Scopo

Lo scopo del programma è la realizzazione di un gioco roguelike. Questo tipo di gioco è caratterizzato da un'interfaccia completamente fruibile da riga di comando mediante l'uso di caratteri ASCII. Il programma dovrà essere in grado di leggere un file di configurazione `.xml` (a runtime) contenente tutti i dati di ingresso della partita. Generalmente ogni gioco avrà un eroe che, esplorando un mondo formato da stanze collegate tra loro e interagendo con esso (saranno presenti nemici e oggetti), dovrà trovare l'uscita.

Generazione Mappa

La mappa di gioco sarà rappresentata da una matrice di `unsigned char` di dimensioni definite da una specifica voce del file di configurazione avente il tag `game` :

```
<game w=256 h=256 exitX=0 exitY=0 chExit='0x0057' chPath='0x0023' chHero='0x0040' chChest='0x00A4' chKey='0x006B' chGP='0x0024'/>
```

I parametri sono:

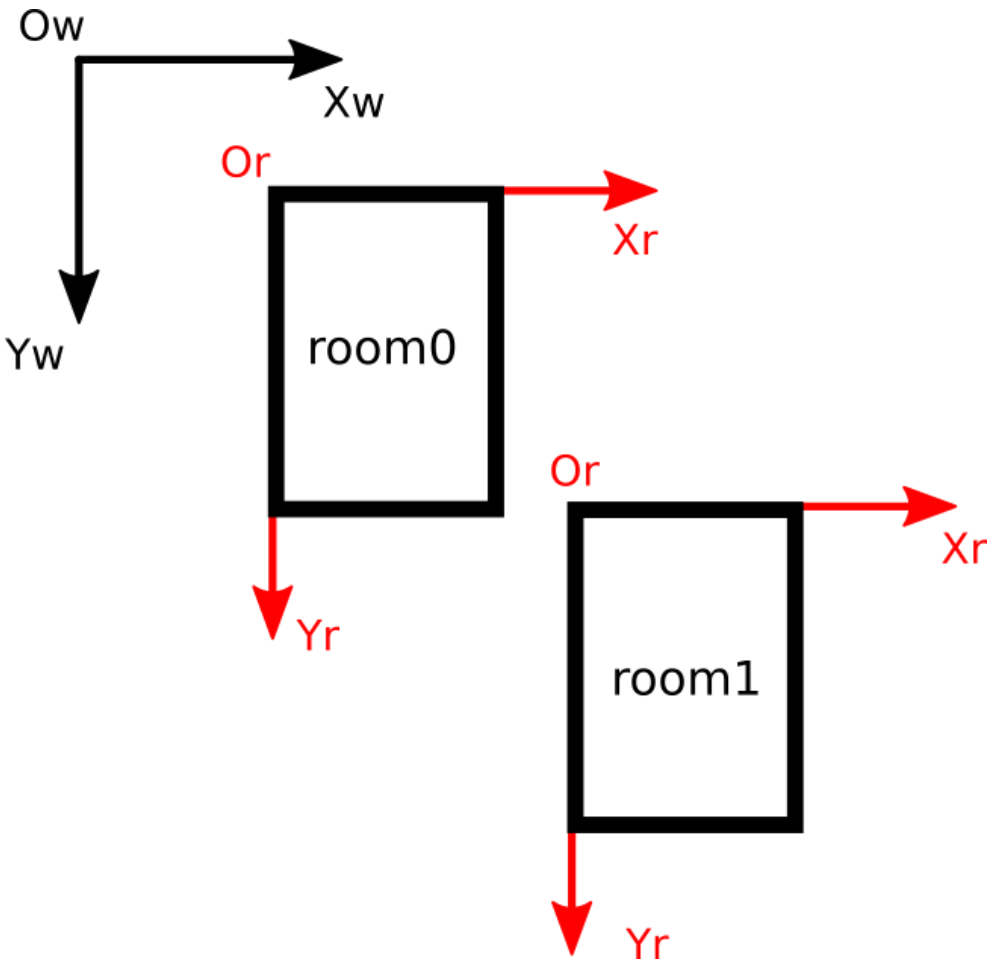
1. `w` e `h` : dimensione della mappa di gioco
2. `exitX` e `exitY` : posizione dell'uscita della mappa (condizione di vittoria) nel riferimento globale (O_w)
3. `chExit` : codice UNICODE del carattere per rappresentare l'uscita
4. `chPath` : codice UNICODE del carattere per rappresentare il pavimento esterno (percorsi che collegano le stanze)
5. `chHero` : codice UNICODE del carattere per rappresentare l'eroe
6. `chChest` : codice UNICODE del carattere per rappresentare un forziere
7. `chKey` : codice UNICODE del carattere per rappresentare una chiave
8. `chGP` : codice UNICODE del carattere per rappresentare le monete d'oro

