BRICKS | TEMA

Fingere o simulare, questo è il dilemma

a cura di:
Morena De Poli

Simulazione, Eventi, Numeri Casuali, Matematica, Statistica, Programmazione

Nella seconda metà dell'anno scolastico solitamente propongo alle mie classi prime dell'ITTS attività introduttive alla programmazione: dall'analisi di semplici problemi si cerca una soluzione algoritmica, cioè quella *sequenza finita e ordinata di passi elementari per ottenere i risultati voluti a partire dai dati noti del problema*.

I diagrammi a blocchi, detti anche *flow-chart*, descrivono l'algoritmo individuato e la codifica in un linguaggio di programmazione è un passaggio successivo.

In genere scelgo problemi di tipo geometrico (calcolo di aree e perimetri di poligoni) o semplici calcoli numerici (sconti, percentuali, medie); ma quest'anno ho individuato il tema della simulazione di eventi come filo conduttore: l'attività è più vicina ai ragazzi essendo alla base di ogni gioco, fornisce loro strumenti per meglio comprendere le applicazioni che quotidianamente utilizzano per svago; è inoltre l'occasione per riflettere sul mondo virtuale.

Uno dei primi a studiare le tecniche di generazione di numeri *casuali* da applicare via software è Donald Knuth e uno dei suoi testi di riferimento è *The Art of The Computer Programming*. Knuth propone metodi deterministici che, con opportune scelte dei parametri, generano numeri uniformemente distribuiti, cioè che hanno tutti la stessa probabilità di presentarsi. In questo ambito non è insolito l'intreccio tra la Matematica e la Statistica.

Ovviamente, ai ragazzi di classe prima tutto questo può solo essere accennato, date le loro conoscenze matematiche e statistiche, ma, a differenza dei concetti teorici, le tecniche di simulazione da applicare sono decisamente semplici e alla loro portata.

Fingere o Simulare?

Nella vita quotidiana usiamo i verbi "fingere" e "simulare" come sinonimi, ma in ambito statistico simulare non ha nulla a che fare con la finzione; significa invece **ricostruire la realtà** in relazione all'oggetto di simulazione.

Se per fingere un mal di pancia basta improvvisarsi attori, per ricostruire un evento è indispensabile conoscerlo a fondo, così da poter rappresentare tutti i suoi modi di essere.

Quando si gioca a calcio con un'applicazione per smartphone non si sta fingendo, e se l'applicazione è costruita con buone tecniche di simulazione, sembrerà di giocare veramente. Anche la preparazione dei piloti di aerei o degli astronauti avviene mediante simulatori e migliore è la simulazione, più veritiera è l'esperienza vissuta.

Il problema sorge se ci si immerge nel mondo virtuale tanto da non saper più distinguerlo da quello reale o da rimanere delusi quando l'esperienza viene vissuta nella realtà: potrebbe essere lo spunto per un progetto interdisciplinare di Educazione alla Cittadinanza Digitale.

Tecniche di simulazione

Veniamo alle tecniche di simulazione partendo dall'evento più semplice: il lancio di una moneta.

La conoscenza dell'oggetto è indispensabile: è una fetta di cilindro, presenta due facce dello stesso peso denominate spesso "testa" e "croce". Se la moneta è truccata una delle facce ha un peso maggiore e nel lancio tale faccia tende a portarsi verso il basso mostrando l'altra; in questo caso è necessario conoscere il rapporto tra i pesi delle due facce.

Tutti i linguaggi di programmazione e gli applicativi di calcolo come i fogli elettronici mettono a disposizione una funzione, di solito detta *random*, che invocata genera numeri *casuali* appartenenti all'intervallo [0,1[.

