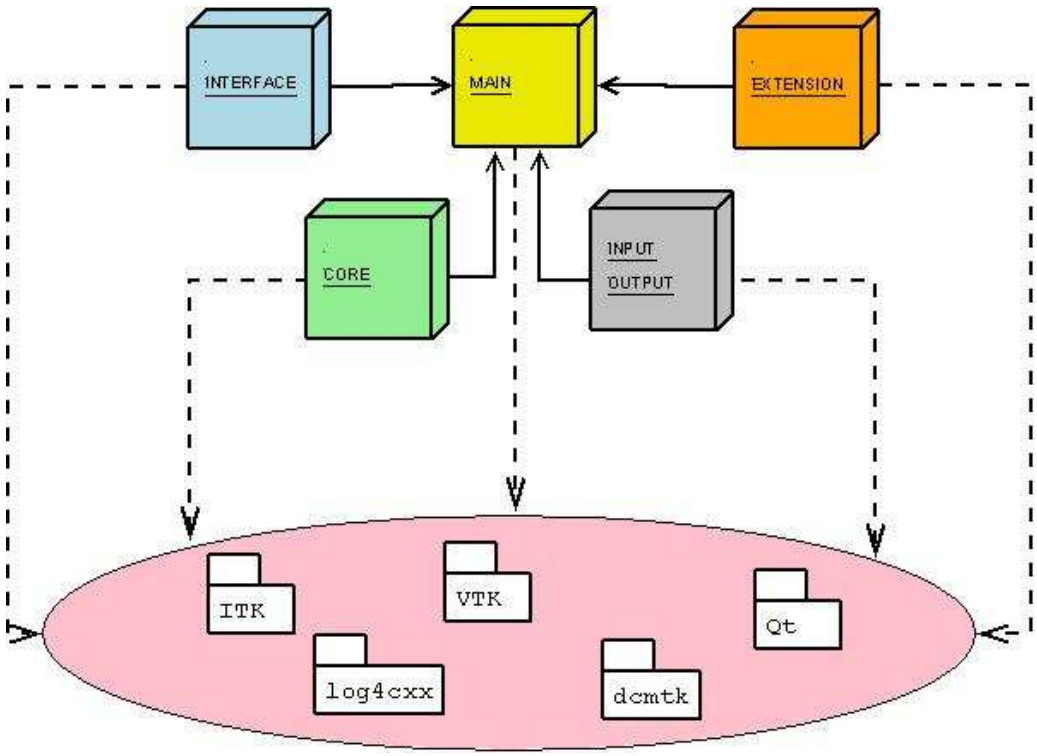


Figura 3.1. *Esquema de la plataforma Starviewer.*

En aquesta figura (***Figura 3.1***) podem observar els diferents mòduls principals que formen la nostra plataforma.

Core: Com el seu nom indica és el nucli de la plataforma. Aquí trobem els diferents visualitzadors, les Tools, els volums, etc. Són classes utilitzades per tots els mòduls per tant és lògic que formin el cor de la plataforma.

MAIN: Trobem el fitxer main i les dades necessàries per engegar l'aplicació.

Interface: Com diu el nom, aquest mòdul conté totes les classes necessàries per crear la pantalla principal, els menús de selecció d'extensió, la configuració base, en definitiva classes de configuració i interfície inicial.

InputOutput: Incorpora totes les operacions d'entrada-sortida de dades, com l'accés a la base de dades d'estudis de pacient, Exportació, impressió, etc.

Extensions: Les extensions són els diferents subaplicacions dins la plataforma. Són totes les funcionalitats de tractament d'imatges i visualització de la plataforma. Les funcionalitats es distribueixen en packages i poden afegir-se a dos submòduls: MAIN i PLAYGROUND.
MAIN són les funcionalitats afegides a l'Starviewer i dins de PLAYGROUND trobem les funcionalitats que s'estan desenvolupant o testejant.

3.2 QT

Les llibreries Qt són un framework estàndard per desenvolupar aplicacions d'alt rendiment . Funcionen sota l'entorn C++ i permeten desenvolupar aplicacions amb interfície gràfica d'usuari(GUI). Són totalment orientades a objectes i multiplataforma, permetent la portabilitat de codi entre Windows, Mac OS, Linux, etc.

Destaquen per la gran facilitat que proporcionen alhora de desenvolupar aplicacions gràfiques. Contenen un conjunt de widgets que cobreixen totes les necessitats d'una aplicació actual amb els seus menús, barres, etc. A més el seu sistema de gestió d'advents és molt senzill i funcional.
Fa servir signals i slots per intercomunicar els objectes a més de gestionar els esdeveniment normals d'una aplicació com poden ser les tecles del teclat o el botons del ratolí.

Pel que fa a la part de gràfics que és bàsica en aquest projecte trobem que les qt suporta models 2D i 3D a més de connexió a bases de dades independents a la plataforma. En el projecte es fan servir les actuals QT4 que són força diferents a les anteriors qt3

WIDGETS

Els widgets són tots els elements visuals que trobem en una interfície d'aplicació. Són els botons, llistes, finestres, menús i poden tenir un esdeveniment associat o simplement aportar informació o disseny. Es poden crear nous widgets com a subclasse dels existents i anar creant un arbre amb "pares i fills". La classe principal és Qwidget i la resta de subclasses es despengen d'aquesta.

Per fer el disseny d'una interfície Qt es pot fer amb codi senzill, però dificulta molt la feina. Nosaltres fem servir Qt Designer.

Signals i Slots

Quan fem servir qualsevol aplicació gràfica tenim l'opció de fer diferents accions. Normalment amb el Mouse premem un botó de la interfície i el programa executa la acció desitjada. Per això es diuen interfície gràfica d'usuari, perquè l'usuari interactua amb ella. Per gestionar aquesta interacció les Qt fan servir un sistema de "signals i slots" (senyal i forat). Estan implementats amb C++, orientats a objectes i permeten molta flexibilitat alhora d'ordenar les accions.

La manera de funcionar és senzilla. Cada Widget emet un **signal** quan es produeix un esdeveniment. Aquest **signal** es connecta a un **slot**. Un slot és una funció d'un altre objecte o classe. Tot això es connecta fent servir la funció **connect()** d'aquesta manera:

connect(botó, SIGNAL(clicked()),qApp, SLOT(sortir()));

botó és l'objecte que fa l'acció
clicked() és l'esdeveniment observat
qApp és l'objecte que rep la crida
Sortir és la funció que s'ha d'executar.

Per poder fer servir aquest mètode cal que la classe que el necessita hereti de Qobject o alguna de les seves subclasses i també ha d'incloure la macro Q_OBJECT a la definició de la classe.

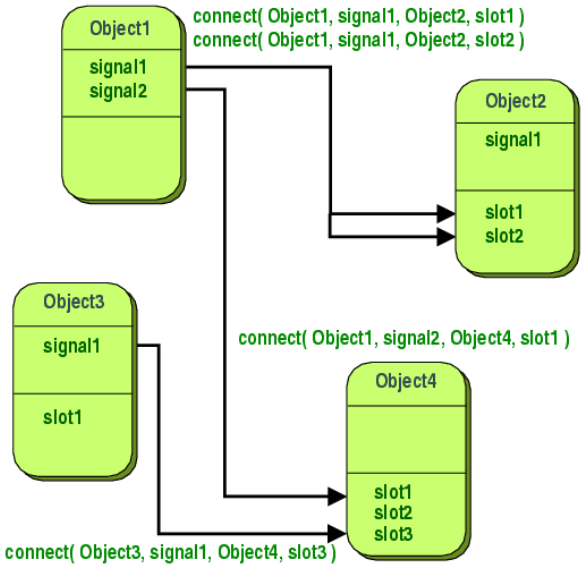


Figura3.2 *:Funcionament dels signals i slots*

Per declarar un signal es reserva un espai propi com és el de públic o private. Els slots també, però podem tenir public slots, protected slots o private slots.

Un signal pot tenir diferents slots connectats a ell. Quan s'emet aquest signal s'executen tots. Per emetre un signal es fa servir la instrucció "emet" seguit del nom del signal i passant els signals que calgui.

Per connectar dos BankAccount diferents fem servir:

BankAccount x, y;

connect(&x, SIGNAL(balanceChanged(int)), &y, SLOT(setBalance(int)));

En el pas de paràmetres només fem servir el tipus de dada que s'envia no el nom.

Observem que quan compilem una aplicació Qt es creen uns objectes moc_nomclasse. Això és perquè els signals i slots necessiten el Meta-Object Compiler(moc) que no és més que un generador de codi C++ estàndard. Genera el codi necessari per crear les connexions.

QMAKE

És un programa que inclouen les QT que genera el makefile necessari per compilar l'aplicació. Fa servir un arxiu .pro on s'especifica quines classes es fan servir. Aquest arxiu el podem fer nosaltres o bé demanar al **qmake** que ens generi un esbós fent **qmake -project**. És important que aquest arxiu sigui correcte ja que sinó no podrem compilar i per tant executar la nostra aplicació.

QT Designer

Aquest programa és un editor d'interfícies Qt. Ens permet fer tot tipus d'aplicacions agregant els diferents widgets que ens proporciona les llibreries Qt. Té un funcionament molt intuïtiu i senzill. Podem seleccionar una plantilla que ens proporciona l'aplicació o bé crear-ne una des de 0. Al programa trobem l'àrea de la nostra finestra al centre, un menú dels widgets de QT a l'esquerra, i a la dreta el menú de propietats de cada widget. També podem veure la relació entre els widgets(pares i fills) i les connexions entre objectes(signal slots).

Els arxius que treballem amb el qtDesigner tenen l'extensió .ui. És un fitxer xml amb d'informació necessària per generar el codi C++ per crear d'interfície. Funciona semblant a com ho fa el moc, però en aquest cas fem servir el programa uic(User Interface Compiler) que proporciona les Qt.

3.3 ITK

Les llibreries ITK ens donen la funcionalitat necessària per efectuar operacions de registre o segmentació d'imatges. Insight ToolKit és de codi obert, implementades en C++, multiplataforma i fan servir programació genèrica(templates).

Estan dissenyades principalment per desenvolupar aplicacions mèdiques tot i que es poden fer servir per qualsevol altre tipus de dades.
Les dades segueixen una estructura de pipeline o flux de dades on les dades el poden processar a trossos, és a dir, anar fent passos entremitjos abans del resultat final.
A més fan servir patrons de disseny com el Factory Method per instanciar objectes o l'Observer per processar els esdeveniment.
Com a complement les ITK suporten el processament paral·lel amb multi-threading i memòria compartida així com els smart pointers , que tenen comptador de referències a objectes.

En la plataforma STARVIEWER es fa servir per les operacions de registre i segmentació. Identifica i classifica les dades d'un mostreig digital obtingut d'un aparell de captació com una TAC o una RM. En el registre es fa servir per alinear les dades de dues mostres diferents. En aquest projecte es fa servir models registrats, però el registre no necessita les llibreries ITK ja que l'alineació de les dades ja ve donada.

3.4 VTK

Visualization ToolKit són unes llibreries gràfiques pel processament d'imatges i visualització. Com tot el fet servir en aquest projecte està implementat en C++, orientat a objectes i multiplataforma.

A diferència per exemple de OpenGL, les VTK tenen un nivell més alt d'abstracció cosa que permet un desenvolupament més senzill d'aplicacions gràfiques. A més conté molts algorismes de visualització geomètrica, volumètrica i filtres pel tractament d'imatges.

És compatible amb molt sistemes de finestres com per exemple Qt i a més les interfícies creades amb VTK poden ser interactives, és a dir, amb esdeveniments associats. Com les ITK, aquestes llibreries fan servir patrons de disseny com l'Observer pel tractament d'aquests esdeveniments, o el Factory Method per la creació d'objectes.

També funciona amb pipelines(flux de dades) per tractar les dades per parts i pot executar aquestes parts paral·lelament(multi-threading). Fa servir comptadors de referència pels objectes.

Les VTK inclouen tot el necessari per visualitzar qualsevol tipus de dades ja siguin punts, superfícies o volums i per això ens ofereixen diferents algoritmes de visualització com poden ser Ray Casting, Marching Cubes, etc. A més incorporen filtres per tractar les dades prèviament i objectes imprescindibles en el procés de visualització com són les llums, càmeres , textures, etc.

Dins aquesta llibreria podem identificar dos tipus de models d'objectes:

3.4.1 Model de gràfics

Aquest model busca traduir les dades gràfiques en imatges. Per fer-ho conté un conjunt d'objectes que inclou els actors i volums(props), les llums, les càmeres, els mappers dels actors i volums, les funcions de transferència, els renderers, render windows, render window interactors i les Look-Up Tables. Tots aquest objectes formen la pipeline de visualització.

Els actors o volums són els objectes més importants de l'escena. És el contingut que es visualitzarà. Podem modificar la seva aparença mitjançant propietats com les LookUp Tables o les funcions de transferència i també podem variar la seva geometria fent servir el mapper. El mapper bàsicament connecta el model de visualització amb el model de gràfics. En aquest model també trobem les llums i les càmeres que funcionen d'una manera molt similar a com ho fan en OpenGL o altres eines de visualització. Per últim, el renderer és l'objecte encarregat de pintar la imatge al render window, és a dir, a la finestra que contindrà l'escena.

Per últim podem agregar interacció de l'usuari afegint l'escena dins l'objecte render window interactor. Així podrem canviar diferents propietats de la nostra escena(rotar, ampliar, canviar colors, etc) capturant els esdeveniments.

3.4.2 Model de visualització

El model de visualització transforma la informació en dades gràfiques que posteriorment es visualitzaran en imatges en el model de gràfics.

Aquí trobem els diferents filtres que s'apliquen a les dades d'entrada per generar noves dades tractades de sortida. Dins aquest apartat estan els algoritmes de tractament d'imatge com poden ser el ray casting o marching cubes.

4 ANALISI DE L'APLICACIÓ

4.1. Anàlisi de requeriments

4.1.1 Requeriments funcionals

Aquí s'engloben les funcions principals de la nostra aplicació. Són totes les accions que es podran fer dins el nostre mòdul d'aplicació.

Visualització de models fusionats:

1. Obrir el volum principal.
2. Obrir el volum secundari.
3. Definir la funció de transferència.
4. Definir quin mètode de fusió utilitzar.
5. Aplicar mètode fusió:
    a. Fusió de propietats.
    b. Fusió de gradient i propietats.
    c. Fusió de colors.
6. Interaccionar amb la visualització
7. Definir la funció de transferència del model fusionat.

4.1.2 Requeriments no funcionals

Bàsicament hem de construir una interfície gràfica d'usuari que s'ajusti a la plataforma STARVIEWER i que permeti a l'usuari accedir a totes les funcionalitats del nostre mòdul.

Haurem d'adequar-nos tant a l'entorn de treball de la plataforma com a l'ús de les seves llibreries. Com hem explicat abans, l'Starviewer fa servir les llibreries Qt, VTK i ITK a més d'estar desenvolupada en C++. Fent servir això ens assegurem que el mòdul sigui també programari lliure.

4.2 DISSENY DE L'APLICACIÓ

En aquest apartat es mostraran els dissenys fets previs a la implementació de l'aplicació. Aquest dissenys són cassos d'ús, les fitxes d'aquests casos d'ús i el diagrama de classes. Els diagrames de seqüència s'han afegit a la part d'implementació per tal d'entendre millor el funcionament

4.2.1 Cas d'ús general de l'aplicació

La figura 4.1 ens mostra

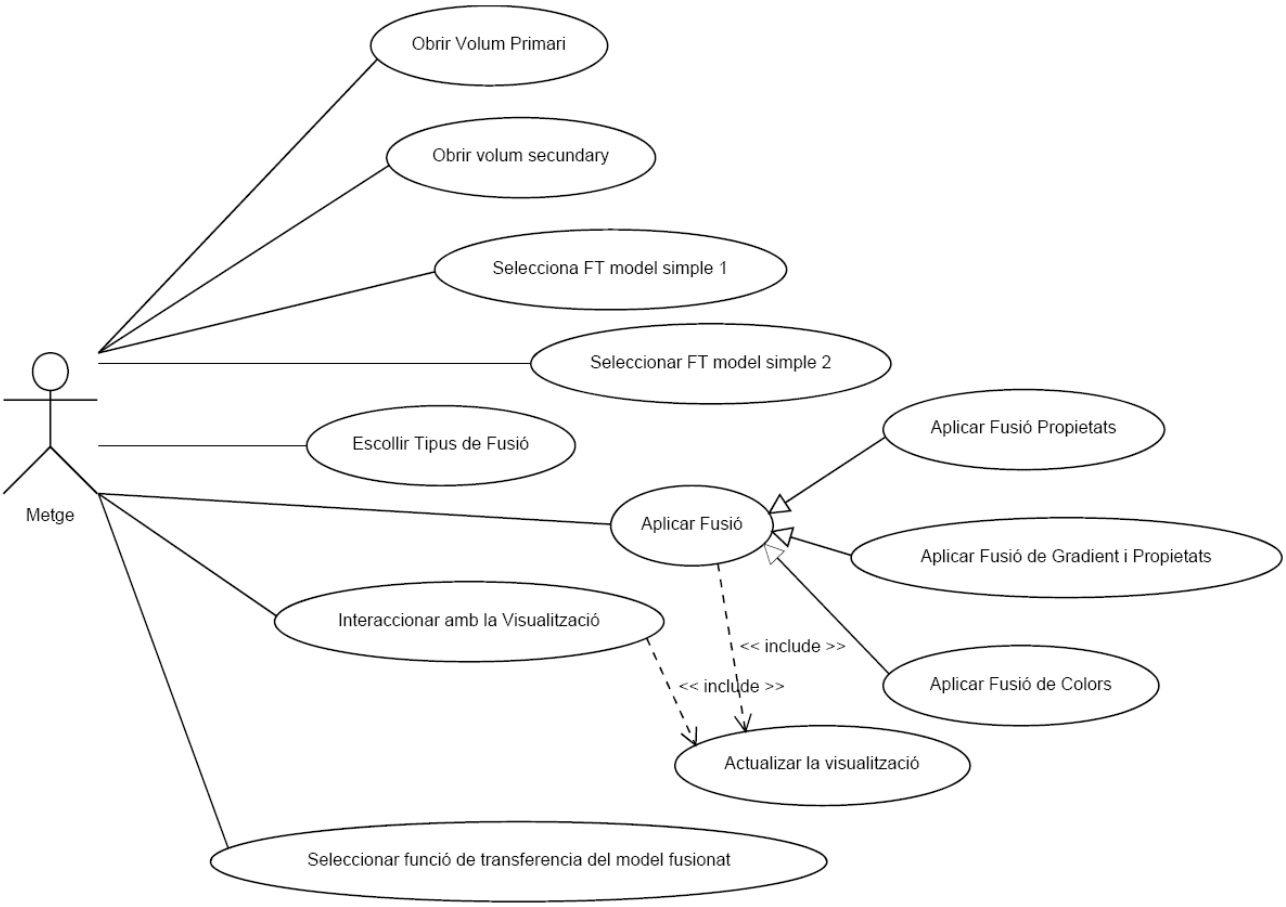


Figura 4.1: *Funcionalitats bàsiques de l'aplicació.*

4.2.2 Fitxes de casos d'ús

En el següent apartat s'especifiquen tots les fitxes de cassos d'ús del disseny de l'aplicació.

