UNIVERSITA' DEGLI STUDI DI PADOVA

Facoltà di Ingegneria

Corso di Laurea in Ingegneria dell'Informazione

TECNICHE DI RICOSTRUZIONE E VISUALIZZAZIONE 3D DI IMMAGINI OTTENUTE CON SCANSIONE TOMOGRAFICA

Tesi di Laurea di:
Nicola Salmaso

Relatore:
Emanuele Menegatti

INDICE

1. INTRODUZIONE

Nel settore della lavorazione del legno risulta fondamentale poter analizzare in dettaglio il materiale grezzo, i tronchi d'albero, prima della loro lavorazione e raffinazione. In modo particolare è importante conoscerne le caratteristiche, la loro composizione, il grado di umidità intrinseca, la presenza di eventuali impurità nascoste come nodi e crepe, non visibili dall'esterno e insidiose al momento dell'intaglio. Per poter largamente ovviare al problema una delle tecniche più efficaci è quella di passare i tronchi all'interno di un tomografo e ottenerne una TAC completa dalla quale si possono studiare i singoli tronchi guardandoli direttamente dall'interno.

La tesi tratta la progettazione e realizzazione di un software in grado di interpretare i risultati di un tomografo ricostruendo il tronco scansionato per poi darne una rappresentazione 2D a schermo con la possibilità di interagire con esso rototraslandolo nello spazio 3D, tagliarlo vedendone dettagliatamente l'interno, applicare profili di taglio simulando il risultato dei passi di lavorazione, osservarlo nel complesso come fosse trasparente ed evidenziare nodi e crepe. Con le possibilità offerte dal software un utente è quindi in grado di studiare al meglio la composizione del tronco e adeguare di conseguenza i passi di lavorazione riducendo al minimo gli sprechi dovuti ad impurità a prima vista nascoste ma largamente evidenziate dalla tomografia.

Per la realizzazione del software si è reso necessario lo studio in dettaglio delle tecniche tipiche del volume rendering, di algoritmi di ray-tracing, dell'architettura CUDA, delle leggi fisico-matematiche necessarie per l'applicazione di ombre agli oggetti e per visualizzarli in trasparenza.

1.1 Il progetto

Obiettivi

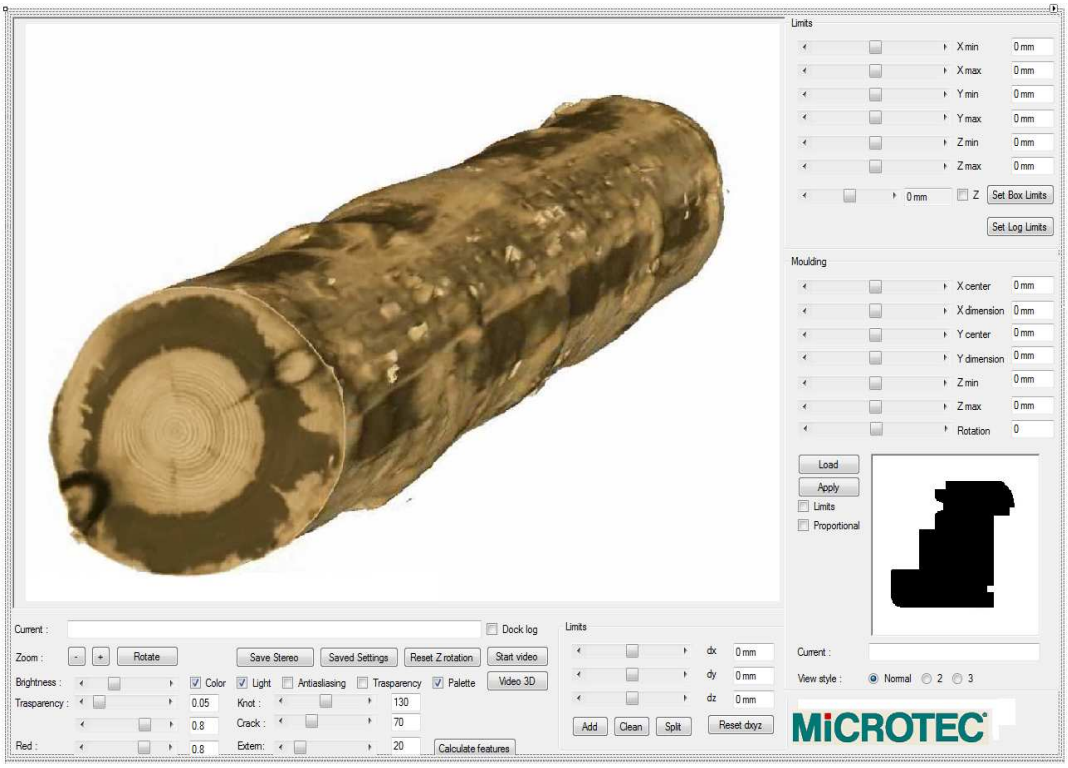
Lo scopo principale del progetto è quello di realizzare un software desktop in grado di visualizzare riproduzioni di oggetti 3D ottenute mediante scansione tomografica di tronchi e di poter interagire direttamente e virtualmente con tali oggetti. Per fare ciò il software è in grado di interpretare file ottenuti mediante una TAC eseguita su un oggetto qualsiasi, ricostruire struttura e composizione dell'oggetto scansionato visualizzandolo sullo schermo. Una volta ricostruito è possibile interagire con l'oggetto, ruotarlo e traslarlo nello spazio, applicargli dei tagli netti per vederne l'interno o dei profili più elaborati per osservare il risultato di un eventuale passo di lavorazione, renderlo gradualmente trasparente per avere una visione d'insieme più completa. Questo software da quindi la possibilità di studiare al meglio la composizione dell'oggetto scansionato senza doverlo fisicamente tagliare, permettendo così di ridurre al minimo gli sprechi ed ottimizzare ogni passo di produzione che preveda l'intaglio o l'asporto di parte di esso.

Materiale a disposizione

I dati di partenza del progetto sono alcune scansioni tomografiche preventivamente eseguite in azienda di alcuni tronchi di diversi tipi di albero mentre per realizzare il software sono stati utilizzati un computer con scheda video compatibile CUDA.

Per la stesura del codice si è utilizzato l'ambiente di sviluppo Visual Studio, integrando alcune librerie proprietarie dell'azienda e scritte in CPP per l'utilizzo semplificato di matrici ed altri operatori matematici. Per quanto riguarda la parte di

codice CUDA si è utilizzata la documentazione disponibile online sul sito dell'nVidia.



Schermata del software in esecuzione.

Ambiti di applicazione

Originariamente il sofrware è stato pensato per individuare manualmente nodi e crepe nascoste all'interno di un tronco e trovare il miglior modo di lavorarlo riducendo gli sprechi di materiale dovuto alle proprie impurità nascoste ma con una serie di migliorie è stato trasformato per poter simulare anche vari passi di lavorazione come la profilatura del legno o la trasformazione in tavole, per produrre video dimostrativi, per analizzare la densità del tronco e la quantità di acqua in esso contenuta, per fare demo presentative dell'azienda.

Importante notare che il software riconosce file ottenuti mediante la scansione TAC di un *qualsiasi* oggetto e non solo tronchi, è quindi possibile utilizzarlo anche in altri ambiti come per esempio quello biomedico permettendo così di studiare un paziente osservandolo direttamente dall'interno senza nessun genere di operazione chirurgica ma solo con l'utilizzo del mouse.

Simulazione software di un passo di lavorazione.

Simulazione di un altro tipo di lavorazione

1.2 MiCROTEC – l'azienda

MiCROTEC è leader tecnologico nel settore dell'opto-elettronica per l'industria della lavorazione del legno fin dal 1980. Si occupa di ogni processo della lavorazione del legno che possa essere razionalizzato, ottimizzato e accelerato tramite corrispondenti sistemi informatici ed elettroni MiCROTEC. Grazie alla concentrazione e alla specializzazione nel mondo del legno e della sua lavorazione, MiCROTEC rappresenta oggi l'innovazione in questa nicchia di mercato, con più di 20 ingegneri che svolgono le attività di ricerca e sviluppo utilizzando i sistemi e le tecnologie più avanzate e lavorano al continuo miglioramento dei prodotti MiCROTEC con l'obiettivo di svelare ogni dettaglio del legno. Attualmente l'azienda opera nelle sedi di Bressanone, Venezia, Linz e Vancouver

MiCROTEC lavora con un impressionante ed innovativo portafoglio tecnologico per razionalizzare, automatizzare ed ottimizzare ogni singolo processo di lavoro volto alla lavorazione del legno. La sfida consiste nel rilevare e comprendere le peculiarità delle caratteristiche organolettiche della materia prima legata alla sua origine regionale da una parte, e dall’altra le richieste ed i bisogni specifici della linea di produzione della stessa impresa data dal suo campo di specializzazione. Per rispondere a queste complesse specificità, MiCROTEC pone una grande attenzione nello sviluppo dei propri sistemi. Sin dal 1980, anno della sua fondazione, MiCROTEC ha concentrato le sue risorse sull’innovazione di tutto il settore della lavorazione del legno potendo oggi contare tra l'altro su:

1. Elaborazione di immagini ad alta risoluzione/definizione
2. Tecnologia ad infrarossi
3. Tecnologia laser
4. Radiografia
5. Tomografia computerizzata

2. IL SISTEMA DI ACQUSIZIONE

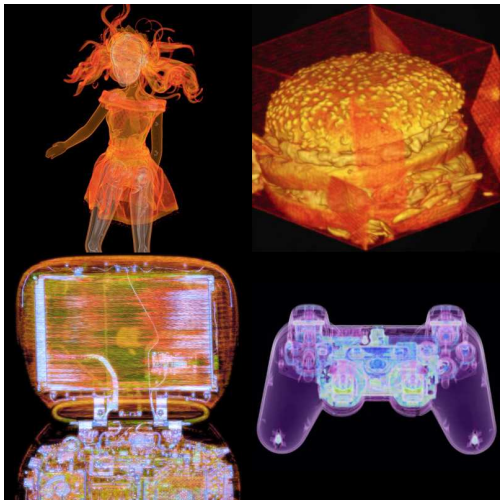
2.1 Tomografia (TAC)

La tomografia, nota anche con l'acronico TAC (tomagrafia assista al computer), è una tecnica che sfrutta radiazioni ionizzanti permettendo di ottenere una ricostruzione tridimensionale di un oggetto scansionato.

