UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Universitat Politècnica de Catalunya (UPC)

Facultat d'Informàtica de Barcelona

Grau en Enginyeria Informàtica

Especialitat de Computació

Visualització avançada de models mèdics

Bernat Oliveras Boada

Dirigit per

Pere-Pau Vázquez Alcocer

Dpt. Ciències de la Computació

Codirigit per

Eva Monclús Lahoya

Grup de recerca ViRVIG

30 d'abril de 2020

Índex

1 Resum **1**

2 Introducció i contextualització **2**

2.1 Imatge mèdica 2

2.1.1 Models volumètrics 3

2.1.1.1 Funció de transferència 4

2.1.1.2 Visualització d'un model volumètric 4

2.1.1.3 Textura (estructura de dades) 4

2.2 Realitat Virtual 4

2.3 Punt de partida 5

3 Abast del projecte **7**

3.1 Objectiu general 7

3.2 Subobjectius 7

3.3 Possibles obstacles i riscos 8

3.4 Primera aposta: Exposure Render 9

3.5 Canvis en els objectius 10

3.6 Metodologia 10

4 Raytracing i raycasting **11**

4.1 Models volumètrics i models de superfície 11

4.2 Traçat de rajos en models de superfície 11

4.2.1 Funcionament del traçat de rajos 11

4.2.2 Font de llum 12

4.3 Model d'il·luminació de Phong 13

4.3.1 Llum ambient 14

4.3.2 Llum difusa 15

4.3.3 Llum especular 15

4.4 Càlcul de la llum especular mitjançant raytracing 16

4.5 Raycasting de models volumètrics 18

4.5.1 Gradient i normal 20

4.6 El nostre càlcul de la il·luminació 20

4.6.1 Contribució de la llum ambient . . . 23
4.6.2 Contribució de la llum difusa . . . 23
4.6.3 Contribució de la llum especular . . . 24
4.6.4 Atenuació de la llum . . . 25
4.6.5 Intensitat de la llum . . . 26
4.6.6 Il·luminació final . . . 28
4.6.7 Càlcul del color del píxel . . . 31

5 Posada en pràctica . . . 35
5.1 Escena amb una llum . . . 35
5.2 Escena amb múltiples llums . . . 36
5.3 Il·luminació final . . . 37
5.4 Interacció amb l'usuari . . . 42
5.4.1 Afegir i eliminar llums . . . 43
5.4.2 Desplaçar llums . . . 43
5.4.3 Altres facilitats . . . 43

6 Conclusions . . . 46

7 Planificació i gestió del projecte . . . 47
7.1 Planificació del projecte . . . 47
7.1.1 Gestió dels possibles obstacles . . . 48
7.1.2 Gantt . . . 50
7.2 Recursos . . . 50
7.2.1 Recursos humans . . . 50
7.2.2 Recursos hardware . . . 50
7.2.3 Recursos software . . . 51
7.3 Pressupost . . . 51
7.3.1 Costs en recursos humans . . . 51
7.3.2 Costs indirectes . . . 52
7.3.2.1 Llicències . . . 52
7.3.2.2 Consum elèctric . . . 52
7.3.2.3 Amortització del hardware . . . 52
7.3.2.4 Consum d'Internet . . . 53

7.3.2.5 Espai de treball 53
7.3.3 Contingència 53
7.3.4 Cost total 53
7.3.5 Prolongació del projecte 54
7.4 Sostenibilitat 55
7.4.1 Impacte ambiental del projecte posat en producció 55
7.4.2 Impacte ambiental de la vida útil 55
7.4.3 Impacte econòmic del projecte posat en producció 55
7.4.4 Impacte econòmic de la vida útil 55
7.4.5 Impacte social del projecte posat en producció 56
7.4.6 Impacte social de la vida útil 56

1 Resum

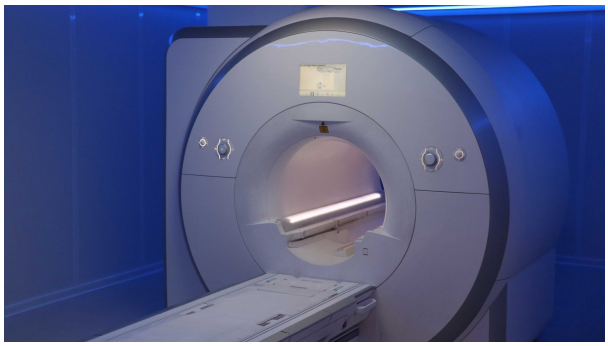
Amb la millora de la informàtica entre altres coses, cada cop s'estan aconseguint més èxits en molts camps. Un d'aquests camps és el de la medicina. Juntament amb altres tecnologies s'ha desenvolupat un sistema que permet la representació visual en dues o tres dimensions de parts reals de cossos humans. Amb el software del grup de recerca ViRVIG i el hardware del Centre de Realitat Virtual, on s'ha desenvolupat aquest projecte, podem visualitzar models mèdics utilitzant Realitat Virtual i de manera interactiva. Un dels problemes principals a l'hora de tractar dades de manera interactiva és la gran quantitat d'informació que es necessita per definir els models de volum, podent arribar fins als 1024^3 vòxels (píxels volumètrics). Per avaluar aquestes dades suficientment ràpid, s'utilitzen algoritmes basats en el traçat de rajos a la GPU. Per aconseguir una visualització realista, és necessari un bon càlcul de la il·luminació, per això l'objectiu principal d'aquest treball era desenvolupar-ne un. Per fer-ho, una de les principals tècniques implementades ha estat el càlcul de l'atenuació de la llum i de les ombres utilitzant el traçat de rajos. També s'ha canviat el software per permetre a l'usuari modificar al seu gust les fonts de llum, donant-li la possibilitat de crear un entorn amb la il·luminació que millor vagi pel seu treball.

2 Introducció i contextualització

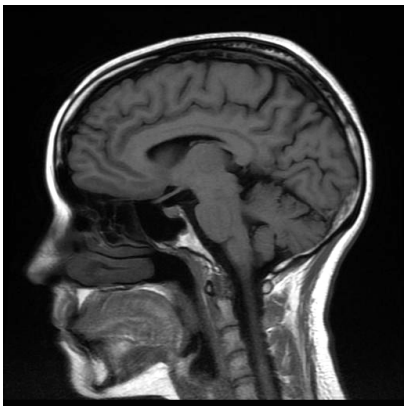
Amb la millora de la informàtica entre altres coses, cada cop s'estan aconseguint més èxits en molts camps. Un d'aquests camps, és el de la medicina. Juntament amb altres tecnologies s'ha desenvolupat un sistema que permet la representació visual en dues o tres dimensions de parts reals de cossos humans. Avui en dia encara se segueix treballant per millorar tant el hardware com el software per fer aquestes representacions més realistes i amb menys cost temporal i de recursos.

2.1 Imatge mèdica

Per obtenir les dades suficients per dur a terme aquestes representacions es pot fer de maneres diferents. Alguns exemples són la Imatge per Ressonància Magnètica (IRM) (vegeu figura 1) que utilitza un potent camp magnètic i radiofreqüència. És especialment útil per prendre dades del cervell, de l'aparell músculoesquelètic, del sistema cardiovascular i la detecció i seguiment de càncers[11]. També es poden utilitzar ones d'ultrasons, produïts per un transductor i captant el seu eco, utilitzant la velocitat del so i el temps que tarda a arribar l'eco es calcula la distància entre el transductor i el límit del teixit, aquest procediment també és conegut amb el nom d'ecografia.[3].



(a) Equip de ressonància magnètica.[22]



(b) Imatge d'una ressonància magnètica.[14]

Figura 1: Ressonància magnètica.

Com hem dit, existeix un sistema que permet la representació visual un cop s'ha obtingut certa informació. Aquest, agafa les dades obtingudes pels aparells mèdics i les tracta per a poder tenir una visualització el màxim detallada possible.

En la visualització de models mèdics acostuma a haver-hi una quantitat de dades bastant elevada, i això comporta una dificultat a l'hora de visualitzar-les. Per poder moure la càmera per l'espai a temps real han d'anar-se processant les dades i, en el cas que aquestes tardin massa a ser tractades, l'usuari ha d'esperar que acabin de processar-se cada cop que interacciona amb l'aplicació. Si es vol prioritzar un ús fluid, acostuma a significar menys qualitat en la representació.

És per això que s'han i se segueixen investigant maneres per a poder reduir el temps d'anàlisi i tractament de dades. Entre aquestes en podem trobar de tipus molt diferents, des de com representar les dades obtingudes fins a reduir el nombre d'elements que són necessaris per a la simulació tenint en compte que la càmera està en un lloc concret.

2.1.1 Models volumètrics

Un model és una representació d'un objecte. Hi ha diferents maneres de representar un objecte, en el món de la medicina s'utilitzen majoritàriament models volumètrics, aquests permeten veure diferents teixits alhora. Per obtenir les dades que ens permetran definir un model volumètric, com ja hem explicat s'acostumen a utilitzar ressonàncies magnètiques, ecografies, etc. Aquests mètodes generen un conjunt d'imatges de dues dimensions que representen una fina capa de l'element que es vol analitzar. Si s'apilen aquestes capes, es pot obtenir un model de vòxels (els vòxels són píxels volumètrics, la paraula ve de Volum ('vo') i Píxel ('xel')), és a dir, es pot representar un conjunt d'imatges 2D en tres dimensions (vegeu figura 2). Aquestes dades es poden representar utilitzant una malla tridimensional, que en cada posició té les dades necessàries per definir aquell vòxel.

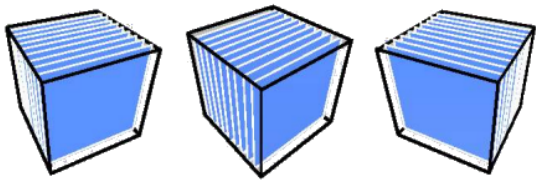


Figura 2: Imatges de dues dimensions apilades per formar-ne una de tres.[17]

En un volum la quantitat de dades acostuma a ser molt gran, és per això que la renderització (generació d'una imatge partint d'un model) dels models volumètrics es realitza a la GPU (Graphic Processing Unit). Això es fa per beneficiar la fluïdesa del càlcul, ja que la GPU està preparada per optimitzar el processament gràfic i ens permet paral·lelitzar les tasques a fer. Això és molt útil, ja que els càlculs per determinar el color de cada píxel tenen independència entre si. La peça de codi que s'executa en la GPU rep el nom de

shader.

Per poder visualitzar correctament els models volumètrics, es necessita poder determinar en quina mesura es veuen més o menys els diferents teixits.

2.1.1.1 Funció de transferència

La funció de transferència és una funció que indica les propietats visuals de les dades d'un model volumètric. A la pràctica, ens permet modificar els valors del color i l'opacitat dels teixits d'un volum. És una de les coses que l'usuari podrà modificar, donant-li així el color i opacitat que desitgi als diferents teixits.

2.1.1.2 Visualització d'un model volumètric

Per visualitzar models, molts cops es genera una representació poligonal d'aquests, que defineix la seva forma utilitzant normalment malles de triangles. El còmput de la contribució visual d'un model sense generar una representació poligonal es diu *Direct Volume Rendering* (DVR).[2]

2.1.1.3 Textura (estructura de dades)

En el món dels gràfics, i en aquest cas per fer el DVR, s'acostumen a utilitzar unes estructures de dades anomenades textures. Una textura és una estructura de dades del tipus *bitmap*, una matriu, on a cada posició s'hi emmagatzemen el que anomenarem "texels", paraula formada entre "texture" i "píxel". Aquests *texels* poden contenir diferent informació d'un punt en concret. Des del color fins a qualsevol altra propietat que li desitgem assignar. Mitjançant textures podem definir un model volumètric sense generar una representació poligonal d'aquest.

2.2 Realitat Virtual

La Realitat Virtual és la representació de les coses a través de mitjans electrònics. Pretén produir la sensació que allò representat és real i donar a l'usuari l'oportunitat d'interaccionar amb l'entorn generat. Té moltes àrees d'aplicació, però on més s'utilitza és en el camp de l'arquitectura, enginyeria aeroespacial, entreteniment i medicina.

Hi ha dos tipus de sistemes de Realitat Virtual, els immersius i els semi-immersius. Els immersius consisteixen en posicionar una pantalla davant de cada ull de l'usuari, creant així una sensació de profunditat molt realista. Al tenir dues pantalles davant els ulls, la persona usant un dispositiu d'aquest tipus no es pot veure a si mateixa ni a les coses que l'envolten. Els sistemes semi-immersius (vegeu la figura 3) consisteixen en

una o més pantalles on es projecten imatges estèreo que simulen la presència d'objectes en un entorn des del punt de vista de l'usuari, d'aquesta manera pot seguir veient a si mateixa i a les coses que l'envolten.

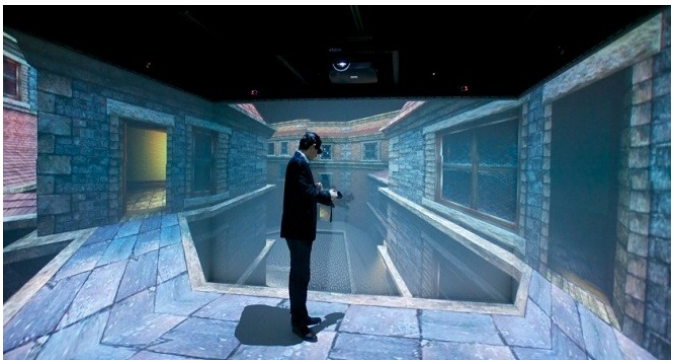


Figura 3: Sistema de Realitat Virtual semi-immersiu.[27]

Per dur a terme aquest treball s'utilitzarà un sistema immersiu anomenat HTC Vive (vegeu la figura 4). Aquest sistema consisteix en unes ulleres de Realitat Virtual dissenyades per a ser utilitzades en una sala amb espai. A aquestes ulleres les acompanyen dos controladors sense fils que permeten interactuar amb la Realitat Virtual i dues estacions base, que són sensors infrarojos que permeten adaptar la Realitat Virtual a escala de l'habitació per a ajudar a les ulleres i als controladors a rastrejar la teva posició i qualsevol moviment d'una forma exacta i al moment.

Figura 4: HTC Vive.[13]

2.3 Punt de partida

Per dur a terme aquest projecte, es parteix del treball de fi de grau "Inspecció interactiva i immersiva de models volumètrics. Aplicació diagnosi mèdica" d'en Joan Fons Sànchez[23], que consistia en crear una aplicació capaç de renderitzar i explorar models volumètrics d'imatges mèdiques. Aquesta aplicació va ser

creada en la plataforma Unity3D.

S'utilitzaran també les instal·lacions del Centre de Realitat Virtual (CRV) i el software del grup de recerca ViRVIG.

En el projecte del qual es parteix, el renderitzat dels models volumètrics es fa utilitzant un *raycasting* simple (vegeu l'apartat 4.5). La il·luminació es basa en el model Phong (vegeu l'apartat 4.3), implementat amb una sola font de llum col·locada a la càmera i en direcció cap a endavant d'on mira l'usuari.

3 Abast del projecte

En un primer moment els objectius plantejats eren aquests.

3.1 Objectiu general

Es volia dotar al software del ViRVIG d'una il·luminació més realista en la renderització del volum en directe. Un cop aquesta estigués implementada, es volia avaluar tant l'eficiència en l'ús de recursos com en cost temporal en comparació a l'anterior model d'il·luminació. També es volia convidar a diversos usuaris per a dur a terme un test d'usabilitat, intentant que, a part d'haver-hi gent amb coneixements de Realitat Virtual hi hagués també metges i radiòlegs per tenir la seva opinió respecte a quina il·luminació era més adequada per realitzar la seva feina.

3.2 Subobjectius

Vam dividir el projecte en subobjectius per poder organitzar millor la feina, vam decidir que serien aquests:

- **Formació:** El primer subobjectiu era entendre els conceptes que formen part de la modelització i renderització dels models 3D, tot i que aquests ja havien estat tractats a l'assignatura Gràfics s'haurien de recuperar i ampliar per poder desenvolupar aquest treball. Per dur-ho a terme, va ser de gran ajuda el llibre "Real-Time Volume Graphics"[12] entre altres recursos que van ajudar a la seva millor comprensió.

- **Entendre el codi:** Un cop enteses les bases, faria falta saber-les aplicar, és per això que es pretenia estudiar el codi ja existent del ViRVIG per entendre com s'havien aplicat els conceptes prèviament estudiats, realitzant petites modificacions per comprovar que s'hagués entès correctament.

- **Modificar la posició de la llum:** Per començar a familiaritzar-se amb el projecte, es volia començar a treballar amb la il·luminació. La primera modificació que es volia fer consistia en retirar la llum implementada en el codi del ViRVIG i col·locar-la en un lloc diferent que es pogués anar modificant. Després permetríem afegir diverses fonts de llum en diferents posicions.

