Tecniche di visualizzazione procedurale in 2D e 3D

Matteo Fiorito, Ivan Miraglia, Oreste Tommasi

GeoMind Srl (Gruppo Abaco SpA)
Via Carducci 64/D, 56010 Ghezzano (PI), tel 050 9911062 fax 05 9911063 email: geomind@geomind.it

Riassunto
Le tecniche di visualizzazione cartografica hanno finora utilizzato principalmente dati vettoriali e tematismi per la produzione di carte tecniche, oppure immagini aere o satellitari per riproduzioni a scopo più foto-realistico, tipicamente in 3D. Nel presente articolo sono illustrate delle tecniche di visualizzazione foto-realistica ottenuta solo da dati di tipo vettoriale tematico: questo approccio, denominato "visualizzazione procedurale", consente di ricostruire rappresentazioni artificiali ma verosimiglianti senza l'utilizzo di ortofoto, con un elevato livello qualitativo e di dettaglio, e con un'enfasi particolare alla ricostruzione della vegetazione. Per applicazioni sia di rappresentazione del mondo reale, sia per scopi di progettazione, le immagini procedurali ottenute con questa tecnica possono essere prodotte in 2D/3D in *real time* su ampie zone di territorio.

Abstract
So far, cartographic visualization techniques have mainly used 1) vector and thematic data for the production of topographic maps, or 2) aerial and satellite images for photo-realistic reproduction purposes, typically in 3D. In this paper, we illustrate techniques for photo-realistic visualization, obtained from vector thematic data. This approach, called "Procedural Visualization" (Ebert et al, 2002), allows to reconstruct artificial but realistic representations, without using orthophotos, with a high level of quality and detail, and with a particular emphasis on the reconstruction of vegetation. Either for applications of representation of the real world, either for design purposes, the procedural images obtained with this technique can be produced in 2D/3D in real time and on large areas.

Introduzione
La riproduzione foto-realistica di terreni naturali è una delle sfide classiche della *computer graphics*, ed è stata estensivamente utilizzata in ambito ludico; ciò ha costituito una spinta fenomenale nello sviluppo di hardware grafici sempre più potenti e sofisticati. Tale riproduzione non solo necessita di una notevole quantità di dettagli geometrici legati al terreno, alla vegetazione, agli edifici e ad altri oggetti, ma, per raggiungere risultati convincenti, deve anche fare i conti con l'intrinseca irregolarità (e complessità) degli elementi e dei fenomeni naturali.

In ambito GIS, dopo un'iniziale diffidenza (dovuta al timore di confondersi con l'ambito ludico), l'utilizzo della visualizzazione 3D è diventata ormai uno standard, con l'utilizzo di ortofoto come texture di rivestimento. Tale approccio basato su ortofoto soffre tuttavia del limite dovuto al fatto che per ottenere dettagli sempre maggiori (es.: 1 cm/pixel) sono necessarie risoluzioni di acquisizione sempre più elevate, con evidenti problemi di dimensioni dei dati e di difficoltà dell'acquisizione stessa.

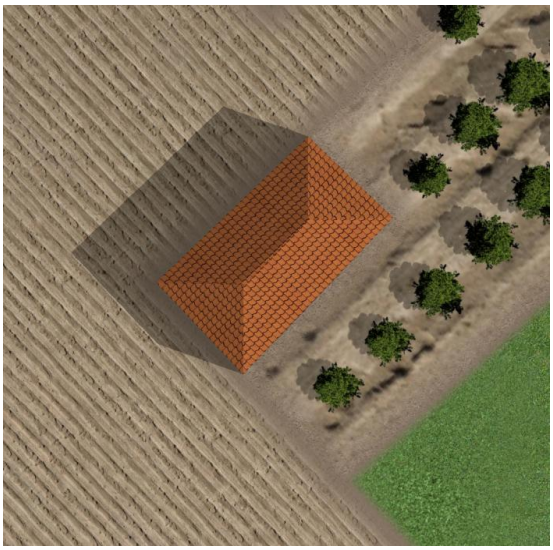


Figura 1. Esempio di visualizzazione procedurale in area agreste.

Inoltre, i metodi di riproduzione foto-realistica che si basano su dati reali acquisiti (es.: ortofoto, immagini oblique) non sono adatti per rappresentare studi di fattibilità o qualsiasi altra situazione non esistente nella realtà. Da alcuni anni ormai l'idea di ricostruire mondi artificiali in maniera procedurale è stata oggetto di numerosi studi e pubblicazioni (Ebert et al, 2002). A questo proposito sono disponibili software commerciali come *3D Nature* e *LandSim3D*, che consentono di costruire mondi del tutto nuovi; tuttavia necessitano di un notevole lavoro di editing e composizione, e non sono utilizzabili attraverso web services.

