Clickomania con Blockly

Violetta Lonati*

Sommario

Clickomania è un solitario, noto anche come *Chain Shot!* o *Same Game*. Il campo di gioco è costituito da una *parete* inizialmente coperta di *mattoni*, uno per *casella*, di colori diversi, e l'obiettivo del gioco è rimuovere dalla parete tutti i mattoni, selezionando via via aree di mattoni adiacenti dello stesso colore, come spiegato meglio piú avanti. Argomento del laboratorio è la realizzazione di un programma per giocare a (una versione semplificata di) Clickomania o, piú precisamente, un programma che aggiorni la parete di gioco in seguito a ogni mossa (click) del giocatore, utilizzando l'*ambiente di programmazione Blockly*.

1 Regole del gioco

Clickomania è un solitario, noto anche come *Chain Shot!* o *Same Game*. Il campo di gioco è costituito da una *parete* 9x9 inizialmente coperta di *mattoni*, uno per *casella*, di colori diversi, e l'obiettivo del gioco è rimuovere dalla parete tutti i mattoni, selezionando via via aree di mattoni adiacenti dello stesso colore.

Ad ogni mossa, il giocatore clicca su un mattone e può partire un'*infezione* che contagia i mattoni vicini. L'*area di contagio* è definita come segue: se il mattone selezionato non ha nessun vicino (sopra, sotto, a destra, a sinistra) dello stesso colore, l'area di contagio è vuota; se invece il mattone selezionato ha uno o piú vicini dello stesso colore, allora l'area di contagio comprende il mattone cliccato, tutti i suoi vicini dello stesso colore, i vicini dei vicini dello stesso colore, e così via, cioè tutti i mattoni *collegati* a quello selezionato e dello stesso colore.

Una semplificazione del gioco consiste nel considerare invece di tutti i mattoni vicini, solo i mattoni sopra o sotto la casella cliccata; si definisce *striscia di contagio* la striscia verticale formata dal mattone cliccato, dai suoi vicini dello stesso colore, i vicini dei vicini dello stesso colore, e così via; in questa semplificazione, l'infezione contagia dunque i mattoni della striscia di contagio del mattone cliccato, invece di quelli dell'area di contagio.

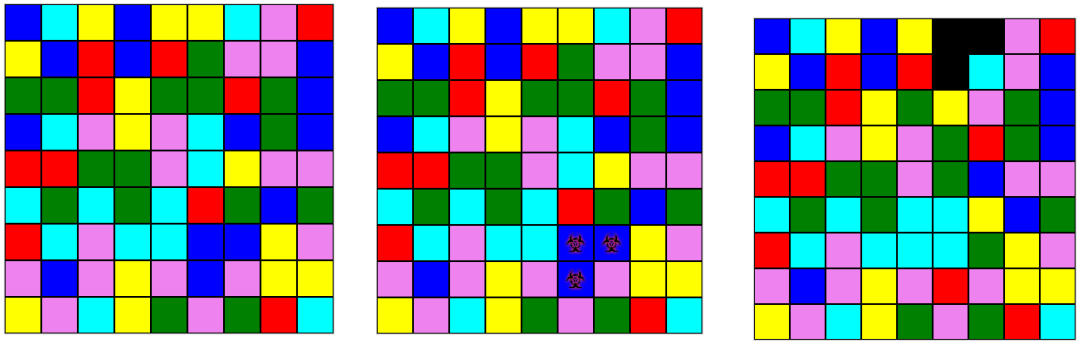
Tutti i mattoni infetti vengono quindi *rimossi* e il campo di gioco si *ricompatta*: in ogni colonna, le caselle lasciate libere dai mattoni rimossi vengono riempite dai mattoni che si trovano sopra, i quali *cadono* verso il basso. Il gioco finisce quando ogni mattone rimasto non ha mattoni vicini dello stesso colore.

Esempio

Le caselle sono numerate dall'alto verso il basso e da sinistra verso destra. Ogni casella è individuata da una coppia di indici (r, c), dove r è l'indice di riga e c è l'indice di colonna. La casella in alto a sinistra ha coordinate $(0, 0)$.

All'inizio di una partita la parete potrebbe apparire come nella seguente figura (a sinistra).

*Il gioco qui descritto è stato proposto per la prima volta in occasione delle finali della sesta edizione del Kangourou dell'Informatica (maggio 2014): il software di gioco è stato sviluppato con la collaborazione di Luca Prigioniero e Pierlauro Sciarelli.



Se viene selezionato il mattone fucsia nella casella di coordinate (7,6), allora non viene rimosso alcun mattone, poiché i mattoni ad esso adiacenti hanno tutti colore diverso da fucsia.

