1. **Modalità d'esame**
   Ciascun candidato dovrà sviluppare il progetto descritto in questo documento e consegnarlo qualche giorno prima dell'esame. In sede d'esame verrà chiesto al candidato di mostrare e spiegare il codice prodotto, potrebbero esser fatte domande teoriche e proposte modifiche da fare sul momento.
   Verso la fine del documento è riportato un elenco di cose essenziali per poter presentare il progetto, seguito da un elenco di cose extra da poter sviluppare opzionalmente.

2. **Descrizione**
   Il progetto in questione è un videogioco roguelike da giocare da terminale prendendo in input delle stringhe ben definite (trovate l'elenco dei comandi in questo documento).
   Scopo del gioco: raggiungere l'uscita del labirinto col maggior numero possibile di gp (monete d'oro) ed exp (punti esperienza) senza farsi uccidere dai mostri.

   Oltre al file xml che descrive come costruire e stampare la mappa di gioco, sarà necessario leggere altri tipi di xml. Alcuni esempi sono presenti in questa cartella:
   https://drive.google.com/drive/folders/1Tbl_275ezqYj703V1otGZReR9QH1AHSt?usp=sharing
   (questi stessi esempi saranno riportati di seguito nel documento)

3. Meccaniche di gioco

Il giocatore si ritrova in un labirinto (può essere un'unica stanza o più stanze collegate tra loro) popolato da mostri che, se abbastanza vicini al giocatore lo inseguiranno o lo attaccheranno.

L'EROE

Il personaggio del giocatore ha una serie di parametri che ne descrivono lo stato:

lvl - il livello del personaggio, eliminando nemici può incrementare
hp - (hit points) i punti vita, se scendono a zero è game over
maxHp - il massimo dei punti vita, gli hp non possono mai superare questa soglia
mp - (mana points) i punti magia, servono per usare le pergamene
maxMp - il massimo di punti magia, gli mp non possono mai superare questa soglia
str - (strength) la forza del personaggio, usata negli attacchi corpo a corpo e con le armi pesanti
dex - (dexterity) la destrezza del personaggio, usata negli attacchi a distanza e con le armi leggere
mnd - (mind) l'acume del personaggio, usato per lanciare gli incantesimi di attacco
wis - (wisdom) la saggezza del personaggio, usata per lanciare gli incantesimi protettivi (e identificare oggetti)
res - (resistance) la resistenza del personaggio, usata per resistere ai danni
movT - (movement time) il tempo impiegato dal personaggio per spostarsi di una casella (float)
actT - (action time) il tempo impiegato dal personaggio per eseguire un'azione (float)
exp - (experience points) i punti esperienza accumulati eliminando i nemici, raggiunto un valore pari a (lvl * 100) il personaggio "spende" quel quantitativo di punti e sale di livello (ottenendo bonus alle statistiche). Ad esempio a lvl 1 servono 100 punti per salire di livello. Se dopo aver eliminato un nemico il personaggio arrivasse a 105 exp, scatterebbe il level up e i 5 punti avanzati rimangono, ora essendo a lvl 2 avrà bisogno di 200 punti per salire nuovamente (altri 195).
keys - il numero di chiavi che possiede il giocatore (servono ad aprire porte chiuse a chiave e forzieri)
gp - (gold pieces) monete d'oro raccolte dal giocatore, rappresentano (assieme a lvl ed exp) il punteggio del giocatore a fine partita

Il personaggio può portare con sé un inventario di massimo 8 elementi.
Gli elementi dell'inventario possono essere oggetto consumabili (come pozioni, bombe o chiavi), equipaggiamenti e incantesimi.
Gli equipaggiamenti si suddividono in armi e armature.
Il personaggio può equipaggiare una sola arma, un solo oggetto e una sola armatura alla volta (gli oggetti devono essere equipaggiati prima di poter essere utilizzati, equipaggiare un oggetto/arma/armatura conta come un'azione).
Gli incantesimi invece non vanno equipaggiati per essere utilizzati.

Quando il personaggio raggiunge la casella di uscita il gioco termina e si calcola il punteggio seguendo la formula: gp * 25 + exp (anche i punti "spesi", calcolabili con la sommatoria (n = 1, n -> lvl - 1) * 100)

Ad ogni turno il personaggio può eseguire un movimento o un'azione, il tempo all'interno del labirinto passerà (incrementando in base al valore di movT o actT del personaggio).
A questo punto tutti i nemici controllano se è arrivato il momento per loro di muoversi.