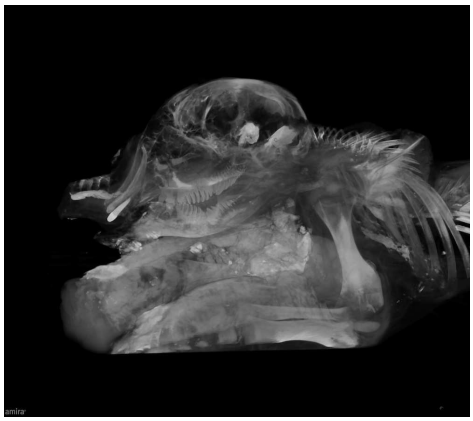
L'ambito di utilizzo più conosciuto è quello medico dove la tomografia viene impiegata per analizzare in dettaglio parte del corpo di un paziente non limitandosi alla superficie o alla semplice struttura ossea ma potendo disporre di una sorta di bisturi virtuale che permette di tagliare ed osservare qualunque dettaglio.

La tomografia viene utilizzata anche in archeologia per analizzare mummie o contenuti di oggetti che per motivi di conservazione non possono venire fisicamente aperti ma sono osservabili solo in questo modo.

L'ambito in cui si pone la tesi è invece quello dello studio non distruttivo dei materiali, in particolare nel caso particolare del software realizzato di tronchi d'albero.



A sinistra, tomografia del corpo umano. A destra tomografie eseguite al check-in di un aeroporto



Tac di un cucciolo di mammoth, eseguita da un gruppo di archeologi

Tomografo e tac

Il macchinario utilizzato per eseguire una tomografia prende il nome di tomografo ed è composto da un emettitore e uno o più rilevatori. L'emettitore di raggi X ruota attorno all'oggetto interessato mentre al lato opposto il rilevatore raccoglie i dati relativi all'immagine di una sezione di tale oggetto. Ad ogni ciclo di scansione il rilevatore cambia posizione in modo di ottenere una nuova sezione. Una volta completata l'operazione si dispone di una sequenza di sezioni generalmente equispaziate dell'oggetto, tanto più vicine e dettagliate quanto più preciso e ricco di sensori è il tomografo.

I processi principali per ottenere le immagini dai dati grezzi sono la convoluzione e la retroproiezione ottenuta tramite trasformata di Radon) Le immagini di partenza di tutte le sezioni vengono normalmente registrate su un sistema di archiviazione (PACS) mentre il rivelatore ad alta efficienza è normalmente costituito da cesio ioduro, calcio fluoruro, cadmio tungstato.

Mentre nell'ambito medico l'emettitore di raggi X deve ridurre l'intensità del proprio raggio per evitare danni al paziente, nel caso dei tronchi questo problema non sussiste e si possono quindi ottenere immagini più nitide e in minor tempo, velocizzando notevolmente il processo di scansione e riducendo il lavoro di ricostruzione del modello 3D.

In determinati casi può risultare conveniente non limitarsi ad ottenere sezioni equispaziate ma aumentare la distanza quando si analizzano zone poco interessanti o molto omogenee per aumentare la densità di sezioni nelle zone dove l'oggetto si rende più eterogeneo o dove è necessario maggiore dettaglio.

Nel caso dei tronchi si è ritenuto opportuno mantenere sezioni equispaziate poiché essendo scansionato per sezioni circolari essendo ogni tronco omogeneo lungo il proprio asse orizzontale.

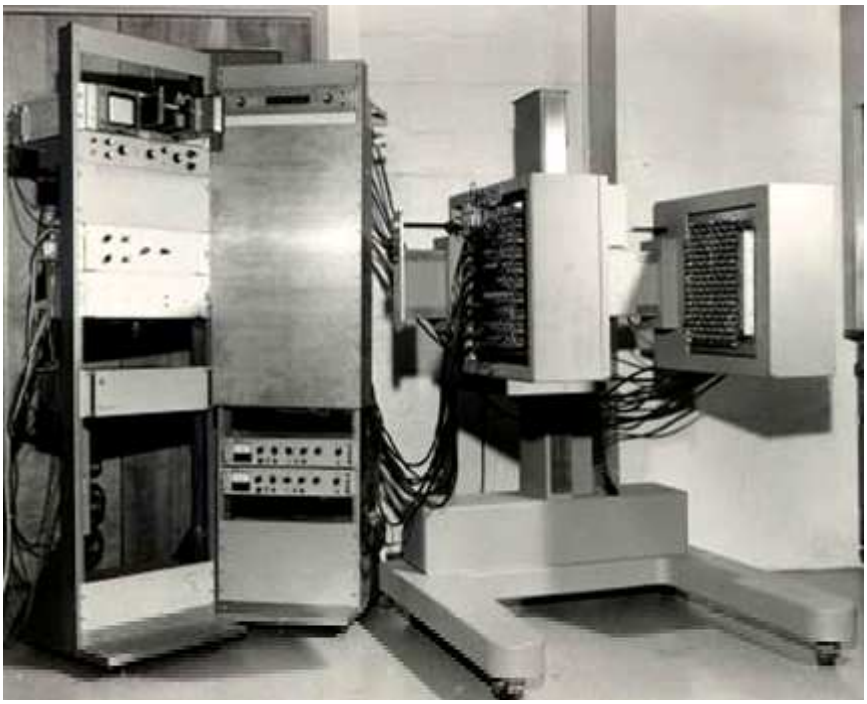
L'immagine del corpo da studiare viene ricostruita misurando l'attenuazione di un singolo fascio di raggi X che lo attraversa. L'attenuazione misurata varia proporzionalmente alla densità di elettroni presenti nei tessuti attraversati, cioè alla loro distribuzione spaziale nello strato corporeo in esame. Essendo le immagini ricostruite di tipo digitale, l'oggetto studiato viene suddiviso in una serie discreta di elementi di volume detti voxel, ai quali corrisponde un elemento unico d'immagine, rappresentato secondo una scala di grigi. Dal valore di tale voxel è possibile dedurre la densità dell'oggetto nel punto tridimensionale corrispondente e analizzando ogni voxel possiamo avere una descrizione precisa di tutto il corpo scansionato. Nel software ogni voxel può assumere un valore da 0 a 255 dove 0 corrisponde ad un elemento di densità nulla, il caso per esempio di quando i raggi non attraversano l'oggetto ma l'atmosfera attorno ad esso, e 255 è il massimo valore di densità misurabile.

La risoluzione spaziale di una scansione è tanto maggiore quando più piccolo è l'elemento volumetrico rappresentato da un voxel. In altre parole l'immagine è tanto più dettagliata quanti più numerosi e densi saranno i voxel nel campionamento.

L'evoluzione della tomografia

Nei primi tomografi l'emissione di un fascio lineare di raggi X da un tubo radiogeno in movimento di traslazione e di rotazione veniva misurato da un rilevatore solidale nel movimento. Il tempo di esecuzione complessivo era dell'ordine dei minuti.

I tomografi di seconda generazione usavano un fascio di raggi X con una geometria a ventaglio da 20 a 30 gradi connesso con un gruppo di una trentina di rilevatori; il tempo di esecuzione era ridotto a decine di secondi.



PC-I, il primo dispositivo di imaging tomografico PET.

Nei tomografi successivi il fascio di raggi X a ventaglio venne allargato all'intervallo da 30 a 50 gradi permettendo così di comprendere tutta la sezione corporea in esame. Tale raggio veniva poi captato da centinaia di rilevatori contrapposti compiendo una rotazione completa attorno all'oggetto. Il tempo si ridusse a meno di 5 secondi. Per fare in modo che i cavi di alimentazione ritornassero nella posizione di partenza ad una rotazione ne seguiva un'altra nel senso inverso obbligando all'acquisizione di un solo strato per volta.
I tomografi di quarta generazione con sensori fissi vennero subito abbandonati. Quelli moderni sono invece un'evoluzione di quelli di terza generazione con la modifica dell'acquisizione, divenuta a spirale. In questo modo con una rotazione continua unidirezionale il tubo radiogeno e i rilevatori sono montati su un anello rotante che si alimenta a "contatti striscianti", senza più il problema dei cavi che si attorcigliano. In questo modo è possibile l'acquisizione delle immagini in modo continuo: mentre l'oggetto scorre orizzontalmente lungo un piano fisso i piani di scansione descrivono un'elica attorno al corpo, ottenendo appunto una scansione "a spirale". Normalmente una rotazione viene effettuata in più o meno un secondo consentendo un'acquisizione completa di un volume corporeo in al massimo un minuto e in un'unica volta.

Negli ultimi modelli si è introdotta la tecnologia a doppio tubo radiogeno, detta *dual source*. Disponendo di due tubi radiogeni che funzionano a differenti energie a causa della differente attenuazione dei tessuti si riesce ad avere una risoluzione di contrasto migliore.

Un tomografo di ultima generazione.

2.2 Scansione e ricostruzione di un oggetto

A differenza della TAC di un paziente quella ad un tronco o a del materiale in genere presenta alcuni vantaggi. Mentre un uomo non può essere esposto a radiazioni ad energia troppo elevata o per troppo tempo un corpo non vivente non presenta questo problema. Questo permette di ridurre i tempi di scansione andando incontro alle esigenze produttive e in contemporanea permette di ottenere immagini molto più dettagliate e dense, ad alta qualità di definizione. In questo modo è possibile aumentare la definizione fino alle esigenze di software fino ai limiti concessi dall'hardware a disposizione.

Digitalizzazione di una TAC

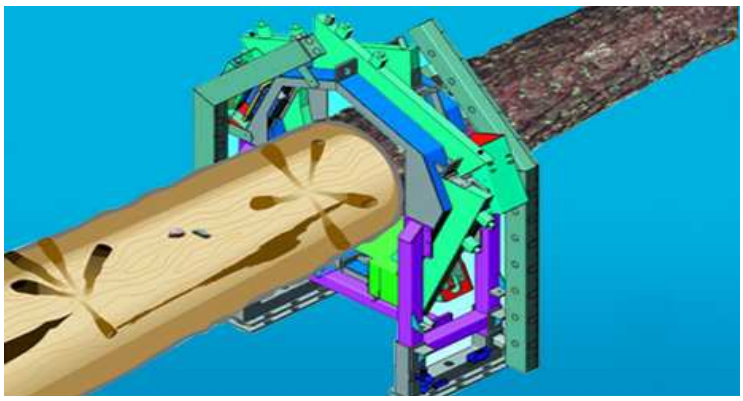
Una volta che il tomografo ha finito la scansione di un oggetto questa viene codificata in memoria come un vettore di immagini, un tensore di dimensione 3 di interi positivi. Il modo più comune e comodo di utilizzare un file ottenuto per tomografia è quello di considerarlo come vettore di lunghezza n di immagini : ad ognuno degli n elementi del vettore corrisponde una delle sezioni, partendo dal primo elemento fino all'ennesimo le sezioni sono generalmente continue ed equispaziate. Questa visione del file permette di trattare singolarmente ogni sezione implementando algoritmi più semplici da applicare ad una singola immagine (di dimensioni note) ed eventualmente da iterare ad ognuna (o una parte) delle n sezioni.

