Elaborazione di Dati Tridimensionali
Progetto finale: CNC simulator

Franzin Alberto · 1012883
Gobbo Nicola · 1014195

22 novembre 2012

0.1 Descrizione del problema

Il progetto consiste nello scrivere un simulatore di macchina a controllo numerico (CNC - Computer Numerical Control) per il controllo della fresatura di un blocco di materiale.

Le specifiche fornite richiedono che il progetto sia eseguibile sia in ambiente Windows/Visual Studio che Linux. Viene anche fornito un diagramma con le principali classi da implementare, per uniformità. Viene richiesto che il simulatore accetti in ingresso un file contenente una lista di posizioni del blocco di materiale e dell'utensile di lavorazione, e la precisione della lavorazione, e che sia in grado di elaborare i movimenti comportati dalla lavorazione e di mostrare l'avanzamento della fresatura.

0.2 Descrizione dei moduli

Nel nostro svolgimento, abbiamo suddiviso il lavoro nei seguenti moduli, che andiamo a descrivere.

0.2.1 Configurator

Il configurator è il modulo che si occupa di leggere i dati di ingresso. Il suo compito è quello di aprire il file contenente le posizioni del blocco e dell'utensile in ogni step registrato della lavorazione originale, e trasferirli al miller.

0.2.2 Miller

Il miller è il componente che ricostrisce la fresatura vera e propria, verificando dove e come l'utensile della macchina compenetra il blocco e rimuove del materiale.

Per gestire in maniera efficiente l'intero processo, è necessario usare come struttura dati di appoggio un Octree non bilanciato, ovvero un albero di arietà 8 che rappresenta molto efficacemente uno spazio tridimensionale. Ogni foglia dell'Octree rappresenta un parallelepipedo di volume e su di esso vengono salvate primariamente le informazioni sullo stato di erosione dei vertici, a cui si aggiungono, per motivi di performance, un rapido accesso alle informazioni sulle coordinate dei propri vertici e un collegamento alla struttura che visualizzerà quel voxel.

Si è deciso di non condividere i vertici comuni tra voxel contigui in quanto il concetto di vicinanza spaziale non viene modellato bene dalla struttura octree, soprattutto se sbilanciata, quindi, il costo computazionale di recuperare i voxel "vicini" sarebbe stato superiore ai vantaggi portati dalla condivisione.

Man mano che l'utensile si muove, il miller converte le rototraslazioni lette da file in una isometria tridimensionale del cutter nei confronti del prodotto. Per ogni "mossa" l'algoritmo di milling discende l'octree scegliendo i rami da percorrere in base a diverse funzioni di intersezione che diventano via via meno precise, ma più veloci, man mano che la profondità aumenta. Giunti alle foglie si verifica se alcuni dei loro vertici risultano interni alla superficie di taglio del cutter, venendo quindi erosi. Le foglie rimaste prive di vertici vengono quindi eliminate dall'albero, mentre per le altre, se la profondità massima non è ancora stata raggiunta, si effettua una divisione in otto parti del loro volume di competenza, aggiungendo un nuovo livello all'albero.

Le operazione effettuate, ovvero rimozione di materiale e modifica della forma del blocco, devono venir comunicate al processo di meshing, il quale, tuttavia, ha tempi di lavorazione differenti dal miller. Per gestire in maniera efficiente l'accesso concorrente ai dati, ogniqualvolta una foglia viene cancellata essa è inserita dal miller in una lista opportuna. Per le foglie aggiunte o modificate, invece, il percorso da esse alla radice viene marcato con un numero di versione incrementale: così facendo il processo di meshing potrà ricavare rapidamente tutte e sole le foglie modificate dalla sua ultima visita.

0.2.3 Mesher

Il mesher è il componente che, a partire dalla lavorazione effettuata dal miller, crea la mesh 3D dell'oggetto lavorato, traducendolo quindi in un oggetto tridimensionale da visualizzare.

