



UNIVERSITÀ DEGLI STUDI  
DI PERUGIA

Tesina Finale di

## **Programmazione di Interfacce Grafiche e Dispositivi Mobili**

Corso di Laurea in Ingegneria Informatica ed Elettronica – A.A. 2020-2021

DIPARTIMENTO DI INGEGNERIA

docente  
Prof. Luca GRILLI

## **JMarioKart**

applicazione desktop JFC/SWING



**Studente**

313486    **Gelsi**    Alessandro    alessandro.gelsi@studenti.unipg.it

Data ultimo aggiornamento: 19 Luglio 2021

# Sommario

1. Descrizione del Problema .....	3
1.1 Super Mario Kart ( ed. 1992) .....	3
1.2 L'applicazione JMario Kart.....	4
2. Specifica dei Requisiti.....	5
3. Progetto .....	7
3.1 Architettura del Sistema Software.....	7
3.2 Model.....	9
3.3. View .....	11
3.4 Controller .....	15
3.5 Problemi Riscontrati.....	17
4. Conclusioni e Sviluppi Futuri .....	18
5. Bibliografia e Sitografia.....	19

# 1. Descrizione del Problema

Il progetto è volto allo sviluppo di un'applicazione grafica, denominata *JMarioKart*, che realizza una versione semplificata e minimizzata del videogioco *Super Mario Kart* storicamente diffuso su piattaforma SNES (Super Nintendo Entertainment System) a partire dal 1992.

L'applicazione sarà implementata utilizzando la tecnologia JFC/Swing in modo da favorire un'ampia portabilità su diversi sistemi operativi, riducendo al minimo eventuali modifiche al codice sorgente. Tuttavia, il codice prodotto sarà testato su piattaforma Windows OS.

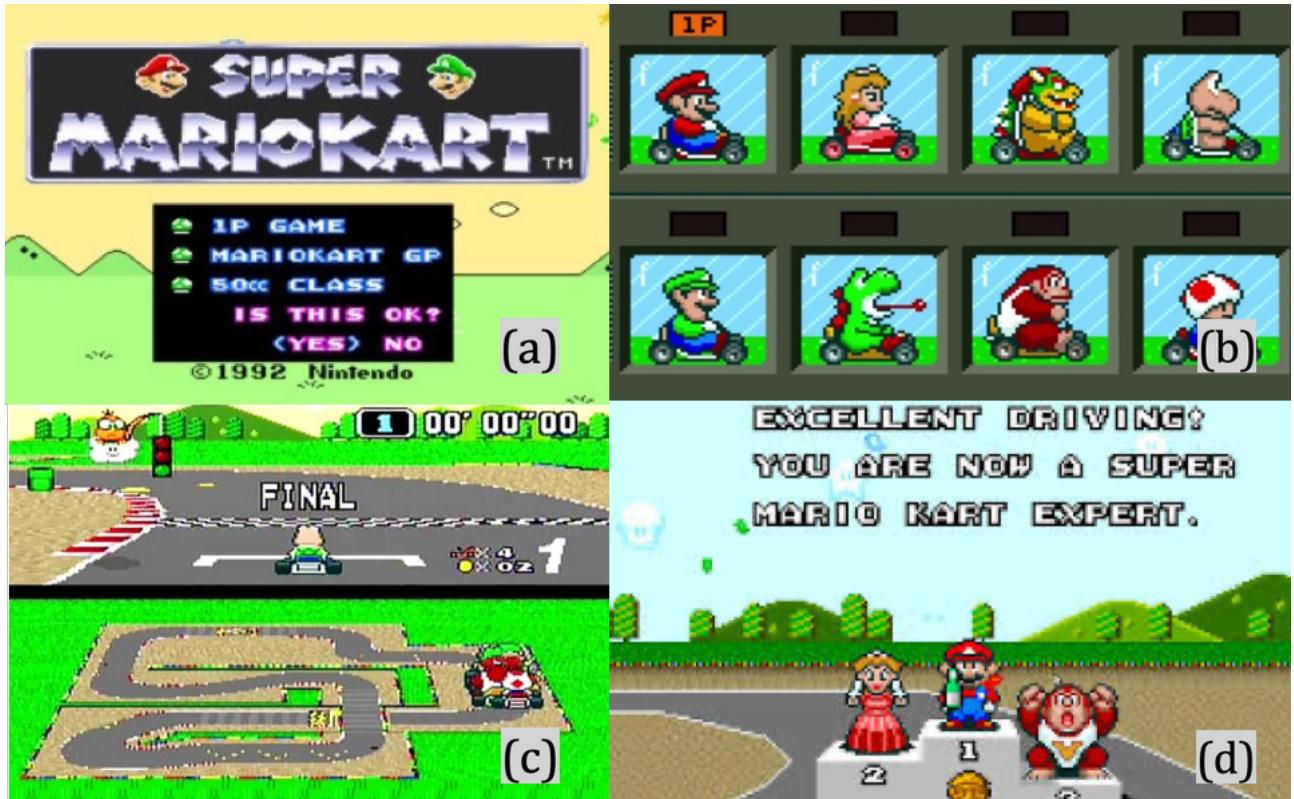
Di seguito sarà data una breve descrizione del videogioco originale, per poi fornire una descrizione della versione semplificata che si intende realizzare.

## 1.1 Super Mario Kart (ed. 1992)

Il gioco originale è considerabile un vero e proprio simulatore di guida immerso però nel mondo Nintendo di Super Mario. L'utente attraverso il controller della console SNES è in grado di pilotare il personaggio da lui scelto gareggiando con gli avversari in un circuito sempre a scelta del videogiocatore. Le gare sono in genere limitate ad un totale di 3 giri a seguito dei quali vi è una grafica relativa alla cerimonia di premiazione. L'obiettivo del giocatore è quello di piazzarsi in prima posizione. All'interno del percorso è possibile interagire con degli elementi che costituiscono potenziamenti o oggetti speciali estratti casualmente in grado di aiutare il pilota a tagliare per primo il traguardo (es. breve accelerazione istantanea, rallentamento veicoli avversari, ecc). I personaggi corrispondono ai personaggi iconici della saga di Super Mario e ad ognuno è generalmente associato un insieme di parametri (compresi tra 1 e 4) relativi a velocità massima, accelerazione e maneggevolezza del mezzo, che si ripercuotono sull'esperienza di guida in gara. Dal punto di vista grafico l'utente interagisce in un primo momento con il menù (*Fig.1(a)* di seguito) che gli consente di scegliere modalità, circuito e categoria. Le modalità disponibili sono giocatore singolo e multi-giocatore, in cui un secondo utente può partecipare alla stessa competizione. La scelta tra i circuiti è piuttosto ampia e in genere non va ad inficiare la prestanza di determinati kart rispetto ad altri. Infine è possibile selezionare la categoria di veicoli tra 50cc, 100cc e 150cc, rilevando un incremento della velocità e della maneggevolezza dei kart in gara. Una volta selezionate queste opzioni si apre la schermata di scelta del personaggio (*Fig.1(b)* di seguito) e dei relativi avversari. A seguito della scelta, l'utente si trova nell'interfaccia di gara, dove è possibile utilizzare la metà superiore dello schermo come schermata di guida comprendente timer, posizione e contagiri, mentre nella parte inferiore dello schermo è possibile tenere d'occhio la propria posizione all'interno del circuito e quella degli avversari. Al termine del terzo giro si genera la grafica relativa ai risultati della gara e relativa premiazione, chiedendo poi se si vuole iniziare un'altra gara.