La sezione del file `.xml` specifica per l'aggiunta delle stanze (e relativi elementi) sarà composta da un tag `<rooms>` che conterrà tutte le stanze (suddivise per tag `<room>`). Ogni stanza singola sarà definita come segue:

```
<room x=5 y=2 width=15 heigth=10 label='arena' chWall='0x2593' chFloor='0x00B7' chPath='0x0023'>
</room>
```

I parametri della stanza sono:

1. `x` e `y` : origine del sistema di riferimento locale O_r della stanza riferito al sistema di riferimento globale O_w. L'asse `x` è diretto verso destra mentre quello `y` verso il basso. Si veda lo schema seguente



Ovviamente le coordinate saranno sempre definite tramite numeri interi (ogni `char` rappresenta il minimo elemento)

2. `width` e `height` : dimensioni della stanza lungo `x` e `y` rispettivamente. Esse sono calcolate nel sistema di riferimento locale O_r
3. `label` : nome della stanza
4. `chWall` : codice UNICODE del carattere per rappresentare un muro (limite esterno della stanza)
5. `chFloor` : codice UNICODE del carattere per rappresentare il pavimento interno della stanza

Le stanze collegate potranno avere le mura esterne sovrapposte (nel caso di due stanze vicine che condividono il muro) oppure potranno essere separate (lo si vede dalle coordinate e dall'estensione). Se due stanze collegate sono separate, dovrà essere disegnato un percorso esterno (con il relativo carattere) da una porta all'altra. Questo percorso non dovrà essere necessariamente il più piccolo possibile.

Elementi

All'interno di ogni stanza potranno essere presenti più elementi (nemici, oggetti, armi, ecc...) che verranno aggiunti mediante tag.

Porte

Le porte saranno definite mediante un tag `<door>` :

```
<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 chUnlocked='0x004F' chLocked='0x00D8'/>
</room>
```

Ogni porta avrà come parametri:

1. `x` e `y` che rappresentano la posizione della porta nel sistema di riferimento locale della stanza O_r
2. `room` che rappresenta la `label` della stanza verso cui la porta indirizza
3. `locked` che è un `bool` (0 o 1) e che indica se la porta richiede una chiave per essere aperta

4. `chUnlocked` : codice UNICODE del carattere per rappresentare la porta chiusa (non locked)
5. `chLocked` : codice UNICODE del carattere per rappresentare la porta chiusa (locked)
   Quando la porta sarà aperta verrà sostituita dal carattere `chFloor` della stanza

Rappresentazione grafica

Per il disegno dei caratteri si dovrà sfruttare la codifica UTF-8. Essa permette di rappresentare diversi codici UNICODE mediante l'uso di uno o più bytes (8 bit ciascuno). La scelta dipende dal valore del codice UNICODE a disposizione e segue una tabella così composta:

Hex	bytes
U-00000000 – U-0000007F	0xxxxxxx
U-00000080 – U-000007FF	110xxxxx 10xxxxxx
U-00000800 – U-0000FFFF	1110xxxx 10xxxxxx 10xxxxxx
U-00010000 – U-001FFFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
U-00200000 – U-03FFFFFF	111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx
U-04000000 – U-7FFFFFFF	1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx

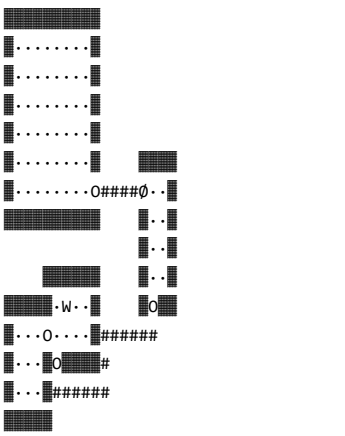
Il numero di bytes dipende, ovviamente, dal valore da rappresentare e parte da un minimo di 1byte (rappresentato da 8bit). Il bit più a sinistra è sempre `0` , gli altri sono il codice binario del carattere UNICODE da rappresentare. Si immagini, ad esempio, di avere un codice `U+0023` (carattere `#`) che, in esadecimale, è il numero `35` rappresentabile con un singolo byte (formato da 7 bit) e in binario: `100011` . Seguendo la codifica, il carattere sarà, quindi, esprimibile in codice binario in questo modo: `0 0100011` (1+7 bit) che in forma esadecimale è uguale a: `0x0023` . Effettuando un `printf("%c",0x0023)` si otterrà, infatti, il carattere voluto.
Se il carattere da stampare ha un codice UNICODE: `U+00AC` (carattere `¬`), il numero intero riferito al codice esadecimale è `172` (maggiore di `128`) quindi si dovrà usare una codifica a 2byte (`172<2047`). Il primo byte avrà i primi 3 bit a sinistra sempre `110` mentre l'altro avrà i primi 2 sempre `10` . Si noti come i primi tre bit rappresentano il numero di byte da usare seguito da `0` , ovvero `110` vuol dire che la codifica richiede 2byte (`11`) da usare (nel caso `1110` si avranno 3byte) e così via. I byte successivi al primo saranno sempre composti dai primi 2 bit a sinistra uguali a `10` . Le posizioni intermedie saranno riempite con il binario del carattere da trovare. Nel caso considerato, il numero `172` è rappresentabile in binario come: `0010101100` (11bit essendo il numero di bit totali 16 ma 5 sono usati per la codifica) quindi si otterrà il numero: `110 00010` , `10 101100` che in esadecimale è rappresentabile come `0xc2` , `0xa2` . Per stampare a schermo il carattere sarà, quindi, sufficiente chiamare `printf("%c%c",0xc2,0xa2)` .

Esempio

Come esempio, se la configurazione è così definita:

```
<game w=20 h=20 exitX=6 exitY=10 chExit='0x0057' chPath='0x0023'/>
<rooms>
<room x=0 y=0 width=10 height=8 label='arena' chWall='0x2593' chFloor='0x00B7' >
    <door x=9 y=6 room='kitchen' locked=0 chUnlocked='0x004F' chLocked='0x00D8'/>
</room>
<room x=14 y=5 width=4 height=6 label='kitchen' chWall='0x2593' chFloor='0x00B7'>
    <door x=0 y=1 room='arena' locked=1 chUnlocked='0x004F' chLocked='0x00D8'/>
    <door x=1 y=5 room='hall' locked=0 chUnlocked='0x004F' chLocked='0x00D8'/>
</room>
<room x=4 y=9 width=6 height=4 label='hall' chWall='0x2593' chFloor='0x00B7'>
    <door x=1 y=3 room='kitchen' locked=0 chUnlocked='0x004F' chLocked='0x00D8'/>
    <door x=0 y=2 room='bedroom' locked=0 chUnlocked='0x004F' chLocked='0x00D8'/>
</room>
<room x=0 y=10 width=5 height=5 label='bedroom' chWall='0x2593' chFloor='0x00B7'>
    <door x=1 y=4 room='hall' locked=0 chUnlocked='0x004F' chLocked='0x00D8'/>
</room>
</rooms>
```

Si otterrà un risultato come quello seguente:



Oppure