Quando è pronto a eseguire del lavoro, richiede all'oggetto che rappresenta il prodotto le ultime modifiche effettuate: per tutte le foglie cancellate o aggiornate vengono eseguite le relative operazioni in

maniera diretta, sfruttando il puntatore all'oggetto grafico contenuto; quelle nuove, invece, vengono inserite nella scena solo se effettivamente visibili. Per facilitare il processo di visualizzazione, la parte di scena rappresentante il prodotto viene modellata con un octree le cui foglie contengono una molteplicità di voxel da rappresentare: questa scelta permette di ottimizzare l'uso delle risorse grafiche in quanto si diminuisce il numero di mesh, aumentandone la dimensione.

Il mesher esegue l'algoritmo Marching Cubes per creare la mesh a partire dai voxel. Marching Cubes permette inoltre di scartare dalla mesh quei voxel che sicuramente non sono visibili, perché totalmente interni o totalmente esterni (quindi eliminati) alla mesh.

0.2.4 Visualizer

Il visualizzatore, infine, è il componente che mostra l'elaborazione in corso e il risultato finale. Il programma può operare in due modalità: `mesh` e `box` (in più, è anche disponibile la modalità operativa che esclude la rappresentazione grafica). La prima visualizza la mesh come generata dal Mesher usando Marching Cubes, mentre la seconda sfrutta gli oggetti `Box` di OpenSceneGraph ed è meno accurata, permettendo però di bypassare Marching Cubes.

Il visualizzatore mostra inoltre in tempo reale informazioni relative alla lavorazione, come ad esempio il numero di frame al secondo o l'attivazione o meno del getto d'acqua.

0.3 UML del moduli

0.4 Strumenti usati, prerequisiti e istruzioni

0.4.1 Strumenti usati

Il progetto è stato sviluppato in C++. Per lo sviluppo in ambiente Linux abbiamo usato Eclipse su Ubuntu 12.04, mentre per l'ambiente Windows è stato usato Visual Studio 2010. Lo strumento usato per la compilazione è CMake ($\geq$ 2.6).

0.4.2 Prerequisiti

Il progetto è stato sviluppato usando le seguenti librerie:

- Boost ($\geq$ 1.48)
- Eigen ($\geq$ 3.1.1)
- OpenSceneGraph ($\geq$ 3.0.0)

0.4.3 Istruzioni

Per compilare il progetto bisogna seguire i seguenti passi:

1. portarsi nella cartella `/path/del/progetto/`
2. lanciare il comando `cmake source/CMakeLists.txt`
   - se in Windows, aggiungere il flag `-G"Visual Studio 10"`
   - se in linux, aggiugnere il flag `-G"Unix Makefiles"`
   - se si vuole compilare in modalità `debug`, aggiungere il flag `-D CMAKE_BUILD_TYPE=Debug`
   - se si vuole compilare in modalità `release`, aggiungere il flag `-D CMAKE_BUILD_TYPE=Release`
3. lanciare il comando `make` per compilare.

Per eseguire il progetto, lanciare il comando
`/path/del/progetto/CNCSimulator opzioni file_positions`
dove:

- le opzioni possono essere:

1. `-s x`, dove `x` è la dimensione minima dei voxel. Minore è `x`, maggiore è la precisione della simulazione
2. `-v box|mesh|none`, per specificare il tipo di visualizzazione
3. `-p` lancia il simulatore in pausa
4. `-f x`, dove `x` indica il rate di apertura del getto d'acqua per la rimozione dei detriti in eccesso
5. `-t x`, dove `x` indica la quantità di materiale da rimuovere prima di attivare il getto d'acqua
6. `-h`, per visualizzare il menu di help completo

- `file_positions` è il file contenente i movimenti da riprodurre.

0.5 Esempio di lavorazione

0.6 Conclusioni

Il simulatore mostra come procede passo dopo passo il lavoro di fresatura, permettendo di regolare precisione della lavorazione e del rendering e la velocità di visualizzazione.

Con i file di esempio messi a disposizione, le operazioni vengono portate a termine correttamente e con buone prestazioni, nonostante non sia stato possibile sfruttare CUDA perché nessuno di noi ha a disposizione una scheda grafica Nvidia.