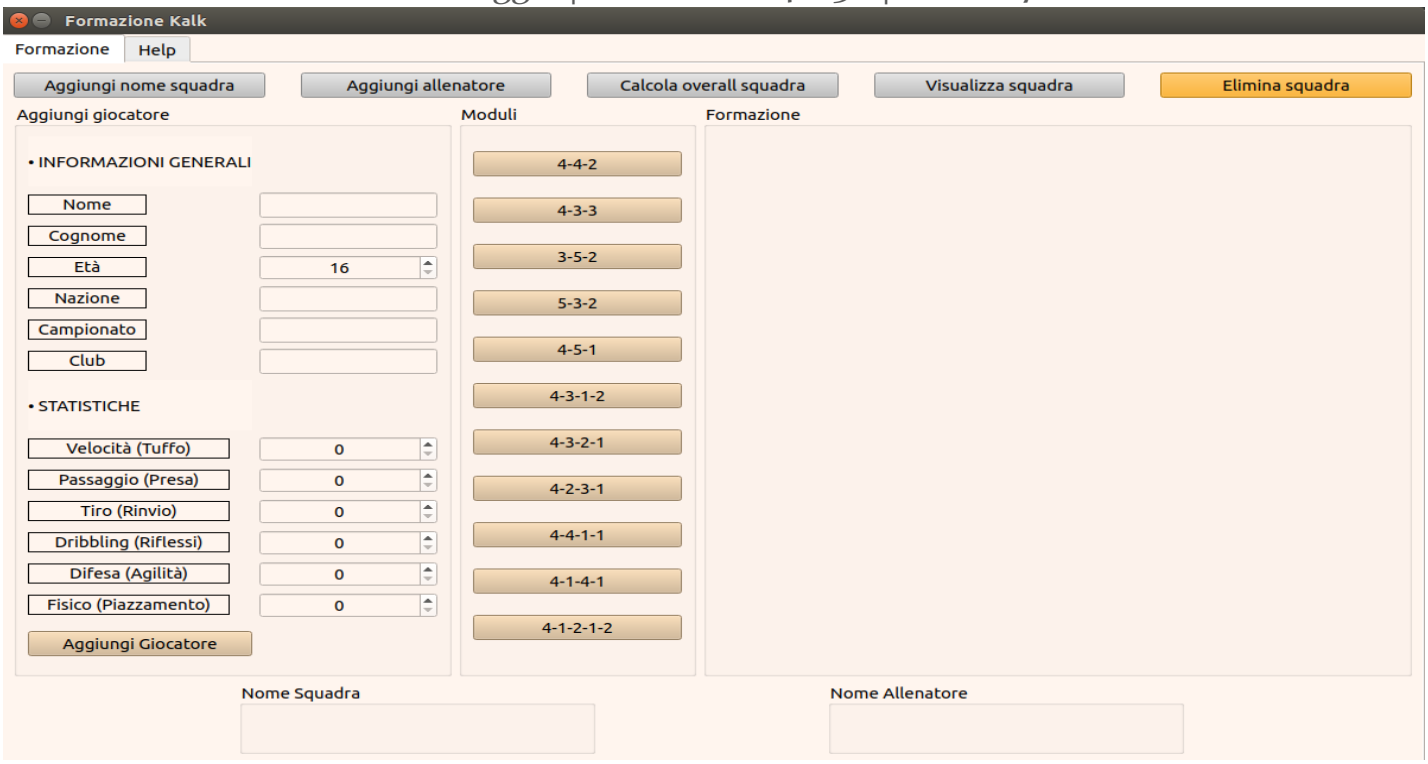
Progetto FormazioneKalk

PROGRAMMAZIONE AD OGGETTI

Nicolò Tartaggia | Matricola 1142836 | A.A 2017/2018



Introduzione

Il progetto FormazioneKalk è un'applicazione che permette la creazione di una formazione di calcio seguendo lo stile del famoso gioco FIFA, prodotto da EA.
Ciò significa che la creazione della formazione prevede, innanzitutto, la creazione dei vari giocatori che ne faranno parte, impostando le varie informazioni generali (nome, cognome, età ecc.) e le varie statistiche (velocità, tiro, passaggio ecc.) di ognuno.
Inoltre, una formazione possiede un nome e un allenatore ed è disposta in campo in base al modulo scelto.
In generale, FormazioneKalk prevede diverse operazioni :
- l'aggiunta di un allenatore;
- l'aggiunta di un nome di squadra;
- l'aggiunta di un modulo;
- l'aggiunta di un giocatore;
- il calcolo dell'overall di un giocatore;
- il calcolo dell'overall complessivo di squadra;
- l'eliminazione dell'intera squadra.

