PROGETTO DI PROGRAMMAZIONE

- A.A. 2016/2017

Ivan Lanese

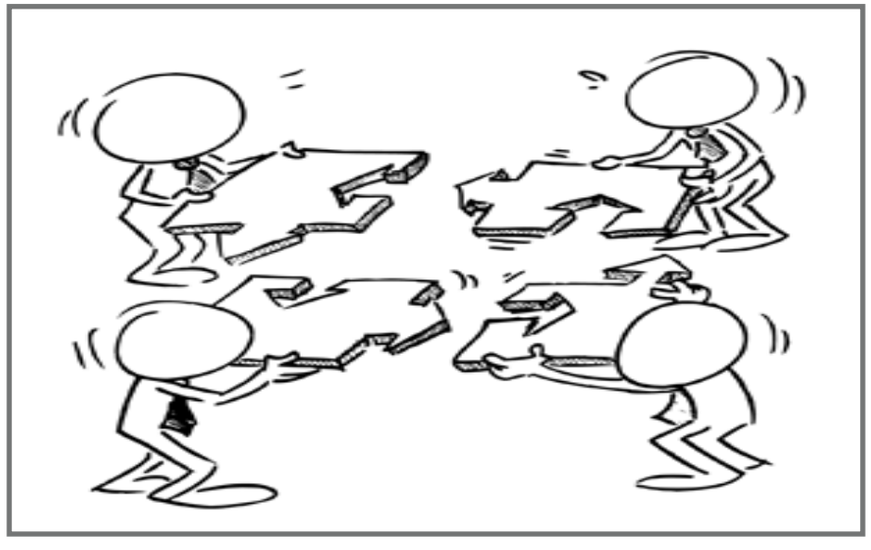
IL PROGETTO SERVE PER

- Farvi sperimentare le tecniche viste a lezione su scala più grande di quanto possibile in una lezione di 3 ore, in direzione di quello che succede nel mondo reale
  - I progetti reali sono spesso MOLTO più grandi
- Mettervi alla prova in uno scenario che lascia spazio alla vostra fantasia, ma pone anche alcuni vincoli

VALUTAZIONE DEL PROGETTO

- Il progetto è parte integrante dell’esame ed è obbligatorio
- Ha un voto massimo di 8 punti che si sommano al voto dello scritto (max 24)
- Il lavoro viene svolto in gruppi, ma la valutazione del progetto è individuale
  - Studenti dello stesso gruppo possono prendere voti diversi
- Chi non ottiene almeno 1 punto alla discussione orale dovrà presentare un nuovo progetto.
- Il progetto potrà essere presentato in qualunque momento entro il 1/11/2017.

I GRUPPI



- I gruppi sono di 3/4 studenti
- Inviate una email con l’elenco dei componenti del gruppo a:
  - Ivan.lanese@gmail.com
- Riceverete un numero associato al vostro gruppo, utile per tutte le comunicazioni successive