L'approccio di visualizzazione procedurale descritto in questo articolo utilizza una base dati vettoriale per riprodurre una rappresentazione verosimile e foto-realistica su zone molto ampie di territorio. Non è necessario che la base dati vettoriale abbia uno schema predefinito, è sufficiente che rappresenti una tematizzazione del suolo; quanto più il dettaglio di tematizzazione è avanzato, tanto più la rappresentazione risulterà realistica, ma l'approccio seguito consente di aggiungere dettagli "verosimili" anche in zone relativamente uniformi.

In questo articolo ci occuperemo in modo particolare della costruzione su larga scala di Texture Procedurali in 2D secondo l'approccio seguito in GeoMind; il passaggio in 3D è diretto per quanto riguarda le texture del suolo, mentre per gli elementi in rilievo (es.: vegetazione, edifici) la terza dimensione comporta la creazione procedurale dei relativi modelli.

Inizieremo dal trattare questioni generali riguardanti le Texture Procedurali, mostreremo qualche esempio, discuteremo aspetti come la proiezione di ombre e l'architettura del sistema, e valuteremo quindi questioni pratiche di performance di *rendering*.

Considerazioni generali sulle Texture Procedurali

Secondo la definizione di "Texture Procedurale" su Wikipedia:

A procedural texture is a computer-generated image created using an algorithm intended to create a realistic representation of natural elements such as wood, marble, granite, metal, stone, and others. Usually, the natural look of the rendered result is achieved by the usage of fractal noise and turbulence functions. These functions are used as a numerical representation of the "randomness" found in nature.

Alcune di queste texture (es.: marmo, granito, nuvole) possono essere create in maniera interamente procedurale (Ashikhmin, 2001), ossia soltanto per mezzo di formule matematiche che consistono nella combinazione di vari tipi di rumore e di funzioni di turbolenza. Nella rappresentazione del territorio queste texture "interamente procedurali" hanno un impiego relativamente limitato, in quanto poco adatte per descrivere campi coltivati e non, vegetazione, ecc.; in questi ultimi casi si fa spesso ricorso a texture base derivate da fotografie reali.

Ripetizione di pattern. Supponiamo, secondo un approccio semplicistico, di rappresentare un campo erboso per mezzo di un'immagine d'erba che si ripete in ogni direzione: anche se, con opportune tecniche, questa immagine di base viene resa periodica per evitare discontinuità sui bordi, è sufficiente un lieve zoom out per vedere chiaramente la ripetizione del pattern dell'immagine, con evidente effetto deleterio alla qualità della visualizzazione.

Texture splatting. Per evitare questo inconveniente di ripetizione di pattern si usa la tecnica denominata *texture splatting* (Bloom, 2000; Botsch et al, 2005), che consiste nel combinare due o più texture in maniera non regolare: tipicamente si utilizza una maschera di trasparenza prodotta per mezzo di una funzione di rumore bidimensionale denominata *simplex noise* (Perlin, 2001), mentre le texture che vengono miscelate hanno scala e direzione diverse.

Ripetibilità del rumore. Ovviamente tutte le funzioni casuali utilizzate nella visualizzazione procedurale devono essere di tipo *pseudo-random*, ossia devono apparire più irregolari possibile, ma devono essere ripetibili (cioè normalmente tornando nella stessa zona devo trovare la stessa situazione). Per garantire quest'effetto normalmente è sufficiente che il *seed*, ossia la fase iniziale del rumore, dipenda solo dalla posizione geografica.

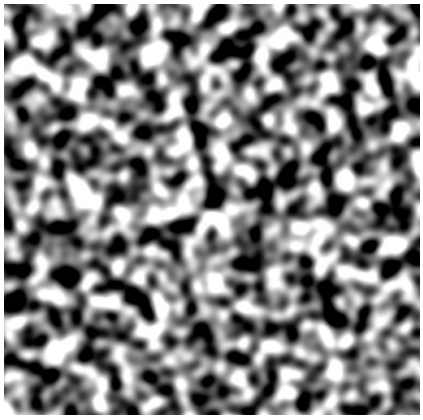


Figura 2. Esempio di rumore procedurale bi-dimensionale (simplex noise) utilizzato per combinare le texture in maniera non regolare.

Tutte queste operazioni (calcolo del rumore frattale, combinazione di texture, ecc.) possono risultare computazionalmente intensive, e quindi possono creare dei seri problemi di performance; tali problemi possono essere almeno parzialmente ovviati tramite l'utilizzo di alcune caratteristiche di programmazione (*Shading Language*) presenti ormai pressoché nella totalità delle moderne schede grafiche.

