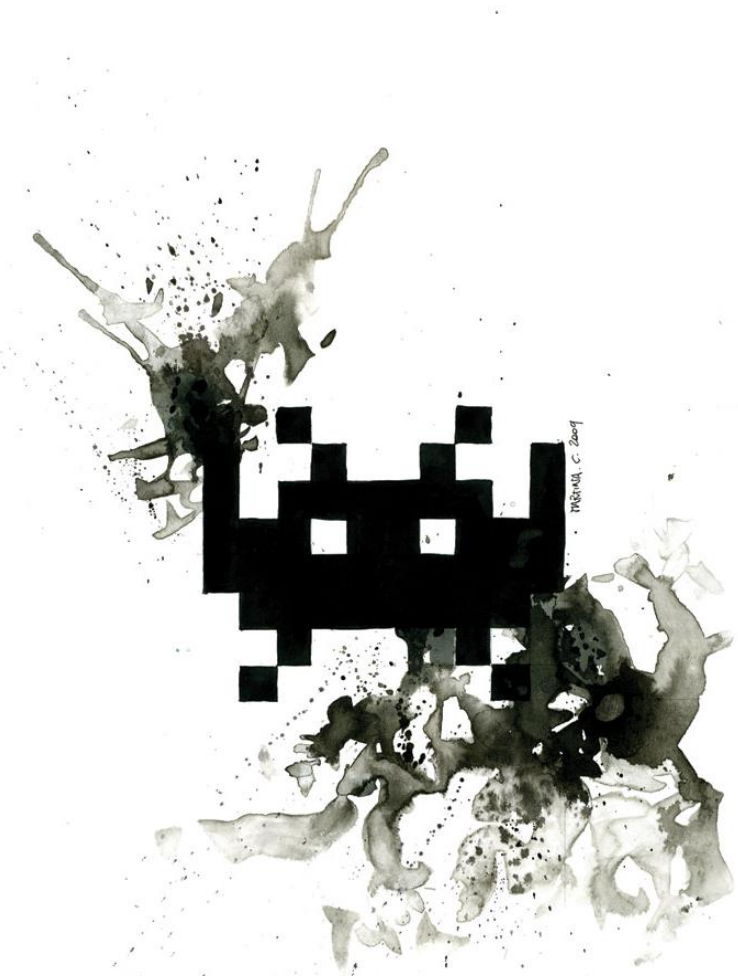
Giammarco Boscaro **ITIS C. Zuccante** **A.S. 2012/13**



documentazione progetto esame
space invaders

INDICE

1. INTRODUZIONE

Il progetto qui presentato consiste in un semplice gioco realizzato in Java ed interfacciato ad Arduino. Esso è il famoso Space Invaders, sviluppato nel 1978, rielaborato e riadattato per renderlo più moderno e offrendo diverse opzioni in più. Il vero gioco è eseguito nel desktop, ed è giocabile senza ulteriori periferiche, utilizzando la tastiera. È possibile però utilizzare Arduino come controller, sfruttando il giroscopio per muovere il giocatore e un pulsante per sparare. Al contempo il punteggio viene visualizzato in un piccolo LCD di 16 colonne e 2 righe. Il software include diverse opzioni, ad esempio, per disattivare gli effetti sonori, il fuoco automatico o il supporto per Arduino. Inoltre è possibile, navigando tra i menu, visualizzare le istruzioni del gioco o consultare la classifica dei punteggi dei giocatori, implementata grazie all'uso di un database Access. Durante la partita è possibile mettere in pausa il gioco, per poi riprendere la partita o tornare al menu. È possibile, infine, inserire il nome del giocatore e scegliere la porta a cui è connesso Arduino, che varia da computer a computer.

COS'È JAVA

Java è un famoso linguaggio di programmazione ormai utilizzato nella maggior parte dei dispositivi elettronici. Fu creato nel 1992 da Sun Microsystems, è può essere eseguito in tutti i dispositivi che supportano la *JVM* (Java Virtual Machine), indipendentemente dall'hardware e dal sistema operativo. Come C++, anche Java è un linguaggio fortemente tipizzato e *object oriented*. A differenza di C, però, Java non viene prima compilato, producendo file in *ByteCode* con estensione *.class*, che verranno poi interpretati, al momento dell'esecuzione, dalla *JVM*. Questa soluzione però può rendere l'esecuzione dei programmi più lenta rispetto ad altri linguaggi. Java può anche essere eseguito su pagine web. In questo caso l'applicazione viene detta applets.