Una sezione è generalmente una matrice di interi non nulli di dimensione (n,m) per un totale di nxm pixel per sezione. Ogni pixel rappresenta un punto del corpo scansionato e ad esso è associato un numero intero che ne rappresenta la densità. Lo 0 è attribuito a valori con densità nulla, dove il raggio passa senza scontrarsi con materia ad esclusione dell'atmosfera, mentre valori maggiori sono associati a densità maggiori.

TAC di un tronco

Per quanto riguarda il software in questione i file contenenti le scansioni TAC sono composti da circa 160 sezioni per ogni metro di tronco mentre ogni sezione ha una risoluzione di 768x768 pixel per un totale di circa 30 MB per ogni metro di tronco. Un file tipico contiene dati relativi a tronchi di 5 metri di lunghezza per un totale di 150 MB.

Oltre alle sezioni ottenute per tomografia il file è composto da un'ulteriore sezione che fa da header : non contiene alcuna immagine ma dati relativi alla tomografia, in particolare il numero di sezioni totali e lunghezza del tronco scansionato in millimetri. Questi dati risultano fondamentali per permettere al software sia di scalare la riproduzione 3D mantenendo le proporzioni originali del tronco sia per utilizzare i millimetri come unità di misura per l'applicazione di tagli e profilature dando la possibilità all'utente di inserire direttamente all'interno di caselle di testo la posizione e la lunghezza di taglio desiderate sia per individuare la posizione esatta di eventuali impurità da escludere dai processi di lavorazione.



Tac di un tronco

3. TECNICHE ED ALGORITMI PER LA VISUALIZZAZIONE 3D

Il settore informatico della modellazione 3D comprende un gran numero di tecnologie ed algoritmi specifici per ogni genere di problema. La prima difficoltà quando si deve trattare un progetto di grafica 3D è quello di capire quali siano le migliore scelte di tali tecnologie per soddisfare i requisiti voluti.

Nel nostro caso sicuramente non si avrà a che fare con modelli matematici costruiti da zero visto che i dati di input comprendono delle scansioni tomografiche. Sarà quindi necessario trasformare il vettore di sezioni in un oggetto 3D interattivo e darne un'immagine 2D vista da un ipotetico osservatore in una data posizione dello spazio.

Ecco che si rende necessaria l'implementazione di un algoritmo di ray-tracing per il volume rendering.

3.1 Volume rendering

Il volume rendering, o rendering volumetrico, è una tecnica utilizzata per visualizzare una proiezione 2D di un oggetto 3D. Tale oggetto 3D deve essere rappresentato come un set di immagini bidimensionali che ne rappresentano le sezioni trasversali cioè gli stessi dati forniti al termine di una scansione tomografica. Proprio per questo il volume rendering si rivela una scelta ideale per l'implementazione del software in analisi.

Box e punto di vista

Per prima cosa bisogna introdurre delle coordinate spaziali per poter definire dove si trova l'oggetto 3D rappresentato e per poter compiere di conseguenza ogni genere di operazione su di esso.

Lo spazio 3D viene delimitato da un box, un cubo di dimensione fissata, implementata direttamente a livello di codice. Tale cubo conterrà il tronco e sarà lo spazio nel quale l'utente potrà muoversi per interagire con l'oggetto. Il cubo definisce quindi un sistema di assi cartesiani a 3 dimensioni, dove un vertice e 3 spigoli tra loro ortogonali e in esso incidenti ne definiscono rispettivamente origine ed assi. Ora abbiamo un sistema di riferimento univoco per poter determinare la posizione spaziale di ciascun punto (voxel) dell'oggetto.

Una volta definito il sistema di assi cartesiani resta da definire un punto di vista, il punto dal quale la proiezione 2D dell'oggetto è ottenuta. Immaginandosi un ipotetico osservatore, il punto di vista è l'insieme del punto in cui esso è posizionato e la direzione in cui esso guarda. Quindi nel complesso il punto di vista, che d'ora in avanti verrà definito come *camera*, è composto da 2 coordinate tridimensionali : il punto nello spazio e la direzione spaziale, che lo identificano univocamente. Ogni volta che il punto di vista cambia anche la proiezione 2D cambia, nonostante l'oggetto 3D rimanga fermo.

Anche se operativamente è la camera a spostarsi guardando relativisticamente la situazione possiamo dire che la camera rimane ferma ma è l'oggetto a muoversi : sarà questo l'effetto percepito dall'utente quando andrà a spostare la camera. Questo verrà visto più in dettaglio quando si parlerà di rotraslazione e Zoom.

Palette

Le sezioni trasversali risultanti dalla tomografia sono in scale di grigio e non a colori. Anche se questo sistema si rivela utile a livello progettuale, con intensità di grigio diverse associate a densità di materia diverse, l'utente finale deve vedere il tronco con il suo colore naturale e non come un oggetto in bianco a nero.

Per poter compiere questa trasformazione è sufficiente una mappa, detta palette, che ad ogni tonalità di grigio associa un colore in modo coerente.
Per realizzare questa trasformazione si è utilizzata una palette già esistente nelle librerie dell'azienda.

Rendering di un tronco al quale sono stati applicati i colori. Notare l'effetto dello shading sulla superficie.

Rototraslazioni e Zoom

Rototraslazione e zoom possono essere ottenute con lo spostamento di origine e direzione della camera.

Una traslazione lungo un asse può essere ottenuta spostando rigidamente l'origine della camera lasciandone inalterata la direzione. Nello specifico, per traslare rigidamente il tronco lungo l'asse x di 10 millimetri sarà necessario spostare l'origine di -10 millimetri lungo lo stesso asse dando l'illusione all'utente di muovere il tronco. Gli spostamenti, proprio perché applicati alla camera ma pensati per traslare l'oggetto, sono sempre con segno opposto a quello della rototraslazione voluta. Quindi, secondo questo esempio se prima l'origine si trovava nel punto (0,0,0) ora si è spostata nel punto (-10,0,0) mentre l'oggetto è rimasto nella posizione precedente (x,y,z). Se vediamo la situazione dal punto di vista inverso, quello percepito dall'utente, tenendo fissa la camera nella posizione (0,0,0) l'oggetto si è spostato dal punto (x,y,z) al punto (x+10,y,z). Ovviamente la stessa procedura si applica per traslazioni lungo gli altri due assi. Per poter operare

traslazioni lungo 2 o tutti e 3 gli assi contemporaneamente basterà combinare più traslazioni

Per quanto riguarda lo zoom questo è ottenuto spostando l'origine della camera, avvicinandola all'oggetto lungo l'asse definito dalla sua direzione. Se la camera ha come origine (x_0,y_0,z_0) e come direzione (x,y,z), con direzione normalizzata, sarà sufficiente spostare l'origine nel punto $(x_0+\alpha x, y_0+\alpha y, z_0+\alpha z)$ dove α è il fattore di zoom che determina quanto la proiezione 2D risulta zoomata. Se $\alpha > 0$ allora l'origine si avvicinerà all'oggetto, ottenendo una proiezione 2D ingrandita, se invece $\alpha < 0$ si otterrà uno zooming all'indietro.

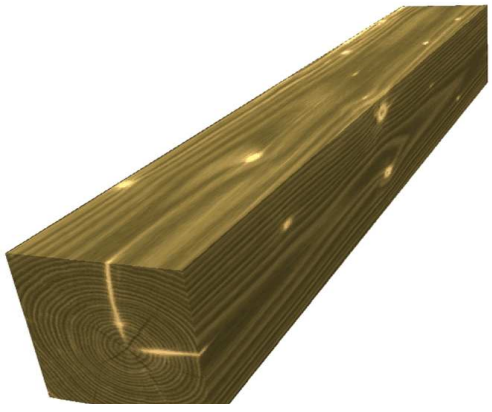
Infine le rotazioni possono essere ottenute mantenendo fissa la distanza tra l'oggetto e la camera, spostando l'origine della camera lungo la superficie della sfera che ha come origine il centro dell'oggetto e come raggio il segmento che va dal centro dell'oggetto all'origine della camera. Si noti che il centro dell'oggetto è il centro del box che definisce lo spazio 3D, se il box è un cubo di lato l l'origine dell'oggetto sarà il punto (l/2, l/2, l/2).

Tutti questi movimenti sono accessibili all'utente tramite interazione con mouse e tastiera. Basterà mantenere premuto il tasto sinistro del mouse e muovere il puntatore per ruotare intuitivamente l'oggetto, usare i tasti direzionali per traslarlo e la rotellina del mouse per lo zoom. Ogni spostamento va a modificare delle caselle di testo (associate a delle scrollbar) tramite le quali si può vedere come e quanto è stato spostato il tronco in termini di millimetri rispetto alla posizione iniziale.

Questo strumento da quindi modo all'utente di operare ogni genere di spostamento 3D e osservare il tronco da ogni punto di vista in modo semplice ed interattivo ma anche preciso inserendo direttamente l'entità dello spostamento voluto in millimetri.

Slicing

Fin qui si è parlato solo di rototraslazioni spaziali ma nessuna operazione è stata applicata al tronco in esame. La prima operazione che andiamo ad analizzare è quella dello *slicing*, ovvero affettare il tronco. Quello che ci interessa è rendere possibile la visione interna del tronco, applicando dei tagli netti che permettano di vederlo come fosse stato segato. Per semplicità e anche è stato scelto di applicare questi tagli solo lungo i 3 assi fondamentali, quello definiti dal box. Permettere il taglio lungo altri piani di quelli individuati dagli assi fondamentali sarebbe stato inutile ai fini pratici, oltre che più complicato.



Nell'immagine il tronco tagliato su tutti i suoi lati.

La tecnica utilizzata per permettere lo slicing è relativamente semplice. Al momento della resa grafica si analizza il punto all'interno della sua sezione, riducendo il problema ad un algoritmo 2D. Non importa sapere a che sezione appartenga il punto, quello che ci interessa sono le sole coordinate (x,y) duedimensionali. Come già detto, ogni sezione ha una risoluzione di 768x768 pixel. Possiamo decidere di mostrare il tronco tagliato a metà lungo l'asse x, mostrandone solo la prima porzione. Per farlo basta limitarsi a non visualizzare un punto del tronco quando questo si trova, all'interno della propria sezione, in una posizione che superi i 384 pixel dell'asse x. Il modo più facile per eseguire un'operazione del genere è quella (che poi è stata implementata) di applicare la stessa maschera ad ogni sezione e di andare quindi a visualizzare una tomografia corretta nella quale sono stati nascosti i pixel relativi alla porzione di tronco non interessata. Applicando contemporaneamente tagli lungo gli assi x ed y si va ad individuare un rettangolo più piccolo all'interno del quadrato 768x768 totale.

