

Tesina di Informatica Grafica **Matteo Mecucci - 23/06/03**

Il presente lavoro si pone come obiettivo quello di mostrare alcuni concetti della computer grafica affrontati durante il corso, così come alcuni sviluppi personali di idee e tecniche che esulano dal programma del corso ma che sono comunque inerenti il campo di interesse. Tali idee provengono principalmente dal libro di testo ([1]) e sono state sviluppate sulla base di esso e di altri testi specifici, in particolare [2] e [3].

Lo sviluppo della tesina si è diviso in due passi distinti: il primo è stato la realizzazione di un parser di grammatiche di Lindenmayer per la generazione a tempo di esecuzione di sistemi poligonali complessi; il secondo è stato la creazione di un sistema di rappresentazione adeguato ai sistemi generati, basato su OpenGL.

Per il primo passo è stato usato come punto di partenza un vecchio parser realizzato in C, trovato nella ricerca di documentazione sull'argomento su Internet. L'autore è Laurens Lapre, ed i sorgenti sono rilasciati in forma gratuita per fini non commerciali. Il parser in questione risale alla metà degli anni 90 ed il sorgente è stato completamente ristrutturato, a partire dalla conversione in C++, fino all'adattamento alle necessità specifiche del programma in esame. In particolare sono state sostituite le routine che creavano file di output in formati più o meno noti (DXF, POV, ecc.) con chiamate di callback verso il client, in questo caso la parte del software che si occupa di compilare le liste OpenGL per il successivo rendering. Sono inoltre state modificate le routine di generazione delle mesh poligonali, in modo che esse generino anche le corrette normali nei vertici (nel senso della computer grafica, ovviamente), per poter così usare il modello di illuminazione Smooth nel rendering. Sono state infine fatte delle modifiche per aumentare la velocità della generazione dei sistemi, con un miglioramento nelle prestazioni piuttosto incoraggiante.

Per la realizzazione della parte grafica, ci si è posti fin da subito l'obiettivo aggiuntivo di offrire qualcosa di più di una semplice e fredda renderizzazione di un modello tridimensionale. Si è cercato di perseguire questo obiettivo cercando di realizzare una collocazione ambientale adeguata ai modelli generati, cercando di non impattare troppo sulle prestazioni del rendering. Poiché i modelli generabili con grammatiche di Lindenmayer più comuni ed affascinanti sono piante, ci si è concentrati solo su questo tipo di modello. L'ambiente di cui si è voluto circondare le piante è un semplice e ristretto panorama cittadino, in cui le piante sono ormai solo a decoro di mattoni ed asfalto. Esso è modellato a tempo di esecuzione in base alle dimensioni della pianta da rappresentare, con l'uso di semplici primitive come quadrilateri, parallelepipedi e coni. Unica eccezione è la gabbia che circonda la base della pianta, generata anch'essa tramite il parser. Ad ogni oggetto rappresentato è stata assegnata una texture adeguata; tutte le texture, tranne quella del cono di luce, sono lette da file Jpeg attraverso una libreria di pubblico dominio (licenza LGPL) realizzata da Rich Geldreich. Quella del cono di luce è invece una semplice texture monodimensionale generata a tempo d'esecuzione ed utilizzata come maschera di trasparenza.

Particolare cura nella realizzazione dell'engine grafico si è cercato di dare all'illuminazione della scena, vero punto cardine della qualità di un'immagine generata dal calcolatore. Sono presenti due luci, una all'infinito con direzione non fissa, l'altra puntiforme e fissata nella scena come una lanterna appesa sopra la strada. Le due luci sono attive alternativamente come illuminazione diurna e notturna. Entrambe proiettano ombre degli oggetti (in particolare della pianta, della lanterna e del muro) sul piano del pavimento e su quello del muro. Le ombre sono realizzate con una tecnica sviluppata a partire da quanto scritto principalmente in [3], con qualche spunto da [2] e con il supporto della teoria di [1].

Oltre al programma vero e proprio, sono necessari all'esecuzione diversi file. Tra essi ovviamente le già citate texture in formato Jpeg, nonché i file con estensione ".ls" (L-Systems) che definiscono istanze della grammatica di Lindenmayer ([4]) usate nella tesina. In particolare sono presenti quella che genera la gabbia e quattro che generano tipi diversi di piante. La prima, molto semplice, è originale, mentre le altre sono state modificate a partire da esempi distribuiti col parser originale suddetto e sono quindi da considerarsi dello stesso autore. Le texture, viceversa, sono tutte originali, create da zero con il pacchetto GIMP o a partire da foto scattate per l'occasione. Allegato alla tesina c'è infine il documento [4] originalmente distribuito con il parser di Laurens Lapre, in cui viene descritto - in modo non propriamente formale - il formato della grammatica accettata dal parser.

Appendice: esecuzione e tasti di controllo del programma

--

Per l'esecuzione, e' sufficiente che tutti i file della tesina siano nella directory corrente. L'eseguibile puo' essere lanciato senza parametri, anche se in ambiente X-window possono essere specificati parametri diretti alla libreria GLUT. Il programma e' stato sviluppato e testato approfonditamente solo in ambiente Windows con una scheda NVidia Geforce 4 MX. Brevi test apparentemente positivi sono stati condotti sempre in ambiente Windows con hardware diverso. Viceversa brevi test su Macintosh hanno dato risultati visivamente errati probabilmente perche' la libreria di lettura delle texture oppure il passaggio delle texture a OpenGL prevedono che il processore abbia un'architettura little-endian.

Una volta lanciato, il programma generera' tutti i modelli necessari, compilandoli in liste OpenGL. Sullo standard output possono essere letti il numero di poligoni generati per ogni sistema di Lindenmayer e le coordinate dei bounding box relativi. Successivamente viene creata una finestra con dimensioni 800x600, ridimensionabile, in cui viene rappresentato lo scenario della prima pianta.

Durante l'esecuzione possono essere usati i seguenti tasti:

- 1: attiva/disattiva la rotazione della telecamera
- 2: attiva/disattiva la rotazione della luce all'infinito
- 3: cambia metodo di rendering della pianta: FILL/WIREFRAME/POINT
- 4: cambia metodo di rendering dell'ambiente
- G: rigenera il modello corrente (stessa grammatica, diverso seed random)
- L: cambia luce: giorno/sole oppure notte/lanterna
- S: attiva/disattiva le ombre
- Esc: esce dal programma
- qualunque altro tasto: cambia scenario

Reference

--

- [1] Computer Graphics: Principles And Practice, 2nd edition - Foley et al., Addison Wesley,
- [2] OpenGL Programming Guide, 3rd edition - OpenGL ARB, Addison Wesley,
- [3] Improving Shadows and Reflections via the Stencil Buffer - paper by Mark J. Kilgard, Nvidia Corporation,
- [4] LParser.txt - Laurens Lapre, descrizione della grammatica di Lindenmayer accettata dal parser dell'autore.

Addendum – 21/06/2005

--

Dalla data di realizzazione ad oggi sono passato in pianta stabile alla piattaforma Macintosh, dove attualmente la tesina sembra funzionare perfettamente, e di cui è distribuito il progetto per XCode.

Per le altre piattaforme è necessario compilare tutti i file cpp contenuti nelle varie cartelle e linkare i file oggetto con le librerie OpenGL e GLUT.

È separatamente disponibile un archivio con l'eseguibile per Windows, la DLL di GLUT e un file batch per la compilazione con il compilatore Visual C++ di Microsoft, disponibile gratuitamente come parte del pacchetto Toolkit 2003:

<http://msdn.microsoft.com/visualc/vctoolkit2003/>