Specifica del software

Si vuole realizzare un'applicazione desktop che rappresenti una variante di "*campo minato*", classico videogioco a singolo giocatore per PC degli anni 90, in cui lo scopo del gioco è scoprire tutte le zone sicure del campo minato senza far esplodere le mine.

In particolare si vuole sviluppare una versione del gioco in grado di supportare due tipologie di campo minato: un **campo tradizionale** e un **campo hardcore**. Con il termine **campo tradizionale** si intende un classico campo minato in cui le mine sono generate successivamente la prima zona scoperta dal giocatore, mentre con il termine **campo hardcore** si intende un campo minato in cui le mine sono generate precedentemente la scoperta della prima zona.

L'applicazione da sviluppare si basa sulle seguenti caratteristiche e regole di gioco:

- Il campo di gioco consiste in un campo rettangolare (o quadrato) composto da molteplici zone quadrate con cui il giocatore può interagire cliccando su di esse con il tasto sinistro e destro del mouse.
- Il campo minato può avere delle dimensioni (lunghezza e altezza) minime di 8x8 e massime di 30x30, mentre per quanto riguarda le mine in esso contenute il loro numero può variare da un minimo di 10 a un massimo corrispondente all'80% delle zone costituenti il campo minato.
- Le zone del campo minato sono inizialmente tutte coperte, per poi essere progressivamente scoperte dal giocatore man mano che il gioco procede. Una volta che tutte le zone sicure sono state scoperte il giocatore vince la partita.
- Il giocatore può scoprire una zona cliccando su di essa con il tasto sinistro del mouse. Se la zona scoperta è minata si perde la partita, se invece è sicura viene visualizzato un numero (da 1 a 8) che indica la quantità di mine presenti attorno ad essa. Tale numero deve essere utilizzato dal giocatore per individuare le mine all'interno del campo minato e stabilire quali successive zone scoprire. Nell'eventualità che la zona sicura scoperta non abbia mine nelle sue immediate vicinanze il gioco scopre automaticamente le zone ad essa vicine fintanto che non vengono scoperte zone sicure che restituiscono un numero.
- Il giocatore può eventualmente anche contrassegnare (visivamente con una bandiera) una zona in cui crede sia presente una mina cliccando su di essa con il tasto destro del mouse. Fintanto che una zona è contrassegnata non può essere scoperta, di conseguenza premendo nuovamente il tasto destro su una zona già contrassegnata come minata questa viene "pulita" o riportata al suo stato originario.

L'applicazione deve inoltre prevedere quattro differenti modalità di partita, allo scopo di agevolare i neofiti e introdurre i più piccoli al gioco. Queste modalità fanno riferimento alle medesime regole di gioco e sono differenziate tra loro unicamente dal numero di tentativi disponibili, ossia dalla quantità di mine che è possibile far esplodere prima di perdere la partita. Nel dettaglio:

- la modalità **classica** prevede un singolo tentativo;
- la modalità **agevolata** prevede tre tentativi;
- la modalità **semplificata** prevede un numero di tentativi variabile (non inferiore a tre e corrispondente al 10% delle mine contenute nel campo minato);
- la modalità **sicura** non prevede un limite al numero di tentativi, impedendo di fatto di perdere la partita.

L'applicazione infine deve prevedere l'eventuale aggiunta in futuro di ulteriori tipologie di campo minato.

Studio del problema

Dalla specifica è possibile individuare i seguenti punti critici:

- **Processo di costruzione di un campo minato**

  Nonostante un campo minato tradizionale e uno hardcore si differenzino tra loro in come e quando le mine vengano generate, il processo di costruzione di un campo minato (che include al suo interno anche la fase di generazione delle mine) è uguale per tutte le tipologie di campo minato.

  Di conseguenza, si ritiene opportuno implementare il design pattern **Template Method**, definendo lo scheletro dell'algoritmo di costruzione di un campo minato all'interno della classe base (rappresentante un generico campo minato) e lasciando alle sottoclassi (campo minato tradizionale e hardcore) i dettagli relativi alla generazione delle mine.

- **Riconfigurazione di un campo minato e delle sue zone**

  Per evitare ad ogni nuova partita di istanziare un nuovo campo minato (della medesima tipologia utilizzata nella partita precedente) e potenzialmente fino a 900 zone si reputa ideale:

  – fare in modo che il campo minato possa essere riconfigurato nelle sue caratteristiche (lunghezza, altezza e numero di mine) anche a seguito della sua creazione;

  – creare fin da subito un campo minato di dimensioni massime, ma consentendo all'esterno di poter accedere unicamente alle zone che rientrano nelle dimensioni impostate per il campo minato;

  – fare in modo che le zone del campo minato possano essere anch'esse riconfigurate successivamente la loro creazione.

