



FIFTEEN GAME15

TKD-ALEX

FifteenGame è un remake del famosissimo [“Gioco del quindici”](#) sviluppato in Processing / Java utilizzando l’IDE di Processing 3.0.

“Il gioco consiste in una tabellina di forma quadrata divisa in quattro righe e quattro colonne (quindi 16 posizioni), su cui sono posizionate 15 tessere quadrate, numerate progressivamente a partire da 1. Le tessere possono scorrere in orizzontale o verticale, ma il loro spostamento è ovviamente limitato dall'esistenza di un singolo spazio vuoto. Lo scopo del gioco è riordinare le tessere dopo averle "mescolate" in modo casuale. La posizione da raggiungere per vincere è quella con il numero 1 in alto a sinistra e gli altri numeri a seguire da sinistra a destra e dall'alto in basso, fino al 15 seguito dalla casella vuota.”

Descrizione dell'interfaccia.

L'interfaccia del gioco è molto minimale e sono stati utilizzati principalmente tre colori:

Verde, **Azzurro,** **Grigio.**

Nella parte superiore è presente la “tabellina” di forma quadrata divisa in quattro righe e quattro colonne per un totale di 16 posizioni. Le varie tessere sono numerate con dei numeri che vanno da uno a sedici nel caso in cui si stia giocando con la modalità “numeri”.

Nella parte inferiore invece sono presenti i bottoni e i vari dettagli di gioco esclusivamente in lingua inglese.

I pulsanti sono per l'esattezza quattro:

1. **Image/Number Mode** Permette di cambiare la modalità di gioco da “numeri” a “immagine”. L'immagine predefinita è la famosissima [Lenna o Lena](#).
2. **Load Image** Permette di caricare un'immagine dal disco fisso.
3. **Restart** Riavvia il gioco iniziando una nuova partita.
4. **Quit** Esce dal gioco.

Le informazioni relative al gioco invece sono:

- **Total Time:** Visualizza il tempo trascorso nel formato min / sec / millisec dall'inizio della partita.
- **Total Moves:** Visualizza il numero totale di mosse effettuate.
- **Best Time:** Visualizza il miglior tempo espresso in secondi.
- **Best Moves:** Visualizza il numero minore di mosse effettuate.

Gli ultimi due dettagli possono anche essere denominati record e vengono salvati sul disco fisso.

Inoltre premendo sulla tastiera il tasto 'i' verrà visualizzato un messaggio contenente informazioni generali relative al gioco e al suo sviluppo.

Screenshot.



Il codice.

Tutti i codici presenti all'interno del file **.pde** sono commentati in lingua inglese dunque la comprensione del codice è molto semplice.

Classi.

La classe più importante è la classe **Box** e le sue derivate **BoxActive** e **BoxEmpty** ovvero le varie caselle che verranno poi gestite da metodi opportunamente sviluppati.

Entrambe le classi hanno come attributi:

- **number** Numero presente all'interno della casella.
- **location** Posizione della casella all'interno della nostra tabella.
- **x** Coordinata x della casella.
- **y** Coordinata y della casella.
- **s** Dimensione della casella. Viene utilizzata una sola variabile dato che le caselle sono quadrate.
- **c** Colore della casella.

I metodi presenti nelle classi (oltre al costruttore) sono:

- **selected()** Metodo che restituire un valore booleano nel caso in cui il mouse si trova sopra l'oggetto in questione.
- **display()** Il metodo display ha il compito di "disegnare" la nostra casella. Esistono più varianti del metodo display;
 - **display(int p)** Prende come parametro un intero utilizzato successivamente per colorare il quadrato disegnato tramite il metodo *rect*. Inoltre all'interno della casella viene scritto il corrispondente numero. Se il metodo in questione è utilizzato dalla classe *BoxEmpty* non viene scritto alcun numero all'interno della casella.
 - **display(PImage p)** Prende come parametro un'immagine che viene stampata tramite il metodo *image*. Nella classe *BoxEmpty* esiste un metodo denominato *displayWin* che funziona pressappoco alla stessa maniera.

Un'altra classe presente all'interno di questo progetto è la classe **myButton**. Non scenderò nei dettagli di questo oggetto poiché contiene come attributi le classiche informazioni (testo, dimensione, coordinate, etc.) e metodi standard (*selected*, *display*, etc.).

Variabili globali.

Nello sviluppo di quest'applicazione sono state utilizzate moltissime variabili globali con lo scopo di rendere tale codice "riutilizzabile" e "personalizzabile" nel caso in cui si voglia cambiare ad esempio la dimensione o il numero delle caselle, il colore dei bottoni e/o delle caselle e altri aspetti dell'interfaccia grafica. Purtroppo nel caso in cui si decida di cambiare il numero di box è necessario rivedere le regole di gioco gestite dal metodo che verrà descritto successivamente.

Le variabili globali più importanti sono:

- **backgroundColor, boxColor, textBox, buttonColor.**
Variabili di tipo *color*. Contengono le informazioni relative ai colori dello sfondo, delle caselle, del testo delle caselle e dei bottoni.
- **boxSize, boxMargin.**
Variabili di tipo *int*. Contengono le dimensioni di un box NxN e il margine tra un box e l'altro. In questo caso un quadrato è grande 96px ed è stato applicato un margine pari a 4 in modo tale da coprire 100px con ogni casella.
- **newEmptyLocation, newEmptyX, newEmptyY.**
Variabili di tipo *int* utilizzate successivamente per spostare il box vuoto.
- **totalMoves, bestTime, bestMoves.**
Variabili di tipo *int*. Contengono le informazioni dei record e dei punteggi attuali.
- **playing, gameOver, originalImg, useImage.**
Variabili di tipo *boolean*. Utilizzati come *flag* per gestire il gioco.
- **nRowsCols, nBoxes.**
Variabili di tipo *int*. Contengono rispettivamente il numero di righe/colonne e il numero totale di caselle generato dal quadrato di $nRowsCols - 1$.