Qui di seguito in *Figura 1* le quattro schermate del videogioco originale:

- Schermata di scelta della modalità di gioco e del circuito.
- Grafica relativa alla scelta del personaggio.
- Interfaccia di gioco, caratterizzata da circuito di gara, tempo sul circuito, numero di giri e posizione del kart sul circuito rispetto agli altri.
- Schermata finale di premiazione.



*Figura 1*

## 1.2 L'applicazione JMarioKart

L'applicazione JMarioKart si propone di offrire al giocatore un'interfaccia iniziale che gli permetta di selezionare in maniera intuitiva il personaggio e di inserire il proprio nome, per poi scendere in pista. L'esperienza di gioco risulta simile a quella precedentemente descritta, ma l'obiettivo dell'utente è realizzare il tempo migliore sul circuito. La grafica di guida è caratterizzata da una vista dall'alto in 2D che permette di visualizzare per intero tutto il circuito. Inoltre il giocatore è sempre in grado di visualizzare il proprio tempo attuale e il numero di giri su giri totali che sono stati effettuati. Quando si giunge al termine della gara si genera una grafica simile alla grafica di premiazione del gioco originale, viene presentato il tempo impiegato per completare il circuito ed è presente la possibilità di ricominciare un'altra gara.

## **2. Specifica dei Requisiti**

L'applicazione JMarioKart che si intende realizzare risulta caratterizzata da due sezioni principali. La prima è un'interfaccia di benvenuto in cui l'utente deve scegliere il proprio personaggio e inserire il proprio nome. La seconda è relativa alla gara e in questa il giocatore deve poter gestire i movimenti del kart nel circuito.

Per questo motivo i requisiti per comodità possono essere divisi in “Requisiti generali” e “Requisiti di gara”.

### **2.1 Requisiti generali**

1. L'interfaccia iniziale deve ricalcare la semplicità propria del gioco tradizionale, essendo quindi pulita ed intuitiva e gestibile attraverso l'utilizzo di mouse.
2. L'utente deve essere in grado di selezionare il personaggio desiderato da una lista che contiene e mostra tutti i disponibili.
3. L'utente ha la possibilità di impostare la scelta del personaggio in maniera casuale.
4. L'utente deve essere in grado di visualizzare un'anteprima animata corrispondente al personaggio selezionato prima di confermare la propria scelta.
5. Il programma deve permettere all'utente di inserire il proprio nome così da riproporlo nelle schermate successive, in modo tale da rendere l'esperienza più personale e inclusiva.
6. La schermata iniziale deve presentare i valori ordinati relativi ai migliori tempi ottenuti dall'utente nelle gare precedenti.
7. La permanenza nella schermata iniziale deve essere sempre accompagnata da un sottofondo musicale.
8. Dopo che l'utente ha confermato la selezione, verrà mostrata la schermata di gara nella stessa finestra.

### **2.2 Requisiti di gara**

1. La schermata di gara deve essere accompagnata per tutto il tempo da un sottofondo musicale.
2. L'interfaccia di gara deve mostrare sempre all'utente il tempo attuale, il numero di giri su giri totali e il suo nome.
3. Il giocatore deve gestire il movimento del kart attraverso il mouse, clickando dei pulsanti a forma di freccetta.
4. Il giocatore deve avere la possibilità di modificare la velocità del kart attraverso la pressione di due pulsanti, volti ad attuare un andamento più lento o più veloce.
5. La schermata mostra il circuito per intero attraverso una vista dall'alto.
6. Il movimento del kart nella pista deve essere fluido e avere un'elevata velocità di reazione ai comandi dell'utente.
7. Le animazioni del kart variano in base alla direzione in cui esso si muove (top, bottom, left e right).

8. Al termine del terzo giro la gara si interrompe e il timer si blocca. L'utente non ha più la possibilità di spostare il kart e nella schermata compaiono delle grafiche e animazione di festa.
9. Il programma mostra una grafica di premiazione quando al termine del terzo giro l'utente preme un pulsante che permette di abbandonare la schermata di gara per passare alla premiazione.
10. Il programma, a seguito della pressione del pulsante di uscita presente nella schermata di gara, apre nella stessa finestra una schermata di congratulazioni, che contiene il tempo appena totalizzato, i migliori tempi precedenti, il nome dell'utente e un'animazione di esultanza del kart precedentemente selezionato.
11. La schermata di congratulazioni è accompagnata da un sottofondo musicale, e permette all'utente di tornare nella schermata iniziale così da iniziare una nuova gara.

### **2.3 Requisiti opzionali**

1. Modalità “1vs1” selezionabile nel menù principale e caratterizzata dalla possibilità di aggiungere un ulteriore utente per dar vita ad una gara tra due kart.
2. Modalità “Torneo” selezionabile dal menù principale e che si presenta come susseguirsi di più gare su circuiti diversi. In base al tempo effettuato si guadagnano dei punti che vanno a sommarsi gara dopo gara, fino ad arrivare ad un punteggio definitivo dato dalla somma di tutti i precedenti.
3. Possibilità di scelta della soundtrack di gara tra una lista espandibile.
4. Possibilità di scelta della categoria del veicolo, con incremento della velocità e maneggevolezza del kart in gara in maniera analoga al gioco originale.
5. Assegnazione di specifiche diverse ad ogni personaggio, con differenziazione di velocità e maneggevolezza da personaggio a personaggio.

## 3. Progetto

In questa sezione viene descritta la struttura dell'applicazione, con un'illustrazione generale della struttura software e una descrizione più dettagliata dei blocchi funzionali che lo compongono.

### 3.1 Architettura del Sistema Software

JMarioKart è un'applicazione implementata seguendo il pattern MVC (Model View Controller). Il Model contiene lo stato dell'applicazione, quindi il kart selezionato con il rispettivo nome, la posizione corrente del kart all'interno del percorso, il record sul percorso e il nome del giocatore corrente. Il View si occupa di rappresentare i dati all'utente raccogliendone l'input: ha il compito di dare al giocatore la possibilità di scegliere il kart selezionato attraverso un'opportuna animazione di anteprima. Permette poi di inserire il proprio nome e di visualizzare i miglior tempi sul circuito. Permette anche all'utente di scegliere il kart in modo randomico e di procedere all'avvio della gara. Il Controller infine ha il compito di processare l'input dell'utente e agire di conseguenza, aggiornando lo stato nel Model e/o comunicando con l'utente tramite la View. In figura 2 è riportata una semplice rappresentazione dell'applicazione, in cui sono evidenziate le associazioni tra le classi e le direzioni flussi informativi.

