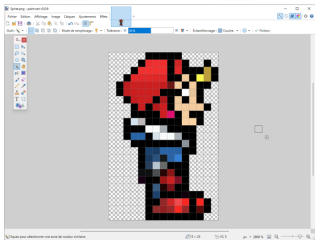
MicoJS Blocks

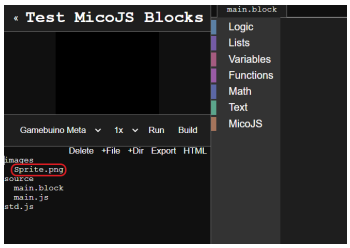
Worksheet 2 : Animating a sprite and controlling it with buttons

Let's start by repeating what we saw at the end of sheet 1: 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 him or just to look pretty, but in any case, it will contribute to the rendering of the code.



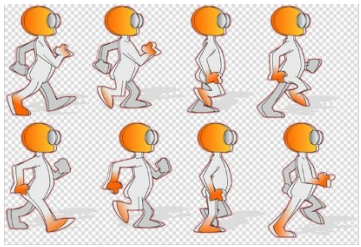
The sprite is basically an image. You create it using a drawing program (Paint in Windows, Aesprite, Gimp, Paint.net, etc.), then drag the file from the file explorer to the "images" area.

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



Well, we also saw in Worksheet 1 how to display it on the screen and how to move it using a very simple program. In our case, the sprite corresponds to a 'frame' of our character. I would say that a frame is the image of an object at a given moment in a given situation. It has a pose that corresponds to its state in that situation.

Let's take an example:

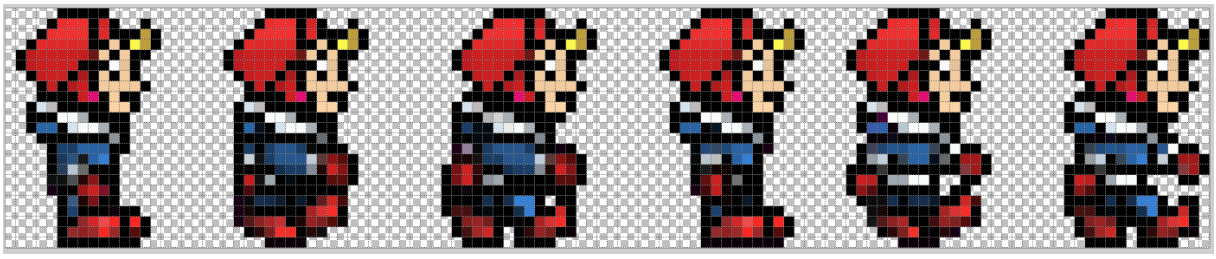


This image includes 8 frames of a character walking. Each frame corresponds to the state of the character when we want to animate it to give the impression that it's walking.

As a general rule, there's also a frame that corresponds to the moment when he's at rest and there would be another series of frames for each of the animations we'd like to add to illustrate his state when he performs other actions.

An animation is a more or less rapid succession of images that gives the impression that the action is taking place. It therefore has a variable number of steps (frames) depending on the desired visual effect, and you should avoid creating too many if you don't want to slow down the game too much or saturate your console's memory. It's up to you to decide what you want, but you can base it on the wide range of games that already exist and on the general observation that the smaller the object, the less useful it is to have a large number of frames.

Let's take the example of our character again. To make it work, we're going to create 6 stages:



We're going to save each step as an image and import them into our program. The first corresponds to our Sprite.png, so we're going to add Sprite2.png, ...Sprite6.png

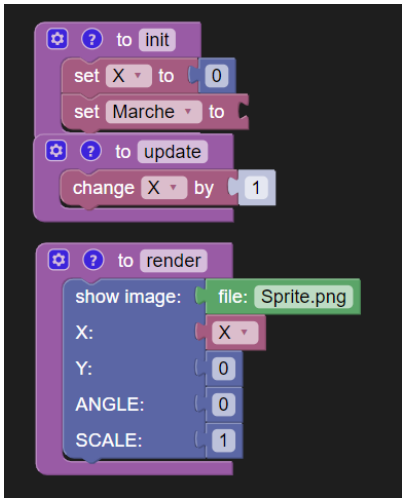
I'll put the images here for simplicity:

After integrating them into your code, you should have a screen similar to this one:

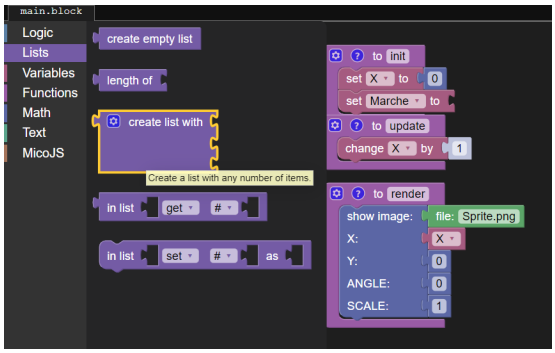
There are several ways of creating our animation. We could display the images manually, using a frame counter for example, but with MicoJS, there is a type of data that is particularly useful for this type of animation: lists.

So we're going to create a "Walk" list by adding each of the images in our animation to it.

To do this, create a "Walk" variable, then take a "set" block and add it to the Init block



But this time we're not going to give it a fixed value. Let's open the "Lists" category and take the "create list with" block and add it to the right of the "set Marche to" block.

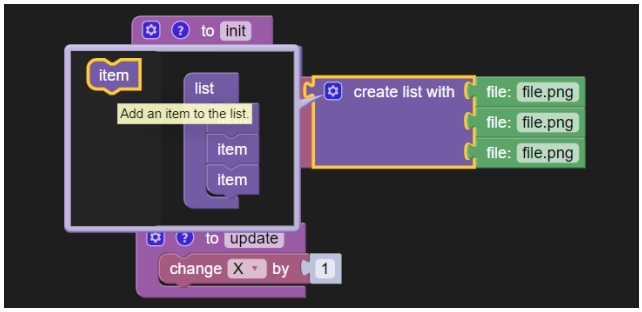


Then, just as we added the Sprite.png file to our "show image" block in the first form, we're going to add the 6 image files to our Walk list.

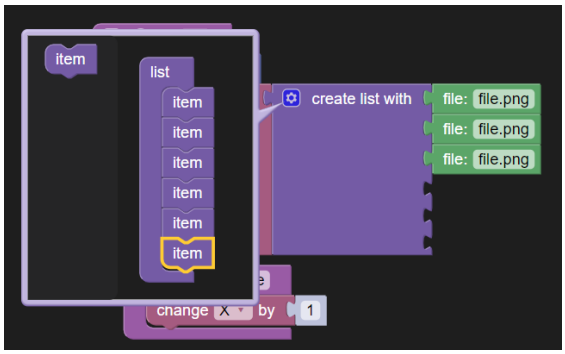
Note that by default, you can only add 3 items to your list. You'll need to add 3 other locations to your list, but as you'll see, it's very simple:

When the block is selected, click on the cogwheel

Take 'Item' and add it 3 times to the bottom of your list



You should get a block that looks like this:

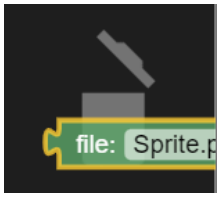


Re-click on the cogwheel to close the object settings, add the other file blocks and modify them with the names of your images, respecting the order so that the animation is logical.

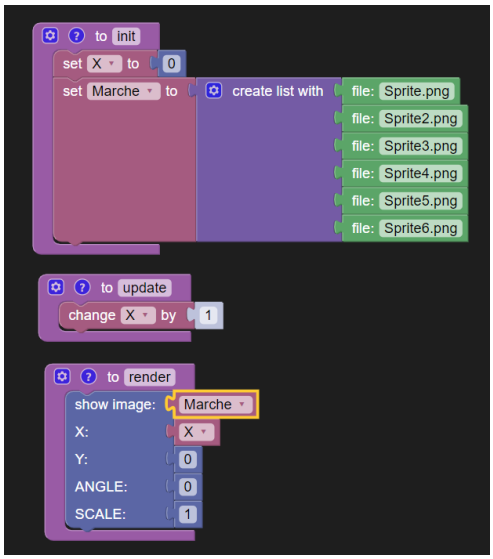
Note: When you modify a file name, click on "Tab" to go directly to the next item (the next file name in the list).

