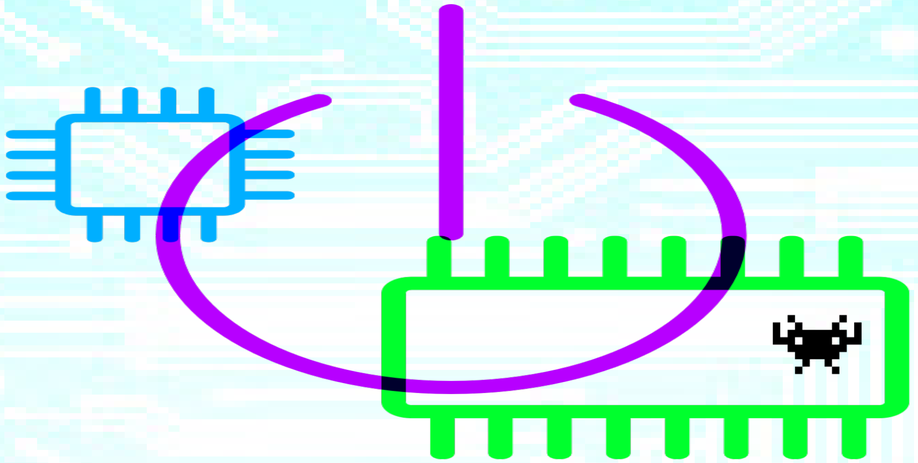


CORSO DI

SISTEMI OPERATIVI: LABORATORIO

UNIVERSITÀ DEGLI STUDI DI CAGLIARI
Dipartimento di Matematica e Informatica
Via Ospedale 72, 09124 Cagliari - Italy

SPECIFICHE PROGETTO

SPACE DEFENDER

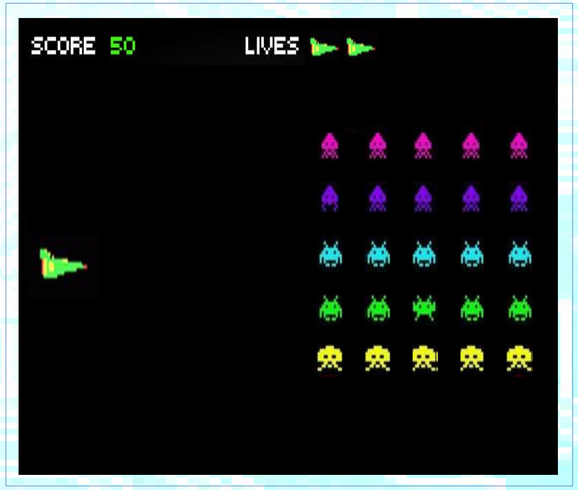
Introduzione

Space Invaders e **Defender** sono dei videogiochi arcade sviluppati negli anni Settanta/Ottanta, il gioco **Space Defender**, che rappresenta il progetto da realizzare, può essere considerato una combinazione di questi due videogiochi.

L'idea è quella di impiegare la struttura di gioco che caratterizza **Space Invaders**, ma con uno sviluppo di tipo orizzontale (come quello di **Defender**) invece che verticale.

+

=



SPACE INVADERS **DEFENDER** **SPACE DEFENDER**

Modalità di consegna

- Il progetto **dovrà essere consegnato entro il termine indicato nel regolamento** mediante una email indirizzata a tutti i docenti e tutor del corso, avente come oggetto “**[SO] Consegna progetto**”
- Verificare attentamente ogni cosa prima dell'invio, in quanto eventuali **versioni del progetto successive al primo invio NON verranno prese in considerazione**
- Il progetto dovrà essere **strutturato su diversi file** (sorgenti ed header) gestiti in modo corretto mediante un opportuno *Makefile* che includa anche la direttiva “*clean*” per la rimozione dei file intermedi.
- Il progetto **dovrà essere sviluppato sulla macchina virtuale** utilizzata per il corso o, comunque, su una macchina con le medesime caratteristiche, in quanto **non verranno prese in considerazione giustificazioni per problemi causati dall'utilizzo di differenti ambienti di sviluppo**
- Il progetto deve essere funzionante in ogni sua parte, altrimenti non verrà preso in considerazione e quindi non discusso