Cliccando invece sul mattone nella casella di coordinate (6,5), verranno infettati tutti i mattoni blu vicini (vedi figura al centro) e il campo da gioco si ricompatterà come nella figura a destra.

2 Blockly

Blockly consente di *costruire dei programmi* a partire da blocchi colorati predefiniti, che si possono incastrare come in un puzzle.

Non c'è un solo programma che permette di giocare a Clickomania: i blocchi a disposizione possono essere combinati a formare tanti diversi programmi, tutti funzionanti. Tuttavia ciascun blocco ha un *costo* e il vostro punteggio sarà tanto maggiore quanto minore sarà la somma dei costi dei blocchi usati.

Una volta costruito un programma, si può iniziare il gioco cliccando sul bottone "Gioca". A questo punto, ad ogni click del giocatore (voi stessi, mentre testate il vostro programma!!), la parete verrà aggiornata in base alle istruzioni del programma costruito.

Blockly consente inoltre di vedere gli effetti dell'esecuzione del programma costruito, attraverso un'animazione, direttamente sulla parete di gioco, e permette di salvare fino a 4 programmi costruiti.

Interfaccia

Lanciate il programma *Clickomania*: vedrete una pagina composta di tre colonne.

- A sinistra trovate vari elementi.
  - La parete di mattoni.
  - Un cursore con ai lati una tartaruga e una lepre, per aumentare/diminuire la velocità dell'animazione.
  - Il bottone "Gioca" che consente di iniziare una partita e che si trasforma in "Reset" durante l'esecuzione del programma; cliccando su "Reset" è possibile iniziare una nuova partita.
  - Le icone relative a salvataggio/visualizzazione/caricamento dei programmi (o versioni di uno stesso programma): cliccando sul primo simbolo si può salvare il programma attualmente in costruzione, soffermandosi col mouse sul secondo simbolo si può visualizzare il programma salvato, cliccando sul terzo simbolo si può caricare il programma salvato in precedenza.
- Al centro trovate un menu di categorie tra cui scegliere i blocchi colorati; soffermandosi col mouse su un blocco si può leggere una descrizione di cosa fa il blocco.

- A destra trovate uno spazio bianco in cui potete trascinare i blocchi scelti dal menu, per costruire le istruzioni del vostro programma. Nell'angolo in basso a destra c'è il simbolo di un cestino nel quale possono essere trascinati i blocchi che non servono più.

  Durante l'aggiornamento della parete in seguito ad una mossa (click), i blocchi in esecuzione saranno via via evidenziati. Per fare degli esperimenti, tra una mossa e l'altra potete modificare il programma; però come soluzione del quesito accetteremo soltanto un programma che non debba essere modificato a seconda della mossa!

Blocchi a disposizione

I blocchi disponibili sono raggruppati nelle seguenti categorie (alcuni blocchi sono presenti in piú di una categoria): Aggiornamento parete, Caselle, Colori, Infezione, Logica, Cicli, Matematica, Variabili.

I blocchi delle categorie Logica, Cicli, Matematica, Variabili hanno costo 0; i costi degli altri blocchi sono riassunti nelle seguenti tabelle.

Aggiornamento parete

Rimuovi i mattoni nell'area di contagio attorno a... e fai cadere tutti i blocchi	40
Rimuovi i mattoni nella striscia di contagio attorno a... e fai cadere tutti i blocchi	40
Fai cadere tutti i blocchi	10
Rimuovi tutti i mattoni infetti	5
Rimuovi il mattone in...	1
Fai scendere il mattone in...	1
Fai scendere la pila di mattoni con base in...	2
Fai cadere il mattone in...	2
Fai cadere la pila di mattoni con base in...	4
Compatta la colonna di indice...	7
Colora di rosso il mattone in... (e varianti)	1

Infezione

Infetta tutti i mattoni nell'area di contagio attorno a...	20
Infetta tutti i mattoni nella striscia di contagio attorno a...	20
Infetta il mattone in...	1
Togli infezione dal mattone in...	1
Togli infezione da tutti i mattoni	5
casella piú in alto con mattone infetto (e varianti)	2
casella qualsiasi con mattone infetto	2
cè un mattone infetto	2
il mattone in... è infetto	0
Rimuovi tutti i mattoni infetti	5
Per ogni casella... contenente un mattone infetto fai...	0

Caselle

casella cliccata	0
casella sopra a...	1
casella in riga =... e colonna =...	0
indice di riga di...	0
indice di colonna di...	0
casella vuota piú in alto (e varianti)	2
una casella vuota qualsiasi	2
cè una casella vuota	2
... è vuota	0

