Generazione procedurale di edifici tramite parametri

Progetto del corso di Computer Graphics - A.A. 2017/2018

di Alba Bisante, Marco Cruciani, Vincenzo Di Ronza, Michele Ferraro

Introduzione

Il presente progetto si prefiggeva inizialmente la realizzazione di un sistema basato su grammatiche. Per difficoltà realizzative, descritte in seguito, si è preferito basare la modellazione su un sistema parametrico, concentrandosi sulla semplicità di utilizzo per l'utente finale, dando la possibilità di creare edifici variegati, grazie all'ausilio di tecniche di probabilità, con un numero minimo di righe di codice, come mostrato nell'applicazione dimostrativa allegata. Si descrivono quindi i punti salienti dell'implementazione.

1. Grammatiche e sistemi parametrici per la generazione procedurale

Si è inizialmente tentato di implementare il sistema di generazione tramite grammatiche context-free. Diverse sono state le difficoltà che hanno spinto verso l'abbandono di tale idea. Tra queste è sicuramente presente la semplice difficoltà implementativa, che costringe a visite degli alberi di derivazione che sono da una parte non semplicissime da realizzare (e quindi difficilmente estensibili qualora si desideri aumentare le possibilità creative), ma, soprattutto, in cui risulta difficile astrarre dal contesto – obiettivo primo dell'utilizzo di una CFG – nel caso in cui, per ottenere il valore di un parametro, bisogni percorrere la struttura di derivazione in maniera nota a priori, che porterebbe quindi necessariamente a definire e "cristallizzare" le regole di derivazione a meno di approcci molto più sofisticati. Può anche risultare complesso per l'utente definire nuove regole nel caso in cui l'utente volesse aggiungerne di proprie. Ultimo punto, il codice prodotto risultava verboso e poco espressivo.

Il sistema parametrico ha tra i vantaggi principali la presenza di una struttura di parametri semplice ed unificata ed un'implementazione il più possibile modulare, facilmente estensibile con nuovi parametri tramite una creazione del modello divisa in vari step su cui è possibile intervenire singolarmente.

È comunque proposta, anche se inutilizzata, una semplice implementazione di CFG, con inclusa la possibilità di eseguire derivazioni left-most.

2. Probabilità

Punto centrale della generazione è la possibilità di avere a disposizione utility di probabilità per ottenere risultati non deterministici (a meno dell'RNG). Un intero file sorgente (nella versione multifile, disponibile su https://github.com/micferr/ProgettoGraphics) è dedicato quindi a tali funzioni. Sono incluse funzioni per variabili aleatorie con distribuzioni uniformi, di Bernoulli e normali, ed applicazioni di queste, come la possibilità di scegliere un elemento da un vettore, in maniera sia uniforme che pesata. Tra i parametri

probabilistici più importanti ricordiamo quelli relativi alle finestre (distribuzione lungo le facciate dell'edificio, distribuzione di finestre aperte e chiuse) ed espansione ricorsiva verso l'alto degli edifici.

3. Modellazione tra 2D e 3D

A causa dell'inagevolezza nel processo di creazione di figure piane in uno spazio tridimensionale, sono stati realizzati metodi per passare in maniera quasi ininterrotta tra le modellazioni in 2D e 3D. In particolare, sono state ampiamente usate due funzioni di conversione tra i due sistemi (to_3d e to_2d), che convertono una coordinata da due a tre dimensioni e viceversa. Nello specifico, to_3d è una funzione che, preso in input un punto (x,y) (o una serie di punti) e un valore Y, genera il punto tridimensionale (x,Y,-y). In questo modo si può disegnare il contorno della figura sul piano XY (paragonabile a un foglio di carta visto frontalmente) e subito riportarlo, in tre dimensioni, sul piano XZ, soprattutto mantenendone l'orientamento originale (orario o antiorario) in modo da essere subito utilizzabile, ad esempio come pavimento di un edificio. to_2d, chiaramente, effettua l'operazione inversa.

Il workflow proposto ed implementato per la creazione di modelli tridimensionali consiste nel disegnare forme in 2D, convertirle in 3D e quindi estruderle quanto necessario.