Valutazione del progetto

- La valutazione verterà sui seguenti aspetti:

- **struttura del progetto**: numero e organizzazione dei file, correttezza del Makefile e corretta definizione della direttiva “*clean*” per la rimozione dei file di compilazione intermedi;

- **scrittura del codice**: ragionevolezza delle soluzioni adottate, comprensibilità e indentazione del codice, qualità dei commenti, e altri aspetti correlati;

- **programmazione concorrente**: corretto avvio e terminazione dei processi/thread, corretta gestione delle pipe, dei semafori, e altri aspetti correlati;

- **Implementazione del gioco**: fluidità del movimento degli oggetti coinvolti e corretta interazione tra essi.

Durante la discussione del progetto verrà valutata la capacità del singolo studente di giustificare/modificare qualunque parte del codice, nonchè qualunque scelta progettuale adottata, nonchè l'abilità nell'argomentare eventuali variazioni indicate dai docenti (per esempio, “perchè due pipe anziché una?”, oppure “perchè questo tipo di gestione della memoria condivisa invece di quest'altro”, ecc.).

Ciascun componente di un gruppo sarà interrogato separatamente.

Versioni richieste

Il progetto dovrà essere realizzato in due diverse versioni:

- **Versione 1:**
  utilizzando correttamente i **processi** per la gestione della elaborazione concorrente e il meccanismo delle **pipes** per la comunicazione
- **Versione 2:**
  Utilizzando correttamente i **thread** per la gestione della elaborazione concorrente e le primitive di sincronizzazione (**semafori** e/o **mutex**) per la gestione dell'accesso concorrente alle risorse

Specifiche del progetto 1/x

- Nell'ambito della programmazione di sistema in ambiente Linux, codificare, compilare ed eseguire su linux un programma dal nome “**Space Defender**”, ispirato ad alcuni storici videogame (“*Space Invaders*” e “*Defender*”). Utilizzare gli strumenti per la elaborazione parallela, la comunicazione e sincronizzazione visti a lezione. Riferirsi alle esercitazioni di laboratorio (per esempio, “guardie e ladri”) per avere un'idea circa la struttura di base da realizzare.

- Visualizzare nella parte sinistra dello schermo un oggetto **astronave**, il cui movimento è limitato a spostamenti di tipo verticale sulle prime colonne dello schermo. L’astronave è rappresentata da una forma a scelta che occupi un’area massima di 6x6 caratteri. Il suo movimento è definito dai tasti freccia (freccia su e freccia giù) della tastiera. Utilizzare un thread o processo (a seconda della versione) che implementi la generazione delle coordinate dell’oggetto astronave

- L’astronave, ogni qualvolta viene premuto il tasto spazio, rilascia 2 oggetti missile rappresentati ciascuno da un singolo carattere che si spostano da sinistra verso destra in modo diagonale, uno verso l'alto e uno verso il basso. Utilizzare thread o processi (a seconda della versione) che implementino la generazione delle coordinate degli oggetti missile

Specifiche del progetto 2/x

- Visualizzare sulla parte destra dello schermo M oggetti “**navicelle nemiche**“ (M è un parametro dell’applicativo che deve essere possibile variare su richiesta dei docenti, eventualmente nei limiti del range di valori indicato dagli studenti del gruppo) che si muovono da destra verso sinistra e dall'alto verso il basso o dal basso verso l'alto, non necessariamente in formazione allineata.

- Ciascun oggetto deve occupare un’area massima di 3x3 caratteri, ed è richiesto che venga utilizzato un thread/processo distinto (a seconda della versione) che implementi la generazione delle coordinate di ciascun singolo oggetto **navicella nemica**, fino a che tutte le **navicelle nemiche** vengono distrutte.

- Ogni singola **navicella nemica** viene distrutta quando viene colpita da un missile lanciato dall'**astronave**. Questo tipo di gestione rappresenta la prima fase della vita delle **navicelle nemiche**. Quando la traiettoria di una **navicella nemica** incrocia quella di un'altra **navicella nemica** le traiettorie invertono il moto in senso verticale, con un effetto rimbalzo.