- **Primer test d'usabilitat:** Un cop aconseguits els objectius anteriors, es faria un test d'usabilitat per comprovar que no hi hagués errors i en el cas que n'hi haguera o hi haguera possibles millores s'arreglaria el codi.

- **Primera millora en la interacció de l'usuari:** Un cop acabades d'implementar les llums en diferents posicions, utilitzant el material del CRV, s'implementaria una funcionalitat que permetés utilitzar un dels comandaments inclosos en el HTC Vive per a posicionar fonts de llum, agafant com a posició de la font de llum la posició del comandament i com a direcció de la llum la direcció del comandament.

- **Segona millora en la interacció de l'usuari:** Després, un cop adquirits els coneixements per a treballar amb el comandament, s'implementaria que el comandament pogués ser utilitzat com a llanterna, és a dir, que fos una font d'il·luminació. Aquestes últimes implementacions podrien ser utilitzades conjuntament amb el nostre model d'il·luminació un cop aquest estigués acabat.

- **Segon test d'usabilitat:** Per acabar, i abans de començar amb els canvis del model d'il·luminació explicat a l'apartat 4.6, es duria a terme un altre test d'usabilitat i es tornaria a arreglar el codi en cas que fos necessari o positiu.

.

3.3 Possibles obstacles i riscos

La visualització de dades mèdiques és capaç de generar de forma interactiva imatges en models volumètrics d'uns 1024^3 vòxels amb algoritmes basats en el traçat de rajos (vegeu l'apartat 4.2.1) a la GPU. Per aconseguir visualitzacions més realistes, calen algoritmes més avançats.

L'objectiu principal d'aquest treball era dotar d'una il·luminació més realista al software del ViRVIG. Per dur a terme això es necessitaria calcular la llum que arriba als vòxels d'una manera més complexa. Això, malauradament afegiria un cost temporal que probablement afectaria la interacció de l'usuari amb el nostre software, fent aquesta de menys agradable ús. Per això, s'hauria de trobar un recurs mitjançant el qual fos compatible una bona il·luminació amb fluïdesa a l'hora d'interaccionar.

El principal risc que es corria era que el nostre codi tingués un cost temporal major de l'esperat. Un cop implementat el nostre codi hi havia la possibilitat que no fos prou eficient i, en conseqüència, que la *frame rate* del HTC Vive fos massa baixa i no ens permetés una visualització fluida de models que prèviament sí que podíem visualitzar. El projecte pretenia també dotar d'una bona experiència de Realitat Virtual i per tant, en el cas de trobar-nos en aquesta situació, tindríem dues opcions, o bé reduir la mida del volum que volem visualitzar, o bé modificar part de l'algorisme per reduir el cost.

3.4 Primera aposta: Exposure Render

En el camp de la visualització de volums, s'ha dut a terme molta investigació i se'n segueix realitzant. Gràcies a això hi ha una gran varietat de tècniques i algorismes de diferents tipus. Alguns per a generar una visualització més realista, d'altres per a disminuir el cost temporal, etc.

El principal problema que es pretenia solucionar amb aquest treball era el de la il·luminació. A l'hora de visualitzar models mèdics ens trobem amb casos d'una quantitat d'informació que fa que una renderització en directe i realista dels models sigui pràcticament inviable. És per això que s'apliquen diverses tècniques per reduir el temps de computació i aconseguir una visualització semblant a la desitjada.

Partint d'això, el que volíem era aconseguir implementar una tècnica que ens permetés obtenir un renderitzat de volum foto-realista i interactiu, és a dir, que pogués donar una imatge de qualitat a temps real mentre l'usuari interacciona amb l'aplicació.

La implementació que semblava més adequada pel nostre objectiu era aplicant el conjunt de tècniques implementades per l'Exposure Render [16]. Aquest s'implementa utilitzant el llenguatge de programació CUDA.

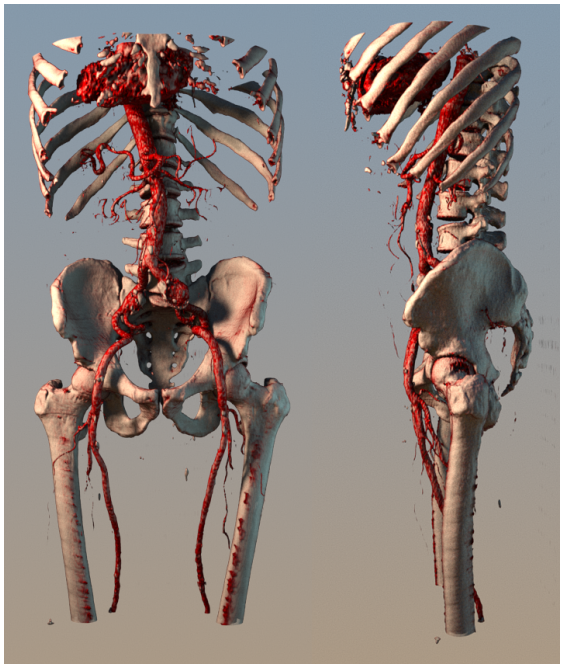


Figura 5: Resultat d'aplicar l'Exposure Render.[10]

3.5 Canvis en els objectius

En el transcurs del Treball Final de Grau, ens hem trobat amb obstacles que han fet canviar els objectius d'aquest. També s'han afegit objectius nous per millorar la interacció amb l'usuari. Exposarem aquests canvis a continuació.

- **Exposure Render:** L'objectiu general era dotar d'una il·luminació més realista al software del ViRVIG, per fer això l'aposta principal era desenvolupar part del conjunt de tècniques utilitzades en l'Exposure Render utilitzant el llenguatge de programació CUDA. Després d'entendre el codi i fer els primers canvis, es va intentar fer funcionar un codi en CUDA des d'Unity, per sorpresa nostra, ens vam adonar que no eren compatibles. Davant aquesta situació la nostra decisió va ser seguir endavant amb el software de Unity i intentar integrar el màxim possible de tècniques que permetessin una il·luminació més realista.

- **Interacció amb l'usuari:** També vam decidir afegir més funcionalitats perquè la il·luminació de la nostra escena fos més interactiva per l'usuari. Aquestes funcionalitats les explicarem en l'apartat 5.4.

3.6 Metodologia

Per a dur a terme el projecte, es va utilitzar metodologia àgil. Es van anar establint diferents objectius, i si les circumstàncies ho requerien, se n'establien de nous. Un cop complerts, en una reunió amb la codirectora s'avaluaven i es feien els canvis necessaris, un cop fets aquests canvis, es tornaven a avaluar i un cop tot enllestit, es passava al següent objectiu.

Dos dies a la setmana, anava a avançar el treball al Centre de Realitat Virtual, on es realitzaven també reunions amb la codirectora, a qui es posava al dia de com anava el projecte i amb qui conjuntament es decidien els propers objectius.

Els dos dies setmanals, que van acabar sent-ne més, s'aprofitaven per utilitzar el hardware de Realitat Virtual del qual es disposa al CRV, ja que no es disposava del material necessari per fer les proves vinculades amb la Realitat Virtual als altres entorns on es treballava.

4 Raytracing i raycasting

4.1 Models volumètrics i models de superfície

Per representar una escena en el món de la informàtica hi ha diverses maneres de definir i tractar els diferents objectes que apareixen en aquesta. En aquest treball, analitzarem un objecte que estarà definit com un model volumètric (vegeu apartat 2.1.1). En comparació amb els models volumètrics, que recordem, permeten observar un element amb diverses capes (per exemple: pell, múscul, os, etc.), els models de superfícies només ens permeten observar la part més exterior d'un model. Per exemple, si tenim un ou representat amb un model de superfície, veuríem només la closca, en canvi, si fos un model volumètric, podríem veure també la clara i el rovell.

4.2 Traçat de rajos en models de superfície

La tècnica anomenada *raytracing* i la tècnica anomenada *raycasting* ens permeten determinar la visibilitat de superfícies traçant rajos imaginaris (vegeu l'apartat 4.2.1) des de l'ull de l'observador fins a un objecte a l'escena. El *raycasting* acostuma a ser més ràpid que el *raytracing*, tot i que aquest últim permet generar una il·luminació més realista. El *raytracing* s'utilitza per renderitzar imatges quietes i per fer efectes visuals però encara no es fa servir en aplicacions de temps reals, com per exemple videojocs, on la velocitat és crítica.[21]

El projecte del qual partim permet renderitzar un model volumètric utilitzant la tècnica anomenada *raycasting*, però abans d'aprofundir en el nostre software explicarem el funcionament de la tècnica de *raytracing* en models de superfícies.

4.2.1 Funcionament del traçat de rajos

Per mostrar un volum per pantalla, es necessita saber de quin color s'ha de pintar cada un dels seus píxels. Per fer-ho, es va píxel per píxel.

Ens podem imaginar un panell entre l'escena i la posició de l'observador, com es mostra en la figura 6. Aquest panell representarà la pantalla de l'observador, per tant es divideix en el mateix nombre de píxels que es volen representar. Per cada píxel, el que es fa és, des de la posició de l'observador es traça un raig en direcció al centre del píxel, és a dir, s'avança de posició fins que es troba un element de l'escena, un cop trobat l'element, aquest determina el color del píxel.

Per calcular el color del píxel en el *raytracing* s'acostumen a tenir en compte diversos efectes òptics com poden ser la reflexió, la refracció, la dispersió, etc. Per fer els càlculs d'alguns d'aquests efectes és necessari traçar altres rajos (vegeu l'apartat 4.4), afegint més cost temporal per calcular la imatge resultant. La principal diferència entre el *raycasting* i el *raytracing* és que en aquest últim es tracen altres rajos per determinar el color final del píxel i en el *raycasting* el color es calcula d'una manera més senzilla.

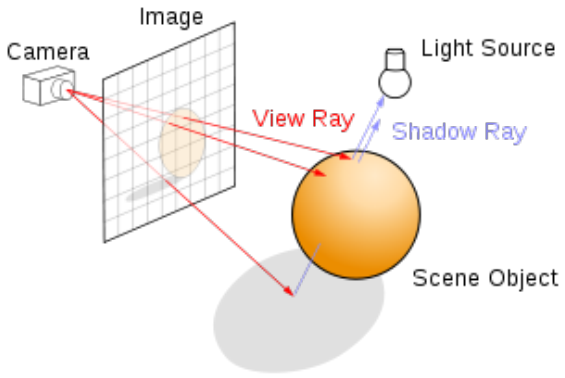


Figura 6: Esquema del funcionament del traçat de rajos.[8]

4.2.2 Font de llum

Per calcular el color dels elements que veiem per pantalla, s'ha de tenir en compte la il·luminació. Quan parlem d'una font de llum, d'una llum o d'una font d'il·luminació, ens referim a un element que genera il·luminació, és a dir, que en una posició en concret hi ha alguna cosa que aporta il·luminació a l'escena. En el món real, una font de llum podria ser una bombeta, el Sol, les llums d'un cotxe, un llumí, etc.

La llum generada pot ser direccional o omnidireccional. Les llums omnidireccionals són aquelles que generen il·luminació en totes direccions, un exemple podria ser una espelma, en canvi, les llums direccionals són aquelles que il·luminen en una direcció en concret, com per exemple les llums d'un cotxe. Les llums direccionals tenen un angle d'obertura que determina la superfície que il·lumina una llum a una distància determinada (vegeu figura 7).

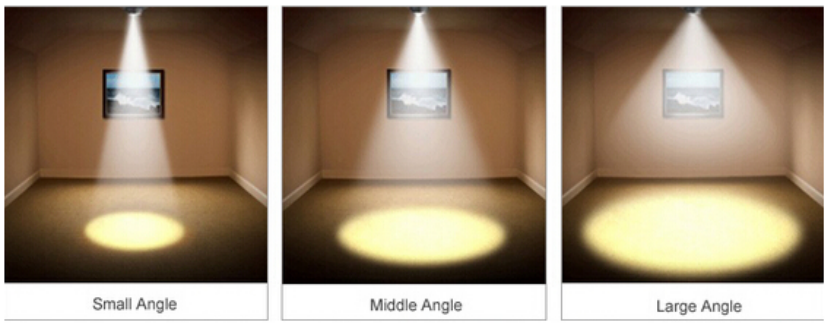


Figura 7: Diferents angles d'obertura d'una llum.[24]

En el treball del qual partim el color de cada píxel es calcula suposant que a la posició de l'observador hi ha una llum puntual omnidireccional, per calcular la contribució d'aquesta llum a cada punt del volum es fa utilitzant el model d'il·luminació Phong. En el següent apartat ens endinsarem en aquesta tècnica.

4.3 Model d'il·luminació de Phong

Hi ha diversos mètodes per calcular la il·luminació en un punt. Nosaltres partim i ens basarem en la tècnica anomenada Phong. Aquesta aproxima el color d'una superfície sumant tres components. Aquests components són la llum difusa, la llum especular i la llum ambient.[6]

Per calcular la il·luminació en un punt amb el model Phong hi ha tres variables que influencien en el resultat. La primera és la posició de l'observador, en la figura 8 és la que determina el vector V. La segona és la normal, és a dir el vector perpendicular a la superfície en el punt que estem analitzant. Finalment s'ha de tenir en compte la posició de la llum, que en la figura 8 determina el vector L. Utilitzant les dues últimes variables podem obtenir el vector R, que indica quina és la direcció de la reflexió de la llum en un punt en concret.

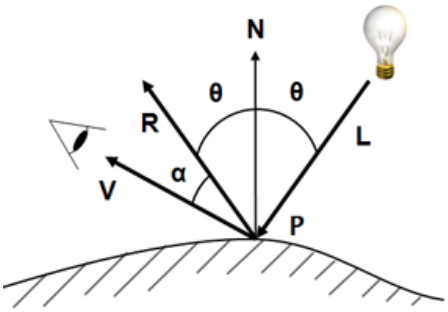
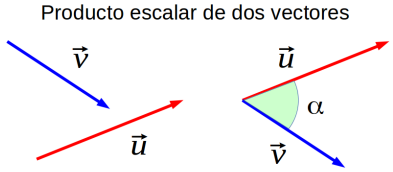


Figura 8: Il·luminació local.[19]

Pels càlculs de la il·luminació serà necessari saber la magnitud de la diferència de direcció entre dos vectors. Per calcular-la farem servir el producte escalar, que, com podem recordar en la figura 9, ens permet trobar el valor del cosinus de l'angle que hi ha entre dos vectors, i, en conseqüència aquest angle. Com més gran sigui l'angle, més diferents seran les direccions dels dos vectors.



$$\vec{u}\cdot\vec{v} = |\vec{u}|\cdot|\vec{v}|\cdot\cos\alpha$$

$$\vec{u}\cdot\vec{v} = u_1\cdot v_1 + u_2\cdot v_2$$

Figura 9: Producte escalar.[1]

A continuació explicarem cada component de la il·luminació Phong. Abans, aclarir que cada component té el seu coeficient, aquest indica en quina quantitat contribueixen al color final, és a dir, el pes que té cada component per determinar el resultat final.

4.3.1 Llum ambient

Aquest component intenta aproximar la contribució de totes les il·luminacions indirectes. Per entendre-ho millor, podem imaginar una sala amb una única font de llum provinent d'una finestra. La llum del sol no arriba directament a totes les superfícies, però tot està il·luminat a causa de les partícules de llum que han rebotat. A aquesta il·luminació la denominarem llum ambient.

Per calcular la llum ambient ho farem de la següent manera:

$$C_a = k_a * C \tag{1}$$

On C_a indica el color de la contribució de la llum ambient, k_a indica el coeficient de la llum ambient i C és el color del punt de la superfície que estem analitzant.

4.3.2 Llum difusa

Aquest component es basa en la llei de Lambert, que determina que la il·luminació produïda per una font de llum sobre una superfície és directament proporcional al cosinus de l'angle que forma la normal amb la direcció del raig de llum, és a dir l'angle que formen N i L. Per tant, podem calcular la llum difusa de la següent manera.

$$C_d = k_d * C * max(N \cdot L, 0) * I \tag{2}$$

On C_d indica el color de la contribució de la llum difusa, k_d el coeficient de la llum difusa, I la intensitat de la llum i $N \cdot L$ el producte escalar dels vectors N i L. Utilitzem el producte escalar, ja que, si els vectors estan normalitzats, aquest és equivalent al valor del cosinus. Agafem el valor més gran entre el producte escalar i 0, ja que, en cas que el producte escalar fos negatiu, agafaríem el valor 0 per evitar errors.

4.3.3 Llum especular

En els objectes amb superfícies reflectants sovint podem veure reflexos de la llum. En el model Phong aquests són els que anomenarem llum especular. Per calcular els reflexos, calcularem el vector R, que és el que determina la direcció en què una superfície reflecteix un raig de llum. Aquest és equivalent al vector simètric al raig de llum, utilitzant com a eix de simetria la normal, com podem veure a la figura 8.

