console vs "virtual environment"

We dissociate the console and the “virtual environment” as the console is the GUI interface that controls blenderVR. The “virtual environment” is the part of the simulation that runs on each node inside `blenderplayer`. To simplify, the console is run by the user, use PySide but cannot import bge python module whereas the “virtual environment” is run by `blenderplayer`, don't have any GUI and can import bge python module.

Console

The console is the GUI of blenderVR. It allows you to choose the configuration file(s) (`configuration` tab), the screen set to use and the simulation file in any tab. In the Run tab, you can start and stop blenderVR and have the log of the console.

You can load the console by invoking *`blenderVR path`*`/blenderVR` (clicking on it or running from a console). You can also add `blenderVR` inside a *`bin`* folder that is included inside your PATH environment variable.

By default, the console does not “known” anything. You have to manually set configuration file, active screen-set, simulation file … However, it stores these relevant informations in its internal data store path (see above). So you have to set these informations the first time you run blenderVR and they remain active (across different running) until you change it with the GUI.

Configuration file(s)

You can specify the XML file inside the `configuration` tab. Don't forget to click on `> Load configuration` to ask blenderVR to read the configuration file (and store it inside its internal data file) ! You should go to run tab and select debug inside the log window to see if there is bug inside your configuration file.

Active screen set

You can choose any screen set that is defined inside your XML configuration file. You also must click on `> Load screen set` to make it active (and register it for further blenderVR usage). The current active screen set is displayed on the right.

Simulation file

Here, you must select the `.blend` file you want to load. For the beginning, you should try the `simple.blend` that resides inside the `samples` folder of blenderVR.

Start/Stop

When everything is defined you can try to start/stop by going to the Run tab. Have a look at the main log window below the Start and the Stop buttons.

Debug window per screen

You can also have a look at the per screen log window : menu `Windows` → `Screens` and select the screen that you want to debug. We suggest, at the beginning, to debug your XML configuration file, to set it to debug mode and activate `Standard output` and `Error output`.

Standard/Error outputs

They will display `stdout` and `stderr` of the instance of blenderplayer. Thus, you will see if there is a bug while running it. When blenderplayer runs correctly, you should disable these options.

Log level

The log level is usefull when blenderplayer runs properly. It can display errors of your processor file.

Daemons

The console use one daemon per screen. The daemon is a python script that:

- connects by network to the console and interact with it ;
- start the instance of blenderplayer (for the “virtual environment”) when required ;
- catch blenderplayer's `stdout` and `stderr` to address them to the console if requested ;
- kill blenderplayer if the “virtual environment” don't gently stop on console request ;
- …

In other words, the daemon manages blenderplayer. It runs on the computer that will run the blenderplayer instance.

Under Linux, this daemon becomes a real UNIX daemon (fork, close input and output …).

The daemon script is included inside blenderVR ; you don't have to tweak it.