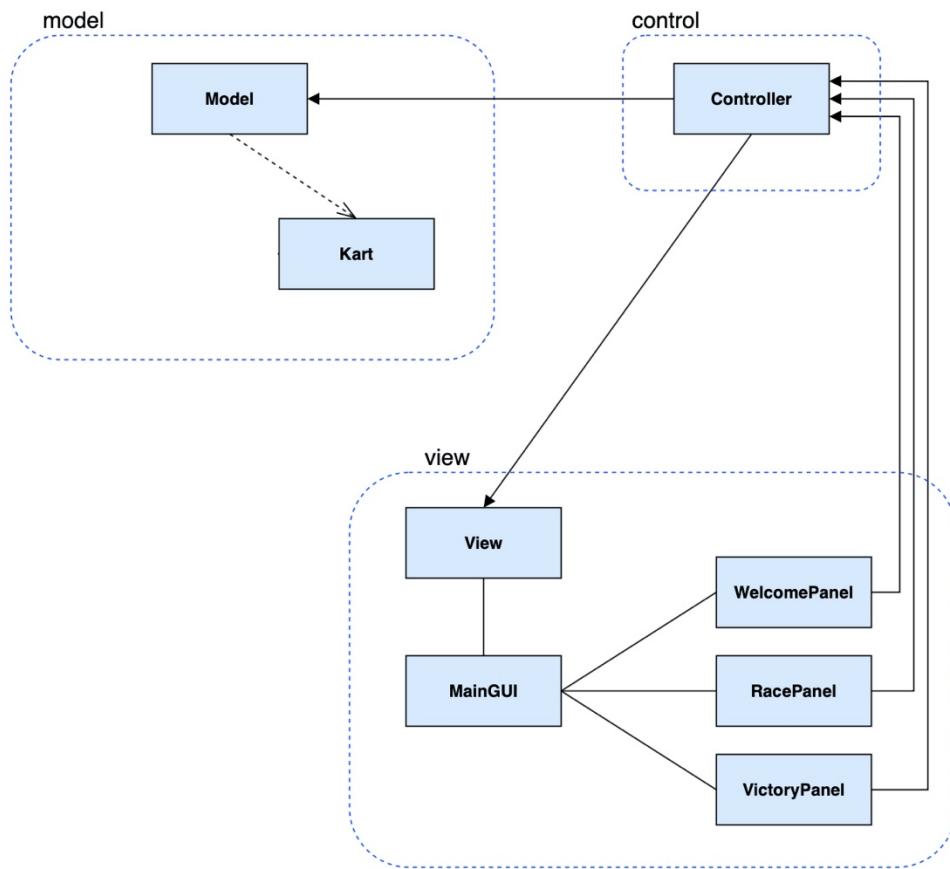
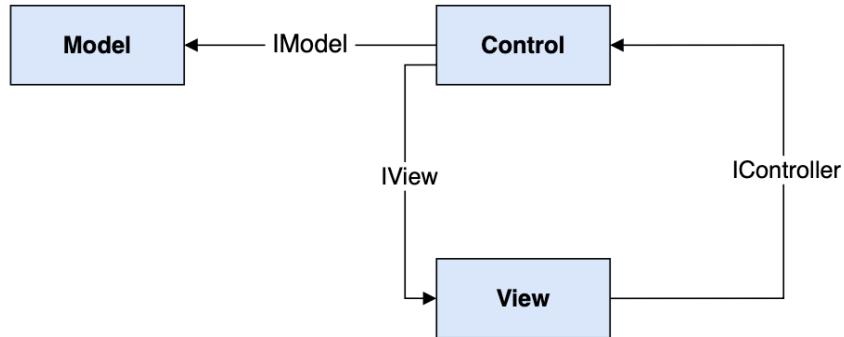


Figura 2: rappresentazione delle classi

In *figura 3* è possibile osservare una semplificazione dell'architettura, con l'obiettivo di evidenziare in modo più chiaro i flussi informativi e le interfacce esposte rispettivamente da Model, Control e View.

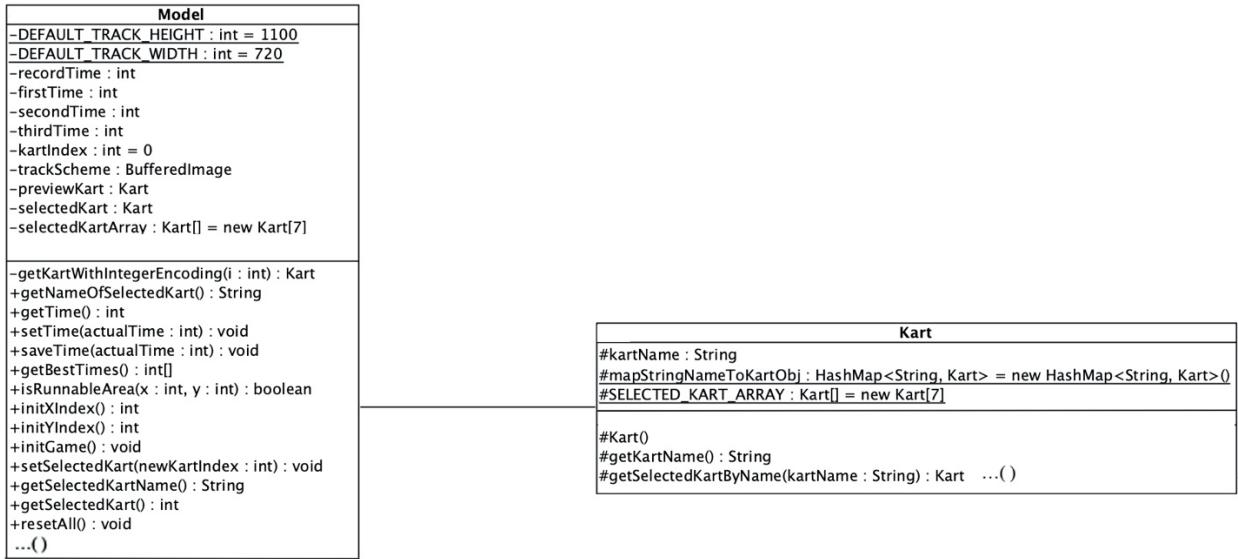


*Figura 3:* rappresentazione di flussi e interfacce

Nelle rappresentazioni precedentemente riportate sono stati omessi la classe “Main”, da cui è solamente lanciata la GUI, la GUI, la classi che estendono Kart e che vanno a differenziare le caratteristiche dei singoli kart, e il package “utils”, contenente la classe “Config”. Questo è inserito nell’architettura del software con l’obiettivo di raggruppare in esso le comunicazioni con il file system, necessarie all’applicativo per reperire musiche, sprites, sfondi e animazioni.

