Andrea Zorzi 561818 Login:@Zux21

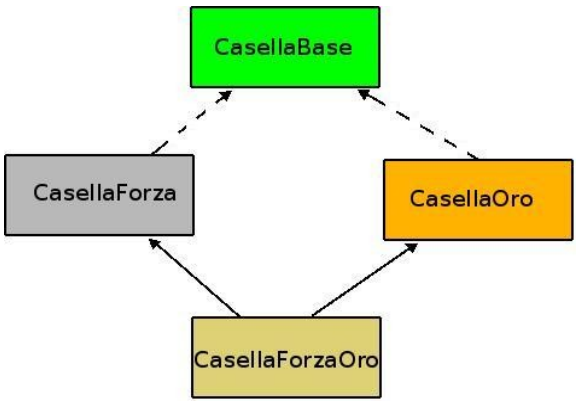
Progetto di Programmazione ad Oggetti

A.A. 2008/2009

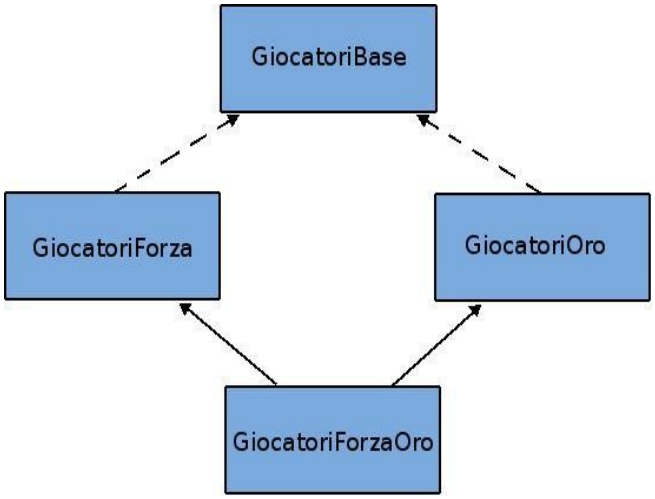
GIOCO DELL'OCA (con morale)

La struttura logica del programma Gioco dell'Oca si compone delle seguenti classi:

- ImpostazioniGioco
- Le classi relative alla casella secondo la seguente struttura gerarchica:



- Le classi relative ai giocatori secondo la seguente struttura gerarchica:



- GiocoOca

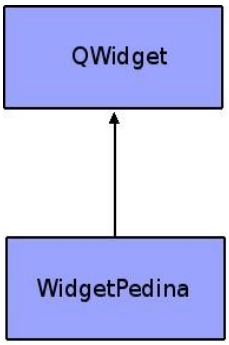
Inoltre, relativamente alla parte logica, è presente una funzione di prototipo **int getRand(int)**

Progetto di Programmazione ad Oggetti
A.A. 2008/2009
GIOCO DELL'OCA (con morale)

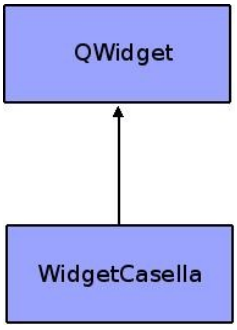
definita nel file functrand.h e implementata in functrand.cpp. Questa funzione e' utilizzata nelle implementazioni delle classi CasellaForza, CasellaOro, GiocoOca, WidgetPedina e GuiGioco per ottenere un valore random compreso tra 1 e l'intero passato a getRand.

La struttura grafica del programma Gioco dell'Oca si compone delle seguenti classi:

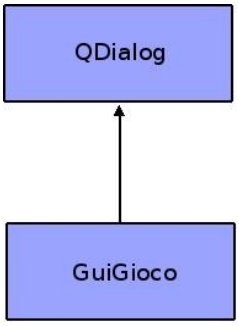
- WidgetPedina



- WidgetCasella



- GuiGioco



Andrea Zorzi 561818 Login:@Zux21

Progetto di Programmazione ad Oggetti
A.A. 2008/2009
GIOCO DELL'OCA (con morale)

1. ImpostazioniGioco

Parte privata:

- **int numerocaselle**: definisce il limite massimo di caselle del gioco
- **string livelloscelto**: indica il livello scelto
- **static char* livellistabiliti[]**: vettore di stringhe che rappresenti i soli quattro possibili livelli
- **bool isLivelloAccettabile(string)const**: funzione di utilità che controlla se la stringa passato per parametro è il nome di uno dei quattro livelli predefiniti.

Parte pubblica:

- **ImpostazioniGioco(int=30,string="")**:costruttore a due parametri, di default, un intero passato per valore che rappresenta il numero totale di caselle e una stringa passata per valore relativa al tipo del livello scelto. Pertanto nel caso l'utente non decida un set personalizzato delle opzioni di gioco è previsto un set di impostazioni di gioco di default.
- **void setNumeroCaselle(int):** imposta il numero di caselle del gioco.
- **void setLivelloScelto(string):** imposta il livello del gioco.
- **int getNumeroCaselle() const:** restituisce il numero di caselle
- **string getLivelloScelto() const:** restituisce il livello stabilito
- **string getLivelloStabilito(int)const:** restituisce il livello corrispondente all'intero passato per parametro (Base=0,Forza=1,Oro=2,Forza-Oro=3).

2. Gerarchia relativa ai giocatori

Per i giocatori si prevede una ereditarietà a diamante. La classe GiocatoriBase è classe base virtuale per le classi GiocatoriForza e GiocatoriOro. La classe GiocatoriForzaOro, derivata da GiocatoriBase e tramite derivazione multipla da GiocatoriForza e GiocatoriOro.
GiocatoriBase è infatti il tipo base dei giocatori. GiocatoriForza contiene le stesse informazioni

Progetto di Programmazione ad Oggetti
A.A. 2008/2009
GIOCO DELL'OCA (con morale)

di GiocatoriBase ma è necessario rappresentare distintamente i giocatori che stanno giocando un livello forza. GiocatoriOro invece contengono anche informazioni aggiuntive quali le quantità d'oro accumulate fino a quel momento dai giocatori e il limite di oro che un giocatore può possedere.

- ## GiocatoriBase

  parte privata:
  - **int giocatori**: il numero di giocatori previsto per il gioco.
  - **bool*turno**: array booleano per gestire il turno di ogni giocatore
  - **bool*traguardo**: array booleano per gestire quali sono i giocatori giunti all'arrivo.

  parte pubblica:
  - **GiocatoriBase(int=3)**:costruttore ad un parametro di default che riceve un intero che rappresenta il numero di giocatori scelti per il gioco. Alloca nello heap l'array turno di dimensione giocatori, settando a true il primo elemento e a false tutti gli altri elementi dell'array.(è il primo giocatore che inizia a giocare). Alloca nello heap l'array traguardo di dimensione giocatori, inizializzando tutti gli elementi a false.
  - **virtual ~GiocatoriBase()**: distruttore virtuale per rendere la classe polimorfa.
  - **void setGiocatori(int)**: imposta il numero di giocatori.
  - **int getNumeroGiocatori()const**: ritorna il numero di giocatori.
  - **int getGiocatoreDiTurno()const**: ritorna il giocatore di turno, ovvero il giocatore di posto i tale che turno[i]==true && turno[i+1]==false.
  - **void impostaGiocatoreDiTurno()**: imposta il turno al giocatore successivo, tenendo conto anche il prossimo giocatore non sia già arrivato al traguardo.
  - **bool isGiocatoreAlTraguardo(int)const**: determina se il giocatore che ha come indice l'intero passato alla funzione, è arrivato o meno all'arrivo.
  - **void setGiocatoreAlTraguardo(int)**: imposta il giocatore che ha come indice l'intero passato alla funzione, al traguardo.

- ## GiocatoriForza

  parte pubblica:
  - **GiocatoriForza(int)**: costruisce il sottooggetto GiocatoriBase. Non è previsto un set

Progetto di Programmazione ad Oggetti
A.A. 2008/2009
GIOCO DELL'OCA (con morale)

di default poiché sarà spiegato il motivo nell'implementazione della parte grafica.

