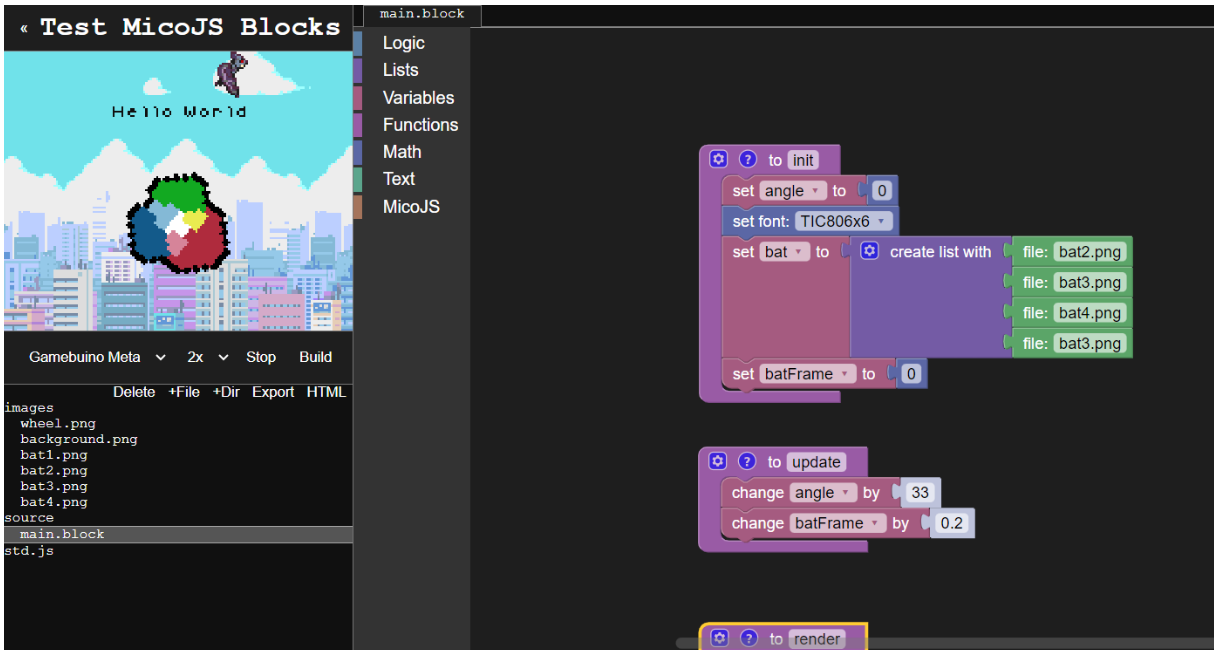
MicoJS Blocks

Introduction and start-up

MicoJS Block is a programming language designed to help you learn to code easily, like other blocks languages such as Scratch, which was designed by MIT. It's one of those languages where you don't need to type code.

The program is written using a visual interface in where blocks are connected each other and their parameters are adjusted to obtain the desired result.