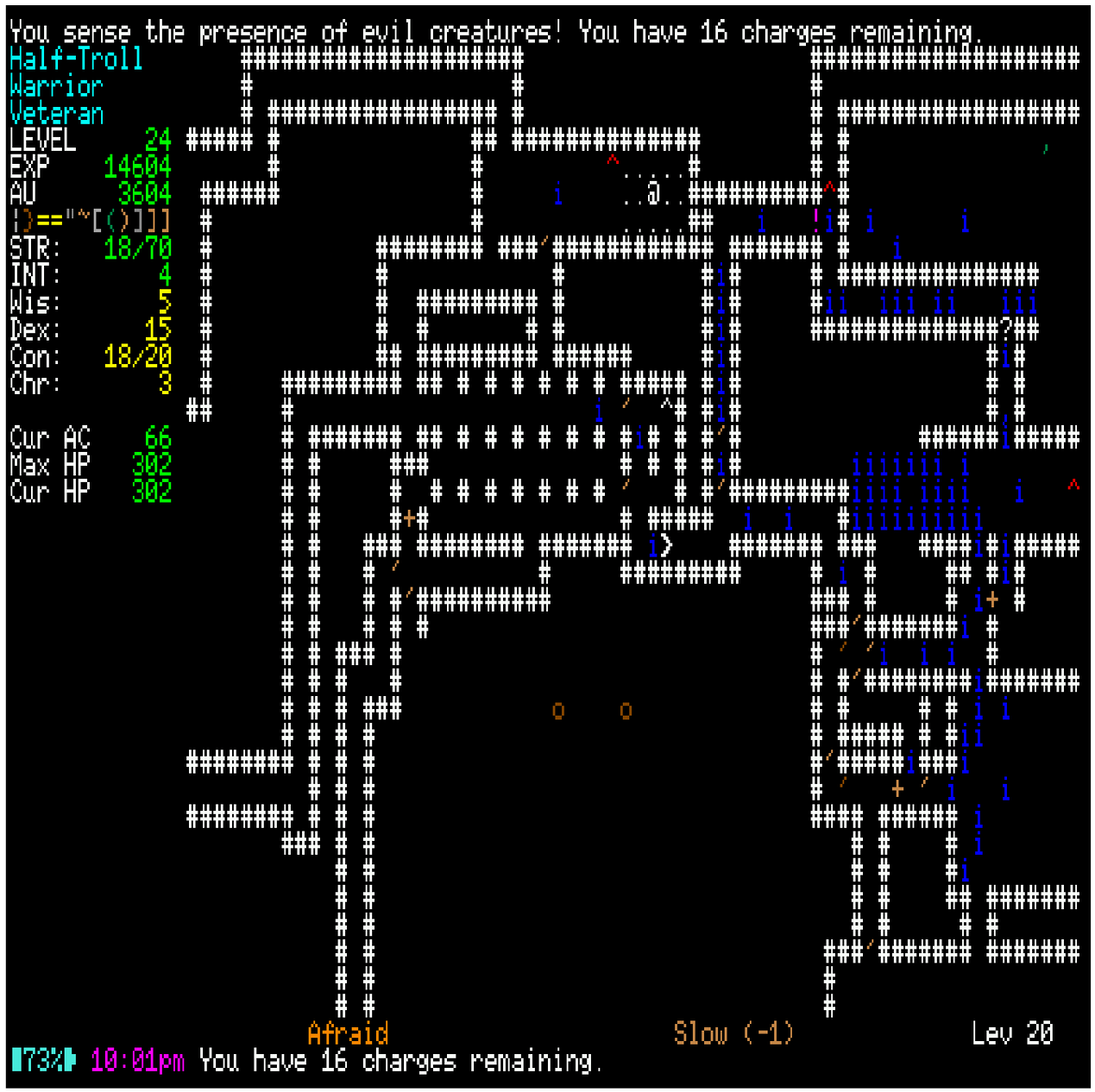
DISCUSSIONI

- Quando avrete terminato il progetto, inviate per email
  - Il codice del progetto
  - Una breve relazione (3-5 pagine) che presenti le principali scelte che avete fatto nella definizione del gioco e nella sua implementazione

- In risposta sarete convocati per la discussione orale

- Tutti i componenti del gruppo dovranno partecipare alla discussione orale
  - Portate il vostro computer

- La discussione riguarda solo il progetto



CONTENUTO DEL GIOCO

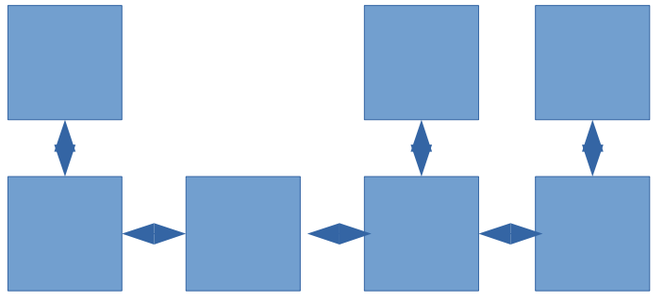
- Una mappa a livelli
- Un personaggio gestito dal giocatore
- Un numero variabile di personaggi gestiti dal computer, ad esempio
  - Mostri
  - Avversari
  - Aiutanti
  - ...