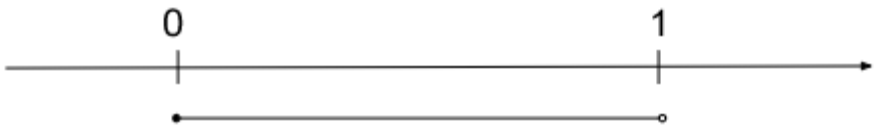

Figura 1 - Intervallo [0,1[, a cui appartiene un numero *random*, sulla retta dei numeri reali

La tecnica più semplice per simulare l'esito del lancio di una moneta prevede di suddividere l'intervallo [0,1[in due parti: perfettamente uguali nel caso della moneta regolare, proporzionate al peso delle due facce nel caso della moneta truccata. Se il numero *random* appartiene alla prima parte dell'intervallo gli viene associato "testa", altrimenti "croce" (o le facce in proporzione al loro peso); nel caso della moneta regolare possiamo modellizzare così:

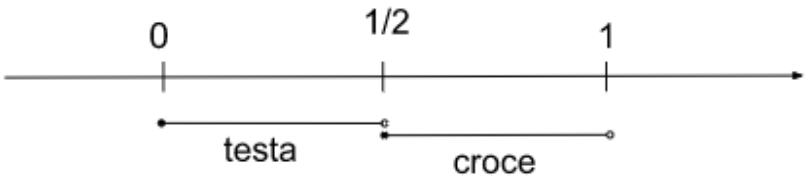

Figura 2 - Simulazione dell'esito del lancio di una moneta regolare

La descrizione con *flow-chart* di questo primo algoritmo di simulazione è:

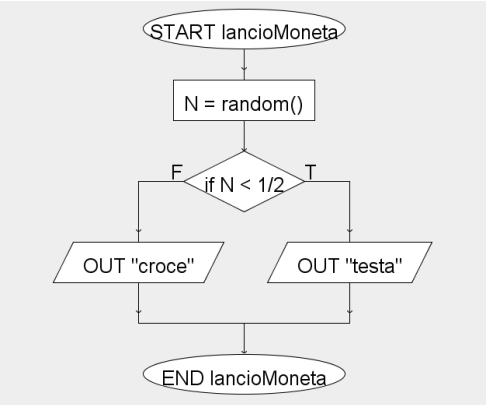

Figura 3 - *Flow-chart* per simulare l'esito del lancio di una moneta regolare

Se volessimo simulare altri eventi come il lancio di un dado o l'estrazione di un numero del Lotto, l'applicazione di tale tecnica imporrebbe di suddividere l'intervallo [0,1[in tante parti quanti sono i possibili esiti: 6 per il dado, 90 per l'estrazione del Lotto.
Osservando il *flow-chart* di Fig. 3 si deduce facilmente che un costrutto di selezione *if* permette di discriminare tra 2 possibilità, perfetto per il caso della moneta; ma per il dado servirebbero 5 *if* per individuare i 6 possibili casi, e ben 89 *if* per il Lotto. Decisamente troppi!

Meglio quindi apportare qualche modifica alla tecnica, ad esempio attuando la logica del piccolo Gauss. Ricordate l'aneddoto? Si racconta che il maestro avesse assegnato il calcolo della somma dei primi 100 numeri naturali; a differenza dei compagni che sommarono partendo da 1 e aggiungendo via via i numeri successivi, Gauss si impegnò a trovare una formula :

Gauss	Compagni
$S(n) = \frac{n(n+1)}{2}$	$S(1) = 1$ $S(2) = 1 + 2 = 3$ $S(3) = 3 + 3 = 6$ ... $S(n) = S(n-1) + n$

Immagino che Gauss avesse qualcosa in più: la capacità di "vedere" i numeri e di giocare con le loro proprietà. Per noi comuni mortali le proprietà commutativa e associativa della somma spesso sono solo degli enunciati da imparare a memoria, mentre Gauss a 9 anni aveva la consapevolezza di poter scambiare l'ordine dei numeri avvicinando 1 e 100, 2 e 99, 3 e 98, e così fino a 50 e 5, per poi sommarli a due a due accorgendosi che si ottiene 101 per 50 volte. La capacità di astrazione ha fatto il resto.

Proviamo allora a modificare la nostra tecnica di simulazione … con l'approccio di Gauss.

Anziché dividere l'intervallo [0,1[in *n* parti, tante quanti i possibili esiti dell'evento da simulare, estrarre un numero *random* e vedere a quale parte appartiene, si può agire algebricamente per trasformare quel numero *random* in un numero appartenente all'insieme dei possibili esiti E = {1 ,2 ,3 , …,*n*}.

