

SilverCoders

DIGITAL LITERACY IMPROVEMENT THROUGH EFFECTIVE
LEARNING EXPERIENCES FOR ADULTS



CHALLENGE #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

STRUCTURE OF THE CHALLENGE

DESCRIPTION

We are going to create a Tic Tac Toe kind of game. It is meant to be played by two persons.

GENERAL GOAL

We are going to create a Tic Tac Toe kind of game, meant to be played by two persons. We will also learn about Arrays, a form of storing data.

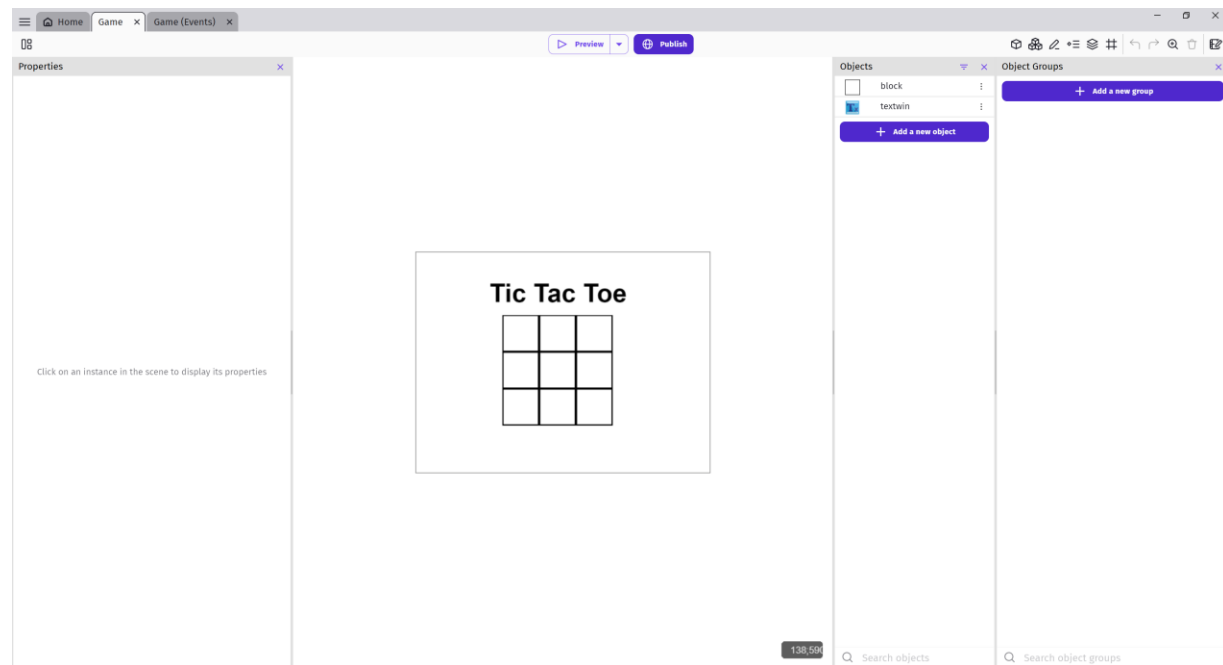
LEARNING OBJECTIVES

In the end of this challenge, you will be able ...:

- To have experience with a visual programming suite and be able to code standard small piece of software with it.
- Know what statements and command lines are and what they mean for a compiler.
- To be able to write instructions using correct syntax and with minimal errors.
- Know what operators are, what they do and which symbols stand for which operators.
- To be able to understand the assignment of values to variables and how to change them.
- To know all the basic arithmetic operations and how to use them.
- Recognize and know how to use all the data structures related to numbers.
- To know the structures linked to the use of text, such as strings and characters.
- To be able to use If statements correctly to execute code according to a certain defined fixed condition.
- To be able to use Arrays.