Un cop sabem cap a on reflecteix una superfície una llum concreta, falta saber en quina mesura veu l'observador aquest reflex. Per calcular-ho es necessita saber la posició de l'observador, ja que, depenent de l'angle que formen el vector V i el vector R de la figura 8, l'observador veurà més o menys un reflex.

Un cop més, al dependre d'un angle per calcular la llum especular, utilitzarem el producte escalar. Com més petit sigui l'angle, més llum especular veurem, ja que el raig reflectit anirà més directament a l'observador. Com més proper a zero sigui l'angle, més gran serà el cosinus, i aquest indica la quantitat de llum especular que rep l'observador.

Així doncs, per calcular la llum especular, utilitzarem la següent fórmula:

$$C_e = k_e * C * max(R \cdot V, 0) * I \tag{3}$$

On C_e indica el color de la contribució de la llum especular, k_e el coeficient de la llum especular i R · V el producte escalar dels vectors R i V.

En la figura 10, podem veure un exemple de les tres contribucions en un model d'il·luminació Phong i el resultat final de la suma total d'aquestes.

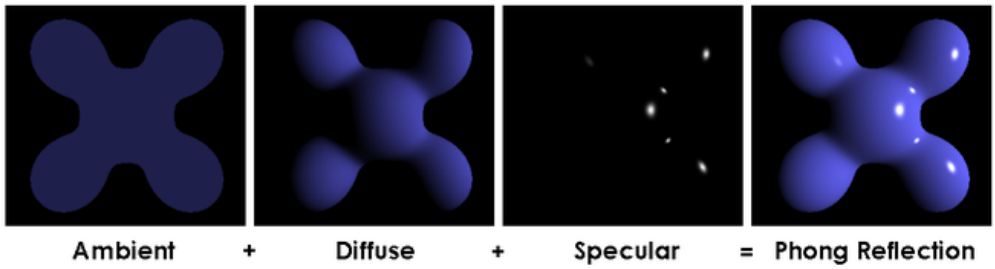


Figura 10: Factors de la il·luminació Phong.[7]

4.4 Càlcul de la llum especular mitjançant raytracing

En el càlcul d'il·luminació utilitzant *raytracing* la llum especular s'acostuma a calcular de la següent manera:

- Des del punt del qual volem calcular la il·luminació es tracen dos rajos, el raig reflectit (figura 11) i el raig refractiu (figura 12), la direcció del raig reflectit es calcula utilitzant la normal com a eix de simetria i la direcció del raig refractiu és la normal negativa.

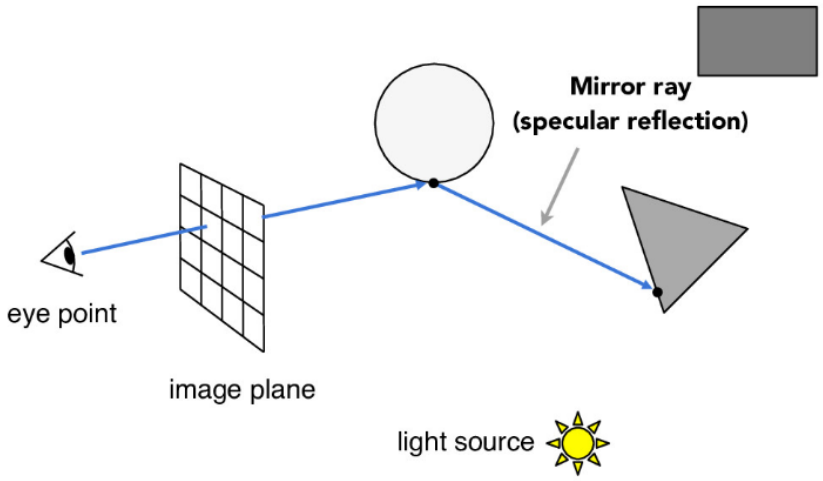


Figura 11: Raig reflectit.[20]

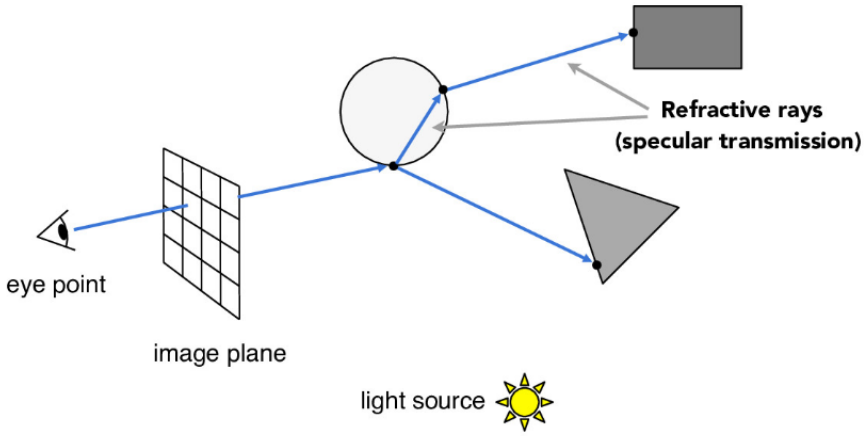


Figura 12: Rajos refractius.

- Aquests rajos es tracen recursivament fins que es troben una superfície sense propietat especular (és a dir, que no reflecteix la llum) o fins que s'arribi al nivell màxim de recursivitat (com més nivells de recursivitat el resultat serà més realista però el càlcul s'allargarà més).

- Per cada punt trobat es traça un raig cap a cada llum per calcular les ombres (vegeu apartat 4.6.4). A la figura 13 podem veure un exemple dels rajos que es traçarien.

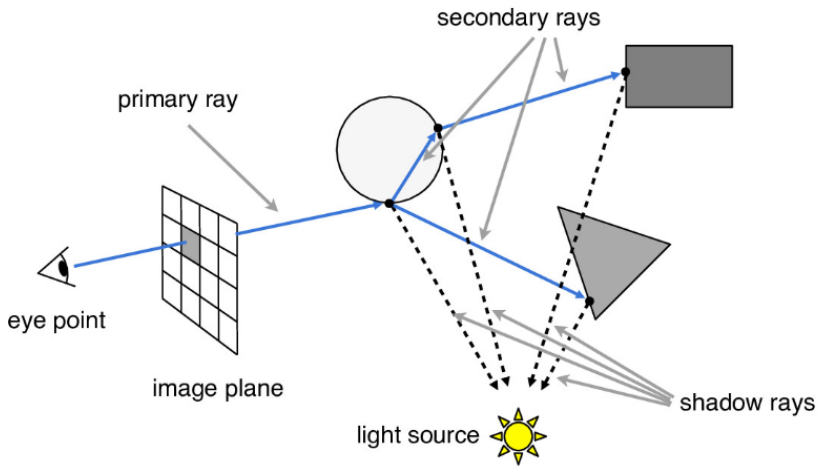


Figura 13: Rajos que influeixen en la llum especular utilitzant *raytracing* recursiu.

- Un cop trobats tots els punts que influeixen es fa el càlcul ponderat de les contribucions de cada raig per determinar el color del píxel.

Com podem veure, el càlcul total del color de cada punt és massa complex per poder-ho fer a temps real, és per això que en la representació de models de forma interactiva no s'acostuma a utilitzar.

4.5 Raycasting de models volumètrics

Com ja hem dit, en el nostre software partim d'una visualització que utilitza la tècnica *raycasting* calculant la il·luminació utilitzant el model Phong.

Per calcular el color de cada píxel quan volem representar un model volumètric, no ens serveix el traçat de rajos que s'utilitza en models de superfícies, que, si recordem, quan trobava un element, aquest determinava el color final del píxel. En el nostre cas, no veiem només una superfície, per tant, no podem només tenir en compte el primer teixit que trobem. Per poder aplicar el traçat de rajos de superfícies als models volumètrics, en el nostre software ho fem de la següent manera.

Llençarem un raig fins que trobi el volum. Un cop trobat aquest volum, avançarem a poc a poc en coordenades de textura, és a dir, com si avancéssim en la malla tridimensional que defineix el nostre volum. Seran passos suficientment petits per no saltar-nos el límit de cap teixit i així tenir en compte tot el que hem de veure. A mesura que anem avançant, cada vegada que canviem de teixit (a l'apartat 4.5.1 explicarem com saber quan hi ha un canvi de teixit) anirem acumulant opacitat i color, de manera que els teixits que

ens haurem anat trobant defineixin el color final de cada píxel. A la figura 14 podem veure un esquema de com funciona el nostre traçat de rajos. Els punts anomenats g_0, g_1 i g_2 formen part d'una superfície opaca i per tant, un cop trobats deixarem de calcular el color, ja que no veurem res més enllà. Per millorar l'eficiència, quan sortim del volum (punts l_0, l_1, etc.) pararem de calcular, ja que no trobarem cap nou teixit.

Com hem dit anirem acumulant color i opacitat a mesura que ens trobem nous teixits, ho farem utilitzant aquestes dues fórmules:

$$ContribucioColor = (1 - OpacitatAcumulada) * OpacitatSuperficieActual \tag{4}$$

$$OpacitatAcumulada = OpacitatAcumulada + (1 - OpacitatAcumulada) * OpacitatSuperficieActual \tag{5}$$

L'opacitat del nou teixit en acumular el seu valor a l'opacitat total, ho farà respecte a l'opacitat que falta per acumular, és a dir si el nou teixit té una opacitat del 50% i ja hem acumulat un 30%, aquest acumularà un 50% del 70% restant. Abans d'acumular l'opacitat calcularem quant contribueix el color del teixit, per fer-ho, multiplicarem el color per la seva contribució, que ve definida pel percentatge d'opacitat que resta per acumular multiplicat per l'opacitat del teixit actual, en l'exemple anterior també vindria definida pel 50% multiplicat pel 70%.

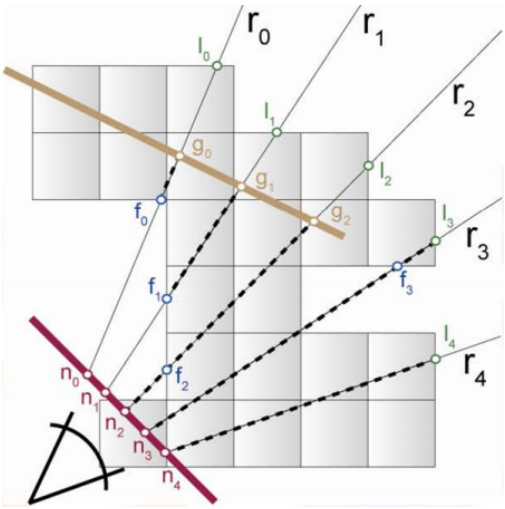


Figura 14: Esquema del funcionament del traçat de rajos en models volumètrics.[18]

4.5.1 Gradient i normal

Quan tracem un raig, ens interessarà saber quan canviem de teixit, ja que, com ja hem explicat, serà quan calculem la contribució del color.

El gradient és una aproximació a la normal d'una superfície, és a dir, al vector perpendicular d'aquesta. Si un punt pertany a la superfície, el seu vector gradient estarà ben definit. Per saber si un punt pertany a la superfície o no, només ens fa falta calcular la magnitud del gradient i, en el cas que superi un cert valor relatiu que establirem segons cada teixit, vol dir que sí que hi pertany.

Per calcular el vector gradient hi ha diversos mètodes. Matemàticament, el gradient es calcula fent una primera derivada i es pot definir de la següent manera:

$$df(x,y,z) = (\frac{df}{dx}, \frac{df}{dy}, \frac{df}{dz}) \tag{6}$$

Aquest valor es pot aproximar utilitzant els primers termes de l'expansió de Taylor. A la pràctica es fa tenint en compte els sis veïns d'un vòxel (recordem que estem en un espai de tres dimensions i per tant té sis veïns i no quatre). S'aproxima utilitzant la següent fórmula:[15]

$$\begin{aligned} df(x,y,z) = {} & (1/2) * (f(x+1,y,z) - f(x-1,y,z)), \\ & (1/2) * (f(x,y+1,z) - f(x-1,y-1,z)), \\ & (1/2) * (f(x,y,z+1) - f(x-1,y,z-1)) \end{aligned} \tag{7}$$

On el valor de la funció f ve determinat per la densitat del vòxel, és a dir, el valor de la propietat del teixit del vòxel.

4.6 El nostre càlcul de la il·luminació

Per fer aquest treball, partim d'un software on s'utilitza el *raycasting* per trobar les superfícies que hem de representar i el càlcul de color es fa utilitzant la tècnica Phong amb una única font de llum col·locada a la posició de l'observador.

Partint d'aquí, nosaltres hem fet els següents canvis:

- Primer hem implementat que l'única font de llum no estigui a la posició de l'observador sinó fixa en una posició. També permetem que l'usuari pugui col·locar-la on vulgui interactivament.

- Hem habilitat que el càlcul de la il·luminació es pugui realitzar quan en una escena hi ha diverses fonts de llums, i l'usuari pougui afegir-les, treure-les i modificar-les al seu gust.

- Finalment hem aplicat tècniques per fer un càlcul de la il·luminació més realista. Les tècniques implementades ens permeten calcular l'atenuació de la llum, les ombres i la intensitat de la llum. Les llums direccionals no il·luminen en la mateixa mesura en tot el seu rang, normalment il·luminen més al mig, el valor que determina si il·luminen més o menys depenent de la direcció de la llum i la posició de l'objecte l'anomenarem intensitat. Quan parlem d'ombres i d'atenuació de la llum ens referirem a l'obstaculització de la llum en un punt a causa d'altres teixits o objectes que hi ha entre el punt i la font de llum.

A continuació explicarem com realitzem el càlcul del color dels píxels en el nostre càlcul d'il·luminació final, és a dir, un cop aplicades les tècniques citades.

Des de la posició de l'observador traçarem un raig per cada píxel de la pantalla en la direcció que li correspon. Quan aquest raig trobi una superfície procedirem amb el càlcul de color, altrament, seguirem avançant el raig. Per saber si el raig està passant per una superfície, com ja hem explicat prèviament, comprovarem que la magnitud del gradient sigui suficientment elevada.

Un cop trobada una superfície, calculem el color del punt que ha trobat el raig. Podria ser el cas que la superfície no fos opaca, i per tant, no només veuríem aquella superfície, en aquest cas, calcularíem el color i el multiplicaríem per la seva contribució, determinada per l'opacitat, per saber quant aporta al color total i seguiríem avançant per comprovar si trobem més teixits. Per trobar la contribució ho farem amb les següents fórmules:

$$ContribucioColor = (1 - OpacitatAcumulada) * OpacitatSuperficieActual \tag{8}$$

$$OpacitatAcumulada = OpacitatAcumulada + (1 - OpacitatAcumulada) * OpacitatSuperficieActual \tag{9}$$

El pseudocodi del nostre *raycasting* és aquest:

```
1 raig, pas = calcularInfoRaig(pixelActual)
2 while(not foraDelVolum){
3       posicioActual = posicioActual + pas
```

```
4       infoVoxel = calcularInfoVoxel()
5       resultat = calcularColor(infoVoxel)
6       colorActual += (1. - opacitatActual) * resultat.color * resultat.
            opacitat
7       opacitatActual += (1. - opacitatActual) * resultat.opacitat
8       if (opacitatActual >= 1.) break
9   }
10  return (colorActual, opacitatActual)
```

on:

```
1   calcularInfoRaig(pixelActual){
2      origenRaig = calcularPosicioEntradaAlVolum(pixelActual)
3      finalRaig = calcularPosicioSortidaDelVolum(pixelActual)
4      raig = finalRaig - OrigenRaig
5      llargariaRaig = length(raig)
6      pas = valor/llargariaRaig
7   }
```

La funció calcularColor s'explica en l'apartat 4.6.6. Per calcular la posició d'entrada i de sortida del volum utilitzarem la caixa contenidora del nostre model volumètric. *Length* és una funció que donat un vector ens permet veure la seva distància euclidiana, aquesta és calculada fent l'arrel quadrada del quadrat de cada coordenada d'un vector. La variable valor és definida pel mínim increment que es pot fer en coordenades de textura per no saltar-se posicions d'aquesta.

Ara falta calcular el color. Com hem vist a l'apartat 4.3 i com podem recordar en la figura 10, hem de tenir en compte els tres tipus de llum explicats, l'ambient, la difusa i l'especular. El color d'un punt el calcularem de la següent manera:

$$ColorPunt = Color * (ContribucioAmbient + ContribucioDifusa + ContribucioEspecular) \quad (10)$$

On Color indica el color de la superfície que hi ha al punt que estem tractant. A continuació explicarem el càlcul de la contribució de cada tipus de llum i un cop fet això resumirem en un pseudocodi el nostre càlcul de la il·luminació.

Pot ser el cas que en el nostre model d'il·luminació hi hagi múltiples llums. Si es donés, calcularíem per

cada llum la seva contribució difusa i la seva contribució especular i sumaríem els resultats.