Visualizzazione a diverse scale. Una delle sfide più difficili nello sviluppo di Texture Procedurali consiste nell'ottenere un buon risultato visivo su un *range* di risoluzioni piuttosto vasto, che vada da scale ad elevato dettaglio (1-2 cm/pixel) a scale medio-basse (alcuni m/pixel). Per far ciò si possono utilizzare vari metodi, tra cui: a) inserire degli elementi di variabilità a diversa scala (es.

lieve modulazione della luminosità), anche in questo caso sfruttando *noise* di tipo *pseudo-random*; b) utilizzare delle texture base a risoluzione diversa, combinabili secondo le tecniche delle componenti principali.

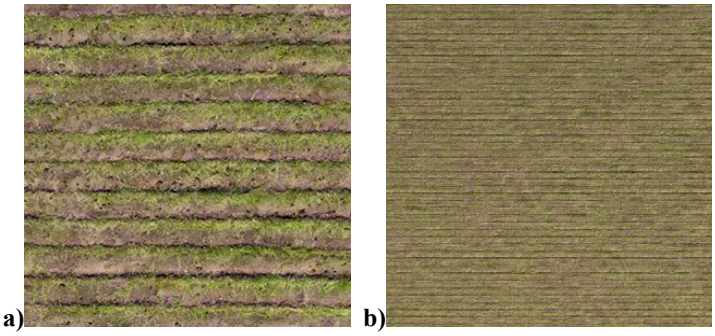


Figura 3. Texture procedurale di campo agricolo a diversi livelli di zoom: a) 5 cm/pixel; b) 20 cm/pixel.

Dal dato vettoriale all'immagine procedurale

L'approccio utilizzato in GeoMind non richiede che il database vettoriale di origine abbia uno schema prefissato, per cui possono essere utilizzati database estremamente generalistici come quello di OpenStreetMap, o database di tipo catastale/agricolo (es.: Figura 4).

Figura 4. Base dati vettoriale.

Figura 5. Immagine procedurale prodotta a partire dalla base vettoriale mostrata in Figura 4.

La classificazione dell'uso del suolo può anche essere non molto dettagliata: nell'esempio di Figura 4, molti poligoni sono classificati come "seminativi": per evitare eccessive ripetizioni, il sistema consente di associare diverse varianti alla stessa categoria; anche in questo caso l'associazione è random, ma ripetibile. Un possibile risultato è mostrato in Figura 5.

Vegetazione

Anche nella visualizzazione procedurale della vegetazione uno degli obiettivi principali consiste nel riuscire a rappresentare l'irregolarità della natura, pur partendo da informazioni vettoriali estremamente generiche (poligoni ricoperti da un certo tipo di vegetazione). La vegetazione si distingue per:

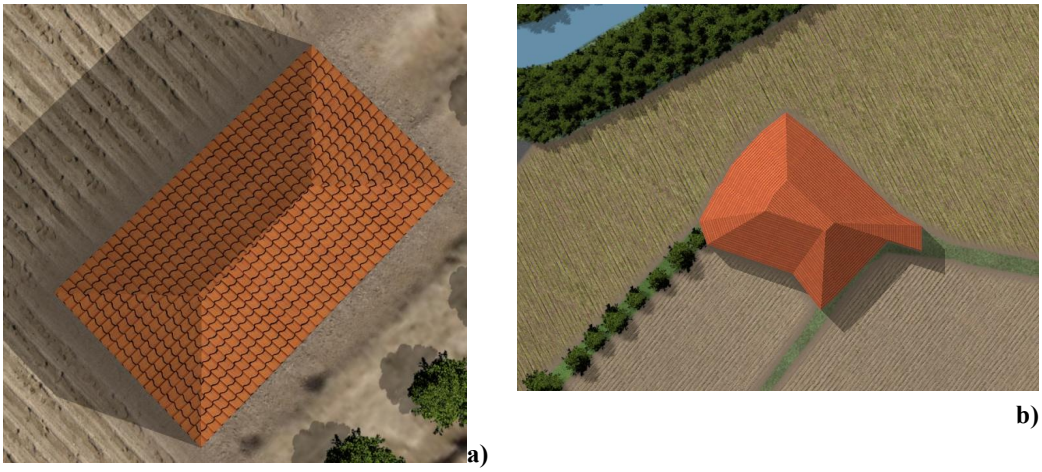
- tipologia di pianta
- densità di distribuzione
- regolarità di distribuzione.

Giocando su questi parametri, è possibile rappresentare pressoché qualsiasi tipo di vegetazione, dai filari di viti alle foreste.