- ## GiocatoriOro

parte privata:

- **int*punteggio**: array di interi che contiene il punteggio (quantità di dobloni d'oro) di ogni giocatore.(ex. punteggio[0] ->è il punteggio del giocatore 0)
- **static int peso_bloccante**: variabile statica che indica il limite massimo di oro che possono portare i giocatori.Fissato a 200.
- **static int rapporto_oro**: variabile statica che indica ogni quanto oro rallentare di una posizione il giocatore interessato. Fissato a 50.

parte pubblica:

- **GiocatoriOro(int)**:costruisce il sottooggetto GiocatoriBase. Anche qui non è previsto un set di default.
- int getPunteggioGiocatore(int) const: ritorna il punteggio del giocatore di indice l'intero passato come parametro alla funzione.
- int*punteggiPerGiocatore(int*)const: dato un array ritorna l'array con il punteggio di ogni giocatore.
- void setPunteggio(int): al giocatore di turno, aggiunge l'intero che rappresenta l'oro della casella all'oro accumulato fino a quel momento presente nel punteggio del giocatore di turno.
- int getPesoBloccante()const: ritorna il peso bloccante.
- int getRapportoOro()const: ritorna il rapporto oro.
- bool blocca(): se il punteggio oro del giocatore di turno è maggiore o uguale al peso bloccante, si azzera il suo punteggio e si ritorna true al chiamante.
- void svuotaPunteggioGiocatore(int): svuota il punteggio del giocatore di indice intero passato alla funzione, impostando il punteggio a -1. (funzione per la gestione sicura della classifica da parte della GUI. In seguito sarà evidenziata meglio la motivazione).

- ## GiocatoriForzaOro

parte pubblica:

- GiocatoriForzaOro(int): costruzione unico sottooggetto della classe GiocatoriBase,

Progetto di Programmazione ad Oggetti
A.A. 2008/2009
GIOCO DELL'OCA (con morale)

e sottooggetti GiocatoriForza e GiocatoriOro.

3. Gerarchia relativa alla casella

Per la casella si prevede una ereditarietà a diamante. La classe CasellaBase è classe base virtuale per le classi CasellaForza e CasellaOro. La classe CasellaForzaOro, derivata da CasellaBase e tramite derivazione multipla da CasellaForza e CasellaOro.
CasellaBase è infatti il tipo base di casella, mentre CasellaForza e CasellaOro sono una specializzazione poiché contengono informazioni aggiuntive quali rispettivamente i propri valori di forza e di oro.CasellaForzaOro contiene entrambe le informazioni di CasellaForza e CasellaOro ed è quindi spiegata l'ereditarietà multipla a diamante.

- **CasellaBase**

  parte privata:
  - **const int numero**: numero della casella

  parte protetta:
  - **ImpostazioniGioco impo**: oggetto ImpostazioniGioco che racchiude le opzioni di gioco definite, più precisamente abbiamo bisogna di sapere il numero massimo di caselle.
  - **GiocatoriBase*gio**: puntatore a GiocatoriBase. Il significato dell'inserimento di questo campo è che in una CasellaBase può capitare qualsiasi giocatore e quindi sarà necessario fare delle differenze(vedi in seguito funzione mossa).E' un puntatore a GiocatoriBase per supportare il poliformismo.

  parte pubblica:
  - **CasellaBase(int,const ImpostazioniGioco &,GiocatoriBase*)**:costruttore a tre parametri che riceve un intero passato per valore che rappresenta il numero della casella,un oggetto ImpostazioniGioco passato per riferimento costante e un puntatore a GiocatoriBase.
  - **int getNumeroCasella() const**: che restituisce il numero della casella.
  - **virtual int mossa(int)**: che riceve un intero x che nel caso base è il valore uscito con il dado, e ritorna un intero y che rappresenta il numero di casella a cui si dovrà portare il giocatore.

Progetto di Programmazione ad Oggetti
A.A. 2008/2009
GIOCO DELL'OCA (con morale)

Il metodo per calcolare y è diverso a seconda del tipo di giocatore che sta muovendo. Se il tipo dinamico di gio è GiocatoriOro(cioè siamo in presenza di un giocatore oro o forzaoro) anche se siamo in una casella base bisogna considerare l'oro posseduto dal giocatore e determinare così un rapporto di rallentamento determinato come punteggio oro del giocatore diviso rapporto oro. Se è diverso da 0 c'è rallentamento; bisogna quindi considerare due casi: se il rallentamento>=x allora lo spostamento da fare si riduce a 0 ed il giocatore rimane fermo, altrimenti se il rallentamento<x, si sottrae al valore x il valore del rallentamento determinando così il nuovo spostamento.
Se il tipo dinamico è diverso da GiocatoriOro(giocatore base o forza) allora lo spostamento è uguale a x.
Successivamente si ottiene y che è calcolato a partire dal campo numero della CasellaBase d'invocazione e possibilmente incrementando dello spostamento, poiché si prende in considerazione il limite di caselle presente nel campo dati impo. Se non è possibile, il giocatore raggiunta l'ultima casella ritorna indietro di quanto stabilito.

