Università degli Studi di Bologna

Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea in Informatica

Materia di Tesi: Metodi Numerici per la Grafica

Generazione di paesaggi tridimensionali: *Un approccio OpenSource per applicazioni Real-Time e Web-Based*

Tesi di Laurea di:
Carlo Camporesi

Relatore:
Chiar.mo Prof. Giulio Casciola

Correlatore:
Dott. Luigi Calori

I Sessione
Anno Accademico 2004-2005

Università degli Studi di Bologna

Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea in Informatica

Materia di Tesi: Metodi Numerici per la Grafica

Generazione di paesaggi tridimensionali: *Un approccio OpenSource per applicazioni Real-Time e Web-Based*

Tesi di Laurea di:
Carlo Camporesi

Relatore:
Chiar.mo Prof. Giulio Casciola

Correlatore:
Dott. Luigi Calori

Parole chiave: OpenSceneGraph ; VTP ; Terreni ; Real-Time ; Web

I Sessione

Anno Accademico 2004-2005

Alla mia famiglia
e agli Addams.

Indice

Elenco delle figure

Capitolo 1

Introduzione

Negli ultimi anni, grazie all'influenza del nuovo businness dell'Home Entertainment, il campo della visualizzazione 3D ha raggiunto aspettative impensabili. La richiesta crescente di unità dedicate alla grafica, con prestazioni maggiori e sempre più realistiche, hanno portato ad un notevole abbattimento dei costi nella produzione di hardware specifico. Applicazioni che un tempo erano utilizzabili solo in costosissime WorkStation professionali oggi trovano largo impiego anche in semplici computer domestici.
Come conseguenza di questo fatto si presenta una richiesta crescente di software e tecniche in grado di generare e poter manipolare, in maniera semplice ed intuitiva, una maggiore quantità di dati. Utilizzando questi dati come base di partenza si possono ricreare modelli 3D con i quali è possibile avere un qualche grado di interazione e, se la fonte di partenza risulta essere di tipo scientifico, l'interazione diventa un metodo di studio.
Attualmente il campo di ricostruzione territoriale è dominato da una serie di aziende che, detenendo la maggior parte del mercato nella gestione dei dati GIS, propongono i loro prodotti a costi estremamente elevati. Questo fattore implica il delinamento di un processo di lavoro *proprietario* che, per aziende piccole o sezioni economicamente svantaggiate (Capitolo 2), risulta essere improponibile.
Questo procedimento di lavoro è stato analizzato prendendo in considerazione le metodologie adottate dal laboratorio *VISIT* del consorzio CINECA, presso il quale questo lavoro di tesi è stato svolto, e dal Centro Nazionale di Ricerca sezione ITABC di Roma, dove è stata analizzata la parte di generazione territoriale sviluppando, in parte, una soluzione alternativa (Capitolo

7).
Questo lavoro di tesi intende proporre una possibile variazione ad un pacchetto software OpenSource, con l'intento di estendere il flusso di lavoro consolidato nell'ambito della ricostruzione territoriale.
Cercando di classificare le metodologie più usate per la gestione di dati GIS (Capitolo 4) e la generazione di territori 3d (Capitolo 3), si è scelto un approccio in grado di entrare nel Work-Flow attuale e di aggiungere diverse potenzialità esplorate, fino ad ora, solo superficialmente.
Dopo una serie di tentativi adibiti alla creazione di un tool in grado svolgere le suddette funzioni, il pacchetto software del progetto Virtual Terrain (Capitolo 6) è stato scelto come ambiente da modificare. Virtual Terrain è stato amalgamato, in maniera più stretta, al MiddleWare OpenSceneGraph (Capitolo 5) in grado di gestire, generare e visualizzare un vasto set di formati 3D.
Utilizzando le principali funzioni di gestione dei dati in OpenSceneGraph ed utilizzando le avanzate potenzialità di manipolazione dinamica di una scena 3D in Enviro (visualizzatore compreso nel pacchetto VTP) è stato sviluppato un metodo di integrazione, all'interno dei terreni caricati, delle strutture di complemento del territorio come: modelli 3D, costruzioni, vegetazione ecc.
Il software modificato è stato testato nell'ambito di due progetti di ricostruzione del Patrimonio Culturale; i progetti ai quali si fa riferimento sono: Appia Antica Project (Sezione 2.4.2) e VEL Project (Sezione 2.4.3).

Capitolo 2

Gestione dei dati territoriali e approccio OpenSource

La storia degli ultimi decenni delle applicazioni nel campo dei Beni Culturali e Paesaggistici, che utilizzano tecniche di computer graphics, ha visto la ricostruzione di modelli di scarso funzionamento e senza possibilità di interazione.
Un ulteriore problema è sempre stato rappresentato dal fatto che queste applicazioni erano fruibili solo attraverso computer molto costosi, dotati di tecnologie avanzate non di uso popolare e non accessibili liberamente da chiunque [21].
Nella maggior parte dei casi i modelli venivano realizzati ed utilizzati da piccole comunità e rimanevano esclusivamanete all'interno di centri di ricerca, università e in laboratori di sviluppo di tecnologie militari.

2.1 La crescita del 3D

Analizzando l'interesse per le applicazioni in ambito 3D, durante gli ultimi 20 anni ci si accorge che vi è stato un incremento sostanziale. Parte del merito di questo incremento è focalizzabile nel campo della Home Entertainment e nella *corsa ai videogiochi*.
Fin dagli anni '80 il successo dei video games, e successivamente dei computer games, si è costantemente allargato alle intere nuove generazioni ed anche al pubblico adulto. In Europa, negli Stati Uniti e in molte parti dell'Asia,

un'altissima percentuale di adolescenti ha provato almeno una volta l'esperienza di un video-gioco o addirittura possiede una piattaforma Playstation, X-Box o Nintendo. Si può quindi identificare la Real-Time Computer Graphic come un'esperienza che entrerà sempre più a far parte, attivamente o passivamente, dell'esperienza quotidiana della popolazione dei paesi cosiddetti *civilizzati*, essendo supportata non solo da un puro e semplice gusto per l'Entertainment ma, principalmente, da un forte interesse economico [33].
Essendo il campo dei videogiochi un ambito in continuo sviluppo (che detiene un gran numero di innovazioni sia in ambito algoritmico che in ambito di hardware) è possibile pensarlo come ambiente di partenza per creare applicazioni anche di ricerca e di studio.
Partendo da questi presupposti, è possibile pensare all'utilizzo di game engine riadattati in modo tale da creare applicativi per gestire simulazioni ed ambienti con un rilevanza fisica e tramite l'utilizzo di dati globalmente riconosciuti dalla comunità scientifica.
Il riadattamento di questi motori dovrebbe essere complementato con la possibilità di sfruttare tutti i nuovi ritrovati in ambito hardware e software (ad esempio con l'aggiunta di animazioni di personaggi, l'aggiunta di comportamenti tramite algoritmi di intelligenza artificiale, l'implementazione di complessi tools che supportano l'interazione multi-utente, ecc.). Inoltre, sull'utilizzo di dati accettati dalla comunità scientifica, è possibile pensare di utilizzare anche modelli ottenuti attraverso operazioni di photomodelling o tecniche avanzate di laser scanning in modo tale da riuscire ad avere delle ricostruzioni il più fedele possibile al mondo reale.
I Sistemi di Realtà Virtuale si pongono come una risposta a molti problemi relativi all'integrazione di diverse tipologie di dati a diverse scale e alla fruizione di essi in modo comunicativo.
Attraverso questa visione si può arrivare a pensare ai Sistemi di Realtà Virtuale come ad un generatore di un nuovo Patrimonio Culturale immateriale che può comprendere in sé una visione approfondita e viva di un Patrimonio Culturale materiale (come ad esempio quello geografico, monumentale ed urbanistico) che va modificandosi nel tempo talvolta fino alla sua totale scomparsa [18].

2.2 GIS e Realtà Viruale

Nell'ambito applicativo di ricostruzione di paesaggi risulta di fondamentale rilevanza l'utilizzo delle tecnologie GIS (Geographic Information System; in italiano Sistema Informativo Territoriale) [31] [33].
Il termine GIS definisce una struttura costituita da un potente insieme di strumenti e tecnologie preposta all'acquisizione, archiviazione, gestione, trasformazione, analisi e visualizzazione di dati spaziali georeferenziati: ovvero documenti o eventi che si riferiscono ad una determinata porzione della superficie terrestre.
I dati spaziali o geografici rappresentano fenomeni del mondo reale e sono caratterizzati: dalla posizione nello spazio rispetto ad un sistema di riferimento e di coordinate, da attributi non spaziali (colore, temperatura, ecc.), dalle reciproche relazioni spaziali (topologiche, direzionali, di distanza).
Utilizzando tecniche GIS ed implementandole all'interno di sistemi di Realtà Virtuale è possibile giungere ad un approccio più diretto con i dati, offrendo all'utente, attraverso l'esperienza dell'immersività, la possibilità di interagire tridimensionalmente, in maniera intuitiva ed immediata con le informazioni all'interno di ambienti virtuali, mantenendo allo stesso tempo la dimensione spaziale di tutte le informazioni oltre alle metainformazioni ad esse collegate. Inoltre, la capacità geo-spaziale dei GIS consente di poter mappare accuratamente la configurazione geografica risolvendo in tal modo una parte essenziale del processo di sviluppo di un ambiente virtuale.
Fino ad oggi applicazioni di ricostruzione territoriale in 3D sono state ampiamente utilizzate da tutti quei settori che hanno un interesse in ambito militare. Infatti, i primi esperimenti di ricostruzione si ritrovano in applicativi di simulazione in ambito bellico.
Ovviamente si trova, come maggior sostenitore di questo tipo di ricerca, il Governo degli Stati Uniti d'America appoggiato da una serie di ditte specializzate nella grafica Real-Time (come esempio la Multigen Paradigm [42]). Negli ultimi anni è stato sviluppato un vero e proprio protocollo per creare e poter interagire con gli ambienti virtuali al fine di sviluppare delle situazioni il più possibile realistiche e plausibili durante le esercitazioni militari.
Purtroppo fino ad oggi i maggiori investimenti economici per sistemi di realtà virtuale sono stati impiegati in progetti con immediata ricaduta economica, politico e militare senza tenere conto degli effettivi benefici che questo genere

di ricostruzioni porterebbero al Patrimonio Culturale.
Andando contro tendenza e sfruttando tutto il sapere accumulato in questi anni nel campo applicativo militare (attraverso i sistemi di realtà virtuale) viene spontanea la creazione di una metodologia per la generazione di paesaggi per la ricostruzione e la valorizzazione del territorio.
Il tentativo è quello di dirigersi verso la creazione di ambienti di lavoro integrati, che rispondano alle necessità della ricerca, della conservazione, dell'archiviazione, ma anche della divulgazione e comunicazione del Patrimonio Culturale.

2.3 OpenSource: un nuova tendenza per la valorizzazione del Patrimonio Culturale

Grazie ad un lavoro continuo e mirato di sensibilizzazione, ma anche alla scarsità di fondi ed ai tagli economici, negli ultimi anni in Italia, si è notata una inversione di tendenza che ha portato sia le aziende che le Pubbliche Amministrazioni a cercare soluzioni alternative in ambito tecnologico.
L'interesse per l'OpenSource sta crescendo, basti pensare ai numerosi studi e monitorizzazioni che sono state promosse dal Ministero per le Innovazioni Tecnologiche; da una recente ricerca risulta chiaro che questi strumenti potrebbero essere la soluzione decisiva per ridurre costi, eliminare duplicazioni di sforzi e velocizzare la diffusione di innovazione [3].
Effettivamente anche la qualità degli strumenti Open sta crescendo, insieme alle comunità che li sviluppano e al materiale informativo disponibile (tutorial, demo, ecc.).
L'Open Movement ha penetrato diversi settori, spesso sconvolgendo, grazie al proprio impatto sociale, le regole tradizionali delle istituzioni e degli enti di ricerca, promuovendo nuove idee e il lavoro condiviso. Così anche settori che fino ad ora erano rimasti esclusi dal poter accedere a tecnologie 3DVR, come quello del Patrimonio Culturale, hanno cominciato a muovere i primi passi.
Questo è reso possibile principalmente dal basso costo che questi sistemi propongono, inoltre, l'utilizzo di un software OpenSource svincola l'utente e lo sviluppatore dalle scelte delle software-house sia dal punto di vista dei formati (ad esempio pensiamo al caso in cui la software house decida di uti-

lizzare un formato proprietario completamente diverso da quello precedente, obbligando così all'acquisto di una nuova release del software) oppure della sopravvivenza stessa di un applicativo; nel caso del fallimento di una software house, infatti, l'utilizzatore rimane senza il prodotto che ha sempre utilizzato con le funzionalità specifiche.
Ultima, ma non di secondaria importanza, è l'usabilità che ha raggiunto dei risultati straordinari tanto da poter essere considerata comprensibile anche da chi non è un professionista informatico. In un software OpenSource è possibile sempre capire la struttura dei dati poichè aperti e modificabili. Così come il modello di sviluppo OpenSource è diverso, anche la filosofia di chi lo utilizza deve essere diversa.
Non si tratta più di trovare il prodotto che assomigli alle esigenze particolari di un utente, finendo poi per adattare esigenze dell'utente e soprattutto i dati al *prodotto trovato*.
Per questi vari motivi si ritiene che OpenSource sia lo strumento ideale per affrontare qualsiasi progetto che necessiti di sperimentazione su qualcosa non disponibile, o che non esista, in commercio; o ancora quando, non disponendo dei finanziamenti adeguati, vi sia la necessità di costruire comunque una dimostrazione significativa per mostrare la validità di un progetto. Si apre così un mare di prospettive e di possibili sviluppi, in questo settore, che deve necessariamente muoversi con cautela nella pochezza economica di cui dispone.

2.4 CINECA e CNR-ITABC: Progetti

In questa sezione verranno contestualizzate le ricerche effettuate all'interno dei progetti di visualizzazione territoriale sviluppati attualmente al *CINECA* (sede nella quale è stato svolto questo progetto di tesi) e al *CNR-ITABC* di Roma.
Il CINECA è un consorzio di 23 università italiane costituito nel 1969 ed è uno dei maggiori centri di calcolo in Europa.
L'obiettivo del CINECA consiste nel supporto alle attività di ricerca della comunità scientifica tramite il supercalcolo e le sue applicazioni. Vi vengono svolti diversi progetti nei più svariati ambiti che vanno dal calcolo parallelo, alla visualizzazione scientifica, anche attraverso applicativi di realtà virtuale.

Grazie allo sforzo del *VISIT lab.* e al CNR-ITABC sono in sviluppo, in questo periodo, diversi progetti per la realizzazione di applicativi di realtà virtuale (o meglio Enhanced Reality) nel contesto dei Beni Culturali e del Decision Making [33].
Sebbene la VR permetta di raggiungere il massimo del realismo se utilizzata in concomitanza di strumenti ad alte prestazione (come ad esempio: i teatri virtuali o i Cave systems), è possibile, grazie alla performance raggiunta dai nuovi PC (dovuto al mercato dell'Home Entertainment), pensare di raggiungere il maggior numero di utenti possibile, tramite Internet, ed un adeguato riadattamento degli applicativi per una fruizione da semplici computer domestici.

2.4.1 Il Progetto Visman e la certosa di Bologna

Il criterio di usare il territorio come elemento di accesso a dati complessi è stato scelto come linea guida nello sviluppo di un insieme di progetti correlati, ideati dal *Progetto Nuove Istituzioni per Comunicare la Città* del Comune di Bologna [17] [18].
Per questo progetto è stato individuata una zona del circondario di Bologna con una rilevanza storica ed artistica, che però, necessita di un intervento di valorizzazione.
È stato focalizzato l'intervento sul complesso della Certosa, il cimitero storico monumentale cittadino che risulta esser uno dei monumenti più importanti della città, oltre ad essere uno dei cimiteri più antichi d'Europa.
Per il progetto è stata realizzata una banca dati digitale con tutte le informazioni sulla Certosa (rilievi digitali, catasto ed anagrafe delle tombe e loro schedatura artistica) con l'obiettivo di comunicare e fare conoscere la Certosa ai bolognesi, alle scuole ed ai turisti.
Si è pensato, per il raggiungimento di questi obiettivi, ad un'applicazione ad alto impatto visuale capace di integrare risorse ed in grado di interagire liberamente con il paesaggio. Sfruttando l'applicativo VISMAN la navigazione risulta essere simile a quella di un videogioco. Per l'audience di tipo accademico è stato implementata nella scena la possibilità di interrogazione ad un DataBase relazionale.
La riflessione alla base dell'applicazione è che, partendo dal territorio, si fa mentalmente ordine su dove siano dislocati i nuclei di interesse culturale e li

Figura 2.1: Il progetto VISMAN: Un immagine della riproduzione del cimitero della Certosa.

si connette in una sorta di rete visuale; una volta entrati nel nodo significativo è quindi possibile ottenere tutte le informazioni salienti collegabili a quel sito. Il database integra insieme ai testi, fonti di tipo multimediale come: testimonianze orali, immagini, filmati ed è stato collegato a una opzione di lettura automatica per favorire una fruizione dei contenuti che non affatichi troppo ed agevoli le persone ad esempio ipovedenti.

Il progetto VISMAN

VISMAN (Visual Scenarios Manager) [18] è un visualizzatore di scenari con particolari funzionalità. Permette di interagire con le visualizzazioni effettuando accessi contestuali ad informazioni digitali relative allo scenario o all'area geografica visualizzata.
La funzionalità principale, che distingue VISMAN da un qualsiasi visualizzatore di scenari applicati in Realtà Virtuale, è la possibilità di riuscire a connettere dati, all'interno di database, con modelli 3D nella scena.
Utilizzando la tecnica di *Picking* sui modelli all'interno della scena è possibile pubblicare in finestre di contenuto le informazioni rappresentanti gli oggetti.

2.4.2 Appia Antica Project

Il progetto dell'*Appia Antica* [22] è un perfetto caso di studio per sperimentare diversi approcci in ambito archeologico e storico.
La via Appia è una delle più importanti vie del mondo Romanico, fu costrutita dal console *Appius Claudius* e connette, tuttora, Roma con Brindisi. Attualmente, nella zona adiacente a Roma, la via Appia e il suo circondario è stato incluso in un area definta Parco Archeologico (dal novembre del 1988). Il parco include molti siti archeologici di rilevante importanza come, ad esempio: la vallata della Caffarella; l'area degli acquedotti Romani ecc.
Dal dicembre 2002 è stato stipulato un accordo di ricerca, tra il CNR-ITABC e la Soprintendenza Archeologica della città di Roma, per lo sviluppo di un progetto di creazione di un archivio spaziale sui monumenti del parco. Alla base di tutto il lavoro vi è un processo di rilevamento spaziale 2D e 3D (grazie a metodologie GIS) con tecniche che utilizzano diversi strumenti per il rileivo (come GPS, stazione totale, scanner laser ecc.) ed una fase di documentazione dei monumenti e di tutte le strutture archeologiche. Il progetto è stato sviluppato utilizzando due approcci differenti. Le finalità dei due metodi di sviluppo risultano comunque le stesse e la base di dati iniziale è la medesima.

Applicazione in Virtools

Tramite l'utilizzo di Virtools[1] è stato ricreato il paesaggio dell'Appia Antica [29] in modo tale da rendere ogni aspetto: scientifico, geografico, antropologico, culturale, narrativo, in maniera semplice ma con valenze storiche ed archeologiche.
L'idea basilare di questo progetto consiste nella creazione di veri e propri centri di documentazione specifici basati su un sitema di realtà virtuale. Le informazioni incluse nel progetto sono adatte sia a studi scientifici ma anche ad una divulgazione culturale per turisti e cittadini.
Il progetto è stato reso più gradevole (ed alla portata di tutti) grazie all'inserimento, nella scena, di personaggi in grado di interagire con lo spettatore. Questi personaggi possono essere dei veri e propri *Avatar*, in grado di guidare lo spettatore dando informazioni sul parco (storiche, mitolgiche o attuali), o

[1] Virtools-Dev è uno strumento per la realtà virtuale che permette un livello di programmazione visuale della scena in maniera molto sofisticata ma elastica.

Figura 2.2: Ninfeo di Egeria; visualizzazione del monumeto all'interno dell'applicativo sviluppato con Virtools-Dev.

modelli animati di supporto alla visualizzazione.
Anche l'aspetto archeologico viene messo in evidenza grazie a modelli di ricostruzione altamente fedeli agli originali e grazie alla possibilità di accedere a contenuti speciali direttamente nella scena.
L'ambito scientifico viene risaltato grazie ad un terzo livello di fruizione dei contenuti in cui le informazioni sono organizzate in diversi temi gerarchici, ad esempio tramite dati topologici, documentazione storica ed artistica, stato di conservazione, dati di restauro, analisi diagnostiche, ipotetiche ricostruzioni dei monumenti ecc.

Applicazione in VTP e OSGActiveX

Uno degli obiettivi che si prefigge questo progetto [30][37] è la creazione di un archivio di dati spaziali (2D e 3D) per la Soprintendenza di Roma sul parco dell'Appia, visibile anche attraverso Internet. Per tale ragione si è dato avvio alla sperimentazione di strumenti, con un approccio completamente diverso e aperto.
Questo progetto è sviluppato con tool OpenSource, dalla modifica dei dati fino alla visualizzazione e fruizione finale.
Il progetto prevede la suddivisione dell'obiettivo finale in due goal intermedi:

Figura 2.3: Sito web Appia Antica Project[37], sezione 3D.

- La creazione di un sistema, di tipo interattivo, in grado di ricostruire il territorio, modificarlo in maniera dinamica e accessibile via web.

- Abilitare un accesso ai dati via web, ristretto al team di lavoro, in modo da rendere possibile la cooperazione di più persone anche in postazioni di lavori distanti.

Attualmente, tramite il pacchetto VTP modificato (capitolo 6) e l'utilizzo del Plug-In OSG Active-X (paragrafo 6.3.5) è stata resa possibile la fruizione ai dati da remoto e la modifica, ancora in locale, dei dati 3D dinamicamente. La navigazione dei dati via web in Real-Time è resa migliore grazie alla possibilità di interagire aggiungedo: punti di vista; attivando e disattivando layer vettoriali che possono essere aggiunti al terreno per avere una comprensione migliore del territorio; abilitare lo switching di modelli sul terreno o di differenti tipologie di terreni.

2.4.3 Virtual Etruscan Landscape Project

Il progetto *VEL* (Virtual Etruscan Landscape) [30] ha come scopo quello di ricostruire l'aspetto della città di Bologna durante l'età del bronzo. Il progetto viene collocato in un contesto di tipo didattico attraverso un processo interdisciplinare in quanto raccoglie le esperienze di un gruppo di archeologi, storici ed esperti informatici.

Per la ricostruzine del paesaggio sono state utilizzate fonti di dati differenti

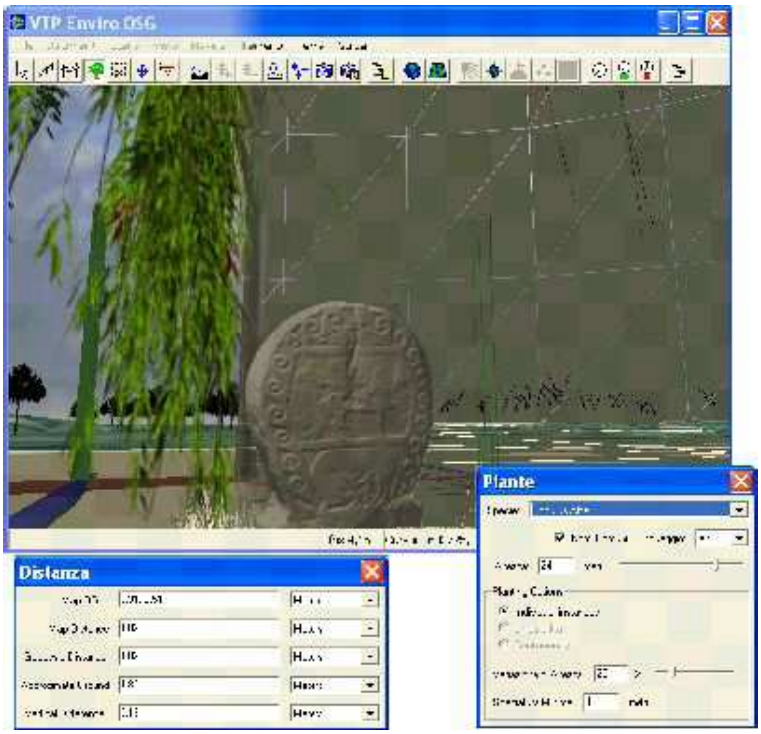


Figura 2.4: Enviro modificato durante la modifica di alcuni parametri di un modello.

che si integrano grazie all'utilizzo della Realtà Virtuale.