Con questa funzionalità è quindi possibile ottenere un tronco spaccato in due, una fetta sottile, un parallelepipedo, affettarlo lungo due piani paralleli o lungo due piani ortogonali, ecc.

Si può effettuare lo slicing anche lungo l'asse z, andando a ridurre il numero di sezioni visualizzate e di fatto accorciando il tronco (di base lungo 5 metri) lungo una o entrambe le sue estremità, fino ad ottenere un disco sottile.

Tutte queste operazioni sono operabili in tempo reale da software e permettono all'utente una prima analisi del tronco. Da tagli di questo tipo si possono infatti vedere eventuali impurità del legno come nodi, crepe o altre imperfezioni.

Applicazione di profili d'intaglio

Una operazione più raffinata del taglio netto lungo un piano è quella dell'applicazione di profili d'intaglio. Per esempio quando si producono serramenti ad un tronco di legno vergine vengono applicati intagli molto più complicati di semplici tagli netti. Il software deve fornire il modo di intagliare in maniera adeguata il tronco secondo questi profili la cui forma non può essere prevista a priori ma varia da azienda ad azienda. La soluzione migliore è quindi quella di fornire questi profili direttamente da file.

Un file di profilo è un'immagine true/false, cioè un immagine composta sola di due colori : bianco e nero. I punti di bianco saranno quelli da non visualizzare, mentre quelli neri individuano il profilo voluto. Una volta aperto e letto un file profilo questo viene convertito direttamente in una maschera di taglio, simile a quella creata per lo slicing ma più complessa. Ogni punto della maschera binaria può valere true o false, nel primo caso significa che quel punto andrà visualizzato, viceversa per il secondo caso.

Ecco che una volta ottenuta questa maschera l'applicazione del profilo si semplifica : si andrà a sfruttare la stessa tecnica usata per lo slicing. Ogni sezione verrà quindi ripulita dei pixel in più lasciando visibile solo la parte definita dal profilo. Applicando traslazioni e rotazione (di 90, 180, 270°) alla maschera binaria è possibile applicare il profilo in posizioni diverse del tronco.

Qui si vede una delle potenzialità maggiori del software. La prima applicazione del profilo potrebbe avere un nodo visibile sulle superficie esterna : un'imperfezione non voluta. Il nodo, oltre ad essere antiestetico, rende più debole il legno e potrebbe staccarsi lasciando un foro col risultato di un prodotto da buttare. Accorgendosene da software è possibile rimediare al problema spostando o ruotando il profilo fino ad ottenere una superficie perfetta e senza impurità, riducendo così lo scarto di pezzi prodotti.

L'immagine mostra l'applicazione di un profilo d'intaglio per finestre.

Lo stesso profilo, applicato prima in un modo e poi ruotato allo scopo di non avere nodi in superficie.

Individuare ed evidenziare nodi, crepe ed altre imperferzioni

Un metodo ancora più potente di visualizzazione delle impurità è quello di renderle maggiormente visibili colorandole di tonalità forti. L'unico problema è quello di individuarle da software. Questo è possibile perchè la tomografia associa ad ogni punto misurato una densità. Tale densità è maggiore nei nodi i quali hanno tutti valori all'interno di uno stretto range, mentre è nulla nele crepe essendo essi delle mancanze di materia. Ora che sappiamo come individuare queste impurità non resta che evidenziarle.

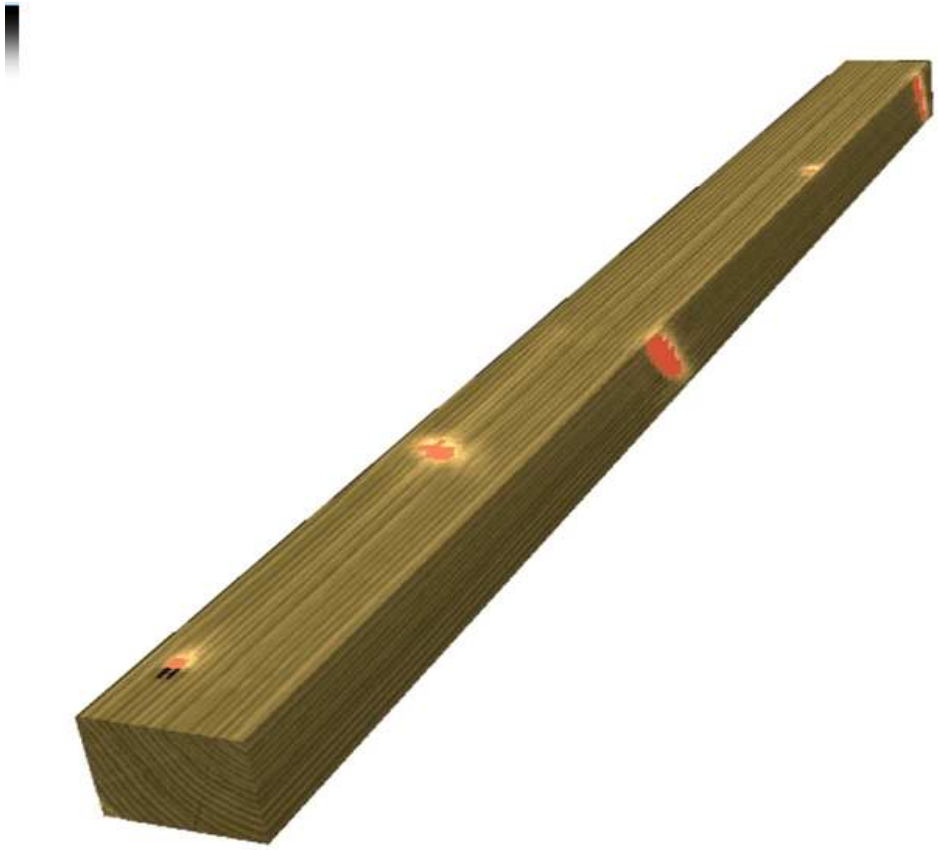
Al momento dell'applicazione della palette questo è possibile tramite alcuni controlli sui valori della scala di grigi. I nodi saranno i punti a densità maggiore, per essi è stato scelto di assegnare, anziché il loro colore naturale, un rosso intenso. La procedura per individuare le crepe è invece più complicata perchè essi hanno la stessa densità (nulla) che si trova all'esterno del tronco. Per distinguerli è necessario prima ottenere un'immagine in bianco a nero in cui ogni punto con densità non nulla (ovvero ogni punto del tronco) è colorata di nero, mentre il resto in bianco. Poi per ogni pixel nero vengono resi neri anche i pixel adiacenti. Questa procedura allarga leggermente il tronco e va a colmare le crepe. Sottraendo da questa immagine la precedente resta un'immagine in cui sono evidenziati solo le crepe. Questa immagine sarà una maschera true/false da usare per sapere dove sono posizionati le crepe, che per scelta saranno colorati di un blu acceso.

Una volta completata questa operazione è possibile, tramite un semplice check-box, abilitare o disabilitare l'evidenziamento delle imperfezione.

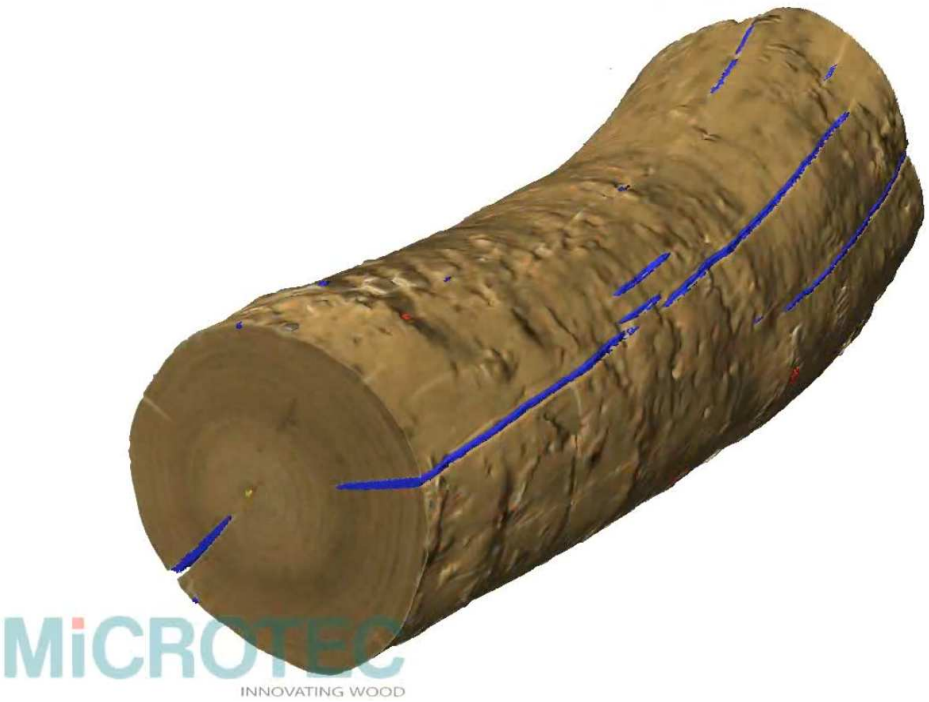
Un altro tipo di evidenziazione riguarda la parte umida del tronco. Se questo è stato tagliato nel momento sbagliato o non è stato lasciato seccare a sufficienza esso può presentare il nucleo umido. In questo caso al densità risulta leggermente maggiore di quella del legno asciutto. Questa imperfezione è stata evidenziata in giallo.

Una modifica che si potrebbe applicare al software ma che non si è resa necessaria perché in genere i tronchi esaminati non sono soggetti a questo tipo di impurità è quella della rilevazione di chiodi all'interno del tronco.
Questi appaiono nella tomografia come dei punti luminosi e hanno una densità maggiore di quella dei nodi. Si potrebbe aggiungere un altro controllo al momento d'applicazione della palette e colorarli, per esempio, di verde.

