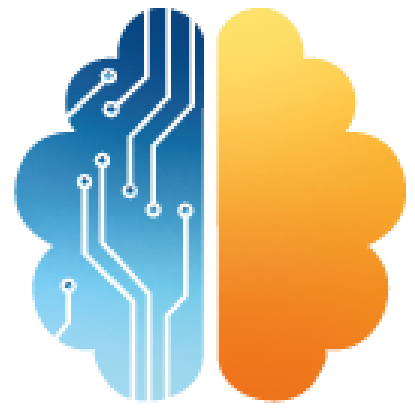


# SilverCoders

DIGITAL LITERACY IMPROVEMENT THROUGH EFFECTIVE  
LEARNING EXPERIENCES FOR ADULTS



## DESAFIO #28 **TIC TAC TOE**

### CODING TRAINING PROGRAMME **FOR +55 ADULTS**



**SILVER CODERS**

ERASMUS+ No. 2020-1-SE01-KA227-ADU-092582



**Co-funded by  
the European Union**

*This document reflects only the author's view and the National Agency and the European Commission are not responsible for any use that may be made of the information it contains*

# ESTRUTURA DO DESAFIO

## DESCRIÇÃO

Vamos criar um jogo do Tic Tac Toe. É para ser jogado por duas pessoas.

## OBJETIVO GERAL

Vamos criar um jogo do Tic Tac Toe, destinado a ser jogado por duas pessoas. Também aprenderemos sobre arrays, uma forma de armazenar dados.

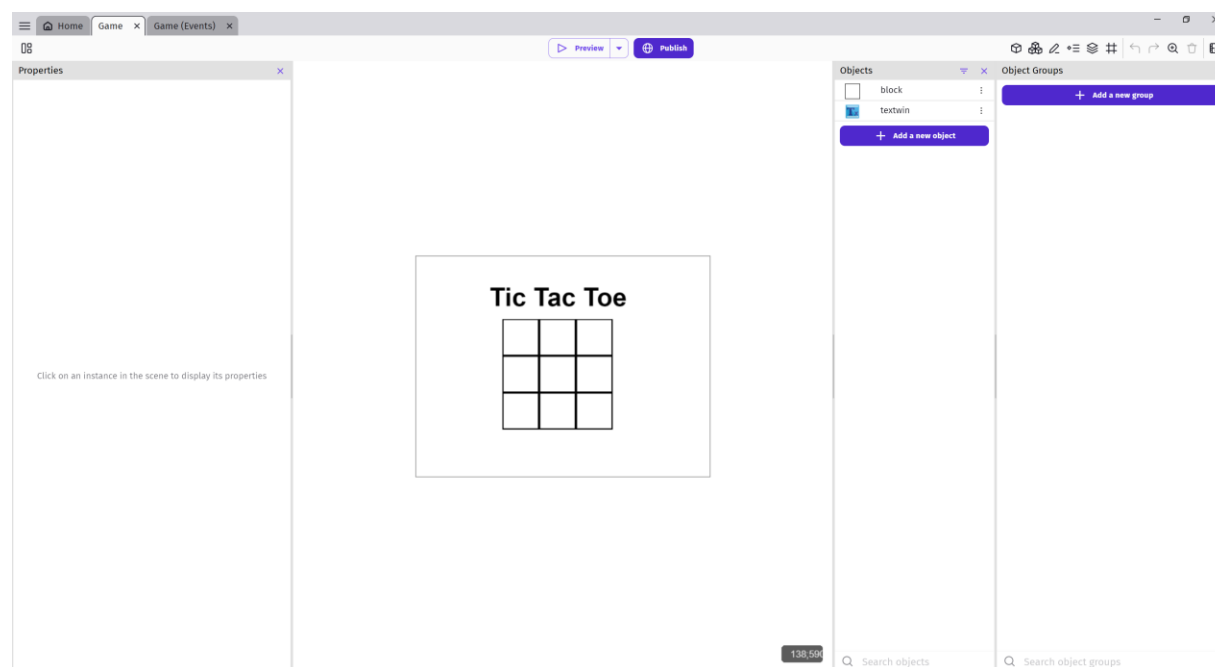
## OBJETIVOS DE APRENDIZAGEM

No final deste desafio, poderás...

1. Ter experiência com uma suíte de programação visual e ser capaz de codificar uma pequena peça de software padrão com ele.
2. Saber o que são declarações e linhas de comando e o que significam para um compilador.
3. Escrever instruções utilizando a sintaxe correta e com erros mínimos.
4. Saber o que são os operadores, o que fazem e quais os símbolos que representam os operadores.
5. Ser capaz de entender a atribuição de valores a variáveis e como mudá-los.
6. Conhecer todas as operações aritméticas básicas e como usá-las.
7. Reconhecer e saber como utilizar todas as estruturas de dados relacionadas com os números.
8. Conhecer as estruturas ligadas ao uso do texto, como cordas e caracteres.
9. Utilizar declarações condicionais.
10. Usar arrays.

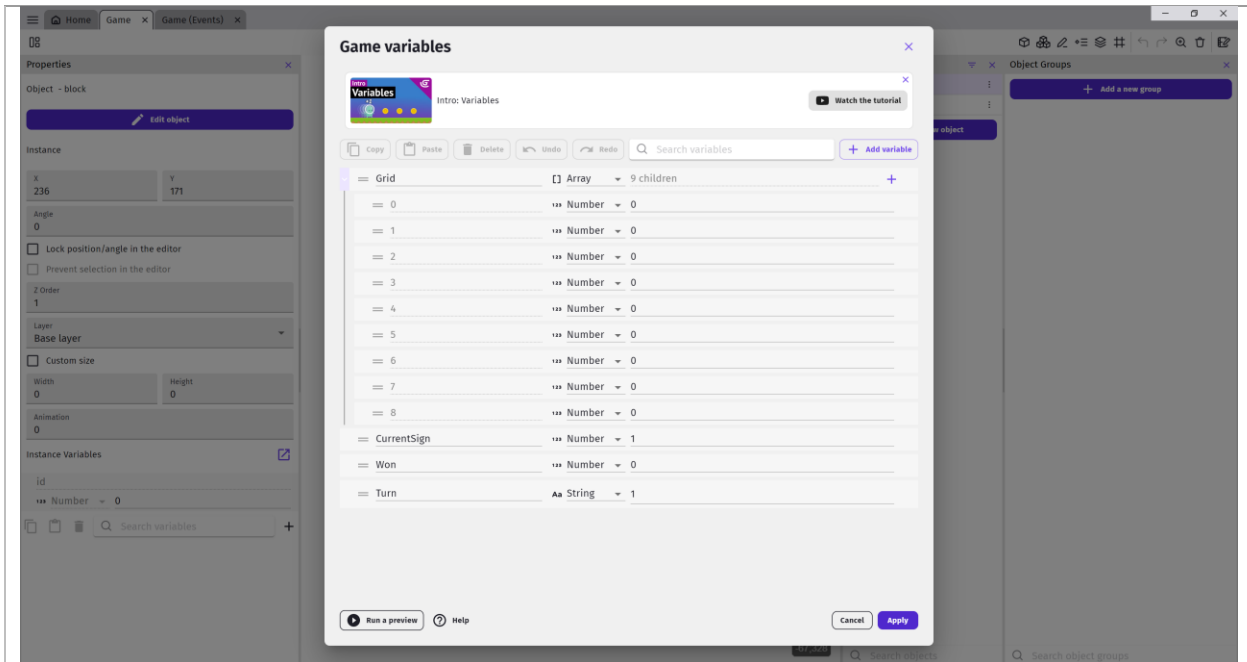
# INSTRUÇÕES

Esta é a sua configuração inicial. Neste caso, fornecemos os objetos básicos que vai precisar para o jogo. Como sempre, comece por verificá-los cuidadosamente.



Há vários aspetos importantes nesta configuração:

1. Cada azulejo de prancha é um sprite de **bloco**. Cada instância ou cópia do **bloco** tem uma variável chamada **id** que o identifica. Então os azulejos superiores são 0,1 e 2. Os azulejos de linha média são 3, 4 e 5 e os inferiores são 6, 7 e 8.
2. O bloco sprite tem 3 quadros: um para o espaço vazio (quadro 0), um para o X (quadro 1) e outro para o O (quadro 2).
3. A cena tem várias variáveis criadas:
  1. CorrenteSssa assinatura indica qual quadro (ou sinal deve ser representado quando escolhemos um azulejo).
  2. **Won** diz-nos se alguém já ganhou.
  3. **Turn** diz-nos se é jogador 1 ou 2 para jogar
4. A variável mais importante é **Grid**, uma matriz com 9 posições que nos diz qual o símbolo que está numa determinada posição. Quando começamos, todas as posições são 0 (vazias).



Também temos o código que começa o jogo e temos a estrutura para o resto do código.

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>
Add condition	Add action
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>
Add condition	
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>
Add condition	Add action
<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>
Add condition	Add action

Este código define aleatoriamente o leitor inicial. Em seguida, verifica se premimos um azulejo vazio, colocamos ali o símbolo do jogador e preenchemos a posição **correspondente da Grelha** com o valor certo.

O que resta é verificar se um jogador. Isto significa verificar se conseguiu colocar 3 símbolos iguais numa linha horizontal, vertical ou diagonal. Vamos fazê-lo verificando a matriz **grid**. Começemos pelas linhas horizontais:

**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 serperate 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

Uma explicação completa é dada sobre os comentários de código.

Agora, para as linhas verticais.

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 finalmente para os diagonais.

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

Temos agora de lidar com a mudança de turno para o próximo jogador.

	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

E se alguém um, vamos felicitá-lo.

<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	<input checked="" type="checkbox"/> 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	<input checked="" type="checkbox"/> Change the text of <input checked="" type="checkbox"/> textwin: set to "Y Won" Add action

## RECURSOS

### Challenge 28 (Basic)