Colori

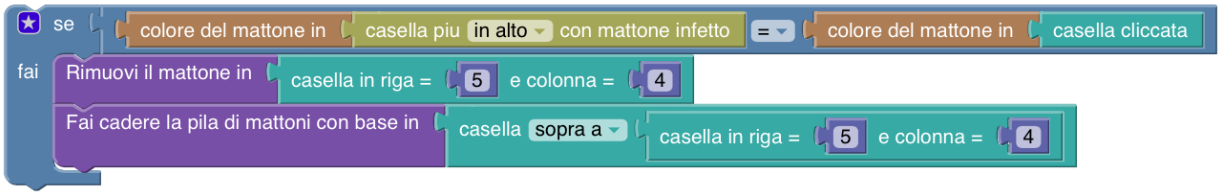
colore del mattone in...	0
casella piú... con mattone di colore...	2
una casella qualsiasi con mattone di colore...	2
il mattone in... è rosso (e varianti)	0
il mattone in... è collegato al mattone in...	10
Colora di rosso il mattone in... (e varianti)	1

Esempi

Il programma più semplice che gioca a Clickomania è il seguente. Ad ogni click, vengono rimossi tutti i mattoni che fanno parte dell'area di contagio attorno alla casella cliccata, e gli spazi vuoti che si vengono a creare vengono eliminati dalla caduta dei mattoni che si trovano sopra.

Vediamo ora qualche esempio un po' più complicato (non è detto che siano di qualche utilità!!):

In questo caso, se la casella cliccata si trova nella riga di indice 5, allora viene infettato il mattone nella casella di coordinate $(1, 4)$ e viene colorato di rosso il mattone nella casella cliccata.



In questo caso, viene considerata la casella più in alto tra quelle contenenti un mattone infetto, assieme alla casella cliccata; se le due caselle contengono un mattone dello stesso colore, allora verrà rimosso il mattone che si trova nella casella di coordinate $(5, 4)$ e cadrà l'intera pila di mattoni posizionata a partire dalla casella di coordinate $(4, 4)$.

In questo caso, se ci sono più di tre mattoni infetti, ne verranno rimossi tre a caso; se ce ne sono tre o meno, verranno rimossi tutti.

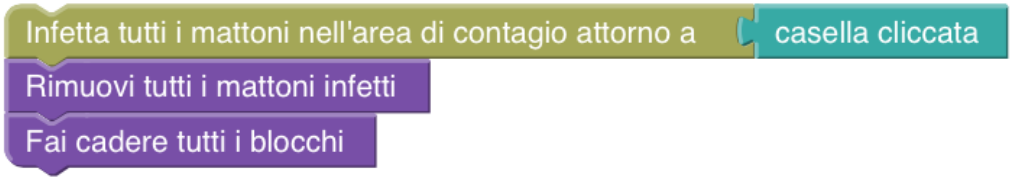
3 Possibili soluzioni (per la versione non semplificata del gioco)

Il programma piú semplice per giocare a Clickomania, di costo 40, è quello proposto nelle istruzioni:

Altre soluzioni meno *costose* si possono costruire articolando il programma in tre fasi:

1. infezione dei mattoni interni all'area di contagio;
2. rimozione dei mattoni infetti;
3. eliminazione delle caselle vuote (create nel passaggio precedente) tramite caduta dei mattoni opportuni.

Per ciascuna di queste tre fasi è disponibile un blocco, quindi una possibile soluzione è fornita dal programma:



Il costo di questa soluzione è pari a $20 + 5 + 10 = 35$.

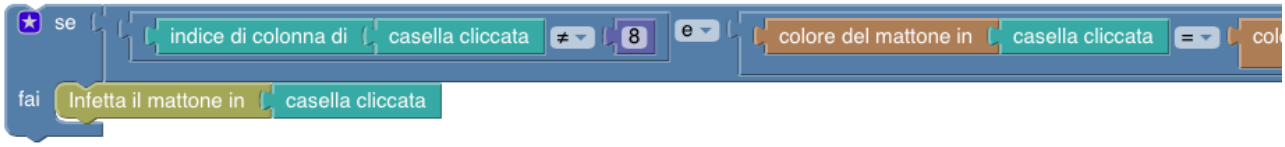
Ciascuna delle tre fasi può essere articolata ulteriormente usando blocchi meno costosi.