4.6.1 Contribució de la llum ambient

Com ja hem explicat en l'apartat on s'explica el model d'il·luminació Phong (4.3), la gran majoria de cops tindrem il·luminació indirecta generada per les altres llums, aquesta, si fos calculada detalladament generaria uns costs de recursos i temps inadmissibles, per tant, simplement l'aproximarem. En el nostre model, suposarem que la llum ambient ens permet veure un 15% del color de l'objecte, aleshores podem definir el que contribuirà la llum ambient al color del punt de la següent manera:

$$ContribuacioAmbient = 0.15 \tag{11}$$

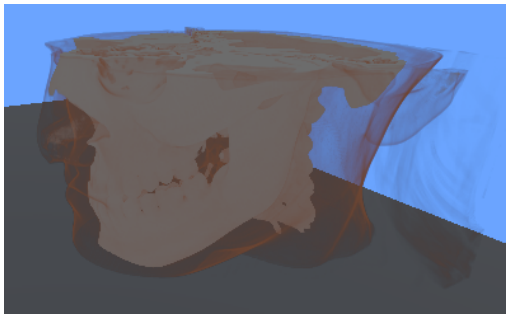


Figura 15: Model il·luminat només amb llum ambient.

4.6.2 Contribució de la llum difusa

Per saber quant contribueix la llum difusa en el punt que estem tractant, recordem, hem de trobar el cosinus de l'angle que forma la normal amb la direcció del raig de llum, el calculem mitjançant el producte escalar entre aquests dos vectors. Per trobar la normal, utilitzarem el gradient, que, com ja hem explicat prèviament, és una aproximació d'aquesta. Per trobar la direcció del raig de llum, només hem de calcular el vector que va des de la posició de la llum al nostre punt, això ho podem calcular restant les coordenades de la posició de la llum a les de la posició del nostre punt. Un cop ja tenim els dos vectors, només ens falta calcular, per fer-ho utilitzarem la següent formula:

$$ContribucioDifusa = 0.65 * max(N \cdot L, 0) * Intensitat * AtenuacioLlum \tag{12}$$

On L és el vector direcció normalitzat del raig de llum i N és la normal normalitzada. El valor 0,65 és el

coeficient que indica quin pes té la llum difusa al color total del píxel. Als apartat 4.6.4 i 4.6.5 explicarem d'on surt i com calculem el valor de la intensitat i de l'atenuació de la llum.

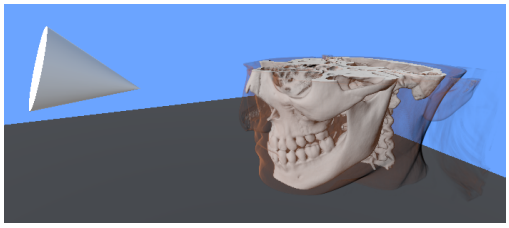


Figura 16: Model il·luminat només amb llum difusa. El con representa la font de llum i la seva cúspide marca la direcció de la llum.

4.6.3 Contribució de la llum especular

Per saber quant contribueix la llum especular en el punt que estem tractant, necessitem saber quanta llum reflectida li arriba a l'observador, per calcular-ho necessitem trobar l'angle que formen el vector del reflex de la llum i el vector que va del nostre punt a la posició de l'observador. Per trobar el valor d'aquest últim vector, restarem les coordenades de la posició del punt a les de la posició de l'observador. Per trobar el valor del vector del reflex de la llum, utilitzarem el vector que va de la llum al nostre punt i la funció *reflect* que donats dos vectors calcula el vector simètric del primer agafant com a eix de simetria el segon.

Un cop calculats els dos vectors, només ens faltarà calcular el cosinus de l'angle que formen per saber en quina mesura el raig reflectit va més o menys directament cap a l'observador. Un cop més ho farem utilitzant el producte escalar, així obtenim la següent fórmula:

$$ContribucioEspecular = 0.2 * max(R \cdot V, 0) * Intensitat * AtenuacioLlum \quad (13)$$

On R és el vector direcció normalitzat del raig de llum reflectit i V és el vector normalitzat que té direcció cap a l'observador partint del punt que estem tractant. El valor 0,2 és el coeficient que indica quin pes té la llum especular al color total del píxel.

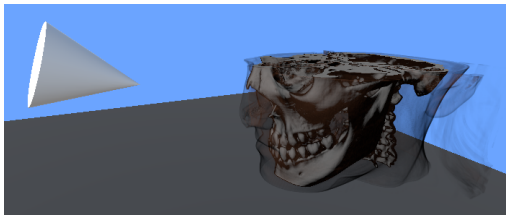


Figura 17: Model il·luminat només amb llum especular.

4.6.4 Atenuació de la llum

El model d'il·luminació del qual partim no té en compte si hi ha obstacles que facin que la llum no pugui arribar al vòxel (ombres) del qual estem calculant la il·luminació. Això provoca que posicions on la llum no arriba, per exemple, l'interior d'un volum perfectament tapat, estiguin il·luminades. Tampoc té en compte si hi ha teixits no opacs que facin que la llum s'atenuï abans d'arribar al vòxel.

Per tenir un model d'il·luminació realista és necessari calcular les ombres i l'atenuació de la llum, per tant, en el nostre ho fem. Per fer-ho, utilitzem una tècnica utilitzada en el *raytracing*. Des del punt del qual estem calculant el color es traça un raig en direcció a les diferents llums i es comprova per cada una quanta opacitat acumulen els obstacles que es troben, així trobem l'atenuació de la llum, és a dir, la quantitat de llum que es perd perquè hi ha obstacles entre el punt i la font d'il·luminació. Per calcular l'opacitat acumulada utilitzarem la fórmula explicada anteriorment, recordem:

$$OpacitatAcumulada = OpacitatAcumulada + (1 - OpacitatAcumulada) * OpacitatSuperficieActual \quad (14)$$

Aquesta opacitat acumulada ens servirà per saber quina quantitat de llum arriba en un punt, millor dit, en quina mesura s'atenua la llum que arriba en un punt a causa dels obstacles que hi ha entremig. El valor d'aquesta atenuació serà un valor entre 0 i 1 que calcularem de la següent manera:

$$AtenuacioLlum = 1 - OpacitatAcumulada \quad (15)$$

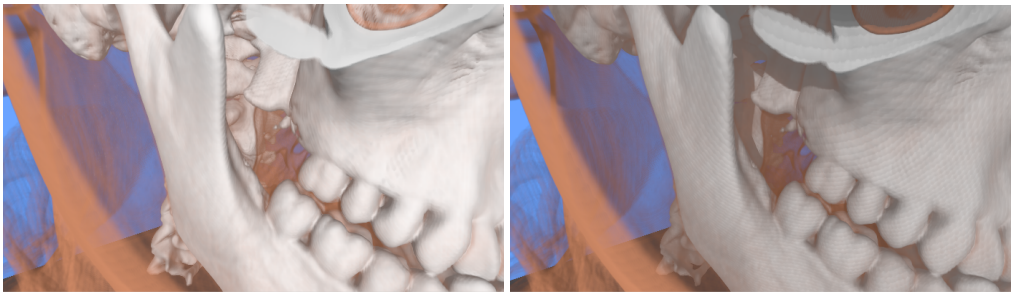


Figura 18: Imatges d'una escena on la pell del volum és poc transparent. A la imatge de l'esquerra no es té en compte l'atenuació de la llum i a la de la dreta si.

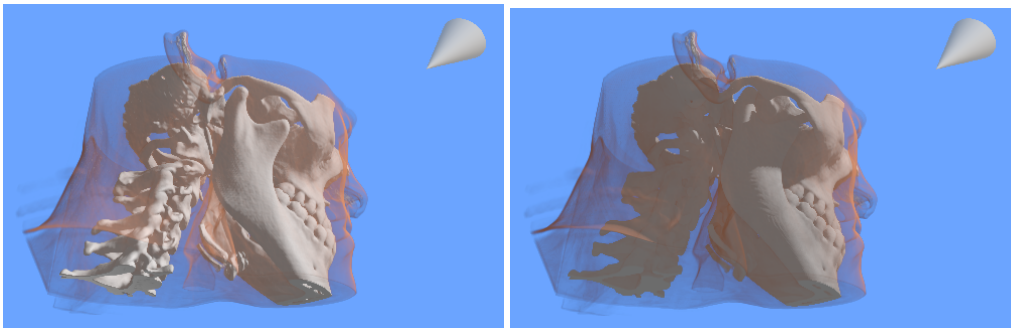


Figura 19: A l'esquerra una escena il·luminada utilitzant el model Phong, a la dreta, la mateixa escena però utilitzant el nostre model d'il·luminació. Es pot apreciar que, l'interior del volum i la columna vertebral, on no arriba llum directa, en el model Phong segueixen estant il·luminats mentre en el nostre no.

4.6.5 Intensitat de la llum

Fins ara utilitzàvem llums puntuals omnidireccionals, és a dir, llums que tenien una posició concreta que era un punt a l'espai. A l'hora de calcular quanta il·luminació aportaven aquestes llums no necessitàvem res més que saber la posició, ja que suposàvem que aquestes llums il·luminaven en totes les direccions. Per fer un model més realista, hem de tenir en compte que les llums, excepte en casos molt concrets com podria ser el Sol, són direccionals, és a dir, estan enfocant cap a un lloc, per exemple, una llanterna, un cotxe, la llum d'un cirurgià, etc. És per això, que aquest model té en compte la direcció de les llums també, com menys enfocat estigui un punt, menys il·luminació rebrà.

Un cop més, per calcular la quantitat d'il·luminació que arriba a un punt en concret tindrem en compte dos vectors: el vector que va des de la llum fins aquest punt i el vector direcció de la llum, és a dir, cap a on

apunta. Com més similars siguin aquests vectors, més llum rebrà el punt que estem tractant. Per això, tornarem a utilitzar el producte escalar, de manera que la intensitat d'una llum serà:

$$Intensitat = max(L \cdot DL, 0) \tag{16}$$

On L és el vector normalitzat que va de la llum al punt i DL és el vector direcció normalitzat de la llum.

Si recordem en l'apartat 4.2.2 s'ha dit que les fonts de llum tenen un angle d'obertura. Fent la potència del valor obtingut del producte escalar podem fer que aquest angle sigui més o menys gran, si l'exponent és gran, la il·luminació se centrarà bastant en el punt central (angle petit) i els voltants seran més foscs, en canvi, si l'exponent és petit, els voltants estaran més il·luminats (angle gran).

$$Intensitat = pow(max(L \cdot DL, 0), exp) \tag{17}$$

On exp és l'exponent.

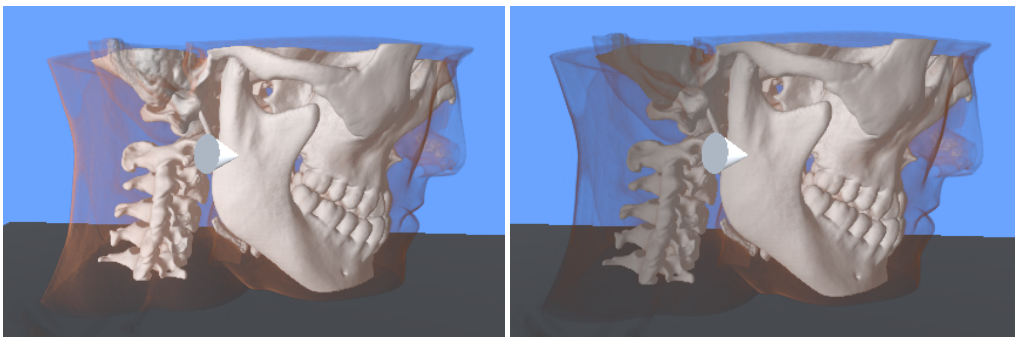
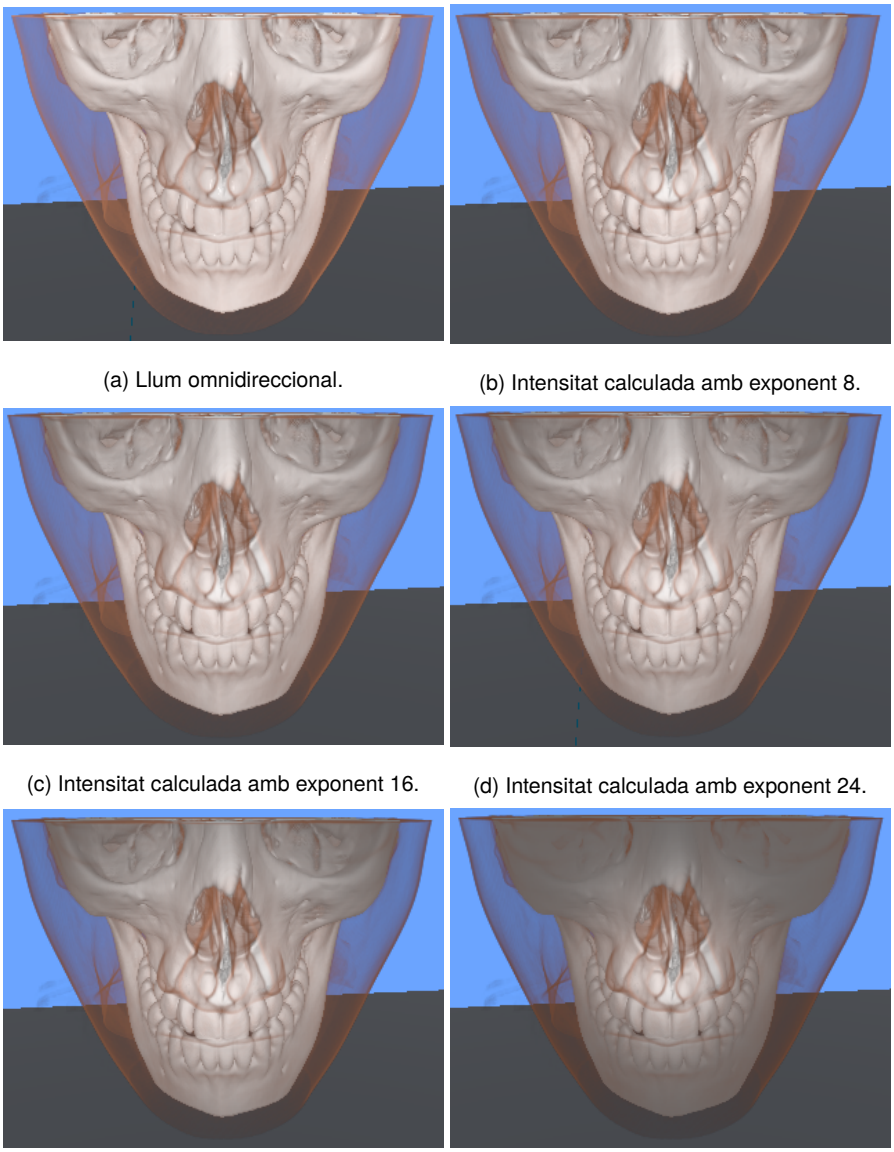


Figura 20: Diferència de la visualització d'un volum amb una llum omnidireccional (esquerra) i amb una llum direccional (dreta). Recordem que el con representa la font de llum i la seva cúspide la direcció. S'ha reduït la mida del con que representa la llum per no obstaculitzar tant la imatge.



(a) Llum omnidireccional.

(b) Intensitat calculada amb exponent 8.

(c) Intensitat calculada amb exponent 16.

(d) Intensitat calculada amb exponent 24.

(e) Intensitat calculada amb exponent 32.

(f) Intensitat calculada amb exponent 100.

Figura 21: Diferència de la visualització d'un volum amb una llum omnidireccional (imatge de dalt a l'esquerra) i amb una llum direccional (la resta) on la intensitat es calcula amb exponents diferents. Es pot observar que com major és l'exponent, menor és l'angle d'obertura.

4.6.6 Il·luminació final

Per acabar, només ens farà falta acumular el color que acabem de calcular al valor final.

El pseudocodi del càlcul de color quedaria així:

```
calcularColor(infoVoxel){
      L = infoVoxel.posicio - llumActual.posicio
      DL = llumActual.direccio
      N = infoVoxel.gradient
      R = reflect(L, N)
      V = infoVoxel.posicio - observador.posicio

      opacitatAcumulada = raigCapALlum(infoVoxel.posicio)
      atenuacioLlum = 1 - opacitatAcumulada
      Intensitat = max(producteEscalar(L, DL), 0)

      ContribuacioAmbient = 0.15
      ContribucioDifusa = 0.65 * max(producteEscalar(N, L), 0) * Intensitat *
           AtenuacioLlum
      ContribucioEspecular = 0.2 * max(producteEscalar(R, V), 0) * Intensitat
           * AtenuacioLlum

      Color Punt = Color * (ContribucioAmbient + ContribucioDifusa +
          ContribucioEspecular)
}
```

on:

```
raigCapALlum(posicioVoxel){
   posicioActual = posicioVoxel
   opacitatAcumulada = 0
      direccioRaig = posicioLlum - posicioVoxel
      pas = direccioRaig * minIncrementTextura
      while(not foraDelVolum opacitatAcumulada < 1){
            posicioActual = posicioActual + pas
            infoVoxel = calcularInfoVoxel(posicioActual)
            if(infoVoxel.esTeixit){
                  opacitatAcumulada += infoVoxel.opacitat
            }
```

```
12          }
13          return opacitatAcumulada
14  }
```

On minIncrementTextura és la distància mínima que es pot recórrer en coordenades de textura per no saltar-se cap vòxel.