Cas d'ús	Obrir Volum Primari
Descripció	S'obre el volum primari necessari per obrir el nostre mòdul
Actors	Metge
Precondició	Aplicació iniciada
Flux principal	1. Es mostra el diàleg per obrir el fitxer 2. Escollir el fitxer desitjat 3. Obrir el fitxer
Fluxos alternatius	Cancel·lar obrir volum secundari
Post condició	S'obre el mòdul d'aplicació. Es visualitza el model primari.

Cas d'ús	Obrir Volum Secundari
Descripció	S'obre el volum secundari necessari per fer la fusió
Actors	Metge
Precondició	El visualitzador té el volum principal obert
Flux principal	1. Es mostra el diàleg per obrir el fitxer 2. Escollir el fitxer desitjat 3. Obrir el fitxer
Fluxos alternatius	Cancel·lar obrir volum secundari
Post condició	Es visualitza el model secundari S'activa el botó per aplicar la fusió

Cas d'ús	Seleccionar FT model simple 1
Descripció	Seleccionem i apliquem la funció de transferència al model primari
Actors	Metge
Precondició	El model ha d'estar obert
Flux principal	1. S'obre l'editor de funció transferència 2. Editar o escollir la FT. 3. Aplicar la FT al model principal 4. Tancar l'editor
Fluxos alternatius	Tancar l'editor sense aplicar la FT
Post condició	S'aplica la FT i s'actualitza la visualització

Cas d'ús	Seleccionar FT model simple 2
Descripció	Seleccionar i aplicar la funció de transferència al model secundari
Actors	Metge
Precondició	El model secundari està obert
Flux principal	1. S'obre l'editor de funció de transferència 2. Editar o escollir la FT. 3. Aplicar la FT al model secundari
Fluxos alternatius	Tancar l'editor sense aplicar FT
Post condició	S'aplica la FT I s'actualitza la visualització

Cas d'ús	Escollir tipus de fusió
Descripció	Escollir entre les opcions de fusió que dona l'aplicació
Actors	Metge
Precondició	Tenir el mòdul obert
Flux principal	1. Clicar sobre el comboBox 2. Escollir el tipus dins la llista 3. Guardar el tipus de fusió que s'aplicarà
Fluxos alternatius	Fer clic fora de la llista i anular en canvi.
Post condició	Es canvia el mètode de fusió

Cas d'ús	Aplicar Fusió
Descripció	S'aplica la fusió de models seleccionada
Actors	Metge
Precondició	Tenir els dos models oberts i haver escollit un tipus de fusió. Per defecte és fusió de propietat 1
Flux principal	Es visualitza el model fusionat
Fluxos alternatius	No n'hi ha
Post condició	S'actualitza el visualitzador amb la representació del model fusionat. S'afegeixen les eines d'interacció.

Cas d'ús	Aplicar Fusió Propietats
Descripció	Aplicar la fusió de propietats dels 2 models oberts.
Actors	Metge
Precondició	Tenir els dos models oberts
Flux principal	1- extreure les dades dels models simples 2- recórrer les dades calculant les noves fusionades 3- crear un volum amb les noves dades 4- visualitzar el model fusionat
Fluxos alternatius	No n'hi ha
Post condició	S'actualitza el visualitzador amb la representació del model fusionat. S'assigna una funció de transferència estàndard.

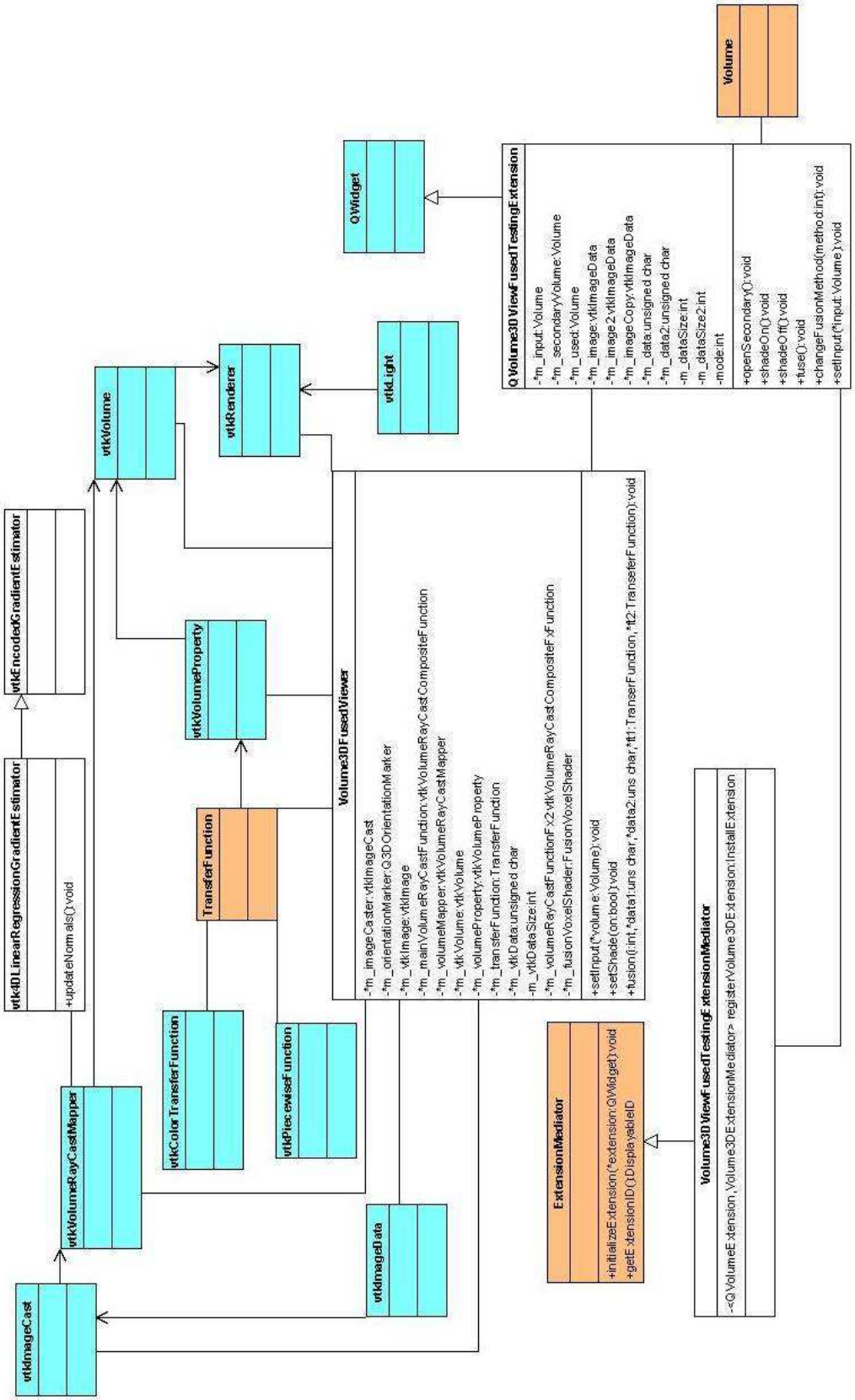
Cas d'ús	Aplicar Fusió de gradient i propietats
Descripció	Aplicar la fusió de gradient i propietats dels dos models
Actors	Metge
Precondició	Tenir els dos models oberts
Flux principal	1- extreure les dades dels models simples 2- recórrer les propietats calculant les noves fusionades 3- recórrer els gradients calculant els nous fusionats 4- crear un volum amb les noves dades 5- visualitzar el model fusionat
Fluxos alternatius	No n'hi ha
Post condició	S'actualitza el visualitzador amb la representació del model fusionat. S'afegeixen les eines d'interacció. S'aplica una funció de transferència estàndard

Cas d'ús	Aplicar Fusió de colors
Descripció	Aplicar la fusió de gradient i propietats dels dos models
Actors	Metge
Precondició	Haver aplicat la fusió
Flux principal	1. extreure les dades dels models simples 2. afegir les propietats en un voxel shader 3. recórrer les propietats i calcular els colors 4. assignar els colors al visualitzador 5. visualitzar el model fusionat
Fluxos alternatius	No fer res
Post condició	La visualització s'actualitza

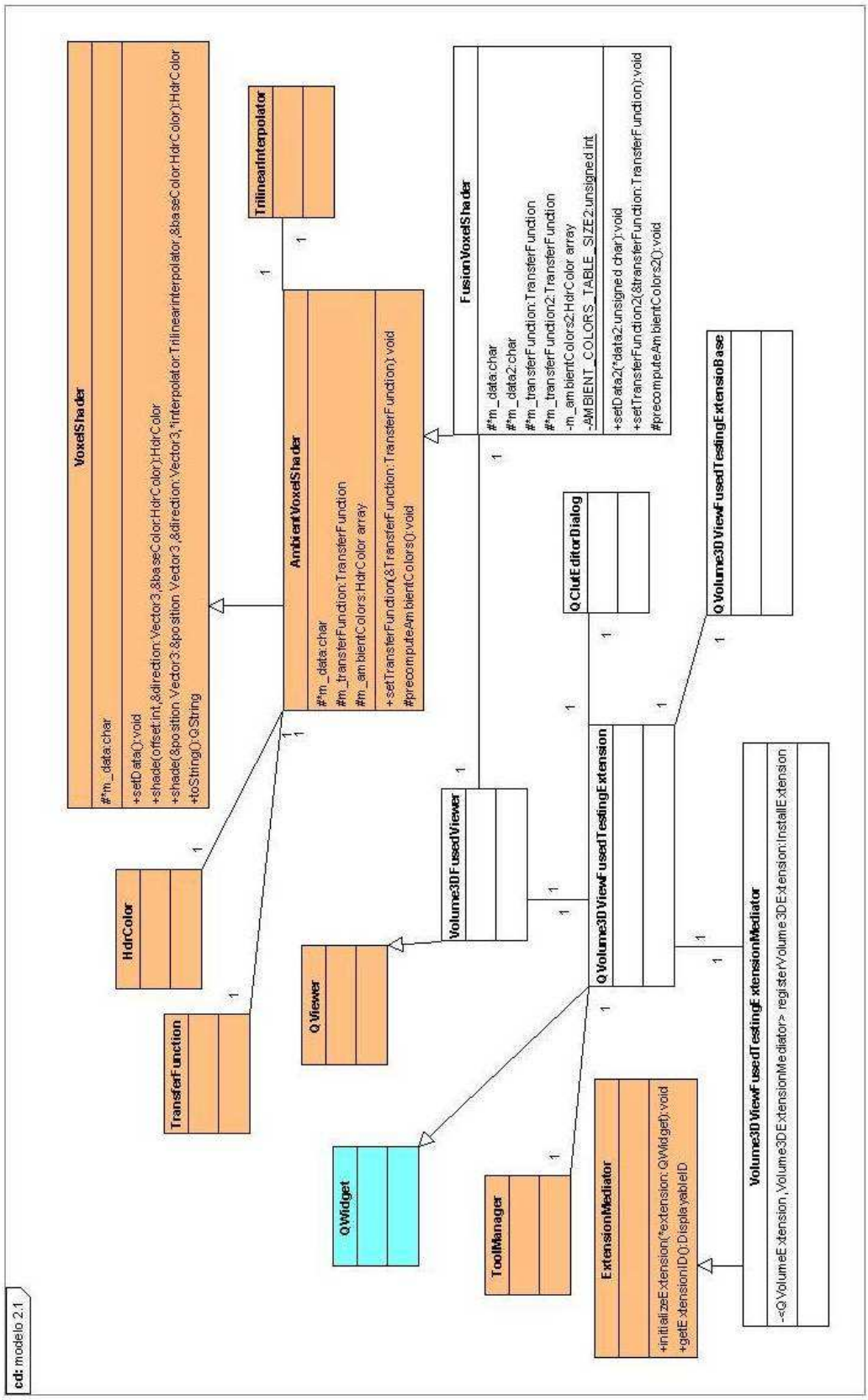
Cas d'ús	Interaccionar amb la visualització
Descripció	Canviar la visualització del model fusionat. Interactuar amb l'objecte obert.
Actors	Metge
Precondició	Haver aplicat la fusió
Flux principal	Escollir la posició, el zoom i l'orientació de la càmera que enfoca el model.
Fluxos alternatius	No fer res
Post condició	La visualització s'actualitza

Cas d'ús	Actualitzar la visualització
Descripció	En realitzar algun canvi de paràmetres ja siguin de la càmera o de la funció de transferència
Actors	Metge
Precondició	Haver fet la fusió
Flux principal	Tenir en compte els paràmetres i visualitzar de nou
Fluxos alternatius	No n'hi ha
Post condició	Visualització del model.

4.2.3 Diagrama de classes general



4.2.4 diagrama de classes general 2



5 Implementació

5.1 PAS PREVI

El primer pas abans de començar a dissenyar l'aplicació ha sigut entendre correctament el funcionament de tota la plataforma feta servir per desenvolupar l'STARVIEWER. És un projecte que porta forces anys desenvolupant-se i té una organització adient a les necessitats. Els aspectes més importants que cal conèixer abans de dissenyar l'aplicació són aquest punt:

- Adaptar-se al funcionament de Kdevelop: Funciona com qualsevol altre IDE tot i que té alguna particularitat.

- Afegir un nou mòdul a la plataforma: Per inserir un nou mòdul a l'STARVIEWER cal modificar arxius de configuració així com seguir unes directrius alhora de crear el nostre nou objecte. Aquests arxius bàsicament són text amb informació de les rutes de les llibreries i directoris per indicar als makefiles que han de compilar.

- Seguir les pautes marcades pels desenvolupadors de la plataforma per tal de tenir una estructuració dels mòduls i del codi intel·ligible per qualsevol altre persona que s'afegeixi a l'ampliació de la plataforma.

Primer s'ha de crear un nou mòdul que englobarà tots els objectes necessaris per realitzar tant la fusió de models fusionats com la interacció de l'usuari amb el visualitzador.

Com que la nostra aplicació serà gràfica (GUI) el primer que hem de fer és construir un formulari, en aquest cas Qt, i inserir-ho dins una classe base de l'extensió (extensionBase).

L'extensió no és més que una classe hereva de ExtensionMediator i inclou un objecte InstallExtension. Aquests objectes gestionen els mòduls adherits a la plataforma. L'ExtensionMediator és un objecte construït segons el patró de disseny Mediator.

La classe creada té aquesta estructura:

```
class Volume3DViewFusedTestingExtensionMediator : public ExtensionMediator
{
   Q_OBJECT
   public:

   Volume3DViewFusedTestingExtensionMediator( QObject * parent = 0 );
   ~Volume3DViewFusedTestingExtensionMediator();

   virtual DisplayableID getExtensionID() const;

   virtual bool initializeExtension(QWidget* extension, const ExtensionContext
&extensionContext);

};

static InstallExtension< QVolume3DViewFusedTestingExtension,
Volume3DViewFusedTestingExtensionMediator >
registerVolume3DViewFusedTestingExtension;

}
```

L'objecte DisplayableID guarda el nom que voldrem posar al nostre mòdul dins del menú de selecció de visualització.

