3° Convegno Tecnico Scientifico
Torino 3, 4, 5 Novembre 2003

Mobile Virtual Reality (MVR): un nuovo modello di realtà virtuale

Mobile Virtual Reality

Andrea Sanna
Politecnico di Torino, Dip. di Automatica e Informatica (DAUIN)
C.so Duca degli Abruzzi 24, I-10129
andrea.sanna@polito.it

Claudio Zunino
Politecnico di Torino, Dip. di Automatica e Informatica (DAUIN)
C.so Duca degli Abruzzi 24, I-10129
claudio.zunino@polito.it

FabrizioLamberti
Politecnico di Torino, Dip. di Automatica e Informatica (DAUIN)
C.so Duca degli Abruzzi 24, I-10129
fabrizio.lamberti@polito.it

ABSTRACT

Una nuova categoria di dispositivi, nota come Personal Digital Assistant (PDA), ha incominciato a diffondersi a partire dalla fine degli anni novanta. Svariate applicazioni software sono state sviluppate per i PDA, ma la grafica 3D di alta qualità è ancora fuori dalla portata delle capacità di calcolo di questi dispositivi. Questo articolo affronta il problema di portare la grafica tridimensionale anche su dispositivi mobili come i PDA proponendo un nuovo paradigma di realtà virtuale denominato Mobile Virtual Reality (MVR); il rendering della scena è eseguito remotamente su un cluster di PC/workstation e poi inviato al palmare.

Le macchine remote sono coordinate mediante l'architettura Chromium che permette di dividere il carico di lavoro tra le schede grafiche dei vari PC e poi di riassemblare l'immagine da inviare all'utente. Dal lato PDA, l'utente può interagire con la scena traslando e ruotando interattivamente l'oggetto 3D. La soluzione proposta consente di visualizzare su PDA modelli estremamente complessi e realistici senza dipendere da soluzioni/prodotti commerciali permettendo, inoltre, di adattare facilmente il sistema ai requisiti di specifiche applicazioni.

Introduzione

La realtà virtuale si è affermata in molte discipline come strumento di ricerca, di intrattenimento e di lavoro. Sebbene le applicazioni di realtà virtuale siano diversificate ed eterogenee, è possibile classificarle in due categorie: Desktop Virtual Reality (DVR) e Immersive Virtual Reality (IVR). Le applicazioni di DVR utilizzano hardware di larga diffusione ricreando il mondo virtuale su comuni monitor; talvolta dispositivi di visualizzazione stereoscopica e di force feedback sono usati per aumentare il realismo e l'interazione con l'ambiente sintetico. IVR utilizza dispositivi più sofisticati

e costosi per dare all'utente un senso di immersione nel mondo virtuale, in particolare, i CAVE permettono di far vivere esperienze altamente realistiche.

Questo articolo propone un nuovo modello di realtà virtuale denominato Mobile Virtual Reality (MVR). L'obiettivo è quello di sfruttare la diffusione dei dispositivi mobili per navigare mondi sintetici. Dispositivi come i Personal Digital Assistant (PDA) rendono l'accesso ad Internet indipendente da vincoli di posizione e il loro utilizzo sta velocemente diventando un importantissimo campo di ricerca; per contro, la limitata capacità di calcolo, la ridotta dimensione dei display e la mancanza di dispositivi di interfacciamento tradizionali sono i principali motivi che hanno finora limitato l'utilizzo di tali dispositivi in applicazioni di realtà virtuale.

Alcuni strumenti software per progettare applicazioni grafiche 3D per PDA esistono già; ad esempio Elite [3] è un motore di rendering tridimensionale basato su Java che consente di creare e visualizzare modelli rappresentati a wireframe. La libreria PocketGL [14] (simile alle ben note OpenGL [13][19]) è scritta in C/C++ e permette di disegnare oggetti 3D e gestire tutte le trasformazioni geometriche. Nonostante questi sforzi, la visualizzazione di oggetti complessi e realistici (task basilare per una vasta gamma di discipline come la visualizzazione scientifica, l'intrattenimento, la realtà virtuale, il CAD e così via) è fuori dalla portata della potenza di calcolo di dispositivi palmari come i PDA.

Un sistema di visualizzazione dovrebbe, in generale, permettere un'esplorazione interattiva dei dati visualizzati; realismo e interattività sono due obiettivi che sui sistemi di visualizzazione tradizionali vengono raggiunti facendo uso di hardware specializzato (ad esempio schede video acceleratici) non (ancora) disponibile per PDA. Una possibile soluzione a questo problema è quella di demandare il task di calcolo ad un'architettura remota e usare il dispositivo palmare solo per la visualizzazione e l'interazione con l'utente. Questa soluzione offre due vantaggi: i dati che devono essere visualizzati sono elaborati da "un'unità" remota, superando i vincoli tecnologici dei PDA, inoltre, la visualizzazione remota non implica la trasmissione dei dati sorgente, in questo modo il sistema è indipendente dal tipo di applicazione/dati.