Utilizzando dati di tipo GIS odierni e immagini satellitari modificate è stato possibile ricreare un terreno avente tutte le informazioni di georeferenzizione necessarie per ricostruire l'esatta posizione di ritrovamento di diversi reperti archeologici. Utilizzando tecniche di fotomodellazione e acquisizione laser, reperti presenti nel museo Civico Archeologico di Bologna, sono tornati in vita e sono stati ricollocati nell'esatta posizione di ritrovamento. Inoltre, grazie a fonti storiche è stato possibile aggiungere elementi al paesaggio (come alberi, animali ed elementi di suddivisione di confini) per ricreare delle

possibili idee di ricostruzione del paesaggio.
L'utilizzo di Enviro modificato (capitolo 6) ha reso più facile la collaborazione dei gruppi interdisciplinari, in quanto la modifica del paesaggio è stata fatta dinamicamente consentendo il confronto di tutti gli esperti. Inoltre, la possibilità nativa di Enviro di accettare librerie di modelli e di vegetazione esterni lo ha reso un tool versatile adatto a questo tipo di esperienza di collaborazione.

Capitolo 3

Metodi di Rendering dei Terreni

Durante la ricostruzione di grandi territori applicati in ambienti real-time bisogna far fronte al problema di gestire una quantità di dati enorme.
Cercando delle soluzioni si nota, nell'immediato, che il problema fondamentale può essere ricondotto nel capire cosa bisogna visualizzare in una scena e, soprattutto, quanto raffinata dovrà essere la visualizzazione in base a parametri come il grado di precisione dei dati iniziali e il dettaglio di visione in base alla distanza dell'osservatore.
Il dibattito su come poter raffinare la renderizzazione è tuttora aperto e molti tipi di approcci e soluzioni sono stati già affrontati. In questo capitolo sarà descritta una rassegna di metodi per la generazione e il rendering di terreni.

3.1 Rappresentazione dei Dati

Il primo problema che ci si pone di fronte durante la renderizzazione di un terreno è il reperimento dei dati e il tipo di dati che devono essere processati. Se il terreno da generare è un terreno che dovrà essere posto in un contesto di fantasia o di gioco la veridicità dei dati di partenza non è un fattore essenziale. Il terreno potrà essere generato anche con variazioni ai dati di partenza e potrà essere modificato e adattato solo ad una buona resa finale. Se il problema di fondo è quello di mantenere un contesto di simulazione fisica allora i dati di partenza dovranno rispettare dei canoni ben precisi e

variazione ad essi dovranno rispettare regole e comportamenti accettati in ambito metematico e geospaziale.
Normalmente, i dati fisici che rappresentano un terreno sono disponibili soprattutto come curve di livello. Durante l'ultimo secolo i geodeti hanno raccolto mappe di curve di livello per riuscire a descrivere tutta la superficie terrestre.
Attualmente, la ricostruzione spaziale della terra è resa più semplice grazie a tutta una serie di applicativi, di strumenti e di metodologie che permettono di ricostruire il territorio con un grado di precisione molto elevato ad un dettaglio ottimo.
I mezzi per la ricostruzione terrestre usati attualmente sono: Remote Sensing, foto aeree, scanner satellitari. Nel contesto informatico i dati territoriali devono venire espressi con una matrice di altezze detta: *HeightField*; essa può essere espressa tramite una matrice di toni di grigio (vedi figura 3.1 centrale) o grazie ad una immagine a colori dove, la luminosità o il colore di ogni pixel, viene correlato con la sua elevazione.
A causa dell'enorme quantità di dati e, quindi, delle corrispondenti mesh di triangoli, gli HeightField non possono essere renderizzati esattamente. Il lavoro che viene fatto è quello di cercare una semplificazione della mesh che cerchi di adagiarsi meglio sui dati di partenza.

3.2 Renderizzazione di tipo statico

In questa linea di pensiero vengono compresi tutti quei metodi che generano, partendo sempre da dati di elevazione e textures, delle geometrie statiche a diverse risoluzioni.
Le geometrie create saranno suddivise in modo tale da possedere una gerarchia di livelli di dettaglio commutabili in base alla posizione dell'osservatore e, se la tecnica adottata rispecchia i canoni della paginazione su disco, la gerarchia sarà salvata su file e caricata solo all'occorrenza.

3.2.1 TINs

TINs (Triagular Irregular Networks [27]) è stato il primo algoritmo ad implementare un metodo per la semplificazione di terreni. L'HeightField viene convertito in una mesh di triangoli irregolari. Le regioni più uniformi verranno convertite con minore presenza di triangoli mentre sono presenti più triangoli nelle zone con curvature superficiali maggiori.
Esistono una varietà di algoritmi che cercano di diminuire la dimensione delle Mesh irregolari cercando di preservare la struttura dei dati originali (anche con tecniche *Wavelet Based Encoding*). Il principio di base, comunque, è lo stesso per tutti i tipi di algoritmi.

L'approccio TINs presenta una serie di svantaggi che ne riducono le pre-

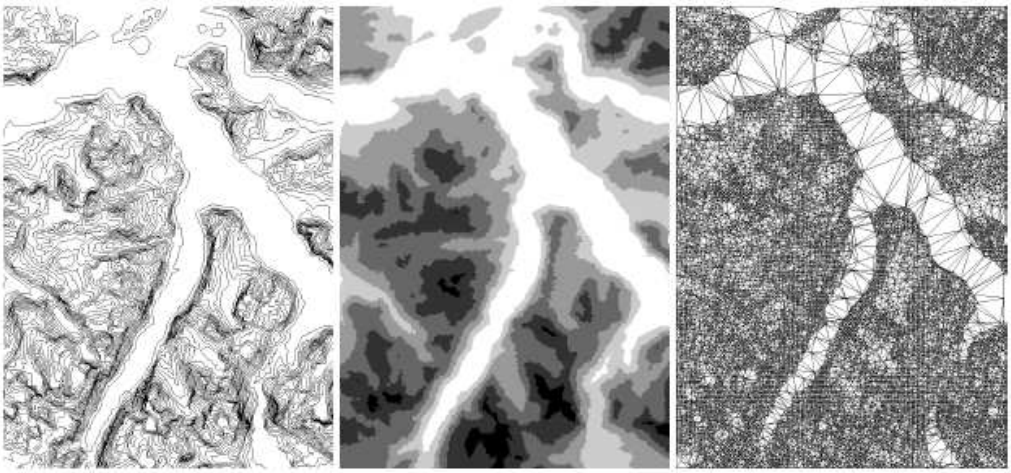


Figura 3.1: Esempio di visualizzazione di *TIN*. Da sinistra a destra: Curve di livello della superficie; DEM a toni di grigio; TIN renderizzato in Wireframe.

stazioni e l'usabilità: dipendentemente dal livello di compressione utilizzata alcuni piccoli dettagli possono essere persi durante la fase di semplificazione dei dati iniziali. Se il punto di vista è vicino a questi dettagli omessi ci si può accorgere dell'errore. Un altro svantaggio di questo approccio risulta essere la scarsa versatilità nella suddivisione dei dettagli. Un terreno ad alta risoluzione rimarrà tale anche se il punto di vista è distante con successivo spreco di risorse della macchina di render.

3.2.2 Static LOD

Il problema fondamentale nella renderizzazione dei territori risulta essere il compromesso tra risoluzione e pesantezza dei modelli generati.
Una delle prime soluzioni proposte è stata quella dei Livelli di dettaglio statici; gli S-LOD [6] [26]. In questa soluzione il terreno viene suddiviso in porzioni (in gergo tecnico *tiles*) che contengono TINs a varie risoluzioni di dettaglio. In base alla distanza dell'osservatore dal modello verranno caricate delle tiles con una suddivisione delle mesh appropriate. Se nella tecnica di generazione dei TIN, vengono utilizzate delle mesh regolari grezze il metodo viene definito *geo-mipmapping* [36].
La risoluzione appropriata per ogni tiles viene scelta in modo tale che lo spazio di errore della proiezione a schermo, di ogni singolo tile, risulti al di sotto di una predefinita soglia di errore di uno o più pixels. Se la soglia rimane sotto un pixel la semplificazione del terreno risulta indistinguibile dalla mesh originale. In pratica la soglia viene presa in modo tale da non appesantire troppo il motore di rendering cercando di mantenere un frame rate ottimo.
Un problema basilare che nasce con questo tipo di metodo lo si riscontra con la comparsa dell'effetto di *Popping*[1] durante lo scambio di tile a differenti livelli di dettaglio. Utilizzando questa tecnica si va a definire tile con differenti livelli di dettaglio e lo switching non continuo tra di esse fa sì che si evidenzi l'effetto di transizione.

Un altro problema che sorge con questo tipo di tecnica si ha nella congiunzione tra le porzioni di terreno. Esistono tuttora degli artifizi per minimizzare e rendere non percepibile questo effetto visivo. L'idea di base che le accomuna consiste nell'aggiunta di ulteriori poligoni. Di seguito vengono presentate in breve due tecniche utilizzate in ambito di generazione di terreni di tipo Static LOD da TerraVista [48] e osgDem di OpenSceneGraph (paragrafo 5.2):

- *Smart Mesh:* L'idea *Smart Mesh* considera ogni tile come composta di cinque componenti poligonali distinte ciascuna dotata dei propri livelli di dettaglio: un parte centrale costituita da tutti i poligoni che non hanno vertici appartenenti al bordo, e quattro parti che costituiscono i bordi. In tal modo qualora ci si trovi nella situazione di avere due tile

[1] fenomeno per cui un elemento visivo della scena (può essere un vertice, un poligono o un modello) compare dal nulla quando entra nel campo visivo oppure diventa visibile quando la condizione di switch tra livelli di dettaglio differenti viene soddisfatta.

adiacenti con livelli di dettaglio differenti i bordi potranno continuare in ogni caso ad avere lo stesso LOD garantendo la necessaria continuità;

- *Skirt:* Come da traduzione *Skirt* significa *gonna*. Ad ogni bordo della tile vengono aggiunte delle *strip* di triangoli che si protendono verso il basso. Questi poligoni mantengono le coordinate texture della tile originaria in modo tale da prendere i colori di bordo. Questa tecnica elimina il problema di visione delle giunture tra tile se l'angolo di visione è differente dallo *Zenith* del terreno.

3.2.3 Mesh progressive

Questo tipo di tecnica è un caso particolare di S-LOD proposto da Hoppe [11] per la semplificazione di modelli; ovviamente questo approccio è riutilizzabile anche per la generazione di terreni.
Ogni tile del terreno viene rappresentata da un set di mesh che derivano dalla

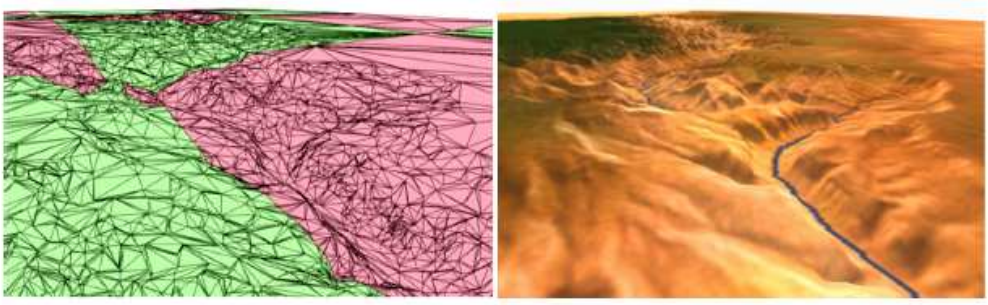


Figura 3.2: Renderizzazione terreno con tecnica *Mesh Progressive* proposta da Hoppe. (immagine tratta da [12]).

mesh originale con la specifica che le operazioni di taglio e di ricongiunzione siano definite dalle regole di Progressive-Mesh [9][10]. La connettività tra tiles viene garantita da una griglia totale che rimpiazza le congiunzioni con un successivo aggiustamento dato dall'alterazione del colore originale.
Il vantaggio di questo approccio risulta essere la buona approssimazione delle mesh ma tutto questo va a discapito delle prestazioni in quanto l'algoritmo ha bisogno di una richiesta di memoria elevata.

3.3 Renderizzazione di tipo dinamico

Utilizzano degli algoritmi che ricostruiscono il territorio in maniera dinamica a partire da texture e dati di elevazione generati ad hoc (HeightFields, TIN, ecc.). La tecnica di renderizzazione utilizzata è conosciuta con il nome di Continous Level Of Detail.
La differenza fondamentale tra questa metodologia e i metodi di rendering statico risiede nel fatto che l'approssimazione viene pensata direttamente sui triangoli e non più sulla suddivisione in tiles.
Punto a sfavore di questa tecnica è il fatto di dover richiedere un pre-trattamento dei dati ed un successivo costo computazionale elevato.

3.3.1 Algoritmo di Lindstrom

Questo algoritmo risulta essere il primo metodo pubblicato che garantisce un consistente frame-rate durante una visualizzazione interattiva con una qualità di immagine abbastanza alta [26]. Il core di questo algoritmo si basa sulla semplificazione della mesh originale in due fasi. Inizialmente viene applicata una suddivisione in blocchi di visualizzazione e successivamente vengono semplificati direttamente i vertici all'interno dei blocchi.
Il primo step ha il compito di generare dei livelli di dettaglio virtuali durante un processo di precalcolo. L'algoritmo cerca di minimizzare il numero di poligoni da renderizzare e cerca di rendere continuo il passaggio tra le aree di superficie a differenti livelli di risoluzione.
Il secondo step si fa carico di capire quando due triangoli devono essere fusi

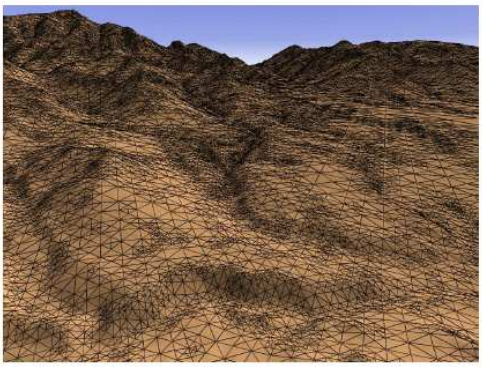


Figura 3.3: Algoritmo proposto da *Lindstrom*. Terreno generato con 40.000 poligoni (Immagine tratta da [25]).

in uno singolo. La condizione di base risulta essere il grado di modifica della differenza tra le inclinazioni dei due triangoli. Le massime differenze verticali tra le due configurazioni, indotte dall'omissione di un vertice, viene riferita come un valore δ di ogni vertice. Se il valore δ cresce la chance di fusione diminuisce.
Proietando il segmento δ sul piano di proiezione a schermo si può facilmente determinare il massimo errore geometrico percepito tra la suddivisione di un triangolo e i corrispondenti sotto triangoli (figura 3.4).
Questo approccio viene applicato ricorsivamente fino al raggiungimento della semplificazione massima consentita dalla mesh di partenza.
Il vantaggio principale, dall'applicazione di questo algoritmo, risulta essere

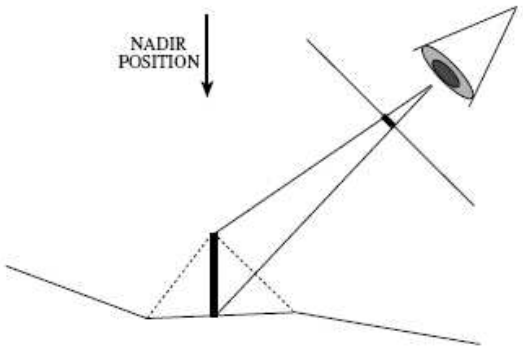


Figura 3.4: Proiezione del segmento δ sul piano di vista.

quello di mantenere una qualità di triangolazione desiderata in base alla soglia di errore dello spazio schermo; purtroppo questo algoritmo richiede un grosso sforzo computazionale specialmente durante lo switch tra blocchi che può consistere in un decremento del frame-rate significativo. Un ulteriore fattore a sfavore lo si trova nella difficoltà di applicare tecniche di ottimizzazione in quanto la suddivisione dei triangoli risulta essere differente per ogni frame.

3.3.2 Algoritmo di Duchaineau

Il metodo di *Ducheineau*, definito Real-time Optimally Adaptive Mesh [20], è un successore della tecnica applicata da *Lindstrom*. La peculiarità risiede nel fatto di preservare una metrica di errore ottimizzata e dei limiti di errore garantiti che permettono la manipolazione dei triangoli direttamente. Inoltre utilizza la metodologia di coerenza frame-to-frame [20] per mantenere un

frame-rate alto.

Vengono implementate due code di priorità per definire le operazioni di

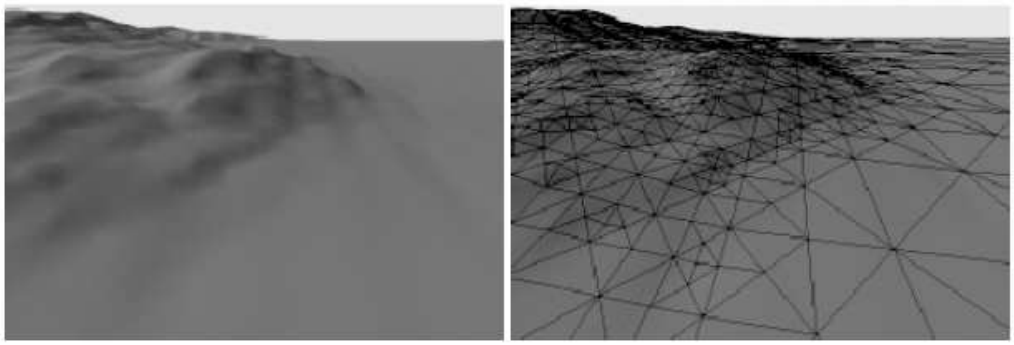


Figura 3.5: Metodo *Duchaineau* (Immagine tratta da [20]).

Split e di Merge su un albero binario di tringoli preprocessato. Il tempo di esecuzione risulta essere direttamente proporzionale al numero di triangoli che variano per frame.
L'albero binario viene definito dividendo i triangoli del nodo radice dal vertice apice fino al punto medio del bordo opposto. In questo modo vengono definiti i figli dei nodi durante la fase di divisione (figura 3.6).

L'idea che guida la suddivisione e la successiva congiunzione è semplice;

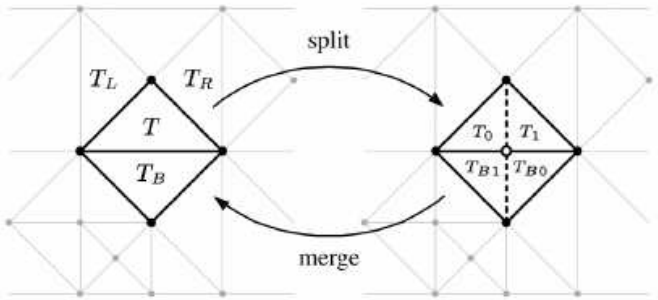


Figura 3.6: Split e Merge di triangoli durante la fase di triangolazione (Immagine tratta da [20]).

inizialmente viene calcolata la priorità per ogni triangolo nella fase di triangolazione e in maniera ripetuta vengono forzate le divisioni (o il merge) dei triangoli con priorità più elevate nelle due code.
La metrica base di errore dell'algoritmo ROAM è definita dalla distanza tra tutti i punti di superficie proiettati nello spazio schermo e dove effettivamente

la fase di triangolazione posiziona i punti.
L'approccio Duchaineau risulta molto simile al Lindstrom aggiungendogli alcune caratteristiche:

- *Riduzione dei dettagli delle facce nascoste:* Può essere specificata una priorità molto bassa ai triangoli visualizzati in *back-facing*;
- *Normal Distortion* Viene definita una priorità elevata ai vertici con una distorsione normale alta riducendo così il rischio di high-lighting speculare;
- *Distorsione delle coordinate texture* anche in questo caso si correla la priorità con poligoni che potrebbero presentare problemi di distorsione sulle texture;
- *View Frustum* Priorità quasi nulla per i poligoni all'esterno del cono di visione.

Purtroppo il tallone d'Achille di questo algorimo risulta essere, anche, il collo di bottiglia sulle prestazioni. Il ricalcolo delle priorità è la fase più onerosa di tutto il procedimento.

3.3.3 Algoritmo di Röettger

La peculiarità dell'algoritmo di *Röettger* [34] si delinea sull'efficace utilizzo della memoria e sull'implementazione della tecnica di *geo-morphing* durante le transizioni di raffinamento della scena.
Il core dell'algoritmo si basa sulla rappresentazione dell'HeightField in una struttura di tipo Quad-Tree salvata in una matrice compatta.
Ogni nodo del Quad-Tree corrisponde ad un massimo di otto triangoli organizzati in una *triangle fan* attorno al punto mediano del nodo (figura 3.7). La mesh risultante è ottenuta semplicemente omettendo i vertici della *triangle fan*.

La triangolazione per uno specifico punto di vista viene decisa in base ad un criterio, per ogni Quad-Tree, interattivamente dalla radice fino alle foglie dell'albero. Il criterio di valutazione per il raffinamento della mesh è basato sulla tecnica *View-Dependent Projection*, esposta nell'algoritmo di Lindstrom (sezione 3.3.1), con la variante del non farsi carico dell'angolo di visione ma

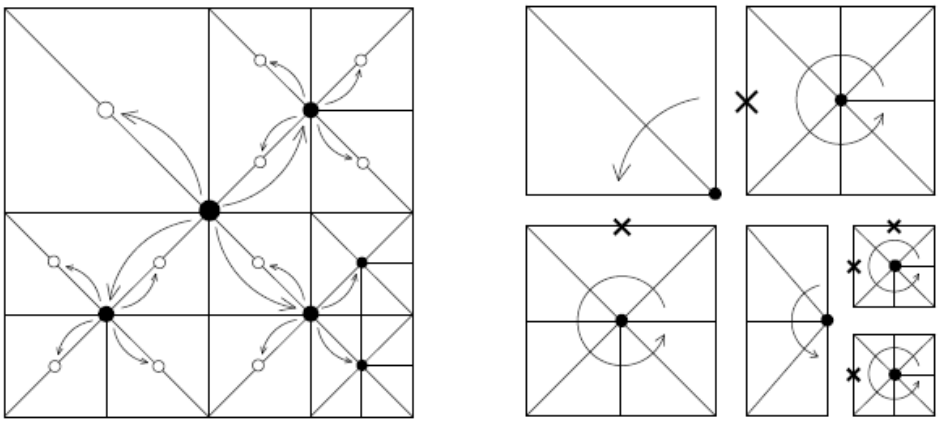


Figura 3.7: Triangolazione di un HeightField di 9x9. *Sinistra:* suddivisione Quad-Tree dell'Heightfield. *Destra:* suddivizione in triangle fan del Quad-Tree.

solo della posizione.
L'algoritmo di Röettger elimina il problema del popping dei vertici aggiungendo alla tecnica di raffinamento una tecnica base di *geo-morphing*. Quando un Quad-Tree viene marcato per il raffinamento la transizione tra il dettaglio attuale e i figli avviene tramite una interpolazione graduale.

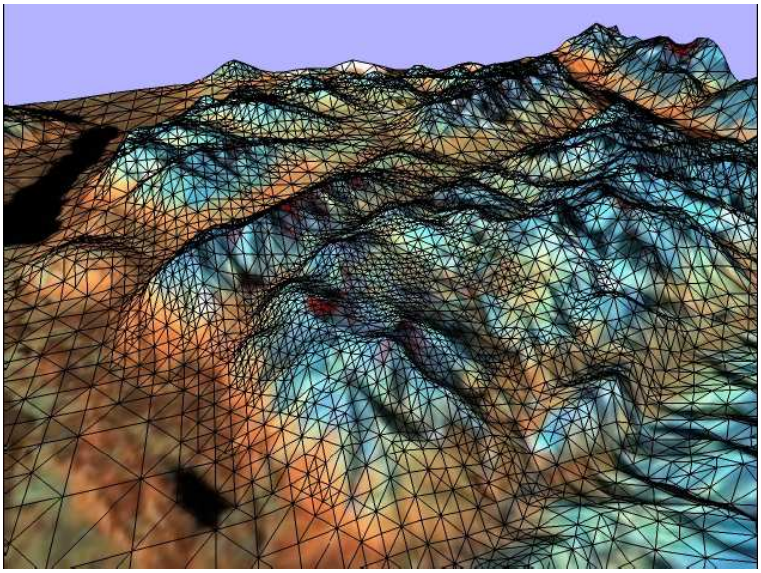


Figura 3.8: Metodo di *Röettger* (Immagine tratta da [34]).

3.4 Metodi a tecnologia mista

Esistono degli algoritmi, ancora in fase sperimentale, che cercano di uniformare le due filosofie viste in precedenza. Questo è l'esempio dell'algoritmo di Chunked LOD.
Come nella generazione di un terreno di tipo statico, a partire da file altimetrici, viene generata una suddivisione in gerarchie di tiles. A differenza del metodo statico puro ogni tile sarà composta da un heightfiled che verrà renderizzato in tempo reale.
Il problema che sta alla base di questi tipi di algoritmi è il fatto che richiedono uno sforzo computazionale ulteriore, durante la fase di rendering e visualizzazione, per il pre-trattamento degli heighfield caricati all'occorrenza.