## 3.2 Model

Le classi appartenenti al blocco Model di JMarioKart si trovano nel package “model”. La struttura del blocco è rappresentata nel diagramma UML in *figura 4*. Si noti che il seguente diagramma, per ragioni di spazio e per mettere in evidenza le parti più interessanti del codice, non riporta tutti le variabili e metodi appartenenti alle classi del pacchetto, ma solamente quelli che rappresentano il cuore dello stesso e risultano dunque più significativi.



*Figura 4:* diagramma UML sintetico delle classi appartenenti al model

Dove in particolare:

- **Kart:** rappresenta un generico kart e le sue variabili costituiscono le sue caratteristiche, tra cui il nome del kart selezionato. Tale scelta è stata effettuata col fine di poter ampliare molto facilmente la scelta di kart presenti nel gioco. Tra ai metodi per l'ottenimento di informazioni sul kart spiccano i metodi che permettono di associare un indice numerico ad un determinato tipo di kart. Le varie scelte sono implementate attraverso classi specifiche che estendono Kart e che risultano rinominate con il nome del kart che rendono funzionale la possibilità di implementazione di ulteriori caratteristiche, quali ad esempio differenze di velocità o di manovrabilità.
- **Model:** contiene lo stato del gioco, rappresentato dal kart selezionato per la gara, (associato all'indice intero dell'array `selectedKart[]`), dalla posizione del kart nel percorso, dal nome inserito dall'utente e dal tempo dell'utente sulla pista. Vi sono dunque i metodi che permettono di fornire informazioni relative allo stato e i metodi che permettono di modificarlo. Il metodo **“`setSelectedKart(int newKartIndex)`** permette la comunicazione, intermediata dal Controller, dalla View al Model, delle scelte effettuate dall'utente in fase di selezione del personaggio selezionato. Il metodo si avvale poi di **“`getKartWithIntegerEncoding(int i)`** che associa un determinato `kartName` ad un indice intero che gli viene passato attraverso il tramite del Controller. Il metodo utilizza inoltre il metodo **“`getSelectedKartByName(String kartName)`** della classe `Kart( )`, il quale tramite un hashmap associa una `String` a un kart. Nel Model sono contenuti i metodi che

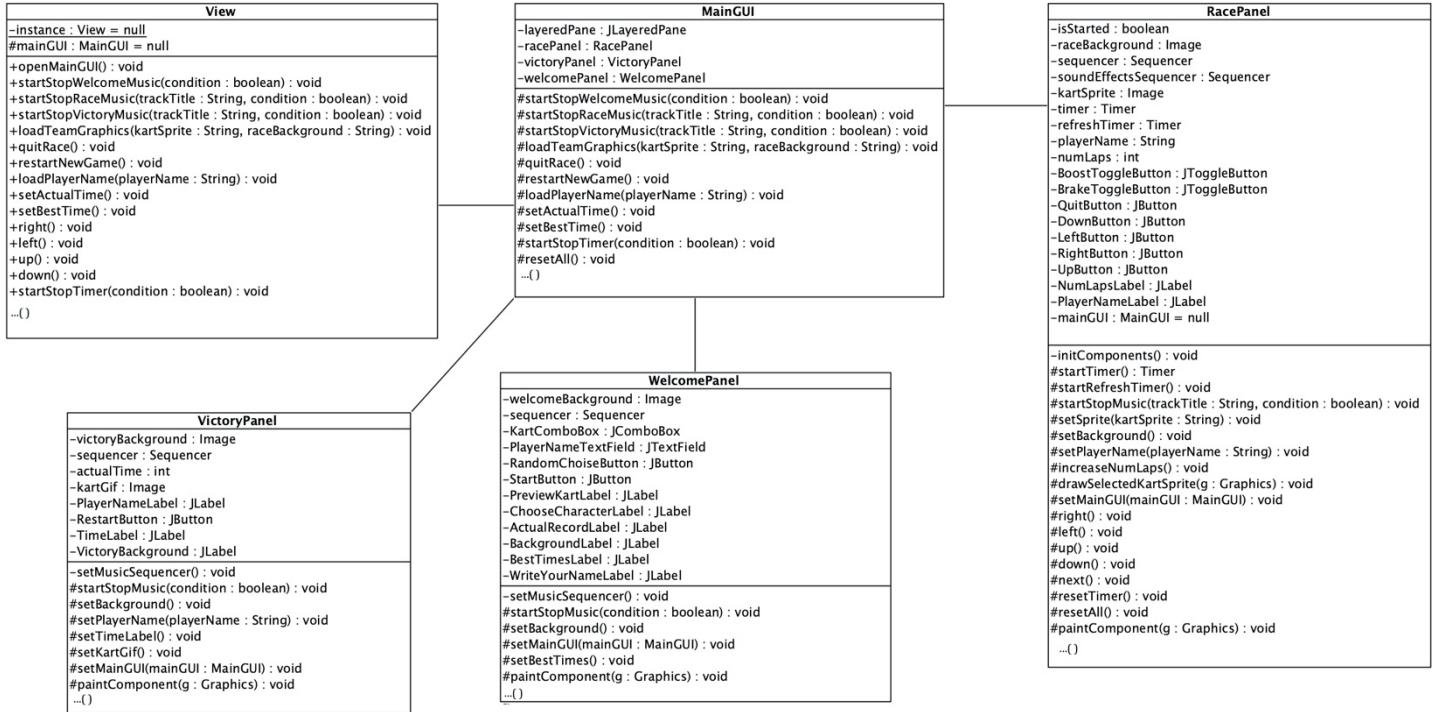
permettono di modificare e di ottenere il nome del giocatore e di salvare il tempo sulla pista.

Durante la fase di gara l'utente è chiamato a scegliere la direzione in cui il kart si muove automaticamente all'interno della mappa. Proprio per questo lo stato del programma è in continua evoluzione, ed è governato dai metodi “**setXIndex (int x)**”, “**setYIndex ( int y)**” e “**isRunnable ( int x, int y)**”. Tali metodi vengono richiamati continuamente attraverso il Controller quando il kart nell'interfaccia di gara è in movimento. I primi due metodi sono utilizzati per modificare lo stato della posizione relativa del kart all'interno del percorso, a livello logico. Lo stato è in continua evoluzione e questa avviene in modo parallelo sia nell'interfaccia grafica che a livello logico nel Model. Il metodo “**isRunnable ( int x, int y)**” risulta particolarmente importante in quanto in fase di gara permette di mantenere il kart all'interno del percorso, attuando una collision detection vincolata ai lati della corsia di gara. Si avvale dell'utilizzo di un'immagine caratterizzata dall'avere pixel neri nelle zone del circuito permesse al kart, e pixel bianchi in quelle non percorribili. Logicamente il pixel di colore nero è associato ad una zona corrispondente graficamente alla zona di percorso, mentre i pixel bianchi a zone che sono al di fuori di esso.