Data la disuguaglianza

$$0 \leq random < 1$$

applicando il Secondo Principio di Equivalenza è possibile moltiplicare i tre membri per il numero *n*, che ovviamente è diverso da 0, ottenendo una disuguaglianza equivalente alla precedente

$$0 \leq n \cdot random < n .$$

Applicando invece il Primo Principio di Equivalenza si somma 1 ai tre membri

$$1 \leq n \cdot random + 1 < n + 1$$

Ancora non basta poiché queste trasformazioni individuano l'intervallo [1,*n*+1[, contenente quindi infiniti numeri reali, mentre il nostro obiettivo è quello di ottenere l'insieme E dei naturali da 1 a *n*.

Applicando all'ultima trasformazione la funzione che considera la parte intera di un numero (si indica con [ ]), si raggiunge lo scopo voluto e per ogni *random* generato si ottiene un numero nell'insieme E

$$1 \leq [n \cdot random + 1] \leq n$$

Nell'estrazione di un numero del Lotto *n* vale 90 e il *flow-chart* che la simula è il seguente:

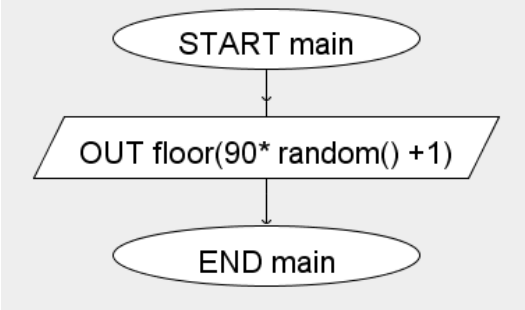

Figura 4 - *Flow-chart* che genera l'esito di una estrazione del Lotto

Decisamente più semplice rispetto a quello con 89 *if* annidati!

Va precisato che l'applicativo per disegnare *flow-chart* (*AlgoBuild*[1]) propone la funzione *floor()* per ottenere la parte intera di un numero; altri software dispongono anche della funzione *int()*; importante quindi consultare il manuale d'uso per individuare la corretta funzione.

Applicare le tecniche di simulazione

Come tradurre algoritmi di simulazione in codice di programmazione ? Quale linguaggio o quale ambiente di programmazione usare?

Di seguito propongo alcune idee di simulazioni, in qualche caso elementari, che costituiscono però la base per approfondimenti successivi.

[1] https://algobuild.com/it/index.html

La scheda micro:bit[2]

È una scheda di prototipazione che mette a disposizione un ambiente per la programmazione visuale a blocchi e uno per quella testuale in *javascript* e in *Python*.

È disponibile un simulatore online della scheda e la si può programmare pur non disponendo fisicamente di essa … a proposito di simulazione ;-).

Possiede molti sensori tra cui l'accelerometro che consente di individuare ogni movimento a cui viene sottoposta. Questo sensore è utile proprio per simulare il lancio della moneta: l'evento "scuotimento" della scheda provoca l'esecuzione delle azioni che attuano l'algoritmo di simulazione.

Va precisato che questo ambiente mette a disposizione la funzione che restituisce un casuale già trasformato con la tecnica mostrata, è quindi sufficiente indicare i due estremi dell'insieme degli esiti da generare:

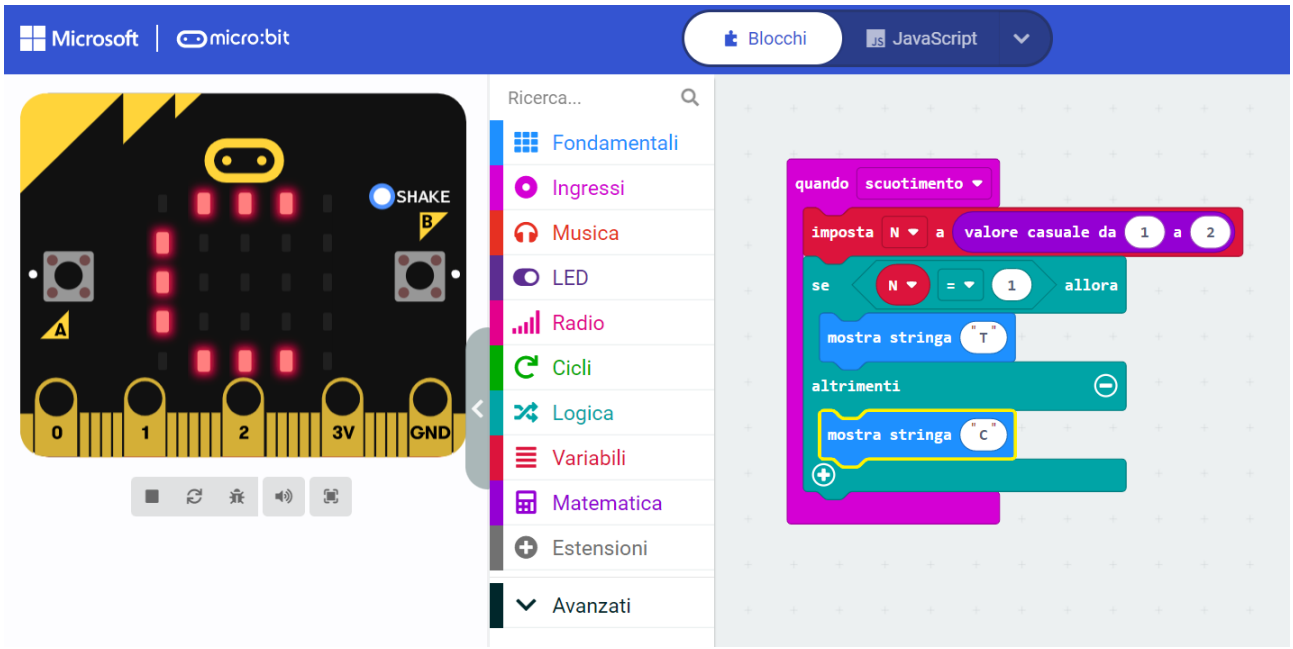

Figura 5 - Simulazione del lancio di una moneta con micro:bit

Sul simulatore il *click* del mouse sul bottone "*shake*" manda in esecuzione il codice contenuto nel blocco fucsia; trasferito il codice nella scheda fisica, lo scuotimento della stessa provoca l'esecuzione di tale codice e di conseguenza la visualizzazione di "C" o "T" sui led: sembrerà di avere a disposizione una vera moneta.

[2] https://microbit.org/

Snap![3]

È un ambiente per la programmazione visuale a blocchi realizzato dall'Università di Berkeley, simile a Scratch sviluppato dal MIT di Boston, ma con molte funzionalità che consentono anche la programmazione ad alto livello.

Nello *stage* è possibile far agire uno *sprite* mediante lo *script* (istruzioni espresse in forma di blocchi) ad esso assegnato, il quale consente anche la gestione dei costumi con cui si può presentare lo *sprite*.

Anche in questo ambiente la funzione *random* è trasformata e può essere usata nella simulazione del lancio del dado in modo che restituisca un numero casuale tra 1 e 6; in questo modo ci si può concentrare maggiormente sugli aspetti creativi della realizzazione dei costumi dello *sprite* più che sugli aspetti della programmazione dello *script* che, in questo caso, risulta davvero banale:

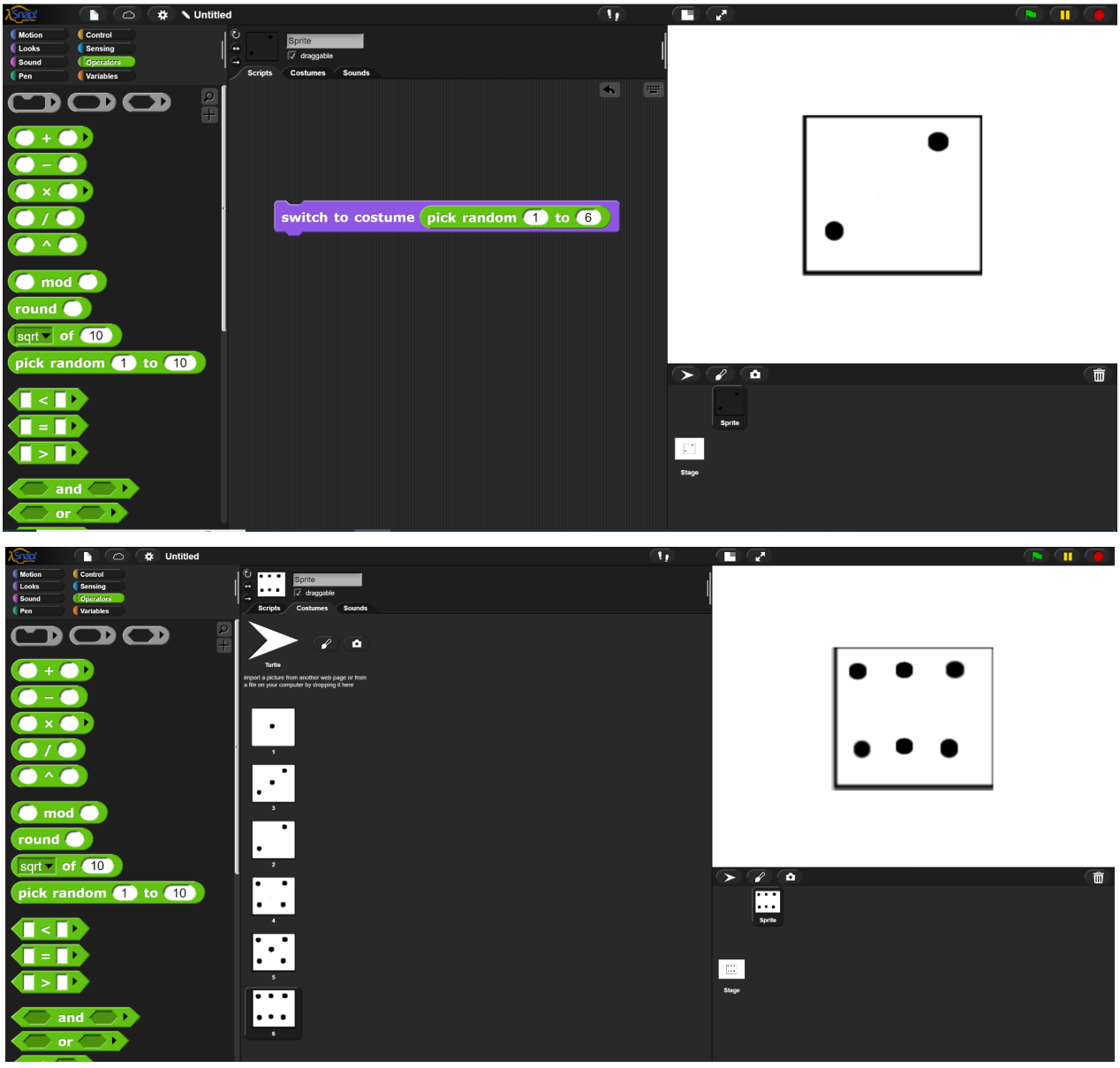

Figura 6 - Simulare il lancio di una dado con Snap!: lo *script* e i costumi

[3] https://snap.berkeley.edu/

È sufficiente effettuare un *click* col mouse sul blocco viola per "lanciare" il dado.

Ma si può fare di più! I blocchi sono suddivisi in categorie, quelli viola consentono di programmare l'aspetto degli *sprite* e anche dello *stage*, quelli verdi mettono a disposizione operazioni e funzioni, quelli gialli della categoria "Controllo" consentono di realizzare cicli, ad esempio per effettuare più volte il lancio del dado così da conteggiare gli esiti delle facce, memorizzati mediante l'uso della struttura lista manipolabile con i blocchi arancioni della categoria "Variabili".

Di seguito lo *script* per i conteggi degli esiti di 100 lanci di un dado:

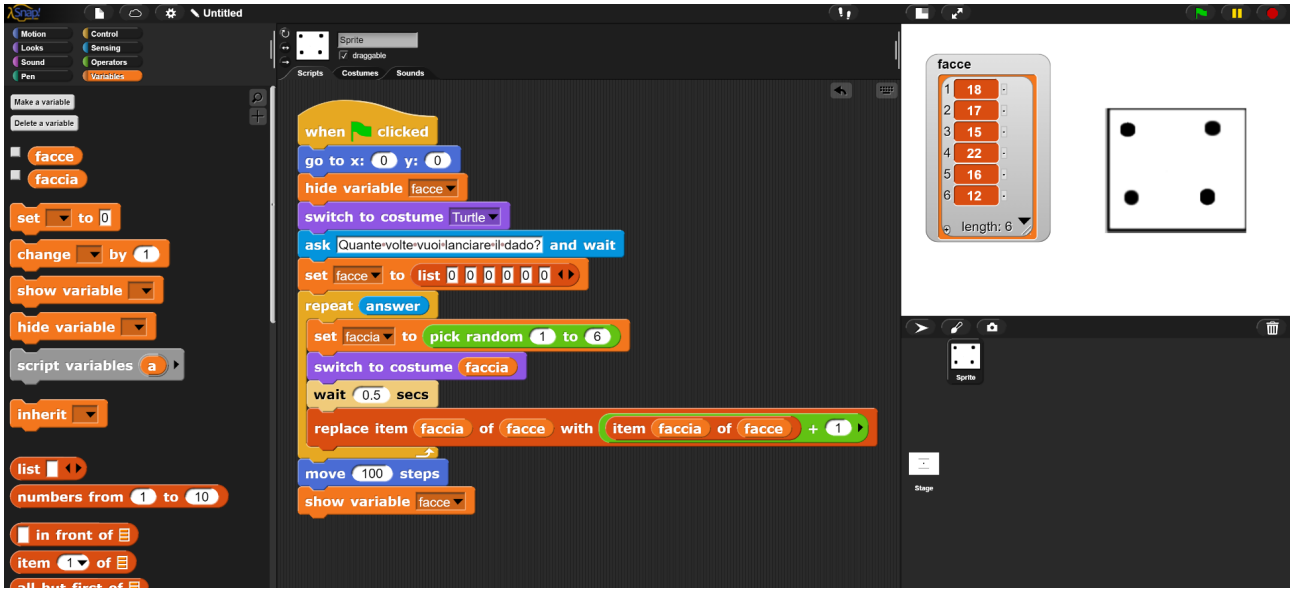

Figura 6 - Simulazione del lancio di 100 dadi con Snap! e conteggio dell'uscita di ogni faccia

Python[4]

Python è uno dei linguaggi di programmazione testuali più usati al mondo. Trova impiego nei campi più disparati, dal web al calcolo numerico, e ha la caratteristica di essere facilissimo da imparare. Quando lo presento ai miei studenti, in meno di un'ora di lezione riescono già a scrivere semplici *script* utilizzando la *shell* di *Python* e il suo ambiente di *edit*, cioè lavorando con la IDLE (*Integrated Development Language Environment*).

Python mette a disposizione la funzione *random()* nella libreria *random* che genera numeri in [0,1[.

Con le strutture dati messe a disposizione da *Python* si possono realizzare in modo semplice simulazioni anche più elaborate delle precedenti, come ad esempio l'estrazione di una carta da un mazzo da Poker.

Tralasciando i Jolly, le carte sono suddivise in 4 semi: Cuori, Quadri, Fiori, Picche; per ogni seme ci sono 13 carte: A, 2, 3,..., 9, 10, J, Q, K.

[4] https://www.python.org/

Si genera quindi un numero *random*, lo si trasforma tra 1 e 4 e si associa a ogni numero un seme; poi si genera un nuovo *random*, lo si trasforma tra 1 e 13 e gli si associa la corretta numerazione della carta.