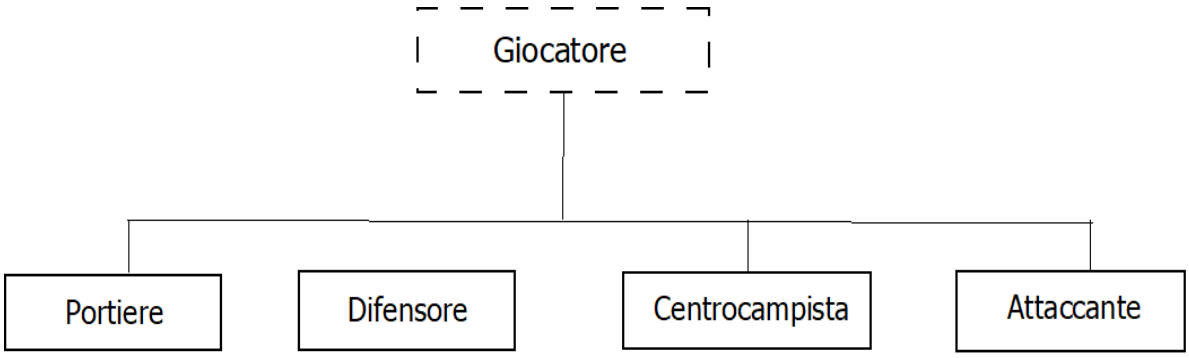
CREAZIONE DI UN GIOCATORE

Seguendo lo stile del gioco a cui si fa riferimento, un giocatore è contraddistinto da due tipi di informazioni: le informazioni generali, quali nome, cognome, età, campionato in cui gioca, club di appartenenza e nazionalità, e le statistiche, ossia un valore numerico che identifica una particolare qualità del giocatore. Infatti, un giocatore possiede 6 statistiche che lo rendono più o meno forte rispetto ad un altro, e sono: velocità, tiro, passaggio, dribbling, difesa e fisico (tuffo, presa, rinvio, riflessi, agilità e piazzamento nel caso il giocatore sia un portiere).
L'insieme di queste statistiche crea un overall, cioè un valore finale che verrà assegnato al giocatore, calcolato secondo dei criteri che spiegheremo in seguito.

CREAZIONE DELLA FORMAZIONE

Una volta creati i vari giocatori, essi verranno aggiunti ad una formazione. La formazione che si vuole creare rispecchia una formazione di calcio presente nel gioco, avente quindi un nome di squadra, un allenatore e un modulo secondo cui disporsi in campo. Inoltre, l'insieme degli overall di ciascun giocatore forma un overall di squadra.

Oggetti di FormazioneKalk



FormazioneKalk presenta una gerarchia come in figura.

-Giocatore: classe base astratta che rappresenta un giocatore e contiene le sue informazioni generali e l'overall. Essa è ereditata dalle seguenti classi concrete che implementano i metodi virtuali puri presenti.

> -Portiere: rappresenta il giocatore all'interno della formazione che ha come ruolo "Portiere". All'interno della formazione vi è un solo portiere.
>
> -Difensore: rappresenta l'insieme dei giocatori all'interno della formazione che hanno come ruolo "Difensore".
>
> -Centrocampista: rappresenta l'insieme dei giocatori all'interno della formazione che hanno come ruolo "Centrocampista".
>
> -Attaccante: rappresenta l'insieme dei giocatori all'interno della formazione che hanno come ruolo "Attaccante".

METODI VIRTUALI PURI

virtual double calcolaOverallGiocatore() const = 0 : restituisce l'overall di un giocatore. Questo metodo è overriddato in ognuna delle classi figlie poiché l'overall dipende in primis dal tipo di giocatore. Per esempio, nel caso il giocatore sia un portiere, l'overall viene calcolato in base alle statistiche del portiere, che differiscono dal resto dei giocatori. In generale, l'overall è calcolato assegnando una percentuale ad ogni statistica e sommando i vari risultati.

Portiere	Difensore	Centrocampista	Attaccante
Tuffo: 20%	Velocità: 7%	Velocità: 7%	Velocità: 30%
Presa: 30%	Tiro: 5%	Tiro: 13%	Tiro: 30%
Rinvio: 5%	Passaggio: 5%	Passaggio:45%	Passaggio: 8%
Riflessi: 30%	Dribbling: 3%	Dribbling: 25%	Dribbling: 25%
Agilità: 5%	Difesa: 40%	Difesa: 5%	Difesa: 2%
Piazzamento: 10%	Fisico: 40%	Fisico: 5%	Fisico: 5%

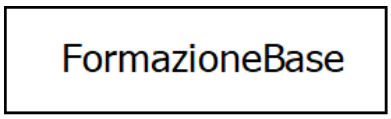
Un difensore che possiede statistiche di difesa e fisico alte avrà un overall in linea con esse, essendo le caratteristiche che si addicono ad un difensore. Un attaccante, nonostante una difesa bassa, avrà un overall in linea con le statistiche di velocità, tiro e dribbling, essendo caratteristiche proprie di un attaccante. Analogamente succede per i centrocampisti.

virtual void SetStats(int,int,int,int,int,int) const = 0: questo metodo riceve come parametri i valori numerici da assegnare alle statistiche di un giocatore. In base al tipo di giocatore (il portiere ha statistiche differenti dal resto), la funzione assegna le statistiche. Infine, con il metodo precedente, calcola l'overall e lo assegna al campo dati overall di Giocatore.

std::string getAllInfo() const = 0: viene restituita una stringa ottenuta concatenando tutte le informazioni generali, le statistiche di un giocatore e l'overall. Anche in questo caso, la scelta di fare overriding è dovuta al fatto che è necessario sapere da che tipo di giocatore si stanno ottenendo le informazioni.