I NEMICI

Ogni nemico ha le stesse statistiche dell'eroe ma in più ha anche:

nextActTime - che indica QUANDO il nemico farà la prossima mossa (movimento o attacco)
attackRange - la distanza* massima (in caselle) da cui può attaccare l'eroe
sightRange - la distanza* massima (in caselle) in cui vede il personaggio (e inizia a seguirlo)
attackStat - stringa che indica quale statistica utilizza per attaccare (str,dex,mnd)

nextActTime inizialmente è 0.
ogni volta che un nemico si muove, il suo nextActTime aumenterà di un valore pari al suo movT, mentre ogni volta che il nemico attacca (o esegue altre azioni) allora il suo nextActTime aumenterà di un valore pari al suo actT

Dopo ogni azione/movimento del personaggio, bisogna controllare il valore di nextActTime di ogni nemico e confrontarlo con quanto tempo è passato NEL gioco. Ogni nemico per cui è arrivato il momento di agire, dovrà agire in quel momento (seguendo la sua AI).

Ad esempio se il gioco è appena iniziato, il tempo trascorso è pari a 0.
Nella scena c'è il personaggio (con movT = 1.5) e un goblin (con nextActTime = 5)
Il personaggio fa tre passi verso nord
Il tempo adesso è a 4.5 (0 → 1.5 → 3.0 → 4.5)
Il goblin ancora non si muove, la sua prima reazione avverrà quando il tempo sarà >= 5
Il personaggio fa un altro passo verso nord
Il tempo adesso è a 6
Il goblin ora agisce (si muove o attacca il personaggio in base alla sua AI)
Se ad esempio il goblin attacca e il suo actT = 4, allora la prossima volta che il goblin agirà sarà quando il tempo trascorso sarà arrivato a 9.
Se invece il goblin si muove e il suo moveT = 2, allora la prossima volta che il goblin agirà sarà quando il tempo trascorso sarà arrivato a 7.

L'intelligenza artificiale dei nemici (per ora) è abbastanza basilare.
Quando è giunto il momento di attivarsi per un nemico, ci sono tre possibilità
1) il nemico è abbastanza vicino all'eroe per attaccarlo (distanza in numero di passi) [attackRange] ---> esegue la sua azione di attacco
2) il nemico è abbastanza vicino all'eroe da vederlo (distanza in numero di passi) [sightRange] ---> esegue la sua azione di movimento verso l'eroe
3) il nemico è lontano ---> azione di movimento in una direzione casuale (casella libera)

* La distanza per la visuale e gli attacchi dei nemici è da considerarsi come "manhattan distance" ovvero si somma la distanza verticale e quella orizzontale.
Ad esempio se un nemico si trova in (1,0) e l'eroe si trova in (2,3) allora la distanza sarà 4 (una casella di distanza in orizzontale e 3 caselle di distanza in verticale).

PROVE DI ABILITA'

A volte al personaggio verrà richiesto di effettuare delle prove di abilità. La difficoltà di queste prove può variare, in media è 15.

Una prova di abilità funziona così:

ogni prova si basa su una delle statistiche del personaggio, viene lanciato un numero casuale tra 1 e 20 (compresi), ci si somma la statistica di riferimento, se il totale è >= alla difficoltà della prova, la prova è superata. Altrimenti la prova è fallita.

Il grado di fallimento di una prova dipende dalla differenza tra il punteggio ottenuto e la difficoltà della prova. Se ad esempio il totale di una prova di STR è 13 e la sua difficoltà era 15, allora il grado di fallimento è 2.

Diverse prove hanno effetti di fallimento diversi.

Aprire una porta chiusa a chiave (locked) senza usare una chiave (prova di STR con difficoltà 15) : se si fallisce la porta resta chiusa si subisce 1 danno per ogni 2 punti di grado di fallimento (ad esempio se si ottiene un punteggio totale di 10, il grado fallimento è 5 e si subiscono 2 danni).

Aprire un forziere senza chiave (prova di DEX con difficoltà 15) : se si fallisce il forziere si apre lo stesso ma scatta una trappola e si subisce 1 danno per ogni 3 punti di grado di fallimento (ad esempio se si ottiene un punteggio totale di 10, il grado di fallimento è 5 e si subisce 1 danno).

