

# SilverCoders

POTENZIAMENTO DELL'ALFABETIZZAZIONE DIGITALE ATTRAVERSO LE  
ESPERIENZE DI APPRENDIMENTO PER ADULTI



## CHALLENGE #28

### TRIS

PROGRAMMA DI FORMAZIONE  
SUL CODING **PER**  
**ADULTI +55**



SILVER CODERS



ERASMUS+ n. 2020-1-SE01-KA227-ADU-092582



Co-funded by  
the European Union

*Questo documento riflette solo il punto di vista dell'autore e l'Agenzia Nazionale e la  
Commissione Europea non sono responsabili dell'uso che può essere fatto delle  
informazioni in esso contenute.*

# STRUTTURA DELLA CHALLENGE

## DESCRIZIONE

In questa challenge verrà creato un gioco simile a Tris, pensato per due giocatori.

## OBIETTIVO GENERALE

In questa challenge verrà creato un gioco pensato per due giocatori simile a Tris. I corsisti approfondiranno anche la loro conoscenza di una forma di memorizzazione dei dati conosciuta come “array”.

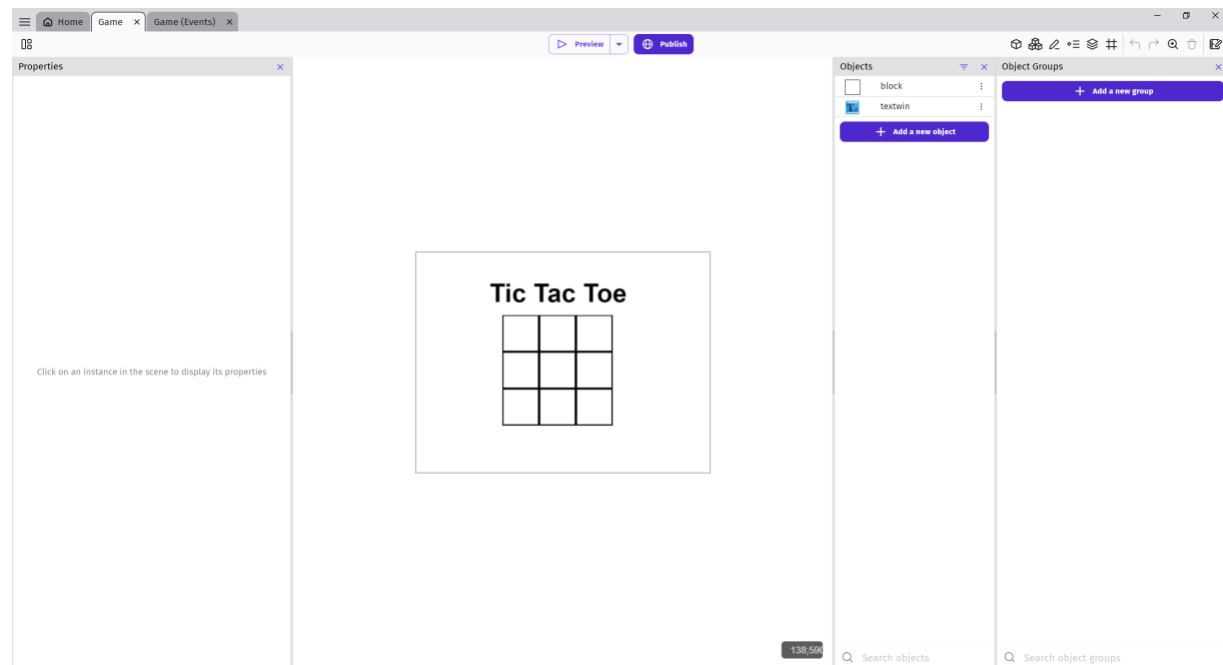
## OBIETTIVI DI APPRENDIMENTO

Al termine di questa challenge, sarai in grado di:

- Acquisire esperienza con una suite di programmazione visuale e codificare con essa semplici software standard.
- Conoscere le dichiarazioni, le linee di comando e il loro significato per il compilatore.
- Scrivere istruzioni utilizzando una sintassi corretta con minimi errori.
- Conoscere gli operatori, la loro funzione e i loro operandi (quali simboli corrispondono a quali operatori).
- Comprendere l'assegnazione di valori alle variabili e come modificarli.
- Conoscere tutte le operazioni aritmetiche di base e come utilizzarle.
- Riconoscere e utilizzare tutte le strutture dati relative ai numeri.
- Identificare le strutture legate all'uso del testo, come stringhe e caratteri.
- Utilizzare correttamente le istruzioni condizionali per eseguire il codice in base ad una condizione fissa definita.
- Utilizzare gli array.

