Breve Relazione sul videogame MANnA.

E io chi sono?

Nome: Mohammad (detto Samir).
Cognome: Fumagalli Tahboub
Anno: 1°.
Corso: Informatica.
Sede: Bologna, matricola: 000 080 6558.

Partendo dal nome del videogioco (MANnA):

Questo videogame da me fatto per l'Università d'Informatica di Bologna è mirato a far verificare la mia idoneità come programmatore, infatti esso è stato realizzato per completare il carico di lavoro a me richiestomi per passare l'esame di programmazione del 1° anno.
Il videogioco MANnA è sottoposto alla cortese attenzione del professore Ivan Lanese che è il docente responsabile della valutazione.

La Parola MANnA racchiude per me 2 parole, mana e manna, la prima ha significato: *"forza vitale"* ed è anche l'icona del videogioco ed è anche uno dei 5 simboli usati per rappresentare il famoso gioco di carte Magic .
La parola mana fa parte del linguaggio di alcuni stati concentrati nel sud-est asiatico.
La seconda parola manna ha invece origine Ebraiche, ed è diventata una parola famosa col tempo per il suo uso biblico, dove essa è un cibo povero, provvidenziale caduto dal cielo, tuttavia ha anche un uso gastronomico.

"È la linfa estratta dalla corteccia di alcune specie di piante del genere Fraxinus (frassini) – Wikipedia". Notare che sia la mana vitale raffigurata tipicamente come forza elementare della natura e la manna gastronomica estratta dalle piante sono entrambe di colore tendente al verde.
Queste due parole sono state scelte da me per esprimere il concetto: *"questo videogioco pur essendo un prodotto grezzo, basilare di un novizio, è pur sempre in parte manifestazione di un pragmatismo vitale"*.

Informazioni utili dal punto di vista Informatico:

Tutti i sistemi operativi supportati: No.
Sistema/i operativo/i supportato/i: Unix, testato su debian xfce 9(Stretch).
Linguaggio di programmazione: C++.
Compiler C++: Clang x86/x64.
Clang Version: 3.8.1-24.
Uso dell'Interfaccia Grafica: Si.
Tool grafico usato: QT (qt creator) 5.9.1.
librerie QT aggiunte oltre alle librerie base/standard: Si.
nome delle librerie aggiunte: qtmultimedia5.
Download QT: https://info.qt.io/download-qt-for-application-development.
(Get your open source package).

Informazioni su QT:

Qt 5.9.1 (x86_64-little_endian-lp64 shared (dynamic) release build; by GCC 5.3.1 20160406 (Red Hat 5.3.1-6)) on "xcb". OS: Debian GNU/Linux 9 (stretch) [linux version 4.9.0-3-amd64].

Informazioni su QT:

Modifiche necessarie al file MANnA.pro:

QT += core gui
QT += multimedia
QT += widgets

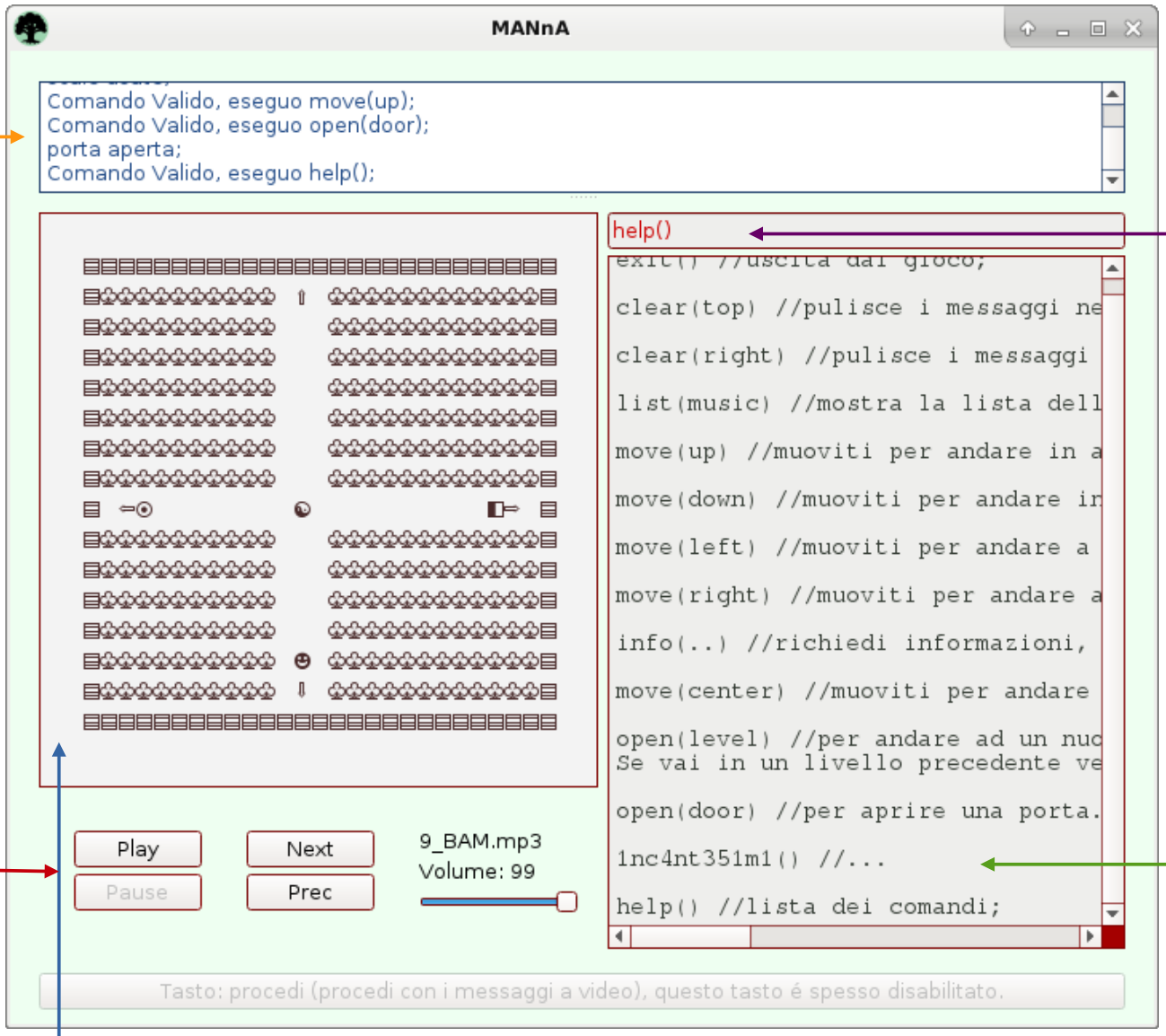
Scelte tecniche di Programmazione:

-Rarissimo uso di espressioni lambda, “le funzioni di ordine superiore”,
unico uso: “[&]()”
-Uso frequente dei collegamenti tra le varie classi tramite connect.
-Funzioni, Oggetti, Puntatori, … delle Classi divisi in:
private: uso privato.
public: uso pubblico.
signals: uso come emittente.
public slots: uso come ricevente .
-Uso sistematico dell’ereditarietà per le classi del gioco per ereditare il QObject.
-Uso frequente del QObject per poter usare il →deleteLater() sugli oggetti.
-Ricreazione delle classi per le liste, circolari e non.
-Minuziosa divisione di classi, funzioni; per uso e scopo.
-Per ogni classe concepita esiste una dichiarazione e una definizione.
la dichiarazione è la parte: Header.
la definizione è la parte: Source.
-Raccolta di Haders e Sources:
gli hader si trovano come: nome_file.h(estensione: h).
i source si trovano come: nome_file.cpp(estensione: cpp).
-Visto che il progetto mi era stato richiesto che fosse orientato principalmente al C++, ho cercato di ridurre al minimo l’uso di librerie QT creator, ma non mi è stata cosa facile, i principali output del QT richiedono una QString o tipi affini appartenenti alle librerie QT, quindi ho spesso dovuto usare una string per poter ottenere i singoli indirizzi del testo ma poi forzatamente ho dovuto riconvertire la string in una QString per poterla mandare in output sull’interfaccia grafica.
-Scelta di includere musica per il sotto fondo, questo aggrava il gioco di un peso che in termini di memoria sul disco rigido, potrebbe non essere gradito.
Il volume digitale si aggira comunque solo sui 93mb, ma è essenziale per poter far caricare l’applicazione. Insieme alla cartella “OZ_Game_sound”, bisogna considerare pure l’icona “Mana_G.png”, se no il gioco non parte. (Da mettersi affianco all’eseguibile).