Aquesta classe conté l'objecte que hem creat per contenir tots els elements necessaris per desenvolupar el nou mòdul, en aquest cas, qVolume3DFusedTestingMediatorBase. De fet el nom ho clarifica, és la base de l'extensió.

Amb això només caldrà inserir el formulari de la nova funcionalitat al ExtensionBase.
Per fer-ho primer cal crear el formulari i en aquest cas s'ha desenvolupat amb un editor de GUI Qt anomenar Qt 4 Designer. Agafant el formulari de l'extensió Volume 3D com a base, s'afegeixen els objectes necessaris per complir el objectius. Es genera un arxiu .ui amb format xml,és a dir, fent servir tags. Un cop tenim el formulari, l'afegim al nostre extensionbase per poder tractar els objectes continguts.

Com que l'objectiu d'aquest projecte està enfocat a la visualització, s'ha agafat com a model a seguir el 3DViewer. Mostra en 3D la informació del pacient fent servir diferents tècniques de visualització. Aquest mòdul es basa en el procés de visualització que especifiquen les llibreries vtk(***Figura 5.1***):

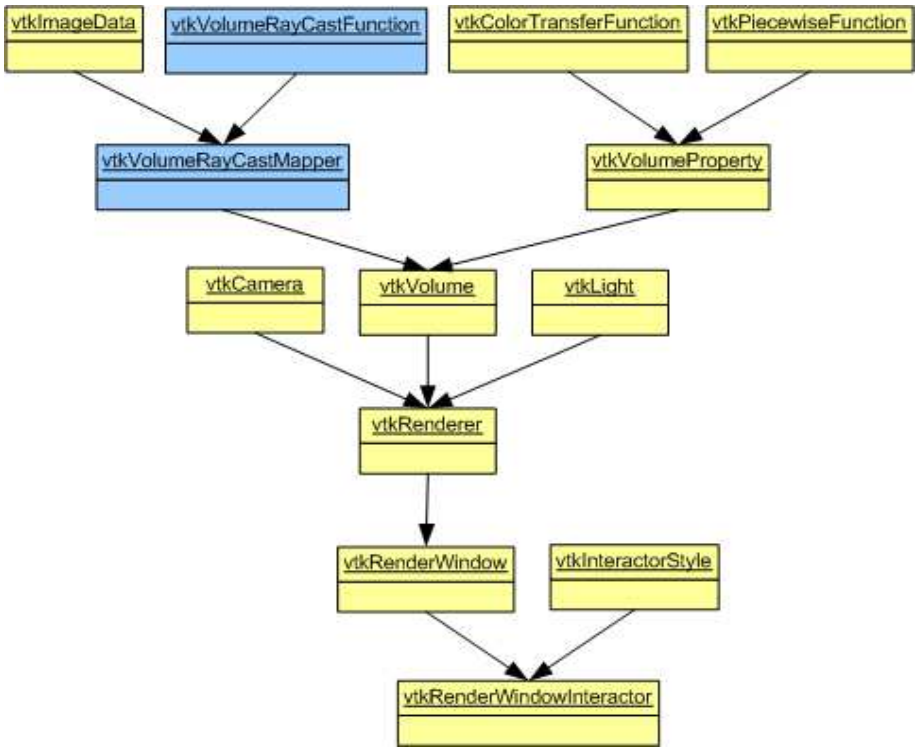


Figura 5.1: *Esquema d'interacció d'objectes de visualització en VTK.*

Observant la figura 4.1 ,podem veure que la informació del pacient la trobem enregistrada a l'objecte vtkImage que són les fotos del pacient a diferents profunditats. Aquesta informació es fa passar per un mapper que s'encarrega d'apilar les imatges i generar la informació necessària per construir un volum.

Al volum se li apliquen unes propietats bàsiques com és la funció de transferència que tradueix les propietats del volum en colors RGBa a la imatge final. Aquest Volum s'enregistra en un objecte vtkVolume. Aquest objecte s'afegeix a l'objecte vtkRenderer que és el que conté tota la informació de l'escena. Processa la informació i visualitza el model en una finestra(vtkRendererWindow). D'això se'n diu Pipeline Visualization.

En aquest projecte es segueix aquesta seqüència per visualitzar. De fet s'ha decidit crear una còpia de l'objecte visualitzador Q3DViewer fet servir al mòdul 3DViewer per tal d'incorporar alguna funcionalitat addicional sense tocar un objecte del nucli de l'aplicació. Aquest objecte s'anomena volume3DFusedViewer.

L'objecte Q3DViewer és herència de la classe Qviewer que alhora herència l'objecte Qwidget, un objecte Qt. D'aquesta manera es relacionen les llibreries VTK amb les Qt i permeten visualitzar informació gràfica VTK dins un formulari Qt. De fet, la classe principal del mòdul 3DViewer (qVolume3DViewTestingExtension) no conté l'objecte visualitzador Q3DViewer.

Es crea al formulari principal del mòdul, un formulari Qt. Des de la classe associada, qVolume3DViewTestingExtension, es veuen tots els objectes i es gestiona la visualització.

Tan el Q3DViewer com el volume3DFusedViewer contenen els objectes esmentats anteriorment en la pipeline de visualització: vtkImageData, vtkVolumeMapper, vtkTranseferFunction,vtkVolumeProperty, vtkVolume, vtkCamera, vtkLight i vtkRenderer. L'associació d'aquests objectes es realitza de la manera següent:

```
vtkRenderer *renderer;
vtkVolume *volume;
vtkVolumeProperty *volumeProperty;
vtkRenderer *vtkRenderer;
TransferFunction = *transferFunction;

renderer = vtkRenderer::New();
vtkWidget->GetRenderWindow()->AddRenderer(renderer );
volumeProperty = vtkVolumeProperty::New();
vtkVolume = vtkVolume::New();

transferFunction = new TransferFunction;

volumeProperty->SetInterpolationTypeToLinear();

volumeProperty->SetColor(transferFunction->getColorTransferFunction() );
volumeProperty->SetScalarOpacity(transferFunction->getOpacityTransferFunction() );
vtkVolume->SetProperty( volumeProperty );
renderer->AddViewProp( vtkVolume );
```

transferFunction és una classe que engloba dos objectes: vtkColorTransferFunction i vtkPiecewiseFunction. Tots dos necessaris en el procés de visualització. Forma part del core de l'aplicació. D'aquesta manera es fa més entenedor l'ús de la funció de transferència.

5.2 Obrir Model Principal

A l'Straviewer el primer que s'ha de fer per treballar amb un model de dades d'un pacient és obrir aquest model. No permet obrir cap mòdul de visualització ni tractament d'imatge sense tenir el pacient carregat.

La plataforma permet carregar diferents tipus de dades com Arxius DICOM amb el seguiment d'un pacient, connectar-se a una base de dades per extreure l'estudi de pacient, o obrir un arxiu amb extensió .mhd que és un model simple guardat en un arxiu binari. Aquest últim és el que es farà servir per aquest mòdul ja que permet obrir models sintètics generats artificialment.

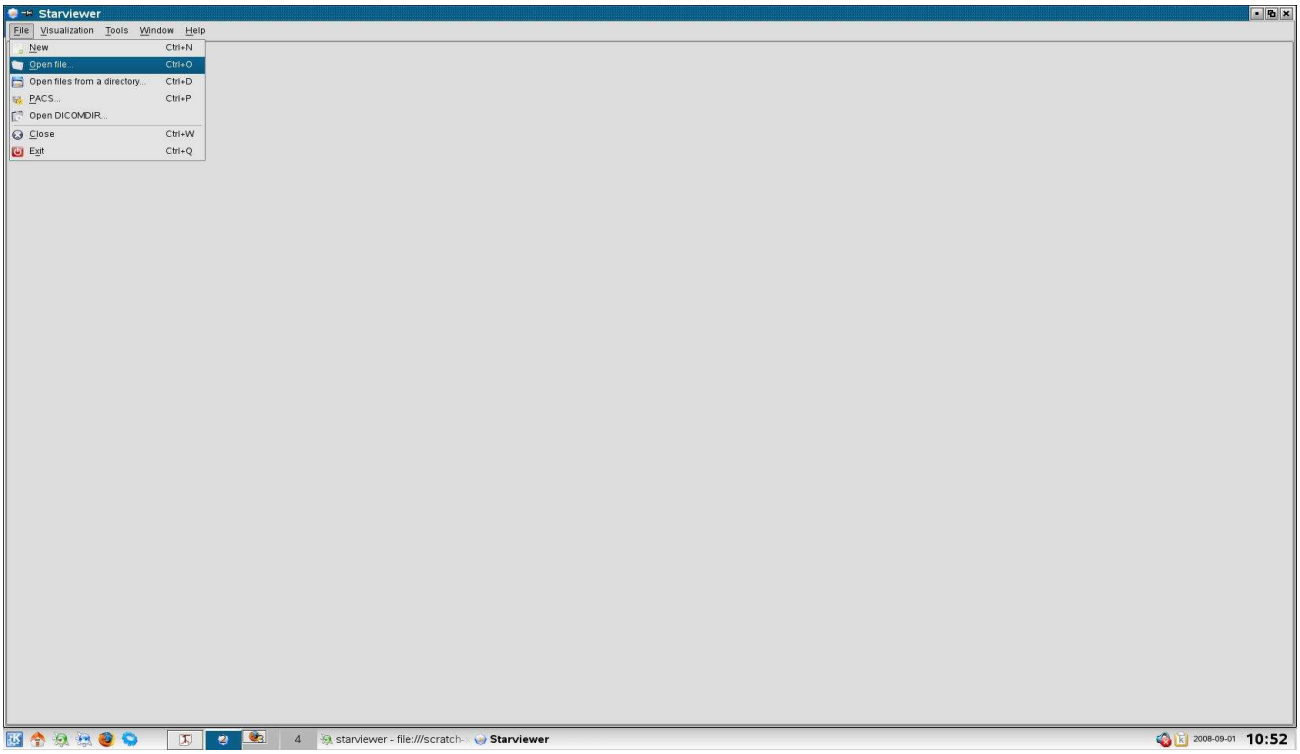


Figura 5.2: *Finestra principal de l'aplicació.*

Al crear una nova extensió dins la plataforma, ja es contempla la càrrega del model de dades en obrir-se. Es model es guarda en un objecte anomenat Volume que permet extreure les dades necessàries pel tractament desitjat.

Figura 5.3: *Diàleg per obrir un fitxer.*

En obrir el model per defecte engega el visualitzador 2D de la plataforma (**Figura 5.4**) i s'activa el menú de selecció de visualització.

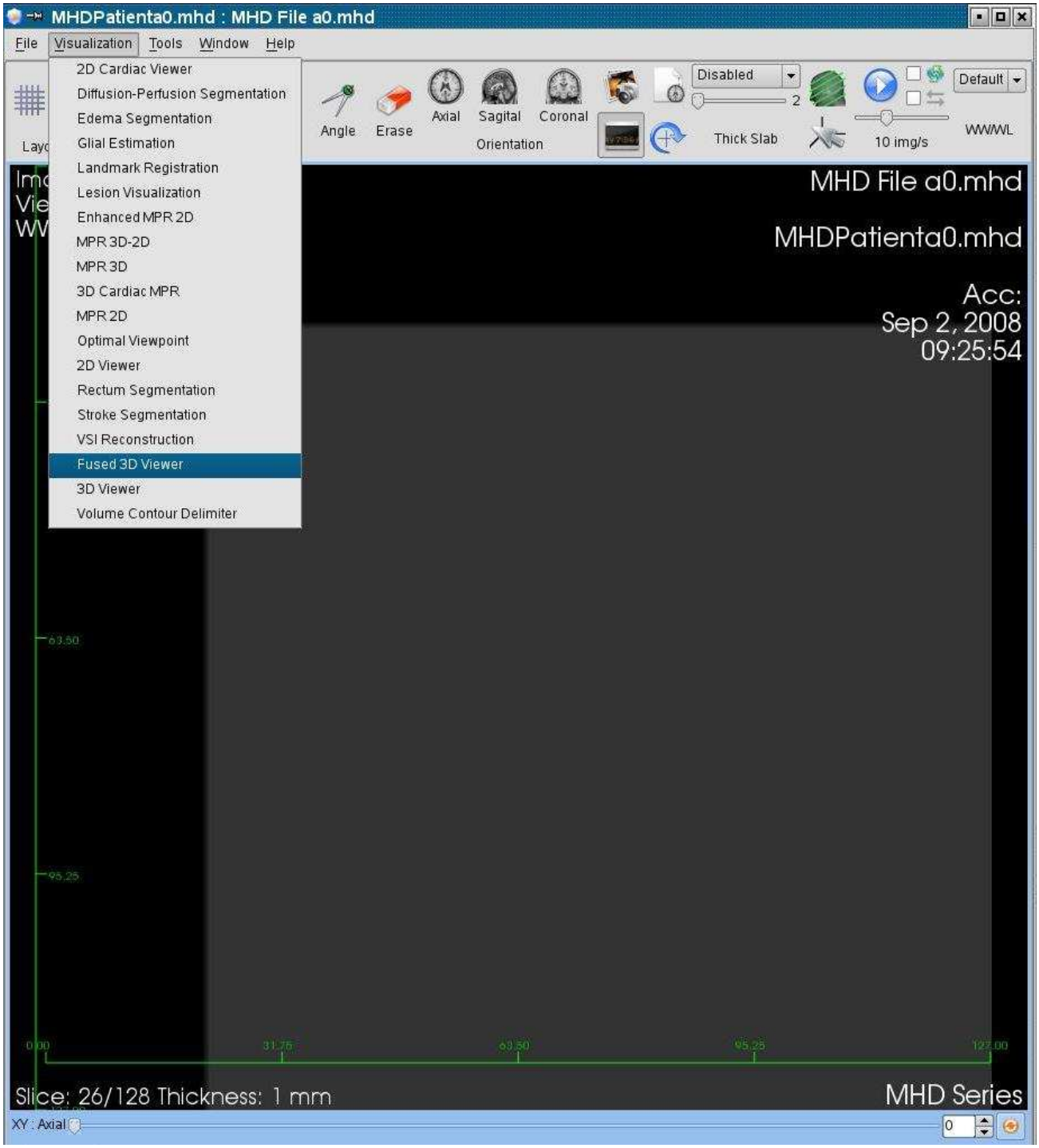


Figura 5.4: *El visualitzador 2D mostra el model de dades imatge a imagte. Ofereix moltes eines com tractar, mesurar, girar, etc per estudiar detingudament el model.*

En seleccionar el mòdul 3DFusedViewer s'obre l'aplicació que s'ha desenvolupat en aquest projecte. En la següent **figura 5.5** podem veure com es carrega el volum obert en el menú principal. És possible que el model apareixi de color negre ja que per defecte s'àssigna una funció de transferència

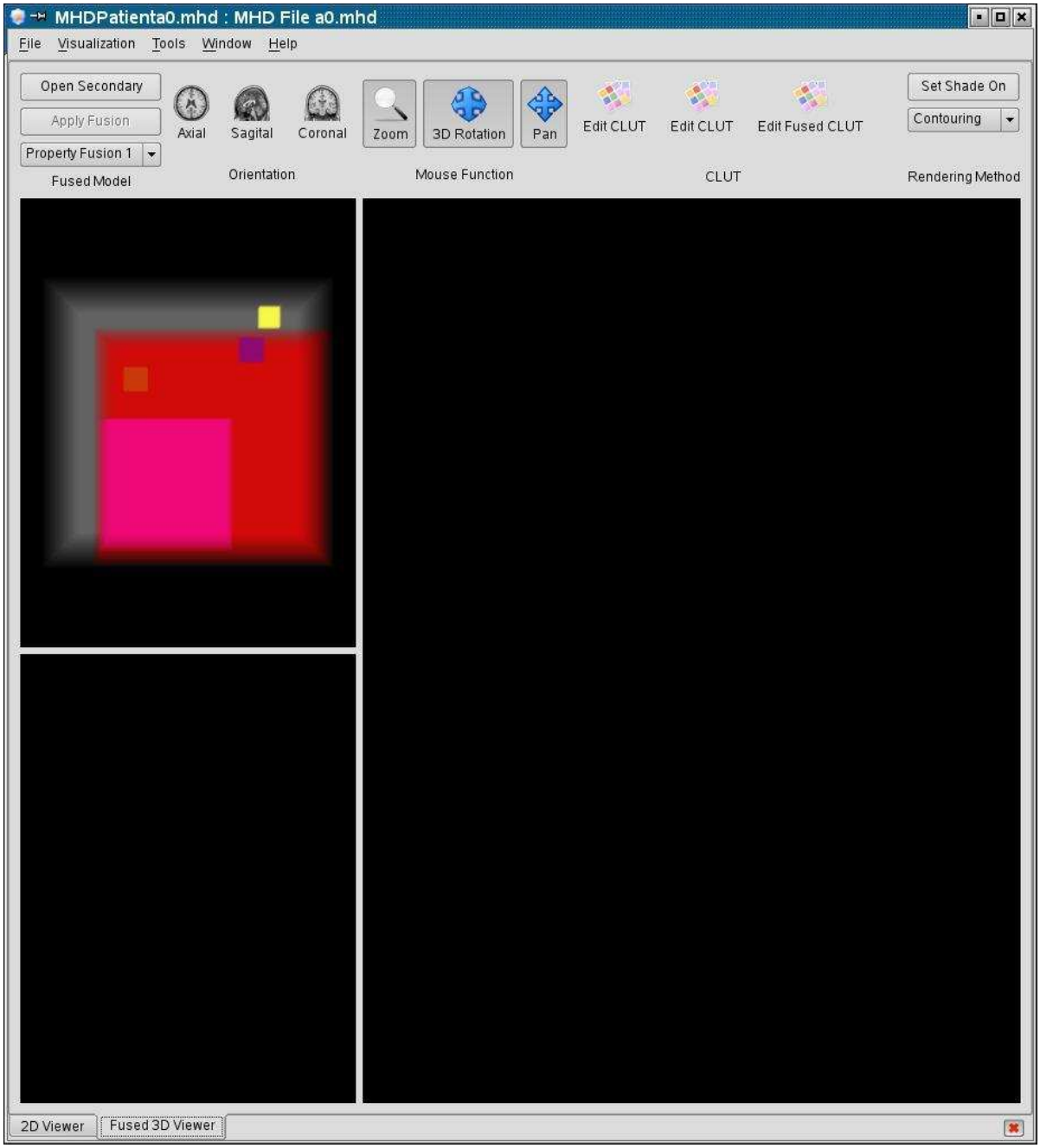


Figura 5.5: *Imatge inicial a l'obrir el mòdul*

5.3 Obrir model secundari

Per fer la visualització de models fusionats necessitem com a mínim dos models de dades, que en aquest cas seran dos objectes volume. El volum principal ja el tenim només obrir el nostre mòdul. Per tant haurem de repetir el procés d'obertura que es fa servir a la finestra principal d'starviewer. Per fer-ho necessitem guardar un segon objecte volum a la nostre aplicació per tal de poder extreure les dades per tractar-les.