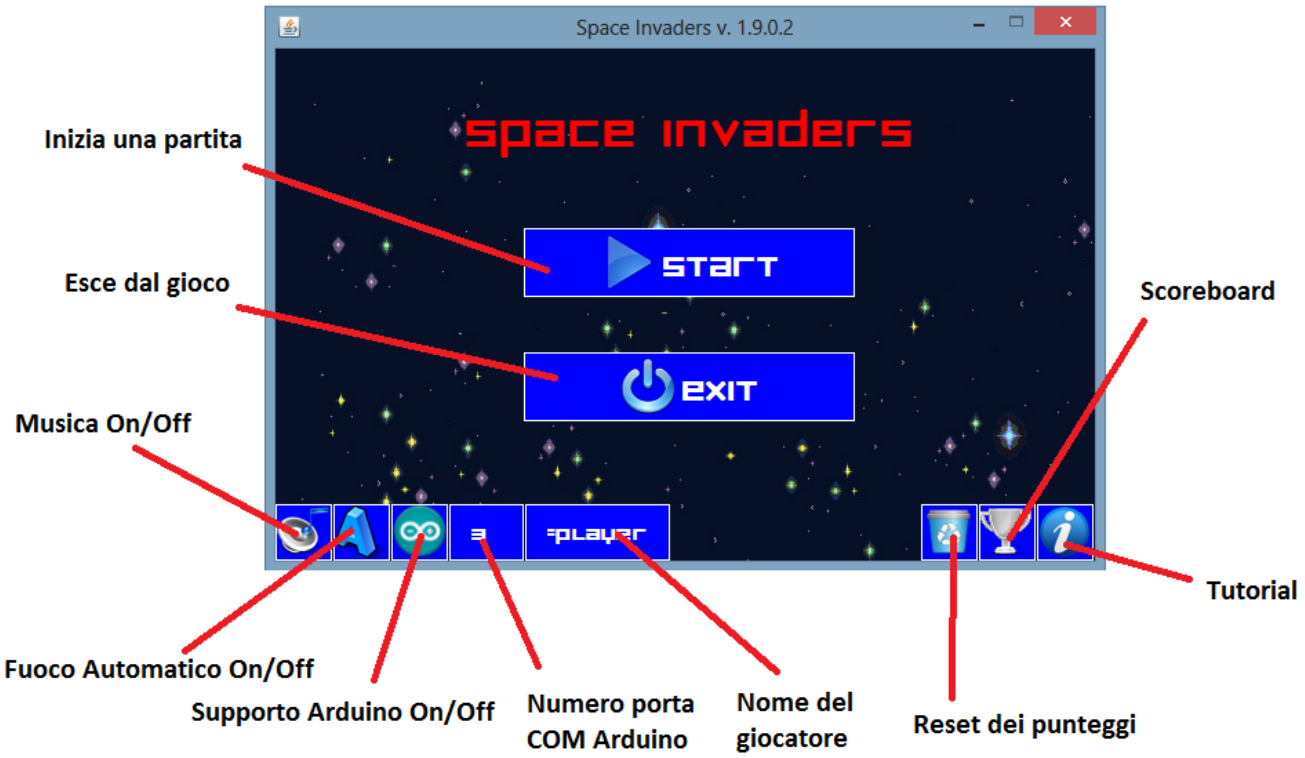
COS'È ARDUINO

Arduino è un framework open source che comprende un linguaggio di programmazione (detto Wiring e simile a C++) facile da apprendere e utilizzare, e una scheda dal prezzo ridotto e prodotta in Italia che permette però di realizzare progetti anche complessi. La scheda di Arduino monta un microcontrollore, che nel nostro caso lavora a 16Mhz, e diversi pin, di cui 6 analogici e 13 digitali, oltre a una porta USB utilizzata per caricare i programmi e per comunicare col computer tramite il serial monitor. Grazie alla comunità, sempre in forte crescita, sono disponibili nel web miriadi di librerie che permettono ad Arduino di lavorare con praticamente tutti i dispositivi elettronici disponibili nel mercato, come sensori, display, motori ecc.

SOFTWARE

L'applicazione desktop viene eseguita in una finestra di dimensioni 600x400 pixel non ridimensionabile.
Sono presentati, di seguito, alcuni screenshot del programma in gioco o mentre si naviga nei menu, correlati di descrizione dei vari elementi presenti sullo schermo.

- Menu principale:



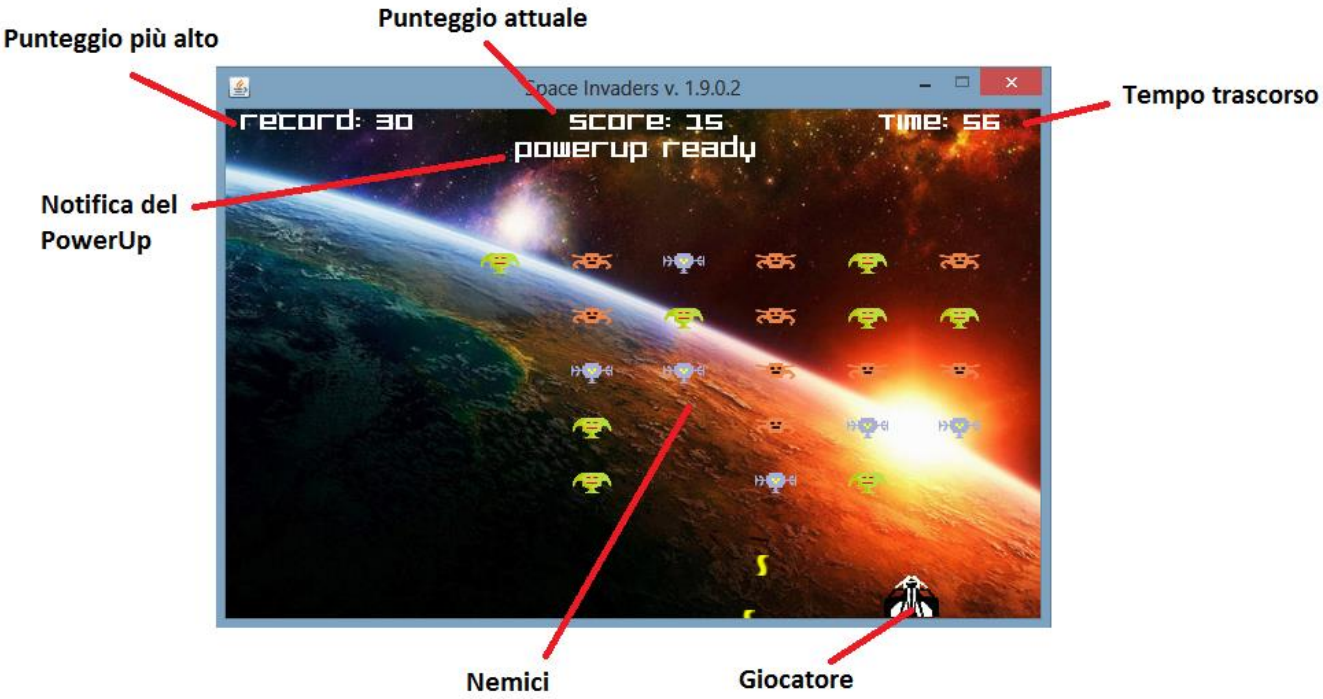
È qui visualizzata la schermata principale del gioco, la quale dà accesso a tutte le funzioni del software. È ovviamente la schermata che appare all'avvio del programma. L'immagine è presa dalla versione 1.9.0.2.