Nel caso degli alberi, per ogni tipologia di pianta si utilizzano alcuni modelli predefiniti (4 sono sufficienti) realizzati con software specializzati nella produzione procedurale di modelli di piante (es. xfrog, http://xfrog.com/). Con questi pochi modelli diversi tra loro si possono ottenere comunque dei notevoli effetti di variabilità cambiando l'angolo di rotazione.

La vegetazione mostrata in Figura 6 è stata realizzata a partire da un numero molto limitato di modelli di piante diverse. Il frutteto al centro è stato realizzato con una distribuzione molto regolare (comunque con piccole variazioni di posizione); la foresta in basso a destra utilizza una elevata densità unita ad una maggiore casualità di distribuzione.

Figura 6. Esempi di vegetazione: filari di viti (in alto a destra e sinistra), frutteto (al centro), foresta (in basso a destra).



a) **b)**

Figura 7. Esempi di costruzione automatica di tetti di edifici: a) tetto regolare rettangolare; b) tetto poligonale estremamente irregolare.

Edifici

Nella visualizzazione procedurale di immagini 2D gli edifici compaiono esclusivamente per i tetti. In questo caso la necessità di dettagli è minore rispetto ad una visualizzazione 3D; partendo da database vettoriali generalizzati come quelli catastali, spesso l'unica informazione disponibile è la pianta dell'edificio (*footprint*) e al massimo l'altezza in gronda.

Con questi pochi elementi, una visualizzazione verosimile consiste nel costruire automaticamente al volo un tetto a spioventi, quale che sia la forma del poligono di *footprint*, per poi effettuarne un *rendering* convincente. La creazione automatica degli spioventi dei tetti viene effettuata per mezzo dell'algoritmo chiamato *straight skeleton* (Aichholzer et al., 1995).

Ombre proiettate

La visualizzazione delle ombre proiettate aggiunge un notevole effetto di realismo alle immagini 2D procedurali. Le superfici inclinate come gli spioventi dei tetti e i pendii dei rilievi hanno un effetto di shading che si calcola in maniera relativamente agevole, in quanto la variazione di luminosità che ne deriva dipende solo dall'angolo tra la superficie e la direzione della luce. Per le ombre proiettate la questione è più complessa, e volendola affrontare rigorosamente la complicazione computazionale può risultare molto consistente. Per questo motivo nel sistema descritto in questo articolo si fatta una scelta di semplificazione, che consiste nel considerare solo le ombre proiettate al suolo: l'agevolazione computazione che si ottiene con questa ipotesi è notevole, e nella maggior parte dei casi l'effetto che si ottiene è pressoché indistinguibile rispetto ad una trattazione rigorosa.

Gli oggetti che proiettano un'ombra sono quelli che hanno un rilievo, come edifici e vegetazione. Per questo motivo la sequenza di disegno risulta essere:

1. suolo
2. ombre proiettate sul suolo
3. oggetti in rilievo.

In quasi tutte le figure precedenti è ben visibile l'effetto dell'ombra proiettata sul suolo.

Configurazione ed architettura del sistema

In Figura 8 è mostrato lo schema a blocchi del sistema di visualizzazione procedurale..

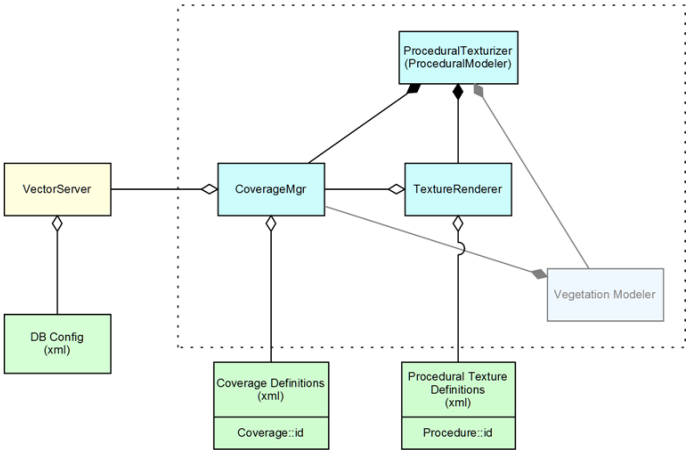


Figura 8. Diagramma dell'architettura del sistema.

