Project report (Group dem03)

Jannik Jaberg and Jonas Rudin

University of Basel
Seminar: Modern Human Machine Interaction (MHMI)
Autumn Semester 2019

Report Requirements

This is a brief overview of the requirements on the Seminar: Modern Human Machine Interaction and some guidelines for the report.

Deliverables

You are expected to hand-in the following items on ADAM until **Tuesday, 28th January 2020 23:59**:

- This **report**, as PDF and the complete sources of your report in the same structure as distributed as a ZIP archive
- A 30 second to 1 minute **video**, showcasing your project and its capabilities
- Your **slides** for the final presentation on January, 30 2020. You will have max. 15 minutes to present your project (you are allowed to use less than the full 15 minutes) and there will be 5-10 minutes of Q&A following your presentation

Structure

The sections in this template are already a guideline for the structure of your report. You are allowed to change the titles. However, please make sure that the content outlined below is included in your report.

Introduction Why is your topic important / interesting (how is it related to the topic of the seminar / existing research projects), what specific goal(s) did your project have and how did you implement them. The introduction should give a brief and relevant overview of the topic and your project and already indicate to the reader the results of your project.

Concepts Describe and justify theoretical concepts / technologies used and reference / discuss / summarize relevant literature here. Show and discuss the architecture of your project.

Implementation Briefly summarize your implementation work. You may also show / discuss the architecture in this section, depending on your project and what you deem appropriate.

Usage A manual for users and developers of your project: How do you set it up, how is it used?

Conclusion Briefly summarize the outcomes of your project, lessons learned and make recommendations for future work (both smaller improvements to your specific project and bigger ideas)

Deadline

The deadline for the hand-in is on **Tuesday, 28th January 2020 23:59**. Please hand-in all the required files in the corresponding form on ADAM.

On the next page, there is a guide on how to use this template. Please read it carefully and stick to it. We will compile a document with all reports together and distribute that to all participants.

HOW TO

This template is used for the student reports of the Seminar: Modern Human Machine Interaction course. Remove this section by editing the `reportContent.tex` file **after** you have understood how to use this template.

Folder Structure

```
proceedings /
  |- reportContent /  <-- YOUR working folder
      |- images /     <-- Images are stored here
      |- sections /   <-- Additional sections / tex-files here
          |- _HOWTO_.tex    <-- *this* How To section
          |- introduction.tex
          |- report.bib
          |- lessons-learned.tex
          '- section2.tex
      '- reportContent.tex  <-- Change your section imports here
  |- proceedings.tex  <-- DO NOT alter this file
  |- llncs.cls                              <-- NOR that
  |- proceedings.py                         <-- NOR that
  |- splncs03.bst                           <-- NOR that
  '- UniBas_Logo_EN_Schwarz_RGB_65.pdf      <-- NOR that
```

Setup

Your report is compiled using the `proceedings.tex` file. Therefore, we suggest that you set the file `proceedings.tex` as the Master-/Root-Document that you can conveniently work on your report. Please be aware that you have to compile the `proceedings.tex` file in the following order:

1. `pdflatex` proceedings.tex
2. `biber` proceedings
3. `pdflatex` proceedings.tex
4. `pdflatex` proceedings.tex

For your convenience, you can compile this document by using the `proceedings.py` utility:

```
$> python proceedings.py -p
```

And subsequently (especially before your submission) clean the auxiliary LaTeXfiles:

```
$> python proceedings.py -z
```

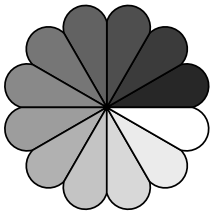


Fig. 1: Example figure stored in the `images` sub-folder and displayed with `\includegraphics[width=0.45\textwidth]{g1-rosette}`

Images

Images placed in the `images` sub-folder can be used directly in the `\includegraphics` command – see also Figure 1. Also, sub-figures are possible: Figures 2a and 2b.

Please DO NOT store your images elsewhere!

Please use your group number (e.g., *1*) in every image file name (e.g., *g1-rosette.pdf* instead of *rosette.pdf*)!