3.1 Infezione dell'area di contagio

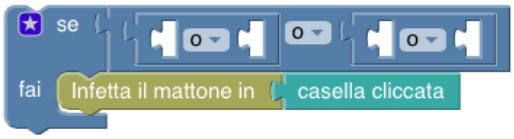
La fase piú complessa è quella che riguarda l'infezione dell'area di contagio. Innanzitutto bisogna stabilire se l'area è vuota oppure no, e per farlo bisogna controllare tutte le caselle vicine a quella cliccata, con un'istruzione del tipo:

Bisogna però fare attenzione ai bordi del campo di gioco: ad esempio se la colonna cliccata si trova nell'ultima colonna, non ha senso chiedere il colore della casella alla sua destra! Un modo per non correre rischi è usare due blocchi "se... fai" annidati:

oppure usare un solo "se... fai" componendo la *condizione* tramite l'*operatore logico* "... e...":



È necessario ripetere l'istruzione precedente 4 volte (una per ciascuna direzione possibile) per un costo complessivo pari a 4 oppure costruirne una sola usando l'operatore logico "... o...":

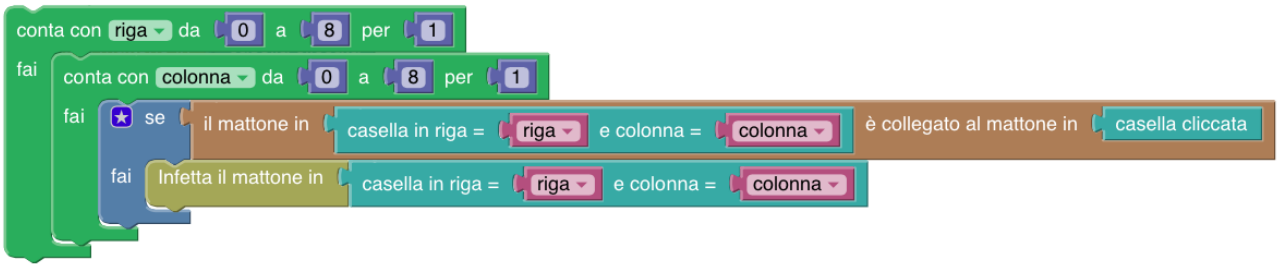


da completare inserendo in ciascuno dei 4 spazi una condizione relativa ad una delle direzioni.

Notate che il blocco "casella sotto a" (e varianti) può essere costruito anche usando gli indici di riga e di colonna, risparmiando quindi una unità di costo ogni volta:

Una volta stabilito che l'area di contagio non è vuota, bisogna determinare quali mattoni ne fanno parte. Una prima strategia, non molto economica, si basa sull'utilizzo del blocco "il mattone in ... è collegato al mattone in ...", di costo 10; la condizione

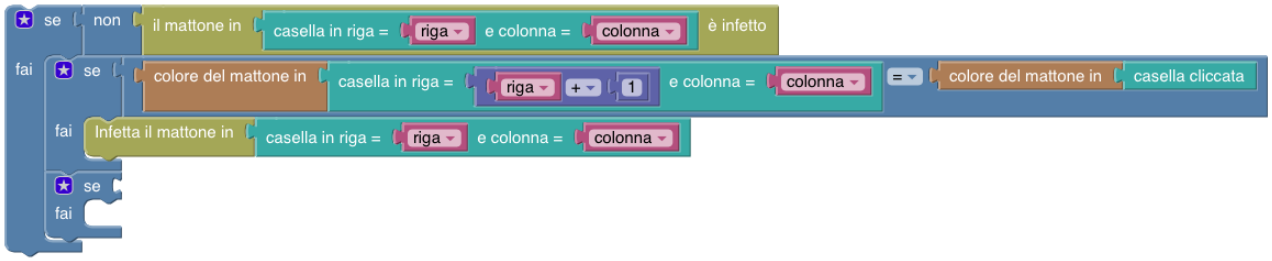
è vera, infatti, se e solo se il mattone in questione fa parte dell'area di contagio. Perciò, esaminando tutte le caselle del campo di gioco e verificando per ciascuna se la condizione precedente è soddisfatta o meno, si riesce a definire l'intera area di contagio. Per poter esaminare tutte le caselle, una sola volta ciascuna, è utile seguire un qualche qualche ordine, ad esempio si può procedere riga per riga partendo dall'alto e da sinistra verso destra, usando due *cicli* "conta con..." annidati:



I blocchi "riga" e "colonna", che rappresentano gli indici di riga e colonna della casella in esame, sono delle *Variabili*; il nome di una variabile in blockly può essere scelto liberamente, ma è utile scegliere un nome che renda esplicito il significato della variabile.