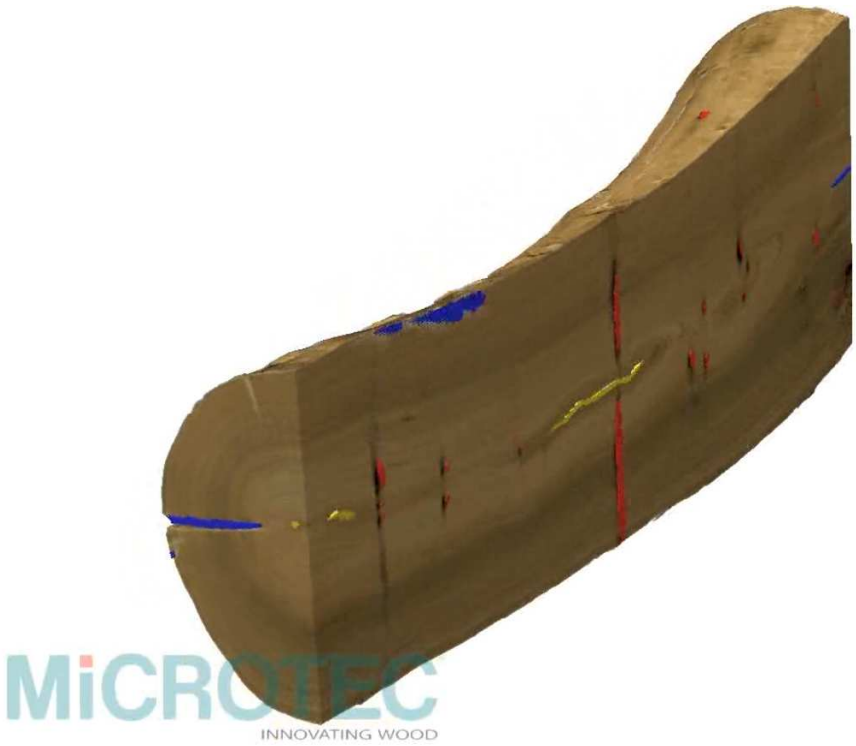
Tavola crepata e piena di nodi prima e dopo dell'evidenziazione delle imperfezioni..



Tronco profilato. I nodi in superficie sono evidenziati in rosso.



In questo due profonde crepe sono evidenziate in blu.



Qui si possono vedere i tre tipi di impurità.
In blu le crepe, in rosso i nodi e in giallo il nucleo umido o il midollo del tronco.

3.2 Algoritmo di Ray-Tracing

Il ray tracing è una tecnica di geometria ottica basata sul calcolo del percorso fatto dalla luce, seguendone i raggi e la loro interazione con le superfici. È usato nella modellazione di sistemi ottici, come lenti per fotocamere, microscopi, telescopi e binocoli. In particolare e nel nostro caso i termine viene utilizzato per un algoritmo di Rendering nel campo della Computer grafica 3D, in cui le visualizzazioni modellate matematicamente delle scene vengono prodotte usando una tecnica che segue i raggi partendo dal punto di vista della telecamera piuttosto che dalle sorgenti di luce**.**

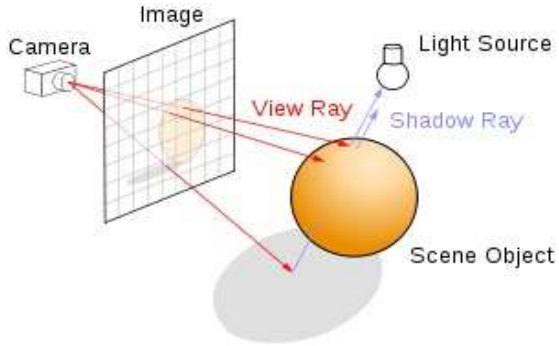


Illustrazione del principio di funzionamento dell'algoritmo di ray tracing.

Voxel

Per poter parlare di ray-tracing è necessario introdurre il concetto di voxel e spiegare com'è stato utilizzato all'interno del software. Il voxel è per il 3D quello che un pixel è per il 2D. In poche parole è un elemento atomico di volume, un punto dello spazio al quale è associato un valore numerico che ne definisce il colore e nel caso del software anche la densità.

Ogni volta che bisogna decidere se visualizzare un punto dello spazio 3D nella proiezione 2D o bisogna decidere come operare è necessario andare a guardare il valore del voxel corrispondente, così come si guardano i pixel uno ad uno per ricostruire l'immagine da visualizzare .

Algoritmo di Ray-Casting

Prima di introdurre nel dettaglio l'algoritmo di ray-tracing è utile spiegare il funzionamento di un altro algoritmo, detto di ray-casting, suo precursore.
Per implementare questi algoritmi torna utile il concetto di camera introdotto nel capitolo dedicato al volume rendering. Partendo dall'origine della camera si fanno partire raggi in varie direzioni, un raggio per ogni pixel di immagine 2D (proiezione dell'oggetto 3D) da ricostruire. Un raggio avanza lungo la direzione ad un passo atomico di lunghezza definita da software, questo passo va reso tanto più piccolo quanto più risoluzione è voluta. Ad ogni passo l'algoritmo va a misurare il valore del voxel incontrato lungo la direzione del raggio. Se tale valore è nullo o molto piccolo l'algoritmo prosegue, avanzo di un passo lungo la direzione. L'algoritmo si ferma quando trova un voxel di valore superiore ad una certa soglia minima, andando a visualizzare tale valore numerico, corrispondente ad un colore, nel pixel relativo al raggio analizzato. Se invece l'algoritmo esce dal box senza aver incontrato nessun voxel di valore positivo il relativo pixel viene lasciato bianco.

A differenza del ray-casting il ray-tracing rende molto più reali le superfici riflettenti simulando le leggi della fisica ottica

Algoritmo di Ray-Tracing

L'algoritmo di ray-tracing è simile a quello di ray-casting con la sola differenza che quando questo viene a contatto di un voxel con valore accettabile non si ferma ma avanza, con la possibilità di far partire 3 nuovi raggi. Questa funzionalità è stata utilizzata per permettere la rifrazione dando un effetto di trasparenza al tronco visionato.

Scelta del passo di campionamento

La scelta del passo di campionamento è molto importante. Un passo troppo grande rischia di far perdere di efficacia l'algoritmo infatti alcuni raggi potrebbero non individuare alcun voxel del tronco saltando da una zona vuota ad un'altra. L'ideale sarebbe disporre di un passo il più piccolo possibile, il problema è che renderebbe l'operazione troppo lenta, contando che l'algoritmo per il software progettato lavora in real-time, mentre operazione vengono compiute istante dopo istante sul tronco.

Come fare per ottenere un buon compromesso tra prestazioni e passo di campionamento verrà spiegato in uno dei prossimi paragrafi.

Ottimizzazione con di maschere di taglio

Quando si applicano maschere di taglio al tronco di fatto si va a ridurre il carico di dati da elaborare. Infatti le maschere operano direttamente sulle sezioni di tronco, andando ad escluderne intere aree. Quando un raggio incontra un punto appartenente ad una di queste aree vuote si limiterà a proseguire oltre, senza andare a guardare il valore del voxel. Questo accorgimento permette di migliorare molto le prestazioni e quindi di ridurre il passo di campionamento quando ci si trova in situazioni del genere. E' così possibile migliorare la qualità dell'immagine quando si applicano maschere di taglio, il che è molto apprezzabile visto che proprio in questi cosi è necessario un dettaglio maggiore, soprattutto per quanto riguarda le parti con intagli più complessi.

Antialiasing e passo di campionamento variabile

Il problema principale riscontrato nell'implementazione dell'algoritmo di ray-tracing è che per ottenere un'immagine pulita è necessario un passo troppo piccolo che rende l'esecuzione real-time troppo lenta e scattosa. Viceversa, aumentando il passo di campionamento ne esce un'immagine pulita per quanto riguarda il centro del tronco ma seghettata si suoi bordi.

La soluzione si è trovata utilizzando un passo di campionamento variabile. Prima viene eseguito l'algoritmo ad un passo fisso, grande abbastanza da garantire il real-time, ottenendo un'immagine seghettata ai bordi. Una volta terminato questo primo rendering viene rieseguito l'algoritmo di ray-tracing, questa volta solamente lungo i bordi seghettati del tronco e con passo di campionamento ridotto di un fattore 4. Questo ha permesso di ottenere al contempo un'immagine pulita ed un effetto real-time senza fastidiosi farfallii.

3.3 Shading

Lo shading è il processo di alterazione del colore basato sull'angolo d'incidenza della luce e sulla distanza dalla sorgente, per dare all'oggetto un effetto fotorealistico ed è uno dei passi presenti nel processo di rendering.

Calcolo approssimato del gradiente di superficie

Per prima cosa è necessario disporre di uno strumento che dia una misura numerica di quanto e come la superficie del tronco varia, più o meno rapidamente. Intuitivamente si ricorre al calcolo del gradiente : si vede la superficie come una funzione scalare in 3 variabili, le coordinate x,y,z es si calcolano le derivate parziali lungo i 3 assi. Questo processo restituisce tre valori, le tre componenti del vettore

gradiente appunto. Ognuna di queste componenti descrive come varia la superficie in un dato punto (quello in cui è calcolato il gradiente) lungo la relativa direzione.

Il metodo più rapido e comunque affidabile di calcolare il gradiente è quello di usare una sua approssimazione. Le sue tre componenti non verranno calcolate come derivate direzionali ma approssimate dal loro rapporto incrementale. Sia f(x,y,z) la funzione che dato un punto (x,y,z) restituisce il valore del voxel in quel punto allora la sua derivata direzionale lungo l'asse x viene approssimata con la formula [f(x+h,y+h,z,h)-f(x,y,z)]/h, dove h è un numero intero piccolo, in genere corrispondente ad un passo dell'algoritmo di ray-tracing. Il gradiente viene poi normalizzato dividendolo per la sua norma.

Questo calcolo del gradiente approssimato viene fatto ogni qual volta l'algoritmo di ray-tracing incontra un punto di densità non nulla e superiore alla soglia minima imposta, cioè quando il raggio incontra un punto solido del tronco. A questo punto non resta che variare il colore del punto in esame dando un senso di profondità tramite ombreggiatura.

Sorgente luminosa e normale alla superficie

Una volta calcolato il gradiente sulla superficie in esame è necessario disporre di una sorgente luminosa per poter applicare delle ombreggiature. Una sorgente luminosa è rappresentata da un vettore di dimensione tre che ne descrive la direzione, mentre l'origine non è importante, la sorgente va vista come se fosse fuori dal box di definizione dello spazio, l'unica cosa che importa è l'angolo con cui incide sulla superficie.

Quindi, una volta definita una direzione spaziale l'unica cosa che resta da fare è calcolarne il prodotto vettoriale con il gradiente precedentemente calcolato. Una volta fatta questa operazione basta moltiplicare il risultato per il valore del colore in quel punto e questo verrà corretto dando l'effetto dell'ombreggiatura. L'effetto finale sarà quello di avere le facce nascoste completamente in ombra, quelle rivolte verso la sorgente più luminose per poi andare a sfumare in maniera realistica. Il risultato finale è quello di un senso di profondità che da corpo all'oggetto, rendendolo una riproduzione realistica del tronco analizzato con il tomografo.

