EyeSimplify

Overview

- A Programming Project
by

Susanne Manke
Ruprecht-Karls-University
Department of Computational Linguistics
(manke@cl.uni-heidelberg.de)

January 2007

Supervisors:
Dr.Markus Demleitner, Dr.Anke Holler

In Coorporation with:
Dr. Lisa Irmen, Institute of Psychology, Heidelberg

In Correspondence with:
Charles Clifton, Psychology Department, Tobin Hall, University of Massachusetts, Amherst

Abstract

This documentation provides information about the EyeSimplify program, written by Susanne Manke. The EyeSimplify program is a framework for processing eyetracking .edf files to an analysis textfile representing four standard eye movement measurements: First Fixation("ff"), First Pass ("fp"), Total Time of Fixations ("total time"/ "tt") and First Pass Regressions Out ("rout").

It is an overall project documentation, but it does not include any information about project history. Further information and all documentary files can be found in the final talk about the project and on the project's website:
http://www.cl.uni-heidelberg.de/~manke/bba.htm.

Furthermore, it is not an user's guide. This is provided in a separate file.

Content

Abstract 2

1. Program Introduction – Conceptual Level 4

  1.1 Before EyeSimplify: What needs to be done 4

  1.2 EyeSimplify: What is done 6

2. Program Introduction – Implementation Level 8

  2.1 Programs used for Development on Windows XP 8

  2.2 Programs based on Development 8

  2.3 Selected Methods explained 9

    2.3.1 The EyeSimplify – class 9

    2.3.2 The Edf2Asc – class 9

    2.3.3 The AddReg – class 9

    2.3.4 The Edprep – class 10

    2.3.5 The EyedryGUI – class 10

    2.3.6 The EyedryAna – class 11

  2.4 Exception Handling 13

3. Testing 14

  3.1 During Development – Functionality Tests 14

  3.2 Final Regression Test 14

4. Evaluation 15

  4.1 Evaluation Plan 15

  4.2 Results 19

5. Learned Lessions 20

  5.1 The Project itself 20

  5.2 The Development 20

7. Plan for Version 2.0 21

  7.1 Final Tests – Usability 21

  7.2 Debugging 21

    7.2.1 Problems with Tkinter and Windows XP, and PMW 21

    7.2.2 The output from ES differs from the output of ED 21

  7.3 Adding Features 21

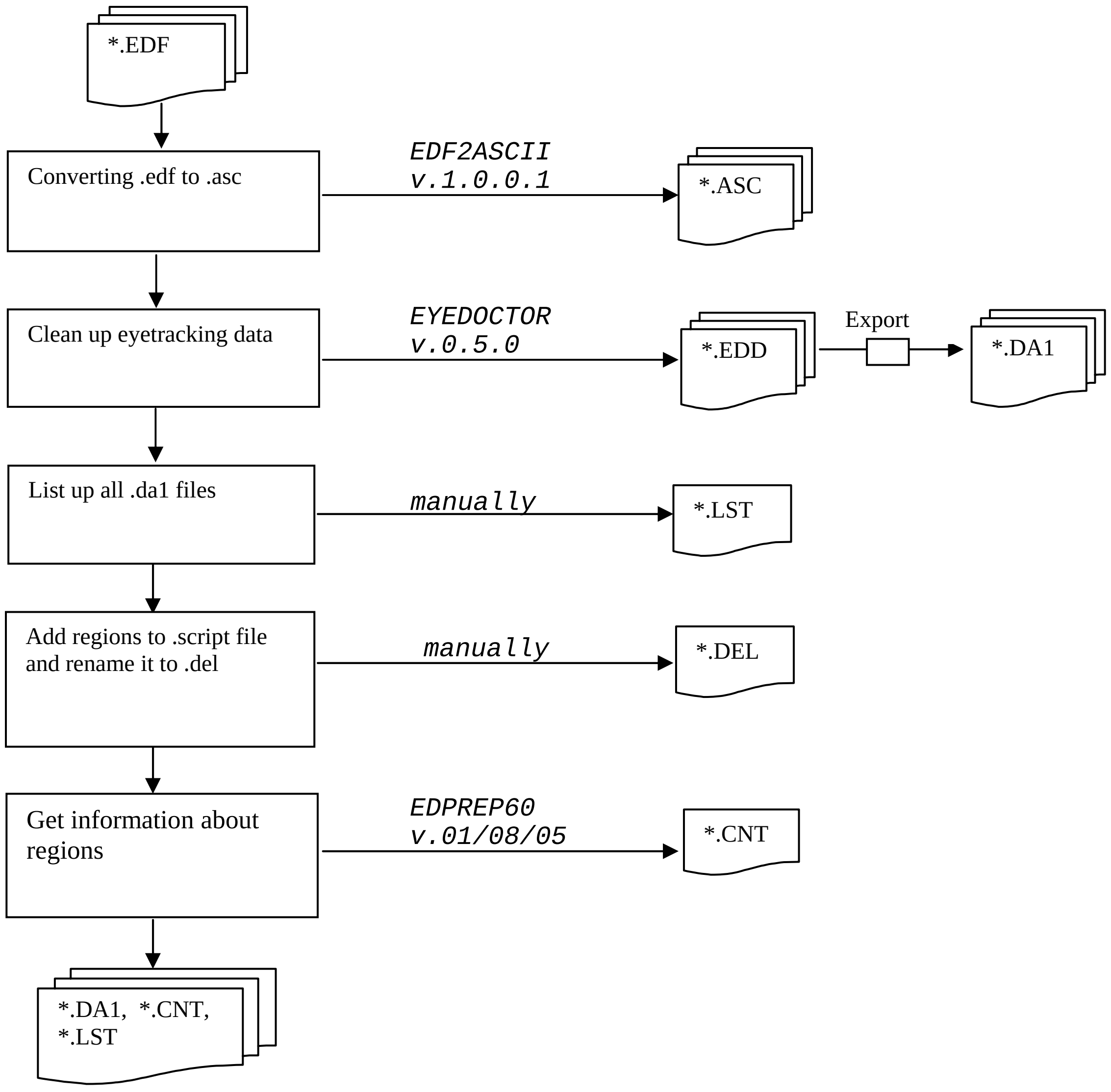
Thank You 22

1. Program Introduction – Conceptual Level

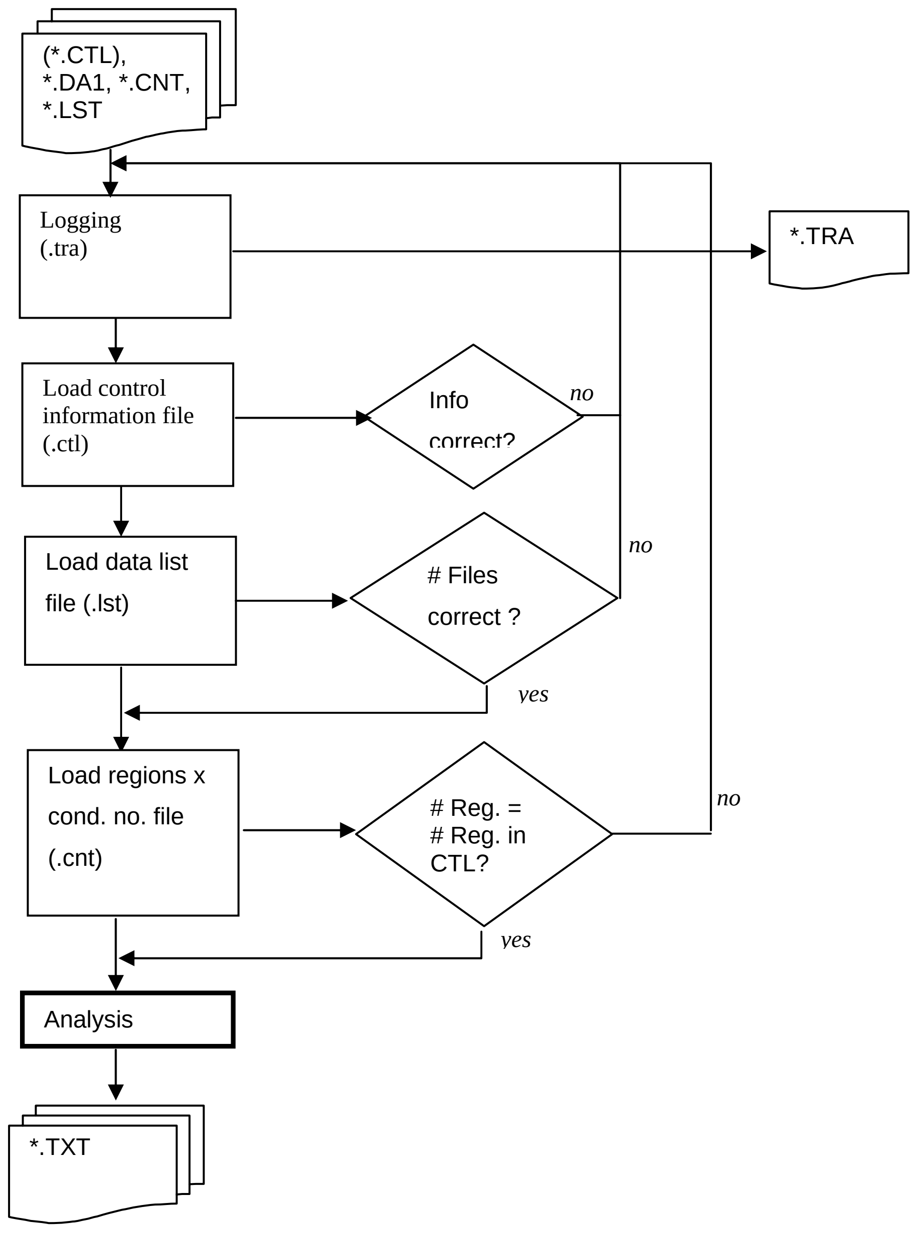
This section gives an overview about the process of analysing eyetracking data.