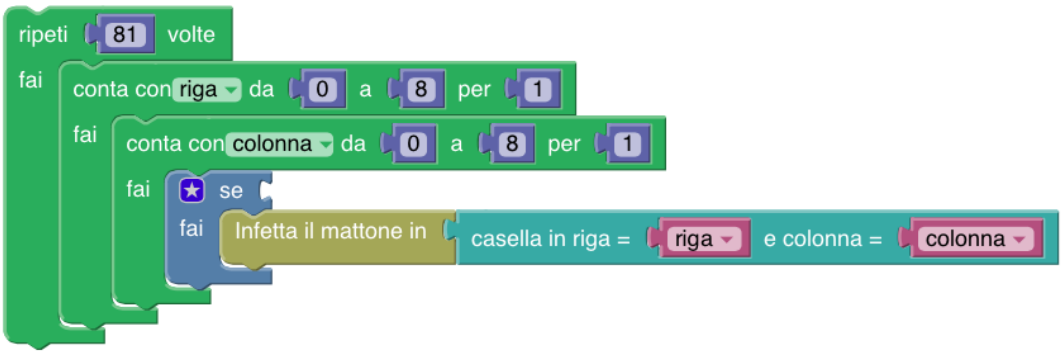
Per ridurre il costo di questa soluzione, è necessario implementare con blocchi piú economici una strategia di propagazione dell'infezione, partendo dalla casella cliccata e spostandosi passo dopo passo alle caselle vicine.

Possiamo sfruttare di nuovo l'uso di due cicli annidati e mantenere la struttura del programma precedente. Per ogni casella che contiene un mattone non infetto, se una delle 4 caselle adiacenti contiene un mattone infetto, il mattone in esame va senz'altro infettato. Tale controllo è realizzato ad esempio dalla seguente porzione di programma:

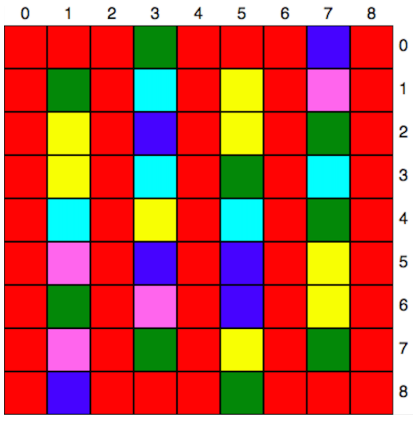


che, come sopra, va completata in modo da considerare tutte e 4 le direzioni, e corretta per evitare il problema dei bordi. Anche in questo caso, i quattro "se... fai" piú interni potrebbero essere sostituiti da un solo "se... fai" con la condizione composta tramite l'operatore logico "... o...", e i due "se... fai" annidati risultanti potrebbero essere sostituiti da un unico "se... fai" con la condizione composta tramite l'operatore logico "... o...". In questo modo si ottiene una soluzione di costo pari a 1. Notate che si è usato il blocco "non" per negare la condizione relativa all'infezione del mattone in esame.

Se l'area di contagio è ampia, una sola *passata* del campo di gioco però non basta poichè l'infezione raggiunge soltanto le caselle adiacenti ai mattoni già infetti; è necessario riesaminare l'intero campo da gioco finché non ci saranno piú nuovi mattoni da infettare. In questo modo, alla prima passata l'infezione passa ai vicini della casella cliccata, alla passata successiva passa ai vicini dei vicini, e cosí via. Come stabilire quando ci si può fermare? Data la dimensione del campo, ripetere l'esame 81 volte sarà senz'altro sufficiente:



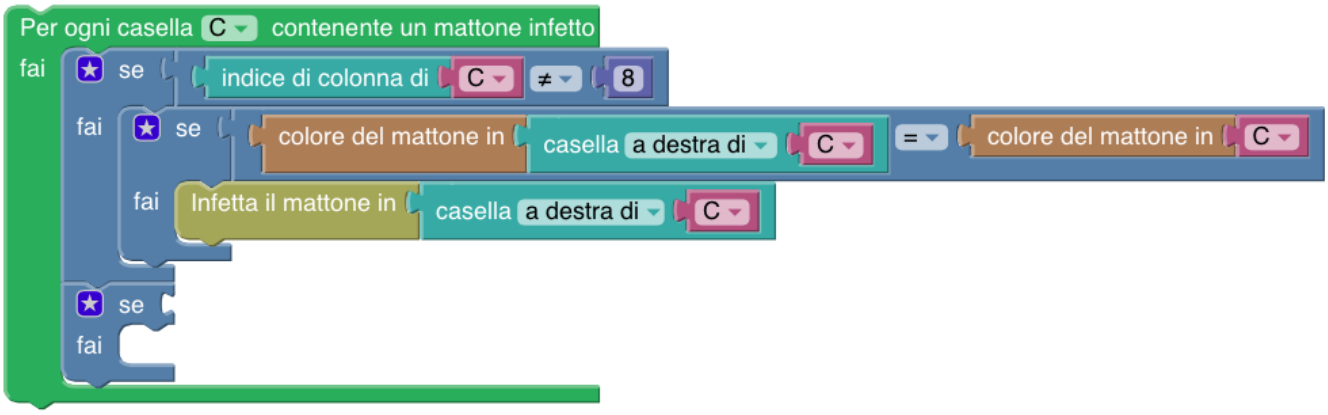
9 ripetizioni invece non bastano: immaginate ad esempio che l'area di contagio sia lunga, stretta e arrotolata come un serpente, come nella figura seguente.