S'ha escollit obrir el nou volum a partir d'un arxiu amb extensió .mhd ja que simplifica molt l'obtenció de dades a més de permetre treballar amb models sintètics.

Al tractar-se d'una aplicació gràfica d'usuari es busca que les accions siguin senzilles i fàcils de fer anar, per això obrirem un diàleg de fitxers(**Figura 5.6**) per seleccionar l'arxiu a obrir tal i com es fa a la plataforma.

Figura 5.6:*Dialeg de fitxer.*

Com en el cas anterior, on obríem el model principal, només caldrà escollir l'arxiu necessari i clicar el botó Open del diàleg. Automàticament es visualitzarà el segon model just a sota del model principal.

El diàleg per obrir el fitxer és un objecte de les llibreries Qt 4 anomenat QFileDialog. Permet moure's per les carpetes i fitxers del nostre ordinador i agafar la ruta en forma de QString(és una cadena de caràcters pròpia de les Qt). Per extreure'n les dades gràfiques del fitxer necessitem un objecte de les VTK nomenat vtkMetaImageReader. Aquest objecte l'assiignarem al nostre objecte Volume i posteriorment el visualitzarem a la nostra aplicació.

Si es vol guardar més d'un volum per realitzar la fusió de més dos models el que es pot fer, apart de buscar una manera d'organitzar la visualització de molts models simples, és fer servir un objecte del core de la platafroma Starviewer anomenat VolumeRepository. Aquest objecte l'únic que fa és contenir una llista d'objectes Volume on es poden afegir, extreure i borrar objectes. Una altra manera seria guardar una llista de Volumes al nostre mòdul, però no té sentit si ja ha sigut implementada aquesta opció.

A la **figura 5.7** es pot veure com obrir un el model secundari al mòdul Fused3DViewer

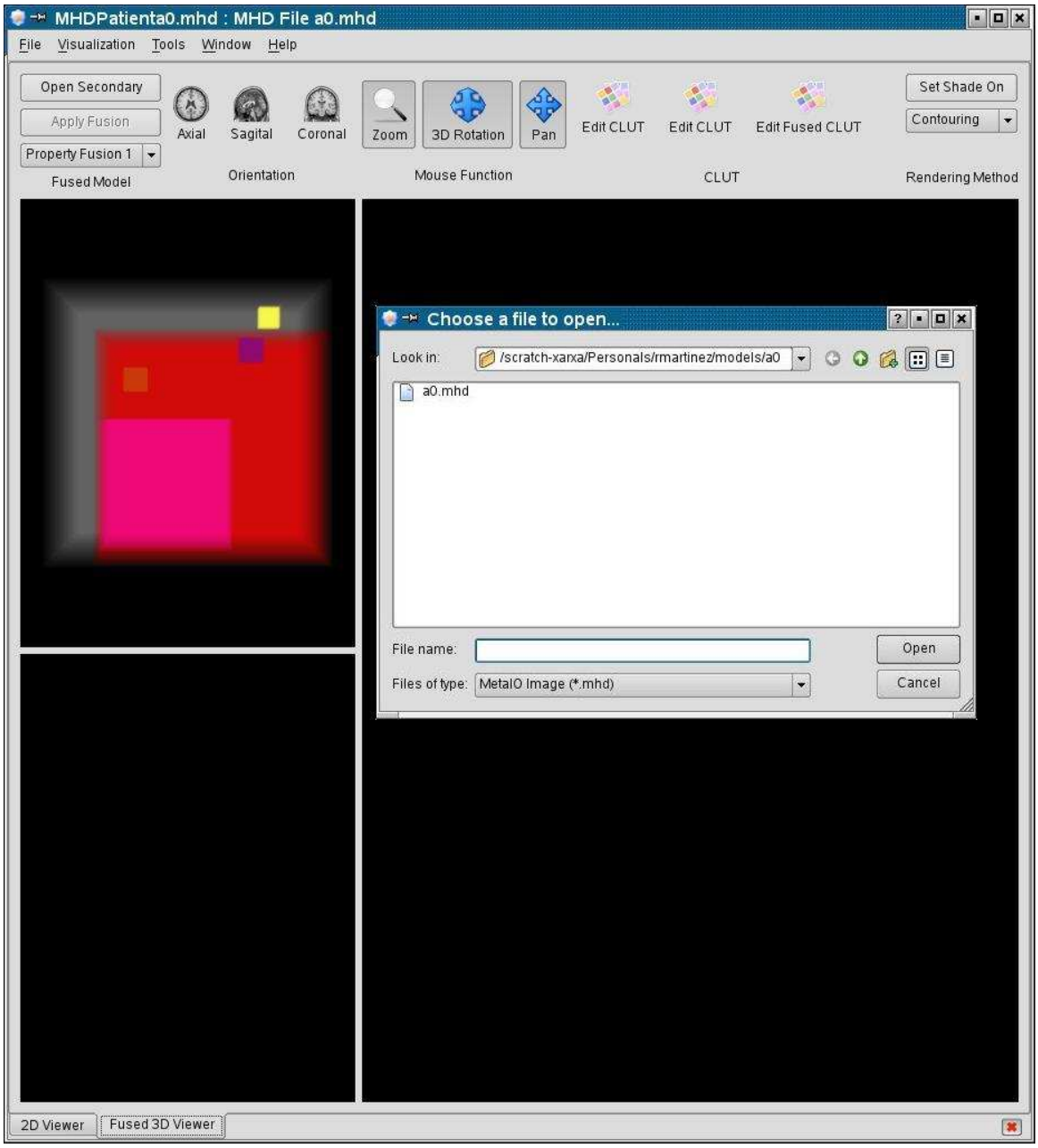
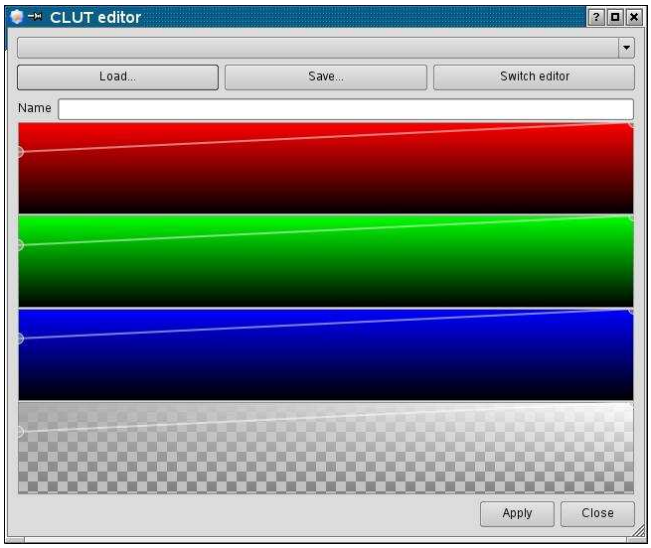


***Figura 5.7**: Diàleg per obrir el fitxer secundari.*

5.4 Assignar Funció de Transferència als models simples

Una de les parts més importants de la visualització en 3D de models de vòxels és aplicar una Funció de Transferència adient a les propietats de volum. Dins la plataforma es va crear una aplicació per assignar-ne una, però fins fa ben poc aquesta feina era molt feixuga ja que cada cop que es visualitzava un model s'havia de buscar la funció de transferència. Una de les ampliacions va ser afegir la possibilitat d'obrir i guardar les configuracions en un fitxer de text. Aquest objecte s'anomena CLUT Editor(Figura 5.8).



En el projecte s'ha afegit aquesta eina per assignar totes les funcions de transferència a tots els visualitzadors.

Figura 5.8: Editor de funció de transferència.

Es pot escollir entre dos tipus d'editor: editor gràfic, vist a la **figura 5.8**, o editor de valors, **figura 5.9.** L'editor gràfic està format per 4 barres de colors, una per cada color RGB i l'última per l'opacitat.

L'altre editor consta de diferents camps on podem especificar cada valor de propietat quin color i opacitat tindrà. S'escull el número exacte o el interval de valors de propietat. Després es selecciona el color en un diàleg que apareix en prémer el botó Select color. Per últim assignarem una opacitat variant el número que apareix en l'últim camp. El valor pot anar de 0, que es transparent, fins 256 que és totalment opac.

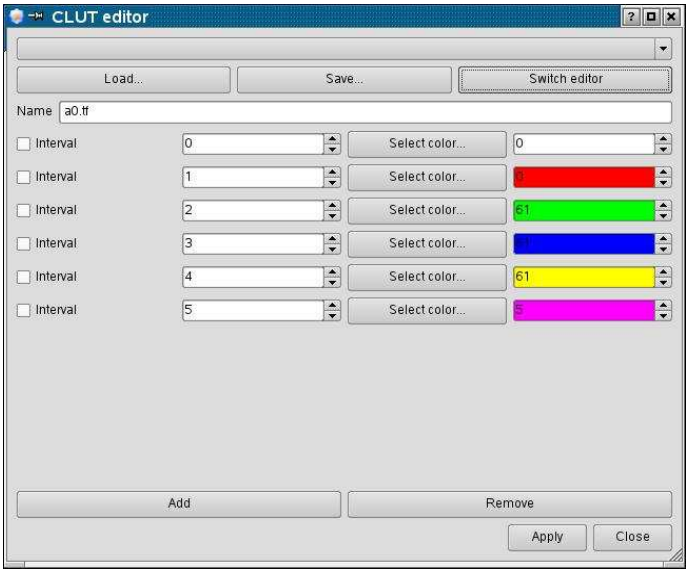


Figura 5.9: *Editor per valors*

En el projecte s'ha decidit ampliar el model que fa servir el visualitzador 3DViewer, que conté l'objecte CLUT Editor dins la classe principal del mòdul. En aquest cas es fan servir 3 CLUTEditor, un per cada visualitzador i tres funcions de transferència diferents. Una de les optimitzacions pot ser fer una ampliació de l'editor perquè permeti escollir a quin objecte se li assignarà la funció de transferència seleccionada mitjançant una llista e selecció.

A la **figura 5.10** es veu el botons afegits per obrir els editors. Un per cada visualitzador.

***Figura 5.10**: Icones de gestor de CLUT.*

Per aconseguir bons resultats de visualització és imprescindible fer servir la gestió de fitxers que contenen funcions de transferència. Aquests fitxers tenen extensió .tf tal i com es pot veure a la **figura 5.11.**

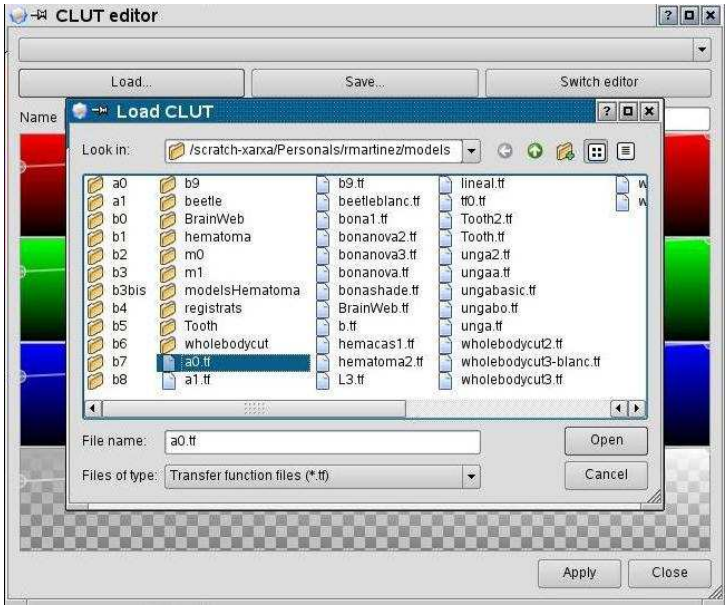


***Figura 5.11**: Eina de gestió de fitxers del CLUT*

Aplicant les dues funcions de transferència obtenim una nova imatge. Si la imatge resultant no és la desitjada sempre es pot canviar la funció de transferència, tants cops com es vulgui.

Trobar una funció de transferència adient a un model és una feina costosa i és molt probable que aquest procés s'hagi de repetir molts cops. De moment s'han d'assignat dues funcions de transferència a dos models però més tard, segons quina fusió es faci, caldrà una nova per visualitzar correctament el resultat.
A la **figura 5.12** es mostra un exemple d'assignació de funció de transferència fent servir l'editor.

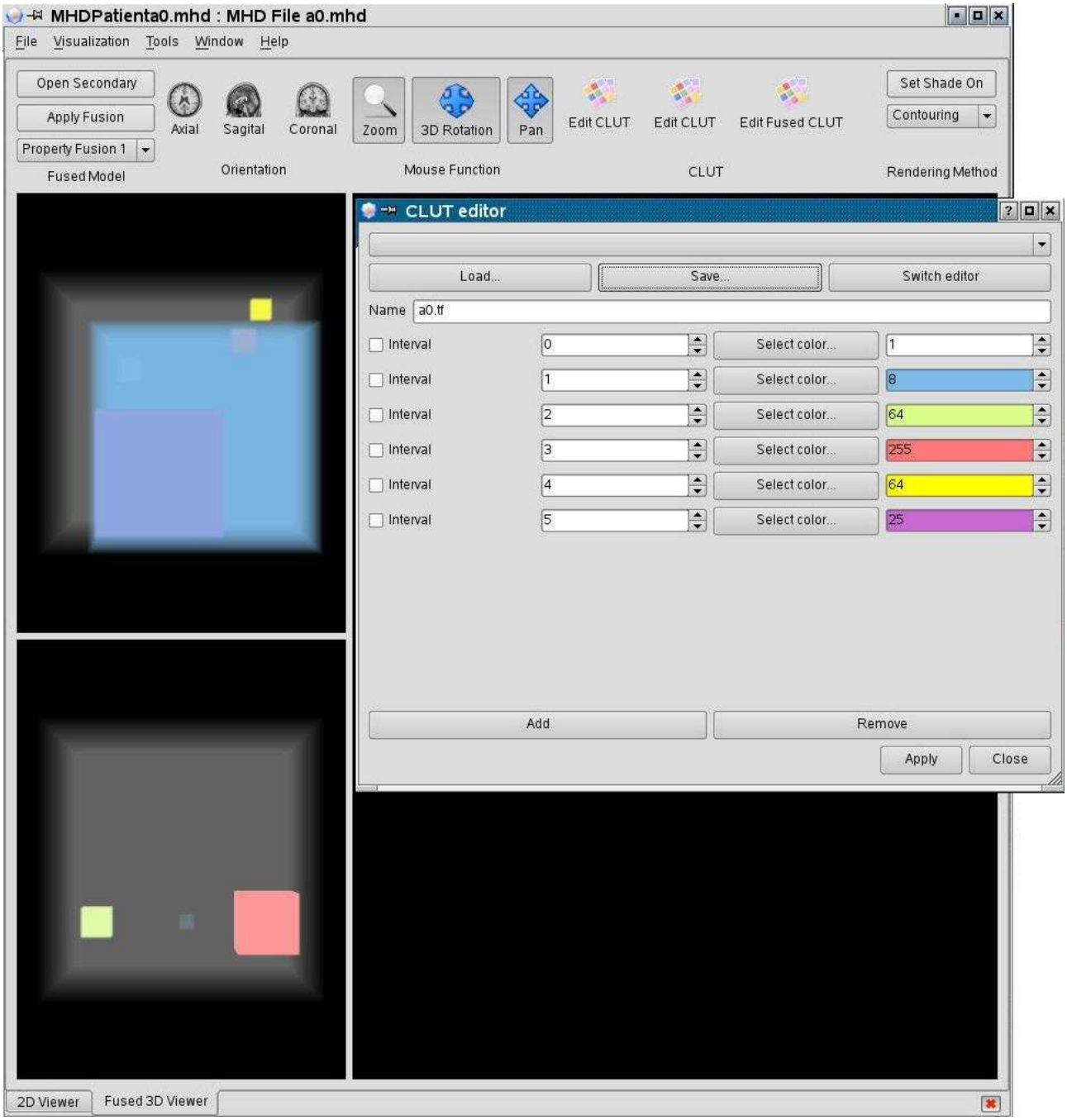


Figura 5.12: *S'assignen les funcions de transferència als dos models amb tonalitats pastel.*

Es pot observar la correspondència de colors dels models amb els assignats a cada valor de propietat. Cada cub té un valor de propietat diferent.