3.4.1 Algoritmo Chunked LOD

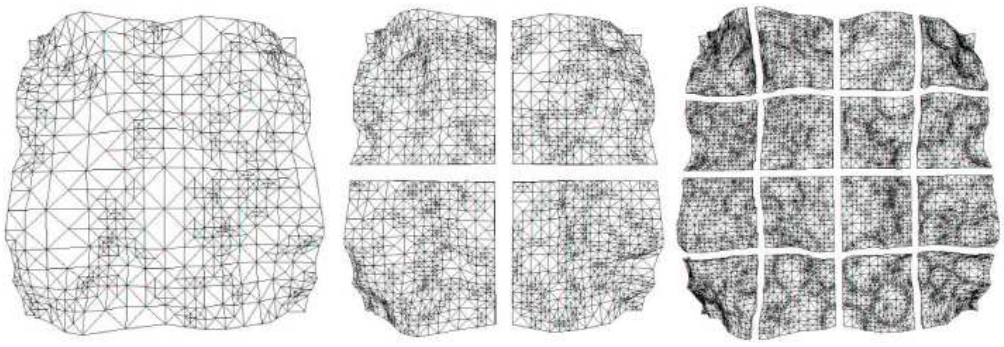


Figura 3.9: I primi tre livelli di un albero *Chuncked LOD* (Immagine tratta da [35]).

Questo algoritmo [35] è una ottima congiunzione delle metodologie di rendering viste fino ad ora. Lo scopo di questo algoritmo risulta essere quello di definire il maggior dettaglio possibile durante il rendering di porzioni enormi di terreno. Grandi terreni, a causa delle loro dimensioni, non possono essere mantenute all'interno di memorie primarie quindi risulta essenziale il caricamento e lo switching da memorie di massa secondarie o addirittura dalla rete.
Questo algoritmo si prefigge di gestire il dettaglio in base alla posizione dell'osservatore (*view-dependent*).
I principali vantaggi che si ottengono da tale algoritmo sono:

- Efficiente gestione dei triangoli all'interno degli insiemi di primitive;
- Integrazione del LOD delle Texture con quello delle geometrie;
- Efficiente *morphing* dei vertici per evitarne il *popping*;
- Leggero carico per la CPU.

Questo a patto di accettare alcune ragionevoli limitazioni:

- Staticità dell'insieme dei dati;
- Utilizzo di maggior numero di triangoli rispetto ad un algoritmo dinamico puro;
- Maggiori dimensioni dei dati dovuti alla fase di pre-elaborazione;

La fase di organizzazione è demandata tramite una fase di pre-elaborazione che genera un albero di tipo Quad-Tree di mesh primitive indipendenti l'una dalle altre (dette Chunck) a partire dall'Heightfield di base. Il metodo di generazione del Quad-Tree è simile a quello di generazione degli Static LOD ma a differenza di questi ultimi nelle tiles non saranno contenuti dei poligoli ma bensì delle mesh.
La generazione dei livelli di mesh avviene tramite la definizione di uno scostamento massimo che implica un effetto di switching tra livelli gradevole. L'effetto di continuità anche tra i livelli viene garantito dalla tecnica di geomorphing applicata anche durante il cambio di tile.
Un ulteriore problema che si può riscontrare in questo tipo di approccio risulta essere lo scostamento tra tile differenti. Ovviamente è possibile applicare degli accorgimenti (come quelli definiti nella sezione 3.2.2) che in genere incrementano il modello di poligoni supplementari.
Infine la struttura a Quad-Tree con nodi fortemente indipendenti e il fatto che mesh e texture LOD siano integrati insieme rendono agevole la gestione del Paging ossia del caricamento diretto da disco dei soli dati necessari.

Capitolo 4

Informazioni geospaziali e librerie

I dati geospaziali cercano di delineare fenomeni del mondo reale, rappresentabili in cartografia, e sono caratterizzati da:

- Posizione nello spazio rispetto ad un sistema di riferimento e di coordinate;
- Attributi non spaziali (colore, temperatura, ecc.);
- Reciproche relazioni spaziali (topologiche, direzionali, di distanza)

4.1 Il problema Cartografico

Il problema cartografico [19][2][28] si pone quando si debba gestire un set di dati geospaziali. In parole povere consiste nel dare una rappresentazione, in un unico sistema di riferimento, di tutta la superficie terrestre.
Un *Sistema di Riferimento* è caratterizzato mediante un sistema di coordinate che può essere fissato grazie all'utilizzo di punti fisici, misure o scelte compatibili di una parte delle coordinate dei punti.
Per riuscire a definire il set di variabili, che definiscono il sistema, bisogna in primo luogo poter dare la posizione di un punto sulla terra rifacendosi ad un modello che, se presenta come caratterisitica quella di essere descrivibile matematicamente e individuabile fisicamente, viene detto *superficie di riferimento*.

Esistono due categorie di superfici di riferimento a seconda che si tratti di una rappresentazione altimentrica o planimetrica:

- *Rappresentazione Planimetrica:* Per la rappresentazione planimetrica, la soluzione è stata quella di mettere in relazione i punti della superficie fisica della Terra con i punti di un sistema cartesiano piano; il passaggio tra superficie fisica della Terra e proiezione cartografica non è diretto, ma, calcolato sulla base di una superficie matematica intermedia detta *ellissoide*;

- *Rappresentazione Altimetrica:* L'impostazione della rappresentazione altimetrica si basa sull'attribuzione di definire ad ogni generico punto P della superficie fisica della Terra una quota. La quota è la sua distanza, lungo la verticale v passante per esso, da una superficie di riferimento detta *geoide*.

Il motivo per il quale si ricerca un modello di riferimento consiste nel fatto che la superficie esterna della Terra è di forma irregolare, non rappresentabile da una superficie matematica; inoltre la densità della massa all'interno di tale superficie non è omogenea.
La tipologia della superficie viene definita in base al tipo di ambito operativo nel quale si deve lavorare. Un riferimento ellissoidico, come menzionato già dalla parola che specifica il sistema, viene impiegato in tutte quelle applicazioni dove è utile solo la conoscenza delle coordinate senza tenere conto del significato dal punto di vista fisico.
Il riferimento geoidico, invece, è essenziale in quanto riesce ad adattarsi meglio alle specifiche fisiche di un rilevamento ma, purtroppo, possiede una descrizione analitica difficile e quindi diventa complicato il trattamento degli angoli e delle distanze.
Detto questo è possibile pensare di suddividere le differenti tipologie di sistemi anche in base al tipo di ambito operativo e alla dimensione di scala sulla quale il sistema dovrà adagiarsi. Per tanto possiamo distinguere sistemi:

- Microlocali (es: controllo di cave dismesse)

- Locali (es: interventi di bonifica)

- Regionali (es: controllo di valanghe)
- Nazionali (es: predizione di eventi calamitosi)
- Continentali/Globali

4.1.1 Il Sistema Planimetrico

I sistemi planimetrici [2] sono nati con l'avvento della cartografia. L'antica scienza della cartografia si prefissava il compito ricercare dei metodi per trasformare (*proiettare*) matematicamente la superficie della terra, o parti di essa, in una superficie piana bidimensionale. Solitamente, le proiezioni, vengono distinte in base alle proprie caratteristiche e al loro metodo di costruzione.
Osservando attentamente il processo di conversione su carta di una proiezione si nota che vi è un inevitabile introduzione di distorsione sui dati e/o sulla geometria. La scelta di uno specifico metodo di proiezione, durante una visualizzazione, è molto importante per la giusta divulgazione dell'informazione stessa ed è estremamente importante sceglierlo in base agli obiettivi che la visualizzazione richiede.
Considerando tali distorsioni si possono stabilire quindi dei criteri di scelta che, nella pratica, sono spesso empirici.

4.1.2 Proiezioni Cartografiche

Le proiezioni sono raggruppate in tipologie differenti a seconda del tipo di proprietà che si vuole fare risaltare. Geometricamente vengono divise in prospettiche, cilindriche, coniche e singolari.

- *Proiezioni Prospettiche:* ottenute mediante un piano tangente alla sfera in un punto qualunque della stessa e al variare del punto di osservazione. Vengono distinte ulteriormente in:
  - *centrografiche:* il punto di osservazione coincide con il centro della sfera e si proietta sul piano della carta gnomonica polare;
  - *stereografiche:* il punto di osservazione è sulla superficie della sfera opposta al piano di proiezione;

  - *ortografiche:* il punto di osservazione è all'infinito e si proietta sul piano tangente al polo opposto.

  Quest'ultimo tipo di proiezione è di fatto il caso limite di una proiezione conica che diventa un piano, infatti presenta paralleli circolari e meridiani rettilinei (sempre nel caso polare). La differenza primaria, con le proiezioni coniche, sta nel fatto che i paralleli sono dei cerchi di 360 gradi mentre nelle proiezioni prospettiche sono degli archi minori di 360 gradi. Questo tipo di proiezione viene usata, di solito, in tutti quei casi di difficile comprensione o nei casi in cui non si ha un'idea specifica sul tipo di proiezione da applicare.

- *Proiezioni Cilindriche:* con questo tipo di proiezioni la terra viene mappata in un cilindro che viene poi srotolato in un piano. La famiglia di proiezioni di Mercatore è la più conosciuta di queste. La caratteristiche principali di una proiezione cilindrica sono:

  - I meridiani e i paralleli sono delle rette con distanze costanti;
  - o Lo spazio tra i paralleli è il fattore fondamentale che distingue una famiglia di proiezione cilindrica da un'altra.

- *Proiezioni Coniche:* sono basate su una proiezione della terra in un cono che, come le cilindriche, viene srotolato. Sono delle proiezioni rare usate solo in casi particolari dove il range di mappatura copre una latitudine molto piccola. Al contrario, per alcuni range particolari, produce una mappa visiva molto *naturale*. La caratteristica principale delle proiezioni coniche risulta essere il fatto che, sui poli, i meridiani, sono rette e tutti i paralleli sono archi concentrici di un cerchio. Come nella famiglia delle proiezioni cilindriche la distanza dei paralleli è l'unica distinzione tra una proiezione conica e le altre.

- *Proiezioni Singolari:* questo tipo di proiezione include tutte le proiezioni che possiedono delle caratteristiche particolari e quindi non rientrano nelle categorie precedenti. Alcune di queste proiezioni sono Bonne, Sanson-Flamsteed sinusoidale e Mollweide. Un caso particolare di proiezione atipica è la Werner Cardiform che è un caso particolare della Bonne ottenuta forzando la latitudine standard a 90 gradi.

Le proiezioni cartografiche, matematicamente [5], vengono suddivise rispetto alle proprietà che mantengono:

- *Equidistanti:* mantengono inalterate le distanze;
- *Equivalenti:* mantengono inalterate le aree;
- *Conformi:* mantengono inalterati gli angoli.

Infine le proiezioni sono suddivise in base al metodo empirico o matematico con le quali sono state generate:

- *Vere:* la generazione è basata su principi geometrici e matematici;
- *Convenzionali:* derivate dalle vere ma incrementate di espedienti che riescono a minimizzarne le deformazioni o generino risultati prefissati.

La scelta di una proiezione dipende dagli obiettivi speciali della rappresentazione sullo scopo in quanto ogni proiezione presenta alcune caratteristiche fondamentali che la distinguono. Come regola generale si può pensare che le proiezioni cilindriche siano efficaci per mappare zone comprese tra i tropici, le proiezioni prospettiche per latitudini alte, mentre le coniche per latitudini medie.

4.1.3 Breve rassegna sulle principali proiezioni

Di seguito un breve rassegna sulle principali proiezioni utilizzate e le relative proprietà principali:

Proiezione stereografica polare

La proiezione stereografica viene definita partendo da un punto sul elissoide di riferimento, detto centro di proiezione. Dal centro di proiezione si proietteranno tutti gli altri punti su di un piano tangente al punto diametralmente opposto al centro di proiezione (riferirsi a fig. 4.1 sinistra).

$$OP' = 2R \tan\left(\frac{\pi}{4} - \frac{\varphi}{4}\right)$$

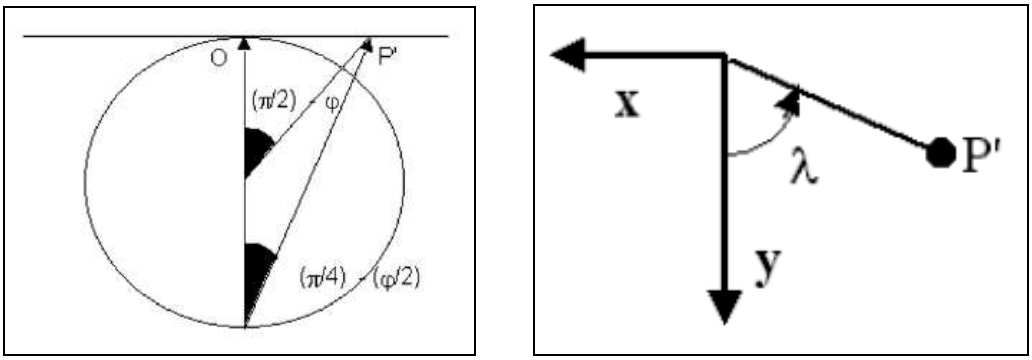


Figura 4.1: *Destra:* Proiezione dei punti. *Sinistra:* Rappresentazione di proiezione.

Assumendo l'asse y della rappresentazione nella direzione in cui si proietta il meridiano fondamentale e l'asse x lungo il parallelo ortogonale, $P = (x, y)$ (fig. 4.1 destra):

$$x = -2R \tan\left(\frac{\pi}{4} - \frac{\varphi}{4}\right) \sin\lambda \tag{4.1.1}$$

$$y = 2R \tan\left(\frac{\pi}{4} - \frac{\varphi}{4}\right) \cos\lambda \tag{4.1.2}$$

Dividendo 4.1.2 con 4.1.1 si ottiene:

$$y = -x \cot\lambda \tag{4.1.3}$$

- Per $\lambda = cost$ si ottiene una retta. I meridiani sono rappresentati da rette uscenti dall'origine;
- Per $\varphi = cost$ si ottiene un cerchio. I paralleli sono circonferenze concentriche;
- La carta risulta conforme, quindi, il modulo di deformazione lineare[1] risulta:

$$M_{Lineare} = \frac{1}{\cos^2\left(\frac{\pi}{4} - \frac{\varphi}{4}\right)} \tag{4.1.4}$$

- L'*ortodromia*[2] che collega due punti sulla terra si può considerare rettilinea.

[1] definizione: $M_{Lineare} = \frac{ds_{Carta}}{ds_{Ellissoide}}$ con ds_{Carta} è la lunghezza dello stesso elemento infinitesino sul piano della carta e $ds_{Ellissoide}$ è la lunghezza di un elemento infinitesino sulla superficie di riferimento

[2] Detta anche *geodetica*. Questa proprietà mantiene geodetiche dell'ellissoide in geodetiche. L'equatore e i meridiani sono esempi di geodetiche mentre i paralleli non lo sono.

Proiezione Mercatore

La proiezione di Mercatore è un caso particolare delle proiezioni di tipo cilindrico (viene imposta la condizione di conformità). Permette di rappresentare i paralleli ed i meridiani mediante un reticolato cartesiano di rette tra di loro ortogonali.

Fondamentalmente tutte le rappresentazioni di questo tipo si basano sul-

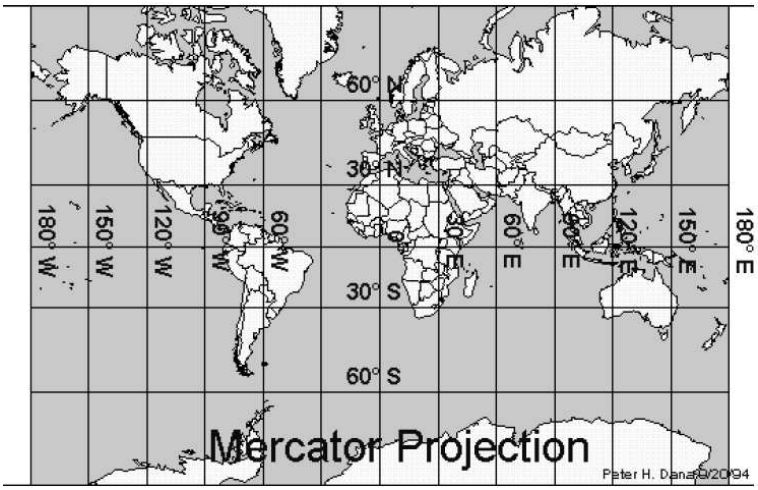


Figura 4.2: Rappresentazione canonica di Mercatore.

l'osservazione che si può proiettare la superficie terrestre dal suo centro su un cilindro ad essa circoscritto, dopodiché basta tagliare il cilindro lungo una sua generatrice e considerarne lo sviluppo su di un piano.

$$\begin{aligned} x &= a\lambda \\ y &= a \ln \left[\left(\tfrac{1-e\sin\varphi}{1+e\sin\varphi} \right)^{\frac{e}{2}} \tan\left(\tfrac{\pi}{4} - \tfrac{\varphi}{2} \right) \right] = au \end{aligned}$$

con u definita *latitudine ridotta*.

- La proiezione di Mercatore presenta il seguente modulo di deformazione lineare:

$$M_{Lineare} = \frac{\left(1 - e^2 \sin^2 \varphi\right)^{\frac{1}{2}}}{\cos\varphi} \tag{4.1.5}$$

- Mantiene la proprietà di *ortodromia* e di *lassodromia*[3].

[3]Trasforma in rette le linee formanti angolo costante con i meridiani.

Proiezione Gauss

È una rappresentazione analitica che può essere immaginata derivata da una proiezione cilindrica trasversa. Sul meridiano centrale la rappresentazione è equidistante e la deformazione cresce rapidamente quando ci si allontana dal meridiano centrale.
La superficie terrestre, per questo motivo, viene rappresentata mediante l'utilizzo di *fusi*[4].

Le trasformate del meridiano centrale di tangenza e dell'equatore sono ret-

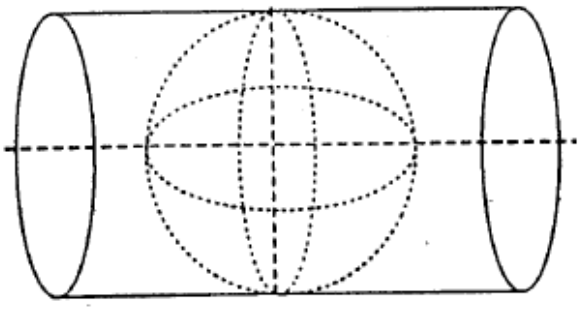


Figura 4.3: Proiezione di Gauss: forma schematica.

te e coincidono con gli assi del sistema di riferimento. Le trasformate dei paralleli sono curve approssimativamente paraboliche; quelle dei meridiani sono curve più complesse e sono via via più inclinate con l'aumentare di φ e λ. Entrambe queste famiglie di curve sono tra loro ortogonali e simmetriche rispetto agli assi di riferimento.

- La rappresentazione è conforme. Il modulo di deformazione lineare dipende quindi solo dalla posizione ed è dato da:

$$M_{Lineare} \cong 1 + \frac{\lambda^2}{2} \cos^2 \varphi \tag{4.1.6}$$

- l'angolo tra la trasformata del meridiano e l'asse y (coincidente con l'angolo che la trasformata del parallelo forma con la parallela all'asse x,

[4]Un fuso è una suddivisione dell'ellissoide in tanti spicchi delimitati da due meridiani. Ogni fuso viene rappresentato considerando il meridiano centrale come meridiano di riferimento.

poiché la carta è conforme e viene mantenuta l'ortogonalità nel passare da ellissoide a piano della carta) è detto convergenza del meridiano ed è dato da:

$$\gamma \cong \lambda \sin\varphi \Big[1 + \frac{\lambda^2}{3}\cos^2\varphi\Big] \tag{4.1.7}$$

- Con l'applicazione delle convergenza dei meridiani, dei moduli di deformazione lineare per segmenti e delle correzioni alle corde (altro termine correttivo che non vediamo), è possibile utilizzare il sistema cartografico di Gauss per eseguire calcoli relativi a figure ellissoidiche solo mediante la geometria piana.

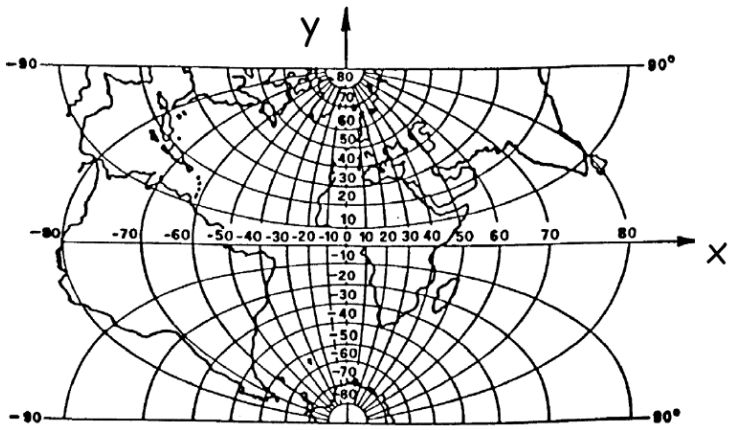


Figura 4.4: Rappresentazione canonica di Gauss.

Proiezione Universal Transverse Mercator

Tale carta è una particolare proiezione di Gauss e viene utilizzata per la rappresentazione globale dell'ellissoide terrestre (80° S $\leq \varphi \leq$ 84° N).
Fu inventata durante la seconda guerra mondiale a scopi militari.
Le principali caratteristiche sono:

- Il globo terrestre viene suddiviso in 60 fusi di ampiezza 6°. I fusi sono numerati a partire da quello compreso tra le longitudini 180° W e 174° W e procedendo verso EST (meridiano opposto a quello di Greenwich). Dal punto di vista geometrico il cilindro di proiezione è tangente al meridiano centrale del fuso;

- Si ha una suddivisione in 20 fasce di ampiezza di 8°.
- Il modulo di deformazione linare risulta:

$$M_{Lineare} = 1 + \left(\frac{\lambda^2}{2}\cos^2\varphi\right) \tag{4.1.8}$$

Per ogni fuso è stato inserito un sistema di coordinate EST-NORD e per ovviare al problema di coordinate negative sono state introdotte due costanti dette *False Origini* rispettivamente di 500.000 per la x e 10.000.000 per la y.

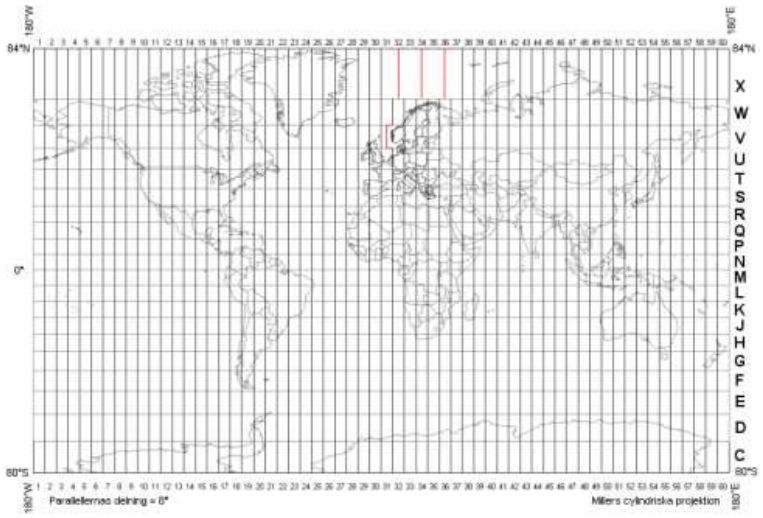
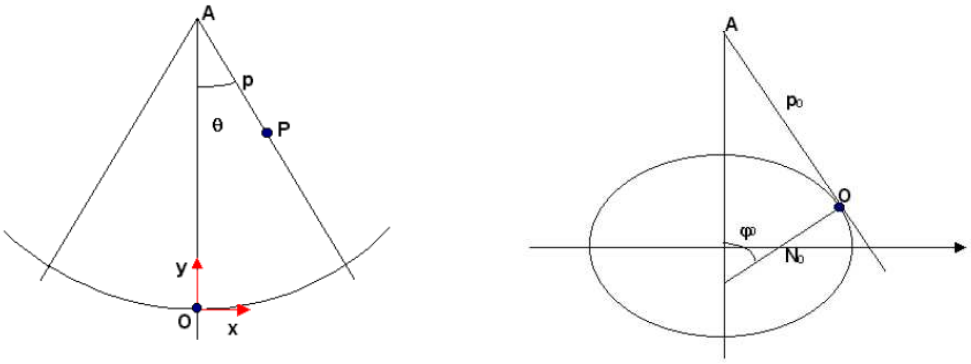


Figura 4.5: Rappresentazione canonica UTM.