Con questa strategia, l'ordine con cui i mattoni vengono infettati dipende dalla vicinanza dal mattone cliccato, mentre usando il blocco "il mattone in... è collegato al mattone in ..." i mattoni venivano infettati nell'ordine in cui erano visitati, cioè riga per riga dall'alto verso il basso e da sinistra verso destra.

Una strategia simile, ma in un certo senso duale, si otterrebbe analizzando il campo di gioco casella per casella, considerando solo le caselle infette e decidendo, di volta in volta, se sussistono le condizioni affinché vadano infettate le caselle vicine. Anche qui, non basta una sola passata; inoltre, dato che il mattone da infettare non è quello in esame ma uno dei suoi quattro vicini, bisogna costruire 4 istruzioni distinte che usano ciascuna il blocco "infetta il mattone in...", ottenendo quindi un costo complessivo più alto.

Per evitare di dover ripetere numerose volte la visita dell'intero campo da gioco, si può sfruttare il blocco "per ogni casella C contenente un mattone infetto": bisogna controllare se C ha dei vicini del colore giusto e, in questo caso, infettarli, tramite uno schema di questo tipo

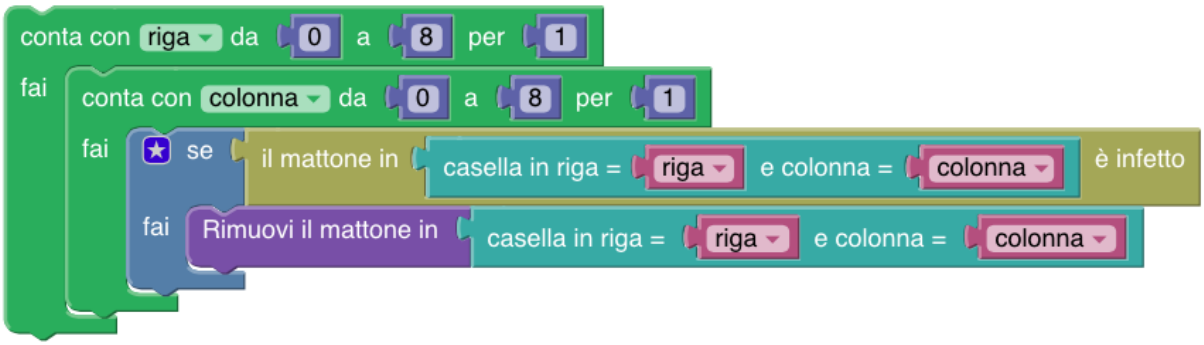


Ad ogni *iterazione* viene esaminata una casella infetta (il ciclo non inizia nemmeno, se non ve ne sono) e vengono eventualmente infettati i mattoni nelle caselle vicine, che saranno esaminati quindi in un'*iterazione* successiva; ogni casella contenente un mattone infetto viene considerata una sola volta nel *ciclo* e quando non ci sono piú caselle con mattoni infetti da esaminare, il ciclo si conclude.

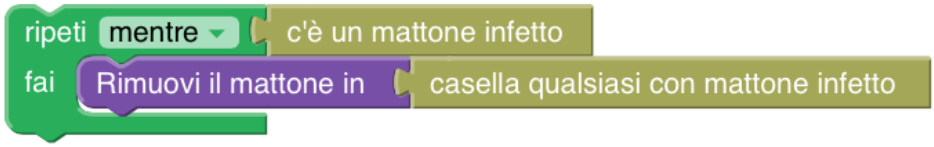
In questo caso l'eliminazione del blocco "casella a destra di..." (o varianti) non è immediato poiché richiede ad esempio l'uso dei blocchi "indice di riga di..." e "indice di colonna di...":

3.2 Rimozione dei mattoni infetti

Una volta che tutti i mattoni nell'area di contagio sono stati infettati, è possibile rimuoverli uno a uno. Un primo approccio consiste nel considerare tutte le caselle in ordine, rimuovendo il mattone nella casella in esame qualora risultasse infetto:

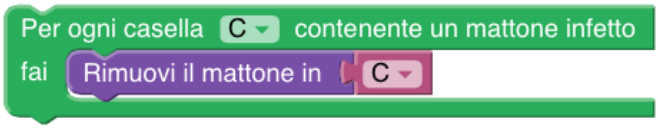


Un'alternativa, piú sintetica ma anche piú costosa (4 unità invece di 1), si ottiene usando il *ciclo* "ripeti mentre":



In questo caso, dato che ad ogni iterazione un mattone infetto viene rimosso, l'insieme dei mattoni infetti si restringe via via fino a svuotarsi del tutto, cosa che determina l'*uscita* dal ciclo. Il blocco "casella qualsiasi con mattone infetto" può essere sostituito anche con il blocco "casella piú in alto con mattone infetto" o una delle sue varianti; in generale cambierà l'ordine con cui sono rimossi i mattoni.

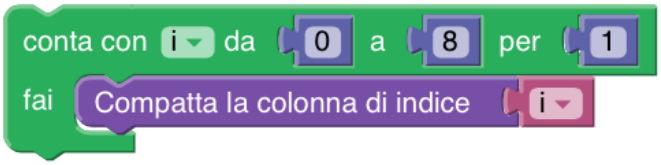
Una terza soluzione usa invece di nuovo il costrutto "Per ogni casella C contenente un mattone infetto":



In questo caso il costo è ridotto a 1 unità e la soluzione appare comunque sintetica perché incorpora (e in un certo senso *nasconde*) all'interno del costrutto la condizione relativa all'infezione.

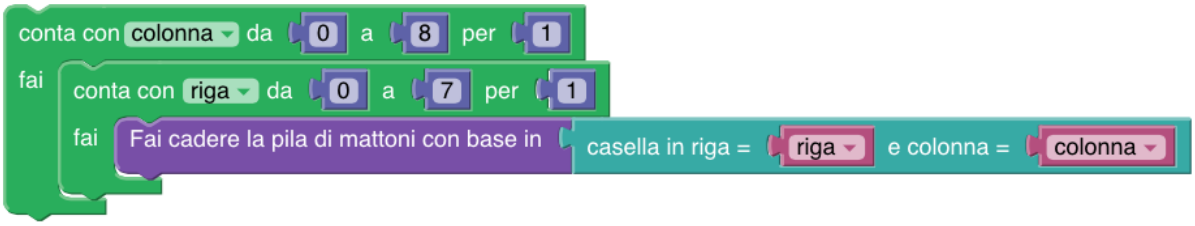
3.3 Caduta dei mattoni ed eliminazione delle caselle vuote

La caduta dei mattoni coinvolge una colonna alla volta; quindi per aggiornare il campo da gioco basta usare il blocco "Compatta la colonna di indice" all'interno di un semplice ciclo "conta con...", per un costo di 7 unità:



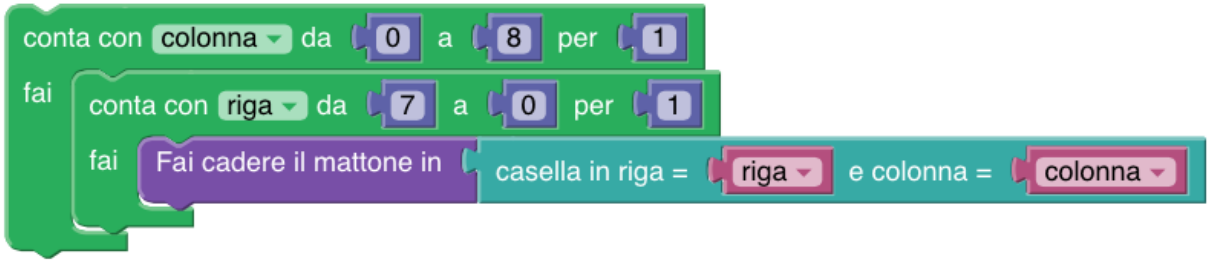
Questo blocco può essere implementato anche usando blocchi meno costosi tra quelli a disposizione: alcuni spostano un solo mattone, altri spostano un'intera pila di mattoni sovrapposti; alcuni fanno scendere un mattone –o una pila di mattoni– di una sola posizione, altri ("Fai cadere...") consentono invece di farli scendere –eventualmente per più di una posizione– fintanto che ci sono caselle vuote sotto. Per tutti questi blocchi, se il mattone non si trova sopra una casella vuota, l'istruzione non ha effetto; inoltre non sono considerate *vuote* le caselle nere, cioè quelle che non contengono mattoni ma che non hanno mattoni nemmeno sopra di loro.

