

Quick start guide

Version 1

1-Installa

The file PicrossC2.capx include all the assets necessary for running the framework.

The layouts are structured as following:

* **L1** for the first level
* **PicrossGen** for generating new picross

The event sheets are structured as following:

* **L1** event load the JSON variable that define the picross for the level L1 (layout L1). Each level event must include UniversalPicross event.
* **UniversalPicross** manage all the common events for Picross levels, like the grid, score… He don’t have to be changed for each level except for grid size change.
* **PicrossGenEvent** is linked to PicrossGen, he manages the JSON level description generation and saving.

The sprites are structured as following:

* **NumberFont** family define the sprites for each number in the grid, and each sprite include 2 frames, one for no-pushed and one for pushed.
* **Bravo** family defines the sprites random choosen for the level finished.
* **Case** sprite for empty place of the grid, **VPanel** for true place of the grid touched, and **XPanel** for false place of the grid touched.
* **TimeFinished** sprite shown when the player don’t have any more time to finish the picross.

**Warning:** Please follow all the instructions carefully. Any error

can make the Picross engine instable and unplayable.

2-Picross Generation Tool

2.1-Defining the picross

To define the picross to generate, open the PicrossGen layout in Construct 2, and after run it in your browser.

(figure 1 preview)

Now draw your picross on the grid.

2.2-Saving the picross

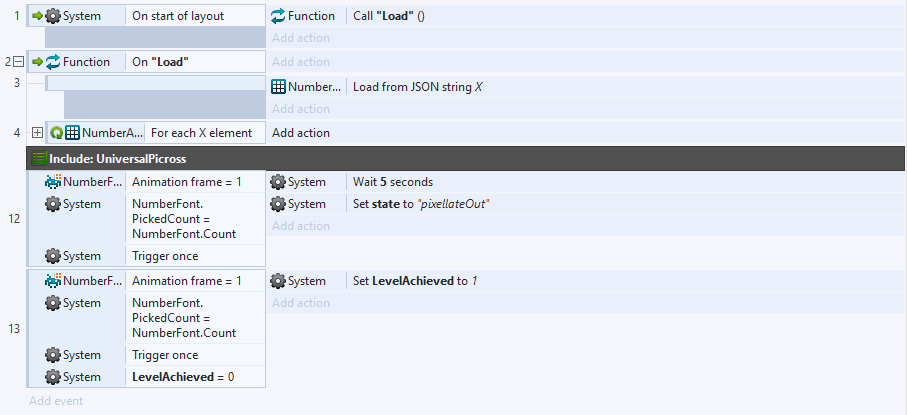
After you finish, click on generate to generate the numbers corresponding of the picross on the grid. Verify them and click on 5 to save the .JSON picross format file on your computer.

**Warning:** Don’t click many times on Generate, only once.

2.3-Loading the picross on a level layout

With the precedent step, we got the JSON file corresponding to the description of the level. We have to loading now in a level layout so the player can play it.

Create a new event that level linked to his layout, and copy entirely the event of the first level included in the project:



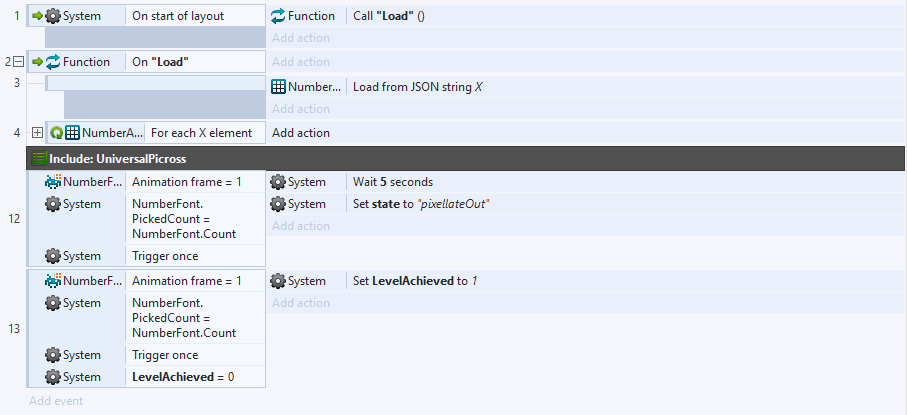
Create a new **global constant text variable** with value the JSON file downloaded precedently (open the file and copy the entire data, and paste it in “” in Construct 2). We will name this variable here *X*.



This variable store the level generated description.

**Notice:** In some Construct 2 versions, the long variable can provocate a bug that show the text upper the event. This doesn’t affect the running of the game.

Now follow this:



Set this as the level number.

Set this as the LevelNumber-1 (here level 1-1=0)

The X variable is the JSON text variable description of the level.

The level is ready, you can test it.