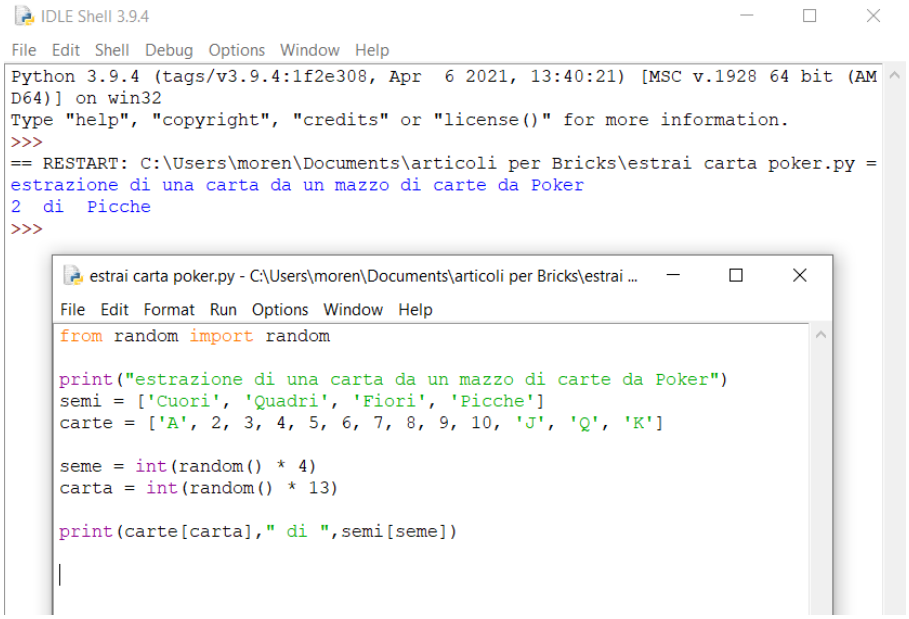

Figura 7 - Simulazione dell'estrazione di una carta da Poker con *Pyhton*

La finestra più piccola mostra l'ambiente di *edit* dove viene scritto lo *script* e da qui viene avviata l'esecuzione con il comando *Run* del menu; la finestra più grande è invece la *shell* dove viene mostrato il risultato dell'esecuzione.

Importata la libreria che contiene la funzione *random()*, vengono create due liste contenenti i nomi dei semi e delle carte, vengono poi generati i due numeri *random()* che, trasformati, permettono di individuare seme e valore della carta estratta.

Importante specificare che ogni ripetizione delle tre istruzioni per generare e mostrare la carta a video, simula l'estrazione da un nuovo mazzo integro, contenente tutte le 52 carte, esattamente come nei lanci ripetuti della moneta e del dado dove l'oggetto viene riutilizzato nella sua interezza.

Sono altre le tecniche da applicare qualora si vogliano distribuire le carte per simulare una partita.

CALC di LibreOffice[5]

L'ultima proposta prevede l'uso del foglio di calcolo CALC di LibreOffice, un software di *Office Automation open source*.

Ho proposto ai miei studenti di simulare il calcio di rigore in una partita: il giocatore di una squadra calcia dal dischetto, il portiere dell'altra squadra prova a parare.

[5] https://it.libreoffice.org/

Conviene iniziare simulando un giocatore poco esperto nei rigori, così che per il portiere sia imprevedibile la zona della porta in cui possa arrivare il pallone.

Si costruisce un modello suddividendo la porta in 9 zone; dal punto di vista del giocatore che calcia le zone sono:

Figura 8 - Modello per rappresentare la porta suddivisa in zone

Dal punto di vista del portiere si possono definire i movimenti: Destra Basso, Destra Alto, Fermo, Sinistra Basso, Sinistra Alto.

La parata avviene quando il portiere si muove nella stessa zona in cui viene calciato il pallone secondo la seguente tabella costruita con CALC:

	A	B	C	D	E	F
1	Movimento / Zona	Dx Basso	Dx Alto	Fermo	Sx Basso	Sx Alto
2	Alto Sx		P			
3	Alto Centro			P		
4	Alto Dx					P
5	Basso Sx	P				
6	Basso Centro			P		
7	Basso Dx				P	
8	Fuori Sx	Pallone esce				
9	Fuori Dx					
10	Fuori Alto					
11						

Figura 9 - Tabella per la parata del calcio di rigore

Si deve poi procedere generando due numeri casuali con la funzione *casuale()*, uno va trasformato tra 1 e 9 per individuare la zona in cui il giocatore manda il pallone, l'altro tra 1 e 5 per individuare il movimento del portiere. Incrociando i due valori secondo la tabella si individua l'esito del rigore.