Il metodo restituisce un valore booleano che rappresenta la percorribilità o meno di una determinata posizione, effettuando un confronto fra i valori RGB relativi al pixel dell'immagine schema e i valori RGB bianco e nero di riferimento. Questo è possibile attraverso l'utilizzo del metodo `getRGB(int x, int y)` proprio della classe `BufferedImage`.

### 3.3. View

Si riporta qui di seguito la struttura del blocco “View”, coincidente con il package “view”. Si noti anche in questo caso, che il seguente diagramma UML (*Figura 5*) non riporta tutti le variabili e i metodi appartenenti alle classi del pacchetto, ma solamente quelle più significativi.



*Figura 5:* diagramma UML sintetico delle classi appartenenti al view

- **MainGUI:** estende JFrame e rappresenta l'unica finestra all'interno della quale l'utente utilizza il programma. Avendo la necessità di visualizzare tre fasi del gioco, ovvero fase di selezione, gara e vittoria, si è fatto uso di un JLayeredPane annesso al JFrame, in modo tale da poter gestire i tre panelli (WelcomePanel, RacePanel e VictoryPanel) su livelli diversi e renderli visibili o meno a seconda della fase di gioco.
- **WelcomePanel:** estende JPanel e rappresenta la fase di selezione del nome del giocatore del kart. Questi ultimi sono selezionabili dall'utente attraverso la JComboBox KartComboBox e rappresentati nel JPanel attraverso un animazione di anteprima. Inoltre, tramite la pressione del RandomChoiseButton, si ha la possibilità di impostare il kart selezionato in modo casuale. Dato che l'utente può cambiare la propria selezione innumerevoli volte prima di iniziare una battaglia, è stato realizzato un sistema che aggiorna il Model, attraverso Controller, comunicandogli le scelte dell'utente, solo una volta che questo avvia la gara tramite la pressione dello StartRaceButton. Infatti da quel momento in poi le scelte dell'utente si concretizzano effettivamente in componenti grafiche del kart e del nome utente da caricare poi nel RacePanel. Inoltre i metodi “`setBackground( )`, `setMusicSequencer( )`, `startStopMusic(boolean condition)`”

permettono di caricare e far visualizzare la schermata iniziale con il giusto background e con l'accompagnamento sonoro e musicale.

- **RacePanel:** estende JPanel e rappresenta la fase di gara. Mostra all'utente lo stato attuale della competizione, quali la posizione delle grafiche relative al kart selezionato, il tempo attuale, il numero di giri su giri totali e il nome del giocatore precedentemente inserito. Il giocatore, attraverso la pressione dei tasti UpButton, DownButton, RightButton e LeftButton è in grado di scegliere la direzione in cui il kart si continua a muovere in modo automatico. Alla pressione dei bottoni di movimento è associata una quaterna di valori booleani, che vengono modificati per indicare effettivamente la direzione in cui il kart si sta muovendo. L'esecuzione avviene richiamando prima Controller, in quanto è necessario valutare che la posizione in cui il kart si sta per spostare sia percorribile e all'interno del percorso. La verifica va eseguita logicamente e deve essere aggiornato lo stato del kart sul circuito. L'aggiornamento della posizione della grafica nell'interfaccia è poi gestita direttamente all'interno del RacePanel, che rispecchia però fedelmente lo stato rappresentato nel Model. La grafica può così spostarsi nella direzione corrispondente a quella selezionata, attraverso l'utilizzo del metodo **next()**. Questo implementa il movimento nella direzione corretta attraverso la lettura della quaterna di valori booleani precedentemente citati. Il metodo **next()** a sua volta si avvale dei metodi **Up()**, **Down()**, **Left()**, **Right()**. Questi attuano concretamente il movimento in quanto, dopo aver verificato la percorribilità della futura posizione invocando il Controller, aggiornano la posizione del kart e modificano lo stato nel Model sostituendo i precedenti valori relativi alla posizione attuale con i nuovi. Sono inoltre presenti due JToggleButton, denominati BoostToggleButton e BrakeToggleButton. Questi due pulsanti permettono all'utente di modificare l'attuale velocità del kart, andando così a modificare direttamente la quantità di pixel che sono incrementati ad ogni istante temporale. I due JToggleButton possono essere utilizzati per ottenere tempi migliori sul percorso, e l'utente in seguito ad una prima pressione, deve effettuarne una seconda se vuole ritornare alla velocità normale del kart. I metodi che si occupano delle animazioni di kart e del cronometro di gara utilizzano un oggetto Timer memorizzato in una variabile d'istanza e con tempi di aggiornamento variabili a seconda del tipo di animazione. Concretamente la frequenza di aggiornamento del Timer refreshTimer va a gestire l'effettiva velocità e fluidità del kart nella pista. Inoltre vi è un metodo **increaseNumLaps()** che modifica le variabili che rappresentano il numero di giri direttamente nel WelcomePanel, incrementando numLaps quando le coordinate che rappresentano la posizione del kart attraversano quelle dei pixel relativi al traguardo.
- **VictoryPanel:** ha il compito di comunicare all'utente il proprio tempo sul circuito al taglio del traguardo e di mostrare il posizionamento in classifica. Presenta inoltre anche un'animazione che riprende quella del personaggio selezionato e riprende anche il nome del giocatore, precedentemente inserito nel WelcomePanel. Inoltre è presente il JButton RestartButton, la cui pressione permette all'utente di tornare al menù iniziale, resettando il tutto e salvando il valore del tempo sul percorso, così da verificare il piazzamento in classifica.

- **View:** attraverso l'interfaccia **IView**, permette al Controller di eseguire operazioni grafiche sulla finestra mostrata all'utente. Tra i metodi più significativi vi sono **loadGraphics(...)**, **startStopMusic(...)**, **up(...)**, **down(...)**, **left(...)**, **right(...)**, **setActualTime(...)**. Il metodo **loadGraphics(String kartSprite, String raceBackground)** ha il compito di passare al **RacePanel** le informazioni riguardanti la scelta fatta dall'utente relativamente al kart, inoltre carica inoltre l'immagine dello sfondo. Il metodo **startStopMusic( String tracktitle, boolean condition)** gestisce nei vari pannelli il caricamento e l'interruzione della riproduzione della musica di sottofondo. I metodi **up()**, **down()**, **left()**, **right()** richiamano i metodi del **RacePanel** relativi allo spostamento della grafica del kart nel percorso. Infine il metodo **setActualTime(...)** permette di caricare nel **VictoryPanel** il tempo precedentemente conseguito al taglio del traguardo.

Di seguito vengono riportate delle immagini dell'applicazione in esecuzione, rispettivamente della fase di selezione, gara e schermata di vittoria.