5.5 Canviar mètode de fusió

Una de les funcionalitats ha de ser permetre a l'usuari de l'aplicació canviar el mètode de fusió fent-ho d'una manera clara i senzilla. Per aconseguir-ho s'ha optat per fer una llista de selecció situada a la part superior de la finestra(**Figura 5.13**) amb els 6 mètodes de fusió. N'hi ha sis perquè per cada tipus de fusió s'han implementat dos maneres de calcular els colors.

Figura 5.13: *A la part esquerra de la interfície d'usuari trobem la llista amb les diferents opcions de fusió.*

Seguint les pautes d'implementació de l'Straviewer, les opcions estan en anglès i incorporen la possibilitat de ser traduïdes a altres idiomes fàcilment.

Les opcions són les següents:

- **Property Fusion 1:** Fusió de propietats fent servir interpolació
- **Property Fusion 2:** Fusió de propietats fent servir alternança de propietats
- **P&G Fusion 1:** Fusió de propietats i gradient fent servir interpolació
- **P&G Fusion 2:** Fusió de propietats i gradient fent servir alternança de propietats
- **Color Fusion :** Fusió de colors fent servir interpolació

El més important un cop seleccionat el mètode de fusió és dir-li a la nostra aplicació que quan l'usuari vulgui realitzar la fusió, faci la correcta.

La manera més senzilla de fer-ho és guardant un paràmetre que contingui el tipus. Aquest tipus pot ser un enter o una cadena de paràmetres. Hem de tenir en compte que al canviar la selecció de la llista es genera un esdeveniment.

Amb les Qt hem de connectar aquest esdeveniment d'un objecte amb una funció:

connect(nom_combo,SIGNAL(activated(int)), SLOT(changeFusionMethod(int)));

SIGNAL és el tipus de senyal emesa, i el paràmetre que conté l'esdeveniment és el tipus d'objecte que genera. SLOT és la funció que es crida i rebrà el paràmetre.

La funció changeFusionMethod(int) el que fa és assignar la variable que conté el tipus de fusió segon el paràmetre.

Én pseudocodi és:

Cas variable:
Opció 1
mètode=Fusió1
Opció 2
mètode=Fusió2
Opció 3
mètode=Fusió3

...
FCas

Lògicament s'ha de tenir en compte el paràmetre que especifica el tipus de fusió quan s'executi la fusió.

5.6 Aplicar Fusió

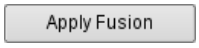
Aquesta funcionalitat encapsula els algoritmes més importants del projecte, els algoritmes que a partir de dos models simples, visualitzen la fusió resultant.

Abans d'entrar en detall, explicarem quins passos previs es segueixen abans de realitzar la fusió.

En l'apartat anterior hem explicat que succeeix al seleccionar un mètode de fusió. Es guarda una variable amb el tipus de fusió que es realitzarà. Quan l'usuari aplica la fusió, amb aquest valor és crida el mètode que s'ha d'executar. Hi ha una funció anomenada fuse() que realitza aquesta tasca. Segons el valor de la variable method crida un mètode o un altre. Com en l'apartat anterior, és una selecció:

Cas variable:

Opció 1
cridaMetodeFusió1()
Opció 2
cridaMetodeFusió2()
...
FCas

L'usuari executa la fusió mitjançant un botó dins el bloc de fusió. Lògicament aquest botó és de les llibreries Qt, concretament de tipus QToolButton 

Per realitzar la fusió de models hem decidit fer-ho fent servir el mètode: **OPS (One Property per Sample)**, és a dir, una propietat per volum. La selecció d'aquest mètode és deguda a la millor tolerància als canvis de càmera i llums i per que s'adapta millor a l'estructuració d'objectes implementats a la plataforma Starviewer. S'haurien de reestructurar objectes bàsics com el Volume o la Funció de transferència perquè suportés múltiples propietats per vòxel.

Tot seguit s'explicarà més concretament la implementació de cada tipus de fusió.

5.6.1 Property Fusion

La fusió de propietats com diu el nom consisteix en fusionar les propietats dels models de dades. Aquest tipus de fusió és prèvia a l'assignació de la funció de transferència, per tant es necessita un model nou de vòxel que contingui les propietats ja fusionades. Tal i com estan estructurades les llibreries i la plataforma, la millor opció és generar un nou volum partint de l'estructura d'un dels dos models, perquè contingui la mateixa estructura i mida; i després assignar-li les propietats calculades segons els models simples.

Per calcular cada propietat fusionada s'agafa el valor de propietat de cada model a fusionar i es fa un càlcul numèric. El mètode per fusionar els valors de propietat poden ser diversos. Es podrien fer fusions d'algunes zones.

Per aplicar aquesta idea al projecte s'ha hagut d'investigar quins passos i quins objectes són necessaris. Seguint l'esquema de la figura X on s'explica els objectes que intervenen en el procés de visualització veiem si es vol extreure les propietats del model abans d'aplicar la funció de transferència, hem d'observar les propietats del objectes vtkImageData i vtkVolumeMapper.

Tractant-se d'informació del model de dades, el més lògic és que sigui vtkImageData l'objecte que necessitem. Mirant la Class Reference de VTK trobem que cap dels mètodes de la classe ens retorna la informació que es vol. Seguint l'herència de classes es pot veure els mètodes dels objectes pare i per tant tots els mètodes possibles de l'objecte en qüestió.

Finalment es fa servir la següent instrucció per extreure les propietats:

```
vtkImage->GetPointData()->GetScalars()->GetVoidPointer( 0 );
```

De la imatge de les dades obtenim els punts del model en forma d'objecte vtkPointData. D'aquest n'agafem una còpia superficial de les dades i un cop tenim aquestes, necessitem un punter per accedir a l'estructura de vòxels. A més podem saber la mida de la llista de vòxels amb el mètode GetSize().

Mirant el diagrama de seqüència és veu més clar el procés i interacció d'elements que hi intervenen.

Aquest pas el repetirem per els 3 models de dades, els dos registrats i el model còpia creat per la fusió. Així fem un recorregut dels arrays per tal de calcular el valor de la nova propietat i assignar-ho al lloc de memòria corresponent al valor del nou model fusionat.

```
*punterfusionat=calculate(*punter1,*punter2)
```

Dins la funció calculate calculem el valor final de la propietat del vòxel. Aquest càlcul pot ser de moltes maneres diferents.

La **figura 5.14** mostra el diagrama de seqüència de la fusió de propietats
Amb aquest diagrama s'entén millor la interacció d'objectes que hi intervenen.

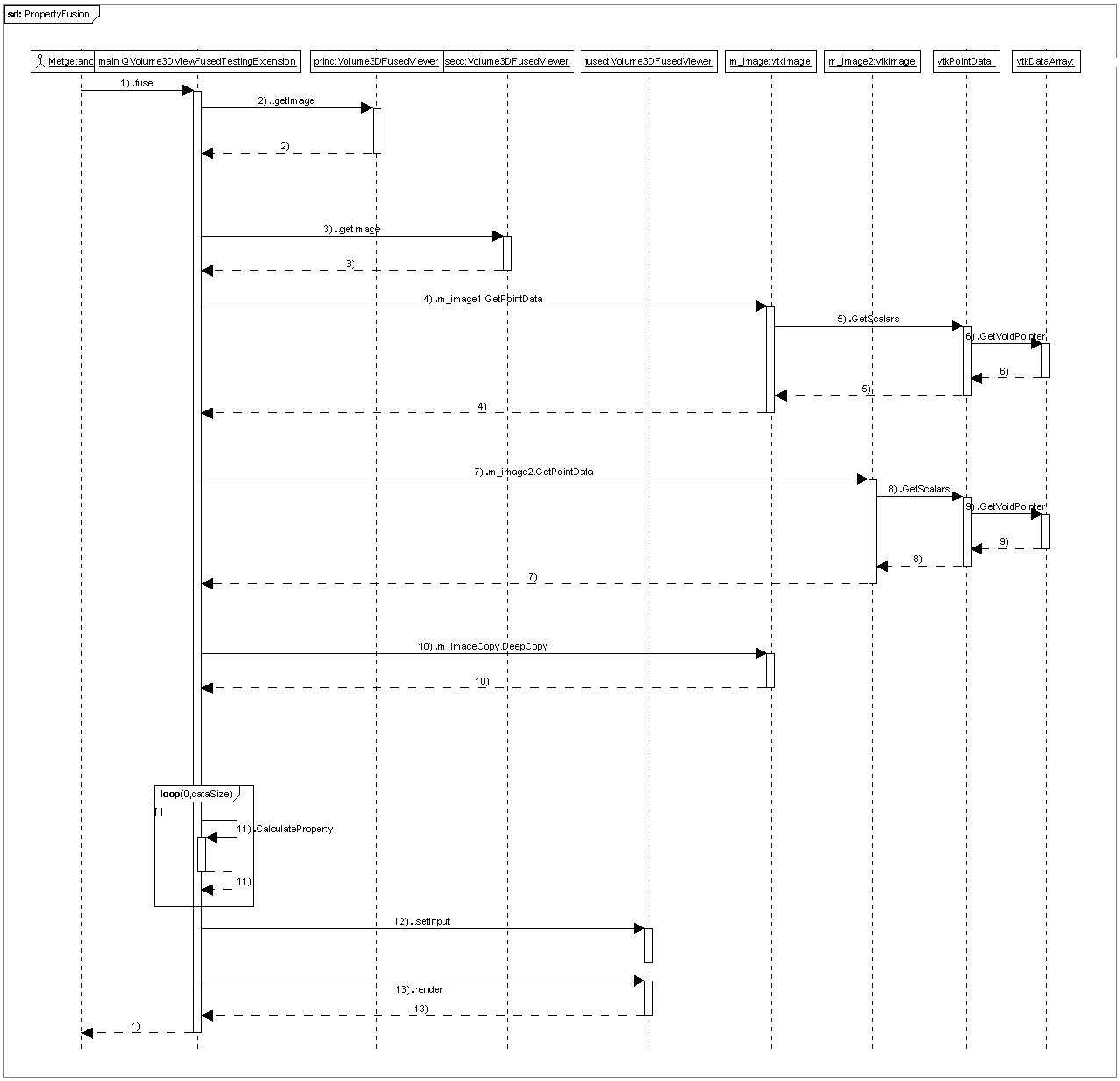


Figura 5.14: *Diagrama de seqüència de fusió de propietats.*

En aquest projecte s'han implementat dues maneres de càlcul:

1- Interpolació Lineal

Lògicament quan es té dos valors i és vol trobar un valor intermedi el primer que es fa és una mitjana. De fet, és una interpolació lineal entre dos valors.
El càlcul és senzill. Es sumen els dos valors i es divideix el resultat per 2.

$$Vpf = (Vp1 + Vp2)/2$$

L'algorisme seria el següent:

```
imatgeFusionada = imatge1->Copia()

mida1, mida2, midaFusionat,i,valor :enter
punter1, punter2, punter3: punter caràcter;
mida1 := imatge1->mida();
mida2 := imatge2->mida();
punter1 := imatge1->getPointer();
punter2 := imatge2->getPointer();
punterFusionat := imatgeFusionada->getPointer();
i := 0;
SI (mida1=mida2=mida3) Llavors

  MENTRE( i < mida1 )
        valor = (punter1->getValor() + punter2->getValor())/2
        punterFusionat->setValor(valor)
        punter1->següent();
        punter2->següent();
        punterFusionat->següent();
  FMENTRE
FISI
```

S'ha de tenir en compte que al interpolar els valors de propietat la funció de transferència variarà radicalment. L'avantatge d'aquest tipus de fusió és que la imatge creada té els objectes ben definits. L'inconvenient principal és trobar una funció de transferència adequada i que poden existir objectes que en un principi no haurien de ser del mateix teixit(valor de propietat) i al fer la interpolació resulten iguals.

Comparem els dos resultats fent una proba d'execució:

Primer de tot escollim quins models obrim. En aquest cas farem servir model sintètics per entendre bé com funciona la fusió. Agafem els model cub1.mhd i cub2.mhd tal i com es veu a la **figura 5.15**.

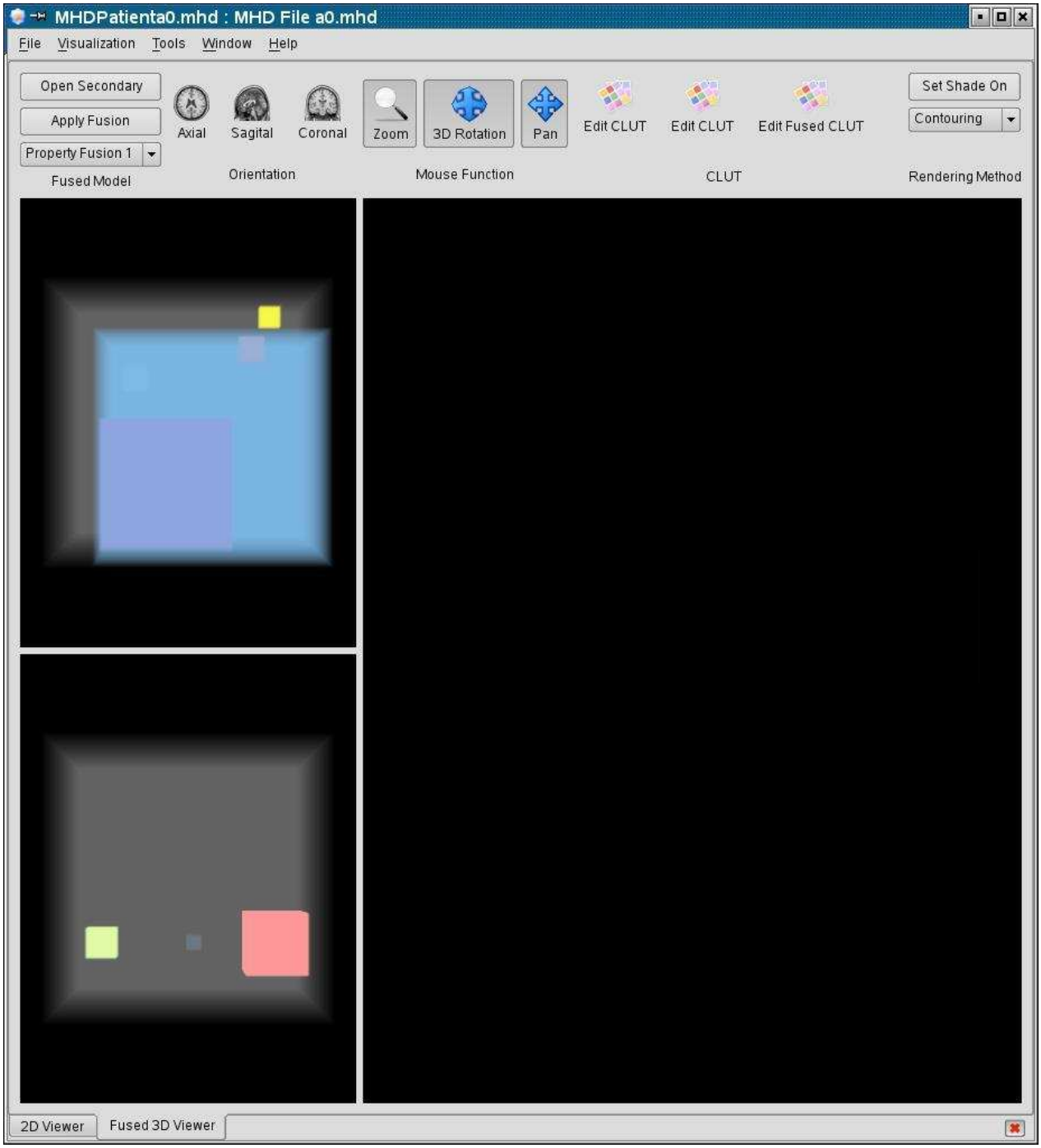


***Figura 5.15**: Obrim els dos volums i apliquem una funció de transferència.*

Cada model de dades té assignada una funció de transferència. Ens assegurem de tenir seleccionada l'opció de fusió desitjada. Per defecte està marcada la property fusion 1 així que si acabem d'obrir el mòdul no cal tocar res.