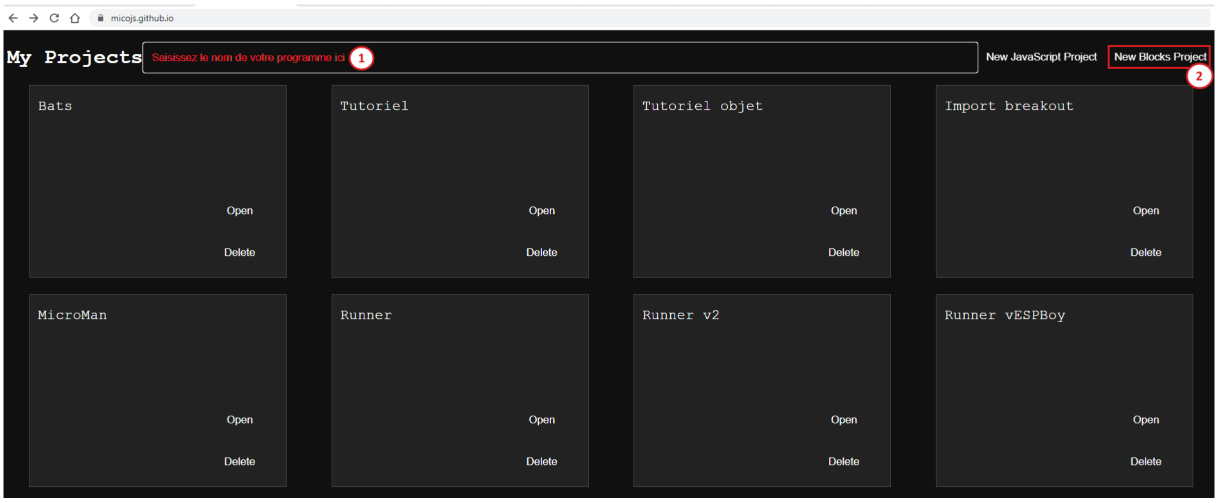
```
images
  wheel.png
  background.png
  bat1.png
  bat2.png
  bat3.png
  bat4.png
source
  main.block
std.js
```

```
to init
  set angle to 0
  set font: TIC806x6
  set bat to create list with file: bat2.png
                              file: bat3.png
                              file: bat4.png
                              file: bat3.png
  set batFrame to 0

to update
  change angle by 33
  change batFrame by 0.2

to render
```

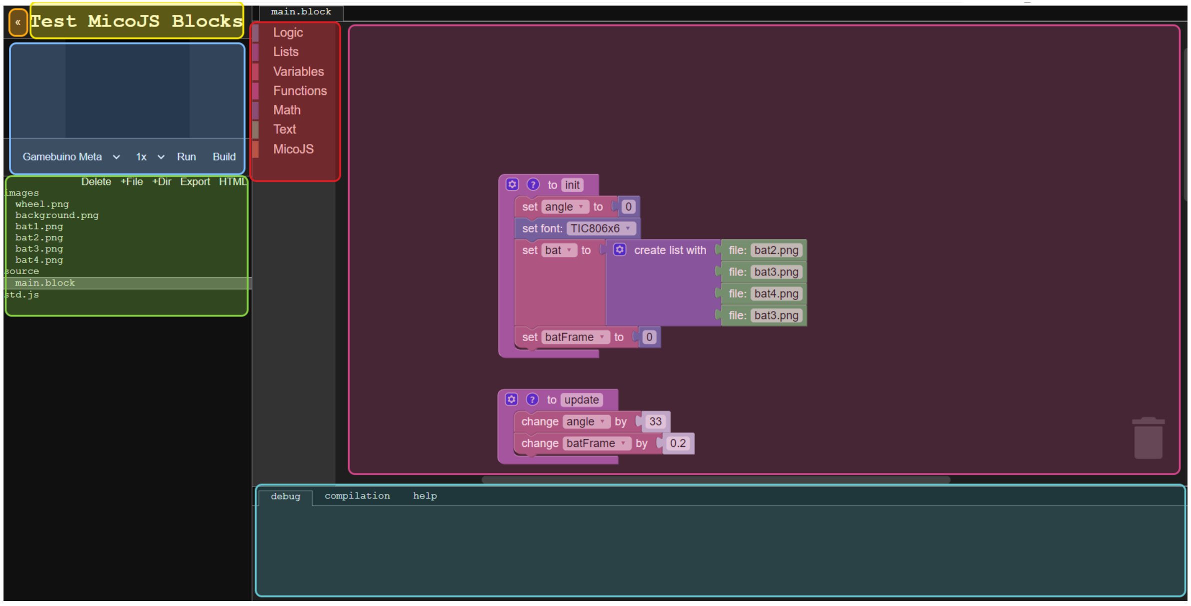
To start MicoJS, you can use this URL on your Chrome browser: https://micojs.github.io/



You can then enter a name for your project (the name of your program) (In 1 on the screenshot above) then validate by clicking on "New Blocks Project" (In 2 on the screenshot above).