- **Rappresentazione delle modalità di gioco**

  Tenendo in considerazione che le varie modalità di gioco fanno riferimento alle medesime regole, che l'unica differenza tra loro è costituita dal numero di tentativi ad esse associate, e che il numero di tentativi della modalità *semplificata* è determinato dal numero di mine presenti nel campo minato, si è deciso di rappresentare questo concetto sotto forma di tipo enumerato, definito all'interno della classe che rappresenta le regole di una partita a campo minato.

- **Possibilità di aggiungere nuove tipologie di campo minato in futuro**

  Vista la possibile aggiunta di nuovi tipi di campo minato è auspicabile che l'interfaccia grafica non sia né strettamente accoppiata a tipi specifici di campo minato né si occupi direttamente di istanziarne uno.

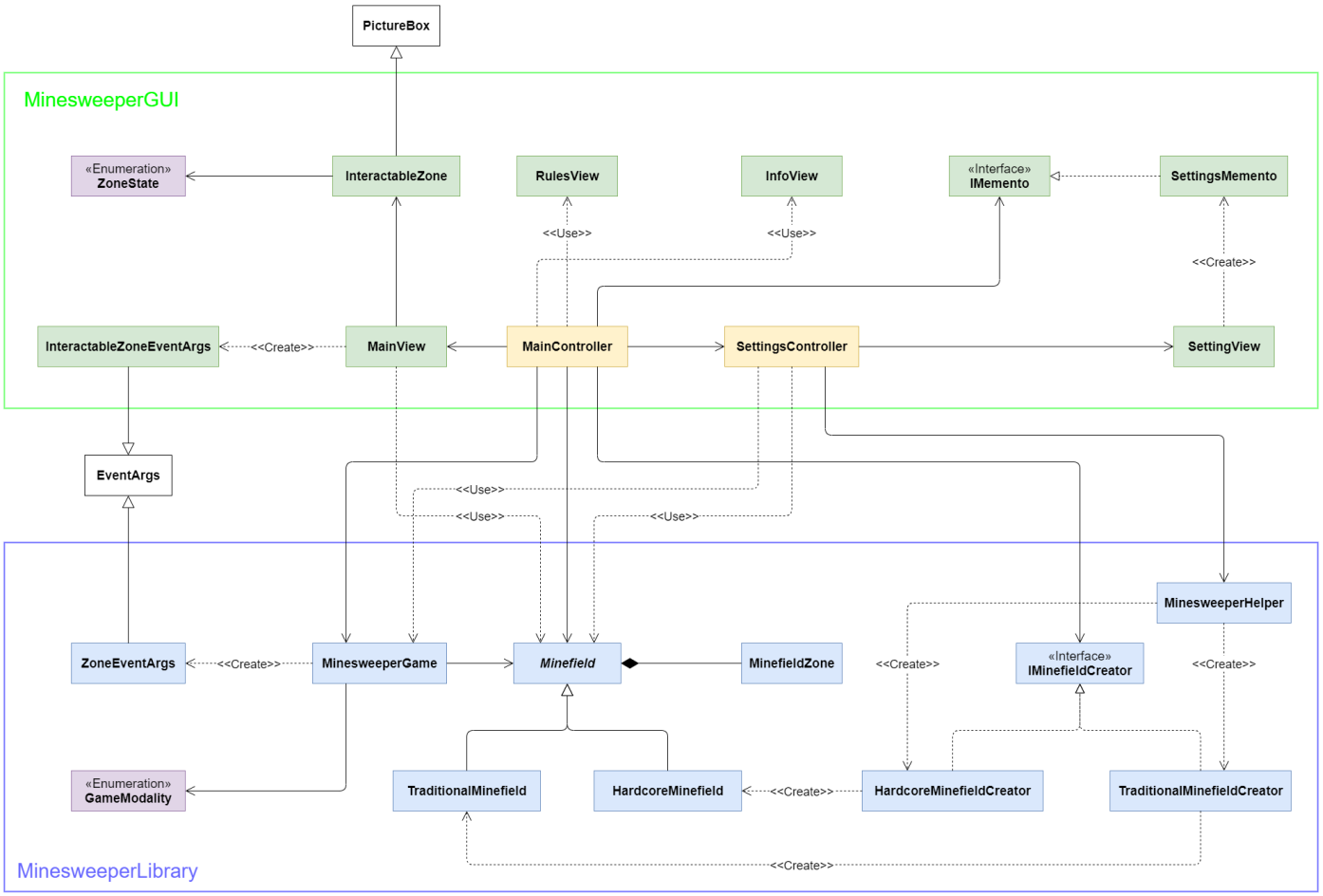
  Di conseguenza, si reputa opportuno utilizzare il design pattern **Factory Method**, in grado di fornire un'interfaccia apposita per la creazione di oggetti, consentendo così di eliminare la dipendenza dell'interfaccia grafica dai tipi concreti di campo minato.