Scelte tecniche di Gioco:

-Uso dell’interfaccia grafica per i seguenti motivi:
compattezza nella rappresentazione,
immediatezza nella comprensione e nell’utilizzo.
-Uso frequente di messaggi d’aiuto, riguardanti operazioni in corso o/ed operazioni future.
-Scelta di Bloccare l’input in certi momenti del gioco:
caricamenti, combattimenti, fine dei turni, messaggi a video, ...
-Uso di colori rilassanti, tendenti al verde chiaro o all’azzurrino.
scelta auto motivata dal fatto che essendo un assiduo frequentatore di YouTube e siti affini, ho potuto notare come il colore verde in tonalità chiare sia molto usato, e funzioni.
-Scelta di ridurre al minimo i bottoni digitali cliccabili, ho preferito impostare un sistema simile alla linea di comando Unix per interagire col gioco.

Descrivo L'Interfaccia:



output riguardante la riproduzione grafica delle singole stanze facente parte di un livello, Inoltre questa finestra di riproduzione testuale è utile per capire come si comporta il giocatore *mob* controllato dal computer, ed è utile anche per capire come sono collegate le varie stanze fra di loro.

Pulsanti Per la gestione dei multimedia, in questo caso dei file musicali.

Solo Riproduzione Grafica, output riguardante:
Note tecniche sull'esecuzione, ogni nota riportata, Dovrebbe rassicurare il giocatore sul corretto, *f*unzionamento del programma/videogioco.

Sezione di input. Riguardante, Solo comandi, Predefiniti.

I comandi predefiniti, sono stati scelti in fase di programmazione

Solo Riproduzione Grafica, output riguardante:
aiuti, inventario, mappa, status, combattimenti...

Uno dei tanti Algoritmi Interessanti:

L'algoritmo che mentre definisce la creazione di un livello x ne crea anche la resa grafica per x stanze:

```
void mylistQStringMaps::add1map(int index_livello)
{
    if (this->testa == NULL) {...}

   else
    {
        int valore_direzionale = 0;
        int tmp_valore_direzionale = 0;
        int new_tmpy = 0, new_tmpx = 0;

        mylistQStringMaps *tmp = new mylistQStringMaps;
        tmp->main = this->coda->main;
        tmp->index = this->coda->index+1;
        tmp->setSimboloPg1(this->coda->main->getSimboloPg1());
        tmp->addRoom();

        tmp->up = NULL;
        tmp->left = NULL;
        tmp->right = NULL;
        tmp->down = NULL;
        this->numero_di_mappe += 1;

        loop:;

        if ((this->main->up_non_fallito != true)
        && (this->main->right_non_fallito != true)
        && (this->main->down_non_fallito != true)
        && (this->main->left_non_fallito != true)) {...}

        //da 0 a 4
        valore_direzionale = rand() % 5;

        new_tmpy = 0;
        new_tmpx = 0;

        if (valore_direzionale == 0) goto loop;
```

```
        new_tmpy = this->main->y_attuale;
        new_tmpx = this->main->x_attuale;

        if (valore_direzionale == 1) {...}

        else if (valore_direzionale == 2) {...}

        else if (valore_direzionale == 3) {...}

        else if (valore_direzionale == 4) {...}

        this->main->oggetto_MiniMap->create_new_miniMap(new_tmpy, new_tmpx,
        this->main->oggetto_MiniMap->stampQStringRoomNumber(index_livello,
                                                            tmp->index),
                                                           index_livello);
        this->main->y_attuale = new_tmpy;
        this->main->x_attuale = new_tmpx;

        this->setPlayerToNewRoom(this->coda);

        mylistQString *save_pointer =
                this->main->oggetto_MiniMap->getPOINTERFromXY(index_livello,
                                                              this->main->y_attuale,
                                                            this->main->x_attuale-1);
        if (save_pointer != NULL ) {...}

        mylistQString *save_pointer2 =
                this->main->oggetto_MiniMap->getPOINTERFromXY(index_livello,
                                                              this->main->y_attuale,
                                                            this->main->x_attuale+1);
        mylistQString *save_pointer3 =
                this->main->oggetto_MiniMap->getPOINTERFromXY(index_livello,
                                                              this->main->y_attuale,
                                                             this->main->x_attuale);
        if (save_pointer2 != NULL) {...}

        this->main->up_non_fallito = true;
        this->main->right_non_fallito = true;
        this->main->down_non_fallito = true;
        this->main->left_non_fallito = true;
    }
}
```