Interface presentation

The interface is divided into several parts, and I've highlighted the blocks by grouping them in a colored block.



```
images
  wheel.png
  background.png
  bat1.png
  bat2.png
  bat3.png
  bat4.png
source
  main.block
std.js
```

```
to init
  set angle to 0
  set font: TIC806x6
  set bat to create list with file: bat2.png
                              file: bat3.png
                              file: bat4.png
                              file: bat3.png
  set batFrame to 0

to update
  change angle by 33
  change batFrame by 0.2
```

Orange block: , this arrow takes you back to the project selection page (MicoJS programs)

Yellow block: it's just the title of your program

Blue block : Here you can see visualization of your code work. You can choose the destination platform from the following: Gamebuino META, Galactic Unicorn, Pokitto, ESPboy, PicoSystem, 32Blit. You can choose a magnification factor, run the program, which is then displayed in the scene, or compile it. Some options may vary depending on the hardware selected.

Pour simplifier ce tutoriel, nous partirons du principe que notre code sera à destination d'une Gamebuino META.

Green block: This is the file management section. You can view all the files included in the project, add or remove them, and export the entire project as a ZIP file or HTML page. This is where you'll also find your sprite images, for example.

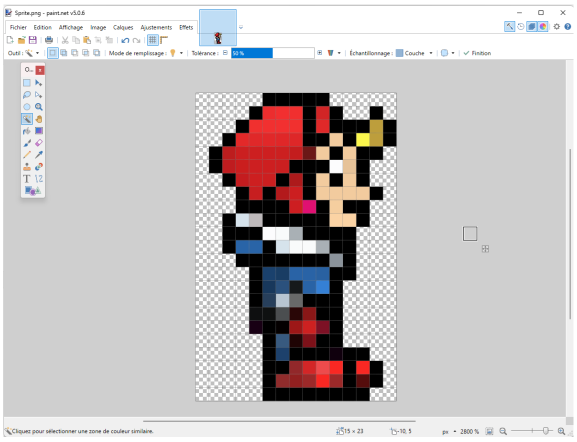
Red block: This is the area for selecting a block to add to our program. Blocks are grouped by theme, but we'll go into more detail when we create our first program.

Pink block: This is where we assemble our code blocks. This is the working area.

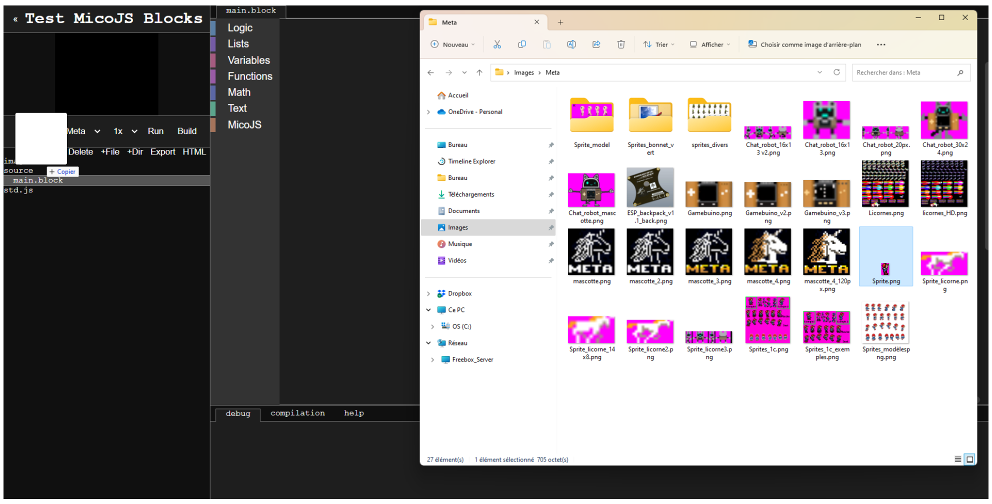
Cyan block: This is the console, where code debugging messages and compilation results are displayed. There is also a tab for easy access to MicoJS help.

Worksheet 1 : Using the interface to add and display a sprite

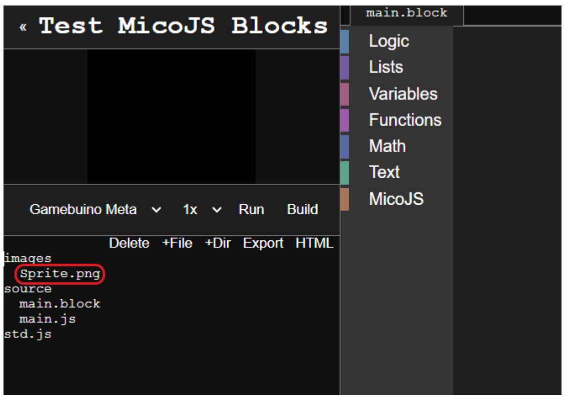
The sprite is an essential graphic resource in our Blocks MicoJS program. It is one of the elements that will be displayed and that will either be controlled by the player in the game, or by the program to interact with it or just to look pretty, but in all cases, it will contribute to the rendering of the code.


Sprite.png - paint.net v5.0.6
Fichier Edition Affichage Image Calques Ajustements Effets

The sprite is basically an image. It is created using a drawing program (Paint in Windows, Aesprite, Gimp, Paint.net, etc.), then the file is dragged from the file explorer into the "images" area.



The sprite is then displayed in the list of resources available for our project



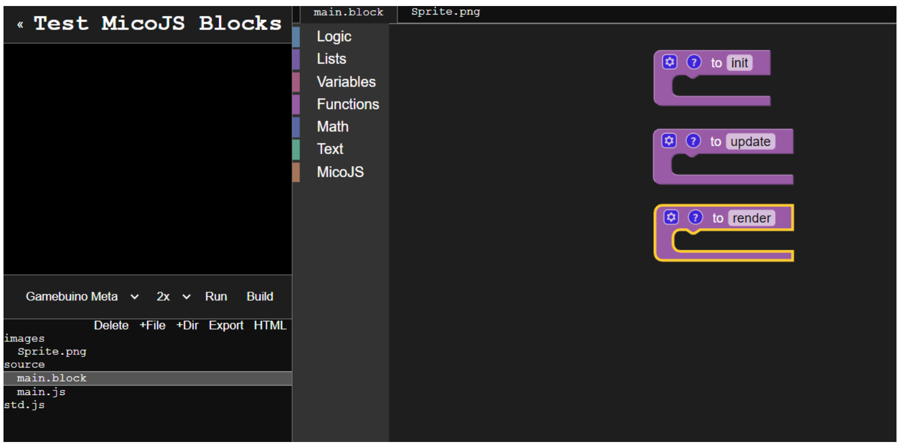
```
images
  Sprite.png
source
  main.block
  main.js
std.js
```

Now it's time to display our sprite.

To do this, let's start with our 3 basic blocks:

- init
- update
- render

These 3 blocks are essential in a MicoJS Blocks program. They correspond to 3 fundamental MicoJS functions:



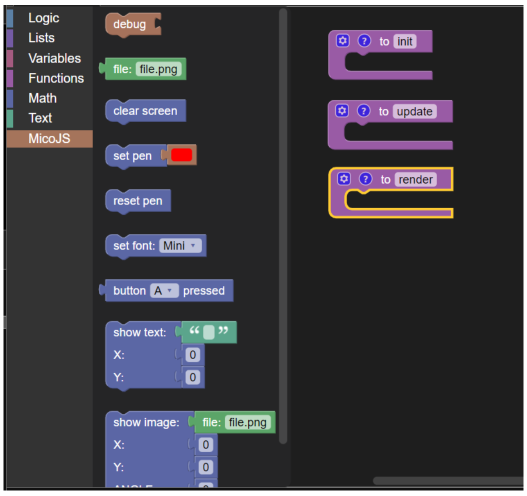
- `init` : The init block is called automatically when the game starts, once everything is loaded and ready to use. Resources and screen mode information may not be available before this point, so init is executed only once, but always first, before the other 2 blocks are called in a loop indefinitely (well, until you stop the program somehow...) ...
  Init is used to initialise the programme: to define everything that needs to be defined before running it.

- `update` : The blocks in the update block are executed in a loop. This is where we check changes to the inputs (the state of the buttons for example, the passage of time and its consequences on the state of the various objects in our program) and where we update the program. This block can be called several times during the display of an image, so nothing should be displayed in this block (i.e. no image or text should be displayed, for example). Here, we calculate the position of the elements, and perform checks (collisions, button presses, game state changes: paused, end of game, etc.), but we'll go into more detail about these things as we progress.

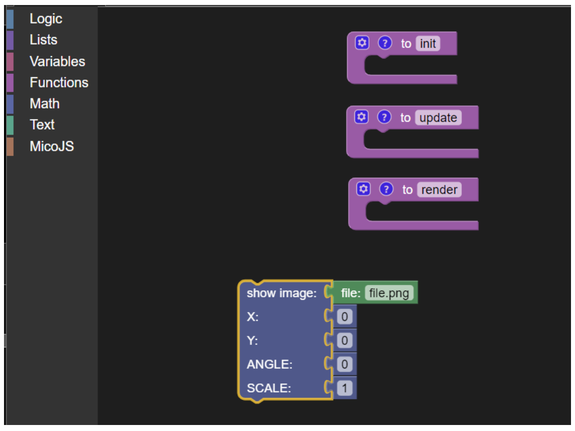
- `render` : This block is used for drawing on the screen. No program variables should be updated here. As all the calculations are already done in the update block, executing this block is normally quick and easy.

This is where we're going to add the block that will display our sprite

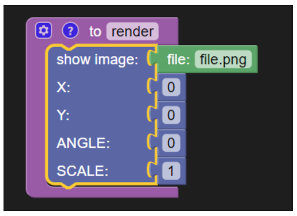
To do this, in the red block selection area, we're going to click on "MicoJS".



The blocks in this section are displayed. Click on the show image block and, holding down the left mouse button, drag it into the render block.

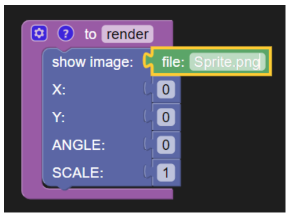


When the block anchor turns yellow, release the left mouse button, you hear a click and the block is added.