ALTRI METODI

La classe Giocatore, avendo diversi campi privati, presenti i classici metodi "get" per accedervi. Inoltre ogni classe presenta costruttore di default ridefinito e costruttore a più parametri per costruire un giocatore con le statistiche passate.



FormazioneKalk contiene, inoltre, una classe concreta FormazioneBase, utilizzata per la creazione di una formazione. Essa contiene la lista di tutti i giocatori e le informazioni relative alla formazione.

METODI

void aggiungiModulo(std::vector<int>) : aggiunge un modulo alla formazione.

void aggiungiAllentore(std::string) : aggiunge un allenatore alla squadra.

void aggiungiNomeSquadra(std::string) : aggiunge un nome alla squadra.

void aggiungiTitolare(Giocatore*) : aggiunge un giocatore alla formazione.

void eliminaSquadra() : elimina tutte le informazioni e i giocatori appartenenti ad una squadra.

void eliminaModulo() : elimina il modulo (essendo un vector di interi).

int calcolaOverallTitolari() const : calcola l'overall dell'intera formazione.

bool SquadraCompleta() const : verifica se la formazione contiene o meno 11 giocatori.

bool CheckPortiere() const : verifica se la formazione contiene un portiere.

bool CheckDifesa() const : verifica quanti difensori sono presenti nella formazione rispetto al modulo selezionato.

bool CheckCentrocampo() const : verifica quanti centrocampisti sono presenti nella formazione rispetto al modulo selezionato.

bool CheckAttacco() const : verifica quanti attaccanti sono presenti nella formazione rispetto al modulo selezionato.

int sommaGiocatoriModulo() const : ritorna quanti giocatori la formazione dovrebbe contenere secondo il modulo passato. Se la somma degli interi contenuti nel vector supera il valore 10 (il portiere non viene contato), la funzione ritorna false.

bool CheckModulo() const: verifica che il modulo selezionato sia corretto, cioè che rispetti un modulo realmente esistente (10-0-0 oppure 1-1-1-1-1-5 non sono validi) e che la somma dei giocatori previsti dal modulo sia corretta (utilizzando il metodo precedente).

Vari metodi di "get" per accedere ai campi privati.

Costruttore di default ridefinito e costruttore a tre parametri per costruire la formazione.

Polimorfismo

In FormazioneKalk l'uso del polimorfismo è essenziale per la creazione dei giocatori che andranno ad aggiungersi alla formazione.
L'utilizzo di una classe base astratta Giocatore permette una creazione preliminare di quello che poi sarà il giocatore effettivo. In seguito, il polimorfismo permette di specializzare ogni giocatore guardando il suo tipo dinamico e richiamando, conseguentemente, i metodi presenti nelle classi figlie.
Inoltre, il distruttore di Giocatore è reso virtuale in modo da richiamare il distruttore della classe concreta in cui ci si trova.

GUI

La parte grafica è stata realizzata attraverso la classe FormazioneGUI, derivata direttamente dalla classe QWidget. Il QWidget FormazioneGUI viene utilizzato all'interno di un QTabWidget, che permette la costruzione di un tab, chiamato Formazione, attraverso il quale si può interagire con l'applicazione. FormazioneGUI svolge il ruolo di controller.
Il All'interno del tab sono contenuti diversi QPushButton che richiamo diversi slot, i quali costruiscono, all'interno del tab stesso, i vari oggetti che mostrano le informazioni relative alla formazione. Inoltre, è possibile creare i giocatori da

aggiungere alla formazione, utilizzando l'apposita sezione.
La MainWindow contiene un secondo tab, helptab, derivato anch'esso da QWidget, che permette la costruzione di un secondo QTabWidget chiamato Help. All'interno di esso vi sono le istruzioni, passo per passo, per la costruzione della formazione e come utilizzare i QPushButton per le operazioni.
Infine, sono state implementate due classi, classe campo e classe info, derivanti entrambe da QGroupBox, utilizzate per una creazione più facile di oggetti all'interno del tab Formazione.

Sviluppo

Progettazione e sviluppo gerarchia:	8 h
Progettazione e sviluppo GUI:	30 h
Studio libreria Qt:	6 h
Debug:	3 h
Relazione:	4 h
Java:	5 h
Totale:	56 h

Informazioni strumenti di sviluppo

Sistema operativo: Ubuntu 16.10

Compilatore: gcc 4.9.2

Qt: Qt Creator 4.4.1 based on Qt 5.9.2

Informazioni sulla compilazione

Per la compilazione del progetto occorrono solamente i comandi qmake e make, dato che il file .pro è già incluso nel progetto.
In quanto alla parte di Java, viene fornita una cartella JavaKalk. Spostarsi nella cartella interna "src" e utilizzare i comandi javac kalk.java e java kalk.