

PROGETTO DI PROGRAMMAZIONE

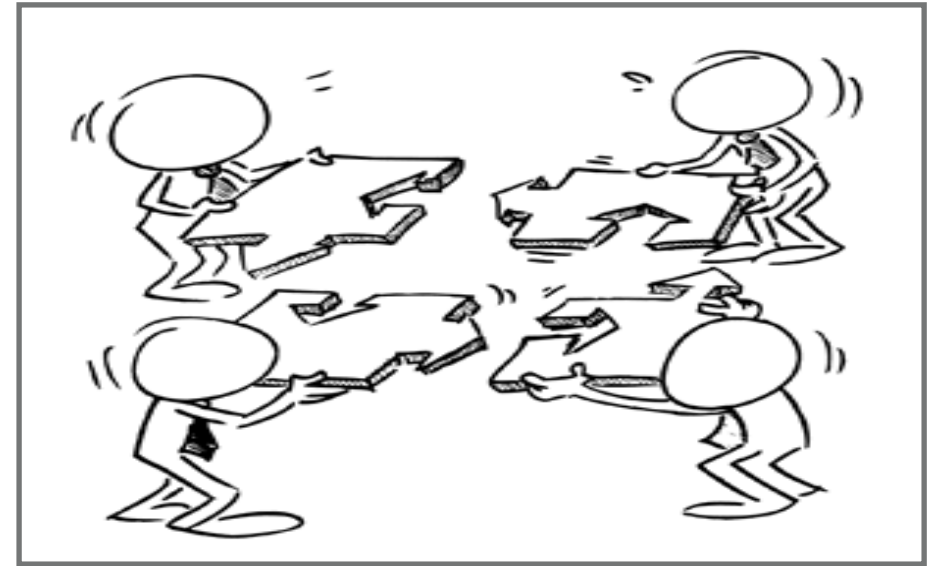
A.A. 2015/2016

Elena Giachino

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 Novembre 2016.

I GRUPPI



- I gruppi sono di 3/4 studenti
- Inviare una email con l'elenco dei componenti del gruppo a:
 - elena.giachino@unibo.it
 - vincenzo.mastandrea3@unibo.it
- Riceverete un numero associato al vostro gruppo, utile per tutte le comunicazioni successive
- Quando avrete terminato il progetto, inviate il codice per email e sarete convocati per la discussione orale (a cui dovranno partecipare tutti i componenti del gruppo)



