An Open-Source Python Framework for the Generation of Questionnaire GUIs without Programming

Daphne Schössow[1], Jakob Bergner[1], Stephan Preihs[1], Jürgen Peissig[1]

[1] *Leibniz University Hannover (LUH), Institute of Communications Technology (IKT), Appelstr. 9A, 30167 Hannover, Email: name.surname@ikt.uni-hannover.de*

Abstract

Questionnaires in paper form are nowadays more and more replaced by digital graphical user interfaces (GUI), which however often can hardly be adapted to one's own wishes. To simplify the use of digital questionnaires, a Python-based open-source framework was developed, that allows to create diverse questionnaires without any programming knowledge. The software has a graphical user interface as well as the possibility to edit the entire questionnaire structure in one configuration text file. Many common question and answer types are already implemented, but the selection is still being extended. Furthermore, interfaces for communication with other software instances and systems are possible, for example OSC. The software is designed so that different components can also run on different computers.

The presented framework is explained with the example of a listening test environment that utilises subsystems for audio, pupillometry, and collection of physiological data, that are controlled and synchronised on three different computers.

Introduction

For the presentation of multimedia content in laboratory studies, it is handy to have a digital questionnaire routine instead of using paper questionnaires. However, not everyone who wishes to conduct a study has the programming knowledge to build a digital graphical user interface from scratch. Another issue that might occur with using different tools is that the built routine is dependent on a specific software version such that the PC gets cluttered with multiple versions and it is easy to lose track of which are actually needed.

The main goal of this project was to create an easy to use software for creating questionnaires without programming knowledge which makes it uncomplicated to reuse them. Further goals were the ability to comfortably run the questionnaires with touch gestures and the possibility to include different subsystems like audio and video playback or pupillometry and other biofeedback.

In the following, first the software and its possibilities and components are presented and then an exemplary setup for a listening test is given.

QUestionnaire Editor SysTem

The QUestionnaire Editor SysTem (QUEST) [1] is a software tool that is split into two main components: the creation of a questionnaire and the actual execution of an experiment. This way the tool integrates well into the workflow of creating a laboratory study as can be seen in figure 1.

QUEST is entirely programmed in Python and uses PyQt [2], a Python wrapper for Qt, for the graphical components. The QUEST source code is documented within a Wiki. Besides the description of all question types and attributes, it contains guidelines for the configuration of external software control as well as optional code extensions.

Figure 1: A typical workflow for QUEST. All steps are available from the launcher.

Editor

The main component of the software is the graphical questionnaire editor. The workspace of the editor is split into three parts: a structural view of the pages and questions, details for the selected element, and a preview of the page to be edited.

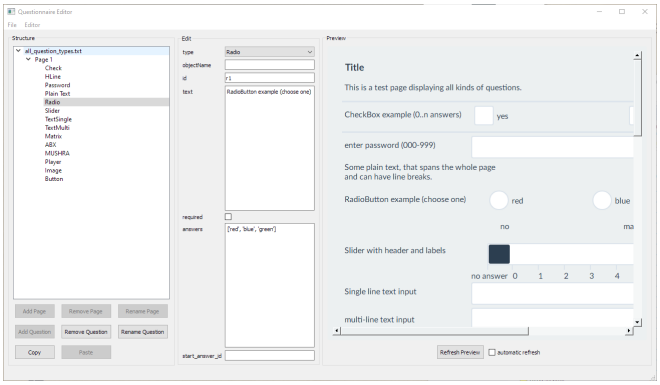

Figure 2: The questionnaire editor showing the details of the question "Radio". On the left is a structural view of the entire questionnaire. In the middle, the details of the question can be edited. On the right is a preview of the current page.

The structural view shows the (default) order of pages and the questions belonging to each page. Both, questions and pages, can be reordered by *drag&drop*. When an element is selected in the structural view, its attributes can be viewed and edited in the details editor. General settings can be edited by clicking the questionnaire name on top of the structure. As long as the questionnaire is executable, a preview of any page can be viewed in the preview section. Whenever a preview is

requested by the user the entire structure of the questionnaire gets validated. If serious issues arise, a popup window with detailed descriptions of the location and cause of the error appears. A general validation, which also includes warnings with details, can be started from the dropdown menu. Another feature from the menu allows exporting the entire GUI preview as .pdf-document.

Structure

A questionnaire file has three structural components. These are global values like the stylesheet or connection information, pages, and questions. Each page needs to have a unique name. The same holds for questions within a page. Concurrent pages can be included in a group to randomise their order, either by predefined orders from a file or by the means of balanced latin squares.

Question Types

The software features a multitude of question types ranging from basic to listening test specific. The basic question types include questions or statements where the participant has to choose one answer (`Radio`), or zero to multiple answers (`Checkbox`). If there is a selection of questions or statements that all have the same answer possibilities (of which one should be chosen), it is also possible to use the question type `Matrix` where the order of items gets randomised (once per questionnaire). Other basic question types include text entry (`Text`). The type of input can be restricted to numerical or a pattern given by a regular expression. The special question type `Password` allows only predefined input values given in a separate file. For range-based answers, the question type `Slider` provides an easy input method. It can have labels at both ends and optionally labelled ticks.

There are also some structural question types. These include a horizontal divider (`HLine`), `Plain Text`, and `Image`.

For listening test questionnaires, there is a basic `Player` module, which sends Open Sound Control (OSC) messages by clicking its Play-button to start a video or audio playback from a (different) PC. QUEST itself just provides the GUI for the questionnaire and thus does not come with any audio or video playback engine but can work with any player that can be controlled with code. Either by using built-in OSC communication or by utilising a script that receives the messages and translates them to other controls. An additional question type is a `MUSHRA` player, which consists of several playable stimuli, each with a rating slider, and a control unit. Another listening test specific question type unit is an `ABX` test. Such a test can be built with basic question types as well, this type aligns the elements automatically and adds randomisation of the order of the stimuli such that it is double-blind during the experiment. Similar to the Player there is an additional `OSCButton`, which can send any predefined OSC message to any device.

Since not every program works with OSC, another question type (`Button`) was created to send network control messages using the ZeroMQ (ZMQ) protocol, which is for example supported by the software *Pupil Capture* [5].

Figure 3: A questionnaire showing all currently possible question types.

Most question types can be set to require an answer. If none is given, the next page will not load and the missing questions are marked red.

An additional functionality of QUEST is to place a button on the bottom of each page with which the participant can send a *help request* at any time. A script to listen for this signal is provided with the software and needs to run on a different PC than the questionnaire.

Running a questionnaire

When the questionnaire is fully configured, it can be run from the launcher. By selecting the configuration file the experiment can start. This way there are no accidental changes in the structure. The structure itself and connections to other systems controlled by the GUI are checked before the questionnaire starts. The GUI runs in fullscreen mode such that other interaction with the PC or tablet is restricted for the participant.

Results are automatically saved into one continuous .csv-file for all participants. If the questionnaire is interrupted before the end of the experiment or there is an issue with the original results file, the results of the run are not lost. Partial results get automatically backed up after each page. Furthermore, the interaction of the participant with the GUI gets logged with timestamps in its own file for each participant. Results of connected systems can be synchronized over timestamps.

Interface

As mentioned before, most of the network communication of this software works with OSC over User Datagram Protocol (UDP). Some digital audio workstations (DAW) like REAPER [3] or Max/MSP [4] allow to be controlled with this protocol. The help-request listener and video-controller are own Python scripts each, which will also listen for OSC messages. To collect pupillometry data during listening tests, in our case Pupil Labs Pupil Core glasses with their software Pupil Capture were used [5]. This software allows being controlled remotely over the network, which makes it possible to send controls from the GUI. However, this software works with ZMQ instead of OSC, which is deemed more reliable by the developers [5]. Thus the GUI has to listen for replies from Pupil Capture.

Example Setup

In the following the setup for the listening tests of the project "Wagner 3.0" is described.