Ad esempio, con un costo di 4 unità, si ottiene questa soluzione:

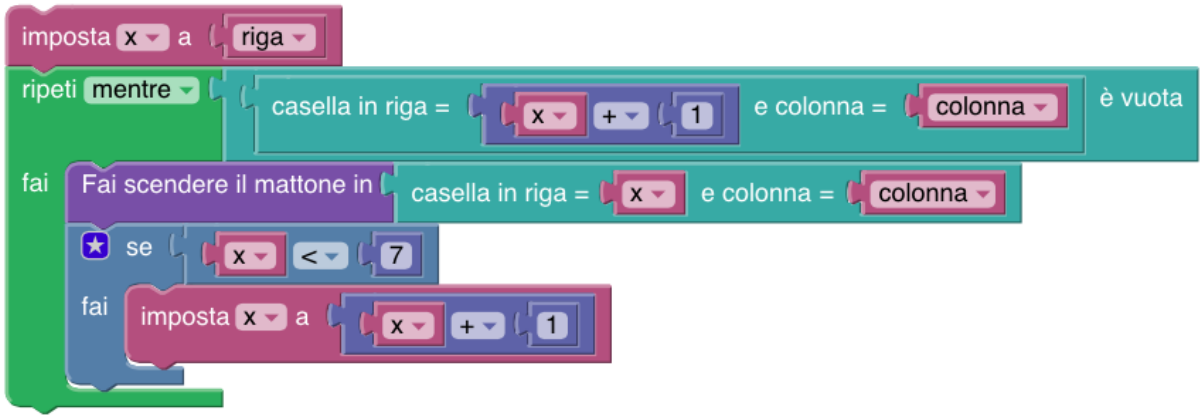


Basta considerare le righe fino all'indice 7 dato che i mattoni nella riga di indice 8 (la piú bassa) non posso scendere ulteriormente.

Sostituendo il blocco "Fai cadere la pila di mattoni..." con "Fai cadere il mattone..." non si avrebbe lo stesso effetto: infatti usando questo blocco le caselle vuote si spostano in alto al piú di una sola posizione, quindi se una casella vuota ha sopra di sé piú di un mattone, alla fine del doppio ciclo ci sarà ancora almeno una casella vuota. Per evitare questo problema si può iterare la procedura un numero abbastanza grande di volte (come nel caso della propagazione dell'infezione), oppure si può aggirare il problema considerando i mattoni dal basso verso l'alto, per un costo complessivo pari a 2:

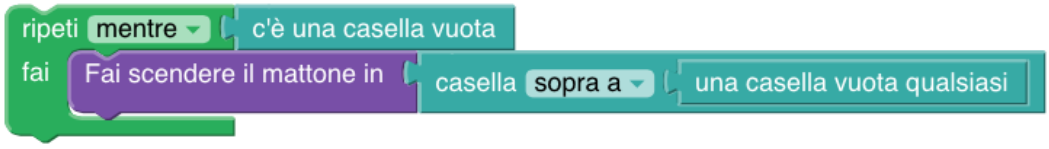


Una soluzione a costo 0 si ottiene implementando il blocco "Fai cadere il mattone..." tramite il blocco "Fai scendere il mattone...":



In questo caso è necessario usare un *contatore* (la variabile x) che tiene traccia di quante sono le posizioni di cui scendere.

In alternativa all'uso del doppio ciclo è possibile usare anche per la caduta dei mattoni il costrutto *ripeti*, sfruttando il fatto che i mattoni da far cadere si trovano sempre sopra una casella vuota.



In questo caso si è usato il blocco “ripeti mentre” con la condizione “c’è una casella vuota”: ad ogni iterazione una casella vuota sale verso la prima riga e, prima o poi, tutte le caselle vuote diventeranno nere, portando quindi alla conclusione del ciclo. È possibile eliminare l’uso di questa condizione (di costo 2) a patto di ripetere la caduta di un mattone sopra una casella vuota qualsiasi un numero sufficientemente grande di volte (quante sono necessarie?), come nel caso della propagazione dell’infezione. Anche il blocco “casella sopra a...” non è immediato da sostituire. Ad esempio, la porzione di codice

potrebbe indicare una casella diversa da quella voluta, poiché le due casella *qualsiasi* non è detto che coincidano! È necessario dunque usare una variabile per memorizzare la casella qualsiasi scelta:

Il rimanente costo di 2 unità relative al blocco “casella vuota qualsiasi” (o dell’analogo “casella vuota piú in alto” e varianti), non è (CREDO!!) eliminabile a meno di esaminare le caselle una per volta ad esempio con l’utilizzo dei due cicli annidati.