Figura 6: Schermata di selezione (WelcomePanel)

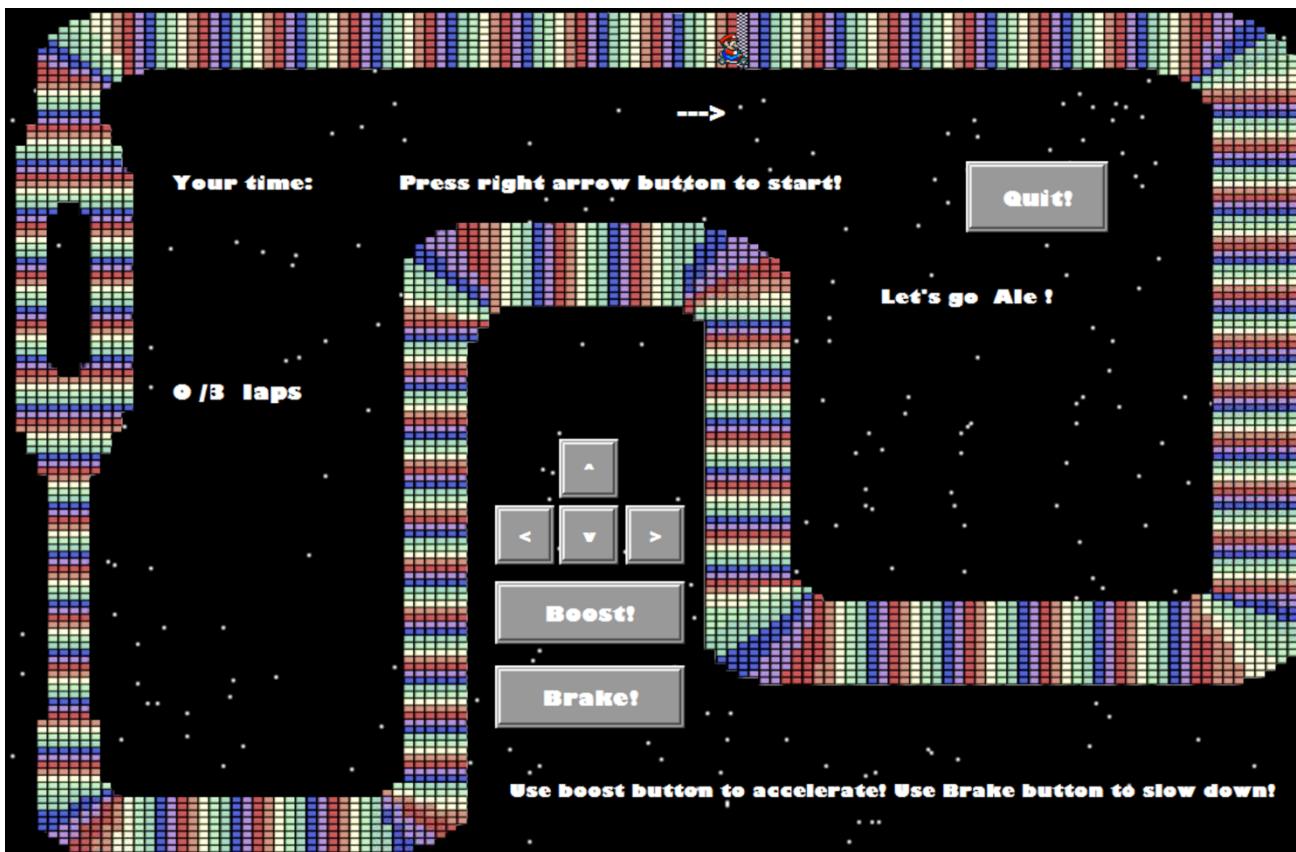


Figura 7: Schermata di gara (RacePanel)

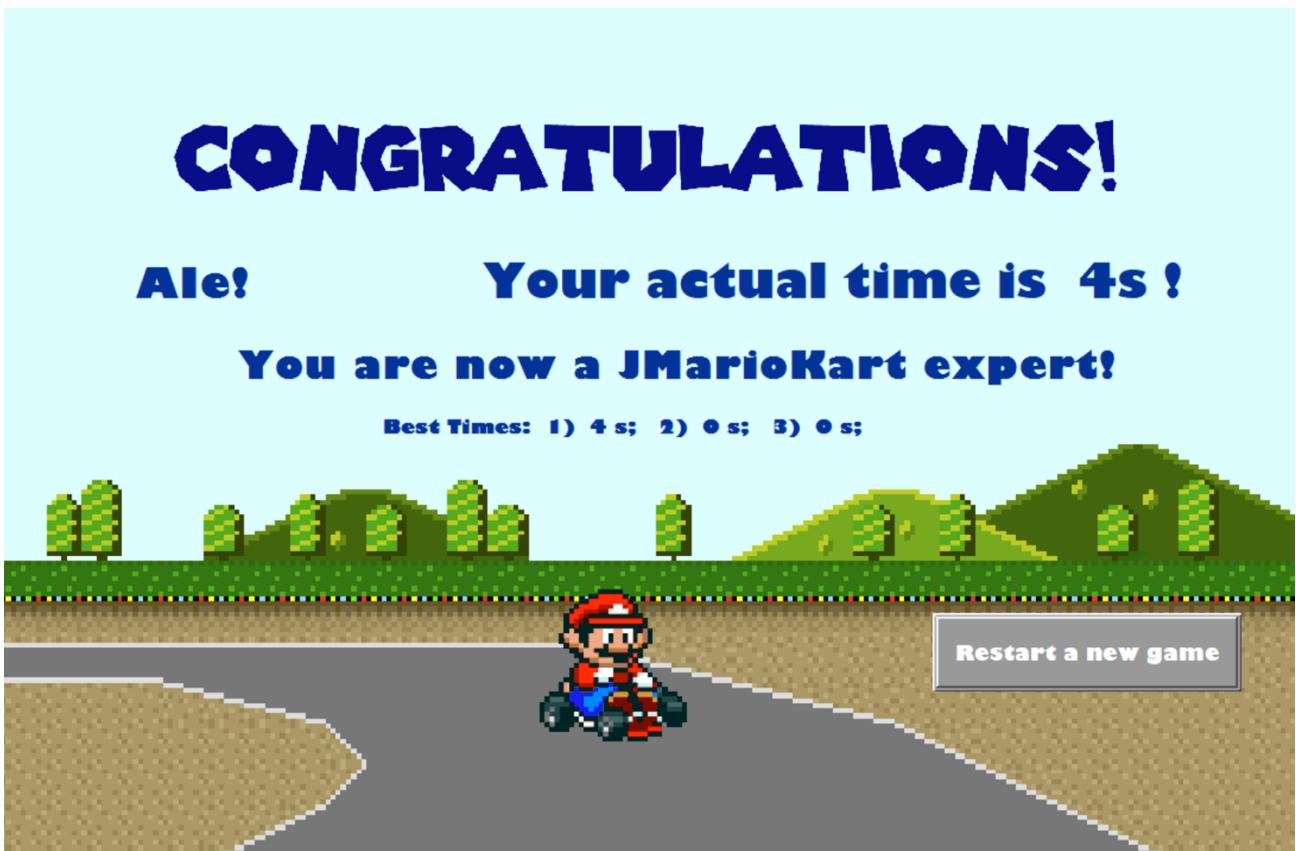


Figura 8: Schermata della vittoria (VictoryPanel)

### 3.4 Controller

Si riporta ora la struttura del blocco “Controller”, coincidente con il package “controller”. Di seguito non vengono trattate tutte le variabili e i metodi appartenenti alle classi del pacchetto, ma solamente quelli principali.

