New methods for OC_manager

Implementing a new method for OC_manger can be done without touching the main code by adding files in different folders.

This document describe the procedure, knowledge of GCODE and R is necessary.

If the step can be done by a succession of GCODE and options can be supplied in a simple table, a method can be created.

In some case, an application table (`appli_tabli`) is necessary, for sample application and derivatization steps. This is also taken care off in the code but the table will be static with only the number of rows changing.

The `Documentation` method is the only exception because it need to use the shell to take pictures wil the rpi camera.

As example, a step of plate heating will be implemented.

tables folder

In the table folder, a CSV file must be created with the following constraints:

- The separator is semi-colon.
- The name must finish with **.csv**.
- Column names must be **Option** and **Value**.
- Row names must be present, the names have no influence though and are here to inform the user.
- The **Option** column contain the options names that the method function will later catch (more later).
- The **Value** column contain the default values for the options.
- If an application table is needed, an option named `nbr_band` must be set in the table.

For plate heating, two options are needed: temperature and time. The file will look like that:

```
Option;Value
Temperature (°C);Temperature;100
Time (min);Time;10
```

eat_tables folder

In the eat_tables folder, a R script must be created:

- This R script contain a function that will take a step object and update it, *i.e.* modify the `gcode`, `info` and `plot` elements.
- The R script must have the same name as the CSV file
- The R script must return the step object

For plate heating, here is what it looks like:

```
# eat_table_heating
function(step){
  # eat_table_heating function V011,24 february 2017
  # home, start heating, and wait the needed time, then home again

  # extract the needed information
  table = step$table
  Temperature=table[table[,1] == "Temperature",2];Time=table[table[,1] == "Time",2]

  # make the gcode
  gcode = c("G28 X0; home X axis",
            "G28 Y0; home Y axis",
            paste0("M190 S",Temperature," ; set temperature, wait to reach, use M140 to set and go to t
            paste0("G4 s",Time*60," ; time wait in secondes"),
            "M190 S0 ; set temperature off",
            "G28 X0; home X axis",
            "G28 Y0; home Y axis"
  )

  # make the new plot
  plot_step=function() {
    plot(c(1),c(1),type="n",main="No plot for heating step")
  }

  # replace the elements in the list
  step$gcode = gcode
  step$plot = plot_step
  step$info = paste0("Heated as at ",Temperature, "°C for ",Time*60, "sec")
  return(step)
}
```

Server side

Everything happens in the server_Method.R file.

When the user click on the `+` button (`input$Method_step_add`), the selected step is added to the Method list. This step is also in the form of a list with different element, *i.e* `table`,`eat_table` (the function we created),`plot`,`gcode` etc... Optionnaly, an other element named `appli_table` is added.

When the user update the step (`input$Method_step_update`), the step is fed to the `eat_table` function.

Additionnaly, it is possible to delete a step but also save and load the method as a whole for later use.

```
Method$l[[step]] = Method$l[[step]]$eat_table(Method$l[[step]])
```

Finally, when the user execute the step (`input$Method_step_exec`), the gcode is written in the file `gcode/Method.gcode` and launch (`main$send_gcode(Method_file)`).

UI side

On the UI side, still in the server_Method.R file, the user select a step with the radioButton `input$Method_steps`, the server will render tables, plot, gcode and info corresponding to this step. When one of the button will be clicked, it will just concern this step.

There is a known bug when step with and without appli_table are mixed, the step without are not accessible anymore, in this case, create separate methods.