- **virtual ~CasellaBase()**: distruttore virtuale poiché la classe è polimorfa.

- ## CasellaForza

parte privata:

- **int valoreForza**: indica la quantità di forza presente nella casella.
  - **static int rapporto_forza**: rapporto forza per determinare l'eventuale effetto accelerativo sul giocatore. è definita di classe uguale a 40 e significa che con un valoreForza di 40 si ottiene un effetto accelerativo di più uno rispetto al numero indicato dal dado.

parte pubblica:

- **CasellaForza(int,const ImpostazioniGioco &,GiocatoriBase*)**: costruttore a tre parametri che costruisce il sottooggetto CasellaBase e assegna a valoreForza un intero random compreso tra 1 e 150.
- **int getForzaCasella() const**: restituisce la forza presente nella casella.
- **int getRapportoForza() const**: restituisce il rapporto forza.
- **void aquistaForza()** : svuota la casella forza azzerando valoreForza, nel momento

Progetto di Programmazione ad Oggetti
A.A. 2008/2009
GIOCO DELL'OCA (con morale)

in cui un giocatore capita nella casella assumendo la forza presente per poi essere usata.

- **virtual int mossa(int)**: overriding del metodo virtuale int mossa(int) di CasellaBase. La casella in cui si è capitati con il dado è una CasellaForza, la forza viene subito usata dal giocatore per avere un effetto accelerativo che possibilmente dà forza al dado, ma non è accumulata dal giocatore dopo che è stata usata. In dettaglio, alla funzione è passato un intero x che rappresenta il numero uscito con il dado. La funzione calcola un rapporto accelerativo(valoreForza/rapporto_forza).Se c'è forza nella casella l'acquista con la funzione acquistaForza(), e se il valore del rapporto accelerativo è diverso da 0 (c'è incremento), questo è passato alla funzione base Casellabase::muovi(int) che opera nel modo descritto in precedenza(in questo caso a CasellaBase::muovi(int) è passato come intero il valore del rapporto accelerativo e non x). Se il rapporto accelerativo è 0 allora non c'è incremento e quindi non cè un effetto accelerativo come prima. Se non è presente forza(valoreForza==0) significa che è già stata usata in precedenza e quindi siamo nel caso in cui un giocatore è nella casella di tipo forza e quindi si passa semplicemente il parametro x alla funzione CasellaBase::muovi(int) poiché ci si muove come nel caso Casellabase. La funzione ritorna un intero che è il numero di casella a cui si dovrà portare il giocatore.

- ## CasellaOro

parte privata:

- **int dobloniOro** : indica la quantità di oro presente nella casella.

parte pubblica:

- **CasellaOro(int,const ImpostazioniGioco &,GiocatoriBase*):** costruttore a tre parametri che costruisce il sottooggetto CasellaBase e assegna a dobloniOro un intero random compreso tra 1 e 40.
- **int getOroCasella() const** : restituisce l'oro presente nella casella.
- **void acquistaOro()** : svuota la casella oro azzerando dobloniOro, nel momento in cui un giocatore accumula l'oro presente nella casella.
- **virtual int mossa(int)** : overriding del metodo virtuale int mossa(int) di CasellaBase.

Progetto di Programmazione ad Oggetti
A.A. 2008/2009
GIOCO DELL'OCA (con morale)

In dettaglio, alla funzione viene passato come parametro un intero x che rappresenta il numero uscito con il dado. Se nella casellaOro in cui il giocatore è capitato son presenti dobloni d'oro allora i dobloni d'oro vengono accumulati dal giocatore. Successivamente la casella è svuotata dell'oro e viene ritornata la posizione della casella.Altrimenti se non sono presenti dobloni d'oro allora la casella è di base e quindi invoca la funzione CasellaBase::muovi(int) passandole x ed il valore ritornato dalla funzione viene ritornato al chiamante.