Variazione della luminosità

Poiché la scelta della sorgente luminosa è arbitraria questa può essere diretta in ogni direzione spaziale. La scelta opportuna di tale direzione rende un effetto grafico migliore ed è stata quindi data la possibilità all'utente di spostare tale sorgente a proprio piacere tramite il semplice click del mouse.

Un'altra possibilità è quella di aumentare la luminosità di questa sorgente : per farlo basta moltiplicare il risultato finale per un numero maggiore di uno per rendere il punto più luminoso, compreso tra zero ed uno per renderlo meno luminoso.

3.4 Rifrazione e trasparenza

Una funzionalità che si è voluta implementare al software è quella di dare la possibilità all'utente di vedere il tronco come fosse trasparente, potendo vedere contemporaneamente non solo la sua superficie ma anche il suo interno.

Per farlo è stato necessario modificare l'algoritmo di ray-tracing con la possibilità di tener conto di un effetto di rifrazione della luce, come se il tronco

fosse vetroso anziché di legno. Il risultato finale è quello di poter analizzare l'oggetto nella sua interezza, avendo un quadro d'insieme più completo.

Mentre la metà posteriore del tronco è mantenuta solida quella anteriore è stata resa trasparente.

Algoritmo

Per attuare questa modifica è sufficiente modificare il passo dell'algoritmo di ray-tracing nel punto in cui questo incontra un voxel di densità accettabile. Inizialmente l'algoritmo di fermava, restituendo il valore misurato dando quindi informazione solo relativamente al primo punto superficiale incontrato dal raggio di luce. E' ovvio che con questo metodo è impossibile avere alcun dato dei punti più interni del tronco.

Il raggio invece non deve fermarsi quando incide sulla superficie ma deve continuare il suo percorso. Esso una volta attraversata la superficie continua a sommare i valori dei voxel incontrati, attenuandoli ogni volta di un fattore sempre maggiore. Una volta uscito dal tronco o superato una soglia di saturazione il nuovo algoritmo si ferma, restituendo un valore che tiene conto di non più un solo voxel ma numerosi.

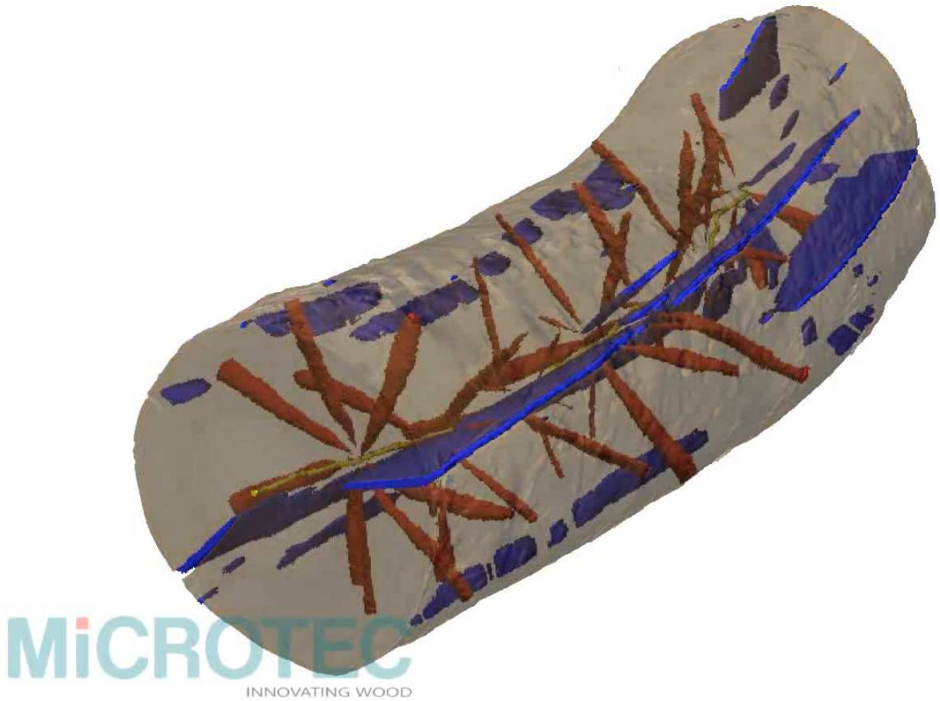
In questo modo nelle zone meno dense si vede l'interno del tronco fino alla superficie esterna nascosta e in quelle più dense si vedono i primi dettagli interni ad esso.

Variazione della trasparenza

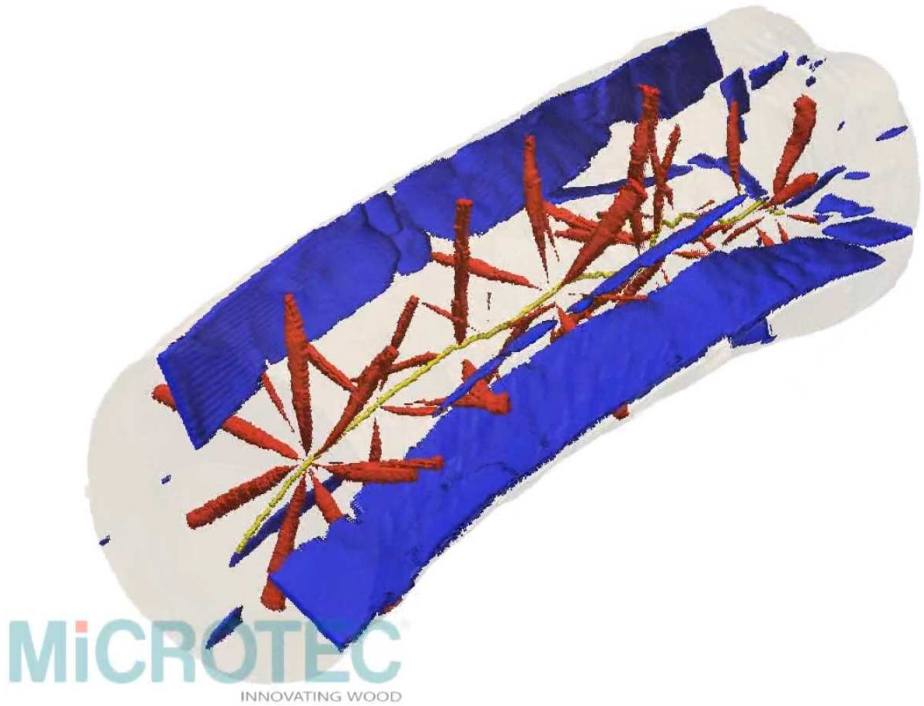
Anche in questo caso è stata data all'utente la possibilità di decidere quanto trasparente vuole vedere il tronco in esame. Per farlo basta variare l'attenuazione che sarà maggiore per un corpo più opaco e minore per un corpo molto trasparente. L'utente può quindi, tramite una scrollbar, scegliere il grado di trasparenza che ritiene più adeguato di volta in volta.

Trasparenza e colorazione

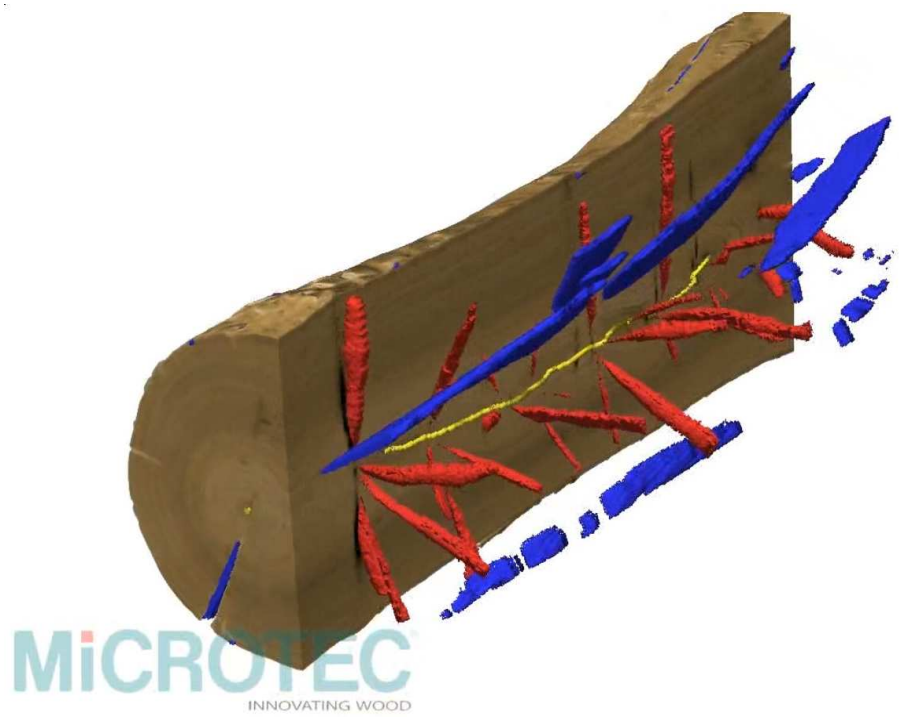
Un effetto particolarmente interessante lo si ha quando, una volta visualizzato il tronco in trasparenza, si sceglie di evidenziarne le imperfezioni.
A questo punto si riescono a vedere nitidamente tutti i suoi nodi e crepe interne, che spiccano di rosso e blu intensi all'interno del tronco vetroso.



Nel tronco semitrasparente sono stati evidenziati i difetti.



L'effetto è maggiormente apprezzabile quando si utilizza il massimo della trasparenza.



Qui si è invece combinato l'effetto di taglio a quello di colorazione delle impurità.

6 L'ARCHITETTURA CUDA

6.1 CUDA

CUDA (Compute Unified Device Architecture) è un'architettura hardware per l'elaborazione parallela creata da NVIDIA. Tramite l'ambiente di sviluppo per CUDA, i programmatori di software possono scrivere applicazioni capaci di eseguire calcolo parallelo sulle GP delle schede video NVIDIA.
I linguaggi di programmazione disponibili nell'ambiente di sviluppo per CUDA, sono estensioni dei linguaggi più diffusi per scrivere programmi. Il principale è 'CUDA-C' (C con estensioni NVIDIA), altri sono estensioni di Python, Fortran, Java e MATLAB.
Programmi che sfruttano l'architettura CUDA possono essere scritti anche utilizzando le librerie software OpenCl e DirectCompute.

Vantaggi

CUDA ha parecchi vantaggi rispetto alle tradizionali tecniche di computazione sulle GPU che usano le API grafiche.