Proiezione conica conforme di Lambert

La proiezione di Lambert fa parte anch'essa delle proiezioni che consentono di rappresentare i paralleli ed i meridiani mediante un reticolato cartesiano di rette tra loro ortogonali. In questo caso, però, ci si propone di ottenere una proiezione equivalente.
Questo tipo di proiezione viene vista come uno sviluppo in cui la superficie di proiezione è un cono tangente all'ellissoide lungo un fissato parallelo dove:

$$x = p\sin\vartheta$$
$$y = p_n - pcos\theta$$



ed inoltre:

$$p_0 = \overrightarrow{AO} = N_0 \cot \varphi_0$$

Di seguito le proprietà più rilevanti di questa proiezione:

- Il modulo di deformazione lineare è dato da:

$$M_{Lineare} = \frac{p \sin \varphi_0}{N cos \varphi} \tag{4.1.9}$$

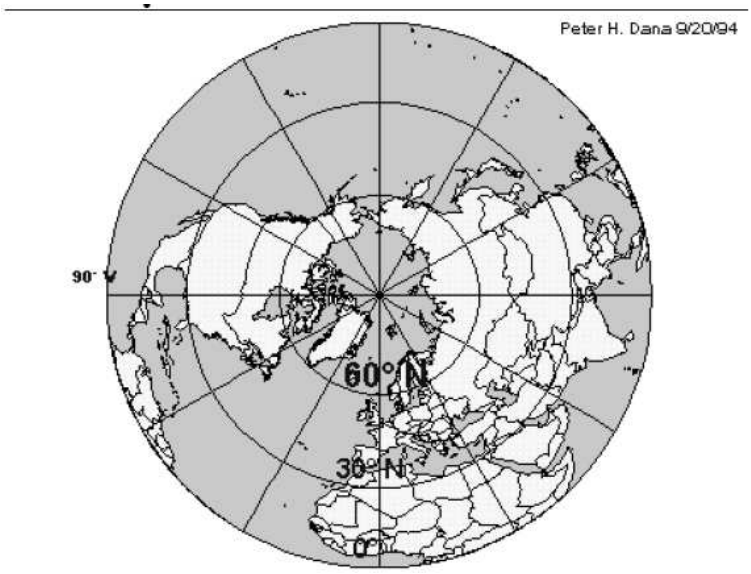


Figura 4.6: Rappresentazione canonica conforme di Lambert.

- Sul parallelo di tangenza (parallelo standard) il modulo di deformazione lineare è unitario, quindi, ai meridiani corrispondono rette parallele

all'asse y, che stanno tra di loro a distanze proporzionali alle differenze di longitudini;

- Ai paralleli le cui distanze diminuiscono al crescere della longitudine (controllare);
- In questo caso si hanno quindi due paralleli sui quali la rappresentazione è equidistante.

Proiezione di Cassini-Soldner

La proiezione di Cassini-Soldner è una rappresentazione analitica ricavata dalla cilindrica inversa.
Considerando un punto $O = (\varphi^0, \lambda^0)$ come origine, le coordinate del punto $P = (\varphi, \lambda)$ nella rappresentazione di Cassini-Soldner coincidono con le coordinate geodetiche rettangolari di P rispetto ad O. Ovvero (figura 4.7):

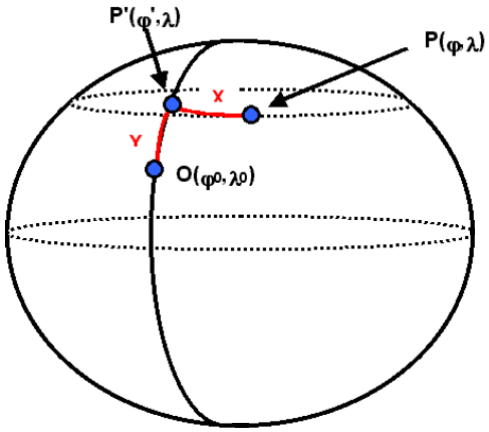


Figura 4.7: Coordinate nella rappresentazione Cassini-Soldner

- $x = PP'$ è la distanza del punto P dal meridiano origine, misurata sull'arco di geodetica perpendicolare al meridiano;
- $y = OP'$ è la distanza misurata sull'arco di meridiano fondamentale.

Questa rappresentazione non gode di particolari proprietà. La massima deformazione lineare si ha nella direzione dell'asse delle x ed è funzione della

distanza dal meridiano centrale. Purtroppo oltre ad una certa distanza il limite di deformazione diventa inaccettabile ma, se le distanze sono contenute (esempio ¡ 70 km), la carta può essere circa considerata equivalente.
Fu utilizzata come rappresentazione del Catasto Italiano.

4.1.4 Il Sistema Altimetrico

Come precedentemente citato, alla base della definizione altimetrica [28] vi è il concetto di *geoide*. I geodeti definirono l'espressione matematica della Terra prendendo come dato di partenza il suo campo gravitazionale. In ogni punto della Terra esiste la forza di gravità, che è la risultante della forza di attrazione Newtoniana e della forza centrifuga.
La Terra presenta delle caratteristiche particolari; avendo una densità non omogenea e una forma irregolare le linee di forza del campo di gravità risultano curvilinee.
I geodeti assunsero come superficie matematica della Terra una superficie che è sempre perpendicolare alle linee di forza del campo gravitazionale, rendendola univoca imponendo che fosse quella passante per un determinato punto fisico della Terra[5]. Questa superficie è il *geoide*. Il geoide è una superficie definibile fisicamente dalla quale è possibile definire le quote dei punti della superficie terrestre. Purtroppo questa definizione ha lo svantaggio di non essere rappresentabile da una equazione matematica operativa.
Come mostrato in figura 4.8 vi sono possibili due definizioni di quota:

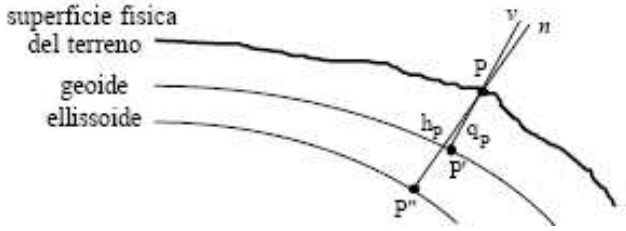


Figura 4.8: Quota Ortometrica e Ellissoidica

- *Quota Ortometrica:* è la distanza qP di P dal geoide misurata lungo la verticale n passante per esso (segmento PP' in figura)

[5]Il punto scelto è costituito dall'interpolazione della superficie media degli oceani resa stazionaria dopo aver sottratto l'influenza di maree, correnti, effetti meteorologici.

- *Quota Ellissoidica:* è la distanza hP del punto P dall'ellissoide misurata lungo la normale all'ellissoide passante per P (segmento PP'' in figura)

Esistono delle relazioni che mettono in corrispondenza un punto di coordinate geografiche (λ, φ) e quota ellissoidica h, con le coordinate X,Y,Z che tale punto ha nel sistema cartesiano geocentrico (X, Y, Z) con:

$$X = \left(\frac{a}{\sqrt{1-e^2 \sin^2 \varphi}} + h \right) \cos \varphi \cos \lambda$$

$$Y = \left(\frac{a}{\sqrt{1-e^2 \sin^2 \varphi}} + h \right) \cos \varphi \sin \lambda$$

$$Z = \left(a\sqrt{1 - e^2 \sin^2 \varphi} + h \right) \sin \varphi$$

Ovviamente è possibile anche il processo inverso che fà sorgere il problema di trasformare la quota ellissoidica in quota ortometrica.
Se non ci fossero irregolarità di distribuzione della densità della massa terrestre nella zona in cui si opera, che supponiamo compresa nel campo geodetico, sarebbe sufficiente conoscere la quota ortometrica di un punto e, conoscendo la quota ellissoidica del medesimo punto, sarebbe semplice ricavare la costante che ci permette di passare dalle quote ellissoidiche alle quote ortometriche.
In realtà il geoide non ha quasi mai un andamento regolare, ma presenta delle ondulazioni, che genera irregolarità nel campo gravitazionale e quindi nel geoide.
Queste ondulazioni, per fortuna, sono molto regolari e grazie a rilevazioni specifiche è possibile generare una serie di *mappe* di correzione, o *datum geodetici*, per generare la quota ortometrica giusta in base all'ellissoide ed al geoide considerato.
In Italia vengono utilizzati in larga scala tre tipi di datum:

- *European Datum 50:* L'ED50 è utilizzato in larga scala nella cartografia regionale di nuova produzione dell'Istituto Geografico Militare associato alla proiezione di tipo UTM Zone 32, 33, 34;

- *Roma40:* il datum Roma40 è stato il più usato a fini geodetici e topografici nella cartografia nazionale e regionale. Viene associato alla proiezione di tipo Gauss-Boaga;

- *WGS84-etrf89:* WGS84-etrf89 è di recente adozione. Viene utilizzato per l'inquadramento della nuova cartografia ufficiale al 25.000 dell'IGM. Viene utilizzato con la proiezione World Geodetic System 84 largamente utilizzata in campo mondiale e sulla rete GPS.

4.2 Librerie

4.2.1 Geospatial Data Abstraction Library

Geospatial Data Abstraction Library, d'ora in poi GDAL [38], è una libreria che si occupa di aprire, maneggiare e salvare formati di dati geospaziali in forma grezza. GDAL è rilasciata tramite licenza di tipo OpenSource con stile X/MIT.
GDAL è una libreria strutturata ed è ormai parte integrante di molti progetti, commerciali e non; tra i più importanti che utilizzano GDAL come strato a basso livello per l'utilizzo di dati geospaziali troviamo: ERMapper, GRASS, OpenEV, VirtualTerrain Project, Demeter.
Il modello di riferimento ai dati che GDAL propone è astratto a singolo strato. L'applicativo viene costruito sopra la libreria e demanderà interamente ad essa l'accesso, la formattazione e le modifiche ai dati. Le peculiarità fondamentali che rendono unica questa libreria sono:

- Le immagini gestite da GDAL vengono suddivise in bande grezze con informazioni aggiuntive comuni ad ogni banda. Il Dataset di GDAL presenta il concetto di pixel e linee applicato ad ogni banda;

- Basandosi sulla libreria *Proj.4* che ha la capacità di riferirsi ad una moltitudine di sistemi di coordinate. Il Dataset, utilizzando la libreria Proj, è in grado di manipolare i dati permettendo trasformazioni di sistemi di coordinate;

- Il Dataset è responsabile di tutte le operazini di georeferenziazione e di geotrasformazini affini sui dati;

- Includendo la libreria *OGR* (che vive internamente al codice di GDAL) riesce a gestire anche dati di tipo vettoriale accostati da metadati di completamento esterni.

Attualmente GDAL supporta una vastissima gamma di formati di dato tra i più conosciuti si trovano: Arc/Info ASCII Grid, Military Elevation Data (.dt0, .dt1), ERMapper Compressed Wavelets (.ecw), GRASS Rasters, TIFF / GeoTIFF (.tif), GeoJPEG JFIF (.jpg), GeoJP2 (.jp2), USGS ASCII DEM (.dem)[6] e molti altri.

OGR Simple Features Library

OGR Simple Features Library [38] è una libreria OpensSource scritta in C++. OGR permette l'accesso in lettura (e a volte in scrittura) ad un vasta gamma di formati di file di tipo vettoriale. Tra i tipi di formati gestiti da OGR si trovano: ESRI Shapefiles, S-57, SDTS, PostGIS, Oracle Spatial, and Mapinfo mid/mif and TAB formats.

4.2.2 Cartographic Projections Library: Proj.4

Cartographic Projections Library [45] è una libreria che si occupa di convertire coordinate generiche geografiche di tipo longitudine, latitudine in coordinate cartesiane $(\lambda, \varphi) \rightarrow (x, y)$. I metodi di conversione si basano sulle tecniche di proiezione visti in precedenza.
Proj.4 se adeguatamente configurata e se sono stati aggiunti i dati di shifting permette, inoltre, di convertire dati da un sistema di coordinate ad un altro.

4.2.3 GeoTIFF Library

GeoTIFF [40] è un formato per mantenere tipi di dati georeferenziati. GeoTIFF appoggia le sue basi sul formato TIFF in quanto è molto popolare e la sua suddivisione semplice in bande grezze lo rende molto semplice.
Il tipo di immagine TIFF può essere utilizzato per salvare immagini satellitari, foto aeree, modelli di elvazione (DEM), mappe e qualsiasi altro tipo di risultato di analisi geografiche. Con il caricamento di un set di metadata geografici, o cartografici, nell'*Header* di una TIFF nasce il supporto GeoTIFF. I dati geografici vengono utilizzati per posizionare l'immagine in una locazione spaziale corretta con le dovute operazioni di scala nel display del

[6] I Digital Elevation Models sono file (in formato ASCII o binario) contenti dati di elevazione spaziale suddivisi in pattern di griglia regolare.

sistema geografico utilizzato.
Il formato GeoTIFF è un formato completamente aperto, di pubblico dominio e non-proprietario. E' stato sviluppato dal Dr. Niles Ritter per conto di NASA-JPL (Jet Propulsion Laboratory).

4.2.4 GeoJPEG Library

GeoJPEG [39] è un clone della libreria GeoTIFF. JPEG, come ben noto, è un formato che permette una compressione molto elevata delle immagini quindi è un ottimo mezzo per scambiarsi grosse mole di informazione.
Come la precedente estende la libreria di repository di immagini JPEG georefenrenziando i dati in esso contenuti. A differenza del formato GeoTIFF i metadata geografici o cartografici risiedono in un file separato denominato *JPEG world file (.jgw)* che, durante la lettura del file immagine jpg, verrà letto e applicato ai dati.
Questa soluzione è stata adottata in quanto l'header JPEG non permette un sovraccarico di metadata differenti da quelli proposti dal protocollo nativo.

4.2.5 GeoJP2 Library

JPEG2000 [41] è una specifica ISO (ISO/IEC 15444) per il salvataggio di immagini con formato basato sulla compressione *wavelet based lossy*. Rispetto al predecessore JPEG applica una migliore compressione e dispone di un modello di immagine più flessibile.
GeoJP2tm è fondamentalmente un file GeoTiff degenerato (senza dati immagine) inclusa nella sezione UUID del formato JPEG2000. Questa intestazione si carica di tutte le informazioni riguardo il sistema di coordinate e il mapping tra le coordinate pixel e le coordinate georeferenziate.
Il formato GeoJP2tm è stato sviluppato da Mapping Science Inc. e, attualmente, GeoJP2tm, è supportato da una vasta serie di case di produzione software. Tra le più importanti si denotano: PCI, Erdas, Lizardtech, GDAL (utilizza la libreria kakadu) e Lizardtech (ex-Mapping Science).

Capitolo 5

SceneGraph e OpenSceneGraph

Fino dall'avvento delle principali librerie come *OpenGL* e *DirectX* la grafica real-time si basava esclusivamente su speciali hardware di generazione di immagini e le librerie venivano scritte sugli applicativi specifici [1]. I formati di dato, ed i relativi Database nei quali dati venivano suddivisi, erano in formati proprietari creati apposta per la specifica macchina di visualizzazione. Inoltre, i programmatori, erano limitati nella modifica dei modelli che dovevano essere processati.
Consci di queste limitazioni grandi *software/hardware house* decisero di sviluppare una serie di linguaggi grafici per riuscire ad intervenire direttamente sulla pipeline di rendering.
In questo modo nacque la libreria grafica *GL* (tuttora conosciuta con il nome di *OpenGL*). OpenGL consiste in uno stream di primitive per intervenire sui poligoni, sui settaggi delle variabili di stato ed infine sulle matrici applicate alle geometrie ed alla vista.
Il passo successivo è stato quello di definire un'astrazione superiore per le gestione della scena di visione e per definire in maniera più dettagliata gli interventi applicabili su di essa. Da queste piccole considerazioni si definisce l'idea di SceneGraph.

5.1 Cos'è uno SceneGraph

Uno Scenegraph è una struttura a grafo che suddivide la scena di visione caricandola di significati semantici in modo tale da potere semplicemente

identificare oggetti, comportamenti e dimensioni [16].
Solitamente uno SceneGraph è un grafo di tipo diretto aciclico di oggetti chiamati nodi. Ogni nodo può avere un padre e m nodi figli. All'estremità dell'albero sono presenti i nodi foglia che rappresentano i singoli oggetti che saranno visualizzati nella scena e conterranno tutte le informazioni riguardanti geometrie, materiali, textures, luci e tutte le caratteristiche applicabili ad un oggetto.
L'idea di SceneGraph, nodi e gruppi fa sì, inoltre, che vengano nascoste in semplici chiamate a funzione molti passaggi e procedure che normalmente occorrono per gestire una libreria grafica a basso livello.
La suddivisione in gerarchie dello scenegraph fa in modo che, durante la fase di renderizzazione della scena, saranno visualizzati solo i nodi visibili nel cono di visione. Questo peculiarità è resa possibile grazie all'attraversamento dalla radice fino alle foglie dell'albero di scena. Se un nodo si trova all'esterno del cono di visione non sarà attraversato e, quindi, anche i relativi figli non verranno processati risparmiando tempo e risorse di calcolo. Questo viene garantito dal fatto che ogni nodo presenta un volume (tipicamente una box o una sfera), detto *Bounding Volume*, che mantiene le dimensioni del nodo e dei relativi figli ad esso collegati.
La suddivisione della scena in un albero di visione permette di semplificare la manipolazione degli oggetti. Applicando dei comportamenti a determinati nodi di un oggetto (comportamenti che possono variare, per esempio, da semplici trasformazioni spaziali a vere e proprie risposte di callback a determinati eventi) è possibile modificare in maniera indipendente parti dell'oggetto in questione o di altri ad esso collegati.
Il concetto di SceneGraph ha la peculiarità di riuscire ad incrementare le prestazioni e supportare un utilizzo migliore delle risorse hardware. L'attraversamento dello SceneGraph, se fatto in maniera ottimizzata, può permettere anche una suddivisione delle strutture in modo tale da rendere la fase di *draw* ancora più performante.
Utilizzando la tecnica di *State-Sorting*, è possibile gestire in maniera ottimizzata gli stati che rappresentano un oggetto. Lo stato non è altro che una rappresentazione di come l'oggetto verrà renderizzato e il ricalcolo di uno stato è, normalmente, una operazione onerosa. Se il raggruppamento degli oggetti avviene tramite il riconoscimento dei tipi, allora, le modifiche

e la resa di determinati stati avviene solo all'interno dello stesso frangente. Inoltre, è possibile definire una sequenza di renderizzazione in base alla variazioni tra stati differenti, in questo modo, viene minimizzata la modifica ai parametri di stato del motore di render durante lo switching tra stati simili (*State-Encapsulation*).

5.2 OpenSceneGraph

OpenSceneGraph è una libreria ad alto livello per lo sviluppo di applicazioni grafiche 3D [43]. È un progetto OpenSource e, confrontandolo con altri progetti dello stesso calibro come SGL o OpenSG, risulta essere il più attivo nello sviluppo.
OpenSceneGraph nasce nel 1998. Diventa un progetto OpenSource nel 1999 e, attualmente, ha raggiunto lo stadio beta con l'uscita della release 0.99. OSG è ispirato pesantemente alla libreria Performer [7] (libreria grafica C misto C++ Closed-Source ad alto livello sviluppata dalla Silicon Graphics) del quale ricalca grosso modo tutta la struttura.
Il core di OSG è scritto interamente in C++ utilizzando la filosofia Object-Oriented in modo tale da incapsulare tutte le funzionalità di OpenGL aumentandone la potenzialità espressiva e ottimizzando direttamente le chiamate a funzione di basso livello.
Il punto di forza di OpenSceneGraph sono le sue prestazioni, la sua produttività, la sua portabilità e la sua scalabilità che vengono guadagnate dall'utilizzo della tecnica dello SceneGraph.
In dettaglio:

- *Prestazioni:* Le prestazioni sono garantite dal fatto di poter utilizzare differenti tipi di algoritmi che effettuano *Frustum Culling* (definisce gli oggetti fuori dalla scena), *Occlusion Culling* (scova gli oggetti nascosti e ne previene il rendering) e *State-Sorting*. OSG è in grado di utilizzare altre tecniche, ad alto livello, per la gestione del processo di rendering quali:
  - la gestione dei livelli di dettaglio;
  - la gestione di mesh con livelli di dettaglio continui.

- *Produttività:* Con produttività si intende la facilità di utilizzo. Grazie alla tecnica dello SceneGraph uno sviluppatore può concentrarsi solo sul contenuto della propria applicazione senza doversi preoccupare della struttura a basso livello del proprio programma.
  Estendendo le funzionalità fissate da colossi come Performer e OpenInventor con tecniche di ingegneria del software (come il *Design-Patterns*[8]) è stato possibile progettare una libreria completamente estendibile e pulita. OSG garantisce l'utilizzo di più di 45 tipi di formati di dato (3D, immagine e video) grazie al meccanismo di estendibilità dinamica dei plug-in.
  OSG presenta anche due tipi di formato proprietario per il salvataggio dei modelli e delle scene: *.osg* (in versione ASCII) mantiene tutte le informazioni riguardanti la struttura dell'albero di scena, gli stati degli oggetti e le geometrie; *.ive* (in versione binaria) è un clone del formato precedente con in più la possibilità di includere all'interno del file le texture applicate sui modelli
  Il codice dello scenegraph può essere esteso con set di *NodeKit* (sia a compile-time che a run-time) che estendono il core di libreria aumentandone l'espressività e le potenzialità. Nella distribuzione, ad esempio, sono già presenti NodeKit che ne estendono l'uso con sistemi particellari (osgParticle), effetti speciali (osgFX), gestioni di database di terreni su larga scala generati da dati geospaziali (osgTerrain), gestione del movimento dei personaggi (osgCal), simulazione delle leggi della fisica (osgVortex), gestione dei linguaggi di Shading (osgNV), ...

- *Portabilità:* OSG è scritto interamente in standard C++. Questo fatto ha permesso di creare del codice quasi del tutto indipendente dalla piattaforma sul quale è compilato rendendo semplice la migrazione tra la maggior parte dei sistemi operativi presenti.
  OSG è distribuito in un'unica distribuzione di sorgenti capace di adattarsi su differenti piattaforme quali: Linux, Linux64, FreBSD, Irix, SGI Redhat, Solaris, MacOSX, Windows, XBox e PlayStation 2.

- *Scalabilità:* essendo altamente portabile OSG può essere adattato anche su macchine di tipologia differente sfruttando appieno le potenzialità disponibili. OSG può facilmente scalare su architetture come semplici

PC fino alle workstation di SGI utilizzando multiprocessori e multipipe. Queste features sono garantite dall'utilizzo di due librerie, OpenProducer e OpenThreads sulle quali il core si appoggia.
OpenThreads permette la gestione dei threads ad alto livello controllandone il corretto funzionamento ed evitando disfunzioni o collisioni.
OpenProducer gestisce i Context OpenGL indipendentemente dalla libreria grafica che si sta utilizzando. In questo modo OSG risulta essere un sistema *Windowing System Indipendent* garantendone una invidiabile interoperabilità con le maggiori librerie di interfacce utenti presenti come: *GLUT, wxWindows, SDL, TclTk, Motif, X*. Producer garantisce, inoltre, l'utilizzo di sistemi multimonitor e multiwindow.