IN BREVE:

la funzione ha nome add1map, trovabile in mylistQStringMaps.cpp, e richiede come parametro un intero(il numero del livello corrente), questo poiché a ogni singola stanza che graficamente viene riportata con “[numero_stanza]” si aggiungono tot spazi alla fine, finché essa non è di unità di misura uguale al numero dato come parametro della funzione add1map + i simboli “[]”.

Per esempio: mettiamo add1map(78), la rappresentazione massima che si raggiungerà è “[78]”, poiché ogni livello ha tot stanze uguale al suo valore.
Ritornando a noi, se la rappresentazione massima che si raggiungerà è “[78]”, allora se l’algoritmo dovrà creare la rappresentazione grafica della stanza numero 5 dovrà elaborare:
“[5] ” anziché [5] poiché tra 5 e 70 c’è una differenza > di 4.

I “-” o “--” o “---” che ogni tanto si intravedono nella mappa di gioco non sono altro che degli spazi vuoti riempiti, questi spazi vengono riempiti se la stanza 5 è collegata direttamente ad un’altra stanza, collegata dalla sua destra alla sinistra di quella stanza.
Quindi 5 per il livello 70 è “[5] ” oppure “[5]-” se 5 è collegata da destra, per esempio a 6 o a 4.

Nella mappa da me riportata su questo documento si vede che la stanza numero 40 appartenente in questo caso al livello 1000 è rappresentata così: “[40]--” pur non avendo collegamenti visibili verso la sua destra, la risposta è presto detta: la visualizzazione della mappa è conseguente all’esplorazione o conseguente a un certo tipo di comandi nel videogioco, quindi è facile intuire che essendo a livello 1000, alla destra della stanza 40 ci sarà sicuramente un’altra stanza > di 207.

LE FASI:

1°: crea una nuova stanza temporanea, la tmp di tipo: mylistQStringMaps.

2°: definisce la tmp,

3° sceglie un valore casuale per attaccare la tmp all’ultima stanza esistente di questo livello, la scelta ricade sulle seguenti direzioni: in alto, in basso, a destra o a sinistra.

4°: se la posizione è libera e nella resa grafica c’è spazio per raffigurare tale spazio procede con l’attaccamento e poi si occupa di settare la nuova rappresentazione con il create_new_miniMap(int, int, QString, int); altrimenti ritorna cono un goto a rielaborare il suo ragionamento, provando una nuova combinazione casuale.

5°: alla fine di tutta questa tribolazione, quando finalmente ha trovato una stanza che è libera sia nella mappa testuale raffigurata nell’output e sia nei suoi collegamenti, allora procede collegandoci appunto il tmp.

controlla in fine se ci vada aggiunta qualche lineetta ‘-’ alla sua rappresentazione grafica, come appunto abbiamo visto che poteva succedere al 5 nel caso fosse posto in un livello 70.

Curiosità al riguardo:

La mappa che è evidentemente una matrice parte da grandezza x=0, y=0.
Si Espande man mano che le vengono richieste ulteriori operazioni di aggiunta.
L’oggetto matrice è: oggetto_MiniMap ed è una classe da me creata con funzioni specifiche per migliorare l’esperienza di gioco.
create_new_miniMap è una funzione dell’oggetto oggetto_MiniMap.

Distinti Saluti
Mohammad Fumagalli Tahboub (Samir),
e Grazie per l’Attenzione riservatami