Progetto Programmazione

Interfaccia Struttura Mappe

All'interno del gioco la struttura delle mappe è organizzata come un edificio reale, eccetto per il numero dei piani che è infinito.

Ogni piano *i* ha esattamente *i* stanze, fino a un massimo di 50 stanze per piano.

Il personaggio si muove all'interno delle stanze di un piano dove puo' trovare:

1. Scala per Salire: questa permette al personaggio di salire in una stanza al prossimo piano.
2. Scala per Scendere: questa permette al personaggio di scendere in una stanza al piano precedente.
3. Porte: per muoversi tra le varie stanze.

Attenzione: il piano viene generato nel momento della prima visita di una sua stanza. L'algortimo collega le stanze in modo casuale in modo che sia possibile visitarle tutte ma non fornisce al giocatore la descrizione del collegamento di queste.

Impostazione Strutture Dati

Mappa:

```
class Mappa
{
    public:
      int id; /*! id mappa id*/
      int location[20][20]; /*! posizione */
      int porte[3] = {-1, -1 ,-1}; /*! porte */

    struct {
        int x; /*! \var int x */
        int y; /*! \var int y */
    }typedef coord;
    coord portexy[3];
}
```

Piano:

```
class Piano
{
  Mappa arr_mappe[50]; /*! array delle mappe del piano*/
  int numero_mappe; /*! \var numero mappe all'interno del piano */
};
```

Lista Torre:

```
class ListaTorre
{
    struct nodo
    {
        nodo *precedente; /*! puntatore al piano precedente*/
        Piano piano; // Oggetto Piano contenente l'array delle stanze
        nodo *successivo; /*! puntatore al piano precedente*/
    };
    typedef nodo *Lista;
};
```

Libreria grafica

Questo gioco è stato realizzato con il supporto della libreria grafica SFML (versione. 2.4.2).

SFML è multi-media

SFML è provvisto di una semplice interfaccia per i vari componenti del tuo pc, per semplificare lo sviluppo di giochi e applicazioni multimediali. E' composto da 5 Moduli: system, window, graphics, audio and network.

SFML è multi-platform

Con SFML la tua applicazione puo' compilare e eseguire in modo indipendente relativamente al sistema operativo usato: Windows, Linux, Mac OS X e presto Android e iOS.

Realizzazone Interfaccia Grafica

```
// Start the game loop
while (window.isOpen())
{
  // Process events
  sf::Event event;
  while (window.pollEvent(event))
  {
    // Close window : exit
    if (event.type == sf::Event::Closed)
      window.close();
  }
  // Clear screen
  window.clear();
  // Draw the elments
  window.draw(elments);
  // Update the window
  window.display();
}
```

Finchè il gioco rimane aperto, aggiorno periodicamente la finestra del gioco e in base agli eventi generati verranno apportate modifiche agli elementi disegnati, che saranno subito visibili l'istante dopo.

Sia i nemici che i personaggi ereditano le funzioni da una classe ElementoGrafico

```
class ElementoGrafico
{
  private:

  public:
  // ATTRIBUTI
  Sprite grafica; /*!  sprite dell'elemento */
  Texture texture; /*! texture utile per caricare l'immagine */
  Utilities util;
  int posX; /*! posizone calcolata in pixel */
  int posY; /*! posizione calcolata in pixel */
  int posX_iniziale, posY_iniziale; /*!  posizione iniziale in termini
di pixel */
  int spostamento_x, spostamento_y; /*! spostamento in termini di
celle */
  int pos_cella_x, pos_cella_y; /*! posizione della cella attualmente
occupta*/

  // serve per settare la posizione nella finestra dell'eroe
  // x e y rappresentano lo spostamento in termini di celle rispetto
alla sua posizione
  void setPosizione(int x, int y);
  // questa funzione muove l'elemento grafico nella sua nuova
posizione
  void muovi();

  // costruttore
  // in base al numero passato la funzione caricherà una determinata
immagine.
  // ad esempio il codice per l'immagine dell'eroe è 01
  ElementoGrafico(int tipo_personaggio);
};
```

Questa classe permette di impostare e settare immagini e posizioni per ogni elemento.

Realizzazione Struttura Gameplay

Ogni elemento all'interno del gioco ha una relativa classe per semplificarne la gestione:

- Menu Iniziale
- Lista Nemici
- Personaggio
- Grafica

- Lista Torre

Vi è poi una classe generica per la gestione delle funzioni indipendenti dalla realizzazione del Gioco:

- Utilities

Le altre classi servono solo come implementazioni delle classe principali prima elencate:

- Mappa
- Piano
- Nemico
- Elemento Grafico

Ecco descritto il Flow del Gioco:

1. Persoanggio (puo' scegliere se)
    1. Muoversi
    2. Attaccare
2. Nemici (eseguono le seguenti azioni)
    1. Se trovano l'eroe vicino a loro lo attaccano
    2. Si muovono in una posizione random all'interno della mappa

---

Per ulteriori informazioni sulla struttura del programma e per avere informazioni dettagliate sul codice è consigliato consultare la documentazione allegata.

MC.