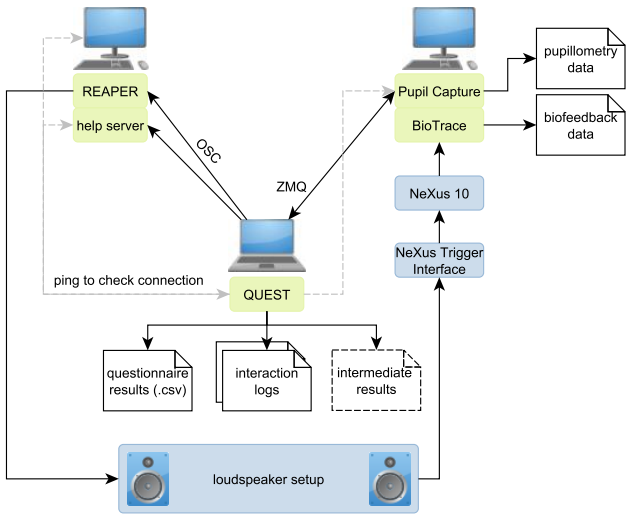

Figure 4: The setup for the listening tests for "Wagner 3.0" using QUEST. Green are software, blue hardware components. White are data files created in a experimental run.

The listening tests are conducted in a laboratory with the participant being seated in the centre of the room surrounded by loudspeakers. For the experiment pupillometry data is collected with the Pupil Core glasses. Additionally, biofeedback data is also collected from the participant using the NeXus 10 [6]. This includes breathing rate, skin conductance, blood volume pulse, and electromyography. However, the software BioTrace is closed source and does not allow to be controlled remotely. To have synchronisation points in the recordings, the NeXus Trigger Interface is utilised. Whenever a stimulus starts playing in REAPER, a unique binary coded signal is sent to the line-in input of the interface. This results in so-called line triggers in the recordings. This is necessary not only to ensure the synchronicity of the subsystems but also because the recordings do not include global clock timestamps. Pupil Capture and BioTrace are running on a PC behind the participant while another PC, on which the DAW is running, is situated in the control room. A REAPER session with markers for the different stimuli was created. Additionally, the help server of QUEST is running on this PC as well such that the participant can easily call for help if needed and the instructor can check the progress of the experiment. To synchronise the three measurements the questionnaire GUI of QUEST logs the timestamp when the Play-button of a `Player`-object is pressed. At the same time an annotation with the name of the stimulus and the time is sent to Pupil Capture via ZMQ. When the REAPER session receives the play signal over OSC a 6 digit binary coded signal, which is unique for each stimulus, is transmitted to the NTI over its line-in port to create trigger points. This experimental setup leads to more than one file as the data from the external systems (Pupil Capture and BioTrace) produce their own files. However, the files can be linked over the participant number and timestamps. The sent triggers can be used for synchronisation as well.

Conclusion

An open-source software for easily creating and using digital questionnaires was programmed. The already present question types allow a wide range of uses and are highly customisable over the editor's GUI. It is also possible to integrate QUEST into one's development cycle as the questionnaire preview can be saved and be used individually from the program itself. As the possibilities of question types for laboratory studies are plenty, the software is still in active development and new features get added over time. Further improvements might include interfaces to different systems, more question types e.g. sound source localisation, a graphical editor for the stylesheets, and much more. As the software is freely available under the GNU GPL 3 license anyone can contribute or extend the software to their need. Currently the software was only tested under Windows 10, thus compatibility with other systems is not guaranteed. Future releases will tackle this issue.

References

[1] Schössow, D.: QUEST - QUestionnaire Editor SysTem, URL: `https://gitlab.uni-hannover.de/da.schoessow/quest`(09.03.2022), DOI: `10.5281/zenodo.6340995`

[2] Riverbank Computing: PyQt, URL: `https://www.riverbankcomputing.com/software/pyqt/`(21.02.2022)

[3] Cockos Incorporated: REAPER Digital Audio Workstation, URL: `https://www.reaper.fm/`(21.02.2022)

[4] Cycling '74: Max/MSP, URL: `https://cycling74.com/products/max`(17.03.2022)

[5] Pupil Labs: Pupil Core Network API, URL: `https://docs.pupil-labs.com/developer/core/network-api/`(21.02.2022)

[6] Mind Media: NeXus and BioTrace+, URL: `https://www.mindmedia.com/de/`(21.02.2022)