El següent pas és aplicar la funció fent clic al botó ApplyFusion. El resultat és mostra a la **figura 5.16**:

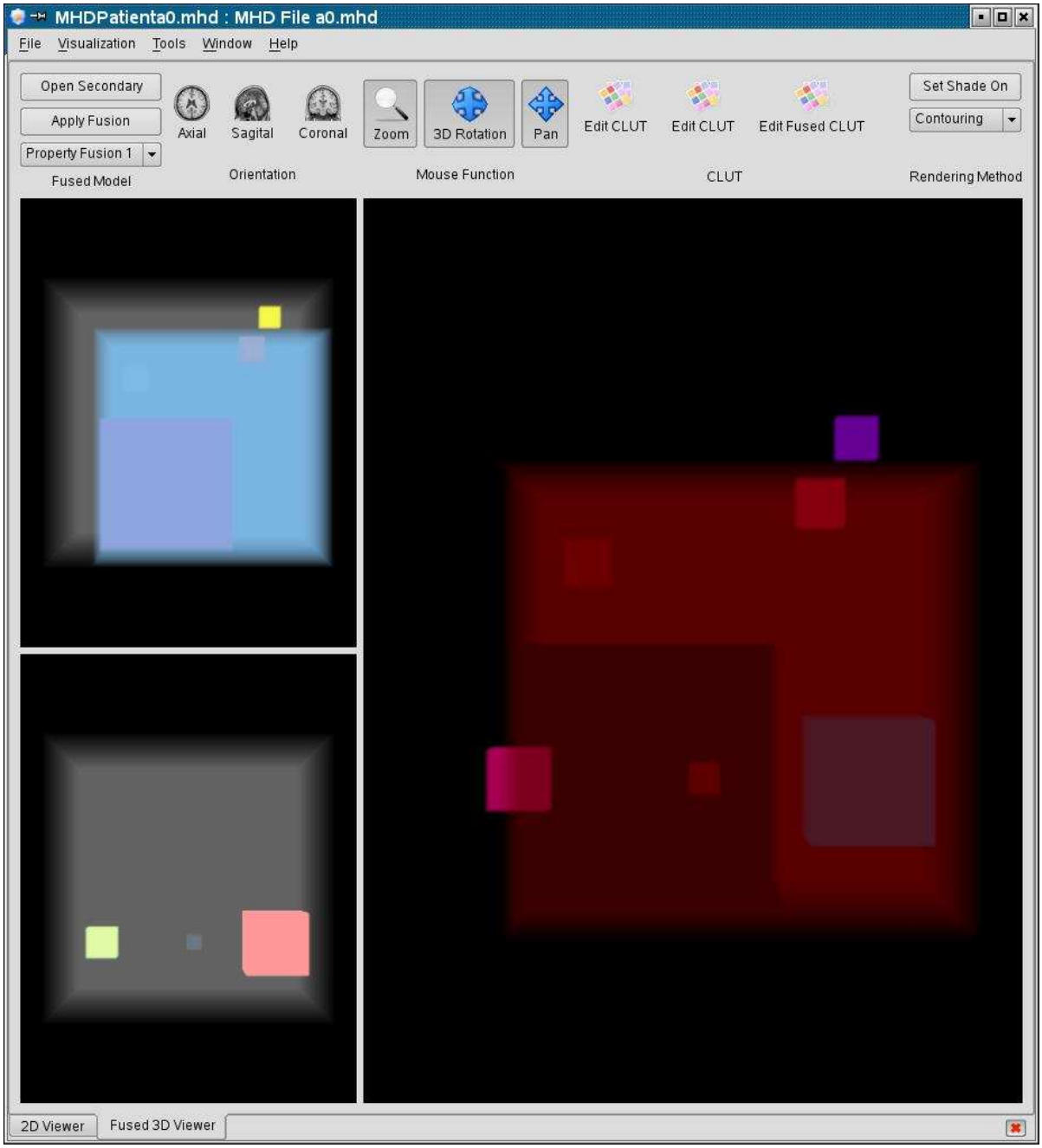


***Figura 5.16**: Un cop realitzada la fusió apareix el model fusionat al visualitzador de la dreta. Per defecte té assignada una funció de transferència, però cal canviar-la per millorar la visualització*

En pocs segons apareix una imatge al visualitzador de la dreta amb el model fusionat. Potser a simple vista no es pot veure que la fusió és correcta o no. En aquest cas s'intueix que el resultat és el que havia de sortir.

Igualment es recomana canviar la funció de transferència per optimitzar el resultat. Un cop canviada la funció de transferència del model fusionat obtenim un resultat molt més clar. A la **figura 5.17** es pot veure com cada objecte està on ha d'estar.

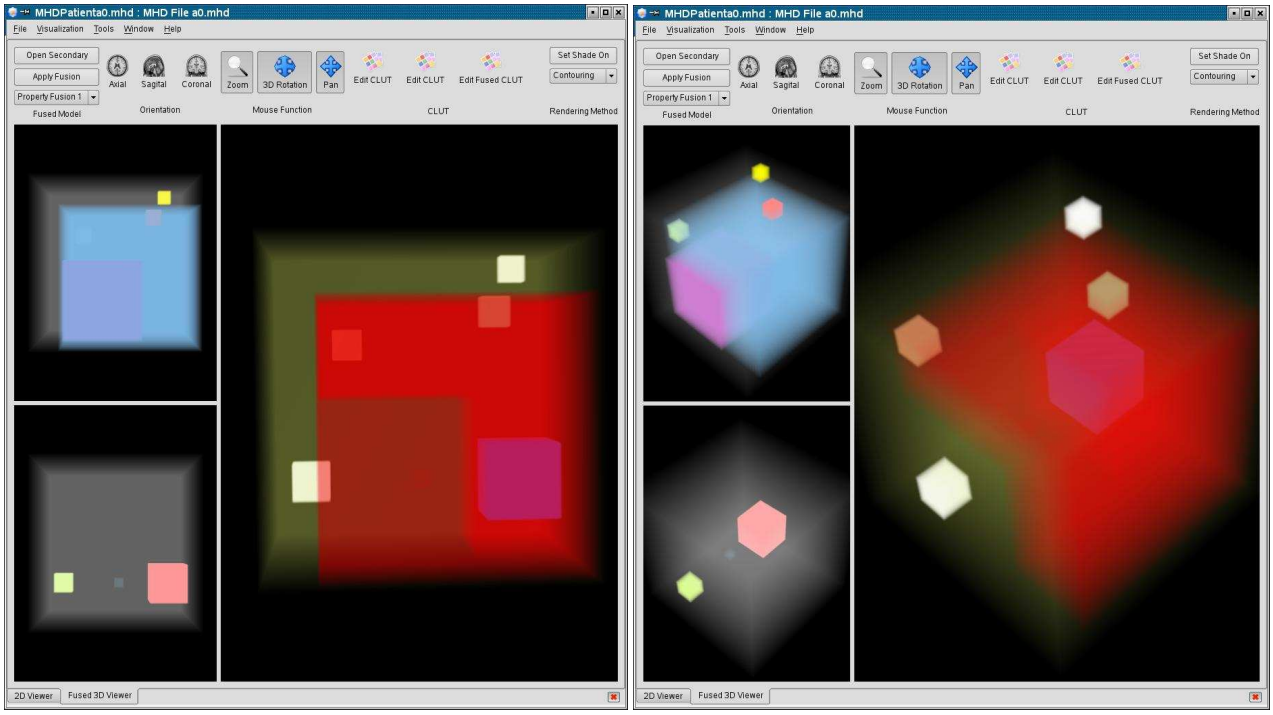


***Figura 5.17**: En questes dues representacions podem veure dos angles diferents del resultat i veure que efectivament és correcte*.

Observem que la fusió ha sigut correcta i la qualitat de la imatge és bona. Els objectes amb un color poc opac deixen passar la llum.

2-Alternància de propietats

L'altre mètode per calcular el valor de propietat fusionat que s'ha implementat és el d'alternar els valors de propietat. Això vol dir que si fem un recorregut per l'estructura de dades del model fusionat trobarem primer el valor de propietat del model 1 i el següent conté el valor del model 2.

Seguim aquest principi.

Vpf(i) =Vp1 -> Vpf (i+1)= Vp2

En el pseudocodi es veu més clar com funciona.

Cal una variable que determini si fer servir un o l'altre. Una manera molt simple d'implementar-ho seria:

```
valor=0;
MENTRE( i < mida1 )
  SI(valor==0)
        punterFusionat->setValor(punter1->getValor());
        valor=1;
  ALTRAMENT
        punterFusionat->setValor(punter2->getValor());
        valor=0;
  FISI
  punter1->següent();
  punter2->següent();
  punterFusionat->següent();
FMENTRE
```

Fem una execució amb els mateixos models i amb la mateixa funció de transferència que a l'apartat anterior per comparar el resultat.

Primer hem de seleccionar l'opció property fusion a la llista. Després clicarem el botó Apply fusion per realitzar la fusió.

A la **figura 5.18** podem veure el resultat de l'execució amb els mateixos objectes que a l'apartat anterior.

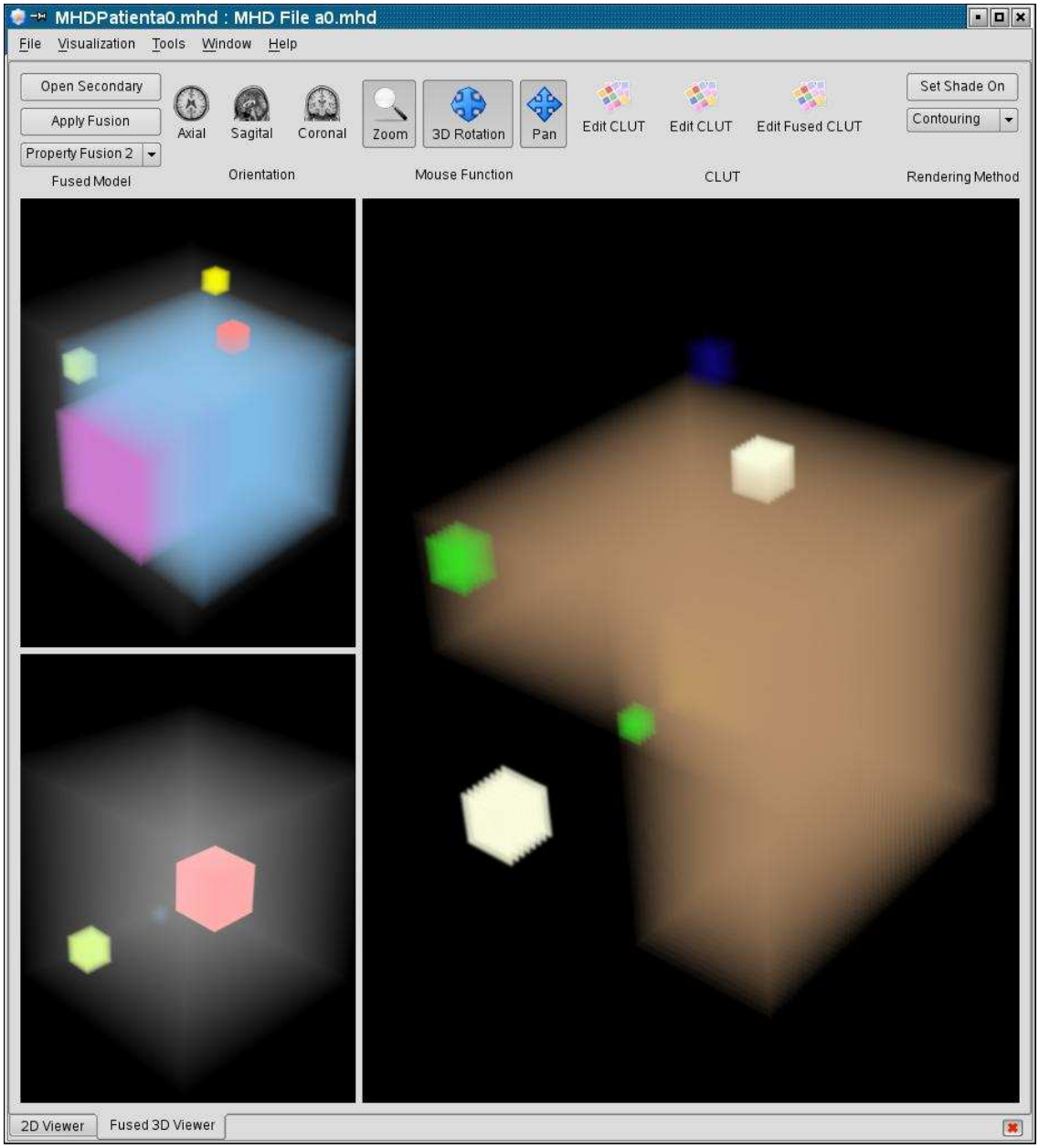


Figura 5.18: *Execució del Property Fusion 2.*

Si ens fixem en la qualitat de la imatge observem que els blocs tenen un aspecte laminat. Dona la sensació de que són plans un rere al altre quan no és així.

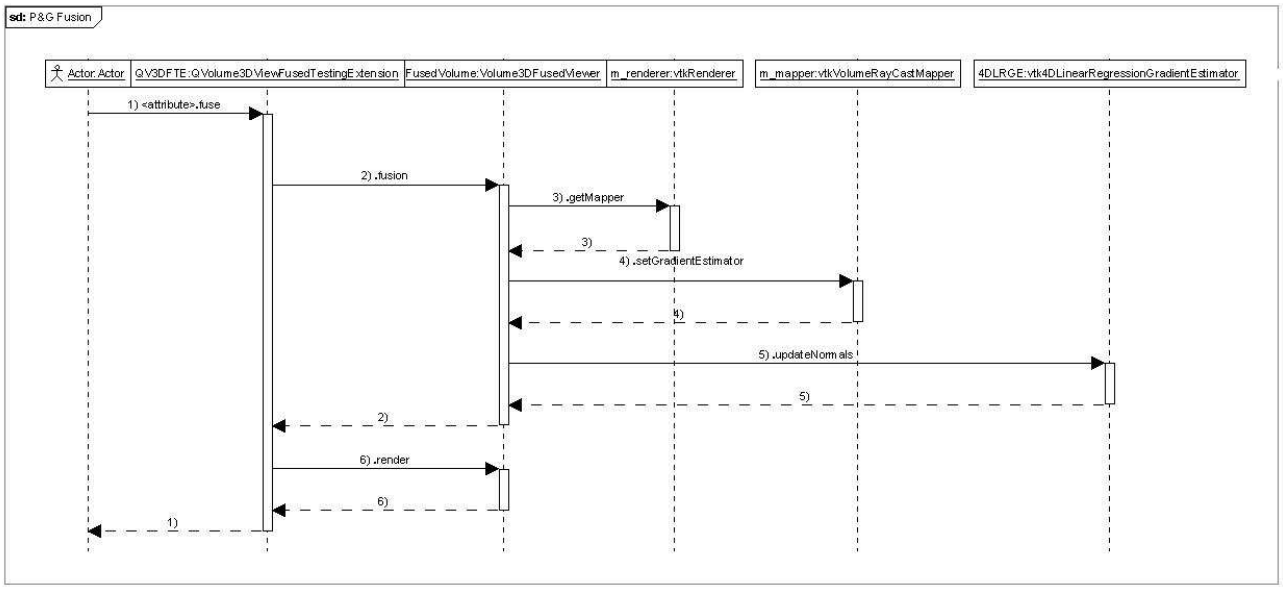
El resultat de cada un dels dos mètodes és molt diferenciat. Fent servir el primer algoritme és possible que apareguin objectes amb el mateix valor de propietat que en un principi no són iguals. Amb l'altre tipus de fusió no es té aquest problema, però la visualització és de menys qualitat ja que és com si es visualitzessin plans alternats, un de cada model.

5.6.2 Fusió de propietats i gradient:

Aquest tipus de fusió es basa en la fusió de propietats amb l'afegitó que incorpora el processat del gradient.

Així doncs és com una ampliació del mètode de fusió de propietats on s'aplica el càlcul dels gradients del model de dades. Per això un cop feta la fusió de propietats es crida el mètode que fusiona el gradient. En aquest cas es necessita accedir a les dades que conté l'objecte Volume3DFusedViewer com són el vtkMapper, el vtkRender, la funció de RayCasting i les propietats del volum.

Es fa servir un objecte que usa la regressió lineal en 4 dimensions per calcular el gradient. Per realitzar aquesta estimació es fa servir l'objecte vtk4DLinearRegressionGradientEstimator. Aquest objecte ha estat desenvolupat per un desenvolupador de la plataforma Starviewer i s'ha decidit incloure-ho perquè aporta més realisme a la imatge final i no comporta un increment de càlcul apreciable respecte a l'estimador que incorpora la llibreria vtk (vtkFiniteDifferenceGradientEstimator).
En el diagrama de seqüència s'entén millor el funcionament i interacció entre objectes.



La figura 5.19 *El pas previ abans de cridar el fusion que es veu en el diagrama de seqüència d'aquest mètode de fusió és idèntic al de la fusió de propietats que s'ha explicat a la* ***figura 5.14****. L'única diferència és que no es visualitza en acabar l'execució sinó que es continua amb la fusió de gradient.*

Per demostrar el seu funcionament farem una execució fent servir dos models sintètics: fig1.mhd i fig2.mhd.

El procés per obrir l'arxiu és el mateix que per tots els arxius. L'únic que hem de canviar és el mètode de fusió i posar-lo a P&G Fusion 1.