Ecco le principali :

- Letture temporali - il codice può essere letto attraverso indirizzi arbitrari di memoria;
- Memoria condivisa - CUDA espone una veloce condivisione di memoria (una regione di 16kb di grandezza) che può essere condivisa fra i thread. Questa può essere usata come una cache gestita da utenti, abilitando grandi larghezze di banda che è possibile usare per strutture texture;
- Veloci downloads e riletture verso e dalla GPU;
- Supporto completo per divisioni intere e operazioni bit-a-bit, incluse strutture texture intere.

Limitazioni

CUDA è un sottoinsieme del linguaggio C privo di ricorsione e puntatori a funzione, più alcune semplici estensioni. Comunque, un singolo processo deve essere eseguito attraverso multiple disgiunzioni di spazi di memoria, diversamente da altri ambienti di runtime C.

Ecco le principali limitazioni :

- Il rendering delle texture non è supportato.
- Per la doppia precisione (supportata nelle nuove GPU come la GTX 260) ci sono diverse deviazioni dallo standard IEEE 754: l'arrotondamento al pari è l'unica approssimazione supportata per: reciproci, divisioni e radici quadrate. In singola precisione, i NANs segnalati e denormalizzati non sono supportati; queste sono specifiche per istruzioni di base, rispetto ad una singola parola di controllo; e la precisione delle cifre decimali di divisioni o radici n-esime è molto minore rispetto alla singola precisione.
- La larghezza di banda e la latenza tra CPU e GPU può essere un collo di bottiglia.
- I thread devo essere eseguiti in multipli di 32 per ottenere migliori prestazioni, con un numero totale di thread nell'ordine di migliaia. I rami dei codici non influiscono nelle prestazioni, a condizione che ciascuno dei 32 thread prenda lo stesso cammino di esecuzione. Il modello di esecuzione SIMD diviene una limitazione significativa per diversi compiti (per esempio l'attraversamento di uno spazio partizionato di strutture dati durante il raytracing).

- Diversamente da OpenCL, GPU dotate di CUDA sono disponibili solo da NVIDIA (GeForce 8 serie superiori, Quadro e Tesla)

6.2 CUDA all'interno del software

L'architettura CUDA sta alla base di questo software di rendering.
Il file ottenuto per tomografia prima viene aperto da software e poi viene caricato nella memoria della scheda video tramite CUDA. A questo punto tutte le operazioni di rendering, dall'algoritmo di ray-tracing all'ottenimento di una proiezione 2D del tronco, vengono eseguite grazie a CUDA. Non a caso la parte del software scritta in linguaggio CUDA prende il nome di kernel : sarà questo il vero nucleo del programma. Tutto il resto è sola gestione dell'interfaccia utente.
E' quindi utile andare ad analizzare in dettaglio questa parte del software.

All'interno del progetto ci sono due file CUDA. Il primo prende il nome di volumeRender.cu e si occupa di gestire la memoria della scheda video, il secondo è il più importante e è stato denominato volumeRenderKernel.cu. Qui verranno eseguite tutte le operazioni di rendering.

Gestione della memoria – dalla tomografia alla scheda video

Il primo passo da seguire per poter passare dal file della scansione tomografica alla visualizzazione a schermo è quello di interpretare l'informazione contenuta in tale file e passare l'intero contenuto alla scheda video. Visto che tutto il lavoro di rendering viene eseguito dalla scheda video, essere deve ricevere non solo il tronco ma anche tutti gli altri oggetti ed informazioni necessari per il rendering stesso. Vanno quindi passate anche tutte le impostazioni che l'utente ha raffinato da interfaccia grafica, il profilo di taglio scelto, la direzione della luce, l'entità della rifrazione, ecc.

La prima operazione da seguire è quella di inizializzare ed allocare la memoria della scheda video in modo che possa contenere tutti i dati su cui si andrà a lavorare. Questo viene per numerose strutture dati ma la parte di codice più interessante e rappresentativa è quella che si occupa del solo tronco.

```
void initRenderCuda(uchar *matLog, uchar *zeroMat, int dimX, int dimY, int dimZ)
{
    volumeSize.width=dimX;
    volumeSize.height=dimY;
    volumeSize.depth=dimZ;

    // create 3D array
    if(d_volumeArray) CUDA_SAFE_CALL(cudaFreeArray(d_volumeArray));
    cudaChannelFormatDesc channelDesc = cudaCreateChannelDesc<uchar>();
    CUDA_SAFE_CALL( cudaMalloc3DArray(&d_volumeArray,&channelDesc,volumeSize) );

    // copy data to 3D array
    cudaMemcpy3DParms copyParams = {0};
    copyParams.srcPtr   = make_cudaPitchedPtr((void*)matLog, dimX*sizeof(uchar),
dimX, dimY);
    copyParams.dstArray = d_volumeArray;
    copyParams.extent   = volumeSize;
    copyParams.kind     = cudaMemcpyHostToDevice;
    CUDA_SAFE_CALL( cudaMemcpy3D(&copyParams) );

    // set texture parameters
    tex.normalized = true;         // access with normalized texture coordinates
    tex.filterMode = cudaFilterModeLinear;      // linear interpolation
    tex.addressMode[0] = cudaAddressModeClamp;  // wrap texture coordinates
    tex.addressMode[1] = cudaAddressModeClamp;

    // bind array to 3D texture
    CUDA_SAFE_CALL(cudaBindTextureToArray(tex, d_volumeArray, channelDesc));
```

```
   //****************************************


    // create transfer function texture
    float4 transferFunc[] = {
        {  0.0, 0.0, 0.0, 0.0, },
              {  1.0, 1.0, 1.0, 1.0, },
    };

    cudaChannelFormatDesc channelDesc4 = cudaCreateChannelDesc<float4>();
    cudaArray* d_transferFuncArray;
    CUDA_SAFE_CALL(cudaMallocArray( &d_transferFuncArray, &channelDesc4,
sizeof(transferFunc)/sizeof(float4), 1));
    CUDA_SAFE_CALL(cudaMemcpyToArray( d_transferFuncArray, 0, 0, transferFunc,
sizeof(transferFunc), cudaMemcpyHostToDevice));
    transferTex.filterMode = cudaFilterModeLinear;
    transferTex.normalized = true;    // access with normalized texture
coordinates
    transferTex.addressMode[0] = cudaAddressModeClamp;   // wrap texture
coordinates

    // Bind the array to the texture
    CUDA_SAFE_CALL(cudaBindTextureToArray( transferTex, d_transferFuncArray,
channelDesc4));
       CUDA_SAFE_CALL(cudaMalloc((void**)&d_maskViewData, dimX*dimY*4));

}
```

La prima istruzione degna di nota è la seguente

```
CUDA_SAFE_CALL( cudaMalloc3DArray(&d_volumeArray,&channelDesc,volumeSize) );
```

Qui, dopo una serie di settaggi di parametri CUDA, viene fisicamente allocato tutto il file di scansione tomografica del tronco, quello su cui tutto il software andrà poi a lavorare. Questa operazione viene seguita da una serie di istruzioni necessarie per definire come utilizzare lo spazio di memoria appena occupato e per legare la texture al tronco.

Tutto il file VolumeRender.cu è formato da funzioni simili a questa in cui si inizializzano spazi di memoria della scheda video e li si inizializza.
Alla fine è importante liberare questo spazio quando non serve più, con una serie di funzioni free() come la seguente

```
void freeRender()
{
       CUDA_SAFE_CALL(cudaFreeArray(d_volumeArray));
       CUDA_SAFE_CALL(cudaFreeArray(d_volumeArray2));
       CUDA_SAFE_CALL(cudaFreeArray(d_volumeArray3));
}
```

Calcolo di un pixel

La funzione getColor(), di cui è omessa la maggior parte del codice, si occupa di richiamare l'algoritmo di ray-tracing, gestirne l'output nel modo migliore e convertire la densità ottenuta in un colore (applicando la palette) associato al pixel corretto.

```
float4 getColor( ... )
{
    //ray origin and destination. Pixel depends on its
```

```
    Ray eyeRay;
    eyeRay.o = make_float3(...);
    eyeRay.d = normalize(u, v, mode.zoom);

    //ray tracing with big step
    sample = getSample( ...);


    //if ray tracing find positive value
    if( sample != -1 )
    {
        //ray tracing with small step
        t-=tBigStep;
       getSample(eyeRay,&t, ...);
    }
}
```

In questa versione semplificata e ripulita di numerosi controlli si possono analizzare le parti fondamentali del codice.
Per prima cosa viene definito il raggio con cui l'algoritmo di ray-tracing andrà ad operare. Ogni pixel dell'immagine 2D di proiezione del tronco avrà un raggio diverso. Una volta definito il raggio viene richiamato l'algoritmo di ray-tracing con l'istruzione getSample(), una funzione che poi andremo a vedere in dettaglio. Se l'algoritmo incontra un punto del tronco si ferma, ritorna al passo precedente in cui non era ancora entrato in contatto col tronco e riesegue un nuovo ciclo di ray-tracing, questa volta con passo di campionamento ridotto di un fattore 10.

La prima versione utilizzava un passo di campionamento a metà tra i due fin dall'inizio col risultato di un'immagine seghettata e poco pulita. Ridurre il passo si è rilevato impossibile perchè il software non era più in grado di funzionare in tempo reale. Questa soluzione ha permesso di ottenere entrambi i risultati in maniera ottimale. L'unico problema residuo è la presenza di alcuni puntini neri su certi bordi del tronco. Questo è dovuto al fatto che il ray-tracing funzionando a passo di campionamento grande perde alcuni punti del bordo che si trovano a metà tra un passo ed il successivo. Per ovviare anche a questo problema è stato modificato l'algoritmo in modo di rieseguire un ray-tracing a passo intermedio solo nei pixel risultati “vuoti” che ne affiancano altri che invece hanno incontrato un punto del tronco.

Ray tracing

Di seguito l'algorimo di ray-tracing, semplificato lasciando in evidenza solo il codice fondamentale.

```
float3 getSample(...)
{
       //position of light
       lightDir=make_float4(lightDirX, lightDirY, lightDirZ, 0.0f);
       lightDir=mul(c_invViewMatrix,lightDir);
       lightDir = -normalize(lightDir);

       //RAY TRACING ALGORITHM
       while(*t <= tfar)
       {
              // position in mm (needed to check limits)
              pos = eyeRay.o + eyeRay.d*(*t);
              // position in pixel (for memory access)
              posPixel = mmToPixel3(pos,boxMm,ref);
              // position normalized to [0,1]      (for texture)
              posPixelTex=make_float3(...);


              // CHECK LIMITS

              // CHECK MOULDINGS
```

```
                sample = tex3D(tex, posPixelTex);

                //THRESHOLD 1 : trasparency

                //THRESHOLD 2 : shading

                *t += tstep;
        }

        return make_float3(R,G,B);
}
```

Per prima cosa la funzione getSample() definisce la posizione della luce, necessaria per l'applicazione dell'ombreggiatura (shading) al tronco.
L'algoritmo di ray-tracing partendo dall'origine della camera, prosegue di passo in passo secondo la direzione del raggio della camera, finchè non esce dal box di definizione dello spazio 3D o non trova un punto del tronco. Una volta calcolata la posizione attuale del raggio vengono fatti dei controlli su tale posizione : se il punto si trova fuori dai limiti di taglio o della maschera di profilo allora l'algoritmo prosegue facendo avanzare di un passo il raggio ed ignorando il punto attuale. Se il blocco di controllo viene superato, l'istruzione evidenziata in verde permette, data una posizione dello spazio, di misurare il valore del voxel corrispondente. A questo punto vengono eseguite delle operazioni per l'applicazione delle rifrazione e dell'ombreggiatura.