Questo articolo propone un'architettura basata su Chromium [2][9]. Chromium permette di unificare la potenza di calcolo di un insieme di acceleratori grafici organizzati come un cluster, fornendo un'interfaccia virtualizzata all'hardware attraverso le API OpenGL. Ogni macchina del cluster (o meglio ogni acceleratore grafico) calcola una porzione del fotogramma che può poi essere riassemblata per essere inviata ad un dispositivo di visualizzazione (come nel caso considerato) o essere usata per pilotare direttamente i cosiddetti tiled display.

L'utente può navigare la scena fornendo comandi di roto-traslazione ad un'applicazione grafica basata su OpenGL; i comandi sono imparti mediante un'apposita interfaccia sviluppata per PDA. Ogni comando corrisponde ad un evento che viene trasmesso ad una macchina server remota che controlla il cluster Chromium; il comando è poi trasformato in modo da essere compreso dall'applicazione grafica e il rendering è distribuito tra i vari acceleratori grafici. I frame riassemblati sono opportunamente codificati e mandati al PDA come una sequenza video. Il PDA può essere connesso alla rete mediante collegamenti wireless IEEE 802.11b o GPRS.

Background

L'idea di separare la parte di calcolo da quella di visualizzazione per permettere l'utilizzo di dispositivi non intrinsecamente dotati della sufficiente potenza di calcolo non è nuova. Ad esempio la Silicon Graphics Inc. ha sviluppato un insieme di soluzioni commerciali; tra queste Vizserver [18] che può essere usato per fornire un accesso remoto, trasparente all'applicazione, verso stazioni grafiche SGI di alto livello. Inoltre, Vizserver abilita funzionalità di visualizzazione collaborativa per utenti multipli. GLR [10] fornisce una semplice interfaccia di programmazione per generare immagini su macchine remote permettendo di variare le dimensioni delle immagini e combinando, in soluzioni ibride, sia il rendering locale sia quello remoto.

Anche un certo numero di soluzioni non commerciali sono state proposte, ma spesso esse mancano di indipendenza dall'architettura/applicazione. Un software di volume-rendering distribuito (Visapult) è stato presentato in [1]; Visapult fornisce visualizzazione remota interattiva usando due componenti: un motore di volume-rendering parallelo basato su MPI [6][12] ed un visualizzatore basato su OpenGL. Enget et al. [4] hanno proposto una soluzione in cui le immagini compresse sono mandate a client Java da un server di visualizzazione; l'interazione con l'applicazione è ottenuta mandando richieste CORBA [15]. Una soluzione studiata appositamente per la visualizzazione di dati varianti nel tempo è stata proposta in [11]; due demoni realizzano l'architettura: il primo riceve i dati dal processo di rendering, li comprime e li invia al secondo demone che li decomprime e li presenta all'utente.

Una soluzione generica per la visualizzazione remota basata sull'accelerazione hardware è stata presentata in [17]; questa soluzione è basata sia sul concetto di linking dinamico [7] sia sulle funzionalità dei protocolli GLX [5]. Le applicazioni (basate su OpenGL) vengono eseguite localmente su un server di rendering, ma l'output è configurato in modo da essere inviato ad un display server; delle funzioni GLX ad-hoc sono state realizzate per supportare questo meccanismo. Il sistema usa l'infrastruttura client-server VNC [16] per la trasmissione dell'immagine.

Calcolo distribuito su architetture cluster

I recenti miglioramenti delle schede video dei Personal Computer permettono l'uso di architetture cluster come valida alternativa ai sistemi multiprocessore proprietari. La Stanford University ha sviluppato WireGL [8]: una piattaforma per il rendering interattivo e scalabile per architetture cluster. WireGL è realizzato usando il classico modello client-server; il client "genera" i comandi grafici che devono essere eseguiti sul server (il cluster). Il server riceve i comandi dal client via rete e divide il carico tra i vari acceleratori grafici che costituiscono il cluster. L'applicazione grafica da eseguire deve essere basata su OpenGL dato che WireGL sostituisce opengl32.dll (o libgl.so su sistemi Unix) con una libreria ad-hoc che presenta la stessa interfaccia OpenGL. L'applicazione grafica originaria, se scritta in OpenGL, non deve essere modificata; a run-time la libreria WireGL viene dinamicamente invocata per trasmettere i comandi grafici al server di rendering

Chromium è un ulteriore sviluppo di WireGL che supporta tre "tipologie" di rendering:

- Rendering di tipo sort-first. Il framebuffer è suddiviso in parti (tile) che possono essere renderizzati in parallelo dal cluster; le varie parti sono usate per pilotare un tiled display.