Scelte architetturali

Diagramma delle classi

Considerando le notevoli dimensioni del diagramma delle classi si è preferito riportare di seguito una sua versione semplificata (che omette i membri delle classi) al fine di facilitarne la lettura e la comprensione.

Se si desidera visionare la versione più dettagliata del diagramma delle classi esso è allegato assieme alla relazione all'interno dell'omonima cartella "Relazione".



Design Pattern

Il progetto riassume i seguenti pattern:

- **Pattern "MVC" (Model-View-Controller)**

  MVC è un pattern architetturale in grado di separare la logica di presentazione dei dati dalla logica applicativa. Esso prevede di suddividere il software in tre elementi interconnessi:

  – **Model**: gestisce direttamente i dati e la logica del dominio dell'applicazione, ed è indipendente dall'interfaccia utente;

  – **View**: visualizza i dati contenuti nel model in una qualche rappresentazione per l'utente;

  – **Controller**: riceve gli input dell'utente attraverso il view e gli gestisce convertendoli in comandi per il model e aggiornando il view.

  Il pattern MVC si basa sulla separazione dei compiti fra i vari componenti del software, favorendo così la riusabilità e la manutenibilità del codice. Per via delle sue caratteristiche esso è solitamente utilizzato per sviluppare applicazioni con interfacce grafiche utente e proprio per queste sue qualità si è scelto di utilizzarlo come struttura base del progetto.

  Nello specifico, con riferimento al precedente diagramma delle classi, è possibile notare che:

- Le classi colorate di *blu* rappresentano il *model*, e costituiscono la libreria;
- Le classi colorare di *verde* rappresentano il *view* e alcuni componenti utilizzati dal *view*;
- La classi colorate di *arancione* rappresentano il *controller*.

- **Pattern "Factory Method"**

  Factory Method è un design pattern creazionale che prevede di definire un'interfaccia per la creazione di oggetti, lasciando alle sottoclassi che implementano tale interfaccia la decisione di quale classe istanziare. Esso pertanto consente, in altri termini, di deferire l'istanziazione di una classe alle sottoclassi.

  Come spiegato nella sezione precedente, per rendere l'interfaccia grafica il più possibile disaccoppiata dai tipi concreti di campo minato (che possono potenzialmente aumentare in futuro, come indicato nelle specifiche) si è deciso di adottare questo pattern. Nel dettaglio, si è definita l'interfaccia per la creazione di un campo minato nella classe "*IMinefieldCreator*", mentre le classi "*TraditionalMinefieldCreator*" e "*HardcoreMinefieldCreator*" definiscono quale specifico tipo concreto di campo minato istanziare.

  In aggiunta a queste classi, che costituiscono il pattern citato, per fare in modo che l'interfaccia grafica non necessiti di modifiche al codice a seguito di nuovi tipi di campo minato aggiunti è stata definita un'ulteriore classe, ossia la classe "*MinesweeperHelper*", la quale fornisce un elenco dei possibili tipi di campo minato istanziabili, restituendo anche per ciascuno di essi il corrispondente creatore.

- **Pattern "Template Method"**

  Template Method è un design pattern comportamentale che prevede di definire la struttura di un algoritmo all'interno di un metodo, delegando alcuni passi dell'algoritmo alle sottoclassi. In altri termini, esso consente di ridefinire e personalizzare parte del comportamento definito da un algoritmo.