Il menu principale permette di accedere al gioco o di visitare i vari sottomenu. Nella zona bassa della schermata sono inoltre visibili tutte le opzioni disponibili.

- Sottomenu

I vari sottomenu sono tre: Tutorial, Weapons & Points e Scores.
Tutorial è una finestra che mostra le istruzioni del gioco. Weapons & Points mostra ogni tipo di laser disponibile nel gioco e il danno che produce. Inoltre visualizza ogni tipo di alieno e il punteggio che si ottiene alla sua morte.
Scores è invece la classifica, dove vengono visualizzati nome, punteggio e data della partita di ogni giocatore.

- Interfaccia in gioco:



È qui possibile vedere la schermata del gioco durante una partita. L'immagine è presa dalla versione 1.9.0.2.

ISTRUZIONI

Di seguito sono riportate le istruzioni per poter giocare con o senza il supporto di Arduino. Le stesse istruzioni sono disponibili in formato digitale nel software stesso, cliccando, nel menu principale, nel pulsante "Informazioni" in basso a destra.

Utilizzando la sola tastiera

- Movimento: AS o Frecce direzionali
- Sparare: Barra Spaziatrice
- PowerUp: F
- Pausa: ESC

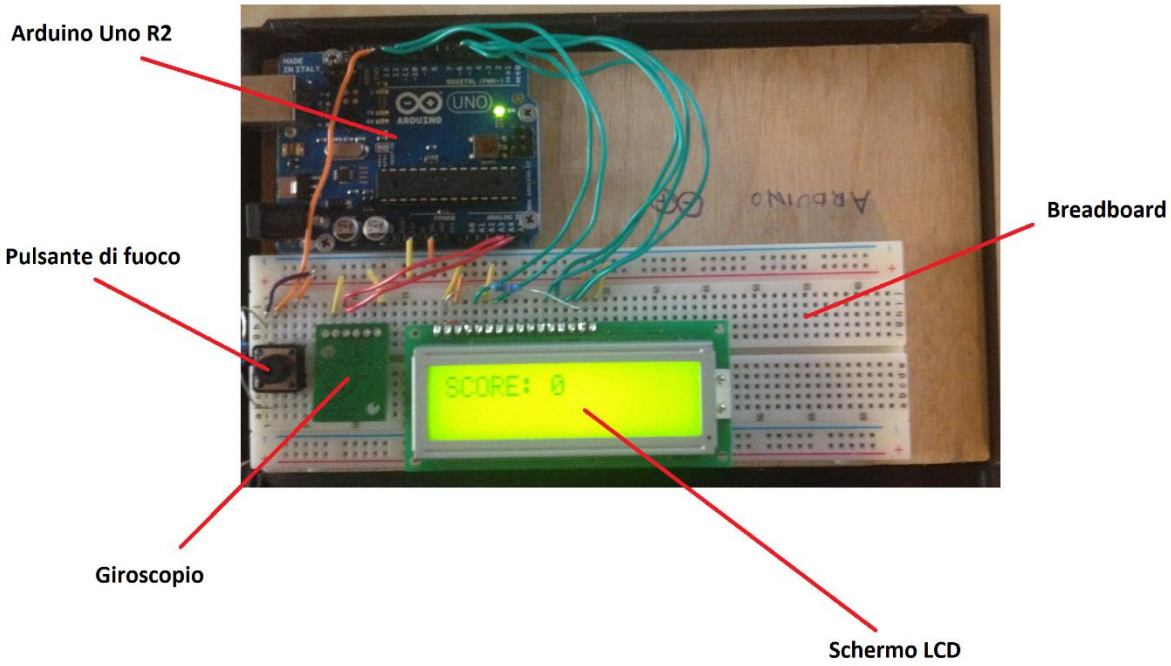
Sfruttando il supporto di Arduino

- Movimento: Giroscopio
- Sparare: Pulsante
- PowerUp: F
- Pausa: ESC

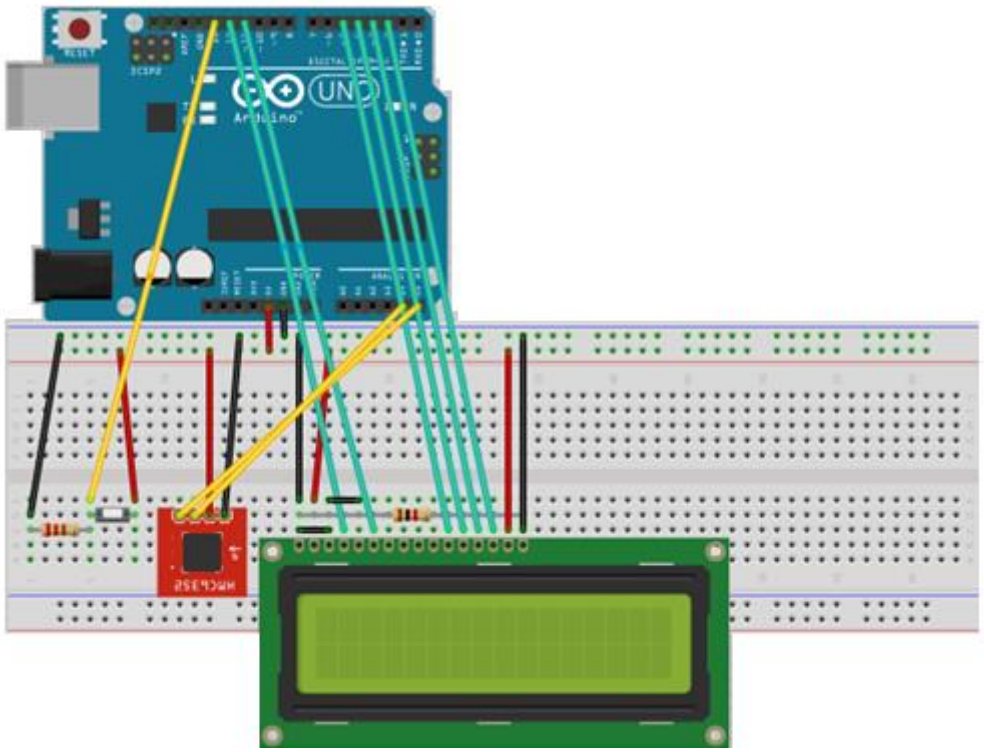
Notare che utilizzando Arduino è comunque necessario l'uso della tastiera per eseguire alcune azioni.

HARDWARE

Come precedentemente indicato, nel progetto è stato utilizzato anche Arduino e diversi dispositivi elettronici, tra cui in giroscopio, un LCD e un pulsante. Di seguito una foto dell'hardware, in funzione, utilizzato nel progetto, correlato di descrizione per ogni elemento visibile.



Inoltre è disponibile il circuito teorico di facile comprensione realizzato tramite Fritzing



2. TEORIA

In questa sezione verrà spiegato il funzionamento del programma, i vari oggetti utilizzati e le diverse funzioni che essi svolgono.

Il gioco vero e proprio è contenuto in un *JPanel*, nel nostro caso chiamato *SpacePanel*, in cui vengono disegnate delle immagini, cioè gli elementi del gioco, alle coordinate indicate, grazie alla libreria *Swing* inclusa in Java. La funzione utilizzata è *paintComponent*, che permette sia di disegnare e colorare forme basilari, sia di stampare a schermo immagini. Per creare del movimento, ad ogni ciclo, il contenuto del pannello viene cancellato ridisegnato alle coordinate richieste. Il pannello possiede, inoltre, dei *KeyBindings*, cioè delle azioni associate alla pressione e/o rilascio di un tasto, necessari per comandare il giocatore. Il tutto è