That's it, our list is ready. We're going to modify the "show image" block so that it no longer displays the sprite: Sprite.png in a fixed way but instead displays our "Walk" animation.

To do this, delete the "file:Sprite.png" block by dragging it onto the bin at the bottom right of the editing area of your program:



Then open the 'Variables' category, take the 'On' block and put it in the place of the block we've just deleted.



Launch your programme and see what happens...

Nothing happens...

This is normal.

To use the list, we need to tell it that we want it to display a particular image from the list.

To do this, we're going to create an index. An index is a variable that will run through our list from start to finish.

Note: a list starts at 0. The first image therefore corresponds to an index of 0 and the last image to 5 in our case (number of elements - 1).

You know how to do this, so I won't explain it again, but you're going to :

- create an Index variable
- initialise it to 0 in the init block
- add 1 to Index in the update block

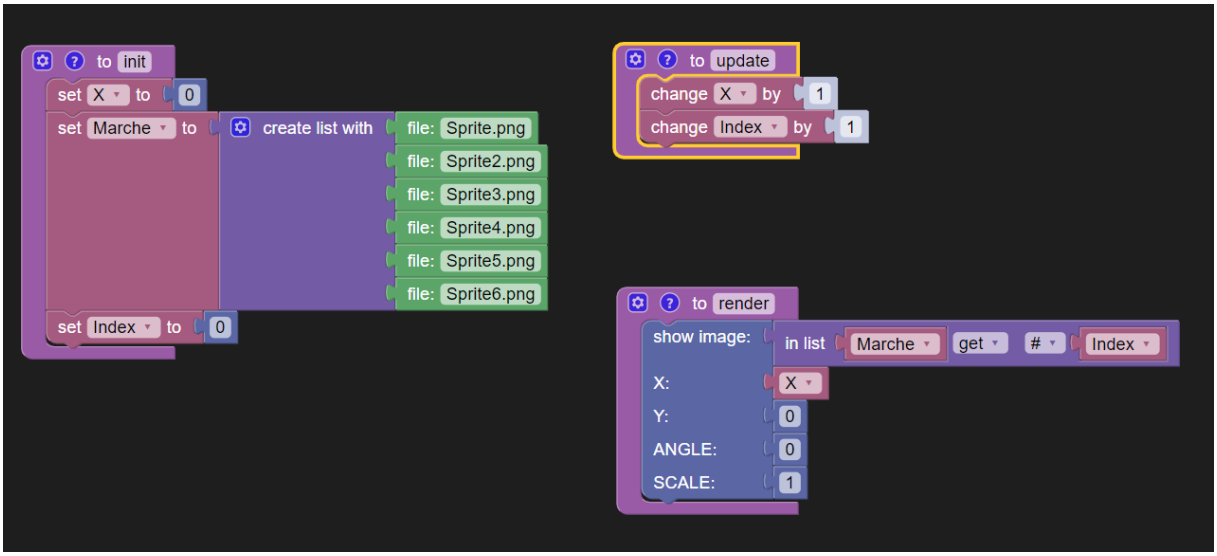
As for the rest, we'll continue together because it's all new.

I'll leave you to make these initial changes before continuing. It should be getting more and more intuitive for you, shouldn't it?

Well then, let's get on with it.

Move the last Walk block we added and put it somewhere in the workspace.

In the 'Lists' category, go and find the 'in list' block and add it in place of the Walk block that we've moved, then in the first location of this block, put the Walk block that you've dropped on the desktop back and in the second location add the Index variable.



Yes, I've reorganised my blocks to make my code easier to read. Organise your workspace as you like so that you can find your way around as easily as possible.

Run your programme to see what happens

The little man moves forward and then disappears...

Why does it do that? I'll leave you to think.

Well, we use Index. It is initialised at 0 and then incremented by 1 at each loop. But our animation only has 6 slots. When Index reaches 6, the program doesn't know what to display... so it doesn't display anything... logical... a program only does what you tell it to...

So we're going to add another fundamental element to programming: a test.

We're going to ask our program to test whether the Index value is greater than 5, and to reset it to 0.