  Come spiegato nella sezione precedente, il processo di creazione di un campo minato è il medesimo per ogni tipo di campo minato ed esso può essere riassunto a grandi linee nei seguenti passi:

  1. Acquisizione delle caratteristiche del campo minato (lunghezza, altezza e numero di mine)
  2. Controllo delle caratteristiche del campo minato
  3. Generazione delle coordinate delle mine
  4. Generazione delle zone minate e delle zone sicure del campo minato

  Considerando che tra questi passi solo quello relativo alla generazione delle coordinate delle mine è diverso nei dettagli per ogni tipologia di campo minato, l'adozione di questo pattern è stata considerata la scelta più ovvia.

  Nel dettaglio, si è implementato il Template Method definendo l'algoritmo di costruzione di un campo minato all'interno della classe "*Minefield*", e rendendo il sotto-metodo relativo alla generazione delle coordinate delle mine un metodo astratto in maniera tale che le classi derivate "*TraditionalMinefield*" e "*HardcoreMinefield*" debbano necessariamente dargli un'implementazione.

- **Pattern "Memento"**

  Memento è un design pattern comportamentale che consente di salvare e ripristinare lo stato precedente di un oggetto senza rivelare i dettagli della sua implementazione. L'oggetto dedicato a contenere lo stato interno di un altro oggetto è per l'appunto noto con il nome di memento.

  La scelta di adottare questo pattern è stata dettata dalla volontà di voler fare in modo che la sezione dell'interfaccia grafica dedicata alla configurazione delle impostazioni di gioco restituisse all'utente le ultime opzioni da lui selezionate e confermate in precedenza.

  Prima di entrare nei dettagli di come questo pattern è stato implementato è importante sottolineare che un memento deve disporre di una doppia interfaccia:

  - una nei confronti dell'oggetto che l'ha generato, più ampia e che consente a quest'ultimo di salvare e ripristinare il suo stato interno.
  - una verso gli altri oggetti, più limitata e che esporrà eventualmente solo funzionalità accessorie.

Detto questo, si è deciso di rappresentare l'interfaccia "limitata" di un oggetto memento attraverso la classe "*IMemento*", mentre la sua interfaccia "più ampia" attraverso la classe "*SettingsMemento*".
Il pattern Memento verrà quindi implementato creando un oggetto memento come istanza della classe "*SettingsMemento*" la quale implementerà l'interfaccia "*IMemento*".

Documentazione sull'utilizzo

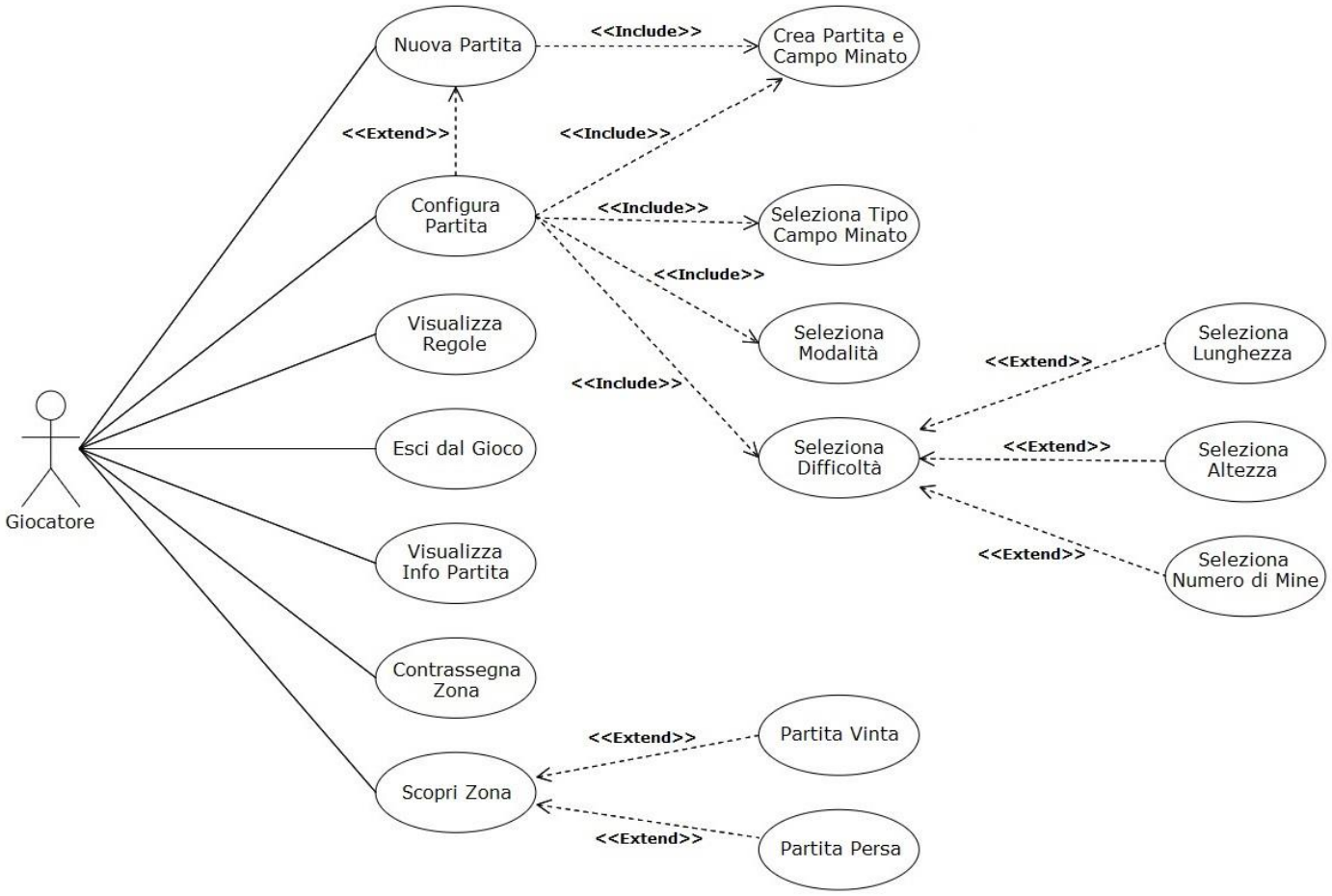
Siccome il progetto è stato strutturato in maniera tale che la logica di un partita a campo minato venisse riassunta all'interno di una libreria, questa può essere riutilizzata in altri contesti.

