Defender Game in ambiente Shark

Simone Fettucciari, Luca Rosadini, Simone Mannori

July 4, 2004

Abstract

Lo sviluppo di una certa tipologia di programmi puo' essere notevolmente accelerato quando l'ambiente si sviluppo fornisce i giusti strumenti. Shark nasce come ambiente di programmazione intrinsecamente multithread, con la possibilita' di avere un completo controllo dell'associazione del task ad un certo tipo di scheduler con certi parametri. Uno dei punti di forza della programmazione multithreading e' quello di poter mandare in esecuzione concorrente una molteplicita' di entita', ognuna con i sui dati privati, ma che condividono lo stesso codice, con la possibilita' di scambiare dati in maniera corente con altre entita'.

Introduzione

Nel nero freddo degli spazi siderali, una malvagia (e quindi sufficientemente stupida) razza aliena ha deciso di prendere il controllo della CPU mangiando tutto il tempo di elaborazione disponibile. La perfida Madre Aliena si sposta industubatata riempiendo lo schermo della sua immonda progenie. Riuscira' il nostro Defender a sterminare l'orda selvaggia prima della fusione del nucleo ?

1 Campo di gioco

Il campo di gioco e' "toroidale" ovvero alto e basso sono chiusi, mentre il lato destro si richiude sul lato sinistro.

2 Entita'

Sul campo di gioco sono presenti le seguenti entita':

2.1 Madre Alieni

La Madre Alieni (bianca) e' un task hard real time lancianto con la pressione del tasto **"M"**. La Madre Alieni ha due compiti: - gestire il suo spostamento - generare in maniera casuale gli ALIENI fino ad un numero massimo preimpostabile. La Madre Alieni termina quando accumula un numero di collisioni (Missili o Astronavi) pari a cinque.

2.2 Alieni

Ogni Alieno e' un task hard real time. L'aspetto dell'alieno e' quello di un disco colorato con una linea orizzontale. Il colore dell'alieno defininisce il suo "tipo" e quindi il suo comportamento. Esistono quattro tipi di alieni:
- BASTARD : [rosso] tende ad inseguire l'astronave secondo un algoritmo definito da un doppio controllore P.I.D. (uno per la x, uno per la y);
- BAD : [viola] usa un algoritmo casuale di spostamento in tutte le direzioni;
- NORMAL : [giallo] si sposta da destra a sinistra con algoritmo casuale per la quota;
- STUPID : [verde] si muove a velocita' costante in linea retta orizzontale.

Gli alieni sono generati nella posizione della Madre Alienie muoiono quando colpiti da un missile o in caso di collisione con l'Astronave.

2.3 Astronave

L'aspetto dell'astronave e' quella di un'astronave (bianca). L'Astronave e' un task hard real time attivato dalla pressione del tasto "A". L'astronave si muove da sinistra verso destra con velocita' costante, mentre lo spostamento verticale ? gestito dall'utente tramite i comandi da tastiera "W" (su) e "S" (giu'). L'astronave puo' sparare un numero massimo di missili impostabile (10 di default). L'astronave muore quando collide con un ALIENO o con la Madre Alieni.

2.4 Missile

Il missile (bianco) e' un task hard real time; si muove orizzontalmente a velocita' doppia rispetto a quella dell'astronave. Il missile viene avviato dal *main(...)* con la pressione della **SPACE BAR**. Il missile termina la sua corsa e la sua esistenza: - quando incontra un Alieno;
- quando arriva al bordo destro;
- quando incontra la Madre Alieni.

2.5 Monitor

Il TASK monitor (non real time) ha il compito di visualizzare i tempi di esecuzione dei vari TASK hard real time.

3 Particolari tecniche utilizzate

3.1 Algoritmi di scheduling

Tutti i task hard real time (MadreAlieni,Alieni, Astronave e Missile) sono assegnati ad uno scheduler EDF (vedi [initfile.c]). EDF permette di utilizzare il processore fino al 100/100. In ogni caso i task non sono particolarmente impegnativi per la CPU utilizzata per i test.

3.2 Protezione delle subroutine di disegno oggetti

Il disegno degli oggetti ha richiesto una protezione con semaforo di mutua esclusione per evitare gli effetti "sfarfallamento" e sovrapposizione degli elementi grafici.

3.3 Protezione delle strutture per lo scambio dati tra task cooperativi

Alcune variabili e strutture dati sono state protette da semafori che implementano un meccanismo di mutua esclusione, in modo tale da garantire la coerenza.

3.4 Modello produttore consumatore per l'instradamento dei comandi dalla tastiera

Siccome non e' possibile leggere concorrentemente la tastiera da task diversi, la lettura dell'input dell'utente e' riservata al *main(...)*. Quando l'interprete dei comandi nel *main(...)* rileva un input associato ad una azione relativa ad altri task (es. lo spostamento dell'Astronave) il carattere di comando viene inserito in una apposita struttura dati. In tale struttura dati e' presente un flag rappresentante lo stato del comando. Il flag di stato assume il valore PRODUCED nel momento della sua generazione e il valore CONSUMED quando acquisito dal task a cui e' destinato.

4 Problemi incontrati

Si riportano gli ostacoli piu' significativi incontrati durante il ciclo di sviluppo.

4.1 Incompatibilita' con le schede video

Purtroppo la versione di Shark impiegata in questo progetto (Shark 1.2) NON e' riusultata compatibile con alcune schede video, anche di tipo recente. Siamo comunque riusciti a completare il lavoro utilizzando i PC del laboratorio di informatica ed un portatile (Acer Travelmate 202 TE) che e' risultato pienamente compatibile. Successive versioni (Shark 1.4) hanno migliorato questo aspetto.

4.2 WCET violation

In una delle prime versioni del programma, ci potevano essere delle situazioni nelle quali (x,y) assumevano valori debolmente negativi. In tale caso le primitive grafiche richiedevano un tempo di elaborazione eccessivo, tale da superare i limiti di WCET impostati. L'uso si opportuni "filtri limitatori" su (x,y) ha risolto il problema.

4.3 JET su piu' task

Non siamo ancora riusciti a capire il perche' dell'impossibilita' di utilizzare le primitive del JET per misurare i tempi di elaborazione di piu' di un task contemporaneamente.

5 Conclusioni

A parte il problema *WCET violation* lo sviluppo ha seguito un percorso lineare.