To do this, open the "Logic" category and drag an "if" block to the end of the "update" block.

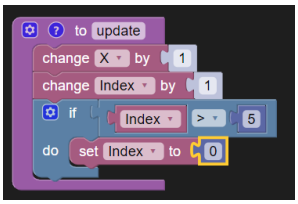
For the top parameter, let's return to "Logic" and take this block:

Set it to run the "Index > 5" test (To change the = to >, just click on it and choose the test you want).

For the parameter below, we'll use the set block to reset Index to 0.

Note To obtain the block with the value 0, you can go and find it in the category and set it or simply right-click on the block we've already added and choose 'Duplicate' from the drop-down list. This option is very practical and we could have used it several times by now. But I'm not going to give you all the tips at once... 😊

Once you've done that, you should get this block:



Start your programme.

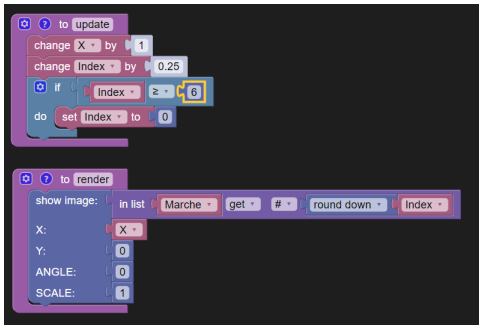
That's better, isn't it?

Good, good, but the animation is too fast for me... what can I do?

No, no, I'm not giving you the solution straight away, just try...

Yes, you need to change the Index increment value in the update block. But that's not all, we also need to change our test to check if Index >= 6 so that the 5th image is displayed all the time it should be and use the round mathematical function. I'll leave you to make the changes.

You should get this:



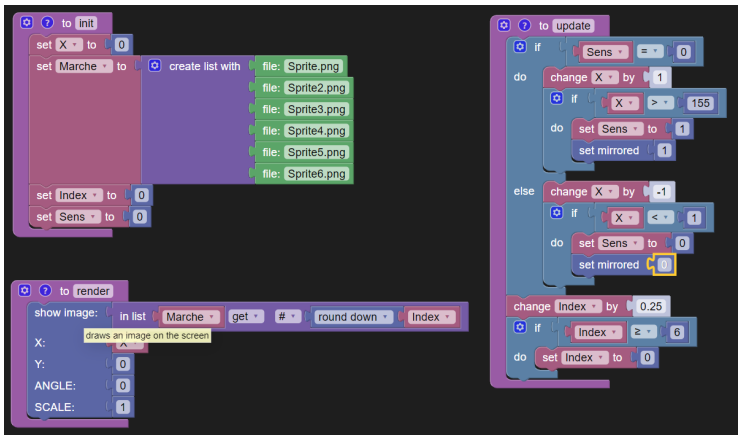
Now we want our character to change direction and go backwards when he reaches the right edge of the screen...

So we need to test the value of X and we're going to add a variable that tells us which way we're going. When the direction is 0, we'll move to the right until we reach the right edge. At that point, we'll change the direction (to one) and use the set mirrored block, which flips the sprite horizontally.

Try to do this on your own by looking at the interface options.

It's always a good idea to look around a bit, you'll always learn new things and what's more, you'll remember better.

Whether you succeed or not, you should end up with a program roughly similar to this one:



And that's it, no character remains on the screen and you now know how to use the tests.

But there's still one important thing we need to know how to use: how to manage the

Let's control our character using the keys.

Our aim is to modify our program so that the character moves to the right if you press the right button and you haven't yet reached the right edge, and of course moves to the left if you press the left button and you haven't yet reached the right edge.

We're going to use the "button" block found in the "MicoJS" category and modify our conditions a little to allow movement only when the key is pressed and movement is authorised.

Try it yourself before looking at the solution below:

That's the end of this file, but with what we've seen together, you can already do quite a few tests.

For example, you could try to create an animation that occurs when button A is pressed, and another when button B is pressed. You can add constraints, such as not being able to do anything other than wait for the action to finish when button A is pressed. Experiment, test, set yourself goals and have fun, you've already got quite a lot. In the next sheet, we'll look at managing 'shots' and collisions...