- **CasellaForzaOro**

  parte pubblica:
  - **CasellaForzaOro(int,const ImpostazioniGioco &,GiocatoriBase*)**: costruzione unico sottooggetto della classe CasellaBase, e sottooggetti CasellaForza e CasellaOro.
  - **virtual int mossa(int)** : overriding del metodo virtuale int mossa(int). In dettaglio, la funzione implementa lo stesso codice di CasellaOro per quanto riguarda lo svuotamento dell'oro, ma successivamente invoca la funzione CasellaForza::muovi(int) per determinare se è possibile l' effetto accelerativo istantaneo.

4. GiocoOca

Parte privata:
Per la pista logica ho pensato di usare un lista di oggetti di tipo Nodo gestiti tramite puntatori smart.
Si definisce una classe Smartp(puntatore smart) a Nodo che contiene un unico campo dati di tipo Nodo* ed in cui ridefinisco assegnazione, costruttore di copia e distruttore che aggiornano opportunamente un campo intero riferimenti. Inoltre ridefinisco la dereferenziazione,l'accesso a membro, l'uguaglianza e la disuguaglianza. Inoltre sempre nella parte privata, viene definita appunto la classe Nodo che contiene un puntatore a CasellaBase di nome casella, un campo dati intero riferimenti e lo Smartp al nodo successivo di nome next.Sempre in Nodo segue la definizione del costruttore e il distruttore, poiché

Progetto di Programmazione ad Oggetti
A.A. 2008/2009
GIOCO DELL'OCA (con morale)

quest'ultimo si occupa di effetuare la delete sul puntatore casella.
La parte privata contiene anche i seguenti campi dati propri:

- **Smartp start**: Smart pointer al primo nodo
- **vector<Smartp> posizione_giocatori**: vector di Smartp che contiene la posizione nella pista logica di ogni giocatore tramite un puntatore smart a Nodo.
- **ImpostazioniGioco impo_gioco**: oggetto di tipo ImpostazioniGioco che racchiude le impostazioni per il GiocoOca.
- **GiocatoriBase*giocatorioca**: puntatore a GiocatoreBase per supportare il polimorfismo.
- **Smartp inserisciInCoda(int,Smartp)**: funzione ricorsiva di utilità/implementazione per l'inserimento degli oggetti Nodo nella lista.Per ogni Nodo, il tipo della casella viene stabilito in modo random generando un numero da 1 a 1000 per i livelli Forza, Oro e ForzaOro. Questo intervallo viene diviso in 4 sottointervalli di 250 se il livello scelto è ForzaOro(perchè le caselle possono essere di tutti e 4 i tipi), e in 2 da 500 se il livello è Forza oppure Oro.Per il livello Base non cè alcuna computazione casuale; per gli altri livelli invece si determina a quale intervallo appartiene l'intero generato casualmente.
- **Smartp cercaPosizione(int)const**:funzione di utilità per la ricerca di un oggetto Nodo nella lista alla posizione dell'intero passato per valore.
- **vector<Smartp>::size_type getIteratoreGiocatoreDiTurno()**: ritorna l'indice(iteratore) per poi ottenere lo Smartp relativo al giocatore di turno.
- **bool arrivo(const Smartp &)const**: controlla se lo Smartp relativo al giocatore è arrivato alla fine della pista logica(arrivo).
- **bool partenza(const Smartp &)const**: controlla se lo Smartp relativo al giocatore è posizionato su start(“il parcheggio esterno” alla pista).

Parte pubblica:

- **GiocoOca()**: costruttore di default per un GiocoOca con impostazioni di default.(livello base e quindi giocatoriBase).Il vector posizione_giocatori è inizializzato con ogni Smartp che punta a start, perchè i giocatori sono posizionati nel parcheggio esterno alla pista.
- **GiocoOca(const ImpostazioniGioco &,GiocatoriBase*)**:costruttore a due parametri per il GiocoOca con le impostazioni scelte dall'utente.Stessa inizializzazione vista in precedenza per il vector posizione_giocatori.