1.1 Before EyeSimplify: What needs to be done

Phase 1. Prepare eyetracking data:

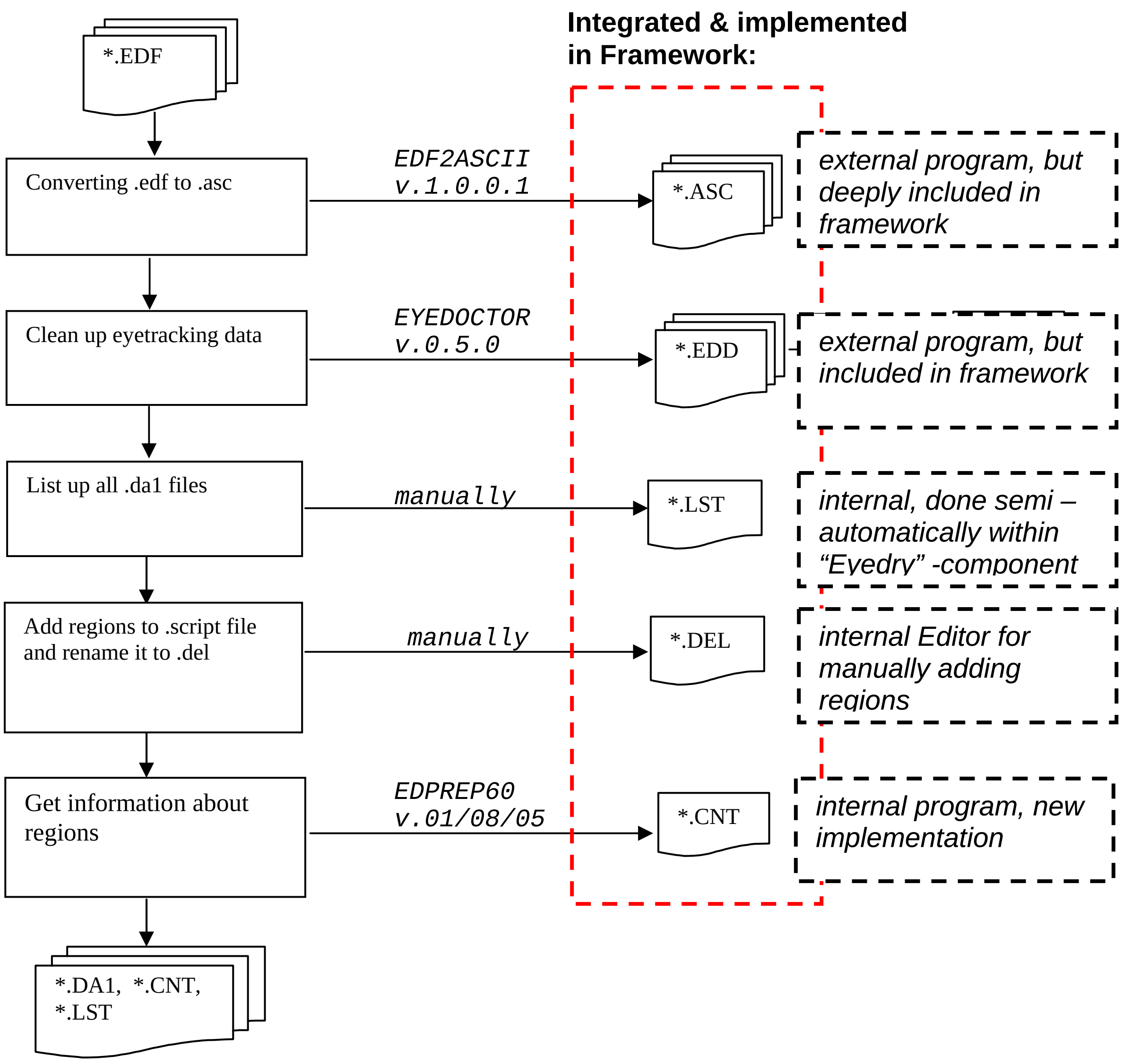


Phase 2. Prepare for analysis and perform analysis:

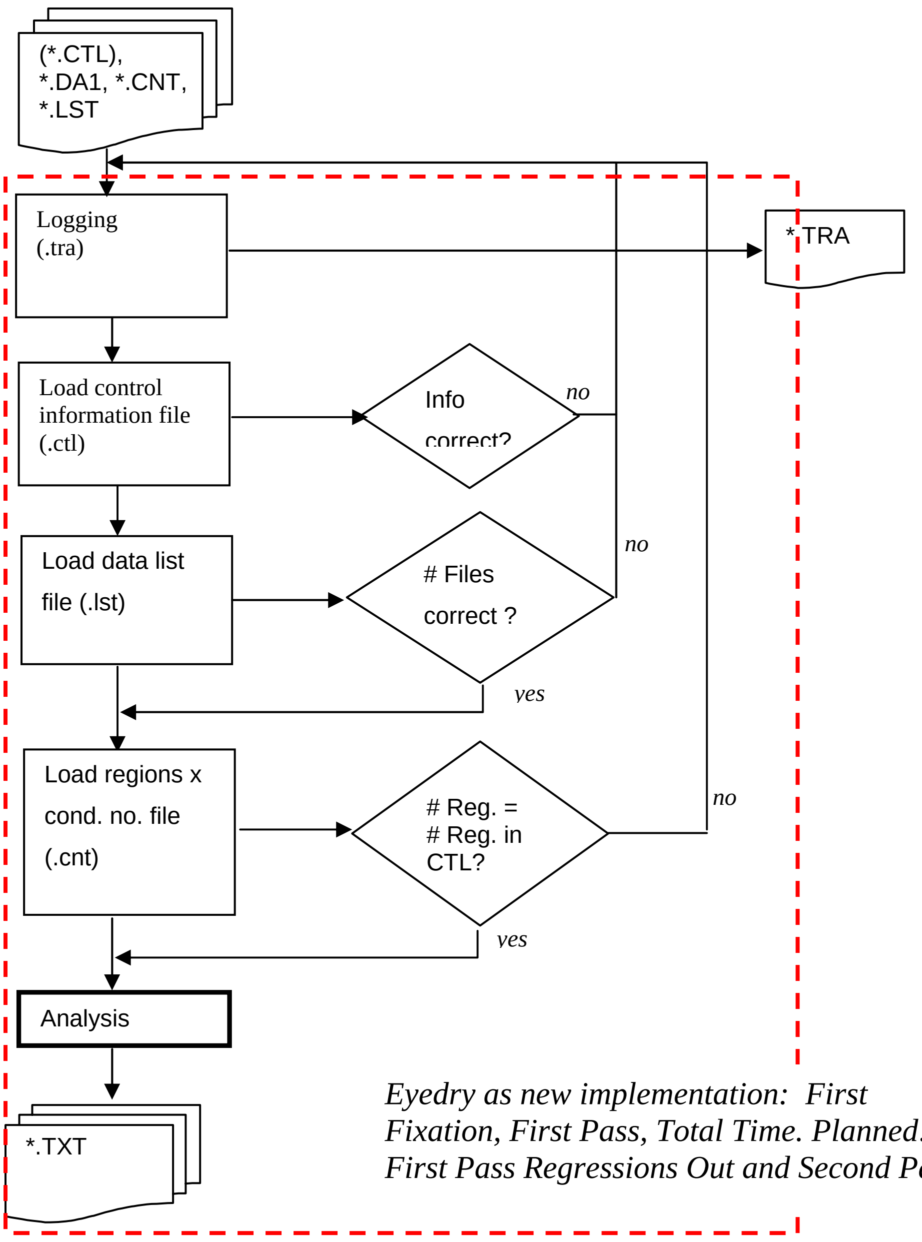


1.2 EyeSimplify: What is done

Phase 1.


