Tommy's Scriptinator 3000 TM

Main Purpose

Tommy's Scriptinator 3000 TM aims to provide an interface between code line programming and graphical user interfaces. This can become especially handy in science, where users in general are less skilled in using multiple programming languages. E.g. students entering the field are often confronted with analysis scripts that are stitched together by another person and spend often a significant amount of time figuring out the workflow. This is where Tommy's Scriptinator 3000 TM slips in.

Scripts can be prepared such that main I/O variables and different additional parameters can be defined in a classical graphical user interface (GUI). Furthermore those scripts can be daisy chained to create analysis pipelines.

Once a pipeline was created it can be shared with anyone using Tommy's Scriptinator 3000 TM. Thus, making it the ideal solution for sharing scripts and communicating dependencies.

main functions

When starting up Tommy's Scriptinator 3000 TM the main playground shows up with the main control buttons located at the top centre of the panel.

main button panel:

- "/" removes all scripts from the playground
- "S" saves the current pipeline to hard drive using the .pipe extension
- "L" loads pipelines using the .pipe extension
- "P" creates a runnable pipeline (will be explained later)
- "!" calls the "About" information window
- "?" shows this document
- "X" closes the application

creating a new script

A new script can be created by double clicking somewhere on the playground. A round icon having a random colour will appear at the respective location. This Icon represents your script. At the moment of creation it will be a default "unknown" script. Double clicking the script lists all features of the script and

allows for modification (will be explained later). Further the script icon can be dragged to any location desired.

connection two scripts

Two scripts can be connected by single clicking a script providing the *from* connection and afterwards single clicking the script to which the first script is connected *to*. When repeating the exact same procedure the connection is removed. Internally each script stores all connections it has *from* other scripts (rather than *to*). Outgoing connections are symbolized by the thicker end of the connection line whereas the opposite is true for incoming connections. In other words the connection goes from thick to thin.

Style Sheet

The default Scriptinator style sheet was created for being used with *Bash* scripts. However by creating the respective style sheet for a different language would make the program compatible to that as well. All functions are set up in a general way such that according to the style sheet the environment is built.

Note that the StyleSheet includes parameters for skipping confirmation dialogs. Make sure that this settings are defined in accordance with the desired behavior.

modifying script properties

If a script was created, it accepts double clicks to expose the properties window. Depending on how the style sheet was set up, the default or template script will be displayed.

The different sections of the properties window contain:

Input All input variables (to be used by the script) are defined here. Thereby the style convention is derived from the settings in the style sheet. Variables defined here can be e.g. used later in the script. Hence file paths or often changed parameters are good candidates for find their places here.

Output All input variables (to be produced by the script) are defined here. Thereby the style convention is derived from the settings in the style sheet. Variables defined here can be e.g. used to provide an output path for the script to save results.

Misc All miscellaneous variables (to be used by the script) are defined here. This could be parameters or an additional waiting time (if that makes sense within the pipeline)

General This section contains information that is internally used by Tommy's Scriptinator 3000 TM. Thereby:

- *label* refers to how the file is internally called
- *file refers* to the absolute path, where the file is stored
- *useqsub* is a boolean statement whether to use qsub
- *shortLabel* is what will be displayed as an overlay on to of the round icons on the playground

Note that those variables must be defined language specific as well.

Qsub Parameters for the use of qsub.

Script Settings In the left column of this sections the number of variables for the respective section can be variated. Note that all variables defined will go into the file header.

Within the right column the control buttons are located.

- *Load Script* will spawn a dialog to select a script in the defined language. If the script already has a valid header, then this information will update the existing panel information.
- *Confirm and Close* closes the Properties window and sets the parameters in the defined way.
- *Write to file* creates a runnable script in the respective language, including the header information wrapped in language specific comments.
- *Cancel* closes the Properties window and reverts the state to it's original state
- *Delete Script* closes the Properties window and deletes Script node

Code All code elements that were part of the sourced script, but not of the header are displayed here. Changes for instance in variable naming in the header section could be applied directly to the code in here.

Qsub support

If qsub support was enabled (setting useqsub to *true*) than a white indicator ring around the script icon will appear to indicate qsub support. This means that a predefined template script (in /templates) will be used to wrap the respective evoking script into a qsub command and submits it.

The P option

This function is used to create executable pipelines. Choosing the **P** or **P**ipe it! option in the main menu, will cause a dialog to spawn asking to select the folder where the pipeline is going to be saved.

After that all scripts that were displayed on the playground will be equipped with a header and copied to the respective pipeline folder in the language specific format. Furthermore a runnable pipeline is create. That is a file in the respective language format that calls all copied scripts in the *correct* order. This means that the hierarchy implicated by the connection lines on the playground will be inferred and scripts will be ordered such that a script that is connected *to* will be called prior to a script that is connected *from*.

License:

remark for those who need it:

Tommy's Scriptinator 3000 TM is not a registered trade mark. The name is a *funny* reference to the computer game "Thimbleweed Park".