racchiuso in un *JFrame*, cioè la finestra del programma, la cui dimensione non è modificabile. Nello stesso frame vi sono altri pannelli che contengono tutti i diversi menu del gioco. Essi sono nascosti e vengono visualizzati alla pressione di un certo tasto o click di un pulsante virtuale grazie all'uso di una layout, sempre inclusa in *Swing*, chiamata *CardLayout*. Questa layout raccoglie tutti i pannelli del frame e ne mostra uno a scelta quando richiesto. Nel pannello di gioco inoltre sono presenti gli oggetti *sounds* e *music*, che permettono di riprodurre la musica in background e gli effetti sonori del gioco e sono entrambi istanze della stessa classe *Sound*. L'oggetto *scores* gestisce, invece, la scoreboard del programma, mentre l'oggetto *serial* gestisce l'interfacciamento e la comunicazione con Arduino. Tutti gli oggetti e le classi citate verranno analizzate in seguito.

I vari menu sono tutti organizzati allo stesso modo e generalmente includono gli stessi elementi: vari pulsanti, aree di testo, etichette e font. Questi componenti sono stati implementati grazie agli oggetti *JButton*, *JTextField*, *JToggleButton*, *JLabel* e *Font*, provenienti dalla libreria *Swing* e disposti sulla spazio disponibile per formare un menu chiaro e semplice.

All'avvio, come in ogni programma sviluppato in Java, viene eseguita la funzione *main*, che crea il frame, inizializza tutti i pannelli e riproduce la musica. All'avvio del gioco, il frame mostra il pannello *SpacePanel*, già pronto all'esecuzione, e fa partire il suo *thread*. A questo punto è possibile giocare. Tornando al menu principale il *thread* verrà bloccato e il pannello verrà re inizializzato, in modo da essere già pronto per una nuova partita. Il programma, ad ogni ciclo, controlla gli interrupt ricevuti dalla tastiera ed esegue le azioni richieste (aggiornando le coordinate degli elementi, attivando i powerUp ecc.), controlla le varie collisioni verificando che la partita non sia conclusa, e ridisegna l'intero contenuto del pannello. Nel caso di utilizzo di Arduino verrà creato un ulteriore *thread* che si occuperà del trasferimento dei dati tra la il computer e la scheda.

CLASSI

Di seguito verranno descritte brevemente le classi più significative del progetto. Per una migliore comprensione è consigliato consultare il codice sorgente, correlato di commenti.

- Main

Come già spiegato, la classe *Main* rappresenta l'intero frame, viene eseguita per prima ed inizializza tutti i pannelli, esegue i suoni, permette di accedere ai vari sottomenu e gestisce tutte le opzioni disponibili nel software.

