INFORMATIONS

GitHub link of the project : https://github.com/luddoz-c/war-of-blobs

Please feel free to open an issue if you detect some problems, want to suggest any improvement of if you want to give us some advice.

CHANGELOG

20/03/19

Creation of the class blob that we're going to use for every new blob that is created. This way, it will be easier to get its position, to change it, to edit its name, get the number of blobs on the grid...

26/03/19

Creation of the `draw_grid` function to create the grid to be printed on the console.

We defined 2 grids :

- The first one is just the interface to be printed thanks to our previous function
- The second one is where all the informations are stored (positions of blobs, empty cells...). It is is a list of lists, containing empty strings where there are no blobs and a `blob` object where there is one blob.

Each blob has 2 names: one for the interface, which is just `b + its number + its weight:` (ex : `b1(45)`), and the other, which is its actual name that is going to be generated randomly, like `Mu`, `Ga`, `Yo`... but not printed in the console.

A blob has also a color, defined as a tuple `(r, g, b)` and a size (which is actually its weight). The `r`, `g` and `b` values are between `0` and `1`, with at most 2 decimals.

27/03/19

Definition of the `changePos` method, which is going to change the value of the blob's position (`self.pos`), edit the grid (list of lists) to move the blob to the right position (erase its old position and define its new one), and also check whether or not there is already a blob to the position where we want to move the blob.

Also, definition of a function that is going to generate a blob randomly (name, position...), and that checks whether or not there is already a blob at the generated position for the blob. Also, we created a function `generate_blobs` that uses the previous function to generated several blobs.

31/03/19

Creation of the `next` function that simulates the next step of the simulation. However, this function is **TEMPORARY** as it is random (which is not the purpose of this simulation).

Addition of the colorization of the blobs : for that, we used the module `ansicolors`, which we imported as `colors`. Now, every blob's weight is colorized in the console, depending on the blob's color attribute (that we had to convert from values in `[0:1]` to values in `[0:255]`)

04/04/19

Creation of the method that checks the priority of each blob, in order to know how to merge them.

08/04/19

Add the other diagonal priority (from top right to bottom left).

09/04/19

- Fix the `check_blobs` function (priority bugs). Sometimes the blobs either didn't merge or merged with the wrong priority.
- Also other fixes on the `check_blobs` function. (2 commits made).

10/04/19

Several changes :

- Renamed `change_pos` method to `move_pos` : the position passed as parameter will be the relative move from the blob's position, not absolute anymore.
- Add `move` method that moves the blob towards the biggest one and, in case there are several ones (same size), towards the nearest.

HOW TO USE ?

You can try these commands :

- `generate_blob(W)`, with `W` being a positive integer, to generate a blob of weight less than `W`
- `generate_blobs(n, W)`, with `n` and `W` being 2 positive integers, to generate `n` blobs of weight less than `W`
- `print(draw_grid())` to print the grid
- *(i)* Generate a new blob `b`, *(ii)* print the grid, *(iii)* then change the blob's position (`b.pos = (x, y)`) and *(iv)* print the grid again to see the changes.