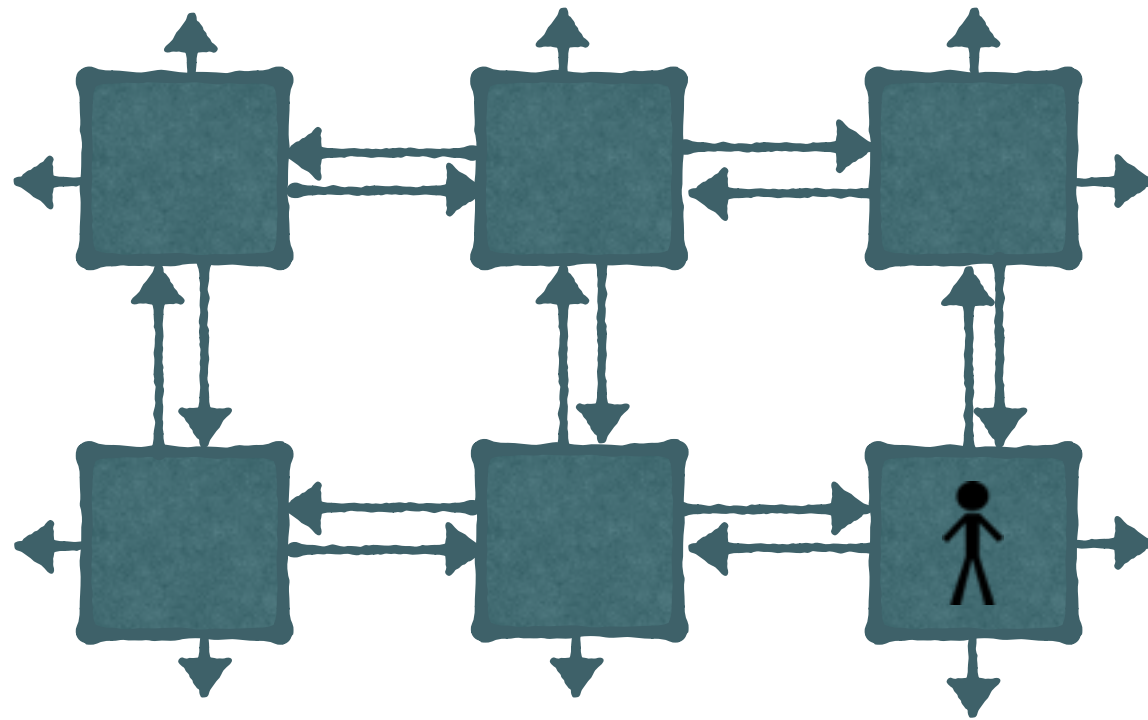
CONTENUTO DEL GIOCO

- Una mappa espandibile
- N pedine
- [...]

SVOLGIMENTO DEL GIOCO

- Le regole del gioco e lo scopo del gioco non fanno parte della specifica: li decidete voi
- Requisiti minimi e obbligatori:
 1. la mappa viene generata ed esplorata dinamicamente
 2. il gioco è per N giocatori
 3. i giocatori svolgono le loro azioni a turno e alla fine di ogni turno la mappa viene stampata a video con la posizione aggiornata dei giocatori
 4. la partita deve terminare (con l'eventuale vittoria di uno dei giocatori)

1-LA MAPPA




- La mappa è una struttura dinamica con una topologia a griglia
- All'inizio è composta di un unico nodo e quattro porte chiuse (NULL)
- L'apertura di una porta causa la creazione del nodo adiacente
- La creazione del nuovo nodo comporta l'aggiornamento delle porte degli eventuali nodi esistenti
- Non c'è un limite alla dimensione della mappa

2-I GIOCATORI

.....

- La mappa accetta un numero arbitrario di giocatori che verrà definito all'inizio della partita
- I giocatori agiscono a turno muovendosi sulla stessa mappa

 *la generazione di una stanza causata dal movimento di un giocatore influenza anche gli altri giocatori*

- azioni obbligatorie: movimento
- i giocatori vengono controllati dalla console. Le azioni vengono indicate al gioco attraverso opportuni comandi
- se le regole (e lo scopo) del gioco che avete stabilito lo richiedono, includete nuove azioni. Alcuni esempi:
 - raccogliere oggetti/punti,
 - azioni di attacco/difesa,
 - azioni di collaborazione con altri giocatori,
 - azioni di spesa, ...

3-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 viene **stampata** con la posizione aggiornata dei giocatori
- 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"

4-SCOPO DEL GIOCO

- Lo scopo del gioco non fa parte della specifica
- Stabilite un obiettivo da raggiungere per la vittoria
- Suggerimenti/Spunti per la scelta dell'obiettivo
 - raggiungere un determinato punteggio/guadagnare una certa somma di denaro/
raggiungere un certo livello di esperienza...
 - "There can be only one" : vince l'ultimo giocatore che non muore
 - trovare un certo oggetto da trovare sulla mappa
 -

PIÙ IN DETTAGLIO: IMPOSTAZIONE DEL PROGETTO

- impostazione del progetto su più file: .hpp e .cpp
- .hpp è l'estensione per l'interfaccia
- ad ogni classe corrispondono due file: es. Node.hpp e Node.cpp
- 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 e i suoi metodi, 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

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

int die = 0;

// set the seed
srand(time(0));

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

- Potete usare questo codice per generare numeri aleatori tra 1 e 6. Lo potete modificare a piacimento per generare numeri in un range più ampio.
- O potrebbe esservi utile adattarlo per altre configurazioni del gioco