- Rendering di tipo sort-last. L'insieme di dati 3D è diviso in N parti che sono renderizzate in parallelo da N processori. Le parti calcolate dai vari processori sono ricomposte per formare l'immagine finale.
- Filtraggio del flusso di comandi OpenGL. Il flusso dei comandi OpenGL può essere intercettato e modificato mediante una Stream Processing Unit (SPU) per realizzare funzioni di rendering ad-hoc.

Gli utilizzatori di Chromium devono decidere quali nodi del loro cluster saranno coinvolti in una sessione di rendering parallelo e quale tipo di comunicazione sarà necessario; questo può essere specificato usando un sistema di configurazione centralizzato. L'architettura Chromium segue, come le WireGL, un paradigma di comunicazione client-server; i nodi client generano flussi di comandi OpenGL, mentre i nodi server gestiscono connessioni di rete multiple in input e interpretano i messaggi su queste connessioni come flussi di comandi OpenGL. Le trasformazioni sui flussi di comandi avvengono mediante le SPU che sono realizzate come librerie caricabili dinamicamente e forniscono l'interfaccia OpenGL.
In una tipica configurazione Chromium, i client usano una SPU che realizza un rendering di tipo sort-first (detta tilesort SPU) per dividere il carico di lavoro tra le macchine del cluster. I server usano la render SPU per veicolare direttamente i flussi di comandi agli acceleratori grafici. Una SPU, detta readback, gestisce poi la composizione dell'immagine finale. In un sistema Chromium esiste un modulo dedicato a inizializzare l'applicazione (denominato crappfaker) che permette la sostituzione dinamica della libreria OpenGL di sistema con quella di Chromium.

L'architettura proposta

L'architettura proposta consiste in un applicativo software, lato server, realizzato come una libreria dinamica e in un'applicazione grafica, lato client, che abilita la visualizzazione remota su dispositivi PDA. In particolare, la libreria dinamica è in grado di controllare "a distanza" qualunque applicazione grafica basata su OpenGL. L'architettura proposta, è mostrato in Figura 1.

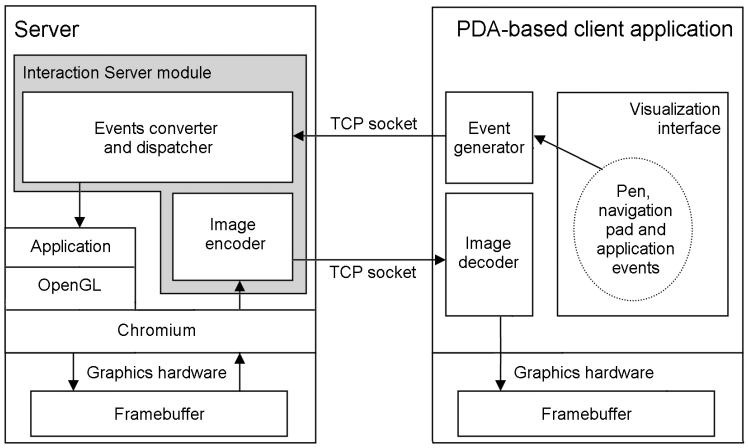

Figura 1 Architettura client server.

L'applicazione lato client

Il controllo remoto del motore di rendering è stato valutato su un dispositivo palmare tradizionale, largamente diffuso sul mercato. Il dispositivo selezionato è Compaq iPaq H3630 PDA equipaggiato con: una CPU StrongARM a 206 MHz, un display LCD touch screen a 12 bit con una risoluzione di 240x320 pixel (2.26x3.02 pollici), 32 MB di RAM e 16 MB di Flash ROM. L'interazione con l'utente è realizzata usando il pennino direttamente sul display, mediante il pad direzionale (usato per emulare le funzioni dei tasti "freccia" di una tastiera tradizionale) e i quattro bottoni di applicazione che possono essere completamente configurati via software.

Mentre in [17] un client VNC è stato usato per controllar remotamente un server-X, in questo lavoro è stato deciso di fornire un supporto speciale per la visualizzazione remota accelerata per PDA sviluppando un'applicazione ad-hoc. L'uso di tecnologie general-purpose, come in [17], può penalizzare le prestazioni complessive del sistema. Questo problema può essere efficacemente risolto progettando e sviluppando un'applicazione ottimizzata lato client.

L'applicazione sviluppata è stata realizzata usando l'ambiente Microsoft eMbedded Visual C++ per WindowsCE 3.0 e il Microsoft Windows Platform Software Development Kit per PocketPC. Differenti architetture per la visualizzazione remota interattiva sono state proposte in letteratura; la maggior parte si basano su Java per garantire l'indipendenza dalla piattaforma [4]. Noi abbiamo scelto di utilizzare il linguaggio "nativo" della piattaforma, in modo da ottimizzare l'uso delle limitate risorse hardware offerte al dispositivo PDA. Nello sviluppo dell'applicazione per il dispositivo mobile è stato scelto un approccio modulare basato su tre componenti software: l'interfaccia di visualizzazione, il decodificatore dei dati da visualizzare e il generatore di eventi.

L'interfaccia di visualizzazione si occupa di presentare i dati e interagire con l'utente. Principalmente gestisce l'area di schermo dedicata al rendering visualizzando il contenuto del framebuffer ricevuto dal server remoto (il cluster di calcolo). Questa interfaccia gestisce anche gli eventi generati dall'utente mediante il pennino, il pad di navigazione e i bottoni di applicazione; i comandi dell'utente sono passati al modulo di generazione degli eventi. L'interfaccia di visualizzazione è mostrata in Figura 2.

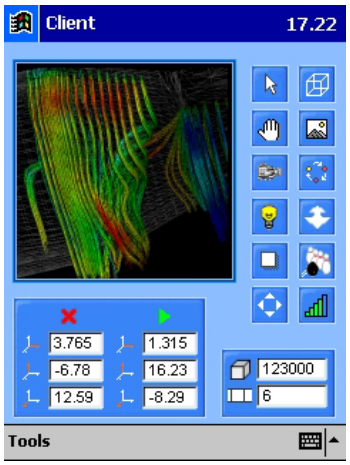

Figura 2 L'interfaccia di visualizzazione.

Una serie di controlli, allocati attorno all'area dedicata alla visualizzazione, permettono all'utente di gestire tutte le funzionalità dell'applicazione remota basata su OpenGL. Il decodificatore dei dati da visualizzare riceve, dal server di calcolo remoto, un flusso di informazioni codificate su un canale TCP stabilito su una connessione wireless. A causa delle differenti larghezze di banda disponibili su connessioni wireless, sono stati considerati diversi schemi di codifica (vedi sezione "Il collegamento di rete"). Diversi decoder sono stati integrati per gestire sia flussi di immagini compresse sia flussi di immagini non compresse. Il generatore di eventi riceve le informazioni riguardanti gli eventi generati dall'interfaccia utente e li converte in pacchetti aventi un formato adatto per la trasmissione attraverso un canale TCP su una connessione di rete a banda stretta.

Il modulo server

Una libreria dinamica multi-thread è stata sviluppata per controllare le comunicazioni con l'interfaccia di visualizzazione sul dispositivo mobile. Questa componente software, nel seguito riferita come Interaction Server, può essere configurata per assegnare opportune funzioni di callback agli eventi generati dall'utente tramite l'interfaccia grafica.

Quando un nuovo evento è generato dall'interfaccia di visualizzazione sul dispositivo palmare, un pacchetto formato da tra valori: codice del comando, posizione x, posizione y è sottomesso all'Interaction Server attraverso il canale wireless; il pacchetto è tradotto, utilizzando un'apposita tabella di conversione, e passato alle funzione di callback appropriata. Una volta che il rendering è stato eseguito, il framebuffer è copiato nella memoria principale e l'Interaction Server codifica l'immagine in un formato adatto alla trasmissione (il formato dipende dal tipo di comunicazione wireless adottata). Questi passi determinano la latenza complessiva (vedi Figura 3) di un'applicazione tra interazione/manipolazione e aggiornamento dell'immagine a video.

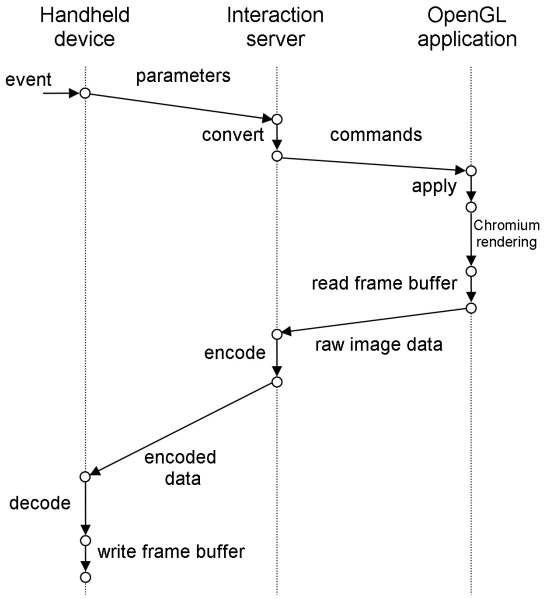

Figura 3 Sequenza temporale delle operazioni.