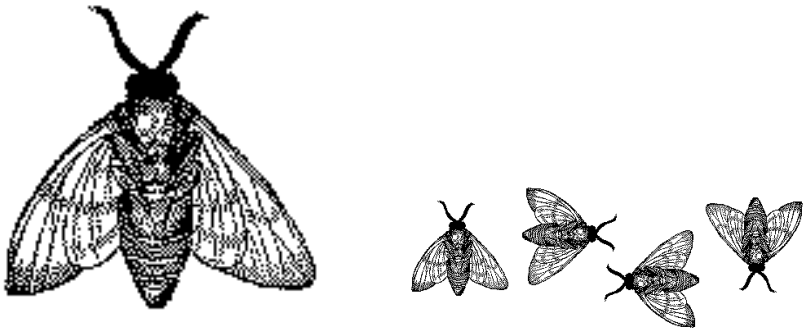
Code

To illustrate code use the `lstlisting` environment:

```
public void foo( final int bar ) {
  System.out.println( bar );
}
```

Labels and References

The proceedings template includes the package *cleveref*, which enables pretty references. One such reference is the following one: Figure 2, which was produced using the command `\Cref{fig:example subfigure}`. To use the full potential of such clever referencing, one have to set up labels, such as in the example before. The label command was `\label{fig:example_subfigure}` (as seen in the source code in line 69). Be aware that you have to prefix every single label you use with your group name, i.e. `g1` to not clash with other group's labels.



(a) A fly (b) Flies

Fig. 2: Example figure stored in the `images` sub-folder

Bibliography

We use the `biblatex` package with biber[1] as backend, which most distributions already include. Your references are stored in the `report.bib` file and used with the `\cite` command. For example: [1]. If the previous example shows `kopka1995guide` - you need to re-run the compilation in the following order: pdflatex, biber, pdflatex, pdflatex. In case you need some help with references in latex, there is an excellent cheatsheet[2].

[1] `http://biblatex-biber.sourceforge.net/`
[2] `http://tug.ctan.org/info/biblatex-cheatsheet/biblatex-cheatsheet.pdf`

1 Introduction

The goal is to input relational query plans via gesture. To realize this, we want to implement a video feed into the Polypheny-DB-UI. The real time video feed should give feedback to the user which gestures are detected and which querries are selected. Firstly, the gestures are hardcoded to the operations (e.g. fist - filter), to afterwards select the fields of an operation. We would like to have droptabels for each field, where also each option is connected to some gesture. To switch between the sidebar and the tree, we need some sort of "enter" gesture as well as an "escape" and switching gesture. At the moment, we were not able to find which gestures are supported by Deepmime, but the project can be realised with at least 10 gestures, even though intuitiveness might suffer.
Secondly, might be difficultly complicated, we thought of a drag and drop style interaction, where the mouse is replaced by the gestures of thumb and index finger. *This is not an activ part of our project* but it should behave like a touchscreen. Precisely, the camera feed and de Polypheny-DB-UI are overlayed so the users sees where the fingers are and the operations can be picked by closing thumb and index finger and dragging it into the working field. To selecet the fields of each operation, the same will apply again by open thumb and index finger, acting as mouse. By shortly tapping them together, a click shall be detected. This method is much more intuitive and has a better HCI.
That makes this project in a way innovative and futuristic. The invention of the touchscreen has been the foundation of HCI without usning a keyboard or a mouse. The next step would be to not even touch anything anymore, which can be realised by voice and gesture control. Alexa made already a big step into this direction. If a reliable gesture control is developed, it will only take a few steps to feel like Tony Stark, while talking to Jarvis and controlling his computer freehanded and only by gestures.

The following must haves we defined in the proposal:

- Camera feed integrated in UI
- Camera feed shows detected gesture
- Operation linked to gesture is highlighted after gesture is detected
- Operation selection has to be confirmed/canceled with "enter"/"escape"-gesture
- To re-enter/switch between side bar and selection, switching gestures are supported
- To select fields of an operation, drooptabels are provided where each option is linked to a gesture, which also has to be confirmed

2 Concepts

Polypheny - Polystore - Database - Relational Algebra as graphical interface with nodes where a sql statement can be built know from Databases

Gesture Recognition - 3d resnet - Concept shown in Pattern Recognition

Own Json Parser

Every Node is represented as node class redone in Python
Gesture gets detected - send to our server as string - parsed to the json tree - then send to polypheny and re arranged

3 Implementation

General introduction to architecture
Schema of architecture
we should have used fucking alexa

4 Usage

just try - if there are errors, you can keep them for you - please do never contact us - thanks

5 Conclusion

outcome: Abfall - not what we wanted - connection between modules painful bc: CORS Sockets API Angular lessons learned - do not fucking use angular if you dont want to commit suicide - else it's a pretty good framework to lose believe in life and want to question your own existence - Plan more in advance - do not lose time with stuff you cant fix - do not believe assistants that it's an easy task nor believe that the project idea is thougt through

References

[1] H Kopka and PW Daly. "A Guide to {\LaTeX}–Document". In: (1995).