		Movimento / Zona	1 Dx Basso	2 Dx Alto	3 Fermo	4 Sx Basso	5 Sx Alto
1	Alto Sx		P				
2	Alto Centro			P			
3	Alto Dx						P
4	Basso Sx	P					
5	Basso Centro			P			
6	Basso Dx				P		
7	Fuori Sx	Pallone esce					
8	Fuori Dx						
9	Fuori Alto						

Zona	Movimento
8	1
Fuori Sx	Dx Basso

Esito	Pallone esce

Figura 10 - Simulazione di un calcio di rigore

Le formule realizzate per la simulazione sono le seguenti:

D13 =INT(CASUALE()*9+1)

E13 =INT(CASUALE()*5+1)

D14=CERCA.VERT(D13;A3:B9;2)

cerca nell'intestazione delle righe la zona in cui è stato calciato il pallone che corrisponde al casuale generato in D13

E14=CERCA.ORIZZ(E13;C1:G2;2)

cerca nell'intestazione delle colonne la zona in cui si muove il portiere che corrisponde al casuale generato in E13

E17 =SE(D13 >= 7; C9 ; SE(INDIRETTO("R"&(D13+2)&"C"&(E13+2);0) = ""; "Gol" ; "Parato"))

se il pallone è stato calciato fuori, viene mostrata la frase presente in C9, altrimenti si cerca nella tabella la cella che incrocia le due azioni

Nella funzione INDIRETTO() il secondo parametro impostato a 0 consente di riferire le celle con il sistema R1C1, così da comporre l'indirizzo della cella cercata in forma di stringa: "R" concatenata (&) con il numero di riga, concatenata con "C", concatenata con il numero di colonna; ai numeri di riga e colonna deve essere sommato il valore 2 per via delle doppie intestazioni della tabella.
Premendo il tasto F9 si realizza una nuova simulazione.

Tempi, metodi e conclusioni

Come già accennato, svolgo queste attività nella seconda parte dell'anno scolastico, quando gli alunni hanno già una certa dimestichezza nell'utilizzo di vari software.

Dopo una prima lezione frontale introduttiva sul concetto di simulazione propongo di simulare i tre eventi più significativi, moneta-dado-Lotto, al fine che emerga dagli studenti stessi la necessità di trovare una tecnica per trasformare i numeri *casuali*, come mostrato in questo articolo.

Già alla terza lezione gli alunni sono pronti per sperimentare in modo autonomo, lavorando in piccoli gruppi di massimo 3 persone. Suggerisco possibili eventi da simulare, oltre a quelli proposti in questo articolo, e lascio loro la scelta sia dell'evento che dell'applicativo da utilizzare per lo sviluppo. Mediamente, altre 4 ore di attività laboratoriale sono sufficienti per ottenere i primi risultati.

Poi spesso accade che, mano a mano che i ragazzi si impadroniscono delle tecniche di simulazione, essi scoprano nuove possibilità di sviluppo e chiedano quindi più tempo per ulteriori approfondimenti. Ad esempio, chi sceglie di simulare il tiro con l'arco si accorge che assomiglia molto al gioco delle freccette, dove il bersaglio però è più complesso; oppure qualcuno si rende conto come introducendo semplici modifiche, la tecnica applicata possa simulare giocatori di diversi livelli di capacità.

Concludendo, quando propongo queste attività alle mie classi, i ragazzi si appassionano al punto che è dato sapere quando e come si comincia, ma difficilmente si può prevedere quando e come si possa finire.

Morena De Poli

morena.depoli@gmail.com

Diplomata Perito industriale in Informatica nel 1981, ho iniziato l'attività di docenza presso l'Istituto Tecnico-Tecnologico nel 1983, prima nelle discipline Laboratorio di Matematica e Laboratorio di Statistica, Calcolo delle Probabilità e Ricerca Operativa, in seguito, dopo la riforma dei Tecnici, nella disciplina Laboratorio di Tecnologie Informatiche.
La pervasività delle discipline insegnate, mi ha indirizzata verso la ricerca didattica per approfondire la valenza delle attività interdisciplinari, organizzando e coordinando progetti di ampio respiro, spesso a tema ambientale.
Ho raccontato l'esperienza nell'insegnamento di applicazioni informatiche a discipline matematiche e statistiche nel testo "Dalla Matematica al Coding. Le sfide di Marco e Sofia: pratiche di allenamento all'uso del pensiero algoritmico" - Edizioni Anicia - Roma e ho collaborato al testo "Dati, Cittadinanza e Coding" - S. Penge - Edizioni Anicia - Roma.
Ho pubblicato articoli descrittivi delle attività didattiche svolte con le mie classi in varie riviste online.