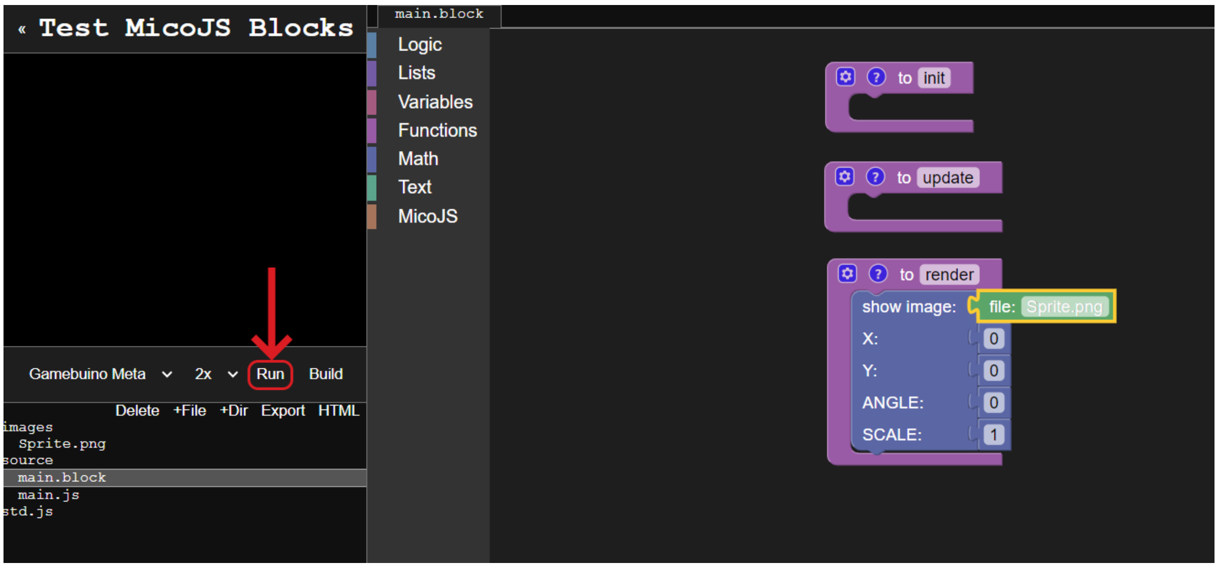


We've added a 'show image' block but it hasn't been configured yet. For the moment, it has default values that we'll need to adjust to suit our needs.

To do this, click on the values you want to change. For example, for me, my sprite is not called file.png but Sprite.png (I wonder where I got such a complicated name from? 😊)



So I'm changing the value of this parameter in the block. You can also play around with the other values if you want to see the result by clicking on "Run".

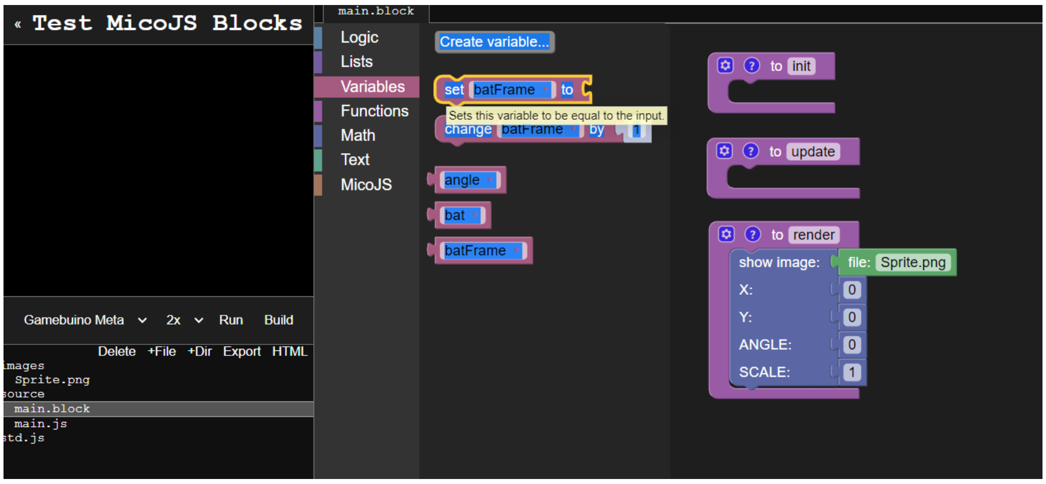


And normally, your sprite is displayed. You can change the values of X and Y. To test, click Stop, then restart the program after making the changes you want.

Nice, isn't it? Well, after that, there's not much point if nothing happens, so we're going to add some instructions to the update block to create movement. To do this, we're going to add another fundamental element: a variable.

A variable is a space in memory for temporarily storing data. It can be the current score, the position of a character, text, etc. When we stop the program or switch off the console, the value is lost.

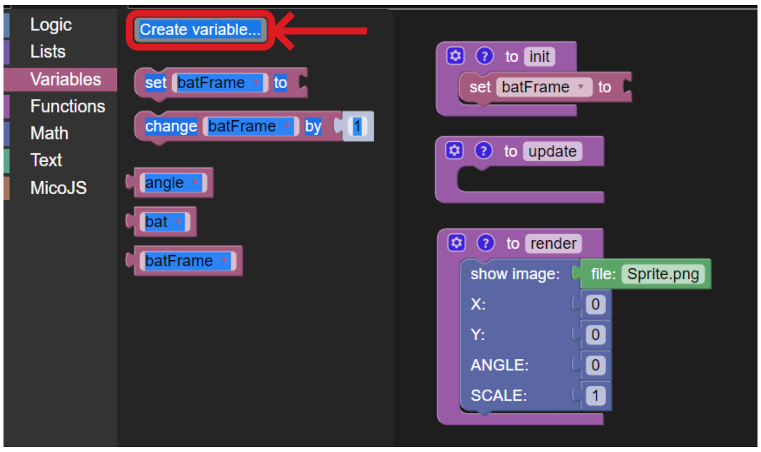
You declare it and give it an initial value, using the set block in the Variables section of the blocks.