La trasmissione delle immagini e il controllo remoto dell'applicazione server sono strettamente disaccoppiate. I dati delle immagini e i parametri veicolati al motore di rendering per calcolare le immagini stesse, sono scambiati su due connessioni TCP separate. Questo modulo software può essere facilmente integrato in applicazioni esistenti e permette a qualunque software basato su OpenGL di essere dotato di un'interfaccia grafica per il controllo remoto da dispositivo mobile.

I comandi di rendering OpenGL sono distribuiti su un cluster di PC usando l'architettura Chromium descritta in sezione "Calcolo distribuito su architetture cluster"; l'uso di Chromium permette di gestire modelli estremamente complessi che sarebbe impossibile visualizzare interattivamente su una singola macchina. Inoltre, Chromium fornisce un insieme di componenti (le SPU) le quali sono state modificate nell'architettura proposta per realizzare i canali di comunicazione tra il modulo server e il PDA (vedi Figura 3).

Il collegamento di rete

Il controllo remoto del motore di rendering è stato valutato sui canali di comunicazione mobile più comuni, sia in configurazione WLAN sia WWAN. Il computer palmare è stato dotato di una scheda di espansione per la trasmissione dei dati su IEEE 802.11b e GPRS. La capacità di banda dello standard IEEE 802.11 è di 5.5 Mpbs (IEEE 802.11a) e 11 Mpbs (IEEE 802.11b). GPRS, invece, ha una banda di trasferimento dei dati, teorica, che varia nell'intervallo dal 14.4 kbps (single time-slot) a 144 kbps (multi time-slot) a seconda delle capacità del terminale utente e del provider. L'unità di espansione fornita dalla Compaq per supportare comunicazioni wireless GSM/GPRS è un terminale multi-slot Class 10. Questo terminale fornisce quattro time-slot in ricezione e 2 in trasmissione, permettendo una banda asimmetrica massima di 57.6 kbps in download e 28.8 kbps in upload.

A causa delle limitate risorse computazionali del PDA, il tempo di esecuzione necessario ad eseguire ben specifiche operazioni (lettura/scrittura da socket, scrittura del framebuffer, aggiornamento del contesto grafico) deve essere considerato nel valutare le prestazioni complessive del sistema. Questa valutazione è indipendente dal tipo di canale adottato per la comunicazione wireless. Per ricevere i dati di un'immagine dall'Interaction Server attraverso un socket, sono necessari 60 ms quando si considera una dimensione di frame di 120x120 pixel. Il procedimento di copia del contenuto del buffer TCP nel frame buffer e l'aggiornamento dell'area di visualizzazione richiede un tempo medio addizionale di circa 80 ms. La somma di questi tempi rappresenta il minimo ritardo tra due aggiornamenti successivi dello schermo e pone un limite superiore di sette frame per secondo al massimo frame rate che può essere ottenuto sul PDA scelto per le prove.

La banda larghezza di banda misurata sperimentalmente per un canale di comunicazione IEEE 802.11b è sufficiente per supportare una trasmissione non compressa delle immagini. La trasmissione di immagini non compresse permette di evitare la fase di decompressione sul dispositivo palmare; questa soluzione aiuta a limitare i tempi di latenza e incrementa l'interattività. Le prestazioni cambiano sostanzialmente con una comunicazione di tipo GPRS. La banda misurata sperimentalmente in download è risultata di 42.5 kbps, diventando un collo di bottiglia per la visualizzazione interattiva. Per garantire un frame rate accettabile è stato quindi necessario attuare degli schemi di codifica per la compressione delle immagini.