Integrated & implemented in Framework:
*.EDF
Converting .edf to .asc
EDF2ASCII v.1.0.0.1
*.ASC
external program, but deeply included in framework
Clean up eyetracking data
EYEDOCTOR v.0.5.0
*.EDD
external program, but included in framework
List up all .da1 files
manually
*.LST
internal, done semi – automatically within "Eyedry" -component
Add regions to .script file and rename it to .del
manually
*.DEL
internal Editor for manually adding regions
Get information about regions
EDPREP60 v.01/08/05
*.CNT
internal program, new implementation
*.DA1, *.CNT, *.LST

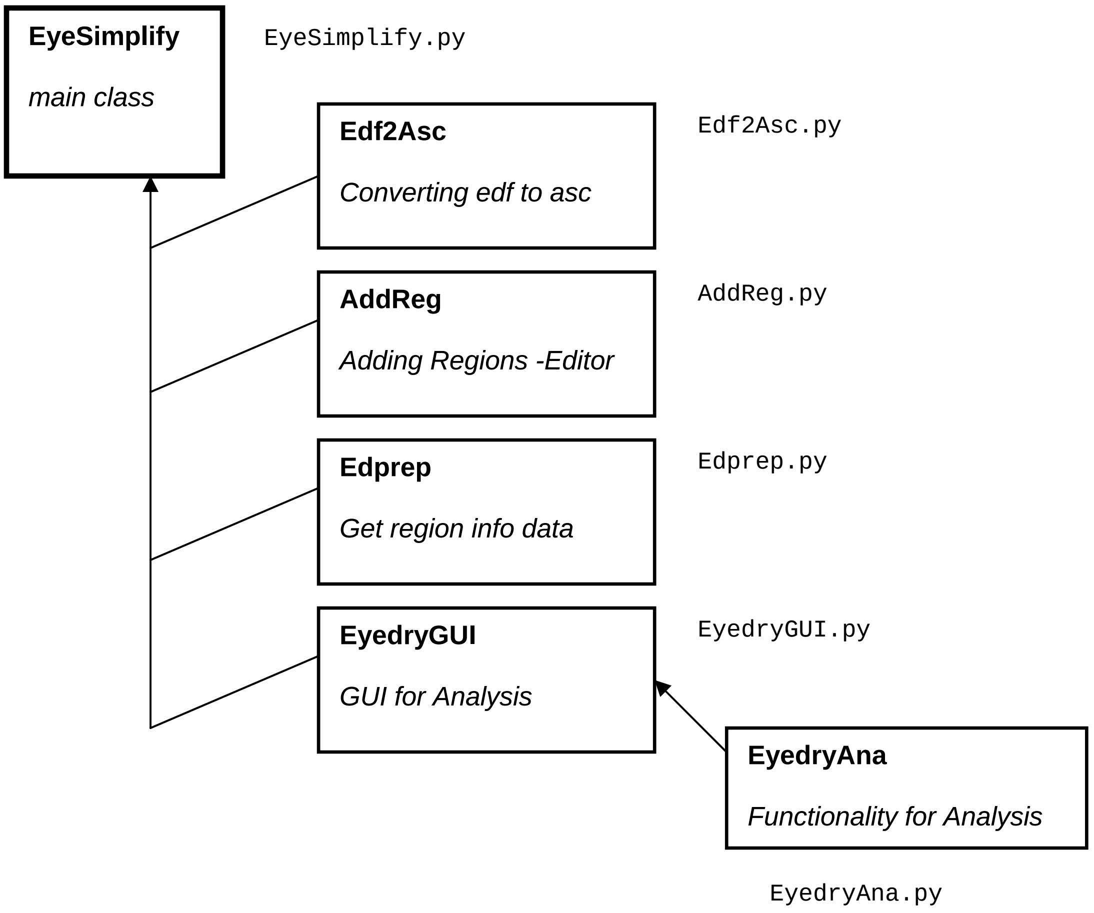
Phase 2.