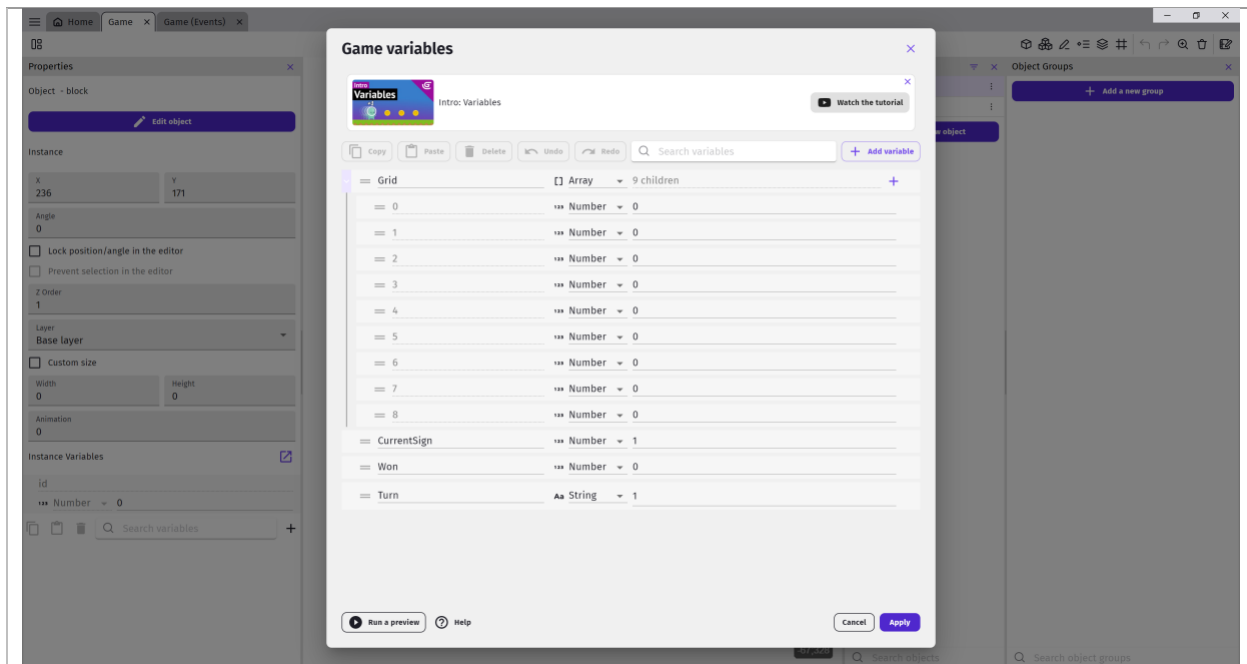
# ISTRUZIONI

Questa è la configurazione iniziale. In questo caso, abbiamo fornito gli oggetti di base di cui avrai bisogno per il gioco. Come al solito, inizia a controllarli con attenzione.



Nella configurazione riportata, si possono distinguere diversi elementi importanti:

- Ogni tessera della tabella è una sprite di **blocco**. Ogni istanza o copia di **blocco** presenta una variabile che lo identifica, chiamata **id**. Le tessere in alto sono 0, 1 e 2. Le tessere della riga media sono 3, 4 e 5, mentre quelle della riga inferiore sono 6, 7 e 8.
- La sprite del **blocco** ha 3 fotogrammi: uno per lo spazio vuoto (fotogramma 0), uno per la X (fotogramma 1) e un altro per la O (fotogramma 2).
- Nella scena sono state create diverse variabili:
  - o **CurrentSign** indica quale cornice/segno deve essere rappresentata/o quando si sceglie una piastrella.
  - o **Won** indica il vincitore.
  - o **Turno** indica il turno del giocatore 1 o 2.
- La variabile più importante è quella **Grid**, un array di 9 posizioni che indica quale simbolo si trova in una determinata posizione. All'inizio tutte le posizioni sono pari a 0 (vuote).