Progetto di Programmazione ad Oggetti
A.A. 2008/2009
GIOCO DELL'OCA (con morale)

- **~GiocoOca()**: distruttore per effettuare la delete sul puntatore giocatorioca.
- **CasellaBase* getCasella(int)const**: riceve un intero che rappresenta il numero della casella e ritorna il puntatore alla casella(campo dati proprio di Nodo) corrispondente.
- **void costruisciPista()**: crea una lista di oggetti Nodo di lunghezza uguale al numero di caselle previste da impo_gioco.
- **int giocataDado(int,bool&)**: riceve un intero x che indica il numero uscito con il dado e un booleano stop passato per riferimento che all'inizio è false. Questo booleano serve per la gestione dello spostamento del giocatore nel caso capiti in una casella non di base. Il booleano viene settato a true quando la casella in cui è capitato il giocatore di turno è di base, o quando è trovata svuotata nel caso sia forza o oro o forzaoro,o quando si blocca un giocatore. Questa gestione è resa necessaria anche per consentire alla parte grafica una miglior gestione di tutte le informazioni durante la partita.Ritorna al chiamante(GUI) la nuova posizione del giocatore che lanciato il dado.
  Funzionamento giocataDado:
  1)seleziono nell'array vettore_giocatori l'iteratore relativo al giocatore di turno(i).
  2)controllo se è nel “parcheggio esterno”. Se si, bisogna muovere secondo mossa base(invocazione della funzione mossa di CasellaBase grazie al binding dinamico).L'intero ritornato da mossa è la nuova posizione dove spostare il giocatore.
  3)altrimenti dalla casella in cui si trova il giocatore invoco la funzione mossa del tipo dinamico della casella((posizione_giocatori[i]->casella)->mossa(x)).(anche qui grazie al binding dinamico). L'intero ritornato da mossa è la nuova posizione dove spostare il giocatore
  4)dynamic cast di giocatorioca al tipo GiocatoriOro* per determinare se bisogna bloccare il giocatore.Se si, si porta il giocatore alla partenza(start) e si setta stop a true.
  5)Nel caso fallisca il cast(siamo nei livelli Oro o ForzaOro) o non venga bloccato, si imposta lo Smartp del giocatore di turno(posizione_giocatori[i]) alla nuova casella determinata in precedenza nei punti 2) o 3).
  6)controllo se il giocatore di turno ha finito il suo turno. Ovvero come spiegato in precedenza controllo la nuova casella in cui è posizionato ora il giocatore. Se la casella è di base, o è di diverso tipo ma è stata svuotata metto stop a true.Altrimenti stop rimane a false e il chiamante(GUI) dovrà in seguito richiamare giocataDado.
- **bool finePartita()**:stabilisce se la partita è finita(assegnando le medaglie nel caso

livello Oro e ForzaOro), altrimenti imposta il prossimo giocatore di turno.

- **static void ordina(int*,int)**:funzione di classe per ordinare un array di interi.

5. Interfaccia grafica (GUI)

L'interfaccia grafica del Gioco dell'Oca è costruita realizzando un oggetto di tipo GuiGioco. La classe GuiGioco è una classe derivata da QDialog, classe base per finestre di dialogo.
La finestra di dialogo è ridimensionabile e possiede una dimensione minima di 800x600.
Quando si avvia il gioco, inizialmente nella finestra di dialogo appare un menù iniziale composto da tre QpushButton(Gioca, Opzioni di gioco, Esci dal gioco) definiti con un layout centrale QVBoxLayout e inseriti in un QGroupBox a sua volta inserito in un main layout QGridLayout.
Premendo subito il pulsante Gioca si chiama lo slot iniziaPartita() il menù sparisce e si gioca una partita di default, ovvero livello base con 30 casella. Più in dettaglio quando viene costruito l'oggetto GuiGioco il costruttore alloca sullo heap un oggetto ImpostazioniGioco che chiama il suo costruttore di default, e un oggetto GiocatoriBase. Ad ogni giocatori base assegna un nome di default.Se si preme il pulsante opzioni di gioco il menù sparisce e si può scegliere il livello attraverso degli QOptionButton e, il numero di caselle della pista e il numero dei giocatori umani attraverso uno QSlider o uno QSpinBox poiché questi sono collegati.Premendo il pulsante indietro si torna al menù e non vengono salvate le impostazioni, mentre premendo ok le opzioni di gioco sono salvate e appare un'altra schermata per l'impostazione personalizzata dei nomi di ogni giocatore. Questa schermata è costituita da un groupbox centrato con all'interno dei QlineEdit con dei Qlabel come buddies, tanti quanto il numero di giocatori scelti in precedenza. Le etichetta(QLabel) di un giocatore ha sempre il corrispondente nome del giocatore di default mentre la QlineEdit associata indica il nome che è stato scelto per ora. Premendo il pulsante indietro si torna alle opzioni di gioco e nel caso sono stati scelti dei nomi per dei giocatori, non sono salvati. Premendo il pulsante ok, invece, si salvano i nomi e si torna al menù principale.
Da questo momento se il giocatore clicca su Gioca giocherà la partita con il suo set di opzioni personalizzate, poiché i campi puntatore GiocatoriBase e ImpostazioniGioco allocati sullo heap sono stati modificati dall'utente nelle opzioni di gioco.