Eyedry as new implementation: First Fixation, First Pass, Total Time. Planned: First Pass Regressions Out and Second Pass.

2. Program Introduction – Implementation Level

This chart shows the EyeSimplify program with its modules:



2.1 Programs used for Development on Windows XP

- Programming Language: Python 2.4.4
- GUI: Tkinter and Python Mega Widgets (PMW) 2.4
- IDE: Python IDLE
- Texteditor: Textpad 4.7.3 (unregistered)

2.2 Programs based on Development

- EDF2ASC (DOS-based version)
- Eyedoctor 0.5.0
- EDPREP60
- Eyedry

These programs were partly available as C – source code.

2.3 Selected Methods explained

This section gives additional information to some methods.
Please look at the embedded documentation and its corresponding pydoc – documentation for more information.

2.3.1 The EyeSimplify – class

EyeSimplify is the main class. It uses all other modules and partly represents the main GUI for Microsoft Windows.

All methods are self-explanatory (or with embedded doc).

If you want to add several links to a tab, just use this code snippet:

```
def showEyedryGUITab (self):
   self.nb.selectpage(5)
```

<u>What is planned for version 2.0:</u>
Use wxPython for GUI implementation.

2.3.2 The Edf2Asc – class

Edf2Asc is the module to convert an .edf file to an .asc file.The DOS-based tool EDF2ASC runs in background and will be started automatically. The .asc files are needed for later processing.

def convert(self):
Takes arguments given by user: .edf file and future .asc filename and run this with command – line tool Edf2Asc.exe

Due to some constraints by Edf2Asc, the chosen .edf file is copied temporarly (as tmp.edf) in the working directory and removed later on.

<u>What is planned for version 2.0:</u>
Implementation of a GUI for Edf2Asc, which enables the user to add more options to the conversion, e.g. left/right eye or batch process for converting more than one .edf file.

2.3.3 The AddReg – class

AddReg is an “editor” to view the .script file or .del file and add the delimiter characters to it and save the file. You can also use any other text editor, but that one included should fasten analysis preparation steps.

def openFile(self):
If you want to get the content of a PMW ScrolledText – widget, a special method is needed. `importfile(file)` .

What is planned for version 2.0:
Improve Editor: Highlight the delimiter characters.

2.3.4 The Edprep – class

Edprep writes the region info from delimited .script file (.del), combined with condition and item no. to a table. It is needed for the later mapping of all information together with the .da1 files.

def runEdprep(self):
Run Edprep. This is similar to the original Edprep program, but it's simplified. It takes every kind of possible del. char. (which is not a char in the .script/.del file), e.g. ^ .

Read the .del file:
Using this regular expression, “`inline =\s*\|(.*?)$")`”
it is ensured that everything is taken from .del file after each line beginning with “inline”.

The next regular expression does the following:
```
ret = re.compile("trial E(\d*)I(\d*)D0(.+?)end
E\d*I\d*D0", re.M).findall(content)
```

It stores the data in a list with strings, containing: condition no. (list index [0]), sentence no. (list index [1]). And you get everything between E#I#D# ... end E#I#D# with index [2].
That is important, because these are the experimental trials we are interested in.

What is planned for version 2.0:
Include the functionality into Eyedry. This separate step is not really needed, but has been implemented for transparency. Otherwise IT IS REQUIRED to implement lots of internal checks to identify possible errors. Error detection is done by user when checking the .cnt file.

2.3.5 The EyedryGUI – class

EyedryGUI is a module to build up the GUI for the general infos needed for the analysis of FF, FP, TT.

def runEyedry(self):
Get values from GUI and run EyeSimplify - Eyedry.
Run EyedryAna:
```
analyser = EyedryAna(fill, long_cutoff, short_cutoff,
cntfilename, filelist)
```

What is planned for version 2.0:
Add more options to GUI as options for the analysis, e.g. constants like Line Length, TT_CutOff, FP_CutOff.

Add additional options to each type of analysis.