- SpacePanel

SpacePanel è il pannello che contiene il gioco vero e proprio. Il pannello implementa l'interfaccia *Runnable*, implementata, tra l'altro, dalla classe *Thread* stessa. Grazie a questa interfaccia è possibile eseguire le istruzioni contenute nella funzione *run* della classe *SpacePanel* come un thread. Ogni ciclo il processo controlla i tasti premuti, aggiorna le coordinate di tutti gli elementi dello schermo, che successivamente vengono ridisegnati da *paintComponent*. Dopodiché viene inviato ad Arduino il punteggio attuale.

- Alien

La classe Alien rappresenta un, appunto, un nemico. Essa contiene tutte le informazioni necessarie al gioco, come la vita totale, la vita attuale, il tipo di alieno, l'immagine che lo rappresenta e la posizione attuale nel pannello. La classe, inoltre, contiene diverse funzioni che indicano se il nemico è morto, se è già stata visualizzata l'esplosione o che si occupano di aggiornare la vita attuale dell'alieno.

- AlienCreator

È una classe che si occupa di creare la "flotta" dei nemici che poi verrà visualizzata su schermo. Vengono creati casualmente combinazioni di 30 alieni tra i 3 disponibili, che verranno disposti in 6 colonne e 5 righe, e salvati in un singolo array. Il pannello di gioco si occuperà poi di leggere, ad ogni ciclo, tutte le informazioni necessarie per ogni alieno e visualizzarlo nel modo corretto a schermo.

- DBManager

Come facilmente intuibile dal nome, questa classe gestisce il database. Essa implementa le funzioni basilari necessarie a manipolare un database, cioè aprire e chiudere la connessione ed eseguire una query. Queste funzione verranno utilizzate dalla classe Score per manipolare effettivamente la scoreboard del programma.

- Score

Questa classe contiene funzioni create ad hoc per il presente progetto e che rendono possibile l'aggiornamento, la visualizzazione e il reset della classifica di punteggi, il tutto attraverso semplici query che verranno eseguite dal *DBManager* Inoltre si occupa di rilevare la data attuale e di scriverla, in un formato standard, nel record, durante l'inserimento di un nuovo punteggio.

- Thumbnails

La suddetta classe è un semplice JPanel contenente solo un immagine di background, indicata al momento dell'inizializzazione dell'oggetto. Viene utilizzata nei menu di tutorial e istruzioni per mostrare le anteprime dei nemici e delle diverse armi utilizzabili.

- Serial

Serial è una versione modificata della classe *SerialTest* presente su Arduino Playground e utilizzata per testare il corretto interfacciamento tra Java e Arduino. Essa permette la comunicazione tra le due realtà tramite la porta USB del computer. Istanziato l'oggetto, viene inizializzata la comunicazione a 115200 nella *COMPort* indicata all'avvio del programma. Il thread viene poi fatto partire. Durante l'esecuzione l'oggetto resterà in attesa dell'arrivo di qualche dato dalla porta seriale. Quando le informazioni sono disponibili, esse vengono inviate direttamente al pannello di gioco, che si preoccuperà di interpretarle. Dallo stesso pannello, al contempo, provengono i dati riguardanti il punteggio del giocatore, che verranno inviati dall'oggetto *serial* ad Arduino tramite la funzione *send*.

- MaxLengthTextDocument

Questa classe svolge l'unica funzione di limitare ad un certo numero di caratteri il contenuto di un *JTextField.*

- Sound

Come indica il nome, questa classe permette di riprodurre effetti sonori e musica del gioco, bloccare la riproduzione in caso di necessità e rendere il gioco muto o, viceversa, ripristinare l'audio. La funzione *playSound* permette effettivamente di leggere il file audio.

3. REALIZZAZIONE

La realizzazione parte da un vecchio progetto Java. Si tratta di un semplicissimo Pong, del 1966, realizzato solo utilizzando le librerie swing e awt e gli elementi basilari messi a disposizione da esse. Successivamente il programma è stato modificato inserendo un menu e delle immagini. Dalla stessa base è nato, poi, un altro progetto: Snake, altro classico gioco di facile realizzazione. Anche Snake prevedeva un menu e l'utilizzo di sprites al posto delle forme geometriche disponibili con swing. Questo progetto, però, permetteva inoltre di collegarsi ad Arduino e mostrare il punteggio in un LCD, permetteva di giocare in multiplayer nello stesso computer e di scegliere il livello su cui giocare. Il software risultava però pesante e scarsamente ottimizzato. Sempre partendo dalla stessa base software è stato poi realizzato il progetto corrente, Space Invaders. Esso include molte più funzioni ed inoltre risulta essere decisamente più leggero e veloce dei progetti antecedenti, oltre ad essere visivamente più pulito.

Il progetto è stato sviluppato utilizzando l'ambiente di sviluppo Eclipse, il quale permette una scrittura veloce e senza errori sintattici, offre diverse opzioni utili allo sviluppatore e un debugger intuitivo. Per lo sviluppo del software di Arduino l'unica scelta possibile è stata utilizzare l'ottimo Development Kit fornito gratuitamente nel sito ufficiale. I vari sprites degli alieni, delle armi e del giocatore sono stati realizzati con Photoshop CS5.5, mentre le icone sono state prese da una repository online. Il database è stato creato tramite Microsoft Access e interfacciato a Java tramite i driver ODBC corrispondenti.

4. VERSIONI

GRAFICA SWING

Nella prima versione realizzata il gioco non aveva alcun menu, nessun supporto ad Arduino o ai Database. Consisteva in sole due classi: Main, cioè il frame, e Game, cioè il panello del gioco. Tutta la grafica era realizzata tramite funzioni di swing come *fillRect* o *fillOval*, che permettono di disegnare forme geometriche nella posizione e della dimensione voluta. La scelta dei colori, inoltre, era ridotta ai soli colori di base resi disponibili dalla libreria. Purtroppo non sono disponibili screenshot di questa versione.

SPRITES

Il software è stato poi drasticamente modificato, eliminando le forme geometriche e sostituendole con immagini, dette sprites. Come si può notare nell'immagine sottostante, questa versione presentava già una scoreboard. I punteggi, però, erano salvati su file di testo. Inoltre il software mancava di menu e sottomenu. L'applicazione è stata costantemente aggiornata e modificato fino ad assumere l'aspetto precedentemente mostrato in diversi screenshot.

IMPLEMENTAZIONE ARDUINO E DATABASE

L'implementazione di Arduino è avvenuta dopo aver realizzato l'interfaccia grafica attuale. L'intervento non è stato invasivo. Nel programma principale è stato necessario solo inserire un nuovo oggetto, *serial*, che richiamava la funzione *send* in un punto preciso dell'algoritmo precedentemente sviluppato. Ponendo poi attenzione a chiudere ed aprire correttamente la connessione con Arduino, in modo da non provocare errori critici, è stato possibile far funzionare correttamente l'interfacciamento.
Anche l'implementazione del database Access, avvenuta successivamente, è stata relativamente semplice e veloce. Avendo già una classe incaricata di gestire lo storico dei punteggi tramite un file di testo, è stato possibile modificarla e renderla in grado di accedere alla base di dati tramite query SQL, mantenendo invariato tutto il resto del codice.

5. PROBLEMI RISCONTRATI

Come in ogni progetto software, durante lo sviluppo si presentano vari problemi, tra cui bug o difetti di progettazione.
Alcuni elementi del software, come l'interfaccia grafica, la gestione dei punteggi o i vari menu, erano già stati testati in progetti antecedenti e praticamente esenti da bug. Grazie a una progettazione teorica su carta e all'uso del debugger, in generale ovviare a difetti di questo tipo non è stato particolarmente impegnativo.
Uno dei problemi riscontrati è stato l'interfacciamento al database. Anche avendo i driver ODBC necessari installati, su certe macchine il programma non poteva connettersi alla base di dati. La discordanza dell'architettura del sistema operativo, a 64 bit, e Microsoft Office, a 32 bit, rendeva impossibile il corretto funzionamento dei driver. Installando un Microsoft Office a 64 bit il problema è stato automaticamente risolto.

Il più grande problema è stato sicuramente trovare un metodo per poter leggere dati da Arduino. Mentre per inviare dati da Java ad Arduino sono necessarie solo poche istruzioni, per il processo inverso è stato necessario modificare, la classe *SerialTest* presente sul sito ufficiale di Arduino. All'inizio una funzione dedicata alla lettura della porta USB veniva richiamata ad ogni ciclo di gioco direttamente dalla classe *SpacePanel*. La soluzione presentava diversi difetti, tra cui calo drastico delle prestazioni e diverse *NullPointerException*, in quanto il codice non era in grado di riconoscere quando il buffer era vuoto. Nella versione finale, un thread si occupa di verificare se nella porta seriale sono presenti byte da leggere. Nel caso la risposta sia affermativa, il thread invia tutte le informazioni direttamente allo *SpacePanel*, cioè al pannello di gioco.

Altri problemi, molto meno gravi, si sono presentati durante lo sviluppo. Sono, però, stati risolti in breve tempo grazie all'utilizzo del manuale di Java o ad una ricerca in internet.

6. MATERIALI E STRUMENTI

A seguire l'elenco di tutti i materiali e strumenti utilizzati, software e hardware, e le varie librerie di Java incluse nel progetto.

MATERIALI

- Arduino Uno R2
- Display LCD 16x2
- Giroscopio CMPS10 6 Assi
- Pulsante
- Resistenze (220Ω, 1kΩ)
- Cavo USB
- Breadboard
- Cavetti vari

SOFTWARE

- Eclipse IDE for Java EE Developers 1.5.2.20130211
- Arduino DevKit 1.0.4
- RXTX Library 64bit 2.2.20081207
- Microsoft Access 2013
- Access Database Engine 64bit 14.0.4730.1010
- Photoshop CS 5.5
- JDK 7u21 (Java Development Kit)
- Fritzing 0.8.0b

LIBRERIE JAVA

- Javax.swing
- Java.awt
- Javax.imageio
- Java.io
- Java.sound.sampled
- Java.util
- Java.sql
- Java.text
- Gnu.io

NB: Photoshop è stato utilizzato per la realizzazione degli sprites e la modifica delle icone. Fritzing, invece, è stato usato per ricreare virtualmente il circuito teorico di Arduino.

7. CODICE

Il progetto è stato sviluppato, come precedentemente detto, utilizzando due linguaggi, Java e Wired (Arduino), oltre che a una piccola quantità di SQL. L'intero programma Java supera qualche migliaio di righe di codice, e per ovvi motivi non è possibile inserirlo in questa documentazione ne analizzare ogni istruzione dettagliatamente. Si è scelto quindi di spiegare in breve l'intero codice di Arduino, formato da meno di 100 righe di codice, e di analizzare solo qualche funzione importante del software scritto in Java. Tutti gli algoritmi sono accompagnati da commenti di quasi ogni istruzione. I file sorgente, totalmente commentati, sono comunque allegati alla documentazione.