Andrea Zorzi 561818 Login:@Zux21

Progetto di Programmazione ad Oggetti
A.A. 2008/2009
GIOCO DELL'OCA (con morale)

- ## Schermata di gioco

Il layout della schermata di gioco è fisso, ovvero la parte del percorso è fissa ma grazie ad una QScrollArea permette di visualizzare bene tutto il percorso attraverso lo scrolling è poiché la finestra è ridimensionabile, dopo una certa risoluzione è possibile visualizzare tutto il percorso senza fare scrolling.L'altra parte fissa è la parte degli strumenti, che racchiude un dado(QLCDNumber), un bottone lancia per lanciare il dado, il gruppo di giocatori che giocano presentato con tanti quadrati (QLabel) quanti sono i giocatori. Ogni quadrato ha colorato di sfondo con il colore corrispondente al nome del giocatore. Il quadrato è utile per capire facilmente chi è il giocatore con la pedina del colore dello sfondo del quadrato, poiché la pedina è piccola per contenere in dimensioni maggiori il nome del giocatore.Sotto questo gruppo c'è un QLabel infoPartita che mostra l'andamento della partita, più precisamente l'andamento del turno che si sta giocando. Non è stata implementata una cronologia. L'ultimo elemento che appartiene layout degli strumenti è il pulsante esci, utilizzabile in ogni momento, sia durante il gioco per abbandonare la partita, sia a partita conclusa, per tornare al menù principale.
Per la creazione della pista in GuiGioco sono stati definiti esternamente alla classe due oggetti grafici WidgetPedina e WidgetCasella. Questo per favorire l'estendibilità, poiché la casella e la pedina potrebbero essere utilizzate per un altro tipo di gioco.

La classe WidgetCasella eredita QWidget ed ha la seguente interfaccia:
Parte privata:

- **CasellaBase*riferimCasella**:puntatore(polimorfo)alla casella logica.
- **int*info**: puntatore all'array di interi allocato nello heap che contiene le informazioni di riferimCasella.

Parte protetta:

- **void paintEvent(QPaintEvent *event)**:disegna la casella.

Parte pubblica:

- **WidgetCasella(CasellaBase*,QWidget*parent=0)**:costruttore a due parametri
- **void disegnaCasella(QPainter*painter)**: funzione usata nel paintevent per disegnare la casella. Controlli per disegnare la casella: se il puntatore rifermCasella è nullo significa che siamo sullo start e quindi è previsto un colore rosso. Se il tipo dinamico di

Progetto di Programmazione ad Oggetti
A.A. 2008/2009
GIOCO DELL'OCA (con morale)

rifermCasella è CasellaForzaOro e la casella non è stata svuotata è previsto un colore argento-oro. Se il tipo dinamico è CasellaForza e la casella non è stata svuotata è previsto un colore argento. Se il tipo dinamico è CasellaOro e la casella non è stata svuotata è previsto un colore oro. In tutti gli altri casi, ovvero la casella è di tipo base o è stata svuotata, è previsto un colore verde intenso.

- **int*getInfo()**: ritorna le informazioni attuali della casella. La funzione aggiorna le informazioni presenti nel oggetto di invocazione(WidgetCasella) che provengono dagli aggiornamenti eseguiti nella casella logica corrispondente.

La classe WidgetPedina eredita QWidget ed ha la seguente interfaccia:
Parte privata:

- **QString nomePedina**: contiene il nome del giocatore scelto nelle impostazioni di gioco o di default.
- **QLinearGradient gradient**: gradiente lineare a cui si applicano i colori per la pedina.
- **int colore[6]**: array di interi che contiene nelle prime tre posizioni i valori interi per RGB del primo colore, e nelle rimanente tre i valori RGB per il secondo colore. Ho scelto due colori per ogni pedina per poi poter applicare un migliore effetto al gradient.