- La prima fase di vita di una **navicella nemica** termina con la sua distruzione, e quando questo avviene, ogni **navicella nemica** distrutta viene sostituita da 4 nuove **navicelle nemiche** (navicelle di secondo livello) formate da più caratteri aventi movimento a piacere ma coordinati fra di loro. Utilizzare **un solo** nuovo thread/processo (a seconda della versione) per la generazione delle coordinate di questo gruppo di navicelle, gestendole come **un unico** oggetto grafico. Quando la traiettoria di un gruppo navicella incrocia quella di un altro gruppo navicella le traiettorie invertono il moto in senso verticale, con un effetto rimbalzo.

Specifiche del progetto 3/x

- Una volta che una **navicella nemica di secondo livello** viene colpita da 2 missili, essa esplode e scompare definitivamente.
- Ciascuna **navicella nemica** (in entrambe le fasi di vita, cioè primo e secondo livello) rilascia ad intervalli regolari un oggetto **bomba** rappresentato da un singolo carattere che si sposta da destra verso sinistra con moto rettilineo; utilizzare un thread/processo (a seconda della versione) che implementi la generazione delle coordinate dell'oggetto **bomba.**
- Se l'**astronave** viene colpita da una **bomba** o entra in contatto con una **navicella nemica** esplode distruggendosi.
- Il gioco finisce se:
  - Tutte le **navicelle nemiche** sono distrutte (il giocatore vince)
  - L'**astronave** (o tutte le astronavi, nel caso siano previste più "vite") viene distrutta (il giocatore perde)
  - Una **navicella nemica** raggiunge la colonna più a sinistra dello schermo (il giocatore perde)

Specifiche del progetto 4/x

- E’ obbligatorio che l’architettura del programma sia basata su un **task** (thread o processo a seconda della versione) per ciascun oggetto che si muove sullo schermo (astronave, navicella nemica di primo livello, gruppo di navicelle nemiche di secondo livello, missile, bomba).

- Ciascuno di questi task si deve occupare di **generare** e **trasmettere** le proprie coordinate al task che si occuperà di: (1) **disegnare** sullo schermo gli oggetti; (2) verificare le **collisioni** o l’uscita di un oggetto dallo schermo e comportarsi di conseguenza.

- Il cuore del programma è di massima incentrato sulla comunicazione delle coordinate fra **N produttori** (gli oggetti in movimento) e **un consumatore** (gestore di disegno e movimento). Verificare **sempre** che siano evitate le scritture su buffer di comunicazione pieno e le letture da buffer di comunicazione vuoto, per ciascuno degli esercizi e che, inoltre, venga effettuata una **corretta terminazione** delle risorse impiegate (processi, thread, e ogni altra risorsa impiegata che necessita di essere terminata/deallocata esplicitamente).

- Un’**architettura di base** alla quale riferirsi è quella delle **esercitazioni di laboratorio** come, per esempio, “*guardie e ladri*”, dove ciascun oggetto ha un task associato che genera la sua posizione in maniera random, o secondo una traiettoria, o seguendo i tasti, e dove ci sono dei task che si occupano di disegnare sullo schermo gli oggetti che si spostano (prima si cancella la vecchia posizione, poi si disegna la nuova), monitorando nel contempo le collisioni .

Specifiche del progetto 5/x

- E’ opportuno scegliere correttamente la durata delle pause negli spostamenti degli oggetti, in modo da avere una visualizzazione comprensibile. Tenere conto della specifica particolare richiesta per la prima fase di vita delle navicelle nemiche
- Opzionalmente saranno estremamente apprezzate versioni che implementino miglioramenti alle specifiche indicate nel presente documento, variazioni che, comunque, devono essere chiaramente indicate nella mail di consegna del progetto, come:
  - astronave e navicelle che non esplodono al primo colpo, ma perdono un carattere ogni volta;
  - Delle linee in basso che scorrendo diano l'idea di un movimento orizzontale (sul genere del gioco originale “Defender”);
  - bombe o missili particolari che esplodendo provocano più danni
  - Introduzione della visualizzazione del punteggio e del numero di vite del giocatore
  - quadri (livelli) successivi del gioco
  - . . . E così via!

BUON LAVORO