Hem de tenir en compte que poden haver-hi diverses llums. En aquest cas, la llum ambient seguirà sent un 15% del color final i la contribució difusa i especular es calcularan per cada llum i se sumaran al color final.

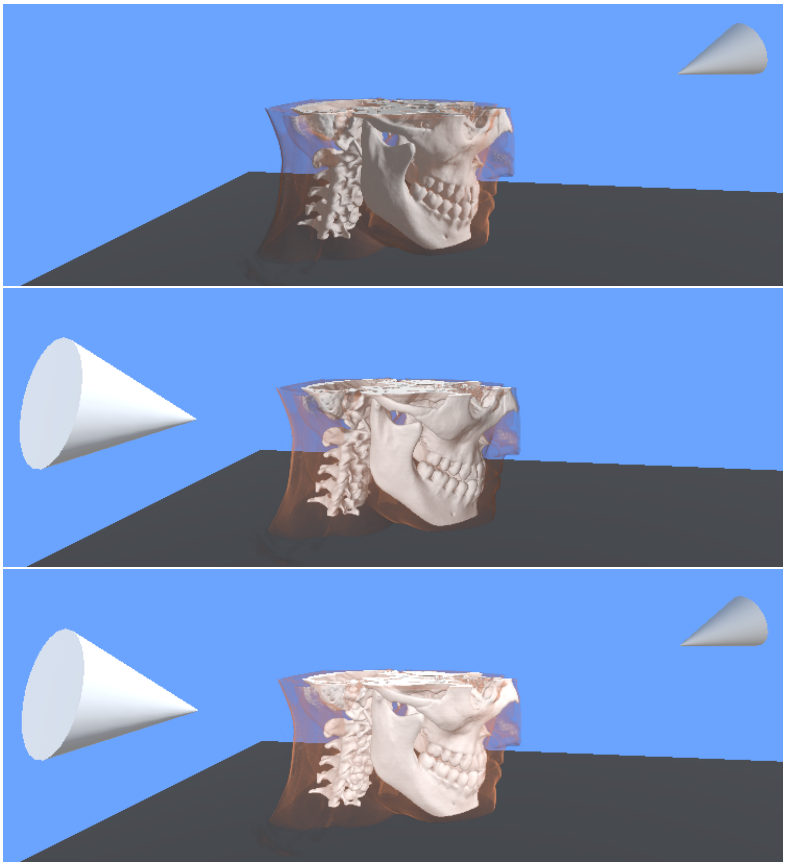


Figura 22: Escena visualitzada utilitzant el model Phong. En la imatge superior podem observar la il·luminació del volum amb una llum, en la del mig podem observar el mateix volum amb una altra llum i finalment podem observar la il·luminació del volum amb les dues llums anteriors.

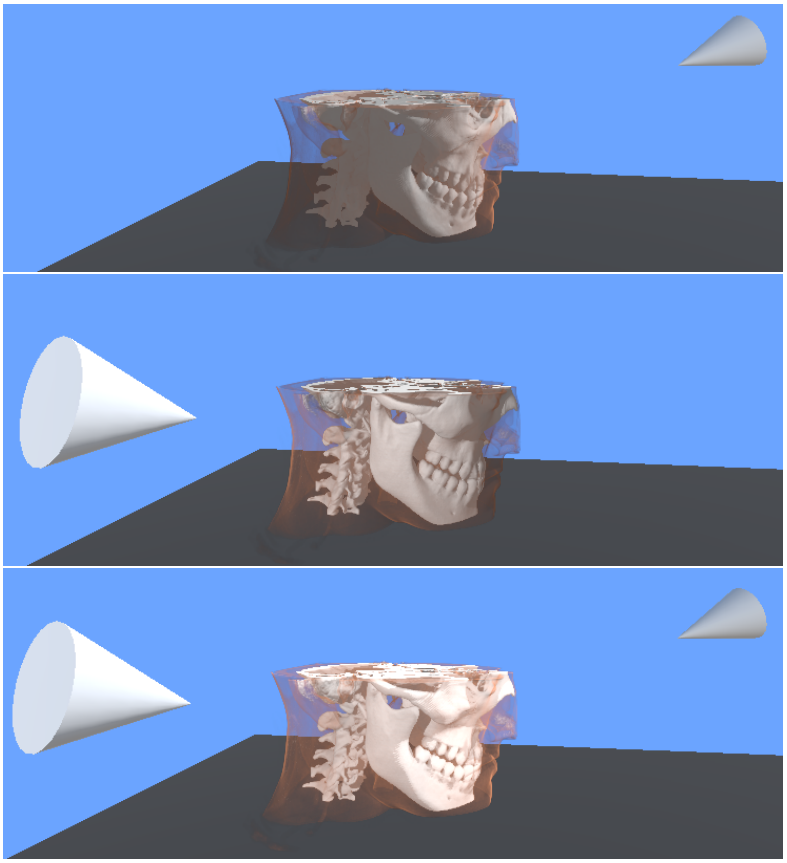


Figura 23: Escena visualitzada utilitzant el nostre model d'il·luminació. En la imatge superior podem observar la il·luminació del volum amb una llum, en la del mig podem observar el mateix volum amb una altra llum i finalment podem observar la il·luminació del volum amb les dues llums anteriors.

4.6.7 Càlcul del color del píxel

Com ja hem dit, per calcular el color de cada píxel llençarem un raig des de la posició de l'observador, en cas que aquest trobi una superfície, calcularem el color generat en aquell punt de la superfície de la manera que acabem d'explicar. Acumularem el color calculat, tenint en compte l'opacitat de la superfície per saber quant contribueix i, en cas que l'opacitat sigui inferior a 1, seguirem avançant el raig fins, o bé trobar una altra superfície, o bé fins que el raig surti del volum. Finalment, el color i l'opacitat acumulats determinaran el color del píxel. Si l'opacitat del píxel no és del 100%, es tindrà en compte el color de fons de l'escena.

Els resultats finals del nostre model de il·luminació seran els següents:

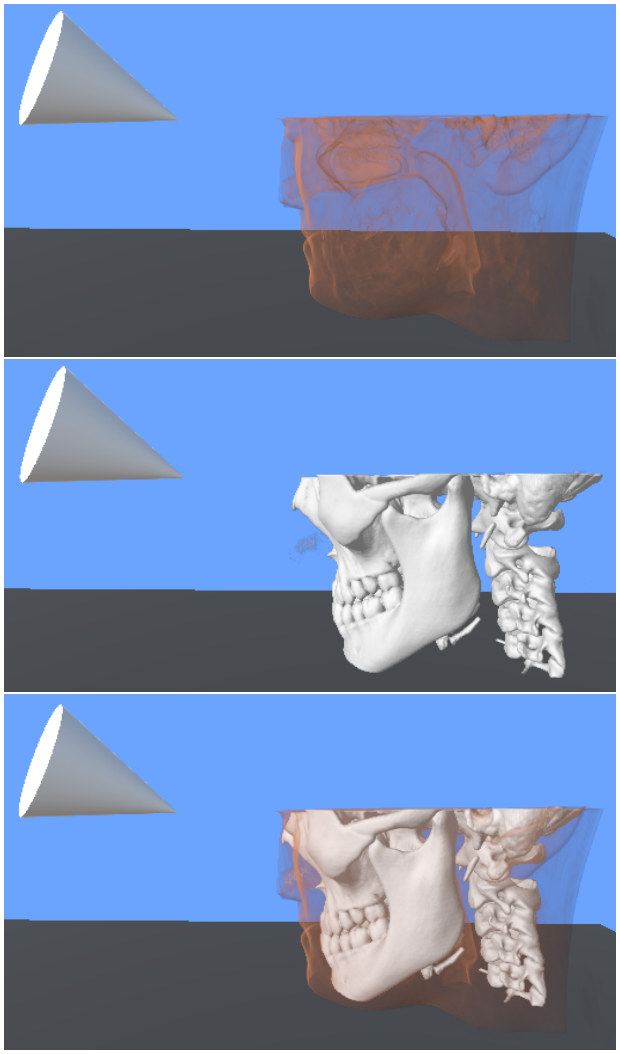


Figura 24: Visualització d'un model volumètric utilitzant el model d'il·luminació Phong amb una llum, a la imatge de dalt calculant només el color teixit de la pell, a la del mig calculant el color de l'os i a la de baix el color total de pell i os.

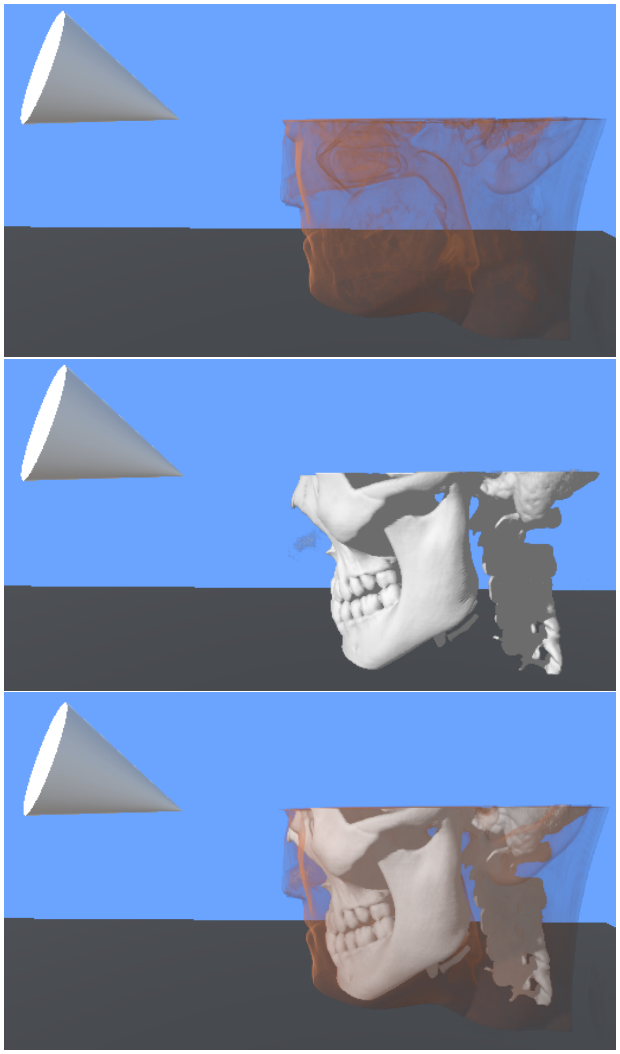


Figura 25: Visualització d'un model volumètric utilitzant el nostre model d'il·luminació amb una llum, a la imatge de dalt calculant només el color teixit de la pell, a la del mig calculant el color de l'os i a la de baix el color total de pell i os.

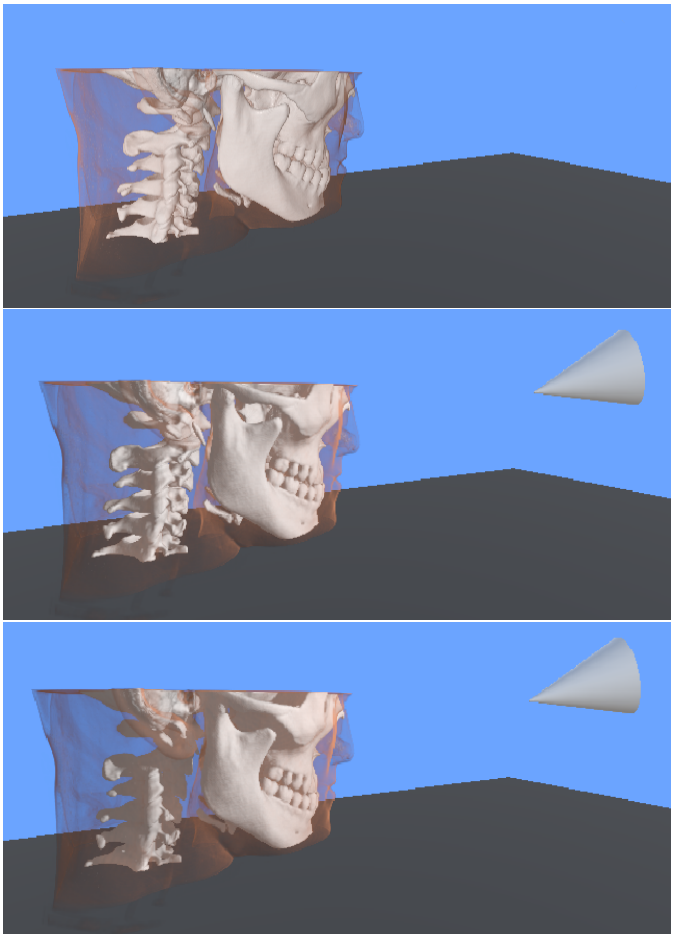


Figura 26: Diferència entre els diferents models d'il·luminació amb una única font llum. A la primera imatge utilitzant el mètode del qual partim (Phong amb una llum a la posició de l'observador), a la segona, utilitzant el model Phong amb la llum (con) en una posició fixe, a l'última, el nostre model final d'il·luminació. En les dues primeres imatges la tècnica del càlcul d'il·luminació és la mateixa, però els resultats no són iguals perquè les fonts de llum estan en posicions diferents. En la tercera imatge en comparació a la segona podem veure com les zones on no arriba llum directa estan menys il·luminades.

5 Posada en pràctica

A l'hora de posar en pràctica el nostre model de llum, primer era necessari entendre el codi ja existent i saber modificar-lo, per això les primeres modificacions van ser el més senzilles possible i després van anar augmentant de complexitat.

5.1 Escena amb una llum

La primera modificació estava destinada a començar a entendre el codi base que seria modificat i saber com treballar amb ell. Aquest codi ens permet visualitzar un model mèdic en Realitat Virtual o simplement per la pantalla del nostre dispositiu. La seva il·luminació consisteix en un model Phong on l'única llum que hi ha està col·locada a l'observador, és a dir, la il·luminació es calcula simulant que a la posició de l'observador hi ha una llum.

Partint d'aquí, es volia modificar aquest codi per tal que la llum en lloc d'estar en l'observador estigués a un lloc estàtic, però que aquest es pogués modificar durant l'execució. Per dur a terme això es van realitzar les següents modificacions:

- Per poder veure els canvis en passar d'una il·luminació a una altra, en lloc de fer els canvis necessaris sobre el mateix codi, es va implementar que al prémer una tecla passés d'un mètode d'il·luminació a l'altre i viceversa.

- Per poder modificar la posició i direcció de la llum durant l'execució, cada cop que s'afegeix una llum es crea un objecte de Unity. Els objectes de Unity tenen tres components, la posició, la rotació i l'escala, que poden ser modificats durant l'execució des de la interfície. Això ens permet que, en associar cada llum amb un objecte, aquesta tingui posició, rotació i escala i aquestes dades puguin ser modificades des de la interfície de Unity. La posició d'aquest objecte es passa en cada *frame* al *shader* que la tracta com la posició de la llum, de manera que es pot percebre casi immediatament el resultat de moure una llum de posició.

  D'aquest objecte en podem extreure la posició en un script i enviar-la al *shader* utilitzant *Uniforms*, que són variables globals del *shader* que se'ls pot assignar un valor des del script. Cal matisar que els scripts estan implementats amb C# que per definir les posicions en els espais 3D utilitza variables del tipus Vector3, que, en aquest cas consisteix en un vector de floats amb 3 posicions i en canvi, el *shader* està implementat amb Cg, que utilitza float3. Tot i això, aquí no hi ha cap inconvenient, ja

que ens permet fer la transformació de variable sense cap problema. El *shader* utilitzarà la posició de la llum per calcular la il·luminació seguint el model Phong (prèviament explicat).

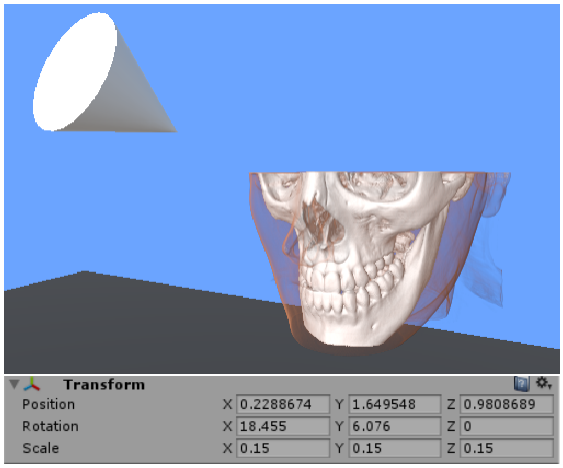


Figura 27: Escena amb il·luminació Phong i amb una font de llum. A sota podem veure la interfície que ens permet modificar les propietats de la font de llum: la posició, la direcció i la mida del con.