3-Advanced Picross Core editing

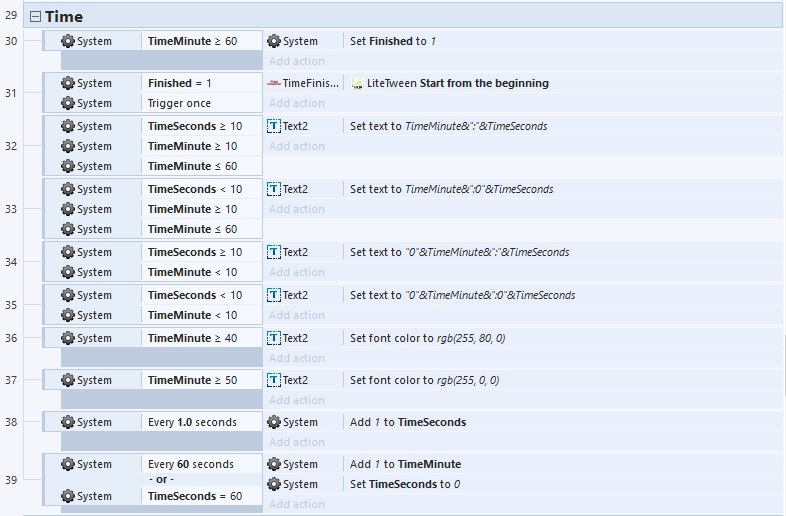
*UniversalPicross grid settings change*

3.1-Change time settings

The core of the PicrossC2 Framework is in **UniversalPicross**.

This event set things like time, grid, transitions… that is common to all picross levels.

The time group is structured as following:



Configure time text color

depending of the time

Configure time display

Set when the game finish (here an hour)

You can also set penalty settings:



Bug handler: this permits to not out

60 minutes with another mistake.

More than 4 errors: add 15 minutes

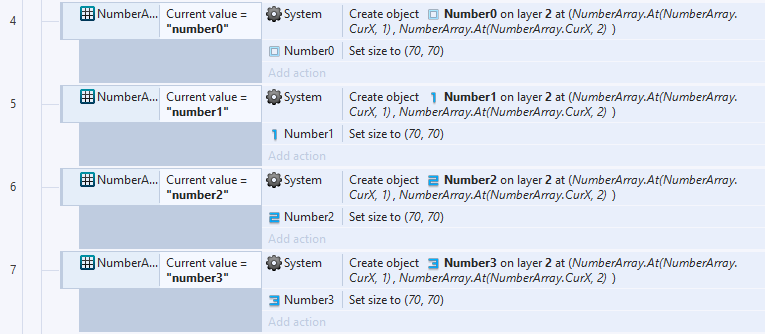
2nd to 4rd error: add 5 minutes

1st error: add 2 minutes

3.2-Change grid settings

3.2.1-Grid placement

Place Case sprite objects along your custom grid. All the Case objects placed in that grid must be the same size. After you placed them, configure the **level event** to use the size you set: *(here 70x70)*



All the Case UID must be in this order:

n+1

n+2

n+3

n+4

n+5

1

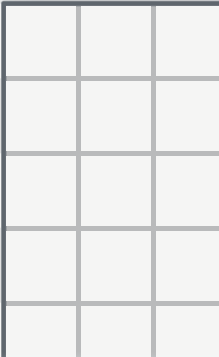
2

3

4

5

…n



UID

Now duplicate the PicrossGen layout, place on it the Case objects and replace PlaceHolder objects (place where the numbers are generated) according to the maximum possible combinations in the line (in blue):

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| For 5: | 1 | 0 | 1 | 0 | 1 |  |  |  |  | =3 |
| For 7: | 1 | 0 | 1 | 0 | 1 | 0 | 1 |  |  | =4 |
| For 9: | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | =5 |

They must be in this UID order:

Top:

|  |  |  |
| --- | --- | --- |
| Line 1 | Line 2 | Line 3… |
| …x+n | …x+2n | …x+3n |
| x+2 | x+n+3 | x+2n+3 |
| x+1 | x+n+2 | x+2n+2 |
| x | x+n+1 | x+2n+1 |

Left:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Line 1** | …x+n | x+2 | x+1 | LUID+1\* |
| **Line 2** | …x+2n | x+n+3 | x+n+2 | x+n+1 |
| **Line 3…** | …x+3n | x+2n+3 | x+2n+2 | x+2n+1 |

*\*Last UID from top PlaceHolder objects (for example LUID+1=x+3n+1)*

3.2.2-Custom Grid Events

3.2.2.1-Custom Grid Picross generation

3.2.2.1.1-Variable configuration

The Picross generation engine guess absolute other instance UID by using a relative UID and calculate others from it.

To configure them, open PicrossGenEvent and set the first PlaceHolder UID (the smallest value) in PlaceHolder1 variable, and set PickVar and OriginalPickVar with the first Case UID value.

**Warning:** Make sure the LUID+1 is correct or numbers will be generated wrongly!

And configure also the size VPanel spawned during the selection according to Case size:



And also here:

