Federico Comerci, 4^C, 19/04/2020

Commento Space Invaders Lezione 1

Ho creato la classe principale contenente il metodo "main" è l'ho chiamata "SpaceInvaders", che estende la classe "JFrame". Il "JFrame" serve per avere tutte le funzionalità di una finestra nel nostro ambiente grafico GUI (per esempio di premere la X per terminare il gioco). Se si preme la X della finestra, invece di terminare il programma, la finestra viene nascosta; per fare in modo di terminare il programma bisogna aggiungere la seguente linea di codice nel costruttore della classe "SpaceInvaders": "this.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);" Il "Canvas" serve per disegnare elementi grafici e permette di catturare gli input dell'utente (tramite tastiera, mouse, ecc...). In Space Invaders il "Canvas" rappresenta lo spazio profondo, di conseguenza ho creato una classe di nome "Space" che estende la classe "Canvas". Per dare un'introduzione al gioco in cui si vede il nome del gioco e del suo autore in maniera grafica ho creato una classe chiamata "Intro" che estende la classe Space. Per poter aggiungere elementi grafici allo spazio profondo ho sovrascritto la funzione: "public void paint() {", poi ho definito la classe "Paint" per scriverci "Space Invaders" e "by Federico Comerci" e infine ho aggiunto la schermata di introduzione alla finestra di gioco nel costruttore della classe "SpaceInvaders". In seguito ho realizzato un'animazione dove la scritta provenga da fuori dello schermo e che "viaggi" verso l'interno. Per ottenere questo effetto ho usato un parametro "i" da aggiungere alla posizione e alla dimensione del carattere. Generando una successione di schermate con parametro "i" che assuma passo passo valori più piccoli, in questo modo do l'impressione che la scritta si muova e diventi più piccola come se si allontanasse. Per ottenere questo funzionamento, ho creato un metodo chiamato "run()" nella classe "Intro" che richiami il metodo di nome "repaint()" ereditato dalla classe "Canvas" in un "for loop" a cui ogni iterazione il parametro "i" viene decrementato. L'unico problema arrivando fino a qua è lo sfarfallio. Lo sfarfallio è dovuto al fatto che l'operazione di formazione dell'immagine sullo schermo e la formazione della immagine da visualizzare nel Canvas non sono sincronizzate. Per risolvere questo problema, ho usato due "Canvas", uno su cui costruire la nuova immagine, e l'altra da visualizzare sullo schermo. Una volta che la nuova immagine è stata completata, ho scambiato i due "Canvas", così che la nuova immagine venga visualizzata, mentre quella vecchia venga cancellata per comporre l'immagine successiva. Questa strategia permette di visualizzare sullo schermo sempre un immagine completata, rimuovendo così lo sfarfallio. La Classe "Canvas" ci dà un strumento chiamato "BufferStrategy" che automaticamente fa questo per noi. Nel metodo "run" ho avviato il meccanismo di "BufferStrategy": "createBufferStrategy(2); strategy = getBufferStrategy();" (il 2 indica che ci bastano 2 buffer). Nel metodo "paint" ho inserito la funzione: "Graphics2D g = (Graphics2D) strategy.getDrawGraphics();" e poi le funzioni: "g.dispose();" e "strategy.show();". L'ultima istruzione causa il nuovo buffer appena composto con la nuova immagine ad essere visualizzato sullo schermo; quando si richiede il buffer, viene fornito il buffer con l'immagine precedente.