Parte protetta:

- **void paintEvent(QPaintEvent *event)**: disegna la pedina

Parte pubblica:

- **WidgetCasella*riferimWidgetCasella**: puntatore all'oggetto WidgetCasella. La pedina deve avere questo riferimento perchè la GuiGioco deve necessariamente conoscere la posizione della pedina, ovvero in quale casella si trova.
- **WidgetPedina(QString,WidgetCasella*,QWidget*parent=0)**:costruttore a tre parametri.
- **void disegnaCasella(QPainter*painter)**:funzione usata nel paint event per disegnare la pedina.
- **int getValoreColore(int)const**: ritorna al chiamante(GuiGioco) il valore Red o Green o Blue del colore 1 o 2 a seconda della posizione(intero) passata alla funzione.
- **QString getNomePedina()const**: ritorna il nome della pedina.

Ritornando alla schermata di gioco di GuiGioco, per quanto riguarda le pedine si utilizza un

Progetto di Programmazione ad Oggetti
A.A. 2008/2009
GIOCO DELL'OCA (con morale)

Qvector<WidgetPedina*>, ovvero un vector di puntatori ad oggetti WidgetPedina allocati sullo heap. Dopo essere stati allocati,gli oggetti sono inseriti in un apposito QGridLayout dove una riga può contenere al massimo tre pedine.
Per quanto riguarda le caselle si utilizza un Qvector<WidgetCasella*> ovvero un vector di puntatori ad oggetti WidgetCasella allocati sullo heap.Dopo essere stati allocati, gli oggetti sono inseriti in apposito layout e sono disposti come una serpentina. il layout viene inserito in un QGroupBox, il cui oggetto QGroupbox eredita QWidget cosìchè è possibile applicare ad esso una scrollarea creando un apposito oggetto QScrollArea.

- ## Sviluppo del gioco.

Quando un giocatore lancia il dado cliccando sul pulsante lancia, viene generato in modo random il valore del dado, e l'intero viene passato alla funzione **void Gioca(int)**.Inoltre il pulsante diventa non cliccabile.
Descrizione della funzione principale **void Gioca(int)**.

1) vengono memorizzate le informazioni riguardo alla posizione del giocatore prima della mossa.
2) Si imposta un booleano stop a false. Finchè è il turno del giocatore di turno(stop==false).Entriamo ora nel ciclo while.
3) si invoca la funzione dell'oggetto GiocoOca **int giocataDado(int,bool&)** passandole l'intero(il numero uscito con il dado) parametro della funzione Gioca e il booleano stop. Questa ritorna il numero della nuova casella dove è posizionato ora il giocatore.
4) si chiama update() che ha come oggetto di invocazione l'oggetto GuiGioco.Quindi viene chiamata la paintEvent della GuiGioco che ottiene la posizione della nuova casella, aggiorna la posizione della pedina del giocatore(memorizzata in un Qvector<QPoint*>) impostando l'ascissa e l'ordinata e poi sposta la pedina alla nuova posizione grazie alla funzione **void move (const QPoint &)** che QwidgetPedina eredita da QWidget.
5) Si ottengono le informazioni della nuova casella attraverso il metodo **int*getInfo()** di WidgetCasella.Vengono poi visualizzate con un message box.
6) si invoca l'update() di WidgetCasella per la casella dove finisce il giocatore perchè se è stata svuotata la casella(nel caso sia forza o oro o forzaoro) bisogna ricolorarla del

Progetto di Programmazione ad Oggetti
A.A. 2008/2009
GIOCO DELL'OCA (con morale)

colore appropriato.

7) se stop è ancora uguale a false si rimane nel ciclo e pertanto si torna al punto 3), altrimenti si esce dal ciclo

8) si visualizzano in infoPartita(QLabel) le informazioni sulla giocata fatta dal giocatore di turno.

9) si invoca la funzione **bool finePartita()** di GiocoOca che ritorna true se la partita è conclusa, altrimenti false.

10) Se ritorna false, allora infoPartita visualizza anche qualè il giocatore del prossimo turno e rende cliccabile il pulsante lancia. Se ritorna true, se il livello è oro o forzaoro si visualizza una classifica dei giocatori;altrimenti infoPartita visualizza il nome del vincitore.