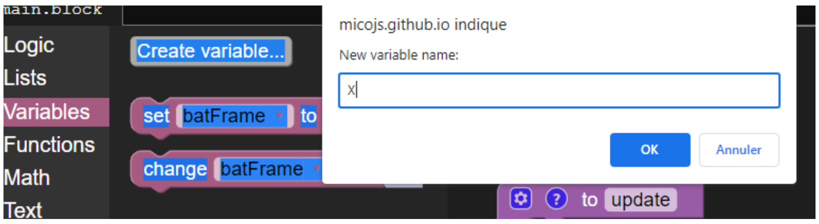
So we take this block and drag it into the init block to initialise our variable when we launch the program.

A variable has a name, and here we've chosen to call our variable X because we're going to use it to move the sprite horizontally, and therefore to modify its abscissa (its value on the X axis). We need a name that clearly describes what it is used for so that the code remains readable. If you use variables called a, b and c, you will quickly get confused when the programme becomes more complex. We'll come back to this later. For now, let's configure our set block as we wish.

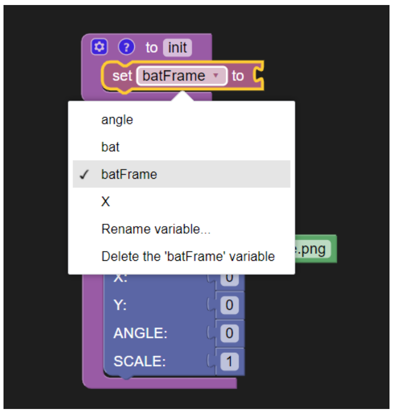
To do this, we need to create our X variable. To do this, still in the Variables category, click on Create variable



Enter its name: X and confirm with "OK".



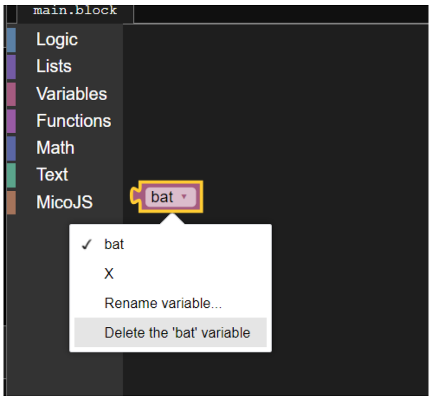
Now that the variable exists, it's available in the set block we've added. If you click on the location of the variable name, it will be available in the list:



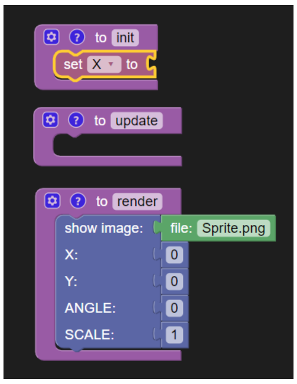
Note: You can take advantage of this to remove unused variables. For example, in my list, I don't use the angle, bat and batFrame variables. To delete them, I click on them to select them and then click on Delete.

Note that instead of creating a new variable, I could also rename one of the old variables to reuse it as I need it. This can be useful if your requirements change as you develop the programme, but we won't go into it now.

The easiest way to delete a variable is to select it in the list of variables and then click Delete in the drop-down list.