Nello specifico, dato che il progetto è stato sviluppato attraverso l'IDE Microsoft Visual Studio, di seguito verranno riassunti i passi da seguire – tramite questo IDE – per riutilizzare la libreria in altri progetti.

- Avviare Microsoft Visual Studio e aprire il progetto su cui si vuole importare la libreria.
- Cliccare sull'opzione "*Progetto*" dal menu principale, e poi sull'opzione "*Aggiungi riferimento…*".
- Dalla finestra di gestione dei riferimenti cliccare sul bottone "*Sfoglia…*".
- Specificare il percorso della libreria, per poi cliccare sul pulsante "*Aggiungi*".
- Nella sezione "*Sfoglia*" della finestra di gestione dei riferimenti assicurarsi che la libreria appena importata abbia il controllo adiacente al suo nome spuntato.
- Cliccare sul pulsante "*OK*".
- A questo punto è sufficiente, durante la stesura del codice, richiamare la libreria tramite l'utilizzo della tradizionale direttiva **using** seguita dal namespace della libreria.

Casi d'uso

Il diagramma dei casi d'uso è il seguente:



Di seguito viene riportata la descrizione dei casi d'uso più significativi e utilizzati:

Caso d'uso: Nuova Partita
ID: UC1
Attore/i: Giocatore (A1)
Precondizioni: nessuna
Corso base degli eventi: 1. A1 clicca sul bottone di gioco oppure sull'opzione "Partita" e poi "Nuova" del menu di gioco. 2. Se è stata acquisita una configurazione di gioco in precedenza viene creato un nuovo campo minato e una nuova partita considerando questa configurazione.
Postcondizioni: • Viene visualizzato un campo minato le cui zone sono tutte coperte e non contrassegnate. • Il timer della partita e il contatore delle zone da contrassegnare vengono reimpostati. • Viene disabilitata l'opzione "Info" del menu di gioco.
Percorsi alternativi: 3. Se il punto 2 fallisce viene eseguito UC2.

Caso d'uso: Configura Partita

ID: UC2

Attore/i: Giocatore (A1)

Precondizioni: nessuna

Corso base degli eventi:

1. A1 clicca sull'opzione "Partita" e poi "Configura" del menu di gioco.
2. Viene visualizzata la finestra delle impostazioni di gioco.
3. Se è stata acquisita una configurazione di gioco in precedenza vengono automaticamente selezionate nella finestra delle impostazioni di gioco le medesime opzioni della configurazione precedente.
4. (opzionale) A1 seleziona il tipo di campo minato.
5. (opzionale) A1 seleziona la modalità di gioco.
6. (opzionale) A1 seleziona la difficoltà di gioco.
7. A1 preme il pulsante "Ok", chiudendo la finestra delle impostazioni di gioco.
8. Viene creato un nuovo campo minato e una nuova partita considerando la configurazione stabilita.