- **img, imgArray[].**
Variabili di tipo *PImage*. La variabile *img* contiene l'immagine che verrà utilizzata nel nostro puzzle, *imgArray[]* invece contiene all'interno di esso i vari "pezzi" di *img*, esso ha come dimensione *nBoxes*.
- **boxes[], empty.**
Variabili di tipo *BoxActive*, *BoxEmpty*. L'array denominato *boxes[]* contiene *nBoxes* di tipo *BoxActive*.

Metodi.

- **setup()** All'interno del metodo *setup* vengono semplicemente inizializzate tutte le variabili gli array etc.
- **draw()** Il metodo *draw* riscrive ogni 15 frame rate (settati all'interno del metodo *setup*) tutti i dettagli di gioco e naturalmente aggiorna anche il timer.
- **displayDetailsGame() displayAllButton()**
Entrambi i metodi vengono richiamati all'interno di *draw*, essi non fanno altro che "disegnare" i vari testi e bottoni. Sono stati riscritti come metodi esterni a *draw* per una maggiore pulizia e chiarezza del codice.
- **shuffleBoxes()** Il metodo in questione crea e inizializza un array con valori da 1 a 15, successivamente all'interno di un ciclo *for* viene estratto (l'array a fine operazione viene shiftato) un numero casuale dall'array dichiarato prima e viene assegnato all'attributo *number* di una determinata cella di *boxes*. Infine viene chiamato il metodo *checkIsSolvable*.
- **checkIsSolvable()** Purtroppo tutte le permutazioni possibili non generano puzzle risolvibili. Tramite questo metodo vengono contati tutti i numeri minori di n_i e sommati tra di loro. Se la sommatoria è un numero dispari purtroppo la permutazione non è valida e bisogna generarne una nuova.

ATTENZIONE! Non tutte le permutazioni rappresentano una configurazione che prevede una soluzione. **In particolare esiste una soluzione se $p = r + \sum_{i=1}^{15} n_i$ è un numero pari.**

Se lo spazio vuoto si trova nella 1° o nella 3° riga $r = 1$ altrimenti $r = 0$. Per semplicità si può sempre creare una configurazione con la tessera vuota posta inizialmente in basso a destra (4° riga, cioè $r = 0$).

n_i è il numero di elementi che si trovano dopo di i ma sono minori di i (nell'array ottenuto linearizzando la matrice andando da sinistra a destra e dall'alto verso il basso).

Ad esempio, nell'array {5, 9, 8, 1, 15, 12, 4, 7, 13, 10, 3, 14, 2, 6, 11}, $n_2 = 0$ (non ci sono valori minori di 2 dopo il 2) mentre $n_{10} = 3$ (ci sono tre valori minori di 10 dopo il 10).

Bisogna quindi generare più configurazioni finché non se ne ottiene una che soddisfa la condizione sopra.

- **mouseReleased()** Viene controllato tramite il metodo *selected* quale cella viene cliccata e successivamente viene chiamato il metodo *playBox*.
- **mousePressed()** Sfrutta il metodo *selected* di tutti i bottoni e svolge le determinate opzioni assegnate ad ogni singolo pulsante.

- **playBox(int s)** Nel metodo seguente vengono gestite e applicate tutte le regole del gioco. Viene preso come parametro un *int* il quale permetterà di ricavare la locazione del box selezionato, dopodiché tramite degli *if* si controllerà la corretta posizione del box vuoto e si procederà con lo spostamento tramite il metodo *moveBox*. Infine viene controllato lo stato della partita richiamando *checkResult*.
- **moveBox(int s)** Anche il metodo *moveBox* riceve come parametro un *int* che ancora una volta avrà il compito di indicare su quale box lavorare. Tramite le variabili di supporto vengono scambiate le locazioni e coordinate del box vuoto con quello selezionato. Infine vengono incrementate le mosse.
- **checkResult()** All'interno del metodo *checkResult* vengono contate tutte le occorrenze *number*, *location* di tutti i singoli box e il numero è uguale a 14 la partita viene considerata come "vinta". Successivamente vengono bloccate tutte le caselle, aggiornati i punteggi e visualizzati i messaggi di "congratulazioni".
- **setNewScore()** Controlla se sono stati realizzati dei record ed eventualmente vengono aggiornati nel file *Scores.txt*.
- **fileSelected(File s)**
Prende in input il tipo *File*. Se il parametro passato non è nullo e il formato del file è supportato viene caricata l'immagine all'interno di *img*. Eventuali errori e/o ulteriori informazioni vengono comunicate tramite degli alert.
- **keyPressed()** Intercetta la pressione dei tasti sulla tastiera.
- **puzzleImage()** Il metodo *puzzleImage* ha il compito di "splittare" tutta l'immagine all'interno di un array di tipo *PImage* denominato *imgArray*. Come prima cosa l'immagine presente all'interno di *img* viene ridimensionata in modo tale che abbia le stesse dimensioni della nostra tabella, successivamente tramite il metodo *get* vengono prese delle "porzioni" di *img* avente dimensione uguale a *boxSize* e inserite all'interno dell'array.

Alessandro Maggio
X81000134
Tkd-Alex