Vediamoli in dettaglio.

```
//THRESHOLD 1 : trasparency
if( mode.trasparencyOn && ... )
{
   R *= (1-prevR*tstep);
   G *= (1-prevG*tstep);
   B *= (1-prevB*tstep);
}
```

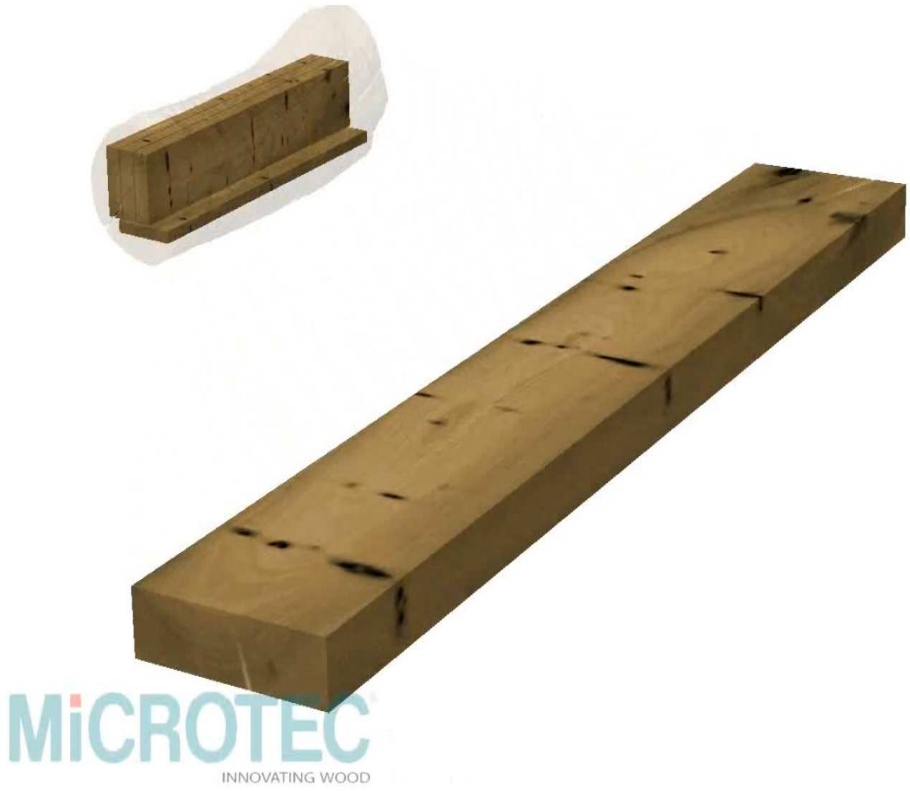
Viene eseguita la stessa operazione su ciascuna delle tre componenti di colore del pixel. Ciascuna di esse viene attenuata tenendo conto del valore della stessa componente al passo precedente. In questo modo ogni pixel terrà conto non solo del primo punto del tronco incontrato ma andrà a sommare anche i punti incontrati successivamente, riducendone di volta in volta il contributo.

```
//THRESHOLD 2 : shading

if( mode.lightOn )
{
   float shadow=mul(lightDir,normal);

   ...

   R *= shadow.x;
   G *= shadow.y;
   B *= shadow.z;

}
```

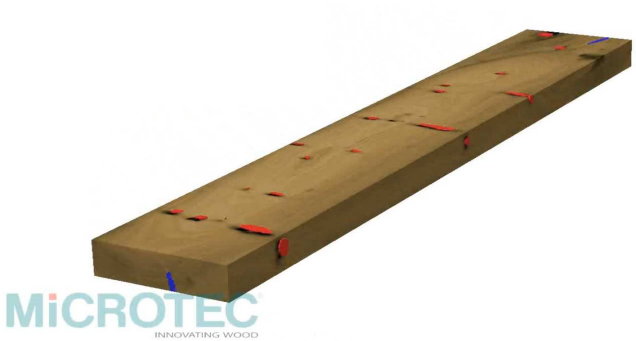
Se l'utente ha impostato l'effetto shading allora il ray-tracing compie un'ulteriore raffinazione sul pixel da mostrare. Viene calcolata l'ombra moltiplicando la normale alla superficie (calcolata prima del passaggio dati a cuda) per la direzione del raggio luminoso. Il valore, normalizzato, viene poi moltiplicato vettorialmente al pixel.

Intaglio e traslazione di singole tavole

Cuda ha reso possibile una nuova modalità d'intaglio. Grazie all'allocazione di memoria è stato possibile memorizzare come oggetti separati parti del tronco. In questo modo il tronco è stato suddiviso in tavole, dei parallelepipedi, le quali possono essere spostate rigidamente nello spazio indipendentemente le une dalle altre. E' stata così possibile l'implementazione di un'animazione in cui il tronco viene tagliato in tavole le quali di volta in volta si accatastano una sopra l'altra.



Sulla sinistra il tronco viene tagliato in tavole. In primo piano la tavola appena intagliata viene analizzata nel dettaglio.



L'analisi delle impurità può essere applicato alla singola tavola del passo di lavorazione.

CONCLUSIONI E SVILUPPI FUTURI

Conclusioni

Gli obbiettivi iniziali, posti ad inizio progetto, sono stati rispettati. Il software finale riesce a svolgere tutte le funzionalità che sono state richieste :

- rendering grafico del tronco tomografato
- possibilità di interazione con esso con la possibilità di ruotarlo e traslarlo
- possibilità di tagliare il tronco lungo le sue facce
- applicazione di profili di taglio personalizzati caricati da file bitmap
- evidenziazione di nodi, crepe, umidità ed altre impurità
- visualizzazione trasparente e semitrasparente del tronco
- simulazione di processi di fabbricazione
- realizzazione di video presentativi e demo

Inoltre il software riesce a funzionare in tempo reale senza rallentamenti e mantenendo una qualità grafica sempre alta, senza mostrare bordi seghettati o imperfezioni grafiche non volute.
Questo è stato possibile grazie ad un'analisi in dettaglio dell'algoritmo di ray-tracing e ad uno studio su come poterne migliorare le presentazioni mantenendo inalterata la qualità grafica.

Sviluppi futuri

Tra gli sviluppi che il futuro del software prevede il principale è quello di integrare la possibilità di una proiezione 3D dell'immagine tramite occhiali stereoscopici. Per farlo il software dovrà essere in grado di convertire l'output, al momento attuale un'immagine 2D, in due immagini ottenute da esso tramite alcune operazioni. Per farlo l'immagine non deve più essere presa da una sola camera ma da due posizionate alla distanza adeguata : quella presente tra i due occhi di una persona. In questo modo le due immagini possono essere inviate agli occhialini 3D i quali si occuperanno delle loro gestione, montandole assieme e dando l'impressione che il tronco esca dallo schermo.

Steroscopia

La stereoscopia è una tecnica di realizzazione e visione di immagini, disegni, fotografie e filmati che permette di dare una illusione di tridimensionalità simile a quella generata dalla visione del sistema visivo umano.

Seppur la maggior parte delle persone pensano che questa tecnica sia molto recente, soprattutto per via dell'introduzione solo recentissima delle proiezioni 3D nei cinema, le prime applicazioni di stereoscopia risalgono al 1800. Già negli anni venti gli stessi Fratelli Lumiere, inventori del cinematografo, produssero una copia del loro primo film in 3D.

Nel 1838 sir Charles Wheatstone riuscì a costruire un oggetto che chiamò stereoscopio il quale combinando una serie di specchi e prismi permetteva la visione di una immagine tridimensionale ottenendola da una immagine bidimensionale.

Nel 1852 venne creata la prima macchina fotografica in grado di scattare foto in 3D che prese il nome di fotocamera binoculare.

Questi tipi di immagini 3D venivano prodotte su cartoncino ma vennero ben presto abbandonate. Nel XX secolo la ricerca approdò verso una nuova tecnologia che utilizzava diapositive su pellicola fotografica. Negli stessi anni si riuscì a produrre l'effetto stereoscopico senza l'utilizzo di nessun supporto ottico per la visione come occhiali o stereoscopi.

Negli anni '50 la stereoscopia si afferma nel mondo del cinema e più di 60 film in 3D vennero prodotti. Il cinema in 3D andò presto in declino, riprendendosi tra la fine degli anni '70 e la metà degli anni '80 per poi sparire di nuovo. Solo negli ultimi anni il cinema 3D è riuscito a riaffermarsi e diffondersi in tutti i cinema moderni.

Come implementarla

Per poter rendere il software in grado di produrre immagini 3D è per prima cosa necessario utilizzare due camere anziché una e produrre quindi due immagini diverse. Saranno poi gli occhiali 3D ed il loro software che si occuperanno di come metterle assieme per dare all'utente l'impressione di vedere i tronchi uscire dallo schermo. Il software dovrà quindi inviare continuamente, in tempo reale, due flussi di immagini, una per l'occhio sinistro e una per quello destro. Gli occhiali 3D alterneranno la proiezione di queste immagini ad una determinata frequenza, scelta dai produttori degli occhiali, che permetterà l'effetto 3D voluto.

Una difficoltà che si potrebbe riscontrare è quella di non avere più una proiezione in tempo reale perché il carico di dati prodotti viene raddoppiato : serve quindi una efficienza doppia e bisogna quindi trovare il modo di ottenerla. Un modo potrebbe essere quello di scegliere un hardware più potente, un altro quello di modificare ulteriormente l'algoritmo di ray-tracing aggiungendo controlli che riducano passi inutili o poco rilevanti dell'algoritmo in determinate situazioni. Probabilmente la soluzione ottimale sarà un mix di entrambe.

BIBLIOGRAFIA

- nVidia CUDA reference site, http://www.nvidia.it/object/cuda_home_new_it.html

- CUDA api reference manual, http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_Toolkit_Reference_Manual.pdf