Identificare un oggetto (prova di WIS con difficoltà 15) : se si fallisce e il grado di fallimento è 10 o meno, l'azione è sprecata, se è maggiore di 10 l'oggetto rimane un mistero (non è più possibile identificarlo, in caso di piante è possibile utilizzarle, ma nell'inventario comparirà come erba non identificata, gli altri tipi di oggetti invece non saranno più utilizzabili)

EQUIPAGGIAMENTO

Il personaggio può portare con sé un massimo di 8 oggetti.

Gli oggetti si suddividono nei seguenti tipi:

Weapon: l'arma con cui il personaggio attacca. Prima di usarla per attaccare, deve essere equipaggiata.

Armor: l'armatura usata dal personaggio, offre dei bonus/malus alle statistiche o ai punti vita.

Potion: pozioni, possono essere utilizzati una sola volta

Herb: erbe, possono essere utilizzati una sola volta (anche se non sono stati identificati)

Scroll: pergamene degli incantesimi, consumano gli MP del personaggio per essere utilizzate

Questi oggetti possono essere trovati in giro per la mappa.

Tutti gli oggetti devono essere identificati prima di poter essere utilizzati (eccetto gli oggetti di partenza del personaggio e le erbe)

L'inventario ha 8 slot, ogni slot è identificato da un indice (da 0 a 7)

I vari oggetti sono descritti approfonditamente nella sezione **Elementi**

4. **Input del giocatore**

   L'input avviene tramite inserimento sul prompt.

   Dopo ogni input il gioco deve processare la situazione (eseguire l'azione del giocatore, muovere i nemici, etc etc)

   I tipi di input sono:

w, a, s, d	per muoversi di una casella rispettivamente verso nord, ovest, sud ed est
equip index	per equipaggiare un oggetto/arma/armatura dall'inventario
atk w/a/s/d	per attaccare con l'arma equipaggiata nella direzione indicata (nord/ovest/sud/est)
use w/a/s/d	per usare l'oggetto equipaggiato (la direzione serve solo ad alcuni oggetti)
cast index w/a/s/d	per usare una magia conosciuta dal personaggio nella direzione indicata
open w/a/s/d	per aprire una porta o un forziere nella direzione indicata porte non chiuse a chiave vengono aperte automaticamente porte chiuse a chiave e forzieri necessitano una chiave, il comando open controllerà nell'inventario del personaggio se c'è almeno una chiave e se presente chiederà al giocatore se vuole usarla (risposte y/n) se il giocatore non ha chiavi oppure risponde di no, chiederà al personaggio se vuole sfondare la porta (bruteforce) o scassinare il forziere (lockpick) (risposte y/n) se il giocatore risponde di sì allora verrà effettuata una prova di STR (per e porte) o una prova di DEX (per i forzieri)
take	per raccogliere tutti gli oggetti adiacenti al personaggio (alla distanza di 1 casella) quando si raccoglie una pianta viene subito effettuata una prova di identificazione
discard index	per scartare un oggetto (su quelli equipaggiati non si può fare)
identify index	identifica l'oggetto nello slot indicato, mostrando una descrizione approfondita dell'oggetto

5. **Elementi (classi, nemici, armi, etc)**

CLASSI DELL'EROE

Le specialità del personaggio saranno indicate da una "classe" (intesa come nei giochi di ruolo, non come in programmazione). Ogni classe indicherà le statistiche di partenza, le statistiche ottenute al level up e l'inventario di partenza. La rappresentazione xml segue questo schema:

```
<heroClass label="Warrior" >
    <baseStats hpMax="20" mpMax="0" str="4" dex="2" mdn="1" wis="2" res="3" movT="1"
actT="2" />
    <levelUpStats hpMax="3" mpMax="0" str="1" dex="0.5" mdn="0.3" wis="0.3" res="0.8"
movT="-0.05" actT="-0.05" />
    <startingEquipment>
        <weapon label="Longsword" />
        <protection label="Leather Armor" />
        <potion label="Healing Potion" />
    </startingEquipment>
</class>
```

All'inizio della partita deve essere fornito al giocatore un elenco delle classi caricate tramite xml e il giocatore deve poter scegliere quale classe giocare.