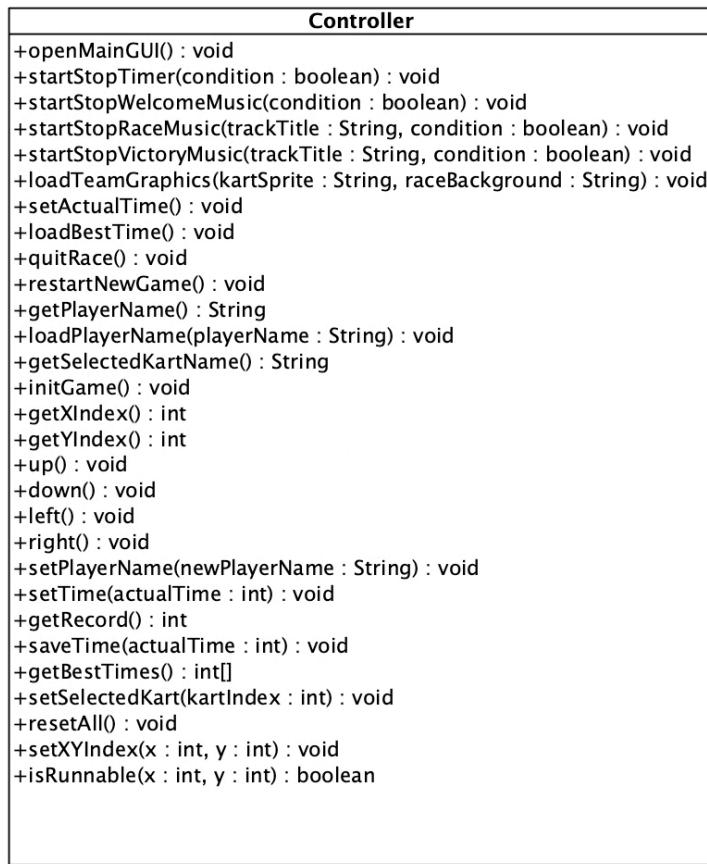


Figura 9: diagramma UML sintetico delle classi appartenenti al controller

Il Controller è invocato per modificare lo stato del Model in seguito ad azioni dell’utente, e per eseguire operazioni di aggiornamento della vista grafica (caricamento componenti, animazioni, ecc.). Strutturalmente è implementazione dell’interfaccia denominata IController.

Relativamente ai metodi implementati dal Controller, quelli maggiormente degni di nota sono **loadGraphics(...)**, **loadPlayerName(...)**, **isRunnable(...)**, **loadBestTimes(...)**, **startStopTimer(...)**, **getSelectedKartName(...)**, **setXYIndex(...)**, **setSelectedKart(...)**, **setPlayerName(...)** e **setTime(...)**. Analizzandoli più nel dettaglio:

- **loadGraphics(String kartSprite, String raceBackground):** permette al WelcomePanel di caricare le componenti grafiche richiamando l’analogo metodo del View. Le informazioni sulla selezione effettuata vengono recuperate dal Model.
- **loadPlayerName( String playerName):** invocato quando l’utente, attraverso la vista grafica, decide di procedere alla schermata di gara. Questo permette dunque di caricare

nel racePanel e nel victoryPanel il nome inserito dal giocatore così da restituire un'esperienza più personale.

- **isRunnable(int x, int y):** si occupa di comunicare al View la percorribilità o meno del pixel selezionato, restituendo un valore booleano, interrogando il Model attraverso il metodo `isRunnable(x, y)` del Model stesso. Questo come precedentemente riportato, valuta i valori posizionali fornendo informazioni sulla percorribilità o meno di un pixel.
- **loadBestTimes( ):** è invocato al termine della gara e si occupa direttamente del caricamento grafico nelle apposite label dei valori relativi ai migliori tempi sul percorso, direttamente reperiti dalle informazioni nel Model.
- **startStopTimer ( ):** viene richiamato nel momento in cui l'utente in fase di gara effettua per la prima volta la pressione sul RightButton, innescando il movimento automatico del kart. Insieme al kart parte dunque anche il Timer responsabile del refresh delle sprite e che si bloccherà al termine del terzo giro.
- **getSelectedKartName( ):** restituisce alle classi che gestiscono la grafica il nome del kart selezionato. Il suo utilizzo è simbiotico a quello del metodo **loadGraphics( )** e permette il corretto caricamento delle grafiche relative al kart selezionato.
- **setXYIndex( int x, int y ):** quando il kart in movimento è il responsabile del continuo aggiornamento delle coordinate che descrivono la sua posizione attuale e modifica i valori dello stato presenti nel Model ;
- **setSelectedKart( int kartIndex):** aggiorna nel Model lo stato che descrive tramite un indice il kart selezionato.
- **setPlayerName(String playerName):** aggiorna nel Model la variabile dello stato che rappresenta il nome inserito dall'utente.
- **setTime( int actualTime):** aggiorna nel Model il tempo relativo al termine dell'attuale gara, aggiornando la variabile del Model `actualTime` e rendendo agevole poi il salvataggio tra i tempi migliori e il caricamento di questa attraverso il Controller nelle grafiche del VictoryPanel.

### 3.5 Problemi Riscontrati

Durante lo sviluppo di JMarioKart sono stati incontrati diversi problemi, relativi principalmente a aspetti di tipo grafico e logico.

In particolare:

- Nel WelcomePanel l'utente ha la possibilità di selezionare il kart che preferisce tra una lista di sette differenti modelli. Per rendere agevole e flessibile la selezione e la comunicazione al Model della propria scelta si è fatto uso di una hashMap che associa ai nomi del kart il relativo Kart Object, fornendo la possibilità di accedere all'elemento attraverso il nome, recuperato dalla JComboBox attraverso il metodo getSelectedItem() della classe JComboBox. La soluzione rende più semplice in caso di un eventuale sviluppo futuro la possibilità di reperire le informazioni relative al singolo kart attraverso il solo nome.
- In un primo momento il RacePanel risultava completamente inerte alle azioni e agli eventi che avrebbero dovuto verificarsi a seguito della pressione su tastiera, associata al relativo KeyListener. A seguito di un'opportuna ricerca si è scoperto che è impossibile intercettare gli eventi di pressione della tastiera attraverso l' implementazione di un JLayeredPane, in quanto gli eventi vengono inviati al componente più radicato e non a quello in focus. Una possibile soluzione è l'utilizzo di un'altra strategia per intercettare gli eventi che è basata sui KeyBindings. Questa permette di associare una determinata azione ad un input da tastiera attraverso la realizzazione di un inputMap. Tuttavia la soluzione che è sembrata più efficace e funzionale è quella che si avvale dell'utilizzo di JButton e del metodo next(...) per modificare le direzioni di movimento del kart con una semplice pressione del tasto attraverso il mouse.
- Nelle Label si ha spesso necessità di andare a capo con le String da mostrare o comunque di riempire interamente lo spazio occupato dal testo, cosa non permessa utilizzando i caratteri speciali come “\n”. A seguito di una ricerca nelle API è stato scelto di utilizzare codice html che è proprio dell'interno di queste componenti. L'utilizzo di <html> risulta in grado di supportare il "<br/>" rendendo possibile l'impaginazione richiesta.
- L'implementazione della collision detection in fase di gara risulta uno degli aspetti più importanti nella giocabilità e fruibilità del gioco. A causa della presenza di sezioni del percorso particolarmente strette in cui era necessaria un'elevata precisione nel definire la parte di percorso percorribile o meno dal kart, l'intuizione è stata di sfruttare uno schema grafico parallelo al percorso effettivamente visibile all'utente. Si è optato per l'utilizzo di un track-scheme in cui il circuito è stilizzato ad un'immagine bicolore, dove il colore nero ricopre la parte percorribile, mentre la zona bianca è relativa alla parte non percorribile. In fase di gara vi è una continua verifica che i pixel in cui il kart si muove siano corrispondenti a pixel di colore nero nel track-scheme, garantendo così un'elevata precisione dovuta alla diretta corrispondenza tra i pixel del percorso che l'utente vede nell'interfaccia di gioco e i pixel appartenenti allo schema in bianco e nero.
- In fase di sviluppo si notava che in determinate circostanze, quando solitamente quando il boost era attivo, non avveniva l'incrementazione del numero di giri, teoricamente

concretizzata dal metodo increaseNumLaps( ) della classe RacePanel del package View. Il metodo in questione andava a verificare in modo diretto se il kart attraversava una determinata porzione di pixel, andando ad incrementare il numero di giri fino ad un massimo di tre. Dopo un'attenta analisi, si è dedotto che la causa di questo problema era la mancata coerenza tra la larghezza in pixel associati al traguardo e il valore dei pixel che venivano incrementati ad ogni refresh del timer. Nel caso in cui il boost era attivo, il numero di pixel incrementati ad ogni refresh del timer era tale da provocare un salto dei valori relativi alla posizione del kart. L'incremento causava il salto della zona sensibile al taglio del traguardo, non permettendo al contatore di salire. La soluzione scelta tra le possibili consiste nella realizzazione di una zona sensibile al taglio del traguardo di dimensione flessibile, non più fissa. La lunghezza variabile è dipendente dall'attuale valore di incremento dei pixel, e permette l'opportuna cattura del taglio del traguardo, evitando il salto di questo.

- Le animazioni, basate su sprite-sheet contenute tra le risorse, sono state abbastanza impegnative. La più significativa è quella relativa al movimento del Kart nel circuito. È risultato necessario trovare un giusto equilibrio tra il delayTime del refreshTimer e l'incremento di pixel nei metodi Up( ), Down( ), Left( ), Right( ). Il refreshTimer gestisce la frequenza di aggiornamento della posizione del kart nel circuito. I metodi Up( ), Down( ), Left( ), Right( ) attuano l'incremento intero della posizione in pixel del kart nella pista. La difficoltà è stata riscontrata nel garantire un'animazione fluida in accordo ad una velocità effettiva del kart tale da rendere l'esperienza fruibile e non eccessivamente difficile per qualsiasi tipologia di utente. Un incremento eccessivo dei pixel per ogni refresh del timer avrebbe causato fenomeni di buffering e andamento a scatti.
- Le GIF animate utilizzate durante la scelta nel WelcomePanel e nel VictoryPanel soffrono di sporadici problemi di flickering, che talvolta si verificano in quanto non gestite bene da Java, e nonostante il doubleBuffering dei pannelli. Effettuando alcune ricerche si è evidenziato che l'origine di questo problema è probabile si identifichi in una gestione non ottimale di tali file da parte di Java, soprattutto nel caso di GIF aventi un elevato numero di frame ripetuti.

## 4. Conclusioni e Sviluppi Futuri

JMarioKart si presenta come un applicativo intuitivo e fruibile da qualsiasi tipo di utente. La scelta dei personaggi e l'inserimento del proprio nome sono estremamente semplici. Il gioco, riportando il nome del giocatore in ogni schermata, vuole creare un'esperienza più personale, unica e inclusiva. La possibilità di scelta casuale del kart permette anche all'utente di partecipare a gare diverse tra loro per grafiche e animazioni, garantendo contemporaneamente la possibilità di scegliere quelle che si preferiscono e affrontando se stesso cercando di superare i propri record sulla pista. Il progetto ha anche come possibile sviluppi futuri ad esempio la possibilità di ampliamento del numero di percorsi disponibili, l'aggiunta di caratteristiche proprie di ciascun Kart (velocità, manovrabilità, ecc.) e la scelta personalizzata di sfondi e

musiche. La scelta tra nuovi circuiti, sfondi e musiche è implementabile attraverso un codice parallelo e con la stessa struttura di quello utilizzato per la scelta del kart e del playerName. Le nuove caratteristiche, differenziate per ciascun kart, sono facilmente implementabili sfruttando le rispettive estensioni della classe Kart. Altre possibili evoluzioni future vedono la concretizzazione dei rimanenti “obiettivi opzionali”, ovvero ad esempio la possibilità di scelta della categoria del veicolo con rispettivo incremento differente delle velocità. Un altro possibile sviluppo consiste nella creazione di diverse modalità di gioco, come le 1 vs 1 e la modalità Torneo. La logica della prima opzione è essenzialmente quella di una gara testa a testa dove vince chi taglia il traguardo per primo, in grado di funzionare sia con un secondo utente sia in modo tale per cui il secondo concorrente è guidato da scelte della cpu. La seconda opzione, relativa alla modalità torneo, è invece caratterizzata dalla scelta iniziale di un determinato numero di circuiti, che andranno poi a susseguirsi generando un tempo complessivo finale.

JMarioKart risulta dunque in conclusione una riproduzione semplificata dell'originale simulatore Nintendo e ha come punti di forza la semplice e intuitiva giocabilità, la fluidità e la coerenza complessiva di tutti gli aspetti grafici.

## 5. Bibliografia e Sitografia

- [1] "SpritersResource ", disponibile al: <https://www.spriters-resource.com/>
- [2] "MarioWiki", disponibile al: [https://www.mariowiki.it/Mario\\_Kart\\_\(serie\)/](https://www.mariowiki.it/Mario_Kart_(serie)/)
- [3] "MarioKartFandom", disponibile al:  
[https://mariokart.fandom.com/wiki/Mario\\_Kart\\_Racing\\_Wiki/](https://mariokart.fandom.com/wiki/Mario_Kart_Racing_Wiki/)