ARDUINO

Di seguito il codice completo del programma in esecuzione su Arduino, con commenti.

```
#include <LiquidCrystal.h>  //librerie utilizzate
#include <Wire.h>

//definizione delle variabili
#define ADDRESS 0x60 //definisce l'indirizzo del giroscopio
const int buttonPin, weight;
const char sclPin, sdaPin;
int score, buttonState;
char x;
//inizializzazione LCD (RS, Enable, D4, D5, D6, D7)
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup() {
        //inizializzazione delle variabili
        sclPin = A5;
        sdaPin = A4;
        buttonPin = 13;
        weight = 24;
        score = 0;
        buttonState = 0;

        //inizializzazione dei pin
        pinMode(buttonPin, INPUT);
        pinMode(sclPin, INPUT);
        pinMode(sdaPin, INPUT);

        digitalWrite(sclPin, LOW);
        digitalWrite(sdaPin, LOW);
        digitalWrite(buttonPin, HIGH);

        //scrive sull'lcd il punteggio 0
        lcd.begin(16, 2);
        lcd.setCursor(0, 0);
        lcd.print("SCORE: 0");

        Serial.begin(115200); //inizializza la trasmissione seriale alla velocità di 115200 baud
}

void loop(){
```

```
        sendGyro(); //invia i dati del giroscopio a java
        checkButton(); //controlla la posizione del pulsante
        showScore(); //stampa il punteggio sull'lcd

        delay(25);
}

void checkButton(){

        buttonState = digitalRead(buttonPin);  //legge lo stato del bottone
        if (buttonState == HIGH){
                Serial.println(255);  //se è premuto invia 255 a java
        }
}

void showScore(){

        if (Serial.available()) { //controlla che la porta seriale sia libera
                score = Serial.read();   //legge i dati da java
                lcd.setCursor(7, 0);  //mostra il punteggio aggiornato
                lcd.print(score);
        }
}

void sendGyro(){

        Wire.begin();  //inizializzazione della connessione col cmps10
        Wire.beginTransmission(ADDRESS);
        Wire.write(5);   //inizio a leggere dal registro 5 (rollio)
        Wire.endTransmission();

        Wire.requestFrom(ADDRESS, 1);  //richiesta di un byte
        while(Wire.available() < 1);
        x = Wire.read(); //lettura del byte
        int value = x/8;
        Serial.println(value);  //invio dei dati a java
}
```

Il codice riportato permette di utilizzare Arduino come joystick per il gioco in esecuzione nel desktop. Il programma invia il valore dell'angolo di inclinazione del giroscopio mentre verifica se il pulsante è stato premuto. Al contempo viene letto il punteggio del giocatore tramite la porta seriale, che viene visualizzato nell'LCD.
I Baud indicano i "simboli" al secondo trasferiti. I simboli posso essere formati da più bit, quindi la il baud rate non è necessariamente corrispondente al bit rate. 115200 è la velocità massima supportata da Arduino.

JAVA

Di seguito alcuni spezzoni particolarmente rilevanti del codice, scritto in Java, dell'applicazione.

- Avvio del programma:

```
public static void main(String[] args) {  //main dell'intera applicazione
        SwingUtilities.invokeLater(new Runnable() {
```

```
                public void run() {
                        Main thisClass = new Main();  //istanzio il gestore
dell'applicazione

      thisClass.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  //operazione di
base quando il frame viene chiuso (in questo caso interruzione del processo)
                        thisClass.setVisible(true);   //il frame è visibile
                }
            });
      }
```

All'avvio, come in ogni applicazione Java, viene eseguita la funzione *main*. Il metodo invokeLater è necessario per eseguire il programma in un ambiente sicuro. Swing, infatti, non è "*thread-safe*". Questo significa che potrebbero presentarsi dei problemi nella gestione dell'interfaccia nel caso vi fossero diversi processi che agiscono nei componenti del programma, come nel nostro caso. Il frame viene creato e reso visibile.

- Implementazione del giroscopio:

```
      private void gyro(){

            if(serialRead < 0){ //se la lettura è negativa

                  playerFlagSx = true; //mi sposto a sinistra
                  playerFlagDx = false;
                  left = serialRead;

            }else if(serialRead > 0){ //se la lettura è positiva

                  playerFlagSx = false;    //mi sposto a destra
                  playerFlagDx = true;
                  right = serialRead;
            }
            else
                  playerFlagSx = playerFlagDx = false; //se il valore è 0 sto
fermo
      }
```

La funzione viene richiamata quando vengono ricevuti dati da Arduino. Viene verificata l'inclinazione del sensore ed in base al responso vengono modificate le "flag direzionali" del giocatore, che permettono di capire in quale direzioni ci si muove. Inoltre viene modificata la velocità di spostamento in base ai gradi di inclinazione del giroscopio.