Esempi:
https://drive.google.com/open?id=1T2YMJwvKnoUcDjJuCMwE7_C4gTYZS1Kj

NEMICI

Le caratteristiche di ogni nemico saranno riportate in xml in questo modo

```
<enemy label="Goblin" ch="0x0047">
    <baseStats hpMax="6" str="4" dex="2" mnd="1" wis="1" res="0" movT="1.5" actT="2.5" />
    <enemyStats atkStat="str" atkRange="1" sight="4" />
</enemy>
```

Ci sarà un xml che rappresenterà il “dizionario dei mostri” con le descrizioni di tutti i mostri.
Poi ogni stanza del labirinto presenterà al suo interno l’elenco di mostri (e oggetti) presenti nella stanza

Esempi:
https://drive.google.com/open?id=1MknotkhuCBwyXeB_b6oyAPrcMi7vngo9

ELEMENTI DELLE STANZE

Gli elementi presenti in ciascuna stanza saranno descritti nell'xml che descrive il labirinto.
Le stanze potranno avere dei <loot> (oggetti da raccogliere), delle <chest> (scrigni chiusi a chiave) e degli <enemy> (i nemici). Per ciascuno di loro indica le coordinate (relative alla stanza) di questi elementi e la label identificativa dell'elemento (ad esempio un enemy con label "Goblin" farà riferimento al "Goblin" inserito nel "dizionario dei mostri", un loot con label "Longsword" farà riferimento all'arma "Longsword" inserita nel "dizionario delle armi"). Nel caso di <loot> e <chest> viene anche indicato l'attributo type che indica il tipo di oggetto da raccogliere (che indica a quale "dizionario" appartiene: weapon, potion, herb…).
Se un loot è di tipo key allora non va cercato in nessun "dizionario", raccoglierlo dovrà incrementare il valore di keys possedute dal personaggio di un valore pari al numero riportato nell'attributo label.
Se un loot o una chest è di tipo gp allora non va cercato in nessun "dizionario", il numero indicato nella label indicherà il numero di monete ottenute.

Lo schema sarà questo

```
<room x=5 y=2 width=15 height=10 label='arena' chWall='0x2593' chFloor='0x00B7'
chPath='0x0023'  >
    <door x=14 y=1 room='kitchen' locked=0 chOpen='0x004F' chLocked='0x00D8'/>
    <loots>
        <loot x=10 y=2 type="key" label="1" />
        <loot x=4 y=2 type="gp" label="15" />
        <loot x=1 y=8 type="weapon" label="Axe" />
        <loot x=8 y=5 type="herb" label="Poison Herb"  />
    </loots>
    <treasureChests>
        <chest x=8 y=6 type="gp" label="250"  />
    </treasureChests>
    <enemies>
        <enemy x=3 y=3 label="Goblin" />
        <enemy x=8 y=7 label="Crossbow Goblin" />
    </enemie>
</room>
```

https://drive.google.com/open?id=1iSJLMH5ORq24SSP70kxhjuAmDZs4pVnC

ARMI

Ogni arma sarà caratterizzata dal tag <weapon> e avrà:

quattro attributi principali:

- label - il nome dell'arma, con cui verrà identificata anche dai forzieri
- durability - quante volte può essere usata prima che venga distrutta (armi antiche, con proiettili, da lancio o bombe hanno un numero limitato di utilizzi) se il valore è 0 allora si può usare infinite volte
- range - la distanza massima (rispetto al personaggio) del punto di applicazione (origine dell'attacco). Se è 0 allora l'attacco ha origine dal personaggio (come nel caso di qualsiasi attacco corpo a corpo), se è maggiore di 0 allora bisogna cercare un nemico nelle caselle davanti al personaggio (nella direzione in cui rivolge l'attacco).
  Il numero di caselle da controllare è pari al valore di range (se range = 5 bisogna controllare le prime 5 caselle davanti al personaggio), il primo nemico (il più vicino) trovato in queste caselle diventa l'origine dell'attacco (immaginate una freccia esplosiva, come colpisce un bersaglio, esplode, l'esplosione potrebbe danneggiare anche i nemici nelle caselle adiacenti a quello colpito).
  Se invece NON ci sono nemici allora il punto di applicazione è l'ultima casella controllata (se range = 5 e in queste cinque caselle non ci sono nemici, l'attacco ha comunque origine nella quinta casella, immaginate ancora una freccia esplosiva, se non colpisce nessun nemico cade comunque a terra -nella quinta casella- ed esplode lì)

l'origine dell'attacco e la direzione in cui guarda il personaggio servono a capire quale caselle saranno bersagliate dall'attacco

opzionalmente il tag <weapon> potrà avere un tag interno <bonusStats> che a sua volta avrà una serie di tag figli (fratelli tra loro) chiamati <effect>. Ogni <effect> rappresenta un bonus (o un malus) ad una statistica del personaggio (applicata solo quando l'oggetto è equipaggiato) e avrà i seguenti attributi:

- stat - quale statistica deve subire il bonus/malus
- value - il valore del bonus/malus

il tag <weapon> avrà sempre un tag interno <areaOfEffect> che a sua volta avrà una serie di tag figli (fratelli tra loro) chiamati <square>. Ogni <square> rappresenta una casella bersagliata e avrà i seguenti attributi

- fd - forward distance : la distanza nella direzione in cui guarda il personaggio, partendo dal punto di applicazione
- rd - right distance : la distanza nella direzione "a destra" rispetto a dove guarda il personaggio, partendo dal punto di applicazione
- pot - la potenza dell'attacco (fa parte della formula per attaccare)
- stat - quale statistica utilizzare per calcolare il danno

Esempio:

```
<weapon label="Longsword" range="0" durability="0">
    <bonusStats>
        <effect stat="str" value="2.0" />
        <effect stat="dex" value="1.0" />
    </bonusStats>
    <areaOfEffect>
        <square fd="1.0" rd="0.0" stat="str" pot="1.0" />
    </areaOfEffect>
</weapoon>
```

Armi di esempio (con schemi di quali caselle vengono bersagliate) sono reperibili qui:
https://drive.google.com/open?id=1WTBlJjVdEscnUcS9CPNxvEC2w1v0bePu

Il calcolo del danno avrà una delle seguenti forme:
se la statistica utilizzata dall'arma è "none" allora l'attributo "pot" indica il valore assoluto dell'attacco:
weapon.pot - defender.res
se invece la statistica è una di quelle del personaggio, la formula sarà
attacker.stat * weapon.pot - defender.res
il danno non può mai essere inferiore a 0

Le armi vengono utilizzate semplicemente col comando
atk
(seguito dalla direzione w/a/s/d)

Prima di poter essere usate per attaccare è necessario equipaggiare l'arma che si vuole utilizzare.

Se si attacca senza avere nessuna arma equipaggiata allora l'effetto sarà come utilizzare un'arma che colpisce unicamente nella casella di fronte al personaggio (nella direzione in cui attacca) usando la statistica str con pot = 1

PROTEZIONI

Ogni protezione sarà caratterizzata dal tag <protection> e avrà un attributo label che ne indica il nome, come child avrà poi uno o più tag di tipo <effect>

ciascun tag <effect> avrà come attributi:

- stat - la statistica su cui ha effetto
- value - il valore del bonus/malus a quella statistica

Gli effetti di una protezione devono essere validi solo se la protezione è equipaggiata (e il personaggio può equipaggiare una sola protezione per volta).

```
<protection label="Leather Armor" ch="0x00BF">
    <effect stat="maxHp" value="5.0" />
    <effect stat="res" value="2.0" />
</protection>
```

POZIONI

Ogni pozione sarà caratterizzata dal tag <potion> e avrà un attributo label che ne indica il nome, come child avrà poi uno o più tag di tipo <effect> . Ciascun tag <effect> avrà come attributi:

- stat - la statistica su cui ha effetto
- value - il valore del bonus/malus a quella statistica

Gli effetti di una pozione vengono applicati quando è utilizzata. Ogni pozione viene consumata (scartata) nel momento in cui viene usata.

```
<potion label="Healing Potion" ch="0x00BF" >
    <effect stat="hp" value="10.0" />
</protection>
```

ERBE

Ogni erba sarà caratterizzata dal tag <herb> e avrà un attributo label che ne indica il nome, come child avrà poi uno o più tag di tipo <effect> . Ciascun tag <effect> avrà come attributi:

- stat - la statistica su cui ha effetto
- value - il valore del bonus/malus a quella statistica

