Instructions for developers: EARLY2015 git usage

-------------------------------

There are two repositories

- **C:\EARLYrepo2015** - EARLY program itself, main repository. Contains all necessary functions with setup, GUI utilities, procedures, etc.

- **C:\Early\_StimDefLeuven2015** - auxiliary repository that contains majority of stimuli. It is an addition to the stimuli stored in C:\EARLYrepo2015\vs10\StimGen\StimDef. Look at StimdefPath, directories are added there. Setup for stimuli path stored in 'C:\EARLYrepo2015\setupInfo\Experiment-only\_LocalStimuli.PNXsetup'.

Having both of them next two branches:

- **Master:** stable branch, all versions should be a working version. Used for experiments. Developer does not work on it. He/she should check it in case any change has been introduced there by someone else (ideally this shouldn't happen), and make sure that develop branch is updated with the changes if any.

- **Develop:** development branch where developer introduce features, fix bugs, etc. A change in this branch must be tested before being merged to master. For this end, ask a researcher to run/simulate a couple of experiments, being aware that he is testing a new version. Developer must take care of setting up EARLY so researcher uses the correct branch. This is only needed for changes that might affect the stimuli generation; trivial changes (such as labels) may be merged directly.

There is a procedure to be followed by researchers and developers in order to make the version control system work.

- Researcher instructions:

1. Run " gomaster " on matlab before running the dashboard for the first time when starting a set of experiments.

The output must be:

"  
You are now in **master and master** branch.

C:\EARLYrepo2015>cd C:\EARLYrepo2015

C:\EARLYrepo2015>git pull

Already up to date. / Updating…

C:\EARLYrepo2015>cd C:\Early\_StimDefLeuven2015

C:\Early\_StimDefLeuven2015>git pull

Already up to date. / Updating...

"

**If the first line is not "master and master",** ask a developer if (s)he is at the lab.

If not, open github desktop and try to change manually to master in both EARLYrepo2015 and Early\_StimDefLeuven2015.

If you get an error window, note it down This means the branch hasn't been switched. This shouldn't happen, but if it happens it would be a strange error that should be looked at carefully. Experiments should always be done on master branch so it is recommended that no experiment is done on develop if this error happens.

1. When there is no EARLY developer and an **unexpected behavior makes EARLY unusable**: if that error could be potentially caused by the last system update, double click the file "RevertToPreviousEARLY". This will undo the last change introduced. Before doing so, note down the error: the setting that made it happen, in which menu, what were the steps that lead to the error... and notify the developers about the situation.

If it is an error that occurs at a function or stimulus that hasn't been modified recently, running the aforementioned file won't fix it most probably because it was introduced before.

1. If the developer is present and such an error occurs, notify him/her.
2. **If a small error is found that can be fixed by the researcher**, introduce the necessary change and notify the developer about it. (S)He will take care that this change is correctly applied to all computers and branches.

- Developer instructions:

1. **Check consistency between branches**: When starting to work, check the status of master branch. This means checking for uncommitted and last committed changes different than "merge branch develop". If there is any change make sure it is a desired change and it does not introduce a bug (maybe ask around).

**Pay special attention to "Revert commit SHA".** This means a researcher faced an unexpected behavior of EARLY and "reverted" last commit, using therefore the previous EARLY version. You should have been notified of this action, if not, find out who did it and why. If there is a bug you should fix it on develop.

If the changes are alright, merge master into develop.

**Hint**: Committed and uncommitted changes are better visualized by GitHub desktop. For uncommitted changes 'git status' on command prompt also does the job. (if developer was the last person using the machine this point does not really apply).

1. Always **work on branch develop**.
2. Check for **'Temporary commit' on develop branch**. It might have happened when a researcher checked out master. See section 5 for details.
3. Check that **local develop is up to date with the remote** (fetch, pull).
4. **Commit your changes daily**: After finishing a session of work there cannot be changes uncommitted in order to be able to checkout to master branch. Commit your temporary work as a temporary commit if it wouldn't be a proper/definitive commit so you can keep working on it and make meaningful commits later on.

Do not 'git stash' your changes because it is repository dependent rather than branch dependent. Committing and then resetting achieve the desired behavior.

**HINT:** You can use file tempCommit.bat to create a commit with all edited and new files with the message "Temporary commit". **This file acts on the current branch.**

Before starting the next session, if needed, remember to reset the temporary commit. You can use file resetLast.bat to undo the last commit but keeping the changes on your working tree. **Check that resetting is necessary before doing it** (on github desktop for example).

**HINT**: Keep in mind that when a researcher uses EARLY, it will be used on branch master. Therefore if there are uncommitted changes on develop branch, they will be committed automatically as "Temporary commit" when a researcher switch to master branch, so you might want to check it before start working and undo it when resuming your work.

1. When a significant commit has been done, always push it. If possible, pull the commit on the other machines.
2. Test changes: as explained in develop branch introduction, changes must be somehow tested before merging them into master. There is no testing suit so it should be done manually until it is implemented (if ever...). Check it with a researcher, clearly stating that (s)he would be running EARLY for an introduced change, and what change it is so (s)he is aware if he would like to use the data.

**Merge develop into master when tested. Push.**

1. Make sure that **machines are always synchronized**: pull and push when significant commits have been done (no 'Temporary commit'). Extremely important when merging into master.