5.2.1 Libreria

La libreria da distribuzione di OpenSceneGraph è organizzata nel seguente modo:

osg questa libreria è il cuore di OSG. Contiene tutte le implementazioni delle classi che rappresentano i nodi. Implementa nodi base, come gruppi, nodi di trasformazione, nodi di geometria, fino ad arrivare a nodi di tipo altamente specifico, come BillBoard, nodi Impostor, nodi Livelli di Dettaglio (LOD), nodi Switch. Nella libreria osg sono presenti anche tutti gli elementi base che sono di indispensabili nella gestione dello scenegraph, come le classi stato, classi di gestione dei volumi, classi per la gestione delle operazioni matematiche;

osgUtil questa libreria contiene molte utility per la gestione della scena. Presenta delle classi per l'ottimizzazione del grafo e delle geometrie; presenta dei metodi per la semplificazione delle strutture; Mantiene la classe SceneView che si occupa di rappresentare la vista globale della scena permettendo ai NodeVisitor di intervenire su di essa; Mantiene tutto il set dei NodeVisitor base per modificare, o semplicemente visitare, l'albero di scena;

osgDB costituisce la libreria per la gestione del database dei formati e per il caricamento dei dati. Implementa la classe virtuale ReaderWriter che è utilizzata per caricare dinamicamente i Plug-In di gestione dei formati

tramite la classe Registry che si occupa precedentemente di interpretare il tipo di formato da leggere. osgDB mantiene anche la classe DataBasePager che consente di effettuare il caricamento indipendente dei dati. Con l'utilizzo di particolari nodi detti di *Paginazione* è possibile poter caricare i dati solo all'occorrenza se opportunamente suddivisi;

osgGA è la libreria che si occupa dell'adattamento agli handler input/output e degli eventi dalla interfaccia grafica a finestre;

osgFX è un framework di effetti speciali. Provvede ad implementare una moltitudine di effetti che si basano sull'hardware grafico disponibile con l'avvento delle nuove GPU;

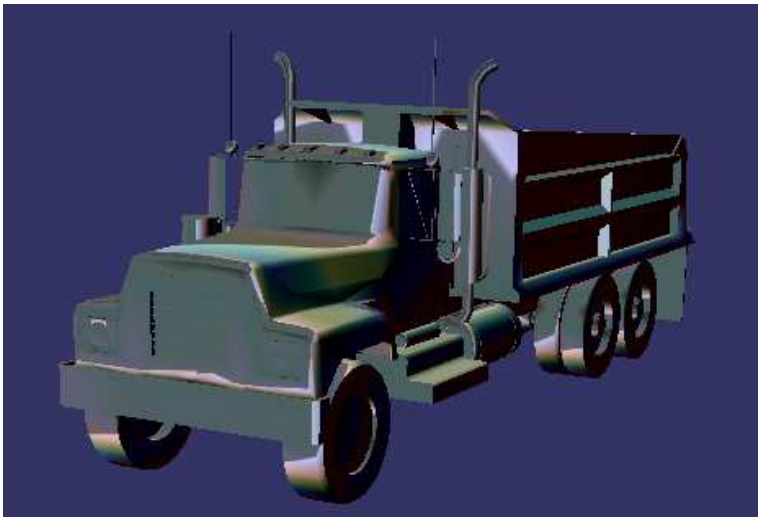


Figura 5.1: Esempio di luce anisotropica applicata ad un modello 3D utilizzando osgfxbrowser di OpenSceneGraph.

osgText in realtà questa libreria è nata come NodeKit ma, ormai, è parte integrante della libreria di base. Implementa molte funzionalità per l'utilizzo di testo all'interno della scena grafica 3D;

osgParticle questo NodeKit si occupa della gestione di sistemi particellari come: fuoco, fumo e fluidi;

osgTerrain Questo NodeKit mantiene una serie di classi per il supporto alla generazione di database geospaziali corretti. osgTerrain implementa il

Figura 5.2: Esempio di utilizzo della libreria osgParticle.

concetto di DataSet e permette una generazione di database di terreni in differenti formati sfruttando le tecniche di paginazione e di Height-Field. Questa libreria è basata sulla libreria Geospatial Data Abstract Library e sulla Cartographic Library.

osgProducer questa libreria implementa un Viewer già pronto con tutta la gestione dei comandi, degli eventi da mouse e tastiera per sviluppare più velocemente le applicazioni. Utilizza gli Handler eventi della libreria osgGA. Estende la libreria Producer in modo tale dal essere contex indipendent, multiprocessor e multipipe.

5.2.2 Tools

Con la distribuzione di OpenSceneGraph sono inclusi, assieme ad una miriade di esempi che vanno a sopperire alla mancata documentazione del codice, una serie applicazioni che permettono un approccio iniziale con la libreria:

osgviewer è un viewer base che permette di visionare oggetti 3D. La sua funzione primaria è quella di esempio di come scrivere un viewer base ma è già abbastanza funzionale da poter essere usato come navigatore 3D;

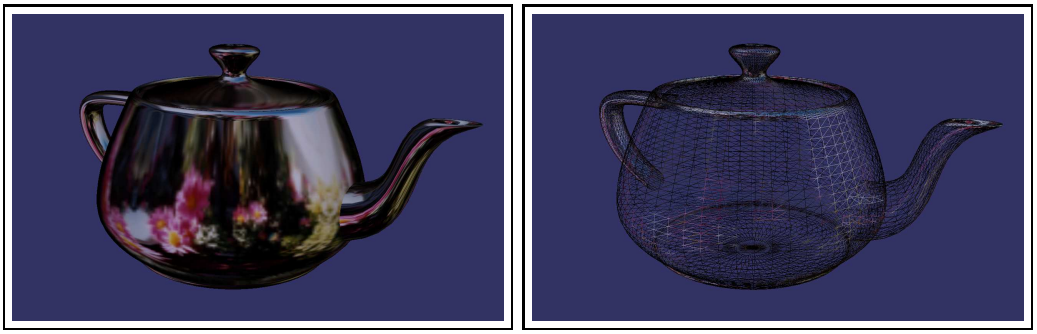


Figura 5.3: Modello caricato in osgviewer. *A sinistra:* Visualizione classica. A destra: Visualizzazione in WireFrame.

osgconv è un programma di utilità per riuscire a convertire semplicemente formati 3D nei formati nativi di OSG. Questa versione permette di convertire anche 3D database in singoli file ottimizzati OSG. I tipi di formati supportati sono quelli compresi nei plug-in di libreria;

osgarchive crea e gestisce file archivio contenenti serie di file per il 3D. Gli archivi generati da osgarchive possono essere letti direttamente da tutti gli applicativi di OSG senza il bisogno di essere scompattati;

osgdem questa utility si occupa di leggere immagini geospaziali e mappe digitali di elevazione per generare un database di terreni 3D su larga scala che applicazioni basate su OSG possono caricare e navigare in tempo reale.

Capitolo 6

Virtual Terrain Project: suite e modifica

Il set di applicativi sviluppato per la generazione, ricostruzione, navigazione e fruizione via web di terreni, applicati in ambienti Real-Time, è stato possibile sfruttando ed ampliando in parte la libreria di rendering 3D OpenSceneGraph, gli applicativi Enviro e VTBuilder (e le relative librerie) del pacchetto VTP.
Dopo un processo di riadattamento e modifica, Virtual Terrain, da semplice toolkit per la visualizzazione di ambienti 3D, è diventato una vera e propria piattaforma per la creazione, il modelling ed il riadattamento di scenari geografici. Abilitandolo all'utilizzo di terreni e modelli generati da fonti esterne (terreni paginati, modelli 3D fotomodellati, etc.) è stato possibile sfruttare tutte le caratteristiche del pacchetto VTP all'interno di un flusso di lavoro di più largo respiro e di maggiore operatività; ciò lo ha reso un medium capace di aggiungere ai terreni importati tutte le proprie *Culture* rendendole esportabili ed utilizzabili da altri software.
Per l'estensione di importazione ed esportazione è stata utilizzata la configurazione basata su OSG in quanto può trattare un vasto set che comprende la maggioranza di formati in ambito 3D ed inoltre è in grado di gestire terreni creati dalla libreria osgTerrain.
Utilizzando dei moduli di OSG è possibile riuscire ad accedere a dati da fonti remote. Questa estensione è stata fondamentale nello sviluppo di un plug-in ActiveX per Internet Explorer in grado di riuscire a reperire i dati, generati

da VTP, da un server web rendendo possibile la navigazione dello scenario. Naturalmente, al fine di rendere la navigazione scorrevole, la generazione dello scenario dovrà essere concepita nel modo più *leggero* possibile. Sia a livello di esportazione dei modelli che in fase di caricamento dei dati via web, alcuni accorgimenti a tal scopo sono già stati presi attraverso: tecniche di caching su disco, utilizzo di modelli replicati nella scena e generazione di strutture poligonali semplici.

6.1 Virtual Terrain Project

Virtual Terrain Project [46] nasce come progetto e come comunità che è interessata alla ricostruzione 3D digitale di territori.
Il compito primario del progetto Virtual Terrain è il raccogliere informazioni sull'andamento delle nuove tecnologie hardware e software nel campo della visualizzazione territoriale sia in ambito OpenSource che commerciale. In questo modo VTP risulta essere una comunità attiva di persone disponibile nello scambio di informazioni e nell'affrontare problematiche riguardo il trattamento di dati geospaziali.
Oltre al ruolo di supervisione che il progetto Virtual Terrain svolge viene sviluppato parallelamente un software OpenSource per la visualizzazione in tempo reale di territori (Enviro) e una serie di tool (VTBuilder, CManager...) per l'elaborazione di dati geospaziali e la gestione dei modelli applicabili al territorio.
Questi applicativi prendono forma adagiandosi su quattro librerie sviluppate all'interno del progetto.

6.1.1 Librerie

vtdata Questa libreria si occupa di generare le classi base per la gestione dei dati, le operazioni matematiche e geospaziali, le operazioni di Input/Output sui file ed infine la gestione dei sistemi di coordinate geospaziali. Queste classi operano sui dati in maniera del tutto trasparente consentendo allo sviluppatore di manipolare e convertire i dati in

maniera semplice ed intuitiva.
La maggior parte di queste classi risultano essere delle classi astratte in quanto servono solo come definizione dei tipi. L'adattamento successivo dei dati alla libreria di renderizzazione 3D utilizzata verrà compiuto dalla libreria vtlib;

vtlib Estende la libreria vtdata con l'abilità di creare geometrie 3D per una corretta visualizzazione di mondi virtuali interattivi. Questa libreria mette in opera l'astrazione di SceneGraph implementando una vasta serie di classi per la gestione dei nodi, della scena di visione, degli oggetti e degli stati. Vtlib, nella versione attuale, ha la capacità di utilizzare quattro tipi differenti di librerie di rendering 3D quali: OpenSceneGraph, SGL, PLIB ed OpenSG. Tra le feature che questa libreria può supportare si denotano:

- supporto ad una serie di algoritmi di renderizzazione di tipo Continous Level of Detail di mesh regolari generate da HeightField tipo Röetger, Demeter ...;
- applicazione di texture map geospecifiche alle mesh regolari;
- creazione e gestione dello skydome e simulazione dell'illuminazione solare;
- generazione procedurale di strade, costruzioni e vegetazione;
- raggi di intersezione con l'HeightField;
- diversi tipi di handler di navigazione della scena.

vtui Gestisce le libreria di interfaccia verso l'utente implementando diverse maschere e finestre. Al momento attuale la libreria di interfaccia supportata è solo wxWindow ma sono presenti diversi esempi di utilizzo del core di vtp con altre User Interface.

xmlhelper Gestisce il salvataggio e lo scambio dei dati tra tool. Il formato di base è in XML ed utilizza la libreria *xerces* per caricare e salvare i file.

6.1.2 Tools

VTBuilder VTBuilder è un software di tipo CAD-like per l'elaborazione e manipolazione di dati geospaziali. Ovviamente, tutti i formati di dati esportabili vengono ottimizzati per il software di visualizzazione Enviro. Permette la conversione in maniera automatica di coperture vettoriali, DEM, immagini georeferenziate in una serie di formati adatti alla creazione di una scena 3D Real-Time.
VTbuilder permette la generazione di terreni in forma di HeightField; manipola una o più serie di texture per essere compatibili col terreno; è in grado di generare coperture di vegetazione realizzate in base a librerie adatte al contesto di ricostruzione; permette, inoltre, la ricostruzione di edifici e di strade in maniera del tutto automatica;

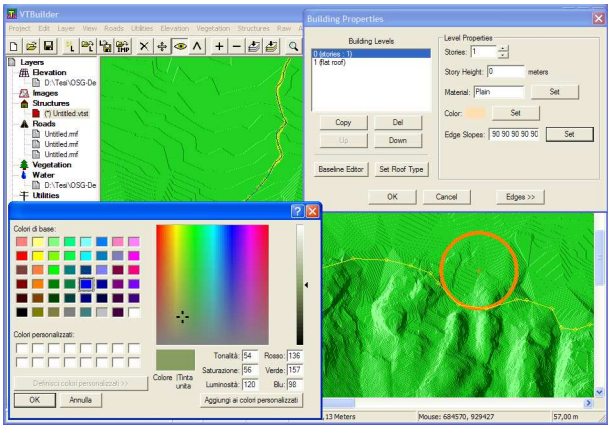


Figura 6.1: Visualizzazione di un progetto GIS con vtBuilder.

VTConvert Tool a linea di comando di conversione tra formati differenti;

CManager Content Manager per la creazione delle librerie contestuali di oggetti utilizzati successivamente nell'adornamento del paesaggio. Permette di posizionare e classificare gli oggetti;

Enviro Enviro è un completo visualizzatore che permette di navigare attraverso i terreni, e di attivare o disattivare la visualizzazione di tutte le parti del modello. In Enviro sono presenti due modalità di visualizzazione: una che mostra il pianeta terra su cui sono indicate tutte le zone selezionabili che presentano un modello navigabile, l'altra che mostra

una rappresentazione dettagliata della zona.

La caratteristica fondamentale che distingue Enviro da qualunque al-

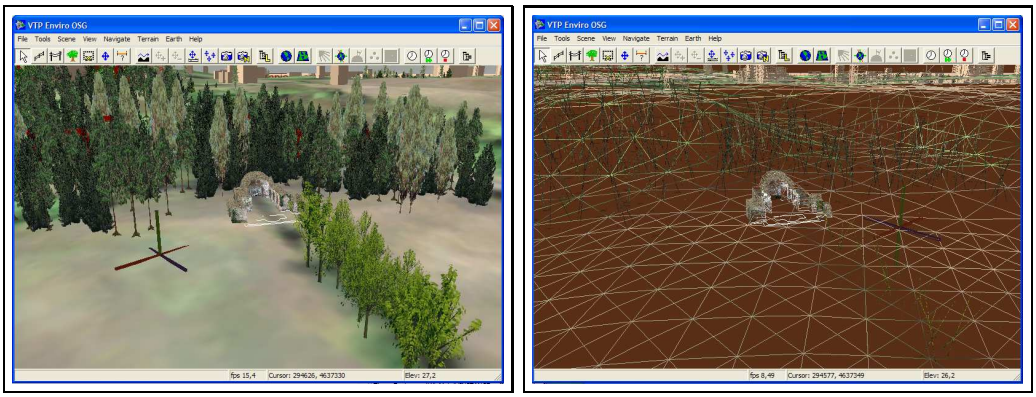


Figura 6.2: Visualizzazione di un progetto nella versione di Enviro di *Release*. *A sinistra:* Visualizione classica. A destra: Visualizzazione in WireFrame.

tro Viewer è il fatto che permette di modificare *al volo* tutte gli oggetti inclusi nel paesaggio. In questo modo è possibile intervenire in maniera del tutto naturale all'interno di una ricostruzione senza doversi preoccupare di una errata modifica. Operazioni possibili sono:

- Aggiunta, posizionamento e cancellazione di modelli;
- Modifica dei parametri applicati alle geometrie procedurali.

6.2 Importazione terreni

Come precedentemente citato, per riuscire ad integrare il pacchetto VTP in un contesto di generazione di territori di più largo respiro si è dovuti intervenire in differenti punti dei software contenuti nella suite.
In questa sezione verranno argomentate le varie modifiche effettuate al tool di visualizzazione *Enviro* in modo tale da abilitarlo ad accogliere formati di dati territoriali generati da software esterni.
Questo intervento è stato necessario in quanto il motore di rendering dei terreni sul quale è basata la visualizzazione utilizza un algoritmo, selezionabile tra diverse tipologie all'avvio di un progetto, di tipo C-LOD; in questo modo i dati territoriali indispensabili sono in forma di HeightField. Come da specifiche nel capitolo 3 questo approccio presenta molti aspetti positivi,

definiti dal tipo di algoritmo utilizzato, ma, gli algoritmi di C-LOD hanno come svantaggio comune il dover utilizzare e processare un HeightField monolitico. Diventa un problema estremamente evidente in presenza di terreni ad una risoluzione ed una estenzione elevata.
Dopo alcune settimane di prove, ed in risposta ad un *feedback* positivo nel lavoro, si è incentrato lo sviluppo verso l'importazione di terreni di tipo statico ottimizzati per una successiva fruizione via Web. Questa fase di intervento è stata il punto di partenza di tutto il lavoro di tesi in questione.
Durante la fase di studio di fattibilità sono state vagliate diverse ipotesi per la modifica di terreni 3D ma, subito dopo alcune prove iniziali e piccoli tentativi, l'attenzione del progetto si è subito focalizzata verso la modifica della suite VTP in assetto OSG.
Enviro è un applicativo ad interfaccia grafica basato sulla libreria wxWindows [47]. Mantenendo la struttura di una semplice applicazione wxWindows il core di Enviro è suddividibile in tre sezioni principali. La prima sezione (d'ora in poi fase di *Init*) costruisce tutte le struttura dati fondamentali per l'applicativo, genera la finestra di log principale, dalla quale è possibile gestire tutti i parametri di progetto, e le finestre principali del tool. La seconda sezione (fase *Frame*) gestisce tutta la creazione della scena, la visualizzazione e gli eventi esterni implementando una macchina a stati.
Infine, la fase di *Exit*, gestisce la chiusura della finestra e la deallocazione delle strutture dati dalla memoria.
Il Widget di Frame, utilizzato nel core di Enviro, implementa una finestra di visualizzazione OpenGL-Based. OpenSceneGraph (e le altre librerie ad alto livello sulle quali Enviro può basarsi) vengono solo utilizzate come contenitori semantici e gestori dell'albero di scena non implementando la RenderWindow direttamente. Questa soluzione ha permesso di riuscire a far convivere le librerie di gestione della scena parallalamente agli algoritmi di renderizzazione del terreno senza dover addentrarsi troppo nelle librerie grafiche ad alto livello.

Step di generazione di un terreno tramite la libreria *vtlib*

Sezionando il codice di *Release* di Enviro, tralasciando tutta la parte di inizializzazione delle maschere di finestra, del recupero dei parametri di configurazione e di creazione della RenderWindow, è possibile identificare la fase

di inizializzazione della scena di visualizzazione.
Analizzando la classe *Enviro* e focalizzandoci sul metodo SetupTerrain, è possibile definire gli step per la generazione di tutte le geometrie del terreno e di scena; cercando di riassumere tutta la procedura sono stati individuati cinque passi che, in base ai dati di input, generano la scena:

- *CreateStep1* Si occupa di settare i parametri geospaziali e di proiezione. In secondo luogo carica il file di HeightField (in formato Binary Terrain - *.bt* - o il file TIN) con successivo instanziamento delle strutture preposte al mantenimento;

- *CreateStep2* Questa fase carica l'eventuale texture del terreno (o mappa di elevazione) modificata in base alla direzione dell'illuminazione solare;

- *CreateStep3* CreateStep3 genera la mesh del terreno (o la geometria statica) e instanzia il motore di rendering in base all'algoritmo di visualizzazione;

- *CreateStep4* Lo step quattro applica alla mesh dinamica la texture, precedentemente caricata, modificandone eventualmente proprietà e materiali in base alla scala di toni di elevazione. Eventualmente viene attivato un ulteriore step di raffinamento delle geometrie in base all'algoritmo di rendering utilizzato (ad esempio è il caso della selezione dell'algoritmo di *Roettger*);

- *CreateStep5* CreateStep5 genera tutte le coperture vettoriali. In particolare genera le coperture delle case, degli alberi, delle label, dei modelli 3D e delle *Fance* procedurali.

Terreni di test

Tutti i test svolti durante questa fase iniziale di sviluppo sono stati possibili attraverso la generazione di quattro territori in formati differenti. Le fonti di generazione sono state le stesse in modo tale da constatare semplicemente dei disaccoppiamenti o errori nella visualizzazione; in particolare, è stato utilizzato un file di elevazione in formato griglia (*GRID*) con georeferenziazione *UTM 33 datum ED50* del Parco naturale dell'Appia (Sezione 2.4.2) e un collage di foto aeree in formato *TIFF* con identica georeferenziazione.

I terreni sono stati creati utilizzando due generatori differenti: *osgdem* e *terravista*[1].
Con *osgdem* sono state prodotte due gerarchie di terreni identici variando solo il formato di salvataggio. Il primo terreno è stato generato in formato *.ive* (formato binario di OSG). Questo formato ha la caratteristica di contenere all'interno sia geometria che texture. Il secondo terreno è stato generato in formato *.osg* (file in formato ascii). Le texture sono salvate in file esterni con formato di tipo *.dds*.
La gerarchia di paginazione dei file è di tipo Quad-Tree ed ogni file mantiene le specifiche di georeferenziozione tramite il nodo *CoordinateSystemNode*[2].
La trasformazione di georeferenziazione vera e propria, invece, è demandata ad un nodo *MatrixTransform* (figlio del nodo sistema di coordinate) che si occupa di applicare le trasformazioni di scala e di traslazione essenziali.
Partendo sempre dalla stessa fonte di dati, con *TerraVista*, sono state generate altre due gerarchie di file. Il primo è in formato OpenFlight (formato dati sviluppato dalla MultiGen[42]). Questa gerarchia presenta un file Master che funge da archivio di richiamo per tutti gli altri file utilizzando la tecnica dei nodi *ExternalReference*[3]. I file richiamati dal master, invece, mantengono al loro interno la geometria suddivisa in Livelli di dettaglio statici in base dalla distanza dell'osservatore. Il formato OpenFlight non permette lo stoccaggio delle texture e Terravista le salva esternamente in formato *.rgb*. Oltre allo svantaggio di generare file a dimensione abbastanza elevata, Terravista non riesce ad includere le specifiche di georeferenziazione all'interno dell'OpenFlight. Per ovviare questa mancanza si è associata una specifica di georeferenziazione esterna tramite gli extent del terreno ed un eventuale fattore di scala dato dalla georeferenziazione.
Infine, come ultima gerarchia di terreni presi in considerazione, è stata generata da Terravista una gerarchia di tipo *TerraPage*. La gerarchia prodotta è

[1] Terravista (sviluppato della Terrex [48]) è un software di modellazione territoriale che consente di gestire ed utilizzare tutti i tipi di dati geospaziali e combinarli insieme producendo in uscita un modello del territorio sotto forma di gerarchia di file OpenFlight (o Terrapage).

[2] Nodo speciale della libreria core di OSG. Mantiene, in formato stringa, la georeferenziazione prodotta dalle librerie GDAL e ProjLib (per informazioni fare riferimento al capitolo 4.2.1 a pagina 41) ed, inoltre, è in grado di generare direttamente le conversioni di coordinate dal sistema di riferimento locale al sistema georeferenziato.

[3] Per spiegazione fare riferimento ai nodi *ProxyNode*: capitolo 6.3.3

Parametro	*Valore*	*Descrizione*
fileextname	flight/master.flt	*Master file (path relativo o indirizzo http)*
filebtname	demparco.bt	*HeightField di supporto*
georeference	0	*Validità della georeferenziazione*
x_down	292478.364877585	*Estensione del terreno*
y_down	4631156.68419493	
z_down	0.000000	
x_up	298878.364877585	
y_up	4638356.68419493	
z_up	0.000000	
x_scalef	1.000000	*Fattori di scala*
y_scalef	1.000000	
z_scalef	1.000000	

Tabella 6.1: Esempio di formattazione file *.ebc*

simile a quella vista con gli OpenFlight ma con la differenza che la georeferenziazione del terreno è specificata all'interno di un file esterno di proiezione generato da Terravista.

6.2.1 Lettura parametri di supporto al Loading

Come primo step di modifica si è intervenuti nella sezione di gestione dei settaggi di progetto, della visualizzazione in Enviro, in modo tale da riuscire ad inserire tutte le informazioni riguardanti i terreni esterni. Per evitare un impatto notevole di intervento nel codice della finestra di settaggio, tutti i dati relativi ai terreni sono stati inseriti all'interno di un file di testo (con estensione *.ebc*); il file contiene il riferimento al terreno da visualizzare [4], il file HeightField normalmente processato da Enviro e le eventuali specifiche di georeferenziazione aggiuntive (esempio è riportato nella tabella 6.1). La scelta di mantenere il file locale di HeightField è imposta dal fatto che il motore di navigazione, il Collision detection con il terreno, la generazione

[4]Nella versione attuale di modifica è possibile specificare la lettura del master file del terreno esterno anche da remoto specificando un indirizzo *http*. In una versione prossima sarà implementata anche la lettura in remoto del file di HeightField.

delle coperture vettoriali e le query sull'altezza rispetto al livello del mare si basano sulle informazioni contenute in esso. Una implementazione di queste features in modo tale che utilizzino il terreno di paginazione è possibile ma implicherebbe una lettura totale del terreno esterno in quanto, ad esempio, il solo movimento del mouse sul terreno genera una query sull'elevazione che produrrebbe un risultato sbagliato in presenza di una Tile a bassa risoluzione. La lettura completa dei livelli ad alta risoluzione di un terreno paginato è comunque più dispendiosa della lettura di un file aggiuntivo di HeightField.

Un ulteriore punto a favore della implementazione sopra descritta risiede nella possibilità di pensare di focalizzare un intervento di modifica su una regione del territorio. Considerando un terreno paginato di dimensioni molto estese e conoscendo solo l'informazione di una finestra di Heightfield è possibile pensare ad un intervento di modifica mirato solo su una piccola zona demandando il problema dello stoccaggio dei dati (o della generazione dell'HeightField) a server remoti sicuramente molto più potenti di una piccola macchina satellite che richiede i dati.

Dopo un attento studio degli step di inizializzazione nella fase di *Frame* si è individuato, come punto di intervento per l'inserimento della lettura del file di inizializzazione *.ebc*, la sezione precedente alla lettura del file di HeightField (nello specifico all'interno della funzione SetupTerrain della classe Enviro precedentemente allo step CreateStep1). In questa zona è stata implementata una classe contenitore che mantiene i nomi dei file da caricare e la georeferenziazione aggiuntiva in caso di mancata specifica interna (caso dei file *.flt*). Una volta ottenute le informazioni sui file sono ripristinati i normali parametri di settaggio con la specifica che è presente un Master file di un territorio esterno da caricare.