5.2 Escena amb múltiples llums

La segona modificació consistia en poder afegir diverses llums durant l'execució del programa. Hauria de poder-se modificar la posició i direcció de les llums durant l'execució. Per fer-ho es va decidir que al polsar la lletra 'j' s'afegiria una llum a la posició de l'observador i enfocant en la direcció de la càmera, i després en cas que fos necessari ja es modificarien des de la interfície d'Unity. A l'implementar aquestes modificacions ens vam trobar amb diversos obstacles, exposats a continuació:

- **Crear múltiples objectes de Unity:** En Unity per poder crear un objecte necessites donar-li un nom. Quan només hi havia una llum no hi havia cap problema, però a l'haver-ne de crear múltiples, totes les llums tenien el mateix nom, i a l'hora d'obtenir la informació de les diverses llums per enviar-la cap al *shader* era impossible, ja que només es podia agafar la informació d'una llum. Per solucionar això, es creen les llums amb el nom "Llum X" on X és un número, anant de l'1 endavant. Per crear-les només hem de fer un bucle on hi ha una variable que comença a 1 i va augmentant i dins el bucle es va comprovant si existeix una llum amb el nom Llum seguit del número de la variable i, en el cas que no existeixi es crea. Per poder enviar aquestes dades al *shader*, es llegeixen en un altre bucle.

- **Passar la informació al *shader*:** Per passar la informació al *shader*, al principi s'havia pensat de

fer-ho mitjançant un array de Vector3 o float3, però Cg no permet fer-ho. Per això es va decidir fer-ho utilitzant dos arrays de *floats*, que per cada llum tindrien 3 posicions, els valors d'aquests després seran agafats de 3 en 3 i tractats com a float3. Un altre obstacle va ser el fet que Cg no permet arrays dinàmics, és a dir, arrays on es puguin anar afegint i eliminant elements. Per això, per poder-li passar els arrays al *shader* és necessari que aquests arrays tinguin una mida concreta. Per això es va limitar el nombre de llums a 10 i establir la mida del vector a 30 elements. Per saber quants elements del vector hauria de tractar com a llums el *shader* només va ser necessari passar-li un *uniform* que indiqués quantes llums hi ha actives a l'escena.

Un cop aconseguida implementar la tasca de permetre afegir múltiples llums, ja teníem els coneixements base per començar a desenvolupar la il·luminació final.

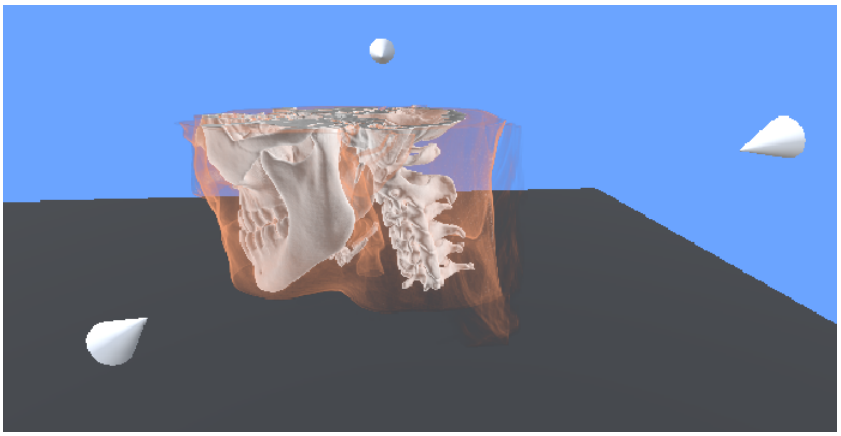


Figura 28: Escena il·luminada amb el model Phong on es calcula la il·luminació tenint en compte tres fonts de llum.

5.3 Il·luminació final

Per saber els punts del volum que estem veient, com ja hem explicat, els trobarem utilitzant la tècnica prèviament explicada anomenada *raycasting*. Un cop trobats calcularem la il·luminació que arriba a cada punt com hem explicat a l'apartat 4.6. A l'hora d'implementar aquesta il·luminació ens hem trobat diversos obstacles exposats a continuació:

- **Bounding box:** A l'hora de calcular l'atenuació de la llum es llença un raig per comprovar els obstacles que hi ha. Per comprovar si el raig cap a la llum era obstaculitzat el que fèiem era avançar en coordenades de textura i comprovar si hi havia vòxels amb gradient no nul per saber si havíem

trobat una superfície, en cas de trobar-nos amb una superfície miràvem l'opacitat i seguíem avançant. Quan duem això a la pràctica ens adonem que no es visualitza correctament.

Com ja hem dit, es treballa amb una gran quantitat de dades, i per poder visualitzar el volum en temps real, no acostumem a tractar tot el volum alhora, ja que disminueix la *frame rate*. Per això, des de la interfície d'Unity podem modificar la *bounding box* (caixa contenidora), per decidir quin tros del volum volem veure.

En comprovar el resultat ens adonem que hi ha un error. Per més que hi hagi una font de llum il·luminant-los directament, els límits del volum que estan tallats a causa de la *bounding box* no estan il·luminats. Això és degut al fet que, a l'utilitzar les coordenades de textura per valorar si un obstacle obstrueix la llum o no, no tenim en compte que el tros del volum que veiem no és el mateix que l'especificat en les coordenades de textura, ja que aquestes representen el volum sencer i no el delimitat per la *bounding box*.

Per arreglar això només feia falta afegir una condició que comprovés si el raig traçat sortia de la *bounding box* i, en cas que si, parés d'analitzar els obstacles que hi ha entremig, ja que aquests en la nostra escena no són visibles. Per fer això vam haver de passar les delimitacions de la *bounding box* al *shader* mitjançant *Uniforms*.

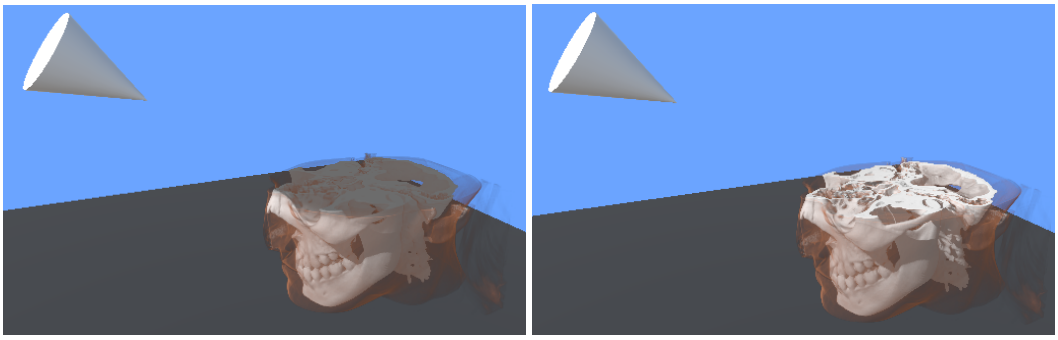


Figura 29: A l'esquerra podem veure una imatge d'abans de corregir l'error de la *bounding box*, a la dreta la mateixa escena sense l'error.

- **Ones:** Un cop vam haver implementat el càlcul de les ombres, ens vam adonar que el resultat no era l'esperat, sinó que era bastant diferent, veiem una imatge com la de la figura 30. Quan aquesta imatge va ser capturada, hi havia una llum posicionada davant del nostre volum.

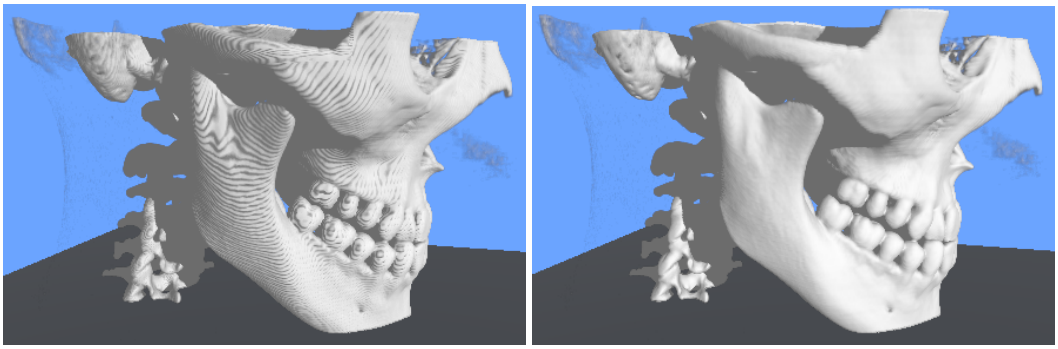


Figura 30: A l'esquerra podem veure una imatge d'abans de corregir l'error de les ones, a la dreta la mateixa escena sense l'error.

Com es pot veure, les ombres de la part oculta de la llum es calculaven bé, però, a la part on la llum incidia vèiem unes ombres amb una forma que ens recordava a la d'unes ones. Aquestes "ones" eren generades pel fet que, en traçar un raig cap a cada llum, començàvem des d'una posició molt propera a la del punt del qual estàvem calculant el color, això feia que en comprovar quins obstacles atenuaven la llum, a vegades es tingués en compte la mateixa superfície on érem com a obstacle. Això ho vam solucionar fent que el raig comencés a traçar-se una mica més endavant i no directament des del punt. Aquesta distància extra havia de ser prou gran per no generar errors però el més petita possible per tenir en compte tots els obstacles. Per calcular la posició inicial del raig ho fem de la següent manera:

$$posicioInicial = posicioVoxel + pas * 10 \tag{18}$$

On pas es calcula igual que al codi de l'apartat 4.6.6.

- **Frame rate:** Un altre problema, aquest no va poder ser solucionat, va ser el del *frame rate*. Un cop implementat el càlcul de les ombres, el *frame rate* es va veure reduït dràsticament. Per cada punt del qual volem calcular el color, hem de traçar un raig per cada llum que hi ha a l'escena. Per tant, per cada píxel del qual volem calcular el seu color, hem de traçar un raig per trobar els punts que influeixen en el color i per cada punt d'aquests, un raig per cada llum. Això augmenta considerablement el nombre de rajos que hem de traçar, augmentant així el temps de càlcul, reduint així els *frames per second* (fps) i fent que l'usuari no pugui interaccionar a temps real amb el nostre software.

A continuació fem una anàlisi de rendiment modificant la mida del volum, la mida del *viewport* i la distància de l'observador fins al centre del volum. Les dades han sigut obtingudes fent la mitja de més de 30 valors que indiquen els fps. Per fer-ho s'ha utilitzat l'ordinador personal especificat en l'apartat 7.2.2.

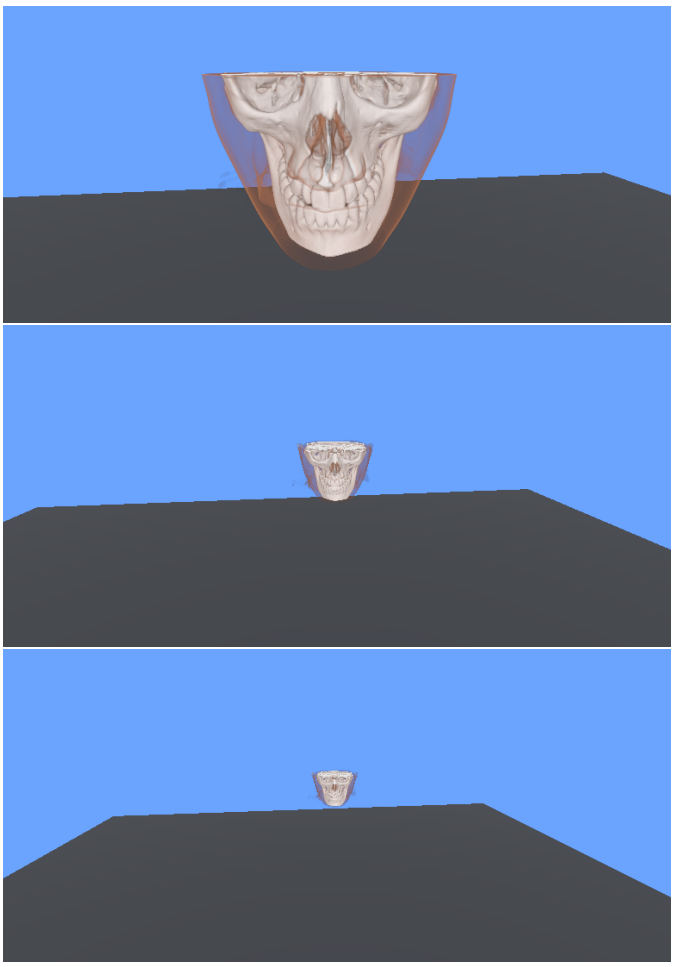


Figura 31: Quan parlem de la distància de l'observador fins al centre del volum, els valors d'aquesta no tenen unitat de mesura, per això us deixem tres imatges per comprendre millor els resultats de l'anàlisi de rendiment. En la imatge de dalt, la distància és d'1, en la del mig, la distància és de 3 i, en la de sota, la distància és de 5.

Average Frames Per Second

	1 llum		2 llums	
Mida del volum	**Phong**	**Nostre model**	**Phong**	**Nostre model**
512x512x256	64.9	0.09	64.5	0.04
400x400x200	97.7	0.33	93.8	0.18
200x200x100	175.7	11.10	175.3	5.15
100x100x50	183.2	77.12	182.7	43.27

Taula 1: Rendiment del nostre model d'il·luminació en comparació amb el model Phong a una distància de 1,5 i amb un *viewport* de 782x415 píxels.

Average Frames Per Second

	1 llum		2 llums	
Mida del viewport	**Phong**	**Nostre model**	**Phong**	**Nostre model**
782x415	116.1	0.43	111.4	0.23
564x300	126.6	0.47	125.4	0.25
376x200	176.7	0.69	173.7	0.40
188x100	178.0	6.96	178.0	3.38

Taula 2: Rendiment del nostre model d'il·luminació en comparació amb el model Phong a una distància de 1,5 i amb un volum de mida 323x323x179

Average Frames Per Second

	1 llum		2 llums	
Distància	**Phong**	**Nostre model**	**Phong**	**Nostre model**
1	102.5	0.34	78.7	0.15
2.5	163.9	0.56	150.6	0.29
4	171.0	1.50	164.2	0.32
5.5	174.8	3.07	170.4	0.41

Taula 3: Rendiment del nostre model d'il·luminació en comparació amb el model Phong amb un *viewport* de 782x415 píxels i amb un volum de mida 323x323x179.

La lentitud a l'hora d'interaccionar ja genera una experiència desagradable en utilitzar el software des d'un ordinador en mode simulació. Si en lloc d'utilitzar-lo en un ordinador en mode simulació ho fem des d'un dispositiu HTC Vive, la situació empitjora.

5.4 Interacció amb l'usuari

Perquè l'usuari tingui una millor experiència amb aquest software, no només fa falta tenir una il·luminació el més realista possible, també fa falta que pugui controlar aquesta il·luminació i transformar-la al seu gust el màxim possible. És per això que per tenir una bona il·luminació ens fa falta permetre a l'usuari col·locar les llums que vulgui i com vulgui. Part d'aquest treball es centra en això, en aquest apartat explicarem les facilitats que s'han implementat.

L'usuari experimentarà el nostre software amb el HTC Vive i, com hem explicat al principi, aquest és un sistema de Realitat Virtual immersiu, és a dir, l'usuari no veurà més enllà del que li mostrarem per pantalla. Això provoca que no vegi el teclat ni el ratolí, per tant, és important que pugui modificar les llums a través del hardware del HTC Vive, utilitzant els controladors, aquests sí que els veurà per pantalla.

Per explicar les facilitats que hem implementat i saber-les utilitzar primer hem de conèixer el hardware, a la figura 32 podem veure un esquema dels controladors dels quals disposem. Per poder treballar amb les llums d'una manera intuïtiva i còmode només necessitarem utilitzar dos botons: el *Trigger* i el *Trackpad*. A continuació desenvoluparem les funcions que els hi hem donat a cada un.

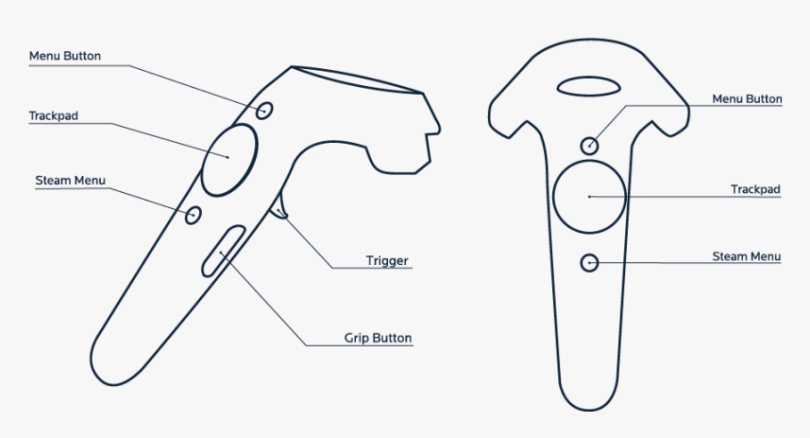


Figura 32: Botons dels controladors del HTC Vive.