Gli effetti di un'erba vengono applicati quando è utilizzata. Ogni erba viene consumata (scartata) nel momento in cui viene usata.
La differenza tra erbe e pozioni è che le erbe non identificate possono essere utilizzate.

```
<herb label="Poison Herbs" ch="0x00BF" >
    <effect stat="hp" value="-2.0" />
    <effect stat="maxHp" value="-2.0" />
</herb>
```

Esempi: https://drive.google.com/open?id=1XQ-Km8zlkp_JS_PIuKZbP7twW4ED1W6J

PERGAMENE

Ogni pergamena sarà caratterizzata dal tag <scroll>. Il loro effetto può essere come l'attacco di un'arma, come l'utilizzo di una pozione o entrambe le cose. Si usano tramite il comando cast, possono essere usate infinite volte (a patto di avere sufficienti MP) e nel file xml avranno i seguenti attributi principali:

- label - il nome dell'arma, con cui verrà identificata anche dai forzieri
- mpCost - quanti MP è necessario avere (e consumare) per poter lanciare l'incantesimo
- range - la distanza (rispetto al personaggio) del punto di applicazione (origine dell'attacco) funziona allo stesso modo del range delle armi

  l'origine dell'attacco e la direzione in cui guarda il personaggio servono a capire quale caselle saranno bersagliate dall'attacco

  opzionalmente il tag <scroll> potrà avere un tag interno <selfEffects> che a sua volta avrà una serie di tag figli (fratelli tra loro) chiamati <effect>. Ogni <effect> rappresenta un bonus (o un malus) ad una statistica del personaggio (applicata quando l'incantesimo viene lanciato, come quando si utilizza una pozione) e avrà i seguenti attributi:
  - stat - quale statistica deve subire il bonus/malus
  - value - il valore del bonus/malus
  - potStat - quale statistica usare per calcolare l'effetto
    (a differenza delle pozioni gli effetti degli incantesimi sul personaggio vengono calcolati come per gli attacchi delle armi **chracter.potStat * scroll.pot**)

  il tag <scroll> può avere un tag interno <areaOfEffect> che a sua volta avrà una serie di tag figli (fratelli tra loro) chiamati <square>. Ogni <square> rappresenta una casella bersagliata e avrà i seguenti attributi
  - fd - forward distance : la distanza nella direzione in cui guarda il personaggio, partendo dal punto di applicazione
  - rd - right distance : la distanza nella direzione "a destra" rispetto a dove guarda il personaggio, partendo dal punto di applicazione
  - pot - la potenza dell'attacco (fa parte della formula per attaccare)
  - stat - quale statistica utilizzare per calcolare il danno

Esempio:

```
<scroll label="Guiding Light" mpCost="3.0" range="20"ch="0x00FE">
    <selfEffects>
        <effect stat="hp" pot="2.0" potStat="wis" />
    </selfEffects>
    <areaOfEffect>
        <square fd="0.0" rd="0.0" stat="mnd" pot="0.8" />
    </areaOfEffect>
</scroll>
```

Altri esempi:
https://drive.google.com/open?id=1h-gwWo6In6B7rpDB0GUm2Y9NxGAOQJTW

6. Interfaccia

L'interfaccia occupa un totale di 120 x 25+n (dove n è il numero di righe dedicate alla history -definita più in basso). L'interfaccia è suddivisa in 4 porzioni.

Mappa

Un riquadro di 81 x 21 (w x h) caratteri rappresenta la mappa, ogni spazio indica una casella. Il personaggio deve essere al centro della mappa.

Potete scegliere liberamente come trattare il posizionamento del personaggio quando si trova sui bordi della mappa, ad esempio nella coordinata (1,1) si può decidere di lasciare caselle nere per le coordinate fuori dalla mappa (come ad esempio (-1,-1)) oppure di far sì che l'inquadratura non possa andare oltre quelle coordinate che porterebbero a visualizzare coordinate esterne alla mappa (in tal caso se il personaggio si trova in (1,1) invece di comparire al centro della schermata, si troverebbe vicino all'angolo in alto a sinistra).

Alcuni caratteri "globali" (come quello per il personaggio, per i forzieri, per i soldi -gp- e per le chiavi -key-) sono definiti nel tag **<game>** dell'xml sulla generazione della mappa.