En la figura 5.20 es pot veure un exemple d'execució on es pot apreciar les diferents millores en la visualització i alhora com es realitza correctament la fusió.

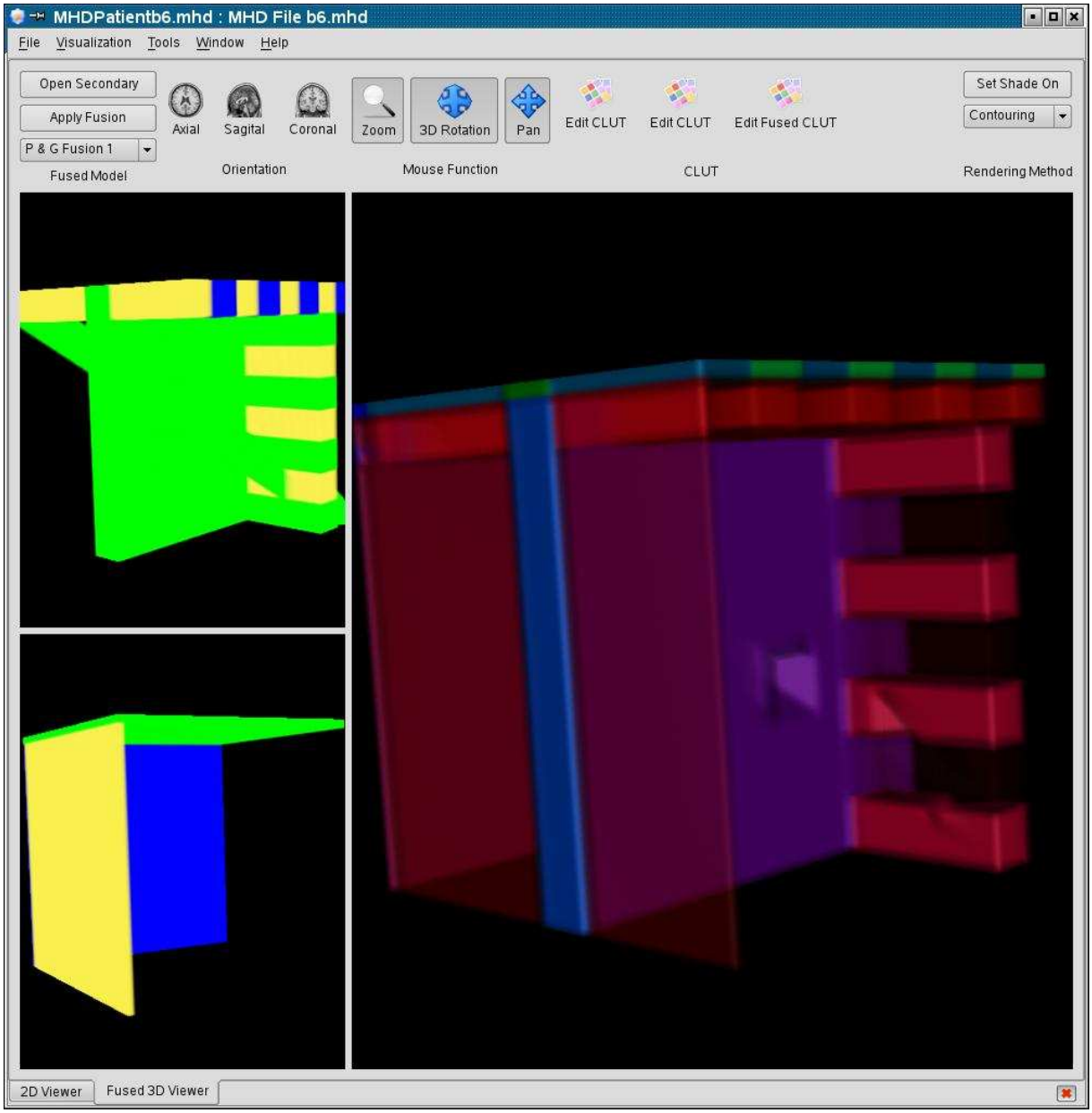


Figura 5.20: *Fusió de dos materials sintètics fent servit P&G Fusion 1.*

A simple vista es pot veure que la qualitat de la imatge generada és millor que la generada anteriorment amb l'anterior tipus de fusió el property fusion. A més veiem que el resultat és correcte. Seguint l'estructura del model fusionat podem veure els altres dos a dins.

Com en el property fusion, tenim una altra manera de calcular les propietats, intercalant-les. Canviem el mètode de fusió per P&G Fusion i apliquem la fusió.

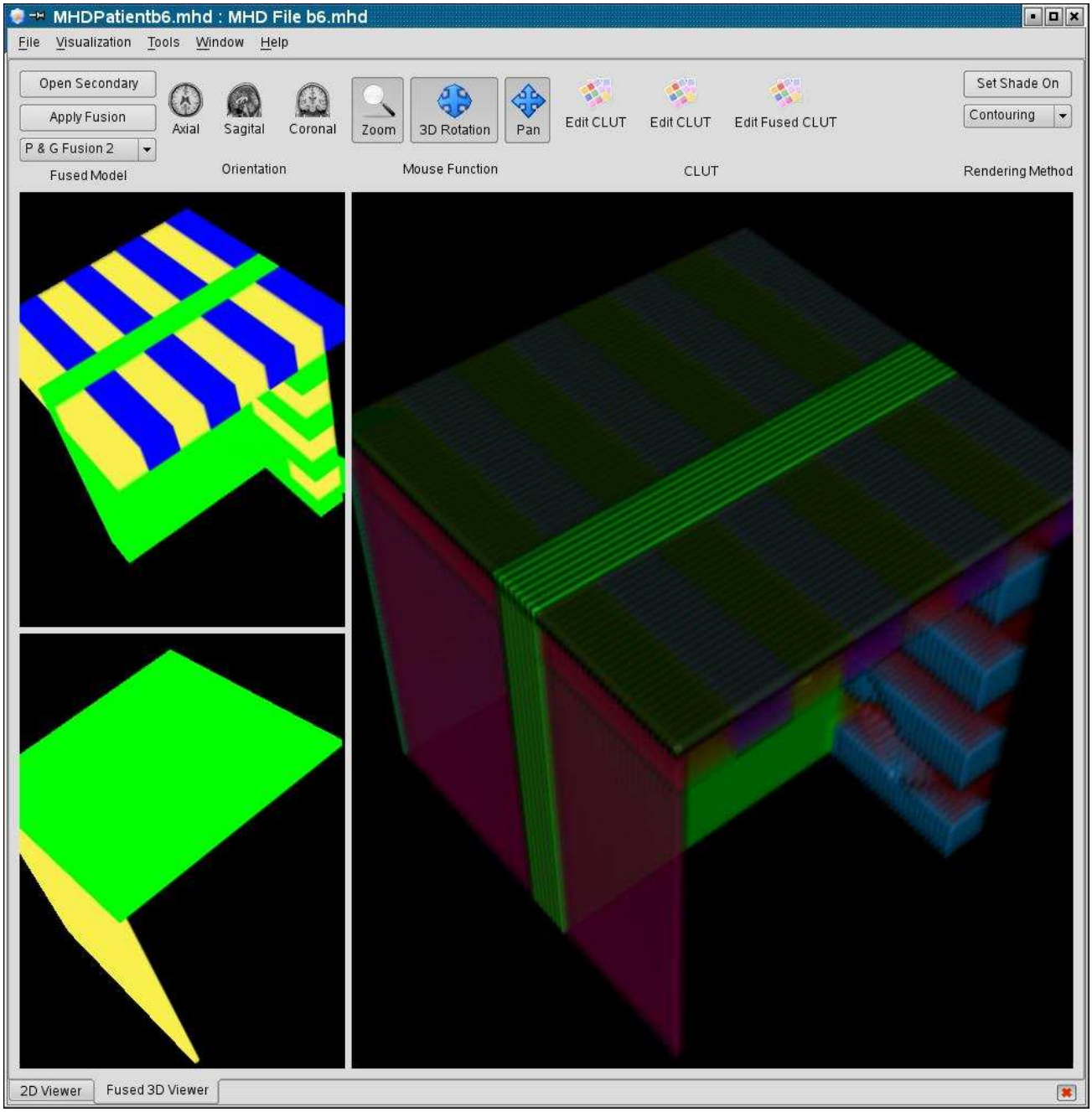


Figura 5.21: *Fusió de dos materials sintètics fent servir P&G Fusion 2.*

A simple vista es pot observar que, com quan aplicàvem intercalació en la fusió de propietats, dona una sensació de discontinuïtat que resta qualitat a la imatge. A més, com que es tenen en compte els gradients que detecten les fronteres, aquestes queden més remarcades. Tot i així es pot observar que la fusió és correcta.

5.6.3 Fusió de Colors

El tercer mètode implementat en el projecte és el de la fusió de colors. La fusió de colors té en compte el color de la imatge visualitzada de cada model i genera una imatge amb els colors i per tant la forma dels dos models en un.

Aquest algorisme és posterior a l'assignació de la funció de transferència, és a dir, un cop assignem la FT extraiem els colors finals del volum a visualitzar i fem els càlculs pertinents per realitzar la fusió.

La peculiaritat principal d'aquesta tècnica és que la fusió es fa en temps de visualització. Per tant podem extreure les propietats RGBa dels volums que volem fusionar, el problema és que en el moment de visualització el que preval és la funció de transferència. En un principi es va intentar fusionar els colors RGBa dins els volums un cop assignada una funció de transferència però es va comprovar que, tot i fer la fusió, al visualitzar, el resultat era idèntic al de la imatge model.

Llavors la solució és interferir en el procés de visualització i fer la fusió sabent els colors finals que tindran els dos models i crear una imatge nova amb aquests colors fusionats. Per aconseguir aquesta fusió cal fer servir una variació d'un tipus d'objecte anomenat voxelshader que el que fa és calcular els colors en temps de visualització. Fent que tingui accés a les dues funcions de transferència i a les dades de les imatges a fusionar podem fusionar els colors finals. Aquest voxelshader s'inclou en la funció de RayCasting que s'aplica al model.

La seqüència comença assignant als voxelShader les propietats de cada model. Aquestes propietats són l'estructura dels models. També cal assignar la funció de transferència que té cada model per tal d'agafar els colors. Després s'ha d'afegir el voxelshader a la funció de raycasting que a la vegada s'inclou en el mapper de visualització del volum. Finalment aquest mapejador s'inclou a l'objecte visualitzador.

Observant el diagrama de seqüència **5.22** s'entén molt millor com funciona l'algorisme.

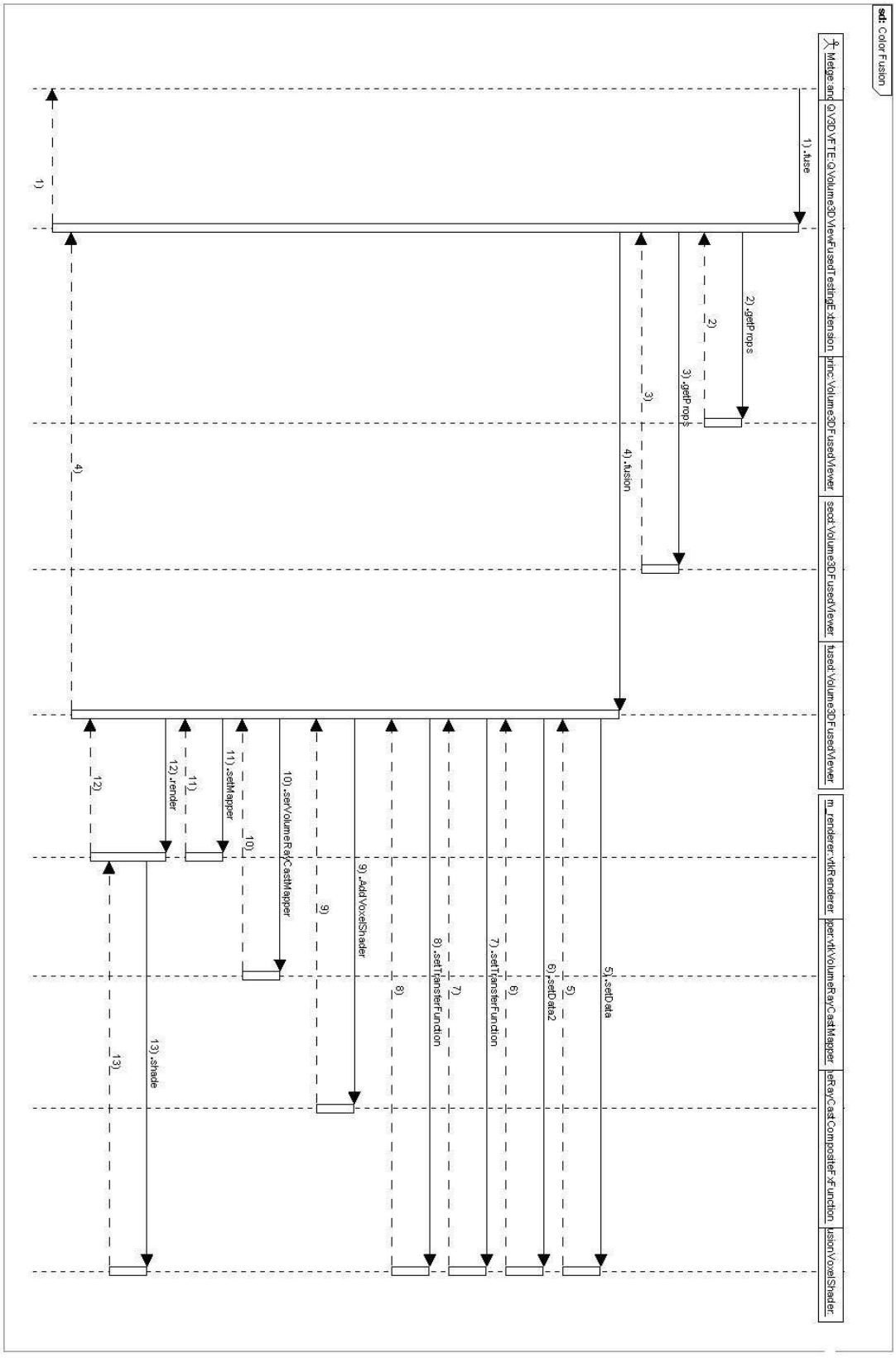


Figura 5.22: *Diagrama de seqüència de la fusió de colors.*

Per realitzar una fusió de colors seguim el procediment explicat anteriorment d'obrir els dos volums, però sobretot s'han d'assignar funcions de transferència als dos models, ja que aquestes funcions de transferència són vitals per fer la fusió. També s'ha de marcar a la llista de selecció l'opció Color Fusion. En aquest cas només es fa l interpolació de colors ja que si agaféssim un color de cada model intercaladament, el resultat no aportaria informació correcta.

A la **figura 5.23** podem observar el resultat de la fusió de colors. S'han escollit dos models sintètics bàsics per poder veure el funcionament.

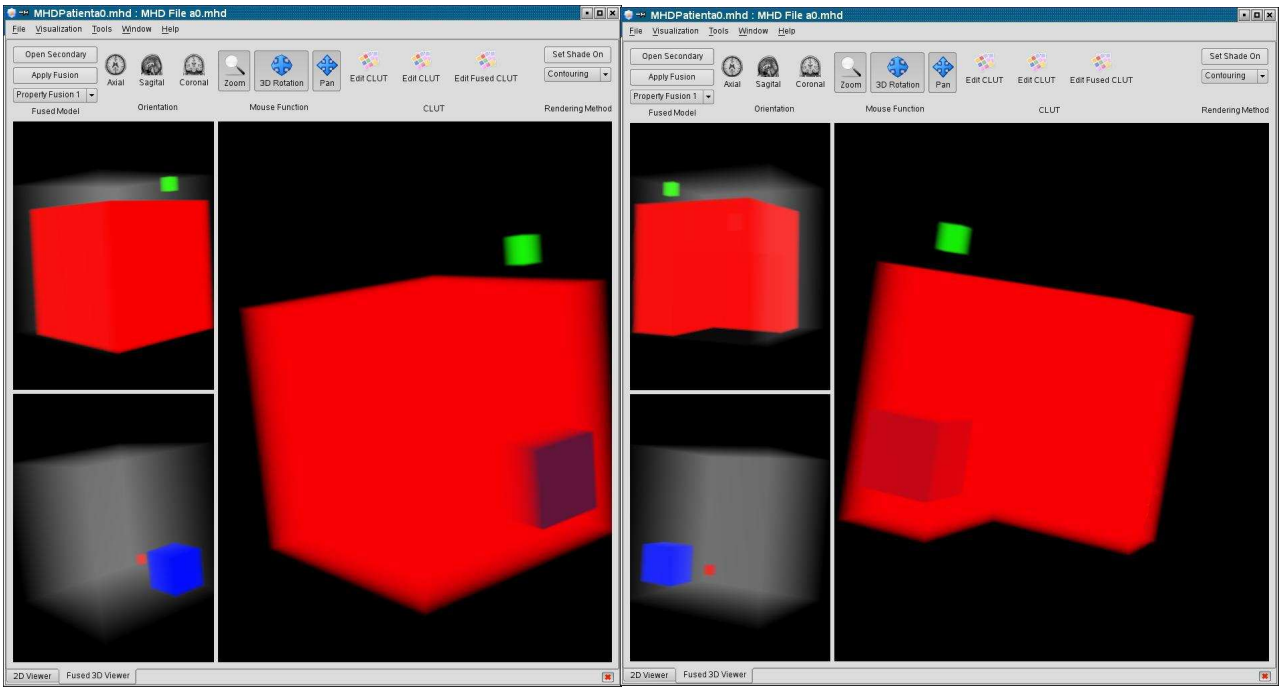


Figura 5.23: *Resultats al fusionar dos models simples fent servir la fusió de colors.*

5.7 Interaccionar amb el visualitzador

Aquesta és una funcionalitat compartida per tots el mètodes que contenen una visualització en 3D. Consisteix en dotar el nostre visualitzador d'eines per interactuar amb l'escena. Aquestes eines estan incloses en el core de la plataforma per tal de que tot visualitzador en 3D les pugui fer servir.