2.3.6 The EyedryAna – class

EyedryAna computes the eye movement measurements that were chosen by user through the EyedryGUI module, which is used in the main module EyeSimplify.

def loadCntFile(self):
Get the region definitions from cnt file. This is done with the help of a regular expression:

```
lineRegEx =
re.compile("^\s*(\d+)\s+(\d+)\s+\d+\s+\d+\s+\d+\s+(.+)
\s*$")
```

Where item no. = regular expression group 1, condition no. = regular expression group 2 and group 3 are region borders (limits).

def loadData(self):
Parse a da1 files for needed info: cond no., item no., subject no., char no., line no., fixation data. This is also done with a regular expression:

```
lineRegEx =
re.compile("^\s*\d+\s+(\d+)\s+(\d+)\s+\d+\s+\d+\s+\d+\
s+\d+\s+\d+\s+(.+)\s*$")
```

Where condition = group 1, item = group 2 and the coordinates of fixations [line no., char no, fix.begin, fix.end] = group 3.

def calculate(self):
Map everything together: fixation info from .da1, region info from .cnt and give computed results (by insertData) to output (done by writeOutput). This method gets all data:
dictionary with “set, cond and item as key” – tuple and region limits, by various checks, like a regression check (have I already been there, or not? -> increment position, if already visited).

What is planned for version 2.0:
Add more eye movement measurements:
First Pass Regressions Out, Second Pass.

def insertData(self):
This function computes the values for chosen analysis.

First Fixation:
Is the duration of first fixation in a region.

Premissions:
If fixation too short, get next fixations.
If fixation too long, ignore next fixations.
Ignore all regressions (character by character)

First Pass Time:
Accumulate all fixations between first fixation and first leaving of that region.

Premissions:
Ignore all regressions (region by region, regressions within a region are valid).
Leaving region to left side is catched separately
If too long fixation during pass, then ignore whole pass

Total Time:
Acculumate all valid fixations (not to short/too long) within a region.

Premissions:
If fixation too long, ignore all following fixations.

What is planned for version 2.0:
Debug FP, TT and FF in cooperation with Chuck Clifton.
There are several values which are not equal to the output that original Eyedry produces: EyeSimplify removes too many values from the measurements.

def writeOutput(self):
Write computed values to output table with or without "0"s.
Additionally, it cut offs total values from the given cut off, which is for Total Time 4000 and for First Pass 2000. These values were given by the original analysis tool and are “hard-coded” in this program.

What is planned for version 2.0:
Implement cut off values as dynamic values, given by the user. (GUI)

Small Little Helpers:

- def getFixation – Get 4er tuple of fixation coordinates
- def getDelta – Compute duration of fixation
- def checkCutoff, def fixTooLong, def fixTooShort – Fixation Check (and Cut Offs)
- def isRegression(self, delta, position) - Regression – Check
- def checkLineEnd(self, region_limit, delta) – Check for Line End

2.4 Exception Handling

All exceptions are IOErrors. In relation to the importance of a message, it is printed as stderr out (if needed for whole procedure), otherwise it's printed to a log file.

The method runEyeDoctor doesn't need any exception, because it is run via os.system and throws an exception anyway.

3. Testing

Testing is an important part of the project. Unfortunately there was not enough time to test the usability of the program, but this is planned for the development in version 2.0.

3.1 During Development – Functionality Tests

The several parts of the program have been tested several times during development. This includes e.g. functionality of buttons, tabs and the logging of Edprep and EyeSimplify.

During the development of the final analysis part, there were lots of tests. In most cases, a file set containing "expling.cnt, mb1n23.da1, test.ctl" and "mb.lst" that were created with the original software Eyedry and Edprep. These files helped the developer, to identify premissions for the measurement of fixations and to check them with created output by Eyedry. It also meant very time-consuming tests like, change something and re-implement it and so on. Finally, it works for the development file set.

3.2 Final Regression Test

A regression test checks the correctness of the software, whenever parts of the software are working properly.

Process: 40 files from original study at Institute of Psychology. A set of result files has been created with Eyedry and than with EyeSimplify, using the same .cnt, .da1 and .lst file.

For this test, the developer wrote a small tool: "EmptyLineKiller". A program that removes empty lines from the Eyedry output and therefore make it syntactically comparable with the EyeSimplify output (title of tab deleted before testing).

Result: FAILED. This is a great problem, because the program cannot be used until it is found why EyeSimplify removes several fixations from the final measurements.

4. Evaluation

An evaluation for EyeSimplify is not a simple task. It is mainly done by comparing the output files with the output created with Eyedry.

4.1 Evaluation Plan

The following evaluation plan has been used to eval EyeSimplify. It contains the task description for the judgers.