SVOLGIMENTO DEL GIOCO

- Le regole del gioco e lo scopo del gioco non fanno parte della specifica: li decidete voi

- Requisiti minimi e obbligatori:
    - la mappa viene generata un livello alla volta ed esplorata dinamicamente
    - il gioco è a turni, in ogni turno muove prima il giocatore umano poi, in ordine prefissato muovono i personaggi gestiti dal computer
    - a fine turno viene stampata a video la mappa del livello corrente aggiornata
    - la partita deve terminare, con la vittoria o la sconfitta del giocatore umano

LA MAPPA



- La mappa è a livelli, collegati da scale, e il gioco parte dal livello 1
- Ogni livello è composto da stanze, il cui numero aumenta all’aumentare del livello
- Le stanze sono disposte su una griglia rettangolare, ma non occupano tutte le posizioni della griglia
- Le stanze sono collegate da porte, ma non sempre ci sono porte per tutte le connessioni possibili (tutte le stanze sono raggiungibili in qualche modo)
- Un livello viene creato quando viene raggiunto per la prima volta, da quel momento in poi la sua struttura non cambia
- Non c’è un limite al numero di livelli

I GIOCATORI

- La mappa accetta un personaggio controllato dal giocatore umano, e un numero variabile di personaggi gestiti dal computer
- Nuovi personaggi gestiti dal computer vengono creati ogni volta che si raggiunge un nuovo livello
- I personaggi agiscono a turno muovendosi sulla stessa mappa

- Azioni obbligatorie:
  - movimento tra le stanze di un livello e su/giù dalle scale
  - almeno un’azione di interazione (combattimento, raccolta/rilascio oggetti, ...)
- Il personaggio umano viene controllato dalla console. Le azioni vengono indicate al gioco attraverso opportuni comandi. Gli altri personaggi sono gestiti interamente dal computer
- Se le regole (e lo scopo) del gioco che avete stabilito lo richiedono, includete altre azioni.

I TURNI DI GIOCO

- Stabilite delle regole per i turni di gioco che identifichino univocamente l’inizio e la fine del turno e le azioni possibili

- Alla fine di ogni turno la mappa del livello corrente viene stampata con la posizione aggiornata dei personaggi

- Spunti/Esempi:
  - “il turno è composto da una azione di movimento e una di combattimento”
  - “il turno è composto da un’unica azione a scelta tra: movimento, acquisto, o rifornimento”
  - “per ogni turno il giocatore ha a disposizione X punti-azione da spendere. Ogni azione ha un costo specifico”

SCOPO DEL GIOCO

- Lo scopo del gioco non fa parte della specifica
- Stabilite un obbiettivo da raggiungere per la vittoria, ad esempio
  - raggiungere un determinato punteggio/guadagnare una certa somma di denaro/raggiungere un certo livello di esperienza...
  - trovare un certo oggetto
- Stabilite delle condizioni che causino la sconfitta
  - morte in combattimento
  - morte per mancanza di risorse (cibo, acqua, ...)

IMPOSTAZIONE DEL PROGETTO

- Il progetto deve essere realizzato usando le classi
- Il progetto è organizzato in più file
- Ad ogni classe corrispondono due file: es. Node.hpp e Node.cpp
- .hpp è l’estensione per l’interfaccia
- L’interfaccia definisce il tipo Node

NODE.HPP

```
class Node{
protected:
   int field;
   …
public:
    …
   void method();
   …




};
```

NODE.CPP

```
#include "Node.hpp"


void Node::method(){

   //do something

}
```

in ogni file in cui si usa il tipo Node occorre importare `Node.hpp`

NODE.HPP

```
#ifndef NODE_HPP_
#define NODE_HPP_
```

```
class Node{
protected:
   int field;
   …
public:
   …
   void method();
   …

};
```

```
#endif    /* NODE_HPP_ */
```

Per evitare l'inclusione multipla della stessa interfaccia si utilizzano le direttive `#define`, `#ifndef` e `#endif` come mostrato a fianco.

UTILITIES: IL DADO

- Per generare numeri casuali tra 1 e 6 si può usare:

```
int die = (rand() % 6) +1;
```

- Occorre inizializzare il generatore di numeri casuali con:

```
srand(time(0));
```

- Occorrono alcune inclusioni:

```
#include<ctime>      // for the time() function
#include<cstdlib>    // for the srand() and rand() functions
```

- Potete modificare il codice a piacimento per generare valori casuali in range diversi