Postcondizioni:

- La configurazione di gioco appena stabilita viene memorizzata.
- Viene visualizzato un campo minato le cui zone sono tutte coperte e non contrassegnate.
- Il timer della partita e il contatore delle zone da contrassegnare vengono reimpostati.
- Viene disabilitata l'opzione "Info" del menu di gioco.

Percorsi alternativi:

9. Se il punto 3 fallisce vengono automaticamente selezionate nella finestra delle impostazioni di gioco le opzioni di default.
10. Se nel punto 6 si è selezionata la difficoltà "Personalizzata" allora A1 può cambiare i valori associati a lunghezza, altezza e numero di mine del campo minato.
11. Se A1 preme il pulsante "Cancella" viene chiusa la finestra delle impostazioni di gioco senza memorizzare la configurazione di gioco.

NOTA: I punti 4, 5 e 6 sono stati indicati come opzionali dal punto di vista del giocatore, tuttavia essi sono richiesti all'interno del caso d'uso UC2. Infatti, come è possibile visionare all'interno del diagramma dei casi d'uso, sono presenti delle relazioni di tipo <<*include*>> tra UC2 e i rispettivi casi d'uso dei punti 4, 5 e 6.

Caso d'uso: Visualizza Info Partita

ID: UC3

Attore/i: Giocatore (A1)

Precondizioni:

- Il campo minato è stato visualizzato all'interno della finestra principale di gioco.
- Almeno una zona del campo minato è stata scoperta.

Corso base degli eventi:

1. A1 clicca sull'opzione "Info" del menu di gioco.

Postcondizioni:

- Viene visualizzata una nuova finestra contenente varie informazioni sulla partita in corso (tipologia e caratteristiche del campo minato; modalità di gioco; tentativi rimanenti e totali).

Percorsi alternativi: nessuno

Caso d'uso: Contrassegna Zona

ID: UC4

Attore/i: Giocatore (A1)

Precondizioni:
- Il campo minato è stato visualizzato all'interno della finestra principale di gioco.
- Almeno una zona del campo minato è stata scoperta.

Corso base degli eventi:
1. A1 clicca con il pulsante destro del mouse su una zona coperta.

Postcondizioni:
- La zona viene contrassegnata come minata se prima non lo era, altrimenti avviene il contrario.
- Il valore del contatore delle zone da contrassegnare viene modificato (viene incrementato di un'unità se la zona viene contrassegnata, altrimenti viene decrementato di un'unità).

Percorsi alternativi:
2. Se A1 clicca con il pulsante destro del mouse su una zona scoperta non accade nulla.

Caso d'uso: Scopri Zona

ID: UC5

Attore/i: Giocatore (A1)

Precondizioni:
- Il campo minato è stato visualizzato all'interno della finestra principale di gioco.

Corso base degli eventi:
1. A1 clicca con il pulsante sinistro del mouse su una zona coperta e non contrassegnata.
2. Se non è stata ancora scoperta nessuna zona viene avviato il timer della partita e viene abilitata l'opzione "Info" del menu di gioco.
3. (A) Se la zona è minata viene visualizzata una mina, viene ridotto il numero di tentativi rimanenti e viene ridotto di un'unità il contatore delle zone da contrassegnare.
3. (B) Se la zona è sicura e non ci sono mine attorno ad essa vengono scoperte le sue zone adiacenti.
3. (C) Se la zona è sicura ma ci sono mine attorno ad essa viene visualizzato il numero di mine (da 1 a 8).

Postcondizioni:
- La zona viene scoperta.

Percorsi alternativi:
4. Se A1 clicca con il pulsante sinistro del mouse su una zona scoperta oppure contrassegnata come minata non accade nulla.
5. Se, nel punto 3A, i tentativi rimanenti raggiungono lo zero A1 perde la partita.
6. Se, nel punto 3B oppure 3C, si scopre l'ultima zona sicura del campo minato A1 vince la partita.

NOTA: Per i punti 3A, 3B e 3C si è deciso di utilizzata questa dicitura allo scopo di evidenziare il fatto che solamente una di queste alternative viene necessariamente eseguita ogni volta che il caso d'uso UC5 viene eseguito.