Sia tecniche di compressione con perdita sia senza perdita sono state considerate. In particolare, sono stati sviluppati dei porting per PocketPC della libreria di compressione senza perdita zlib e della libreria di compressione con perdita libjpeg. I rapporti di compressione ottenuti e i tempi di codifica misurati sono riportati in Tabella 1. La decompressione di un frame su PDA è un'operazione particolarmente onerosa; per decodificare un frame 120x120 pixel, sono necessari da 188 ms per la zlib e 144 ms per libjpeg. Considerando anche il tempo necessario per la lettura del flusso di dati da socket e l'aggiornamento dell'area di schermo dedicata alla visualizzazione, il frame rate calcolato è inferiore a quello ottenibile, teoricamente, con un collegamento GPRS. A dispetto degli algoritmi di compressione, l'ammontare di dati che deve essere trasmesso al client è ancora troppo elevato per ottenere un frame rate interattivo. Una soluzione è quella di scalare l'immagine (ad esempio dimezzando la risoluzione) quando l'utente interagisce con la scena; appena l'utente smette di manipolare l'oggetto, la scene è inviata al PDA alla risoluzione massima. La combinazione di schemi di codifica e riduzione della dimensione delle immagini permette di ottenere, per il PDA scelto, un frame rate di 2 fps anche su una connessione GPRS a banda stretta.

Tabella 1 Risultati sperimentali.

	Dimensione frame		**Tempo di decodifica**		**Frame rate sperimentale**	
Metodo di codifica	Full-size	Half-size	Full-size	Half-size	IEEE 802.11b	GPRS
Non compresso	15 kB	5 kB	0 ms	0 ms	6	-
Zlib	6 kB	3 kB	188 ms	104 ms	-	-
Jpeg (90%)	3 kB	3 kB	144 ms	82 ms	-	2

Realizzazione e risultati

Le prestazioni dell'architettura proposta sono state valutate realizzando uno scenario di visualizzazione remota 3D basato su Chromium. La configurazione di Chromium adottata è quella mostrata in Figura 4; questa configurazione è basata su una macchina "centrale" sui vengono eseguiti sia il crappfaker sia l'applicazione di rendering basata su OpenGL e su un cluster di otto PC. La macchina centrale è equipaggiata con un bi-processore Pentium III 800 MHz, mentre le macchine del cluster sono bi-processori AMD Athlon MP a1.4 GHz con schede grafiche GeForce 2 che eseguono un server Chromium.

La SPU tilesort sulla macchina centrale ordina i flussi grafici in otto parti (tile) che sono poi veicolati su una LAN Gigabit Ethernet ai PC del cluster. Ogni server Chromium intermedio serializza il flusso di dati in arrivo e passa i risultati ad una SPU readback. La SPU readback è "ereditata" dalla SPU render che localmente calcola la porzione di immagine ad essa destinata. Alla fine, il contenuto del framebuffer è passato alla SPU send che trasmette i dati ad un server finale incaricato di riassemblare i risultati parziali.

Alcune applicazioni dell'architettura proposta sono mostrate in Figura 5 e Figura 6.

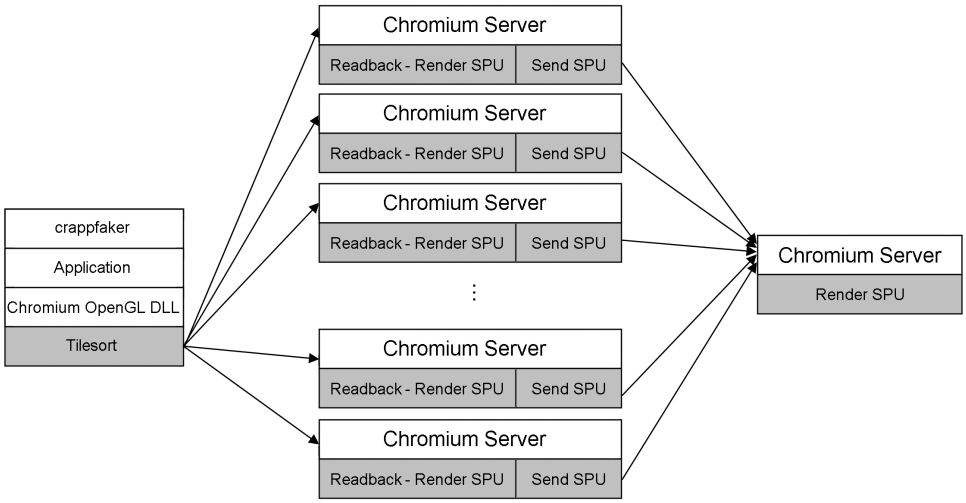

Figura 4 Schema della configurazione proposta.

Figura 5 Esempio di visualizzazione remota su canale IEEE 802.11b. Nell'esempio è visualizzato un modello medico complesso.