Il ruolo dei moduli mostrati è il seguente:

- *VectorServer*: è il fornitore esterno di dati vettoriali, tipicamente costituito da un DB; non ha la minima informazione circa la visualizzazione procedurale;
- *TextureRenderer*: è il modulo capace di effettaure il rendering delle texture procedurali, secondo le definizioni di texture procedurali disponibili;

- *CoverageMgr*: è il modulo che fa da ponte, capace di trasformare generici dati vettoriali in modellazione procedurale; questo modulo è in grado di produrre anche vegetazione procedurale;
- *ProceduralTexturizer*: è l'interfaccia principale attraverso la quale si pilota il sistema, utilizzabile sia in programmi desktop, sia in web services.

La definizione delle singole texture procedurali (ossia di ogni materiale) è contenuta in un file XML (*Procedural Texture Definitions*): questa definizione contiene gli ingredienti necessari alla produzione della texture procedurale (texture base, noise), le regole di composizione ed i relativi parametri.

L'accoppiamento tra il DB vettoriale e le texture procedurali avviene per mezzo di un altro file XML (*Coverage Definitions*), che stabilisce, su ciascun poligono, quale procedura deve essere utilizzata, e ciò avviene per mezzo di semplici *query* definite sul DB.

Considerazioni sulle prestazioni

Per un sistema di rendering in real time come quello qui descritto, i "colli di bottiglia" dal punto di vista delle prestazioni sono essenzialmente due: 1) l'accesso al database, 2) il rendering.

Per quanto riguarda il rendering, l'uso di schede grafiche accelerate può essere di grande aiuto; tuttavia, poiché tale soluzione è meno facile da usare in sistemi di cloud computing, si è utilizzato un approccio basato su CPU *multi-core*.

Risoluzione (m/pixel)	DB Retrieval (ms)	Rendering (ms)	Totale (ms)
0.25	204	188	392
0.5	204	180	384
1	218	152	370
2	234	187	421
4	250	406	656
8	343	1576	1919
16	593	2184	2777

Tabella 9. Tempi di data retrieval dal DB e tempi di rendering su dei dati campione per un'immagine 512x512 pixel; test effettuato su un PC con processore Intel i5-3450 quad core, @3.10 GHz.

Come si osserva dal test riportato in Tabella 9, ad alte risoluzioni il tempo di rendering è inferiore al tempo di retrieval dal DB, per cui un'accelerazione della parte grafica porterebbe a dei benefici parziali.

A basse risoluzioni (≥ 4 m/pixel) il tempo di rendering diventa prevalente, in quanto a dimensioni fisse di immagine (in pixel) l'area geografica coperta aumenta, e il numero di oggetti presenti (soprattutto alberi) aumenta considerevolmente. In questo caso l'utilizzo di GPU potrebbe portare benefici consistenti; d'altra parte, si potrebbe anche utilizzare un'impostazione mista, in cui le alte risoluzioni sono renderizzate in real time, mentre le basse risoluzioni (che occupano meno spazio disco) sono pre-renderizzate.

Conclusioni e prospettive

Nel presente articolo si è preso in esame un approccio di visualizzazione procedurale 2D, per mezzo di un prototipo sviluppato da GeoMind, che presenta caratteristiche di totale flessibilità rispetto ai dati vettoriali di base, e con prestazioni che lo rendono utilizzabile in *real time* in sistemi *web services*. Particolare attenzione è stata dedicata all'ottenimento di buone prestazioni qualitative dalle

risoluzioni più elevate (1-2 cm/pixel) fino a risoluzioni medio/basse, senza ripetizione di pattern, ed in completa continuità al variare della risoluzione.

Lo *step* successivo, oltre che nel raffinamento e particolarizzazione delle texture procedurali sviluppate finora, consisterà nell'utilizzare questo approccio "immediato" anche in ambito di visualizzazione 3D con utilizzo di GPU.

Bibliografia

Aichholzer O., Aurenhammer F.; Alberts D.; Gärtner B. (1995). "A novel type of skeleton for polygons", Journal of Universal Computer Science 1 (12): 752–761. MR 1392429

Ashikhmin M. (2001) "Synthesizing natural textures", in Symposium on Interactive 3D Graphics, pp. 217–226

Bloom C. (2000), "Terrain Texture Compositing by Blending in the Frame-Buffer", disponibile online all'indirizzo: http://www.cbloom.com/3d/techdocs/splatting.txt

Botsch M., Hornung A., Zwicker M., Kobbelt L. (2005) "High-Quality Surface Splatting on Today's GPUs", in Eurographics Symposium on Point-Based Graphics.

Ebert D.S., Musgrave F.K., Peachey D., Perlin S., Worley S. (2002) *Texturing and Modeling: A Procedural Approach 3rd*, Publisher Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

Perlin K, (2001) "Noise hardware", in Real-Time Shading SIGGRAPH Course Notes, Olano M., (Ed.).