6.2.2 Matrice di trasformazione e Loading

Come per l'inserimento dei paramtri esterni il punto di intervento per questa modifica risiede sempre all'interno della classe Enviro nel metodo SetupTerrain. In primo luogo si è modificato il metodo, appartenente alla libreria vtlib della classe vtTerrain, CreateStep3. Questa funzione ha il compito di generare, in base al tipo di algoritmo di C-LOD selezionato o al TIN, la Mesh (dinamica o statica) per la visualizzazione del terreno. In particolare, se la mesh è di tipo dinamico viene bypassata la creazione della griglia introdu-

cendo una struttura fittizia minima che ha il compito di ingannare il terrain engine in fase di rendering. Questa scelta trova ragione solo su una costatazione di impatto sul codice in quanto, dopo alcuni esperimenti di modifica radicale sulle chiamate dirette ai motori di renderizzazione, ci si è resi conto che le variazioni iniziavano a diventare ingestibili nel tempo a disposizione per questo lavoro di tesi.
Sempre durante questo step di inizializzazione si è intervenuti implementando una funzione di loading e trasformazione del terreno in coordinate locali dell'environment corrente in Enviro. In primo luogo si è intervenuti ricavando tutti i parametri essenziali per gestire una corretta trasformazione e si è rinvenuto, nella classe vtLocalConversion, tutte le informazioni di traslazione e di scala dalle coordinate georeferenziate a quelle locali. A questo punto, in base al tipo di terreno caricato ed alle informazioni rinvenute nel file ebc, è stata generata la matrice di trasformazione che verrà applicata al modello 3D del terreno. Per motivi, forse più di pensiero che di effettiva praticità, Enviro ed OSG differiscono per l'orientazione degli assi di riferimento; tenendo conto di questa piccola divergenza si è dovuta applicare una ulteriore rotazione di 90° sull'asse X in modo tale da trasportare i modelli da un sistema di riferimento di tipo Y up ad un sistema di riferimento Z up.
Utilizzando una peculiarità portante di OSG, nello specifico la classe astratta *ReaderWriter* (appartenente al namespace *osgDB*) e l'utilizzo dei Plug-In di lettura e scrittura dei file, il caricamento dei dati è stato demandato direttamente alla libreria grafica di base.
L'operazione di caricamento e di salvataggio dei dati in OpenSceneGraph coinvolge diverse classi. Tutte queste classi sono appartenenti o derivate da quelle contenute all'interno della libreria *osgDB* la quale rappresenta l'interfaccia che OSG utilizza per accedere ad una moltitudine di formati di dato differenti. Di seguito sono state riportate le classi principali che si fanno carico di gestire questo tipo di operazione

- *ReaderWriter:* è una classe astratta che rappresenta principalmente un'interfaccia per la lettura e la scrittura dei file in formati esterni. I ReaderWriter per i singoli formati, il cui nome, per convenzione, viene definito come ReaderWriter a cui viene concatenata l'estensione (es. ReaderWriterIVE, ReaderWriterFLT, ReaderWriterGIF ecc.), sono quindi sottoclassi che ereditano dalla classe virtuale nelle quali

vengono specificate le opportune implementazioni. Queste sotto classi diventeranno, in fase di compilazione, i Plug-In di OpenSceneGraph che veranno caricati all'occorrenza durante una chiamata di loading.

- *Registry:* questa classe è una *singleton factory*, ossia una classe che genera istanze uniche, utilizzata per istanziare e registrare a tempo di esecuzione i singoli ReaderWriter.

- *DatabasePager:* la classe DatabasePager si occupa del caricamento dei file in background, e aggiorna e sincronizza i modelli caricati con lo SceneGraph.

6.2.3 Abilitazione del DatabasePager

Come ultimo ritocco al tool di visualizzazione si è intervenuti nella sezione di renderizzazione della scena abilitando i *Thread* di gestione del meccanismo di paginazione ed applicando il modello appena caricato nella lista di aggiornamento del DatabasePager.
Come prerequisito per il corretto funzionamento del DatabasePager vi è l'instanziamento di un Timer per la gestione dei nodi *expired*[5] ed una variabile di tipo persistente per il conteggio dei frame trascorsi durante due o più aggiornamenti.
Come esplicitato nel codice del motore di visualizzazione di OSG (più precisamente all'interno della libreria osgProducer) la fase di aggiornamento del DatabasePager deve partire precedentemente alla fase di *Culling* della scena e subito successivamente alla fase di *Draw*.
Nella fase precedente alla sezione di Culling viene settato il *FrameStamp* corrente, generato dal timer di sistema, sulla struttura dello SceneGraph e sui Thread di paginazione.
La fase successiva alla renderizzazione della scena, invece, viene utilizzata per abilitare il DatabasePager ad aggiornare lo SceneGraph corrente. L'aggiornamento dello SceneGraph deve sottostare ad alcuni accorgimenti di tempistica

[5] nodi di paginazione che possono essere eliminati dallo SceneGraph in quanto non visualizzati per un tempo lungo. In questo modo viene applicata una politica di ottimizzazione e di gestione delle risorse.

per non cadere in problemi di freeze durante la navigazione della scena[6]. Ovviamente sono diversi i parametri che vanno a influire su questo dato. Prima di tutto è la mole di dati nello SceneGraph ed in secondo luogo troviamo l'aggiornamento esterno attuato, in questo caso, dal DatabasePager sull'albero di scena. Una volta che il DatabasePager è riuscito a caricare delle strutture, l'aggiornamento dello SceneGraph viene fatto solo se vi è ancora del tempo disponibile nel tempo stimato dalle procedure per far si che il frame rate rimanga sopra i 30fps. In caso negativo questa fase di Update viene saltata e viene incrementata una variabile sussidiaria che indica le passate di renderizzazione senza aggiornamento esterno. Quando questo valore supera una determinata soglia l'aggiornamento viene forzato a discapito delle prestazioni.

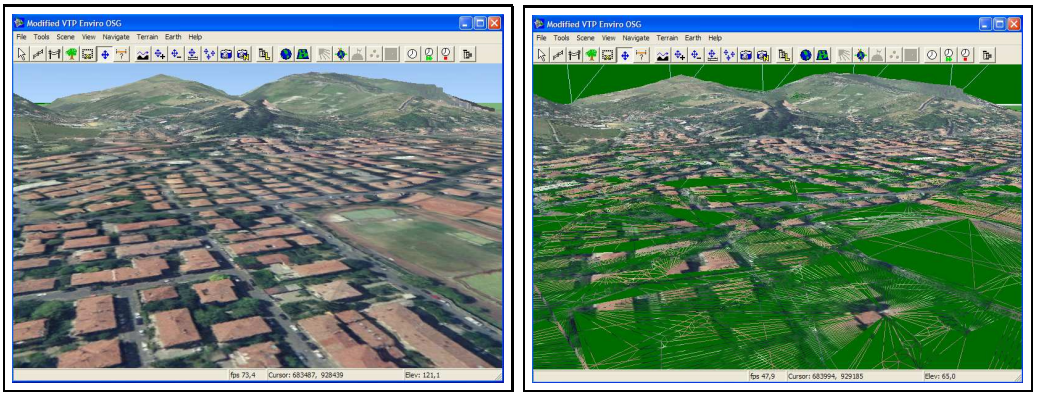


Figura 6.3: Terreno esterno importato in Enviro modificato. *A sinistra:* Visualizione classica. A destra: Visualizzazione in WireFrame.

6.3 Esportazione delle Culture

Con il termine *Culture* si intendono tutti quegli elementi, indipendenti dal terreno, che vanno ad incrementare il potere espressivo della scena all'interno di Enviro. Partendo solitamente da informazioni di tipo GIS, Enviro crea delle geometrie aggiuntive al paesaggio definendo, in questo modo, una scena più gradevole e reale.
Lasciando inalterate le parti di navigazione e di interazione con il paesaggio,

[6]il frame rate per second dovrebbe mantenersi al di sopra dei 30fps (60fps se la visualizzazione è stereo)

in questa parte di lavoro si è intervenuti cercando di abilitare un metodo di salvataggio delle strutture generate e modificate dal tool in una forma riutilizzabile anche da altre applicazioni.
Enviro, nella versione di Release, permette il salvataggio delle modifiche apportate alla scena solo all'interno dei file di settaggio, specificati nella fase di generazione del progetto di visulizzazione, che vengono caricati all'avvio per generare la scena.

Il lavoro di aggiunta delle funzioni di esportazione delle strutture è partito

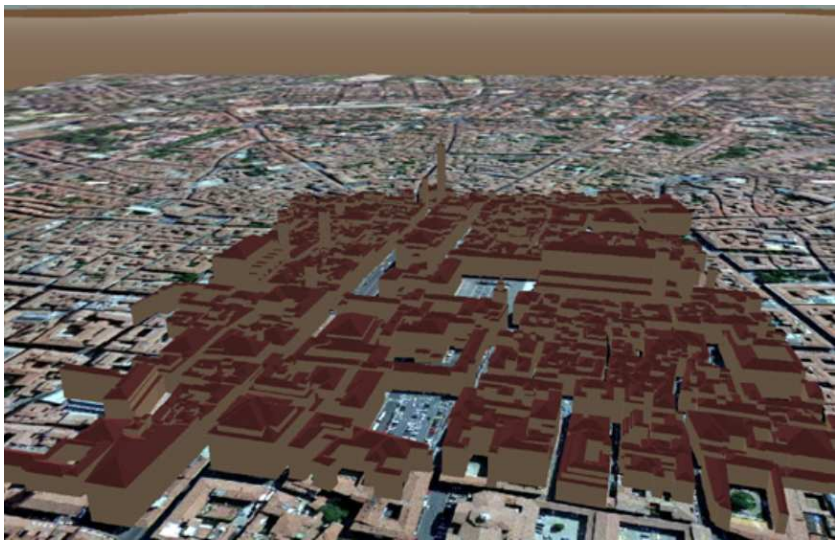


Figura 6.4: Esempio di generazione di strutture in Enviro utilizzando file contenenti dati vettoriale.

notando, nell'interfaccia a finestre di Enviro, un voce di salvataggio della scena OSG implementata solo superficialmente. Visionando il codice si è scoperto che questa funzione si limita solo a salvare tutto l'albero di scena attivo in un determinato momento durante la navigazione. Questo approccio è purtroppo fallimentare in quanto si viene a creare una struttura enorme contenente il terreno e tutte le strutture sussidiarie non fondamentali (come ad esempio, la mesh del terreno resa statica durante il *frame* di salvataggio, le geometrie del sole e degli orizzonti, ecc.) senza preoccuparsi del salvataggio delle texture, o dei comportamenti, e degli stati di alcuni oggetti particolari (come le label). A salvataggio terminato, quindi, si ottiene un file di dimesioni notevoli contenete il *freeze* poligonale della scena.
Da questa nota si evince che un interesse da parte della comunità di sviluppo

c'è nel salvataggio delle strutture in formati non proprietari al programma. Durante la seconda fase di questo lavoro di tesi si è cercato di determinare un metodo più strutturato e performante nell'esportazione di parti della scena cercando di produrre file di geometrie (e texture) includendo i comportamenti di determinati oggetti, riutilizzabili con programmi di terze parti. Inoltre, un ulteriore obiettivo, è stato quello di cercare di definire dei file con delle peculiarità e delle dimensioni adatte ad una fruizione in remoto delle strutture generate.

Figura 6.5: Esempi di Culture caricati durante una sessione di visualizzazione in Enviro. *A sinistra:* Visualizione di un progetto con terreno esterno generato da immagini satellitari. A destra: Visualizzazione con terreno importato generato dalla cartografia catastale.

Creazione della maschera di esportazione

Per una gestione intuitiva delle strutture da salvare il primo passo è stato quello di generare una finestra di salvaggio adeguata. Per mantenere coerenza con la struttura di progetto di Enviro la finestra di gestione del salvataggio è stata implementata tramite la libreria *vtui* facendogli derivare la classe AutoDialog. La struttura della maschera è suddivisa nel seguente modo:

- *Selezione Culture da salvare:* In questa sezione è stato reso disponibile la selezione dei tipi di strutture da salvare in base ad una serie

di CheckBox. La selezione di alcune particolari Culture abilitano dei parametri di salvataggio nella sezione Proprietà di salvataggio;

- *Gestione basilare dei file:* In questa sezione viene richiesta la definizione di un suffisso al nome dei file che verranno prodotti durante la fase di salvataggio e verrà richiesta la directory nella quale verranno salvati i file. La selezione della directory può essere fatta anche tramite una speciale finestra di dialog;

- *Gestione proprietà di salvataggio:* In questa sezione sono presenti tutte le opzioni di salvataggio. La definizione delle opzioni ha avuto una evoluzione graduale durante la fase di sviluppo in quanto l'aggiunta di una feature comportava anche la definizione di una specifica maschera di selezione all'interno del dialog di salvataggio. In questa sezione è possibile:

  - Selezionare il tipo di formato nel quale i file vengono salvati (nello specifico .osg e .ive);
  - Ricomprimere le texture applicate ai modelli;
  - Convertire le coordinate dei vertici e la mappatura della texture del terreno sulle strutture;
  - Abilitare il blending delle texture terreno applicate alle strutture con il materiale della struttura;
  - Salvare le texture dei modelli in file separati;
  - Decidere il metodo di salvataggio (file suddivisi per tipi di Culture, suddivisione delle strutture in griglie di files, salvataggio delle Culture all'interno della gerarchia di paginazione del terreno visualizzato);
  - Definire le dimensioni della griglia statica di salvataggio;

6.3.1 Punti di reperimento dei dati

Tutto ciò che viene visualizzato durante una fase di renderizzazione ha una propria struttura formale definita dall'albero di scena. L'albero è costruito in modo tale che ogni nodo presenti una propria struttura e sia distinguibile

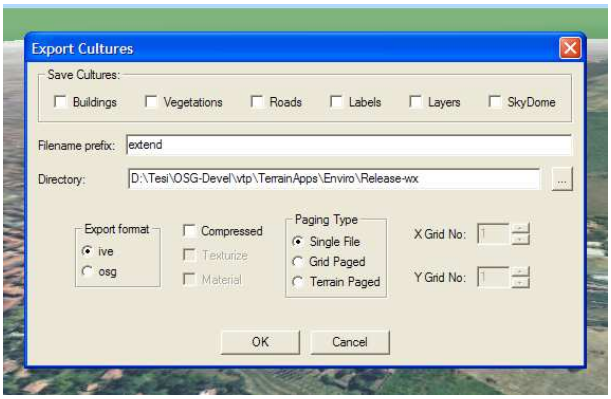


Figura 6.6: Finestra di esportazione strutture da Enviro.

dagli altri; questa caratteristica fa sì che ogni singola geometria sia reperibile, nell'albero di scena, e, soprattutto, sia salvabile. Questa è stata l'intuizione portante per adattare Enviro all'esportazione delle Culture.
Ogni struttura viene generata, dopo una serie di passi e raffinamenti, e collocata in classi che ne mantengono l'informazione. Ogni classe viene poi collocata, a sua volta, in una griglia ideale, di suddivisione del terreno, che tiene traccia del posizionamento della geometria sottostante e di altri parametri dipendenti dall'oggetto (in particolar modo, della distanza di Level Of Detail[7] dall'osservatore). Queste strutture fanno parte della libreria *vtlib*; ogni classe di questa libreria è stata pensata in modo tale da derivare le classi base della libreria di SceneGraph sottostante. In questo modo si viene a ricreare l'albero di scena senza toccare direttamente le funzioni di libreria ma utilizzando solo le chiamate di gestione delle griglie utilizzate.
Le strutture contenitore delle geometrie, definite esportabili, sono state rilevate all'interno del codice e sono state rese visibili alla classe che si occupa del salvataggio. Nella versione attuale di Enviro modificato sono state definite come salvabili le strutture contenenti: le coperture di vegetazione, i layer vettoriali generati dall'importazione di label geometriche, le label sul terreno, la copertura di strade, la volta celeste ed infine le geometrie delle case[8] e dei modelli 3D importati nella scena.

[7]distanza minima che definisce l'inserimento del modello nella renderizzazione della scena

[8]Le geometrie delle case vengono processate, in una fase iniziale, da vtBuilder per generare un file di vertici (scritto in XML) che Enviro caricherà per generare i poligoni

Precedentemente alla fase di salvataggio vera e propria ogni struttura viene identificata tramite un enumeratore e di ogni oggetto preso in considerazione viene estratta la struttura a più basso livello di OSG. A questo punto il processo di salvataggio può avere inizio ed il metodo vtTerrain::SavingStructuresByType ne gestisce tutto il lavoro.

6.3.2 Metodi di Salvataggio

Lo sviluppo della feature di salvataggio delle strutture ha seguito un corso di tipo incrementale per quanto riguarda la ricerca di metodologie di salvataggio per una fruizione da remoto accettabile. Questo sviluppo è stato guidato anche dalla risoluzione di diverse problematiche insorte durante il riutilizzo delle strutture generate.
Come primo step, durante la fase di salvataggio, la struttura ad albero (contenente le strutture da salvare) viene clonata nella sua integralità in modo tale da poter essere gestita e manipolata all'occorrenza senza preoccuparsi di invalidare l'integralità dell'albero utilizzato per la renderizzazione. Alla stessa maniera viene clonato anche il nodo root del terreno esterno ma con la differenza che la copia sarà effettuata in maniera blanda senza propagazione della clonazione attraverso i nodi figli. Questa decisione è stata presa in quanto questo nodo è utilizzato per determinare l'estensione effettiva del terreno caricato (che può essere differente dall'HeightField), nel metodo di salvataggio High-Tile (Sezione 6.3.2) e per generare il file Archivio (capitolo 6.3.4).
Successivamente viene ricreata una matrice di proiezione delle strutture da coordinate locali dell'Environment alle coordinate spaziali nelle quali vive il terreno esterno. Per scoprire questa matrice è stato sufficiente prendere la trasformazione, generata durante la fase di Importazione del terreno, e produrre la matrice inversa. In questo modo se la proiezione iniziale è corretta anche la proiezione opposta lo è.
In base ai parametri di salvataggio è possibile, durante una fase di preprocesso, salvare le texture in file separati, nel caso di un salvataggio su file con formato .osg (WriteImageFilesVisitor sez. 6.3.3), o cercare di comprimere le texture in modo tale che, se la scelta del formato da salvare è di tipo .ive, la

degli edifici. VtBuilder è stato modificato per generare delle geometrie più leggeri rispetto a quelle generate nella versione di Release.

dimensione dei file generati cali notevolmente (CompressTexturesVisitor sez. 6.3.3);
Tuttora le possibilità di salvataggio sono di tre tipi:

Salvataggio su file divisi per strutture: Questo è stato il primo metodo di salvataggio sviluppato. La sua implementazione è decisamente più semplice rispetto alle successive ma è stata di fondamentale importanza per attuare i test sui dati salvati e per capire come poter intervenire sulle strutture.
Questo metodo si fa carico, solamente, di prendere le strutture da salvare ed applicargli la matrice di riproiezione.
Come ogni metodo proposto in questo capitolo il salvataggio di file suddivisi in base al tipo di struttura presenta delle particolarità apprezzabili ma anche dei limiti che possono essere più o meno significativi.
Come parametro positivo nell'applicazione di questo tipo di salvataggio si rinviene la generazione di file contenenti dati suddivisi in base alla tipologia; infatti, la struttura salvata risulta essere un albero contenente solo gerarchie di un determinato tipo. La lettura, in contemporanea, dei file prodotti in questo modo ed il terreno di riferimento, col quale sono stati generati, produce una esatta copia del terreno visualizzato all'interno di Enviro. Inoltre, suddividendo la scena in file con dati monolitici, è possibile pensare a dei programmi *batch* che, prendendo come input questi file, possono attuare delle modifiche, generare delle suddivisioni o produrre delle viste a sezioni di queste strutture.
Come punto a sfavore di questa metodologia si delinea il fatto che più i dati di partenza sono grandi e più i file risultanti lo saranno. Una lettura di questi file con un metodo di paginazione (durante la renderizzazione) è praticamente impensabile in quanto la priorità data ai Thread di lettura dei file, in questa fase, è estremamente bassa rispetto ad una lettura dei dati precedentemente alla renderizzazione; essendo già la lettura in locale difficoltosa risulta essere improbabile una fruizione diretta a questi file in remoto durante una renderizzazione.
Le strutture di tipo Label, ed i Layer vettoriali, vengono salvate solo attraverso questa metodologia in quanto è utile tenerle separate dalle altre, ed anche, perchè il loro livello di visibilità non deve dipendere dal terreno in quanto definiscono solo informazioni di supporto al terreno

che possono essere omesse a piacere.

Salvataggio con paginazione a griglia: Questo approccio si basa sulla suddivisione spaziale del terreno in una griglia bidimensionale di dimensione scelta dall'utente; ad ogni cella della griglia viene associato un file di salvataggio nel quale verranno incluse tutte le strutture con il proprio baricentro all'interno della sezione spaziale che la cella definisce (vedi WritePagedFilesVisitor sez. 6.3.3).

Questo metodo risulta essere un metodo poco performante per la strutturazione dei file ma è stato un primo test di utilizzo dei nodi di Paging[9]. I parametri che influiscono sulle performance di questi file risultano essere: le dimensioni della griglia (specularmente, il numero dei file generato), la quantità e la pesantezza delle strutture da salvare, la dimensione del terreno e quindi la dimensione spaziale che una cella occupa sul terreno.

Se il numero di file diventa troppo grande (per esempio una griglia $20 * 20$ produrrà 400 file se vi è almeno una struttura all'interno di una cella) il DatabasePager risulterà intasato di richieste di caricamento di file che non riuscirà ad evadere durante le fasi di Frame. Come conseguenza di questo fatto, quando scadranno le *dead-line* di richieste di file *non-expired*, saranno possibili fastidiosi rallentamenti del *frame-rate*.

Allo stesso modo, se ricadono troppe strutture in una determinata cella le dimensione dei file risultano essere troppo grandi per un caricamento fluido[10].

Salvataggio sulla High-Tile del terreno: Il salvataggio sulla High-Tile consiste nel riuscire, per ogni struttura da salvare, ad individuare, percorrendo la gerarchia di paginazione del terreno esterno, la Tile a più alto livello di dettaglio che la contiene.

Questa metodo è stato il più complicato nello sviluppo ed è molto di-

[9] Nodi speciali che mantengono i propri figli all'interno di file salvati esternamente. Durante la fase di Update di renderizzazione, se il punto di vista entra all'interno del raggio di LOD, invocano il DatabasePager per richiamare il file con il figlio da visualizzare.

[10] Su una architettura Pentium-4, 1GB Ram, Hdd 7200rpm, scheda video nvidia 5700pro, suddivisione del terreno $5 * 5$, osgviewer con settaggio delle priorità dei Thread di paginazione standard e dimensione dei file prodotti di al massimo 10mb il caricamento di un file viene evaso in un periodo tra i 25-45 secondi.

spendioso a livello di risorse, ma l'effetto finale risulta essere gradevole e la gestione della paginazione, sfruttando la paginazione del terreno iniziale già ottimizzata, è discreta (ovviamente cercando di non inserire modelli di dimensioni notevoli) per la lettura dei dati da remoto.
La tecnica implementata definisce una serie di NodeVisitor (vedere sezione 6.3.3) capaci, oltre che individuare la posizione di un oggetto all'interno di una Tile specifica (ed ad inserirne il nodo come figlio), di rimodellare le geometrie per aggiungerle all'interno dei nodi Geode[11] del modello del terreno. Questa feature è fondamentale per riuscire a riutilizzare gli StateSet del terreno ed a mappare le texture del terreno sui modelli delle abitazioni.

6.3.3 Interventi sugli alberi di scena: I NodeVisitor

OSG NodeVisitor è una classe appartente al core di OpenSceneGraph. Questa classe applica operazioni *safe*[12] all'albero di scena soffermadosi sui nodi richiesti durante la stessa fase di attraversamento. L'implementazione del NodeVisitor è basata sul GOF Visitor pattern [8] ed implementa l'algoritmo di *Double Dispatch* per definire il metodo di attraversamento da chiamare in base al tipo di nodo processato.
Derivando dalla classe NodeVisitor, una serie di visitatori specializzati sono stati creati in modo tale attraversare e modellare gli alberi OSG per gestire una corretta esportazione:

CompressTexturesVisitor: Questo NodeVisitor viene applicato durante l'esportazione delle strutture di tipo Building, Vegetation e Road se la scelta dei file di salvataggio ricade sul formato .ive.
CompressTextures visitor cerca tutti i nodi di tipo Geode e ne isola i vari StateSet. Una volta rilevati tutti gli StateSet ne ricava le texture risettandone il formato interno e demandando al driver OpenGL la compressione;

[11] Nodo OSG contente le geometrie, le coordinate texture, e le normali di un oggetto poligonale.

[12] Operazioni sui nodi in base alla specifica del tipo. Questa metodologia svincola lo sviluppatore a preoccuparsi dell'attraversamento dell'albero di scena in modo tale da definire solo l'operazione da applicare.