Alcune considerazioni

Le prestazioni ottenute sono un compromesso risultante dalla larghezza di banda degli attuali canali di comunicazione wireless e gli schemi di codifica adottati nella trasmissione dei frame. La ristrettezza della banda di canali di comunicazione wireless come il GPRS impone l'adozione di opportuni schemi di decodifica; per contro, la limitata potenza di calcolo del palmare usato per le prove limita la complessità degli schemi di compressione che possono essere impiegati e perciò costituisce un limite al massimo frame rate ottenibile.

La maggior parte dei PDA PocketPC, appartenenti alla generazione del CompaqiPaq usato negli esperimenti, è basata su una CPU a 206 MHz Intel StrongARM SA 110.

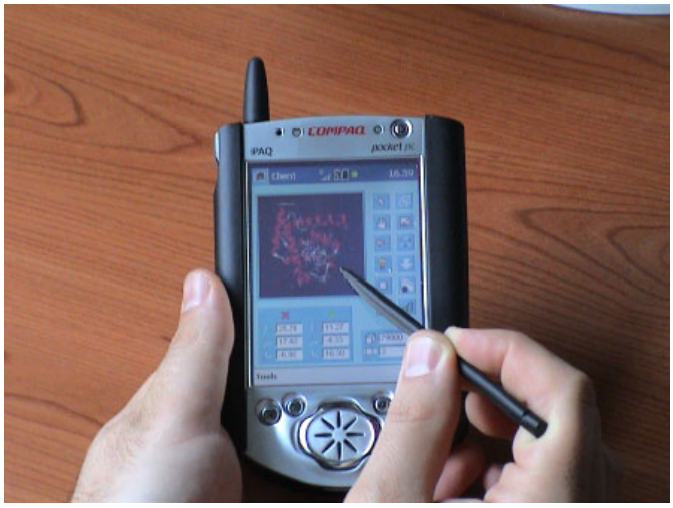

Figura 6 Visualizzazione remota di un modello bio-molecolare complesso su GPRS.

Le prestazioni di alcuni di questi dispositivi sono state valutate mediante una serie di programmi di test e risultano praticamente "uniformi" (vedi Figura 7). Per questo motivo non c'è ragione di credere che l'adozione di un altro PDA della stessa categoria di quello scelto per gli esperimenti possa cambiare in modo significativo le prestazioni del sistema.

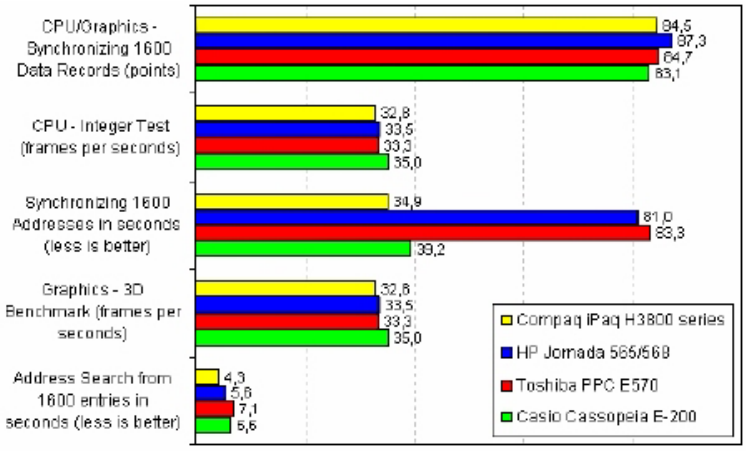

Figura 7 Risultati dei benckmark eseguiti su PDA della stessa categoria di quello scelto per gli esperimenti.

Il problema delle limitate risorse di calcolo disponibili su PDA diventerà meno critico in un prossimo futuro grazie ai miglioramenti che saranno apportati alle tecnologie per dispositivi mobili. Ad esempio, negli ultimi mesi è stata presentata una nuova generazione di PDA, basata sul processore Intel a 400 MHz Xscale PXA 250, che fornisce direttamente un supporto per le

applicazioni multimediali e la grafica 3D. I test dimostrano come i miglioramenti siano significativi rispetto ai PDA equipaggiati con la CPU StrongARM (vedi Figura 8).

