Laboratory of Computer Science Education

Report progetto finale

Ivan Martini #207597
Giulia Peserico #224048

Giugno 2021

1 Introduzione

1.1 Idea

La proposta che vogliamo portare come attività didattica accattivante a dei ragazzi di terza media e primi anni delle superiori, è un laboratorio per la costruzione di un videogioco retrò, un **"roguelike"**.

L'idea nasce dal fatto che molto spesso la realizzazione personale di qualcosa di concreto, come può essere un gioco, riesce ad attrarre maggiormente l'interesse e la partecipazione degli studenti in ambito scolastico. Infatti, abbiamo preso spunto dal principio del *game based learning*. Per questo motivo, per introdurre i concetti di programmazione, abbiamo pensato a una sessione di più laboratori in cui gli studenti stessi costruiscono passo dopo passo il loro progetto in modo incrementale.

La nostra idea, infatti, è quella di non partire subito con obiettivi troppi grandi, dove i risultati arrivano dopo molto tempo e si rischia di perdere qualcuno per strada, ma quella di affrontare il progetto diviso in parti più piccole e di avere costantemente un raggiungimento delle aspettative. Inoltre, un altro aspetto che vogliamo sottolineare è che volutamente è stato deciso di tenere lo sviluppo più semplice possibile, in modo da permettere ai ragazzi/e di poter realizzare quasi tutto da soli (sono state implementate solo delle funzioni di supporto per non creare ulteriore carico cognitivo). Questo pensiamo che possa dare *maggior soddisfazione* ai ragazzi alla fine del progetto. In ultimo, volendo lasciar spazio alla creatività, non abbiamo influito negli obiettivi e nelle grafiche, permettendo ai ragazzi e ragazze di personalizzare tutto come meglio credono.

1.2 Motivazione e punti di forza

Siamo consapevoli che un'attività di questo tipo è molto particolare, ma abbiamo voluto continuare per questa strada scegliendo di rispettare alcune linee guida, non dettate dalla consegna, per essere soddisfatti del nostro prodotto.

Il primo punto è la scelta del linguaggio. Abbiamo impostato il laboratorio sul linguaggio `python`, in quanto riteniamo che possa essere quello più comodo per degli studenti novizi, anche se l'attività in sè **non è dipendente dal linguaggio**. L'unica libreria che viene utilizzata è `curses`, che è compatibile con la maggioranza dei linguaggi procedurali o di scripting (`C, C++, Java, Python, Pascal, Nodejs, Ruby`) ed è installata di default su tutti i sistemi operativi *unix like*. Se i ragazzi partecipanti all'attività dovessero già conoscere un linguaggio testuale di programmazione, allora l'intero laboratorio potrà essere tradotto (un bel lavoro, certo!) in modo da poter partire con una base nota e sviluppare il proprio lavoro.

Un altro punto di forza, che ricalca il precedente, è **l'indipendenza da IDE**. Non è necessario nulla oltre a un *editor di testo* e un *terminale* per ottenere i propri risultati, rendendo davvero accessibile a tutti l'attività. Lavorando in questo modo, si evitano artifici che gli IDE presentano e si aiutano i ragazzi a capire meglio il funzionamento del codice che hanno scritto.

Il terzo punto di forza è **la semplicità**, dal punto di vista della programmazione. Sicuramente questo laboratorio ha delle parti che possono risultare complesse, ma si tratta sempre di istruzioni elementari che si ripetono (assegnamenti, selezioni, cicli e operazioni sui vettori), le quali dovrebbero essere note ai ragazzi. Si evita così di fare un laboratorio che si basi solo sul **copiare e incollare** del codice senza comprenderne il significato, che è una situazione purtroppo ricorrente quando si ha poco tempo e si vuole proporre qualcosa di abbastanza complesso.

L'ultimo punto è la possibilità di personalizzazione che il progetto in sè offre. Utilizzando dei costrutti e una grafica molto semplici, viene lasciato molto spazio alla creatività e profondità che i ragazzi possono

inserire nel loro progetto. L'obiettivo finale è quello di portare i ragazzi a sviluppare le capacità per produrre da soli un gioco che contenga le loro idee, e non solo un gioco che riproduca ciò che gli è stato insegnato durante i laboratori. E' sicuramente un obiettivo molto coraggioso, ma crediamo che sia tutt'altro che irraggiungibile.

2 Descrizione dell'attività

2.1 Requisiti

- La base di un un qualsiasi paradigma procedurale (assegnamento, selezione, cicli e vettori). Abbiamo scelto python per la semplicità, ma non è dipendente da librerie o proprietà del linguaggio. Non è necessaria una conoscenza profonda dell'argomento.
- Tanta voglia di imparare e mettersi in gioco

Per dei neofiti si potrebbe pensare a un minicorso di un paio di giorni sulle basi di python, o del linguaggio scelto

2.2 Obiettivi

- Sviluppare un videogioco un po' particolare, fruibile in tutto e per tutto. La grafica sarà 2D è si utilizzerà un automa cellulare per generare degli ambienti irregolari (simili a caverne).
- Generare una maggior comprensione dei concetti basilari della programmazione, come funzioni e di variabili.
- Applicare gli stessi concetti in un progetto strutturato e complesso, mostrando anche la potenza che delle semplici istruzioni possono avere, se utilizzate nel modo corretto.

2.3 Tools utilizzati

- Linguaggio di programmazione Python: è stato scelto Python per semplicità, ma come detto non è strettamente necessario. La struttura del progetto e le librerie utilizzate non vincolano un linguaggio specifico
- Libreria per visualizzazione testuale su un terminale `curses`
- *(Opzionale)* Processing: è stato scelto Processing.py come linguaggio per applicazioni grafiche per semplicità. Abbiamo inserito un piccolo progetto per mostrare la sua utilità, ma non è strettamente legato all'attività in sè (per quanto questa possa venire riadattata e sviluppatata interamente con processing).

2.4 Struttura

L'organizzazione dello sviluppo del nostro laboratorio può essere suddiviso principalmente in due parti: Una più *introduttiva*, per prendere confidenza con il linguaggio e una più *produttiva*, dove si andrà a generare l'ambiente.

2.4.1 Introduzione iniziale

Per permettere ai ragazzi di seguire al meglio l'attività, pensiamo che possa essere utile un'introduzione agli argomenti trattati durante il lavoratorio, ad esempio un ripasso di Python e i costrutti base di programmazione. Questo permette di allineare le conoscenze dei partecipanti e di facilitare la comprensione dei passi successivi. Terminata questa parte, si può procedere ad implementare la prima mappa e i movimenti del personaggio (spostamento nella matrice + controllo sui muri).

2.4.2 Generazione della mappa

A questo punto il nostro eroe dovrebbe essere in grado di muoversi all'interno della mappa vuota che abbiamo creato, quindi è il momento di generare una mappa più interessante. L'obiettivo sarà quello di utilizzare un automa cellulare opportunamente modificato in modo da generare degli ambienti che sembrino delle caverne, sfruttando l'automa per chiudere i buchi o erodere i gruppi troppo piccoli. Un esempio pratico di questa applicazione è *Minecraft*, dove un automa cellulare 3-dimensionale viene utilizzato per generare proprio le caverne.

Il primo passo è quindi quello di codificare un automa cellulare tradizionale assieme ai ragazzi, seguendo le regole di John Conway per il gioco della vita. Da questo punto di partenza possiamo quindi cambiare le regole e partire da una matrice inizializzata casualmente (magari tramite un seme, in modo da rendere la funzione *procedurale* e quindi deterministica) in modo da costruire il nostro "generatore di caverne".

In particolare, una cellula con più di 5 vicini vivi diventa viva (per riempire i buchi), una cellula con meno di 4 vicini vivi diventa morta (per erodere i piccoli agglomerati) e una cella con 4 o 5 cellule vicine assume il valore precedente.

2.4.3 Ulteriore attività con Processing

Come già detto in precedenza, partendo da una base come quella illustrata sopra, è possibile pensare di generare grafiche più visuali aggiungendo solo i *tools* che si preferiscono.
Una nostra proposta è quella di utilizzare *Processing*, un linguaggio che permette di creare applicazioni grafiche in maniera molto semplice. Per questo scopo, è stato pensato un **laboratorio aggiuntivo** con l'utilizzo di questo strumento, nel quale verrà fatta una piccola introduzione all'argomento e verranno evidenziate le funzioni della libreria maggiormente utilizzate nell'attività, in modo da semplificare l'implementazione. I passi principali seguiranno prevalentemente quelli già fatti nei laboratori precedenti.

3 Setup e dettagli

Tutte le informazioni per preparare l'ambiente di sviluppo sono contenute nella repository Github. La repo contiene 2 branches: la branch **master** che contiene i *files iniziali da modificare* e il branch **solution**, che contiene i *files completi*, cioè quelli che i ragazzi dovrebbero raggiungere. I passi da eseguire sono gli stessi evidenziati nella presentazione, sempre contenuta nella repository (che andrà eventualmente riadattata a seconda delle esigenze).

4 Sviluppi futuri

Arrivati alla fine di questi laboratori, possiamo dire che ci si può aprire un mondo. Infatti, ci sono diversi e interessanti tipi di sviluppi futuri. Ad esempio, nel campo dell'**algoritmica**, le evoluzioni possibili potrebbero essere il posizionamento iniziale del giocatore, algoritmi per la connessione di aree della caverna irraggiungibili, introduzione di mostri tramite intelligenze artificiali (basilari) e pathfinding, ecc... Oppure, un'altra area interessante è quella della **logica di gioco**, la quale dà completamente spazio alla creatività. L'aggiunta di un inventario/equipaggiamento, la divisione del gioco in più livelli e la loro parametrizzazione della difficoltà, la generazione di tesori/oggetti casuali, sono tutte idee fattibili per dare un'evoluzione al gioco rendendolo sempre più accattivante. Inoltre si può lavorare sullo **storytelling** e sulle abilità narrative dei ragazzi, per costruire la storia che ruota intorno al protagonista.