WriteImageFilesVisitor: WriteImageFiles visitor, allo stesso modo del visitor precedente, attraversa tutto l'albero di struttura e ne ricava gli StateSet. Da ogni StateSet viene ricavata l'eventuale texture e viene salvata su file in base al nome originale (contenuta come informazione aggiuntiva nella struttura della texture).
Questo NodeVisitor è applicato solo durante il salvataggio dei file esportati in formato .osg in quanto, questo formato, è di tipo ASCII e quindi le texture saranno esterne;

BuildBillBoardsVisitor: Durante la generazione dei file esterni si è cercato di mantenere alcuni comportamenti che le strutture possono, eventualmente, presentare durante una visualizzazione in Enviro. Nel caso delle Label i nodi generati nella fase di inizializzazione risultano essere dei semplici nodi Text; ogni nodo Text presenta una matrice di trasformazione che viene aggiornata, durante la fase di Frame, da uno specifico Engine che ne ruota la geometria; grazie a questa rotazione, ogni label, risulta sempre visibile all'osservatore.
Per riuscire a salvare questo effetto si è fatto uso dei nodi speciali di tipo BillBoard[13]. BuildBillBoards visitor cerca nell'albero di nodi Label tutte le strutture di tipo Geode e le rimpiazza creando dei nodi BillBoard con asse di rotazione definito dall'asse Y. Inoltre questo visitor crea i nodi Text, contenenti le informazioni sul font da utilizzare e la stringa di testo, che verranno applicati come figli al nodo BillBoard.

BuildProxyNodeVisitor: BuildProxyNode visitor è stato creato per cercare di ottimizzare le strutture degli alberi che fanno parte della copertura di vegetazione. Questo meccanismo è essenziale per ridurre le dimensioni dei file generati in quanto ogni singola istanza di un albero, durante il salvataggio diretto dell'albero, genererebbe della geometria aggiuntiva all'interno dei file e le texture verrebbero replicate inutilmente.
Come il precedente, anche questo visitor applica all'albero visitato una operazione di rimpiazzamento in modo da sostituire i nodi di tipo Geode con nodi di tipo ProxyNode[14].

[13]Nodi che, durante la fase di Update dell'albero di scena, vengono ruotati automaticamente in base ad un asse definito durante la creazione.

[14]Nodi di paginazione che richiamano i propri figli da file esterni non appena vengono

Utilizzando una struttura di cache questo NodeVisitor definisce se un Geode è stato replicato nella scena oppure se la sua struttura è differente da quelli già presenti. Se la geometria non risulta in cache viene salvata in un file satellite ed il nome del file, assieme alla geometria (utilizzata per il confronto), vengono inseriti in cache. Successivamente il Geode, nell'albero, viene rimpiazzato da un ProxyNode che richiama il file appena salvato. Al contrario, se un nodo è già presente in cache il NodeVisitor ha solo il compito di rimpiazzare il Geode con un ProxyNode che richiamerà il file contente la geometria corretta.

WritePagedFilesVisitor: WritePagedFiles visitor è il primo NodeVisitor, sviluppato in questo lavoro di tesi, che cerca di generare delle strutture di paginazione. WritePagedFiles visitor viene utilizzato durante il metodo di salvataggio a griglia statica e permette anche il suo concatenamento con i NodeVisitor visti in precedenza.
I nodi di struttura coinvolti dal suo intervento sono i nodi di tipo LOD nei quali sono contenute le geometrie create da Enviro. Ogni figlio del nodo LOD, estratto dall'albero, verrà aggiunto in un vettore di nodi Group (idealmente il vettore rappresenta la griglia di suddivisione del terreno). L'indice di posizionamento sul vettore viene ricavato in base alla posizione spaziale definita dal calcolo del baricentro della BoundingBox del nodo LOD da aggiungere.
A visita completata ogni elemento del vettore, se contiene dei figli, verrà salvato in un file e, per ogni file salvato, sarà creato un nodo PagedLOD settando la distanza di caricamento pari alla distanza di LOD. La distanza di LOD viene presa dai settaggi del menù *Camera View* di Enviro. Infine, i nodi PagedLOD verranno aggiunti ad un nodo MatrixTransform, contenente la proiezione in coordinate georeferenziate, e salvata successivamente in un file archivio (sezione 6.3.4).

DeStructLODVistor: Questo visitor, assieme ai NodeVisitor successivi, si occupa di inserire le strutture all'interno delle Tile, a più alto livello di dettaglio, nel quale è suddiviso il terreno importato. Questo NodeVisitor viene richiamato solo tramite la scelta di salvataggio con metodo

attraversati dal NodeVisitor di Update. A differenza dei nodi PagedLOD non è possibile specificare la RangeList utilizzata per definire la distanza di Level Of Detail.

di tipo High-Tile.
Il compito principale che svolge consiste nel reperimento di tutti i nodi di tipo LOD dell'albero di scena passatogli e del richiamo, per ogni struttura individuata, del visitor GetLowerTile che si occupa dell'inserimento nella Tile.
Se la copertura da esportare è di tipo costruzioni (ovvero tutte le geometrie di palazzi generati in automatico da Enviro) e, se la scelta di salvataggio comprende anche la texturizzazione delle geometrie delle case con la texture del terreno, un ulteriore NodeVisitor viene richiamato; il FindGeode visitor. Compiuto il proprio lavoro il FindGeode visitor restituisce un vettore di nodi che saranno passati, uno alla volta, al GetLowerTile visitor.

FindGeodeVisitor: Sul sotto albero ricavato dal visitor DeStructLOD, il NodeVisitor FindGeode cerca tutte le strutture di tipo Geode e la matrice di trasformazione applicata al nodo.
Il compito che svolge consiste nello smembrare il nodo Geode ricavandone tutti i pacchetti di geometria definiti nei nodi Geometry figli di Geode. Ad ogni geometria verrà riapplicata la matrice di trasformazione iniziale e la matrice verrà inserita in un vettore per essere poi passata al visitor GetLowerTile che si occuperà dell'inserimento nella Tile a più basso livello.
Lo smembramento del nodo LOD, passato a questo visitor, è stato forzato dall'implementazione del codice che cerca di convertire tutte le coordinate dei poligoni nel sistema di riferimento della Tile dove verranno incluse. Questa riproiezione è essenziale per riuscire a generare delle coordinate texture corrette per mappare la texture del terreno sulle geometrie sussidiarie da inserire.

GetLowerTileVistor: GetLowerTile vistor è il cuore del metodo di salvataggio High-Tile.
A differenza di tutti i visitor visti in precedenza GetLowerTile viene applicato al nodo del terreno cercado i PagedLOD di cui è composto. Tramite l'utilizzo dell'IntersectVisitor[15] viene definito il PagedLOD che

[15] L'IntersectVisitor attraversa l'albero di scena, sul quale viene applicato, e identifica eventuali collisioni con un segmento definito durante la fase di instanziamento. Tramite

contiene la geometria da inserire e viene caricato il figlio contenente la risoluzione più elevata. In maniera ricorsiva viene instanziato un nuovo GetLowerTile visitor passandogli tutte le strutture utili per l'inserimento. Quando la Tile foglia viene scovata il processo di ricorsione si ferma ed il penultimo visitor instanziato si occuperà dell'inserimento.
L'inserimento della geometria nel terreno può avvenire in due modi:

- *Senza texturizzazione:* in questo caso la geometria viene inserita come figlia di un nodo di trasformazione che definisce la proiezione di georeferenziazione del terreno. La matrice di trasformazione e la Tile ricavata vengono inserite in una classe apposita utilizzata per: lo stoccaggio delle strutture fino alla fase di salvataggio in un file e per applicare un meccanismo di cache in modo tale da non dover riapplicare l'algoritmo di ritrovamento delle *High Tile* già individuate;
- *Con texturizzazione:* questo caso risulta molto simile al precendente; in aggiunta, questa metodologia, prevede che ogni nodo incluso attraversi una ulteriore fase di manipolazione. Innanzitutto, tramite l'IntersectVisitor che ha determinato la collisione della geometria con la Tile (o una sezione di essa), viene isolato il Geode del terreno e ne viene calcolata la BoundingBox. Successivamente viene calcolata la matrice di trasformazione che, dal sistema di riferimento della geometria, porta i vertici nel sistema di riferimento del terreno. La formula di conversione è definita da:

$$M_{Trasformazione} = L * G * T_l^{-1} \tag{6.3.1}$$

dove:

  - L è la matrice di trasformazione delle coordinate locali della geometria da inserire nel sistema di riferimento utilizzato in Enviro;
  - G è la matrice generale di riproiezione delle strutture nella georeferenziazione del terreno;

l'IntersectVisitor è possibile ricavare la geometria di collisione direttamente.

– T_l^{-1} è la matrice che identifica la trasformazione inversa di riproiezione dalla georeferenziazione del terreno alle coordinate locali di OSG;

Come già menzionato in diverse parti di questo capitolo, la matrice di proiezione, per riportare le geometrie nel sistema di riferimento di OSG, presenta una rotazione di conversione al sistema di riferimento Y up. In questa fase la rotazione va esplicitata direttamente sulle normali dei vertici altrimenti la rifrazione della luce risulterebbe errata. Quindi la Z, della normale, viene invertita con la Y.
Gli ultimi due passi, applicati dal GetLowerTile visitor alle geometrie, risultano essere: la ricostruzione delle coordinate texture sul modello (tramite una semplice funzione di proporzione tra la coordinata del vertice e la dimensione del terreno) e l'attivazione dell'effetto di Blending tra l'attuale materiale dell'oggetto e la texture del terreno applicata (attivabile tramite l'attivazione di un parametro nello StateSet del modello).
Ora, la geometria del modello ricostruita viene inserita tra le geometrie della Tile.

Come precedentemente citato, per evitare una ricerca successiva di una Tile già individuata, si è implementata una struttura di cache. La cache viene acceduta dal GetLowerTile visitor precedentemente la fase di ricorsione con una conseguente diminuzione del tempo di calcolo.
La struttura di cache, alla fine del processo, verrà passata ai successivi DeStructLOD vistor (ed alla funzione che si occuperà di salvare le Tile modificate) in modo tale da supportare l'inserimento di tutti i tipi di strutture all'interno delle stesse Tile del terreno.

L'oggetto vtLodGrid StructGrid

Un ulteriore intervento significativo al codice di Enviro è stato applicato replicando l'oggetto, richiamato dalla classe vtTerrain, di tipo vtLodGrid (appartenente alla libreria vtlib). Nella fase di inizializzazione della scena è stata creata una ulteriore classe vtLodGrid che andrà a contenere le strutture di tipo abitazione generate in automatico.

La classe vtLodGrid, nella versione di Release, si occupa di mantenere tutte le strutture di tipo abitazione (generate nella fase di inizializzazione), i modelli esterni e le fance procedurali in una griglia che ne definisce il livello di dettaglio.
Con l'abilitazione del programma ad un salvataggio con texturizzazione delle case è stato necessario riadattare le strutture di contenimento della classe vtTerrain in modo tale da riuscire a distinguere strutture generate dinamicamente da coperture vettoriali e strutture che devono mantenere le proprie caratteristiche.
Per fare questo si è deciso di aggiungere un ulteriore struttura vtLodGrid. Durante la fase di importazione (o di inserimento dinamico nella visualizzazione) i modelli da inserire vengono aggiunti nella struttura di supporto chiamata StructGridImported invece che nell'oggetto StructGrid dove saranno inserite solo le strutture generate da coperture.
Per mantenere coerenza, con le funzioni di modifica alle impostazioni degli

Figura 6.7: Esportazione di fotomodelli e vegetazione all'interno di un terreno paginato.

alberi di scena e nei salvataggi dei file di configurazione di Enviro, sono state replicate anche le chiamate alle strutture mantenendo inalterate le maschere di selezione dei parametri (ad esempio la modifica della distanza di vista

alla struttura StructGrid comporterà anche la modifica parallella alla classe gemella).
Durante la fase di salvataggio delle strutture, tramite i file proprietari di Enviro, si è deciso, anche in questo caso, di mantenere separate le due strutture generando due file di configurazione indipendenti.

6.3.4 Accorgimenti finali

Successivamente alla applicazione dei vari NodeVisitor agli alberi di scena delle strutture da salvare, si è implemento uno SceneGraph di supporto per riuscire a richiamare tutti i file generati durante il salvataggio. La struttura archivio è semplicemente un nodo gruppo che presenta dei figli di tipo PagedLOD (creati dai NodeVisitor di salvataggio o, in questa fase, se il metodo di salvataggio scelto è di tipo File Singolo) e verrà salvata con il nome di *vtArchieve*.

Come precedentemente descritto, il compito di ogni PagedLOD è quello di

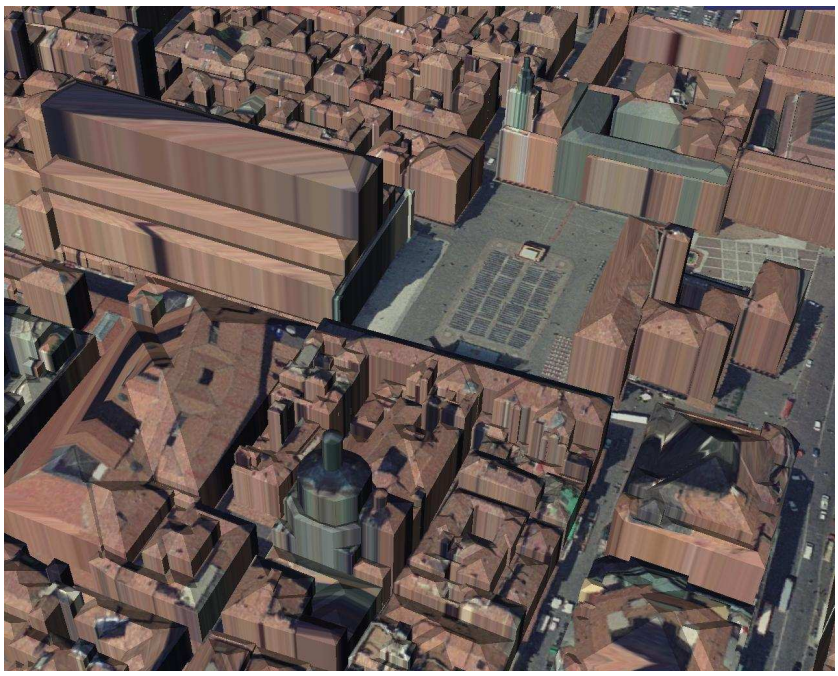


Figura 6.8: Vista su Piazza Maggiore. Risultato dell'esportazione sul terreno delle costruzioni texturizzate.

richiamare i file generati ed il master file del terreno importato in modo tale

da ricostruire una visualizzazione simile a quella proposta da Enviro.
La generazione del vtArchieve in questo modo (ovvero un file che può non contere geometria al suo interno) produce un effetto indesiderato; se viene caricato da un player standard di OSG provoca confusione e la definizione del punto di vista dell'osservatore viene settata nel punto di default del Viewer ovvero il punto origine del sistema di riferimento.
In questo modo viene visualizzata una scena vuota in quanto il modello si troverà a delle coordinate completamente differenti rispetto al punto di origine del sistema di riferimento con il successivo problema del ritrovamanto del modello nella scena. Questa malfunzione viene accentuata anche dal fatto che, i terreni e le strutture, vivono in sistemi di riferimento con georeferenziazione; se, ad esempio, considerando un terreno con georeferenziazione di tipo UTM standard, le coordinate dei vertici avranno valori dell'oridine di 10^5 per la X e 10^6 per la Y.
Per ovviare a questo inconveninte, nel file archivio, è stata aggiunta una geometria semplice di supporto con applicata la georeferenzizione iniziale. In base agli extent del terreno è stato generato un piano ed è stato inserito nella scena al di sotto del terreno. In questo modo la camera viene subito posizionata correttamente di fronte alle geometrie da caricare.
All'interno del file archivio, inoltre, è stata inserita una ulteriore struttura; lo SkyDome. Questa struttura viene creata da Enviro durante la fase di inizializzazione in base ad una texture, scelta dall'utente, e dalla geometria che rappresenta una semi sfera.

Lo SkyDome viene posizionato al di sopra del terreno in modo tale che risulti all'interno della semisfera creando l'effetto desiderato di volta celeste. A differenza della visulizzazione all'interno di Enviro, la soluzione proposta nel file archivio, non mantiene il comportamento di fissare il punto di vista dell'osservatore al centro della calotta e mantenere questa invariante anche durante il movimento della camera.
La scelta del salvataggio dello SkyDome può essere deciso tramite la maschera di salvataggio e omessa all'occorrenza.

6.3.5 OSG Active-X

Grazie ad uno studio di tesi di Marcello Morgotti, sempre presso il Consorzio CINECA, è stato sviluppato un Active-X in grado di caricare via Web

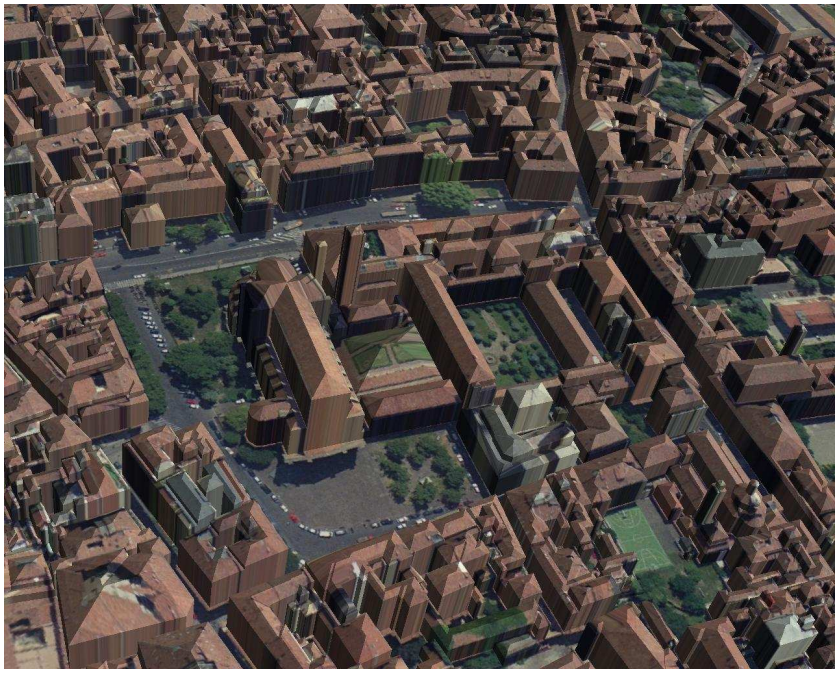


Figura 6.9: Vista su Piazza San Francesco.

modelli di territori in formato *.txp*. Successivamente, Silvano Imboden, ne ha modificato le potenzialità integrando all'interno dell'Active-X il player di default di OSG. Sfruttando, inoltre, le potenzialità del Plug-In *net* (incluso dalla versione 0.9.8 di OpenSceneGraph) è possibile, oltre che visualizzare modelli 3D all'interno di un browser da fonti locali, riuscire a caricare tali modelli da server remoti.

L'esportazione di Culture direttamente sul terreno paginato, come precedentemente descritto, è stata pensata proprio per riuscire a definire delle strutture in grado di essere caricati tramite questo Plug-In per browser.
Grazie all'intervento di Luigi Calori è stato integrato, inoltre, un linguaggio di scripting per il Web (PHP) ed un DataBase in modo tale da riuscire a comandare l'Active-X direttamente dalle pagine web. Lo sviluppo di questa idea ha permesso di riuscire a gestire la visualizzazione del modello inserendo dei punti di vista e permettendo il movimento della camera guidato da un punto di vista ad un altro con il semplice click del mouse.
Come implementazione futura vi è in progetto di riuscire a far colloquiare direttamente l'Active-X con delle informazioni aggiuntive della scena contenute all'interno del database. Questo processo implica un movimento del flusso

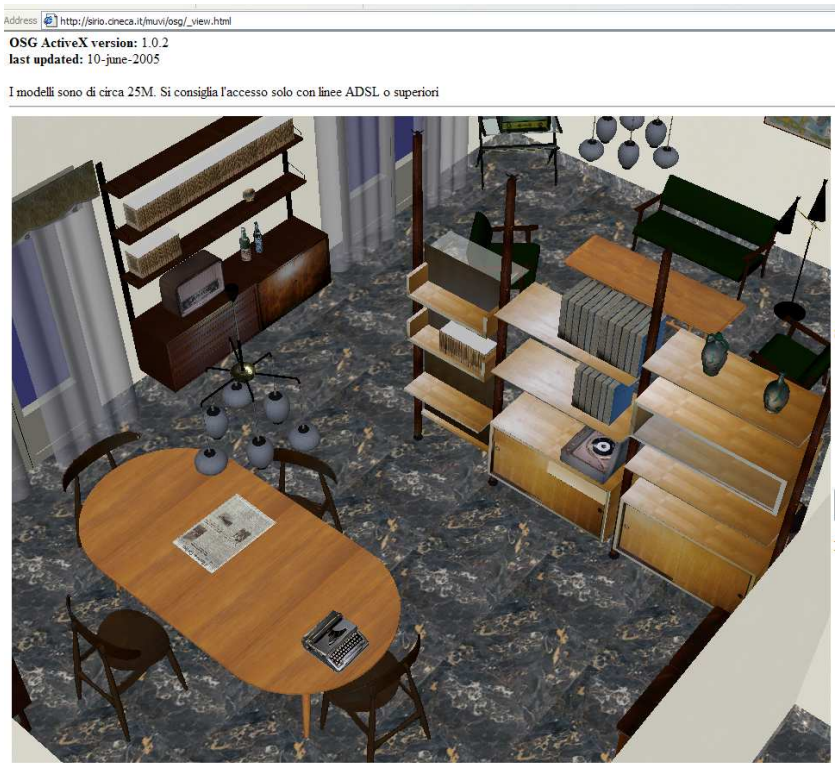


Figura 6.10: Visualizzazione del progetto MUVI[44] attraverso OsgActive-X.

dei dati in maniera opposta rispetto a quello sviluppato per la modifica dei punti di vista. Tramite un click nella scena, devono essere inviate delle query di interrogazione, dall'Active-X al DataBase, in modo tale da visualizzare delle informazioni aggiuntive all'interno della pagina Web nella quale risiede l'Active-X.

Capitolo 7

CINECA e CNR: Protocollo di generazione territoriale

In questo capitolo verrà analizzato il protocollo di lavoro utilizzato presso il laboratorio *VISIT* del consorzio CINECA e dal CNR-ITABC per la generazione di paesaggi tridimensionali visualizzabili in tempo reale in un ambiente di realtà virtuale.
Il protocollo di lavoro è stato consolidato dall'esperienza affinata durante tre anni di sperimentazioni e ricerca; la scoperta dei *Terrain Generator* da un lato, che consentono di mantenere uno stretto legame con la spazialità dei dati, e la costruzione, dall'altro lato, di un software appositamente creato per la visualizzazione e la interazione con tali dati (VISMAN par. 2.4.1), ha portato un notevole progresso nel metodo di creazione di applicazioni di Realtà Virtuale.
Nelle sezioni successive verrà esposto il metodo di lavoro consolidato presso i due enti e verrà, successivamente, ipotizzata una variante in grado di accettare il software modificato, oggetto di questa tesi di laurea, integrandolo nel processo di ricostruzione territoriale.

7.1 Processo di lavoro consolidato

Il Work-Flow è suddivisibile attraverso le seguenti fasi:

Studio dei Contenuti Questa fase prevede lo studio delle fonti (storiche,

archeologiche, cartografiche, ecc.) e dei dati a disposizione come input (carte, foto aeree, da satellite, dati GPS, dati Scanner Laser, ecc.);

Elaborazione di un Progetto GIS Questa fase ha il compito di importare e sovrapporre, secondo un unico sistema di coordinate geografiche, tutti i dati, precedentemente analizzati, all'interno di un progetto GIS. Le informazioni in questa maniera possono essere confrontate, analizzate ed elaborate in modo da ottenere nuovi dati e nuovi *layer informativi*. Ogni nuovo layer generato ha il compito di rappresentare un *tema* ovvero una rappresentazione specifica afferente ad un determinato luogo e momento storico (fiumi, strade, edifici e altre strutture, campi, ecc.);

Elaborazione di uno o più Progetti di Realtà Virtuale A seconda dei contenuti e degli obiettivi da raggiungere viene elaborato un progetto in grado di generare dei dati utilizzati dall'applicazione di visualizzazione;

Costruzione dell'applicazione di Realtà Virtuale Utilizzando un Environment di visualizzazione, come VirTools-Dev o VISMAN (Sezione 2.4.1), e applicando particolari specifiche ai dati creati viene creata l'applicazione di visualizzazione finale.