5.4.1 Afegir i eliminar llums

Com ja hem explicat, el nostre software permet calcular la il·luminació d'una escena amb diverses fonts de llum. És necessari doncs, que l'usuari pugui afegir i eliminar llums quan vulgui. Li permetrem fer-ho d'una manera senzilla.

Utilitzant el controlador i polsant el botó *trackpad*, l'usuari podrà afegir una llum. Aquesta s'afegirà a la posició del controlador i tindrà la direcció d'aquest també. Per eliminar una llum haurà de polsar el mateix botó però posicionant el controlador prop de la llum que vol eliminar.

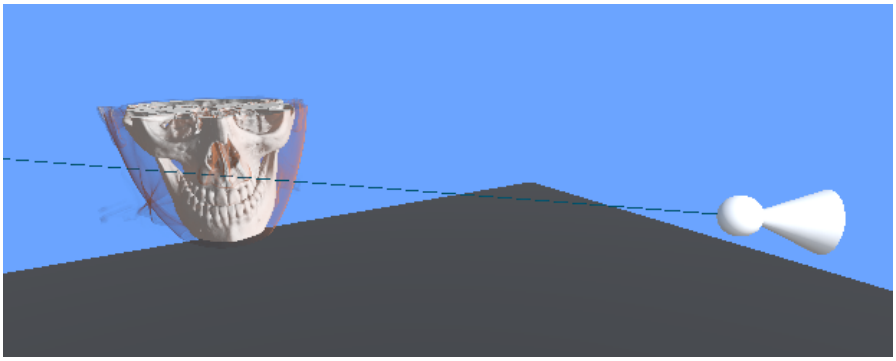


Figura 33: Llum recentment afegida, l'esfera representa el controlador, el con representa la llum.

5.4.2 Desplaçar llums

Per desplaçar les llums utilitzarem el botó *trigger*. Per poder-les desplaçar, haurem de posicionar un altre cop el controlador prop de la llum que volem desplaçar, per fer-ho intuïtiu, no només s'ha de polsar el *trigger* sinó que s'ha de mantenir polsat fins que tinguem el controlador al lloc on vulguem deixar la llum. Aleshores, un cop deixem anar el botó, la llum es quedarà a l'última posició a la qual estava el controlador quan el *trigger* estava polsat i en la direcció en la qual estava mirant.

En el següent apartat s'explica com és la interacció amb l'usuari perquè aquest pugui saber quan pot eliminar i modificar llums i quan les està desplaçant.

5.4.3 Altres facilitats

Per poder controlar les llums que hi ha a l'escena, necessitem que aquestes siguin visibles, ja que, si només fan la funció de llum però no les podem veure, a l'hora de treballar amb elles ens serà impossible, ja que no sabrem on són. És per això que se les ha fet visibles. Podrien ser representades, per exemple

com una esfera, però com ja hem dit, les llums les tractem tenint en compte la seva direcció, per tant ens interessa saber cap a on estan apuntant, és per això que les representem en forma de con, on la cúspide indica la direcció.

En les diferents facilitats, hem vist que haurem de posicionar el controlador prop de la llum per poder-la eliminar o desplaçar. Però, com sap l'usuari si està suficientment prop d'una llum? Perquè aquest ho pugui saber, en apropar el controlador a una llum i quan aquest estigui dins el rang per modificar-la, aquesta canviarà la seva mida, fent-se més gran i, per tant destacant més entre les altres. Així, en casos que hi hagi diverses llums en posicions semblants sabrem quina estem a punt de modificar. En casos com aquest amb diverses llums properes al controlador, la llum que destacarà i per tant la que modificarem en cas de polsar un dels botons serà la més propera al controlador.

Quan estem mantenint el *trigger* polsat per modificar la posició i direcció d'una llum, també volem tenir la certesa que realment l'estem desplaçant i que no hem perdut el seu control, és per això, que mentre estiguem desplaçant una llum, aquesta es farà més petita i es mourà conjuntament amb el controlador. Al fer-se més petita, no només sabrem que l'estem desplaçant sinó que també ens ajudarà a triar amb més precisió la posició on la volem.

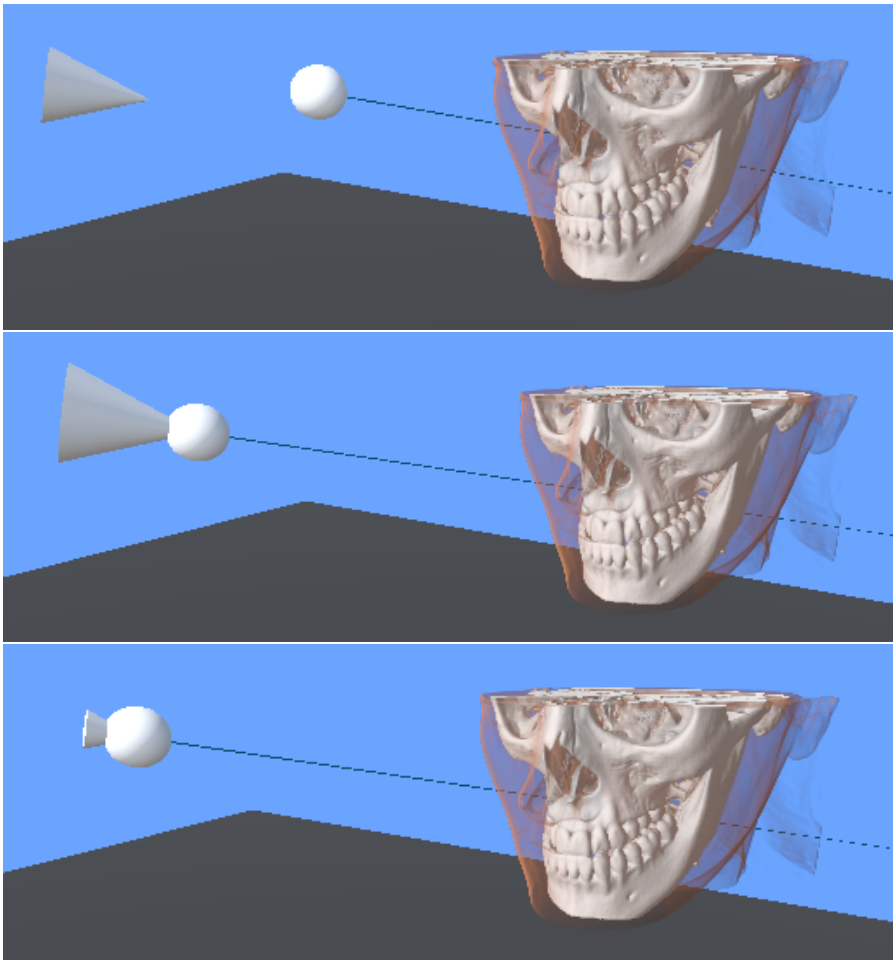


Figura 34: En la primera imatge, una font de llum en la seva mida normal. En la segona, una font de llum quan el controlador està suficientment aprop per modificar-la. En la tercera, una llum sent modificada de posició i direcció utilitzant el controlador.

Una de les altres facilitats que preteníem implementar, era permetre utilitzar el controlador com a llanterna. A l'implementar el que s'ha explicat en aquest apartat, ens vam adonar que no era necessari, ja que l'usuari podia obtenir una llanterna fàcilment, simplement havia de crear una llum i agafar-la, un cop fet això, el comandament s'havia transformat en una llanterna. Com que la llum es fa petita durant el desplaçament, no obstrueix la visió de l'usuari si aquest l'utilitza com a llanterna.

6 Conclusions

En aquest projecte hem ampliat un software ja existent que permet visualitzar models mèdics en entorns 3D immersius. Ho hem fet habilitant la interacció amb les fonts de llum per part de l'usuari i implementant un model d'il·luminació més realista.

Els resultats de la millora en la interacció de l'usuari han sigut els esperats. Aquests permeten a la persona que utilitza el nostre software triar un dels tres models d'il·luminació que hi ha implementats (una llum en la posició de l'observador amb el model Phong, diverses llums a l'escena amb el model Phong i diverses llums amb el model d'il·luminació explicat en l'apartat 4.6). També li permeten afegir i modificar les diferents fonts de llum al seu gust i utilitzar un controlador com a llanterna. Això permet als usuaris gaudir d'una experiència més interactiva, també els permet preparar la il·luminació que millor els vagi per l'anàlisi dels seus models.

Els resultats del model d'il·luminació que hem implementat no són exactament els esperats. Per una banda, ja que no s'han pogut implementar totes les tècniques a causa de la incompatibilitat del nostre software amb el qual volíem utilitzar, per altra banda, perquè la magnitud del càlcul provoca que no es pugui tenir un *frame rate* adequat. Tot i això, la visualització del model volumètric amb aquest model d'il·luminació és més realista que l'anterior.

Per tant, s'ha millorat la interacció amb l'usuari, un dels principals objectius del treball. Malauradament, tot i que també s'ha millorat la il·luminació dels models, no s'ha obtingut l'eficiència esperada al fer-ho, en tot cas, el càlcul de la il·luminació Phong amb diverses llums es pot mantenir i el model d'il·luminació final també, però utilitzar-lo només en casos específics on la interacció no sigui l'objectiu principal o en casos que el model sigui suficientment petit. En un futur, quan el hardware sigui més potent, és probable que es pugui utilitzar interactivament.

Si es volgués seguir desenvolupant aquest projecte, per la part de la il·luminació es podrien aplicar tècniques utilitzades en l'Exposure Render. També es podria estudiar la manera de fer el traçat de rajos més eficient per reduir el cost d'execució. Per la part d'interacció amb l'usuari es podria habilitar a aquest canviar de model d'il·luminació mitjançant el controlador, ja que actualment per fer-ho ha d'utilitzar el teclat.

7 Planificació i gestió del projecte

Aquest projecte va ser planificat abans de començar la posada en pràctica. Això va fer que no es tinguessin en compte diferents factors que en un futur van fer modificar les nostres tasques i amb elles la planificació d'aquest projecte.

7.1 Planificació del projecte

A l'hora de planificar el projecte, ens trobàvem davant d'un treball extens i, a causa de la complexitat i la poca flexibilitat que hi hauria si es dugués a terme sense organitzar-lo, es va decidir que es planificaria dividint-lo en diverses tasques. Van ser les següents:

Treball previ: Que incloïa tant la formació bàsica, que obtindríem amb la lectura del llibre "Real-Time Volume Graphics" i d'altres documents, com la formació concreta sobre el conjunt d'algorismes i tècniques que volíem implementar.

Gestió del projecte: Que englobava la planificació del projecte, és a dir, reunions, documentació, l'abast, la planificació, el pressupost i l'informe de sostenibilitat. També englobava la redacció de la memòria, la preparació i la realització de la presentació.

Comprensió del codi: Com que partíem d'un codi ja existent, era necessari entendre'l per saber treballar sobre d'aquest, és per això que primer s'estudiaria i es comprovaria que s'hagués entès correctament fent petites modificacions i comprovant els resultats.

Primeres modificacions: Consistia en la implementació del càlcul d'il·luminació de diverses fonts de llum en diferents llocs. Primer s'implementaria i després es comprovaria i corregiria perquè funcionés correctament.

Per comprovar que aquesta implementació no hagués afectat al *frame rate* ni a la fluïdesa de la interacció de l'usuari amb el nostre software, es faria un test d'usabilitat.

Comandament com a llanterna: El codi permet la visualització de models mèdics mitjançant Realitat Virtual. Al Centre de Realitat Virtual, on s'ha desenvolupat el projecte, podem trobar diversos HTC Vive (Ulleres de Realitat Virtual) amb els seus respectius controladors. La finalitat d'aquesta tasca al principi era implementar que un dels controladors permetés col·locar fonts d'il·luminació, amb la mateixa posició i direcció del comandament quan l'usuari premés un botó. Un cop funcionés, la segona finalitat era que

aquest controlador es pogués utilitzar com una llanterna, és a dir, que generés il·luminació des del lloc on es trobés el controlador i en la seva direcció. Això, no només ens aportaria una nova funcionalitat interessant sinó que ens ajudaria a practicar com treballar sobre el codi a nivells més complexos que als que hauríem arribat amb les primeres modificacions.

Un cop acabat, faríem un altre test d'usabilitat per comprovar que fos còmode d'utilitzar per a l'usuari i que, no es veiés massa afectada la *frame rate* ni la fluïdesa de la interacció.

Implementació parcial del nostre model d'il·luminació: Aquesta tasca consistia en analitzar i implementar les tècniques que resultessin més senzilles. Després es farien comprovacions per a comprovar que realment s'havien obtingut els resultats esperats, en el cas negatiu, es corregirien. També es millorarien els aspectes que es creguessin oportuns.

Implementació final del nostre model d'il·luminació: Finalment s'acabaria d'implementar i es faria un últim test d'usabilitat per acabar de polir tots els detalls possibles perquè el resultat fos el màxim usable possible.

7.1.1 Gestió dels possibles obstacles

El projecte consistia en definir, implementar i avaluar un nou model d'il·luminació. El principal risc que es corria al dur a terme aquest projecte era que els resultats no fossin els esperats i que per tant l'experiència en utilitzar el nostre software mitjançant Realitat Virtual no fos agradable, ja fos per la manca de fluïdesa o pel fet que el resultat no fos productiu a l'hora d'utilitzar aquesta tècnica en el camp de la medicina. Probablement aquest últim obstacle no ocorregués, ja que dotar als models d'una il·luminació realista ajuda a comprendre millor les distàncies.

No obstant això, en ser el principal objectiu l'avaluació d'aquesta tècnica, no es valoraria negativament que els resultats no fossin els esperats, ja que independentment dels resultats, aquest treball contribuiria en la comunitat científica.

Més tard ens vam adonar que hi havia un altre possible obstacle: que el software del qual disposava no fos compatible amb el llenguatge de programació que volíem utilitzar.

Vam aglutinar les tasques que havíem decidit fer en la següent taula i diagrama de Gantt, on també vam determinar la durada i la dependència de cada tasca.

Tasca	Durada	Dependències
1. Treball previ	**35 h**	
1.1. Formació bàsica	25 h	
1.2. Formació teòrica sobre les tècniques que volíem integrar	10 h	1.1
2. Gestió del projecte	**120 h**	
2.1. Planificació del projecte	32.5 h	
2.2. Redacció de la memòria	77.5 h	2.1.
2.3. Presentació	10 h	2.2.
3. Comprensió del codi	**20 h**	
3.1 Entendre el funcionament del projecte inicial	15 h	
3.2 Comprovar-ho fent petites modificacions	5h	3.1.
4. Primeres modificacions	**15 h**	
4.1. Implementació de les primeres modificacions	10 h	3.
4.2. Comprovació, correcció i millora de les primeres modificacions	5 h	4.1.
5. Comandament com a llanterna	**40 h**	
5.1. Implementació de la llanterna usant el comandament	35 h	3.
5.2. Comprovació, correcció i millora de la llanterna	5 h	5.1.
6. Implementació parcial del nostre model d'il·luminació	**100 h**	
6.1. Implementació parcial del nostre model d'il·luminació	80 h	3.
6.2. Comprovació, correcció i millora de la implementació	20 h	6.1.
7. Implementació final del nostre model d'il·luminació	**140 h**	
7.1. Implementació final del nostre model d'il·luminació	100 h	6.
7.2. Comprovació, correcció i millora de la implementació	40 h	7.1.
Total	**470 h**	

Taula 4: Tasques a realitzar durant el projecte.

7.1.2 Gantt

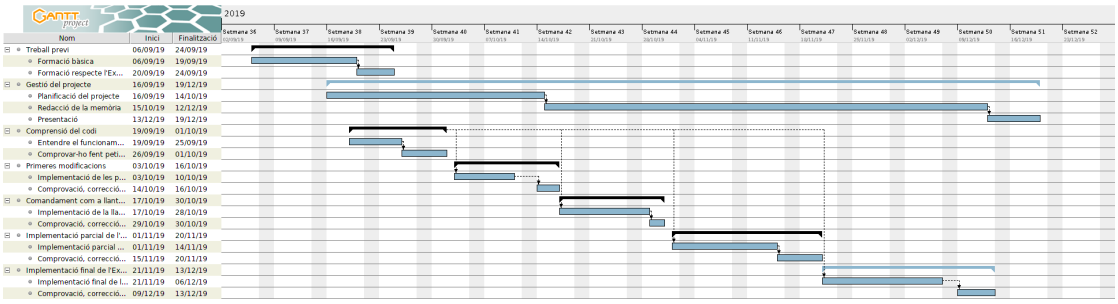


Figura 35: Diagrama de Gantt d'aquest treball.