È presente anche il codice di avvio del gioco e la struttura della parte restante del codice.

When starting the scene, assign the first Turn to a random player 0 or 1, where 0 is X, and 1 is O	
<ul style="list-style-type: none"> <li>At the beginning of the scene</li> </ul>	<ul style="list-style-type: none"> <li>Change the scene variable <b>Turn</b>: set to Random(1)</li> </ul>
Check whose Turn it is and set the appropriate animation to the block	
<ul style="list-style-type: none"> <li>Left mouse button was released</li> </ul>	<ul style="list-style-type: none"> <li>Add action</li> </ul>
Each box (block) has different ids representing there distance from the first box (Box at top left is 0 and the increments by 1 from there to the right and to the bottom row)	
The Grid index that matches the id of the block is set to 1 if it X and 2 if it is Y	
<ul style="list-style-type: none"> <li>The scene variable <b>Turn</b> = 0</li> <li>The cursor/touch is on <b>block</b></li> <li>The number of the animation of <b>block</b> = 0</li> </ul>	<ul style="list-style-type: none"> <li>Change the number of the animation of <b>block</b>: set to 1</li> <li>Change the scene variable <b>Turn</b>: set to 1</li> <li>Change the scene variable <b>Grid[block.Variable(id)]</b>: set to 1</li> </ul>
<ul style="list-style-type: none"> <li>The scene variable <b>Turn</b> = 1</li> <li>The cursor/touch is on <b>block</b></li> <li>The number of the animation of <b>block</b> = 0</li> </ul>	<ul style="list-style-type: none"> <li>Change the number of the animation of <b>block</b>: set to 2</li> <li>Change the scene variable <b>Turn</b>: set to 0</li> <li>Change the scene variable <b>Grid[block.Variable(id)]</b>: set to 2</li> </ul>

Questo codice imposta casualmente il giocatore iniziale. Inoltre, segnala quando il giocatore ha premuto una tessera vuota, inserisce il simbolo corrispondente al giocatore e riempie la posizione **della griglia** corrispondente con il valore corretto.

Ciò che resta da fare è verificare se uno dei due giocatori è riuscito a inserire 3 simboli uguali su una linea orizzontale, verticale o diagonale. A tal fine, è possibile controllare l'array **Grid**. Comincia con le linee orizzontali:

**Winning system**

The value of each box is stored in an array (from 0 to 8) and their value can be 0 (means empty) 1 (means X) 2 (means O)

```
[0 | 1 | 2]
[3 | 4 | 5]
[6 | 7 | 8]
```

☒ The scene variable ☒ Won = 0

Add condition

CurrentRow represents the current row we are checking (if there is a match of 3). CurrentColumn represents the current Column we are checking (if there is a match of 3). CurrentSign represents the current sign (1= X or 2 = O) we are checking for a match

Checking for horizontal matches,  
CurrentRow is added to the the current Grid index (box) value so that we are iterating through all the columns with 3 values neighboring them instead of having seperate events for each row that is,  
CurrentRow increments by 3  
When Current Row is 0, ( 0+0 = 0 | 1 + 0 = 1 | 2 + 0 = 2 )

```
[0 | 1 | 2]
[ | | ]
[ | | ]
```

When CurrentRow is 3, ( 0+3 = 3 | 1 + 3 = 4 | 2 + 3 = 5 )

```
[ | | ]
[3 | 4 | 5]
[ | | ]
```

When CurrentRow is 6, ( 0+6 = 6 | 1 + 6 = 7 | 2 + 6 = 8 )

```
[ | | ]
[ | | ]
[6 | 7 | 8]
```

☒ The scene variable ☒ Grid[0+Variable(CurrentRow)] = Variable(CurrentSign)  
☒ The scene variable ☒ Grid[1+Variable(CurrentRow)] = Variable(CurrentSign)  
☒ The scene variable ☒ Grid[2+Variable(CurrentRow)] = Variable(CurrentSign)

Add condition

☒ Change the scene variable ☒ Won: set to Variable(CurrentSign)

Add action

La spiegazione completa è riportata nei commenti al codice.

Ora, per le linee verticali:

Checking for vertical matches  
CurrentColumn increments by 1  
When CurrentColumn is 0, ( 0+0 = 0 | 3 + 0 = 1 | 6 + 0 = 1 )

```
[0 | | ]
[3 | | ]
[6 | | ]
```

When CurrentColumn is 1, ( 0 + 1 = 0 | 3 + 1 = 1 | 6 + 1 = 1 )

```
[ | 1 | ]
[ | 4 | ]
[ | 7 | ]
```

When CurrentColumn is 2, ( 0 + 2 = 0 | 3 + 2 = 1 | 6 + 2 = 1 )

```
[ | | 2 ]
[ | | 5 ]
[ | | 8 ]
```

☒ The scene variable ☒ Grid[0+Variable(CurrentColumn)] = Variable(CurrentSign)  
☒ The scene variable ☒ Grid[3+Variable(CurrentColumn)] = Variable(CurrentSign)  
☒ The scene variable ☒ Grid[6+Variable(CurrentColumn)] = Variable(CurrentSign)

Add condition

☒ Change the scene variable ☒ Won: set to Variable(CurrentSign)

Add action

E infine per le diagonali:

Checking for diagonal matches,  
[ | | 2]      [0 | | ]  
[ | 4 | ] OR [ | 4 | ]  
[6 | | ]      [ | | 8]

☒ The scene variable ☒ Grid[2] = Variable(CurrentSign)  
☒ The scene variable ☒ Grid[4] = Variable(CurrentSign)  
☒ The scene variable ☒ Grid[6] = Variable(CurrentSign)

Add condition

☒ Change the scene variable ☒ Won: set to Variable(CurrentSign)

Add action

☒ The scene variable ☒ Grid[0] = Variable(CurrentSign)  
☒ The scene variable ☒ Grid[4] = Variable(CurrentSign)  
☒ The scene variable ☒ Grid[8] = Variable(CurrentSign)

Add condition

☒ Change the scene variable ☒ Won: set to Variable(CurrentSign)

Add action

Add condition

☒ Change the scene variable ☒ CurrentRow: add 3  
☒ Change the scene variable ☒ CurrentColumn: add 1  
☒ Change the scene variable ☒ CurrentSign: add 1

Add action

Ora occupati del passaggio del turno al giocatore successivo:

Add action	
When CurrentSign is greater than 3, it is reset to 1 (X)	
<input checked="" type="checkbox"/> The scene variable <input checked="" type="checkbox"/> CurrentSign > 2 Add condition	<input checked="" type="checkbox"/> Change the scene variable <input checked="" type="checkbox"/> CurrentSign: set to 1 Add action
When CurrentRow is greater than 6, that is at row 3 (last row)	
<input checked="" type="checkbox"/> The scene variable <input checked="" type="checkbox"/> CurrentRow > 6 Add condition	<input checked="" type="checkbox"/> Change the scene variable <input checked="" type="checkbox"/> CurrentRow: set to 0 Add action
When CurrentRow is greater/equal than 3, that is at column 3 (last column)	
<input checked="" type="checkbox"/> The scene variable <input checked="" type="checkbox"/> CurrentColumn ≥ 3 Add condition	<input checked="" type="checkbox"/> Change the scene variable <input checked="" type="checkbox"/> CurrentColumn: set to 0 Add action

Se quest'altro giocatore è riuscito a inserire 3 simboli uguali su una linea orizzontale, verticale o diagonale, congratulati con lui/lei!

<input checked="" type="checkbox"/> The scene variable <input checked="" type="checkbox"/> Won ≠ 0 <input checked="" type="checkbox"/> Trigger once Add condition	Add action
The title is changed according to which player won	
<input checked="" type="checkbox"/> The scene variable <input checked="" type="checkbox"/> Won = 1 Add condition	txr Change the text of <input checked="" type="checkbox"/> textwin: set to "X Won" Add action
<input checked="" type="checkbox"/> The scene variable <input checked="" type="checkbox"/> Won = 2 Add condition	txr Change the text of <input checked="" type="checkbox"/> textwin: set to "Y Won" Add action

## RISORSE

### Challenge 28 (Base)