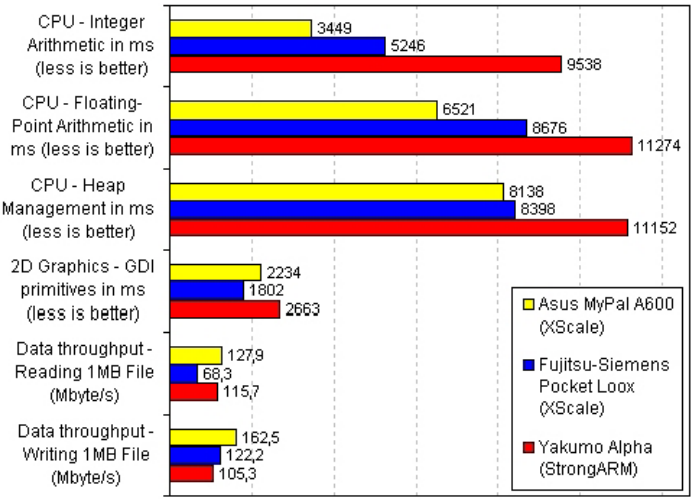

Figura 8 Confronto tra PDA di ultima generazione e palmari equipaggiati con CPU StrongARM.

Inoltre, vanno ricordati i limiti introdotti dai canali di comunicazione considerati, che potranno essere superati con le nuove tecnologie di trasmissione come, ad esempio, IEEE 802.11g.

Conclusioni

In questo articolo è stata presentata un'architettura che abilita la visualizzazione remota interattiva di applicazioni grafiche basate su OpenGL. Le prestazioni dell'architettura sono state valutate nell'ambito di uno scenario di rendering remoto accelerato costituito da un cluster di PC, configurati mediante Chromium, per effettuare il rendering parallelo delle immagini. La soluzione proposta permette di visualizzare modelli estremamente realistici e complessi, in modo interattivo, su dispositivi, come i PDA, dotati di limitate risorse computazionali e interconnessi mediante tecnologie wireless.

Il lavoro futuro sarà mirato allo sviluppo di schemi di codifica specializzati orientati allo streaming; tali schemi saranno specializzati per canali di comunicazione wireless; inoltre, le tecniche di codifica dovranno considerare gli aspetti di sicurezza coinvolti nella trasmissione di dati "confidenziali" specialmente per reti pubbliche come il GPRS.

Bibliografia

1. Bethel, W. 2000. Visualization viewpoints: Visualization dot com. IEEE Computer Graphics and Applications 20:3, 17–20.
2. Chromium. http://sourceforge.net/projects/chromium/.
3. Elite. http://home.rochester.rr.com/ohommes/elite/.

4. Engel, K., Sommer, O., and Ertl, T. 2000. A Framework for Interactive Hardware Accelerated Remote 3D-Visualization. In Proceedings of EG/IEEE TCVG Symposium on Visualization VisSym 2000, 167–177,291.
5. GLX. http://www.sgi.com/software/opensource/glx
6. Gropp, W., Lusk, E., and Skjellum, A. 1999. Using MPI 2nd Edition: Portable Parallel Programming with the Message Passing Interface. MIT Press, Cambridge, Massachusetts, USA.
7. Ho, W.W., and Olsson, R.A. 1991. An approach to genuine dynamic linking. Software - Practice and Experience 21:4, 375–390.
8. Humphreys, G., Eldridge, M., Buck, I., Stoll, G., Everett, M., and Hanrahan, P. 2001. Wiregl: a scalable graphics system for clusters. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques, ACM Press, 129–140.
9. Humphreys, G., Houston, M., Ng, R., Frank, R., Ahern, S., Kirchner, P.D., and Klosowski, J.T. 2002. Chromium: a stream-processing framework for interactive rendering on clusters. In Proceedings of the 29th annual conference on Computer graphics and interactive techniques, ACM Press, 693–702.
10. Kilgard, M., 2002. GLR, an OpenGL Render Server Facilitiy. Silicon Graphics, Inc. (http://reality.sgi.com).
11. Ma, K.L., and Camp, D. M. 2000. High performance visualization of time-varying volume data over a wide-area network status. In Proceedings of the 2000 ACM/IEEE conference on Supercomputing (CDROM), IEEE Computer Society Press, 29.
12. MPI. http://www-unix.mcs.anl.gov/mpi
13. OpenGL. http://www.opengl.org
14. PocketGL. http://pierrel5.free.fr/pocketglb/readmefirst.htm
15. Pope, A. 1997. The CORBA Reference Guide. Addison Wesley.
16. Richardson, T., Stafford-Fraser, Q., Wood, K.R., and Hopper, A. 1998. Virtual network computing. IEEE Internet Computing 2:1, 33–38.
17. Stegmaier, S., Magallón, M., and Ertl, T. 2002. A Generic Solution for Hardware-Accelerated Remote Visualization. In Proceedings of EG/IEEE TCVG Symposium on Visualization VisSym 2002.
18. Vizserver. Silicon Graphics, Inc. http://www.sgi.com/.
19. Woo, M., Neider, J., Davis, T., and Shreiner, D. 1999. OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2 3rd Edition. Addison Wesley.