7.2 Recursos

7.2.1 Recursos humans

Per a dur a terme aquest projecte han intervingut diverses persones que cal tenir en compte. Primer de tot, en partir d'un altre treball, hi ha un codi que ha estat desenvolupat per un estudiant i supervisat pel seu tutor/a i que posteriorment va ser modificat per gent diversa.

Per a realitzar el treball, han participat tres persones, en Pere-Pau Vázquez Alcocer que ha fet la funció de director del projecte. Després n'Eva Monclús Lahoya, co-directora del projecte, que ha fet un seguiment més presencial al lloc on s'ha realitzat el projecte, és a dir al Centre de Realitat Virtual. Per acabar jo que he realitzat la funció de programador.

7.2.2 Recursos hardware

També han sigut necessaris recursos hardware, pel treball s'han utilitzat els següents:

- HTC Vive amb dos comandaments.
- Ordinador connectat al HTC Vive amb sistema operatiu Windows 10 64 bits, processador Intel Core i7-6800K (3.4 GHz), targeta gràfica Nvidia GeForce GTX 970, memòria RAM de 32 GB.
- Ordinador per treballar a casa: Windows 10 Pro 64 bits, processador i7-7500U (2.9 GHz), targeta gràfica NVIDIA GeoForce 930MX, memòria RAM de 8GB.

7.2.3 Recursos software

En ser un projecte que partia d'un de més gran, s'utilitzaria Git, un sistema de control de versions. Més concretament s'utilitzaria l'eina anomenada GitLab, creant una branca amb el mateix codi del projecte complet i modificant-la, en acabar el treball, el responsable del projecte ajuntaria els codis en cas que es cregués convenient. Al treballar tant al CRV com remotament, aquesta eina ens ajudaria també a fer menys feixuc el traspàs de codi i les seves modificacions.

La plataforma la qual s'utilitzaria per implementar el nostre model d'il·luminació era Unity3D, utilitzant els llenguatges de programació C# i CG utilitzant HSL, una llibreria per realitzar computacions científiques. També s'anava a utilitzar Compute Unified Device Architecture (CUDA), una plataforma i llenguatge de programació per treballar amb computació paral·lela i accelerar així l'execució del codi, després vam descobrir la incompatibilitat de CUDA amb el nostre software.

Per acabar, la memòria seria escrita utilitzant LaTeX, un llenguatge de programació per desenvolupar treballs professionalment. Més concretament s'utilitzarà Overleaf, un editor de LaTeX online.

7.3 Pressupost

En aquest apartat es van identificar els diferents costs del projecte, tenint en compte els recursos que acabem de mencionar entre d'altres. Es van analitzar diversos elements, aquests eren els costs de recursos humans i costs indirectes. Finalment es van agrupar per fer el pressupost total del treball.

7.3.1 Costs en recursos humans

En aquest treball participarien principalment tres persones, el director, la codirectora i l'estudiant. Vam avaluar el cost del director i la codirectora, Segons UPC Transparent[25] un Titular d'Escola Universitària treballa 1640 hores anuals i té un sou brut de 30.926,88 € anuals, que representarien 18,86 € per hora. La compensació econòmica d'un estudiant que està cursant el TFG a una empresa segons el conveni de la UPC ha de ser de 9 €/hora.[5]

En l'apartat de planificació del projecte (7.1) es va avaluar que la dedicació al TFG seria d'aproximadament unes 470 hores per part de l'estudiant. Aquest temps, suposant que el temps de l'estudiant està valorat com a mínim com indica el conveni mencionat anteriorment, suposaria un cost de 4.230 €, tenint en compte la seguretat social, 5.710,50 €. A més a més, el temps aproximat que dedicaria el director d'aquest projecte seria d'unes 50 hores dedicades a reunions i revisió de la memòria. El temps aproximat que dedicaria la

codirectora seria d'unes 100 hores, dedicades a reunions, seguiment i revisió de la memòria. En total 150 hores que, a un sou de 18,66 €/hora suposarien un cost de 2.799 €, tenint en compte la seguretat social, 3.778,65 €.

El cost dels recursos humans utilitzats seria la suma dels tres participants del projecte, un total de 9.489,15 €.

7.3.2 Costs indirectes

Per a calcular els costs indirectes es van tenir en compte diversos elements. Aquests eren el cost de les llicències, el consum elèctric, el consum d'Internet, l'amortització del hardware i per acabar l'espai de treball.

7.3.2.1 Llicències

Tot el software que s'utilitzaria per realitzar aquest projecte és gratuït exceptuant el Visual Studio. La UPC té un conveni amb Microsoft, propietària de Visual Studio, i per tant resulta difícil calcular exactament el cost d'usar Visual Studio. En el cas que fos comprat sense conveni, resultaria en un cost de 641 €[26].

7.3.2.2 Consum elèctric

Per a calcular el cost del consum elèctric es van tenir en compte els dos ordinadors utilitzats i el dispositiu HTC Vive. Es va agafar com a preu del kWh el valor de 0,1198 €/kWh.

Hardware	Potència	Hores d'ús	Consum total	Cost
Ordinador CRV	450 W	150 h	67,5 kWh	8,09 €
Ordinador personal	335 W	320 h	107,2 kWh	12,84 €
HTC Vive	6 W	100 h	0,6 kWh	0,07 €
Total				**21 €**

Taula 5: Consum elèctric del hardware.

7.3.2.3 Amortització del hardware

Principalment el hardware utilitzat havia de ser l'exposat anteriorment, en aquest apartat es va calcular el cost del hardware tenint en compte el seu temps de vida útil i el temps que s'havia d'utilitzar. Per calcular aquesta despesa es va tenir en compte que tant l'ordinador del CRV com el HTC Vive serien utilitzats

paral·lelament per dues persones. Els càlculs es van fer agafant com a temps d'ús del hardware durant el projecte quatre mesos.

Hardware	Preu	Temps de vida	Amortització
Ordinador CRV	1200 €	5 anys	80 €
Ordinador personal	400 €	5 anys	26,67 €
HTC Vive	786,95 €	3 anys	87,44 €
Total			**194,11 €**

Taula 6: Amortització del hardware.

7.3.2.4 Consum d'Internet

El cost de la tarifa personal d'Internet és d'uns 40 € per mes, a repartir entre 5 persones que la utilitzen, per tant, el cost és de 8 € per mes, i si el projecte es desenvolupa durant quatre mesos, resultarà en un total de 32 €. Al CRV es pot utilitzar Eduroam, així que el cost és quasi negligible.

7.3.2.5 Espai de treball

El treball s'ha realitzat al Centre de Realitat Virtual (CRV), principalment s'havia d'utilitzar un despatx on hi ha 12 ordinadors, el preu d'una sala com aquesta té un cost aproximat d'uns 150 €/mes, durant quatre mesos són 600 €, si ho dividim pel nombre d'ordinadors, l'espai on s'havia de treballar tindria un cost d'uns 50 €.

7.3.3 Contingència

Durant el treball era possible que ens sorgissin imprevistos, vam suposar que els costos d'aquests no resultarien més d'un 15% dels costos finals. Per tant com a màxim tindrien un cost extra de 10.427,26*0,15 = 1.564,09 €.

7.3.4 Cost total

Tenint en compte tots els costos mencionats anteriorment vam calcular el pressupost total del projecte.

	Cost
Recursos humans	**9.489,15 €**
Estudiant	5.710,50 €
Director	1.259,55 €
Co-directora	2.519,10 €
Costs indirectes	**938,11 €**
Llicència Visual Studio	641 €
Consum elèctric ordinador CRV	8,09 €
Consum elèctric ordinador personal	12,84 €
Consum elèctric HTC Vive	0,07 €
Amortització ordinador CRV	80 €
Amortització ordinador personal	26,67 €
Amortització HTC Vive	87,44 €
Consum d'Internet	32 €
Espai de treball	50 €
Contingència	**1.564,09 €**
Total	**11.991,35 €**

Taula 7: Pressupost total del projecte.

7.3.5 Prolongació del projecte

Tot i haver de ser entregat a finals de gener, aquest projecte es va allargar i es va decidir entregar a finals d'abril, això va provocar que alguns costs augmentessin, ja que en lloc de calcular per 4 mesos s'havia allargat a 7.

En el cas de l'amortització del hardware, va passar de 194,11 € a 339,69 €. El cost del consum d'Internet va passar de 32 € a 56 €. El cost de l'espai de treball va passar de 50 € a 87,5 €. En total **207,08 €** més dels esperats, fent que el pressupost passés a ser de 10.634,34 €, si un altre cop tinguéssim en compte que els costos extres podien suposar fins a un 15% dels costos finals, el pressupost total del projecte passaria a ser de **12.229,49 €**.

7.4 Sostenibilitat

Abans de començar el projecte vam realitzar aquesta anàlisi de la sostenibilitat del projecte:

7.4.1 Impacte ambiental del projecte posat en producció

Aquest projecte tindria impacte ambiental a causa del consum d'energia produït durant el desenvolupament del nostre software. Com hem vist a l'apartat de consum elèctric (7.3.2.2), el consum total d'electricitat havia de ser de 175,3 kWh, si suposem que per cada kWh es produeixen 0,41 kg de CO_2[9], el projecte generaria 0,071873 tones de CO_2. Per a minimitzar el màxim possible aquesta quantitat, només s'utilitzarien els ordinadors i el HTC Vive quan fossin necessaris i s'apagarien un cop ja no ho fossin.

Tot i això no podíem obviar que per crear els recursos hardware que s'havien d'utilitzar s'haguessin utilitzat uns recursos naturals. L'extracció d'aquests recursos naturals normalment genera una quantitat de CO_2 bastant elevada, i a això li havíem de sumar els processos de producció fins a arribar al producte final. També havíem de tenir en compte que molts d'aquests recursos són limitats.

7.4.2 Impacte ambiental de la vida útil

Quan s'acabés la vida útil d'aquest projecte, la implementació del nostre model d'il·luminació podria ser reutilitzada per altres projectes, disminuint així la petjada econòmica d'aquests.

Per altra banda, un cop s'acabés la vida útil dels dispositius hardware s'haurien generat uns residus que sí que contribuirien en la deterioració del medi ambient. Un cop acabada la seva vida útil hi hauria dues opcions, o bé reciclar-los o bé donar-los a alguna organització que es dediqués a la reutilització d'aquests recursos.

7.4.3 Impacte econòmic del projecte posat en producció

L'impacte econòmic d'aquest projecte posat en producció era degut al material utilitzat, el software utilitzat, l'energia consumida, al lloguer de l'espai, l'ús d'internet i els recursos humans principalment, els costs d'aquests van ser calculats en el pressupost (7.3).

7.4.4 Impacte econòmic de la vida útil

La implementació del nostre model d'il·luminació no influiria en l'àmbit econòmic, a no ser que es volgués comercialitzar, que no era la finalitat d'aquest projecte.

7.4.5 Impacte social del projecte posat en producció

A l'hora de ser posat en producció, l'impacte social que tindria aquest treball seria principalment cap a mi. Seria un procés d'aprenentatge diferent dels que estava acostumat, adquirint una formació que possiblement influiria en la meva futura vida professional.

Com ja s'ha comentat abans, per crear els recursos hardware que s'utilitzarien són necessaris recursos naturals, malauradament, part dels recursos utilitzats són extrets de les mines en condicions pèssimes, sota amenaces i amb alts riscs de salut.[4]

7.4.6 Impacte social de la vida útil

Aquest treball estava destinat principalment a aplicacions mèdiques. És per això que els principals beneficis socials serien en el camp de la medicina, millorant un software que serveix per estudiar el cos humà i d'altres amb gran precisió. Tot i això, aquest treball no només podria ser aplicat en el camp de la medicina, també podria ser aplicat en el camp de la Realitat Virtual en general. En principi no hi hauria efectes negatius en l'àmbit social.

Referències

[1] Fran Barrionuevo. *Producto Escalar*. URL: https://www.pinterest.es/pin/670332725760952681/?lp=true.

[2] Charl Botha Bernhard Preim. "Visual Computing for Medicine (Second Edition)". A: (2014). URL: https://www.sciencedirect.com/topics/computer-science/volume-rendering.

[3] National Institute of Biomedical Imaging i Bioengineering (NIBIB). *Ultrasonido*. URL: https://www.nibib.nih.gov/espanol/temas-cientificos/ultrasonido.

[4] *Blood In The Mobile. Frank Piasecki Poulsen*. 2010.

[5] *Compensació econòmica estudiant TFG*. URL: https://telecos.upc.edu/ca/empreses/convenis-de-cooperacio-educativa/informacio-per-a-estudiants%5C#section-13.

[6] Department of Computer Sciences of the University of Texas at Austin. *Graphics – Spring 2013. Illumination I: The Phong Illumination Model*. URL: https://www.cs.utexas.edu/~bajaj/graphics2012/cs354/lectures/lect14.pdf.

[7] Wikipedia. The free encyclopedia. *Phong shading*. URL: https://en.wikipedia.org/wiki/Phong_shading.

[8] Wikipedia. The free encyclopedia. *Raytrace Diagram*. URL: https://commons.wikimedia.org/wiki/File:Ray_trace_diagram.png.

[9] Gobierno de España. Ministerio para la transición ecològica. *Factores de Emision*. URL: https://www.miteco.gob.es/es/cambio-climatico/temas/mitigacion-politicas-y-medidas/factores_emision_tcm30-479095.pdf.

[10] *Exposure Render 1.1*. URL: http://medvis.org/2011/10/26/exposure-render-1-1-now-available/.

[11] Dra. Cristina Muñoz Gil. *¿Qué es una resonancia magnética?* URL: https://www.salud.mapfre.es/pruebas-diagnosticas/otras-pruebas-diagnosticas/resonancia-magnetica/.

[12] Markus Hadwiger et al. *Real-time Volume Graphics*. Natick, MA, USA: A. K. Peters, Ltd., 2006. ISBN: 1568812663.

[13] *HTC Vive – Sistema de realidad virtual*. URL: https://www.amazon.com.mx/HTC-Vive-Sistema-realidad-virtual/dp/B00VF5NT4I.

[14] Diagnostico Especializado por Imagen. *Resonancia magnética de cerebro*. URL: https://www.deidiagnostico.com/resoancia_magnetica_de_cerebro/.

[15] Marcus Jonsson. "Volume rendering". A: (oct. de 2005). URL: http://www8.cs.umu.se/education/examina/Rapporter/MarcusJonsson.pdf.

[16] Thomas Kroes, Frits H. Post i Charl P. Botha. "Exposure Render: An Interactive Photo-Realistic Volume Rendering Framework". A: *PLOS ONE* 7 (jul. de 2012). DOI: 10.1371/journal.pone.0038586. URL: https://doi.org/10.1371/journal.pone.0038586.

[17] Eidgenössische Technische Hochschule Zürich. Computer Graphics Laboratory. *Stuttgart Visualization Course, Driect Volume Rendering*. URL: https://cgl.ethz.ch/teaching/former/scivis_07/Notes/stuff/StuttgartCourse/VIS-Modules-06-Direct_Volume_Rendering.pdf.

[18] Timo Ropinski Markus Hadwiger i Patric Ljung. *Advanced Illumination Techniques for GPU Raycasting. Eurographics 2006 conference in Vienna*. URL: http://www.real-time-volume-graphics.org/?page_id=28.

[19] Maverick. *Practical 1: Local Illumination Models*. URL: https://maverick.inria.fr/Members/Nicolas.Holzschuch/cours/Shaders/tp1_en.html.

[20] Jonathan Ragan-Kelley Ren Ng. *Recursive Ray Tracing*. URL: https://www.cs.utexas.edu/~bajaj/graphics2012/cs354/lectures/lect14.pdf.

[21] Dr. Jon Peddie. "What's the Difference Between Ray Tracing, Ray Casting, and Ray Charles?" A: (febr. de 2016). URL: https://www.electronicdesign.com/technologies/displays/article/21801219/whats-the-difference-between-ray-tracing-ray-casting-and-ray-charles.

[22] Sociedad Chilena de Radiología. *Seguridad en Resonancia Magnética*. URL: https://www.sochradi.cl/informacion-a-pacientes/abdomen-y-pelvis/seguridad-resonancia-magnetica/.

[23] Joan Fons Sànchez. *Inspecció interactiva i immersiva de models volumètrics. Aplicació diagnosi mèdica*. URL: https://upcommons.upc.edu/handle/2117/110238.

[24] Liteharbor Lighting Technology. *How many do you know about beam angles*. URL: http://www.liteharbor.com/resources/light-knowledge/201_How-many-do-you-know-about-beam-angles.html.

[25] *UPC Transparent*. URL: https://www.upc.edu/transparencia/ca/informacio-de-personal.

[26] *Visual Studio Store*. URL: https://www.microsoft.com/es-es/store/collections/visualstudio.

[27] Álvaro Serra Zornoza. *Trayendo los sueños de la realidad virtual a la vida – London Technology Week*. URL: https://rincondelatecnologia.com/realidad-virtual-london/.