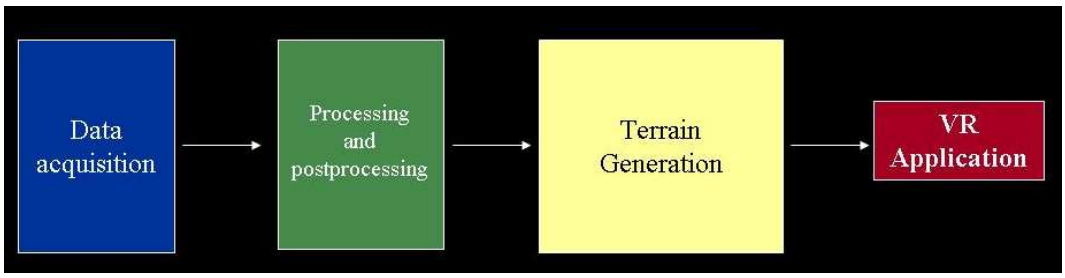


Figura 7.1: Metodo di lavoro per la generazione di applicazioni di Realtà Virtuale.

In seguito verranno analizzate più in dettaglio le parti di *Elaborazione dei Progetti di Realtà Virtuale* e *Costruzione dell'applicazione di Realtà Virtuale* in quanto, il lavoro di tesi proposto, può prenderne parte ed estenderne il funzionamento anche verso un settore meno esplorato della fruizione di scenari 3D da Web.
Per quanto riguarda la parte di *Elaborazione dei Progetti di Realtà Virtuale* è possibile raffinare la metodologia individuando i seguenti punti:

Generazione di Modelli 3d Utilizzando software di modellazione 3D, come Multigen Creator, 3DStudio Max, Blender o Photomodeler, vengono ricostruiti i modelli 3D da aggiungere nella scena. In questa fase il modellatore può lavorare indipendentemente, costruendo modelli tridimensionali a diversi Livelli di Dettaglio (a seconda della complessità degli oggetti), in scala metrica, orientamento nord-sud e con il centro della base del modello posizionata sull'origine degli assi del Sistema di Riferimento;

Creazione di Database In questa fase vengono ricreati i DataBase contenenti tutte le informazioni legate al ricostruzione da effetture. Il DataBase potrà essere interrogato all'interno della visualizzazione;

Creazione di Risorse Multimediali Con risorse multimediali si intendono tutti quei formati di dati contenenti informazioni sulla ricostruzione. Le informazioni avranno un carattere statico e potranno essere salvate in formati differenti come: schede HTML, file audio, file video, animazioni ed immagini;

Generazione del Paesaggio I software, utilizzati in questa fase di lavoro, prendono il nome di *Terrain Generator*. Il risultato che si ottiene è un modello tridimensionale del paesaggio in coordinate geografiche assolute. Per la ricostruzione del paesaggio devono essere preparati ed esportati dal GIS tre categorie di dati: il Digital Elevation Model, le GeoImage (ortofoto, satellitari o aeree, o semplici texture generiche) e dati Culture[1]. Una volta importati all'interno del Terrain Generator (al VISIT lab ed al CNR vengono utilizzati i software TerraVista e CTS) vengono assegnati attributi specifici per ciascuna Culture e vengono assegnati dei parametri al terreno per una resa finale adatta ad applicazioni Real-Time ovvero distanze di LOD, dimensioni delle Tile ecc. Ognuna di queste operazioni influenzerà la resa del territorio finale.

La sezione di *Costruzione dell'applicazione di Realtà Virtuale* viene caratterizzata dalla crezione del progetto globale di scena. In questa fase, utilizzando il visualizzatore VISMAN, viene ricreato l'applicativo finale. Il primo passo

[1]Le Culture esportate dal GIS sono file vettoriali (linee, punti, aree-poligoni) georiferiti, rappresentanti i caratteri naturali o antropici del paesaggio.

per ricreare la scena di visualizzazione è l'unione di tutti i modelli con il terreno. Successivamente ad ogni modello interattivo potranno essere inseriti dei comportementi ad esempio: il linking di un modello a contenuti esterni, lo switching tra modelli con differenti rappresentazioni, visualizzare in sequenze continue una serie di oggetti e varie altre opzioni.

Figura 7.2: Inserimento nel terreno di modelli esterni utilizzando il software di modellazione Multigen Creator.

7.2 Variazioni al Work-Flow

La suite Virtual Terrain modificata (Capitolo 7) può cercare di infiltrarsi nel processo di lavoro appena descritto. Attualmente il software è in grado di affiancarsi agli altri applicativi preesistenti ed, in alcuni casi particolari, addirittura di rimpiazzarli. Con ulteriori modifiche al pacchetto VTP, con l'utilizzo di software di modellazione differenti da quelli utilizzati al momento e con lo sviluppo di un viewer con capacità di interazione, è possibile pensare ad un protocollo di lavoro parallelo, a quello precedentemente esposto, in grado di utilizzare solo strumenti di tipo OpenSource.
Oggi, i punti di variazione al processo di lavoro stilato in questi anni di sperimentazioni può essere integrato con le seguenti modifiche:

- Enviro modificato ha la capacità di caricare terreni generati da programmi esterni alla suite VTP. Questo fatto ha permesso al Viewer di collocarsi come potenziale visualizzatore in scenari di riscostruzione;

- Utilizzando il tool di libreria osgdem di OpenSceneGraph (sez. 5.2.2), è possibile generare dei territori allo stesso modo dei Terrain Generator. Questo tool basilare è in grado solo di generare un territorio da DEM con l'inserimento di texture o ortofoto. OsgDem, nella versione attuale, non è ancora in grado accettare file di Culture quindi il risultato dell'applicativo è un terreno ben formato (con una suddivisione e dimensione adatta al Web) ma spoglio;

- Riuscendo ad importare terreni generati esternamente è possibile, inoltre, riuscire ad esportare tutte le Culture importate all'interno di Enviro. Se il terreno esterno caricato risulta essere stato creato da osgdem è possibile pensare ad una esportazione delle Culture all'interno delle Tile del terreno. In questo modo si viene a creare un terreno con caratteristiche simili a quelle di un Terrain Generator commerciale;

- Enviro possiede, come caratteristica di base, la possibilità di modificare la scena di visualizzazione interattivamente. Abilitandolo alla importazioni di file generati tramite Terrain Generator è possibile pensarlo come ad un viewer di territori e ad un modellatore dinamico della scena;

- La possibilità di generare territori con Culture inserite all'interno e con una suddivisione in grado di permettere una fruizione del territorio via Web ha permesso l'apertura di un nuovo filone di sviluppo legato alla visualizzazione remota di ambienti 3D.

Capitolo 8

Conclusioni e sviluppi futuri

Analizzando i risultati ottenuti durante questo lavoro di tesi e osservandone la concreta applicazione, è stato possibile dedurre che la realizzazione di un processo di sviluppo, in ambito applicativo GIS, utilizzando strumenti di tipo OpenSource è possibile.
Il problema fondamentale riscontrato, soprattutto nei primi mesi di lavoro, è riconducibile alla scarsa documentazione disponibile riguardo il codice dei tool di Virtual Terrain e sulla libreria OpenSceneGraph. Questa mancanza è stata comunque compensata dall'appoggio, in primo luogo, delle due comunità di sviluppo e, in secondo luogo, dalla presenza di un gran numero di esempi di codice al quale si è fatto riferimento. Altri problemi sono poi stati riscontrati solo in ambito applicativo, come già esaminato nella sezione di implementazione del codice.
Attualmente, il software modificato, presenta delle feature che comprendono:

- l'importazione di terreni generati da fonti estere. In questo modo è possibile una visualizzazione più reale degli scenari grazie alla complessità del terreno importato;

- la possibilità di esportazione, con differenti metodologie, di, modelli, vegetazione, strutture vettoriali, strutture di abitazioni ed etichette di testo, all'interno del terreno caricato. Questa potenzialità rende Enviro (il visualizzatore del pacchetto VTP) uno strumento in grado di essere utilizzato come Builder interattivo di una scena 3D creata partendo da dati di tipo GIS.

Come sviluppi futuri è possibile pensare a delle variazioni, a breve termine, della suite VTP in modo tale da applicare ai tool delle ulteriori capacità per avvicinarle il più possibile al progetto appena esposto. Queste modifiche sono riassunte nei seguenti punti:

vtBuilder Integrazione del tool GIS:

- Per una corretta integrazione dei territori a paginazione statica all'interno del visualizzatore Enviro sarebbe ideale l'integrazione della libreria osgTerrain[1] all'interno delle finestre di vtBuilder. In questo modo si creerebbe una interfaccia a finestre per la libreria (tuttora l'applicativo osgdem ne è sprovvisto) e utilizzando lo stesso Environment si riuscirebbe ad utilizzare gli stessi dati sia per la generazione del file di HeightField *.bt* sia per la generazione del territorio, evitando così problemi causati da errori durante il trattamento e la modifica di dati georeferenziati;
- Tuttora vtBuilder non permette il trattamento di coperture vettoriali contenti informazioni sui fiumi. L'idea di implementazione potrebbe ricalcare la soluzione già proposta dal *terrain generator* commerciale TerraVista di Terrex; ricreando sul DEM il letto del fiume (o del lago), potrebbe essere ricoperto da strip di poligoni contenti la texture dell'acqua, o, addirittura, applicando degli shaders per rendere un effetto visivo ancora più reale;
- Il generatore di coperture di strade tuttora risulta inefficente. Il motore attuale di generazione dei poligoni si limita esclusivamente a definire delle strip prendendo come punti di curvatura i Control Point degli Shape importati. Questa soluzione risulta errata in quanto non vengono considerate variazioni del DEM intermedie ai punti di curvatura. Una soluzione potrebbe essere quella di applicare dei metodi di interpolazione che consentano di far adagiare correttamente la strada sul terreno;
- Una ulteriore aggiunta alle potezialità di vtBuilder potrebbe essere apportata aggiungendo la possibilità di importare, sempre da coperture vettoriali o file di punti georeferenziati, specifiche a

[1] Questa libreria è il core operativo del tool osgdem esposto nel paragrafo 5.2.2

modelli o coperture procedurali (come muri di cinta, impianti di illuminazione ecc.) allo stesso modo dell'importer di vegetazione con la libreria di vegetazione specifica.

Enviro Integrazione del visualizzatore:

- Come precedentemente descritto (nel paragrafo 6.2.1) cercare di slegare Enviro dal file di HeightField potrebbe non essere la soluzione ottimale ma, durante la fase di importazione, una possibile implementazione aggiuntiva potrebbe risultare nel cercare di abilitare una lettura del file di supporto *.bt* da un server remoto allo stesso modo del terreno paginato. Questo permetterebbe al tool di lavorare su zone localizzate del territorio (funzionalità già testata con il file di HeightField in locale) visualizzando nel contempo tutto il terreno paginato;
- Attualmente la texturizzazione degli edifici, all'interno delle Tile ad alto livello di dettaglio, presenta alcuni problemi dovuti alla gestione puntale delle strutture; le Tile di inserimento vengono individuate in base al baricentro della struttura da inserire. Questa soluzione comporta il problema di ritexturizzare edifici che ricadono in parte all'interno della Tile designata. Potrebbero essere possibili due soluzioni, ovvero: applicare un metodo di multitexture sull'edificio (cercando di coinvolgere tutte le Tile sulle quali la struttura si appoggia) oppure cercare di spezzare la geometria dell'edificio, in modo tale da posizionare le porzioni sulle Tile corrette[2];
- Una ulteriore modifica, per rendere i terreni esportati più gradevoli, si potrebbe applicare cercando di generare dei livelli di dettaglio delle strutture da esportare; l'inserimento, di queste geometrie semplificate, all'interno di Tile intermedie produrrebbe una visualizzazione continua eliminando il problema di *Popping*, quando viene raggiunta la Tile ad alto livello di dettaglio, e permettendo una visualizzazione delle strutture esportate con un raggio di visione più ampio;

[2]la valutazione del metodo più corretto, e meno invasivo, sulla resa finale viene lasciata ad uno studio futuro.

- essendo l'esportazione delle strutture e la generazione del territorio due processi molto costosi, a livello computazionale, e lunghi, in termini di tempo, sarebbe utile riuscire a slegare Enviro da questi compiti. La produzione di programmi batch che si occupano di fare il lavoro di inserimento delle strutture e rigenerazione del territorio produrrebbe la serie di tools utili per demandare il lavoro pesante a macchine adibite per farlo;
- durante la generazione del progetto VEL (paragrafo 2.4.3) un lavoro di tipo *certosino* è stato attuato nella crezione dei boschi e dei campi di grano. Il completamento delle funzionalità di editing di Enviro tramite la creazione di un tool *Pennello*, per l'inserimento di aree coltivate o di boschi[3], potrebbe essere una buona soluzione per evitare un lavoro macchinoso e frustrante;
- Abilitare Enviro alla scrittura di dati in remoto gli permetterebbe di colloquiare con un server esterno in maniera tale da generare dei repository remoti dei terreni sui quali più persone potrebbero fare riferimento. Questa potenzialità potrebbe essere utile per la generazione di un tool capace di far interagire più persone su un progetto comune di lavoro anche da sedi distanti;
- L'applicazione di metodi di Shading, MultiPass Texture, linguaggi Shader, l'abilitazione del motore di ParticleSystem di OSG (per la generazione di effetti tipo fuoco, pioggia e fumo) e il Rendering Volumetrico (per generare masse nuvolose) produrrebbe una visualizzazione ancora più realistica.
- Enviro potrebbe essere esteso con la capacità di adattarsi a differenti RenderWindow, in modo tale da essere sfruttato anche in sitemi Multiple Desktop, e la possibilità di renderizzare l'Environment di scena attravarso una visione di tipo stereoscopica.

Queste e molte altre modifiche potrebbero essere apportate ancora ai tool per generare un FrameWork, ed un protocollo, di lavoro consolidato ed otti-

[3]Stile Traintz: è un simulatore ferroviario. Presenta un particolare Plug-In per la generazione di ambienti in maiera dinamica. Utilizza il game engine *AuranJet* sviluppato dalla casa inglese Auran.

male; come primo step si è pensato ad un sistema di ricostruzione territoriale formato da differenti componenti che dovrebbero contenere:

1. Un plug-in di visualizzazione, possibilmente integrabile nei principali browser e multipiattaforma, o un player della scena (simile a TerraExplorer di Skyline[4]) che consenta la visualizzazione e l'interrogazione remota di modelli del territorio molto estesi e risoluti, struttando tecniche di gerarchie di Livelli di Dettaglio e Paginazioni;

2. Un client di editing che abbia le stesse funzionalità del viewer ma che consenta all'utente di modificare alcuni degli aspetti del territorio, come l'inserimento di modelli in posizioni georeferenziate in modo interattivo e possa comunicare con la componente server-side del repository per l'inserimento dei dati editati;

3. Una componente server che si occupi sia di fornire i dati richiesti dai client di visualizzazione che di accettare le richieste di inserimento e modifica prodotti dai client di editing. Una volta accettate le modifiche, dipendentemente dal tipo di modifica da apportare al territorio, potrebbe essere pensabile l'inserimento diretto nel terreno (ad esempio il caso di un inserimento di tipo puntuale di alberi, edifici ecc.) oppure l'utilizzo di tool di rebuilding in grado (tramite temporizzazioni o su richiesta) di attivare operazioni più onerose, come la rigenerazione del modello territoriale;

Il punto di partenza per il concreto sviluppo di questo FrameWork è possibile collocando il pacchetto VTP, modificato in questo lavoro di tesi, come client di interazione sul paesaggio, ed il plug-in OSG Active-X, come viewer di interrogazione del modello.
Come ultimo appunto di modifica si intende sottolineare che ulteriori aggiornamenti prevedono una riscrittura del codice in maniera più elegante in modo da: rispettare il più possibile il paradigma Object-Oriented e consentire una integrazione con il codice di OpenSceneGraph, ed il codice di libreria di VTP, più consona e perfezionata possibile.

[4]Fa parte del pacchetto TerraSuite; è un player che permette di navigare all'interno di territori traendo i dati da fonti remote.

Capitolo 9

Ringraziamenti

Sono arrivato in fondo! Si certo, l'ho toccato anche parecchie volte in questi anni ma alla fine ce l'ho fatta. È arrivato il momento di ricordare tutte le persone che in questi anni mi hanno dato man forte per superare tutte le avversità (universitarie e non) e che mi sono state vicine.
In primo luogo vorrei abbracciare i miei genitori che mi hanno *sopportato, mantenuto e motivato* in tutti questi anni di studio (venti non sono mica pochi). Un abbraccio anche a mio fratello Davide che mi è stato vicino soprattutto nei primi anni *fuori casa*.
Il conseguimento di questa tesi è stato possibile grazie al continuo contributo di tutte le figure che, con il loro continuo supporto professionale e lo scambio di idee, ed esperienze, ha fatto si che il lavoro procedesse con continui risultati.
In particolare vorrei ringraziare il mio supervisore Luigi Calori che, in questi nove mesi, mi ha seguito costantemente, lasciandomi spazio in tutte le scelte implementative, facendomi ragionare nelle mie prese di posizione (spesso sbagliate) da programmatore in erba e per avermi *scarrozzato*, spesso, nel ritorno a casa dal CINECA.
Ringrazio inoltre Sofia Pescarin che mi ha introdotto nel campo della ricostruzione digitale, indirizzando il mio lavoro ed inserendolo in alcuni progetti di ricerca. Ringrazio inoltre Sofia per aver creduto nelle mie potenzialità permettendomi di interagire con un gruppo di lavoro eccezionale: il CNR-ITABC.
Ringrazio il mio Relatore Prof. Giulio Casciola per aver accettato la mia proposta di tesi ed avermi permesso di svolgerla al CINECA.

Vorrei ringraziare i componenti del VISIT lab e tutte le persone con le quali ho condiviso questi mesi di lavoro. In particolare: Antonella, Francesca, Luca, Marcello, Michele, Silvano e Tiziano.
Grazie mille agli amici di Bologna senza i quali questi anni di università sarebbero stati noiosi e senza senso: Chiara, Fernando ed Enza, la Ballerina, Cristina e Marco, Gloria, Jack, Enrico, Pio, Alessio, Benedetta, il Faggio, Donato, Gianfilippo, la Ballera, Davide, Carlotta, Lele e Pietro. Ed agli amici di Sant'Agata (e dintorni): Giovanna, Cristian ed Elisa, Matteo, Andrea, Claudio e Nicole, Katia, il cugino Marco e Francesca.
Un ringraziamento speciale va a Manuel che mi è stato vicino e mi ha aiutato soprattutto in questi mesi di lavoro difficili.
Vorrei ringraziare la comunità di OpenSceneGraph (nello specifico Don Burns e Robert Osfield) e di Virtual Terrain (nella persona di Ben Discoe) per avermi dato un supporto tecnico eccellente in questi mesi di sviluppo.
Infine, ringrazio tutti gli amici che, anche se ho dimenticato in questo elenco, sono stati importanti in questi anni.

Bibliografia

[1] Bar-Zeev A. SceneGraphs: Past, Present and Future. Technical report, RealityPrime Technology Services, 2003.

[2] Robinson A.H. *Elements of cartography.* Wiley & Sons, Inc., New York, 1995.

[3] Meo A.R. *Indagine conoscitiva sul software a codice a sorgente aperto nella Pubblica Amministrazione.* Ministero per l'Innovazione e le Tecnologie, Roma, 2003.

[4] Buschmann, Meunier, Rohnert, Sommerland, and Stal. *Pattern-Oriented Software Architecture: A System of Patterns.* Wiilley, 1996.

[5] Lardicci C. *Aspetti geometrici della cartografia.* Archimede 34, 1982.

[6] Koller D., Lindstrom P., Ribarsky W., Hodges L.F., Faust N., and Turner G. Virtual gis: A real-time 3d geographic information system. *Proc. Visualization 95*, pages 94–100, 1996.

[7] Eckel G. and Jones K. OpenGL Performer Programming Guide, 2004.

[8] Gamma, Helm, Johnson, and Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1994.

[9] Hoppe H. Progressive meshes. in computer graphics. *Proc. SIGGRAPH 96*, pages 99–108, 1996.

[10] Hoppe H. View-dependent refinement of progressive meshes. in computer graphics. *Proc. SIGGRAPH 97*, pages 189–198, 1997.

[11] Hoppe H. Smooth view-dependant level-of-detail control and its application to terrain rendering. *Proc. Visualization 98*, pages 35–420, 1998.

[12] Hoppe H. View-dependant level-of-detail control and its application to terrain rendering. *Proc. Visualization 98*, pages 35–42, 1998.

[13] Samet H. *Applications of Spatial Data Structures.* Addison-Wesley, 1990.

[14] Barcelo J., Forte M., and Sanders D. Virtual reality in archaeology. *ArcheoPress*, pages 247–263, 2000.

[15] Neider J., Davis T., and Mason W. *OpenGL Programming Guide: The Red Book.* Addison-Wesley, 1994.

[16] Java 3d api specification, 1999.

[17] Calori L., Diamanti T., Guidazzoli A., Liguori M.C., Mauri M.A., and Valentini L. Certosa virtual museum: a dynamic multilevel desktop vr application. *Proc. Eurographics 2003*, 2003.

[18] Calori L., Diamanti T., Felicori M., Guidazzoli A., Liguori M.C., Mauri M.A., Pescarin S., and Valentini L. Databases and virtual environments: a good match for communicating complex cultural sites. *Proc. SIGGRAPH 2004*, 2004.

[19] Treinish L.A. Cartographic projections. Technical report, IBM, Yorktown Heights, NY, 2002.

[20] Duchaineau M., Wolinsky M., Sigeti D.E., Miller M.C., Aldrich C., and Mineev-Weinstein M.B. Roaming terrain: Real-time optimally adapting meshes. *Proc. Visualization 97*, pages 81–88, 1997.

[21] Forte M. Cybernetics, ecology of mind and virtual heritage. 2005.

[22] Forte M., Gòmez L., Pescarin S., Pietroni E., and Vico L. Integrating technologies: The appia antica project. *ArcheoPress*, 2003.

[23] Forte M. and Pescarin S. The virtual reconstruction of the archaeological landscape. *XXIV Rencontres internationales d'Archéologie et d'Histoire d'Antibes*, 2004.

[24] Woo M., Neider J., Davis T., and Shreiner D. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2 3rd ed.* Addison-Wesley, 1999.

[25] Lindstrom P., Koller D., Ribarsky W., Hodges L. F., Faust N., and Turner G. Real-time: Continuous level of detail rendering of height fields. *Proc. SIGGRAPH 96*, pages 109–118, 1996.

[26] Lindstrom P. and Pascucci V. Visualization of large terrains made easy. *IEEE Visualization 2001*, 2001.

[27] Fowler R.J. and Little R.J. Automatic extraction of irregular network digital terrain models. *Computer Graphics*, 13:199–207, 1997.

[28] Rumor. Corso di sistemi informativi territoriali i. Technical Report 12-26, Diploma di Sistemi Informativi Territoriali I, 2005.

[29] Pescarin S., Pietroni E., Dell'Unto N., and Forte M. Real-time interactive reconstruction of archaeological landscapes: an opensource approach from gis to virtual reality. Technical report, CINECA, 2005.

[30] Pescarin S. and Calori L. Appia antica project. Technical report, CINECA, 2005.

[31] Pescarin S. and Forte M. Dal gis alla realtà virtuale: nuove prospettive per la ricostruzione del paesaggio archeologico, in antichità altoadriatiche. *atti del II Incontro Scientifico - Strumenti della salvaguardia del patrimonio culturale: Carta del rischio archeologico e Catalogazione informatizzata; esempi italiani ed applicabilità in Albania*, 2003.

[32] Pescarin S., Liguori M., Diamanti T., and Guidazzoli A. Realtà virtuale: verso una evoluzione dei sistemi informativi geografici. *Atti del Convegno - GIS e Beni Culturali*, 2003.

[33] Pescarin S., Diamanti T., Guidazzoli A., Liguori M., and Felicori M. Dal gis alla realtà virtuale. applicazioni per i beni culturali e il decision making. *MondoGIS*, 37, 2002.

[34] Röttger S., Heidrich W., Slusallek P., and Seidel H. Real-time generation of continuous levels of detail for height fields. *Università di Norimberga*, 2004.

[35] Thatcher U. Rendering massive terrains using chuncked level of detail control, 2002.

[36] De Boer W.H. Fast terrain rendering using geometrical mipmapping. *E-mersion Project*, 2000.

[37] Appia Antica Project. `www.appia.itabc.cnr.it`.

[38] GDAL. `www.remotesensing.org/gdal`.

[39] GeoJpeg. `www.remotesensing.org/gdal/frmt_jpeg.html`.

[40] GeoTiff. `www.remotesensing.org/geotiff`.

[41] Jpeg 2000. `www.remotesensing.org/jpeg2000`.

[42] Multigen Paradigms. `www.multigen.com`.

[43] OpenSceneGraph. `www.openscenegraph.org`.

[44] Progetto MUVI. `http://sirio.cineca.it/muvi`.

[45] Projection Library. `www.remotesensing.org/proj`.

[46] Virtual Terrain Project. `www.vterrain.org`.

[47] wxWindows. `www.wxwindows.org`.

[48] Terrex. `www.terrex.com`.