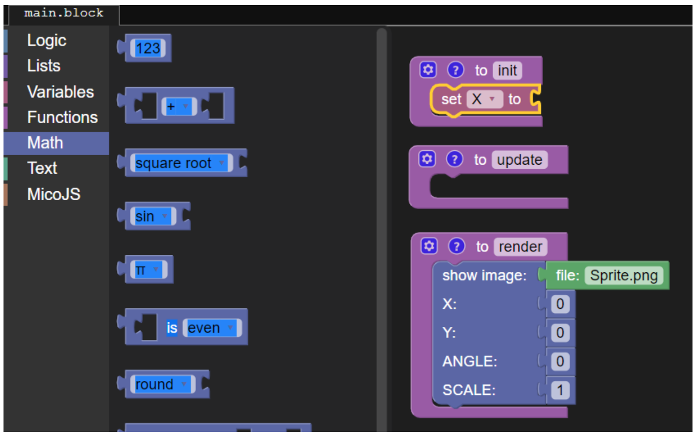


But let's start again. We currently have our block set as follows:

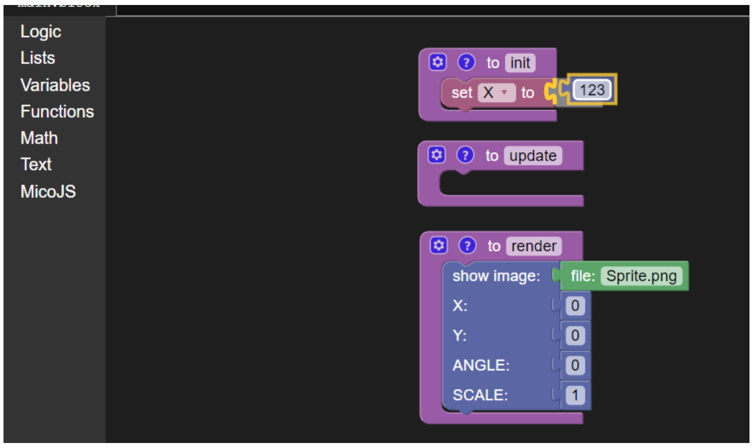


So we tell it that we want to initialise the variable to a value that we haven't yet added. We can see that it is possible to hook a block to the right of the set block. We can, for example, add another block corresponding to a variable so that X takes the value of this variable, or add a fixed value, which is what we'll be doing to initialise X.

To add a fixed value, open the Math section



Drag the first block to the right of our set block.



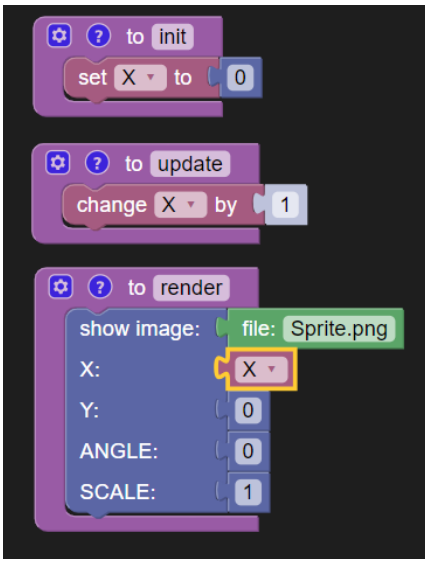
```
to init
  set X to 123
to update
to render
  show image: file: Sprite.png
    X: 0
    Y: 0
    ANGLE: 0
    SCALE: 1
```

Then change the value there to 0

Open the Variables category and add the change block so that it increases by 1 each time the loop is executed.

Finally, replace the fixed value 0 of the show image block located in the render block with the variable X.

I won't explain how to do this, you've already seen it all, but in the end you should get this program:



```
to init
  set X to 0
to update
  change X by 1
to render
  show image: file: Sprite.png
    X: X
    Y: 0
    ANGLE: 0
    SCALE: 1
```

Use Run to launch it.

Your character starts on the left side and moves forward... then exits the screen and disappears forever, or at least until you click "Stop" and then "Run" again.

This is normal, the programme only does what you ask it to. You ask it to increase X at each occurrence of the loop, and it does so. At no point do we control the fact that X is greater than the size of the screen. It therefore continues until it reaches its limits.

We're going to have to add a few things, but that'll be in a later stage. For now, congratulations, you've learned how to launch MicoJS Blocks, use its interface and create your first program. Have fun with what you've seen and do your own tests and experiments before moving on to the next page.