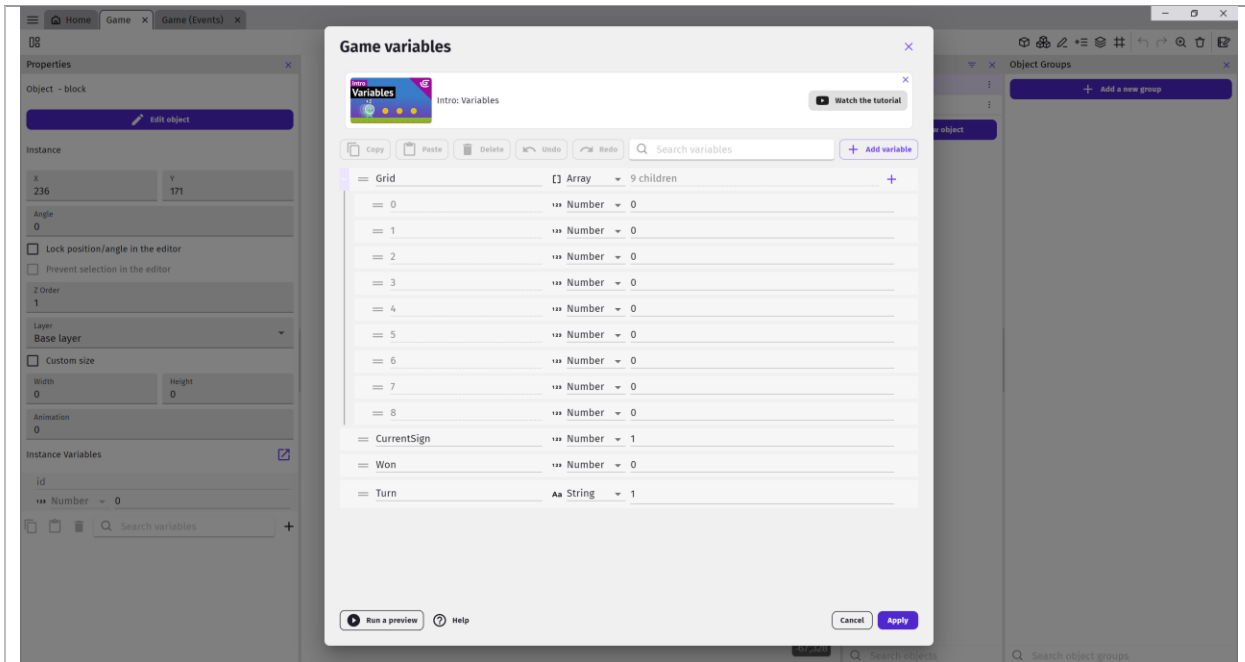
INSTRUCTIONS

This is your initial setup. In this case we provided the basic objects that you'll need for the game. As usual start by checking them carefully.



There are several important aspects in this setup:

- Each board tile is a **block** sprite. Each instance or copy of **block** has a variable called **id** that identifies it. So the top tiles are 0,1 and 2. The medium row tiles are 3, 4 and 5 and the bottom ones are 6, 7 and 8.
- The **block** sprite has 3 frames: one for the empty space (frame 0), one for the X (frame 1) and another for the O (frame 2).
- The scene has several variables created:
 - o **CurrentSign** indicates which frame (or sign should be represented when we choose a tile).
 - o **Won** tells us if someone already won.
 - o **Turn** tells us if it is player 1 or 2 to play
- The most important variable is **Grid**, an array with 9 positions that tells us which symbol is in a certain position. When we start all the positions are 0 (empty).



We also have the code that starts the game and we have the structure for the rest of the code.

When starting the scene, assign the first Turn to a random player 0 or 1, where 0 is X, and 1 is O	
At the beginning of the scene	Change the scene variable Turn : set to Random(1)
Add condition	Add action
Check whose Turn it is and set the appropriate animation to the block	
Left mouse button was released	Add action
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	
The scene variable Turn = 0	Change the number of the animation of block : set to 1
The cursor/touch is on block	Change the scene variable Turn : set to 1
The number of the animation of block = 0	Change the scene variable Grid[block.Variable(id)]: set to 1
Add condition	Add action
The scene variable Turn = 1	Change the number of the animation of block : set to 2
The cursor/touch is on block	Change the scene variable Turn : set to 0
The number of the animation of block = 0	Change the scene variable Grid[block.Variable(id)]: set to 2
Add condition	Add action

This code randomly sets the starting player. Then it checks if we pressed an empty tile, put there the player symbol and fills the corresponding **Grid** position with the right value.

What is left is to check if a player one. That means checking if he managed to put 3 equal symbols on a horizontal, vertical or diagonal line. We will do this by checking the **Grid** array. Let's start with the horizontal lines:

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

A complete explanation is given on the code comments.

Now, for the vertical lines.

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

And finally for the diagonal ones.

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

We now need to deal with the change of turn to the next player.

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

And if someone one, let's congratulate him/her.

<input checked="" type="checkbox"/> The scene variable <input checked="" type="checkbox"/> Won ≠ 0 Add condition	Add action
<input checked="" type="checkbox"/> Trigger once Add condition	
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

RESOURCES

Challenge 28 (Basic)