- Movimento della flotta di alieni:

```
private void moveAliens(){    //movimento della griglia di alieni

      if(nowTime - timeToMove >= alienTic){     //se è passato 1 secondo
dall'ultimo movimento
            timeToMove = System.currentTimeMillis();

            if(aliensList[aliensList.length-1].x >= this.getWidth() - 40){
      //se la griglia è quasi fuori dai bordi
                  derLeft = true; //cambio di direzione del movimento
                  hasChanged = true;     //segnale che la direzione è cambiata
            }
            else if(aliensList[0].x <= 10){
                  derLeft = false;  //idem vedi sopra
```

```
                hasChanged = true;
            }

            for(int i = 0; i < aliensList.length; i++){

                aliensList[i].setImage("src/img/alien"+ aliensList[i].number +
anim_aliens%2 +".png"); //cambio delle immagini degli alieni per creare un
animazione

                if(derLeft)
                    aliensList[i].x-=10;    //cambio di coordinate
orizzontali in base alla direzione dello spostamento
                else
                    aliensList[i].x+=10;
                if(hasChanged)
                    aliensList[i].y += 15; //cambio di coordinate verticali
se ho raggiunto il bordo

            }

            anim_aliens++;    //incremento della variabile di supporto per le
animazioni degli alieni

            hasChanged = false;
        }
    }
```

Lo spostamento degli alieni avviene ogni secondo. È necessario quindi salvare in una variabile l'ultima volta che ciò è avvenuto. Nel caso in cui questo lasso di tempo sia passato, gli alieni vengono spostati di 10px a destra o a sinistra, in base alla direzione in cui erano precedentemente diretti. La variabile *derLeft* indica, tramite un valore booleano, se la flotta procedeva verso sinistra o no. Nel caso in cui i nemici raggiungano i bordi della finestra, la direzione viene cambiata. La variabile *hasChanged* ricorda che, essendo cambiato il verso, è necessario abbassare l'intero blocco di alieni in basso di 15px.
Questa funzione, inoltre, ogni secondo cambia l'immagine degli alieni, in modo da creare l'effetto animazione, alternando le due disponibili.
Allo stesso modo è realizzata l'animazione dei laser, sia del giocatore che degli alieni.

- Animazione dei laser:

```
gc.drawImage(actual_laser[anim_laser%3], laserX,laserY ,10 , 15, null);
      //disegno delle animazioni del laser del giocatore e degli alieni
            gc.drawImage(laser_yellow_anim[anim_laser%3], alienShot1X,
alienShot1Y , 10, 15, null);  //la variabile di supporto per l'animazione da
sempre un numero compreso tra 0 e 2
            gc.drawImage(laser_yellow_anim[anim_laser%3], alienShot2X,
alienShot2Y , 10, 15, null);  //cosi l'array scorre e ricomincia autonomamente
ed ogni ciclo l'immagine cambia producendo
            anim_laser ++;          //incremento della variabile per le
animazione
```

Le istruzioni sovrastanti si trovano all'interno della funzione *paintComponent*, nella quale *gc* è l'oggetto *Graphics*, che permette di disegnare elementi nello schermo. Nel nostro caso viene utilizzato il metodo *drawImage* per stampare sul pannello un immagine .png. Le immagini dei laser sono contenute in un array. Grazie all'operazione %3, ogni ciclo viene letta una cella diversa senza mai uscire dai limiti del vettore. La lettura sequenziale delle immagini permette di ricreare l'animazione.

- Esempio di KeyBinding

```
class RightAction extends AbstractAction{

                private static final long serialVersionUID =
5105285857544616683L;  //aggiunto da eclipse automaticamente

                public void actionPerformed(ActionEvent ev) {
                      playerFlagSx = false;
                playerFlagDx = true;    //il giocatore va verso destra
                }
      }
```

Per effettuare un *KeyBinding*, cioè associare una serie di istruzioni alla pressione o rilascio di un tasto, è prima necessario creare una classe che estende *AbstractAction*. Questa classe necessita della funzione *actionPerformed*, che indica cosa accade quando l'azione *RightAction* viene richiamata. In questo caso l'azione verrà associata alla pressione della freccia destra, quindi le "flag direzionali" del giocatore verranno modificate in modo che il personaggio si muova destra. Per implementare effettivamente il KeyBinding è necessario un ulteriore passaggio.

```
rAct = new RAction();//istanzio l'azione
getInputMap().put(KeyStroke.getKeyStroke("RIGHT") ,"rightPressed"); //alla key
associo un evento col nome personalizzato
getActionMap().put( "rightPressed", rightAct); //all'evento associo un azione
```

Istanziata la classe appena creata è necessario associare al tasto *RIGHT* un evento, in questo caso *rightPressed (NB: è possibile chiamare l'evento in qualsiasi modo)*. Successivamente a questo evento viene effettivamente associata l'azione che dev'essere eseguita. Perché il KeyBinding funzioni correttamente è necessario che il pannello sia in focus. Richiedere il focus è facile, basta richiamare la funzione

```
      requestFocusInWindow();       //richiesta del focus
```

SQL

Il database del programma è molto semplice. Consiste in un'unica tabella dove vengono salvati nome del giocatore, punteggio e data.

```
      CREATE TABLE Scoreboard(

            User CHAR(9) NOT NULL CHECK(User<>""),
            Score INTEGER NOT NULL CHECK(Score>=0),
            ScoreDate CHAR(10) NOT NULL CHECK(ScoreDate<>""),
            ID AUTO_INCREMENT NOT NULL PRIMARY KEY
      )
```

Anche le query utilizzate per manipolare il database durante l'esecuzione del programma sono semplici, non contenendo neanche una JOIN.
Per resettare la classifica dei punteggi la query utilizzata è la seguente:

```
DELETE FROM Scoreboard
```

La quale cancella l'intero contenuto dell'unica tabella della base di dati. Per visualizzare la scoreboard nel menu predisposto, invece, viene utilizzata la query:

```
SELECT User, Score, ScoreDate  FROM Scoreboard ORDER BY Score DESC
```

Che seleziona dalla tabella tutti gli attributi tranne l'ID e li mostra in ordine decrescente, dal punteggio più alto al più basso.
Per visualizzare il record, durante la partita, viene infine utilizzata questa query:

```
SELECT MAX(Score) AS Max FROM Scoreboard
```

Che semplicemente seleziona il punteggio massimo dalla tabella.