L'eina bàsica és la de modificació dels paràmetres de la càmera. Aquest paràmetres són:

- **Posició**

  Canviant les coordenades de posició de la càmera es pot aconseguir l'efecte de moviment de l'objecte. De fet el que es mou és la càmera respecte a l'objecte i no al contrari.

- **Zoom**

  Potser una variant del canvi de posició. Si ens apropem a l'objecte, dona la sensació que es fa més gran. Si ens allunyem, semblarà que es fa petit.

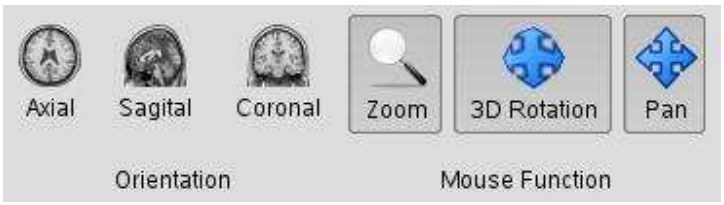
- **Rotació**

  Si fem rotar la càmera en cercle mantenint la distància al objecte, obtindrem un efecte de rotació.

- Orientació

  Assigna una posició al model. Aquestes posicions es fan servir en l'entorn mèdic. Són l'axial, la sagital i la coronal.

Aquestes eines s'inclouen en l'aplicació només engegar el mòdul Fused3DViewer. Per defecte estan activades, però es poden desactivar fent clic a les icones del grup Mouse Function. S'anomenen funcions de ratolí perquè amb el ratolí es controla el seu funcionament.



- La posició es controla fent servir el botó del mig del ratolí
- El zoom es pot fer servir amb la rodeta del ratolí o arrossegant amb el botó esquerre premut.
- La rotació es fa movent el ratolí amb el botó dret premut.

6. AVALUACIÓ DE RESULTATS

6.1 Entorn de proves

Les proves i execucions de l'aplicació s'han fer sobre un ordinador amb les següents característiques:

- Processador Intel Pentium P4 CPU 3.00Hz.
- 1524 MB DDR RAM
- Targeta Gràfica NVIDIA GeForce 6800 GT 256 MB
- Sistema Operatiu GNU/Linux OpenSuse 10
- Kernel del sistema 2.6.18.2-34-default
- X.Org versió 11
- Qt 4.2.1
- VTK 5.0.0

6.2 Anàlisi de resultats

Fusió de propietats:

Observant les captures de pantalla de l'apartat d'implementació podem comparar les diferències de fusió de propietats fent interpolació lineal o alternant propietats entre els models simples. A simple vista s'observa que l'aspecte general de la visualització és de més qualitat aplicant property fusion 1.

L'avantatja principal de la fusió alternant propietats és que es veuen tots els objectes de l'escena. No existeix el problema de trobar errors amb propietats iguals alhora de fusionar i a més és més fàcil identificar tots els objectes que haurien de sortir a l'escena.

Pel que fa a temps d'execució els dos algoritmes triguem pràcticament el mateix temps, ja que l'increment en cost del càlcul de la mitjana és inapreciable.

La feina més difícil i costosa és trobar una funció de transferència adient a cada model. Si els models són sintètics no és molt complicat ja que normalment són uniformes en la forma els valors de propietat s'han assignat forçadament, és a dir, es coneixen. Això facilita la definició de la funció, però un cop feta la fusió aquest valors canvien de rang i a més podem trobar objectes en un principi diferents amb valors finals iguals.

Fusió de gradient i propietats:

El comportament d'aquesta fusió és semblant al de la fusió de propietats. Estudiant les execucions es veu a simple vista que les imatges generades són de millor qualitat. Això és degut a l'ús del càlcul del gradient. El que s'ha observat és que fent servir el mètode d'intercalar valors no s'obté una imatge ben definida i potser no es recomana una utilització per l'àmbit mèdic.

L'altre mètode sí que retorna els resultats esperats abans de fer la implementació. Pot ser molt útil per una simulació de models registrats ja sigui per estudi d'alguna malaltia o per la representació gràfica d'algun estudi mèdic.

Els temps d'execució sense ser excessius, són molt més alts que els del mètode de fusió degut al càlcul dels gradients que és pràcticament instantani.

Fusió de colors:

Pel que fa als resultats obtinguts de fusió de colors podem observar una bona qualitat d'imatge del model fusionat. La gran avantatge d'aquest mètode és que si tenim ben definides les funcions de transferència dels model a fusionar, resultarà força senzill interpretar els resultats. El problema és que si amb la suma de colors, alguna zona no queda ben definida s'ha de canviar la unció de transferència d'algun dels dos models simples.

Amb la fusió de colors el temps d'execució canvia radicalment. És molt més costós en temps ja que s'ha de fer un recorregut cúbic del model de dades. Això te cost cúbic ja que són 3 iteracions, una per cada dimensió. L'avantatge és que tot i tenir dos models, el recorregut el fem servir pels dos alhora, l'únic que es fa és agafar les propietats dels dos models.

7. MILLORES I AMPLIACIONS:

En aquest projecte s'han implementat i estudiat totes les tècniques i mètodes necessaris per assolir els objectius mínims prefixats. Tot i això cal reconèixer que moltes d'aquestes tècniques podrien ser millorades o fins i tot ampliades.
En aquest capítol resumirem quines podrien ser aquestes millores i com es podrien aplicar.

Permetre fusionar n models

Una de les opcions interessants seria dissenyar l'aplicació per permetre la fusió d'un número variable de models. S'hauria d'enregistrar la informació de cada model creant una llista de model, en aquest cas objectes Volume. Starviewer ja incorpora aquesta llista de volums. Només s'hauria de canviar alguna funció implementada perquè suporti les llistes.

Treballar amb model registrats reals

En principi aquest projecte està destinat a ser usat en un futur dins la plataforma Starviewer que està destinada a un àmbit mèdic professional. Tal i com està construït el mòdul, es poden visualitzar dades mèdiques. L'únic problema és que encara no es disposa d'una eina eficient per registrar models. S'han provat dissenys preliminars i es visualitzen correctament, però no tenen suficient qualitat per ser tractats.

Millorar la interfície d'usuari

L'objectiu primordial ha sigut aconseguir la visualització de models fusionats tal i com s'ha explicat als objectius, i el disseny de la interfície no s'ha cuidat tant com si fos una aplicació final per a un client o empresa. Com que en un futur aquest mòdul ha de formar part de la plataforma Starviewer, és una de les parts que segur s'haurà de millorar. La solució recau més en el disseny que en la implementació tot i que alguna ampliació, com podria ser que sortíssim *tooltips* d'ajuda o dreceres de teclat per agilitzar l' interacció amb l'usuari.

Traduir l'aplicació a altres idiomes

El mòdul d'aplicació ha sigut fet tot en anglès tant i com marquen els estàndards de programació de la plataforma Starviewer. Tot i això es contempla la possibilitat d'escollir el idioma en que es mostrarà tots els textos. Per fer aquestes traduccions existeix una aplicació anomenada **Qt Linguist** per inserir les traduccions dels textos.

Poder realitzar captures de pantalla

Al tractar-se d'una aplicació mèdica, és molt probable que l'usuari final, és a dir, el metge o especialista desitgi guardar el resultat de la visualització.

Millora rendiment de l'aplicació

En general els algoritmes implementats en aquest projecte tenen un temps de resposta relativament ràpid, tot i que la fusió de colors té un temps de resposta més lent. Una possible millora seria incorporar algun procés de càlcul dins de la GPU(Grafic Processor Unit) per tal de no tenir ocupada la CPU. A més aquest processadors són específics de càlcul. El problema recau en que les llibreries fetes servir en la plataforma no contemplen aquesta possibilitat. Una adaptació a aquesta funcionalitat podria portar moltíssima feina i inclús reestructuració global de la part de visualització 3D de l'aplicació.

Optimització de codi.

Tal i com especifica l'estàndard de metodologia de programació fet servir en aquest projecte, l'eXtrem Programming, un cop assolits els objectius de l'aplicació, una de les coses que es pot millorar és la qualitat i l'enteniment del codi font. Com que en un futur el mòdul potser inclòs en la plataforma com a mòdul del core, és necessari que el seu codi sigui entenedor i correcte ja que molt probablement altres desenvolupadors facin servir o hagin de modificar alguna funcionalitat.

Afegir efectes d'obscurances en la visualització.

Avui en dia existeixen tècniques de visualització avançada que es podrien aplicar a la totes les visualitzacions de models del projecte. Si més no, per no fer l'aplicació massa lenta en càlcul, aplicar aquests algoritmes sobre la imatge fusionada final. Les vtk incorporen alguna tècnica com la d'aplicar shaders, però per solucionar aquest problema s'hauria d'interferir en el procés de rendering. Mitjançant voxel shaders es podria solucionar aquest apartat, tot i que, com hem dit abans, ralentitzaria molt la visualització.

Afegir llums a l'escena.

Una altra possible millora interessant seria afegir més llums a l'escena per aconseguir remarcar alguna zona del model 3D o simplement per afegir més realisme. En aquest projecte s'han fet testos amb les llums de les vtk, les vtkImage. Amb models superficials el funcionament és correcte però els resultats amb els models de vòxels no han sigut profitosos. Existeix un altre objecte anomenat vtkLightKit que també s'ha testejat amb els mateixos resultats.

8. CONCLUSIONS

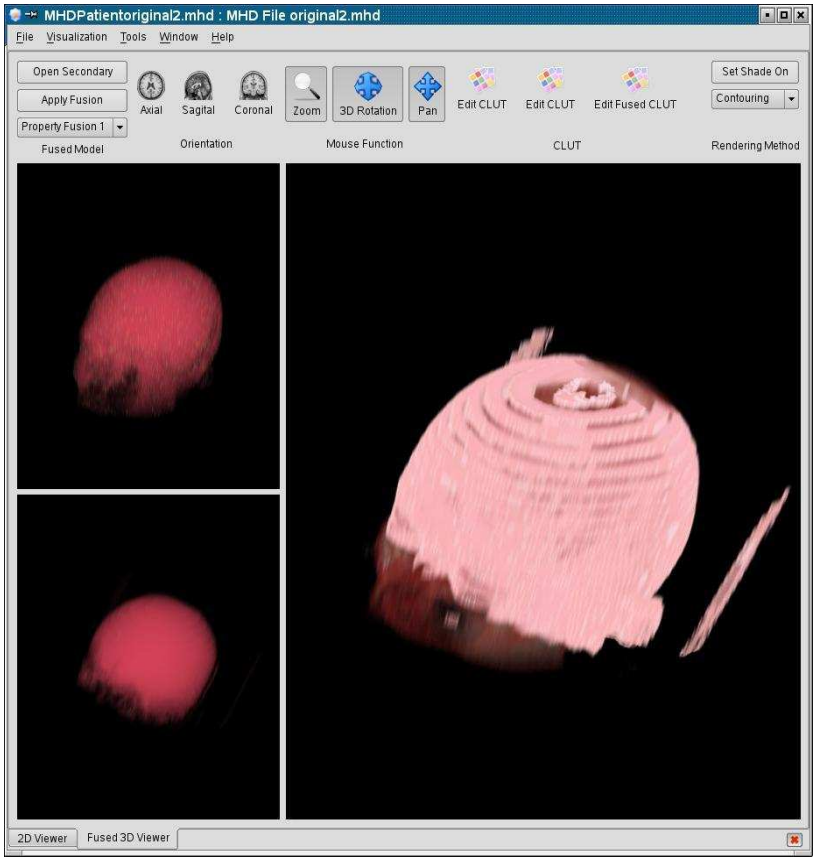
L'objectiu principal d'aquest projecte era implementar la visualització 3D de models fusionats i aplicar totes les tècniques possibles per realitzar aquesta fusió.

Per assolir aquest objectiu es van anar assolint els objectius més especifics.

En el projecte s'han estudiat els mètodes de visualització tan superficial com volumètrica. S'han assolit coneixements de com visualitzar dades en tres dimensions i com tractar les dades per poder representar-les. S'ha investigat com visualitzar un model simple així com afegir un mòdul dins de la plataforma de visualització d'imatge mèdica Starviewer.

S'ha entès el procés i l'algoritme de visualització en 3D d'un model de voxels fent servir diferents tècniques(Ray Casting,Isosuperficies, Contouring, Texture 3D, etc)

A més s'ha estat estudiant conjuntament amb altres desenvolupadors de la plataforma, models d'il·luminació avançada aplicable a la visualització 3D. La tècnica bàsica de visualització aplicada ha sigut el Ray Casting Volumètric. A més s'han fet servir en algun cas funcions d'ombres.



Un cop estudiat com visualitzar un model simple, s'ha estudiat i implementat un mòdul dins de l'aplicació que permet visualitzar models fusionats aplicant diferents tipus de fusió afegint totes les eines possibles que aporta la plataforma. S'han estudiat aquests mètodes de fusió segons l'etapa del procés de visualització i reproduir-lo fens servir llibreries gràfiques.

S'han estudiat tots els mètodes de fusió i s'han aplicat els que són més interessants alhora de fer servir model de dades mèdics. S'han desestimat aquells que no aporten cap millora alhora d'estudiar dades de models fusionats. S'ha treballat amb models sintètics per fer l'estudi i probes d'execucions. S'han realitzat proves amb models reals registrats **figura 8.1**.

Figura 8.1: *Visualització d'un model real fusionat.*

L'usuari és capaç d'interactuar amb l'aplicació per tal d'obtenir el millor resultat possible. Pot canviar les opcions de visualització i els paràmetres de la funció de transferència de cada model(els dos simples i el model fusionat).

A més, s'ha entès el que és integrar una modularitat en un projecte gran desenvolupat per diverses persones alhora i en continua evolució.

8.1 Conclusió personal

Tot informàtic ha d'aprendre a fer projectes ja sigui professionalment o en investigació. Com a estudiant d'últim curs d'informàtica i futur Enginyer Tècnic en Informàtica de Gestió s'ha d'aconseguir realitzar un projecte, en aquest cas un Projecte Final de Carrera i assolir uns coneixement suficients per l'entrada en el món laboral.

Vaig escollir aquest projecte per diverses raons:

Sempre m'ha agradat la informàtica gràfica i tot el que engloba aquest món: Visualització 3D, simulació, realitat virtual, etc. I una bona manera de poder aprendre molts coneixements en aquest àmbit era fer un projecte dins del grup de recerca IMA(Informàtica i Matemàtica Aplicada). Dins d'aquest grup hi ha un projecte que porta molts anys engegat anomenat Starviewer. Aquest projecte és una aplicació d'imatge mèdica que es desenvolupa juntament amb l'hospital Trueta de Girona.

Darrera d'aquest projecte hi ha i hi ha hagut moltes persones treballant. Al món laboral normalment els projectes es desenvolupen en grups de treball de moltes persones també i aquesta ha sigut una molt bona experiència per al dia de demà.

A més aquest departament es dedica a investigar en l'àmbit de la informàtica gràfica, un camp amb molt de futur i on no és fàcil tenir accés a material d'última generació.

Un dels objectius marcats és continuar lligat amb la investigació de gràfics de cares a cursar l'enginyeria informàtica superior.

S'ha treballat en un molt bon ambient de treball i en companyia de bons professionals i amics del sector i s'ha intentat aplicar tots els coneixements adquirits durant tots els anys de formació.

9. BIBLIOGRAFIA

Llibreries:

Vtk

-**The Visualization Toolkit 3rd Edition:** Will Schroeder, Ken Martin, Bill Lorensen. KitWare 2004

-**The VTK User's Guide 5th Edition:** Kitware, 2006

Qt

- C++ GUI Programming With Qt 4: Jasmine Blanchette, Mark Summerfield. Prentice Hall 2006

-The Book of Qt 4, The Art of Building Qt Applications: Daniel Molketin. 2004

10. WEBGRAFIA

Vtk

- http://www.vtk.org/doc/release/5.0/html/ (Referència de Classes

Qt

- http://doc.trolltech.com/4.4/classes.html (Referència de Classes)

Kdevelop

- http://docs.kde.org/development/en/kdevelop/kdevelop/ (Manual d'usuari)

Qt Designer

http://doc.trolltech.com/4.0/designer-manual.html (Manual d'usuari)