4. Definizione degli edifici

Sono state proposte tre diverse tipologie di edificio, ognuna con i propri parametri: edifici di pianta custom, edifici regolari, ed edifici definiti tramite "main points" (per utilizzare la terminologia del codice). La prima tipologia è stata poco utilizzata nella pratica, e serve più per mostrare le possibilità di espansione del sistema. Un edificio regolare è semplicemente un edificio, simile a una torre, la cui pianta sia un poligono regolare. È però sulla terza categoria che si è concentrato lo sviluppo. Un edificio definito tramite main points ha una pianta definita tramite una sequenza di punti, che ne mostrano la linea di sviluppo come una linea spezzata, e un valore di larghezza, che indica appunto quanto espandere tale linea. Un vantaggio di questo modello è la possibilità di avere non solo un'indicazione geometrica della forma della costruzione, ma anche una direzione di sviluppo (necessaria, ad esempio, per tetti di tipo crossgabled, applicabili solo a questa tipologia di edifici).

5. Espansione ricorsiva

Tra i parametri probabilistici più importanti vanno necessariamente menzionati quelli relativi all'espansione ricorsiva degli edifici, che fa un uso molto efficace del sistema a main points. Viene utilizzata la funzione random_substrings per ottenere liste disgiunte di main_points consecutivi che vanno a formare la base per creare ricorsivamente le strutture superiori. In maniera molto semplice a livello di implementazione e l'uso di alcune euristiche estetiche si sono ottenuti risultati interessanti con poco sforzo.

6. Euristiche

I risultati principali mostrati nei render di esempio sono stati ottenuti tramite tuning sia dei parametri procedurali (in particolare i range delle variabili aleatorie e le relative distribuzioni di probabilità) in modo da ottenere risultati non solo esteticamente piacevoli ma anche plausibili. È questo che giustifica molti valori di default, in particolare quelli autogenerati da make_rand_building_params. Ne commentiamo alcuni.

Nella generazione ricorsiva degli edifici, si fa sempre in modo di tagliare via almeno un main point ad ogni piano: questo evita a priori la creazione di edifici improponibilmente alti. Quando restano sequenze di un solo main point, tale punto viene invece usato come centro per la costruzione di un edificio di tipo regolare. Inoltre, parti di edifici generate ricorsivamente ad uno stesso step (cioè che poggiano su uno stesso tetto) condividono la totalità dei parametri, ad eccezione di quelli che ne descrivono la pianta.

Diversi edifici sono anche affiancati da un'altra costruzione di tipo regolare, questa viene posta affiancata al lato più lungo. Questa intuizione risulta particolarmente utile negli edifici con main point, in quanto il lato più lungo è sempre seguito, seguendo la pianta in senso antiorario, da una svolta a sinistra, che minimizza la probabilità che la torre appena aggiunta collida col resto dell'edificio (è una politica sicuramente non perfetta, ma comunque cost-effective rispetto alla semplicità dell'implementazione, e comunque facilmente modificabile in quanto l'argomento che si occupa della decisione è funzionale).

7. Librerie esterne

La realizzazione del progetto è stata semplificata dall'uso di due librerie esterne, Poly2Tri (https://github.com/greenm01/poly2tri) e Clipper (http://www.angusj.com/delphi/clipper.php). Poly2Tri è risultata particolarmente utile per la triangolazione delle piante degli edifici: se per edifici con main point o regolari si riesce facilmente a fare a meno della triangolazione, edifici di forme più complesse necessitano tale tecnica. La creazione di questi edifici, seppur lasciata in secondo piano, è pienamente funzionante nell'implementazione corrente. La possibilità di triangolare poligoni arbitrari, anche con buchi interni, può comunque risultare utile per sviluppi futuri.

Per quanto riguarda Clipper, la libreria è stata usata "soltanto" per espandere la pianta degli edifici nella generazione dei "cornicioni" che separano i vari piani. L'operazione di offsetting dei poligoni presentava infatti molti edge-case che rendevano estremamente difficoltosa un'implementazione manuale.

8. Risultati

Si mostrano di seguito alcuni risultati della libreria, renderizzati tramite l'app yitrace inclusa in Yocto/GL.