Scheda Personaggio

A destra della mappa c'è la scheda del personaggio con le seguenti righe 18

Nome
Classe
Lvl

Hp	n/m			Mp	n/m		
STR	n	DEX	n	MND	n	WIS	n
ActT	n	MovT	n			RES	n
Exp	n/m						
Gp	n			Keys	n		

Inventory
0 E : itemName (la E indica che è equipaggiato)
1 : itemName
2 E : itemName
3 : (se non c'è scritto il nome dell'oggetto, vuol dire che lo slot è libero)
4 :
5 :
6 :
7 :

(lasciare 2 righe libere alla fine, se si dovesse espandere l'inventario fino a 10 elementi)

History

Sotto la mappa ci sono N righe con la history delle cose avvenute (le prove fatte dal giocatore e i loro esiti, gli attacchi del giocatore e dei nemici e gli eventuali danni inflitti, le porte aperte, etc etc)
il numero N è configurabile (tramite file o all'inizio della partita), se si sceglie 5 ad esempio, si vedranno le ultime 5 azioni fatte (dal giocatore o dai nemici)

Input

Infine un'ultima riga deve chiedere al giocatore cosa vuole fare e prendere l'input del giocatore.

Esempio:
https://drive.google.com/open?id=1Fj1-ljbAHEakWuESfqOUJZIFt_CJg2Rm

7. **Generazione mappa**
   La mappa sarà caricata a runtime mediante l’apertura di un file xml. Questo file seguirà lo standard definito nel documento che avete già ricevuto.

8. **MVP (Minimum Viable Product) ovvero il minimo richiesto**
   Il progetto, per essere ammesso in sede di discussione deve presentare almeno le seguenti cose implementate e funzionanti:
   - generazione della mappa e gestione fine partita (raggiunta la casella di uscita)
   - le armi possono non usare la regola dei punti di applicazione e dare soltanto i bonus alle statistiche (quindi niente effetti ad area o a distanza, gli archi funzioneranno come le spade)
   - si possono tralasciare le prove di abilità (quindi gli oggetti si identificheranno sempre al primo colpo, le porte e i forzieri si apriranno sempre anche senza chiave)

9. **VIP (Very important projects)**
   Chi implementa queste caratteristiche otterrà dei bonus durante l'esame:
   - Colore
     ad esempio la lettera P del personaggio è sempre verde
     le lettere dei nemici cambiano in base a quanta salute hanno (75-100% sono gialli chiari, 50-75% sono gialli scuri, 25%-50% sono rossi chiari, 0-25% sono rossi scuri)
     le E degli oggetti equipaggiati nell'inventario sono sempre gialle
     gli oggetti nell'inventario cambiano colore in base al tipo (blu chiaro per le pozioni, grigio per le armi, verde chiaro per le piante, viola chiaro per le pergamene, bianco per le armature)
     i valori di HP e MP variano in base alle percentuali (come per i nemici sulla mappa) seguendo lo schema:
     HP (75-100% verde -> 50-75% giallo chiaro -> 25-50% giallo scuro -> 0-25% rosso chiaro -> 0 rosso scuro)
     MP (75-100% viola chiaro> 50-75% viola scuro -> 25-50% blu scuro -> 0-25% blu chiaro -> 0 grigio).
     Per abilitare la colorazione nel terminale è possibile usare la libreria termcolor.hpp (https://github.com/ikalnytskyi/termcolor). E' necessario includere il file termcolor.hpp e usare il comando std::cout<<termcolor::COLORE; per abilitare la stampa a schermo del COLORE selezionato tra quelli disponibili (vedi README su github) o definirlo con i valori specifici. Attenzione, in questo modo tutte le scritte successive saranno colorate con COLORE a meno di resettare (termcolor::reset) o assegnare un altro colore.

   - Nebbia di guerra: tutta la mappa non sarà visibile fin da subito ma rivelata piano piano:
     solo le caselle entro una distanza N dall'eroe saranno "rivelate"
     una volta "rivelata" una casella rimane sempre visibile
     i nemici invece saranno visibili solo se si trovano entro una distanza N
     OPPURE se hanno attaccato il personaggio nel loro ultimo turno

   - I nemici, per seguire l'eroe, sfruttano l'algoritmo A* (questo algoritmo non fa parte del programma del corso ed è un argomento molto ampio già da sé, qui potete trovarne una buona descrizione https://www.raywenderlich.com/3016-introduction-to-a-pathfinding)