- 1. Convert Edf - Files -
No conversion eval needed, because EDF2ASC is the same program used for preparing data for analysis.

- 2. Clean Up ASC Files -
Clean Up is the same process, but in Eyesimplify: without searching for the program executable.

- 3. Adding Regions of Interest -
No Evaluation, because regions of interest have to be added manually at the whole file and this can be done with the included editor in ES or with any other text editor.

- 4. Get Region Data -
This program is originally known as EDPREP. You can find more information
in the user's guide.

EyeSimplify contains a newly implemented and more feasible implementation of this tool.

SUBJECT 1

- File: expling.del

- Open the EDPREP60 tool, given in your working directory.

- Insert the following:
    - smallest condition no: 1
    - largest condition no: 14
    - smallest experimental sentence no: 1
    - largest experimental sentence no: 36
    - delimiter character: ^
    - name of delimited input file: expling.del
    - count file: test.cnt

[Duration, using EDPREP60 tool: 50.5 sec]

SUBJECT 2

- File: expling.del

- Open EyeSimplify, tab 4 "Get Region Data".

- Insert the following:
    - 1 - smallest condition
    - 9 - largest condition
    - 1 - smallest sentence
    - 36 - largest sentence
    - keep given delimiter character

- Click on "generate cnt file":
    - give dialogue a .del file
    - give dialogue a .cnt file

[Duration, using EyeSimplify - tab 4: 23.7 sec]

<u>-5. Data Analysis -</u>

Type of Analysis: First Fixation

File Set 1: mb1n12.da1, mb1n23.da1
File Set 2: mb3n5.da1

SUBJECT 1

- Analysis with Eyedry:

    File Set 1: mb1n12.da1, mb1n23.da1

    Use given files:
    - mb2.lst – Data file list
    - expling.cnt – region info file
    - test.ctl – general info for Eyedry

    Please start the Eyedry program.

    Follow the questions on the screen with given answers:
    1. `Want a hard copy ? y/n :` ***`n`***
    2. `What is output trace file name? :` ***`test.tra`***
    3. `Type an identifying string to print out:` **`<enter>`**
    4. `What is maximum number of fixations on an item:` ***`100`***
    5. `Type name of file containing control info:` ***`test.ctl`***
    6. *`type`* ***`<enter>`***
    7. `Name of file that lists data files ? :` ***`mb2.lst`***

```
8. Any exceptions file? y/n : n
9. Control (cnt) file name? : expling.cnt


Analysis Menu:
  a. Choose a number for an analysis: 1

First Fixation:
    a. 1, r, n, n, ff, - , - , - , -, n
    b. insert output file: tested.txt
    c. Do you want a typeout of the item by item
       data? n
```

[Duration, Analysis with Eyedry: 49.6 sec]

- Analysis with EyeSimplify:

  File Set 2: mb3n5.da1

  Use given file:
    - expling.cnt – region info file

  (*no other file is needed, because for one file you don't even need a .lst*)

  Please start the EyeSimplify program.
  Go to tab 5.

  Do the following:
  1. Data List:
     Add "mb3n5.da1" to the list
     Bug: *Move with mouse over the window and all widgets will appear*.
  2. Add "expling.cnt" to the list
  3. Choose First Fixation
  4. Click on "Generate"
  5. Click on „Save as" and save file as: „testes.txt"

[Duration, Analysis with EyeSimplify: 16.6 sec]

SUBJECT 2

- Analysis with EyeSimplify:

  File Set 1: mb1n12.da1, mb1n23.da1

  Use given file:
    - expling.cnt – region info file

  (*no other file is needed, because for two files you don't even need a .lst*)

  Please start the EyeSimplify program.

Go to tab 5.

Do the following:

1. Data List:
   Add “mb1n12.da1, mb1n23.da1” to the list
   Bug: *Move with mouse over the window and all widgets will appear*.
2. Add “expling.cnt” to the list
3. Choose First Fixation
4. Click on “Generate”
5. Click on „Save as“ and save file as: „testes2.txt“

[Duration, Analysis with EyeSimplify: 20.2 sec]

- Analysis with Eyedry:

  File Set 2: mb3n5.da1

  Use given files:
  - mb1.lst – Data file list
  - expling.cnt – region info file
  - test.ctl – general info for Eyedry

  Please start the Eyedry program.

  Follow the questions on the screen with given answers:

```
1. Want a hard copy ? y/n : n
2. What is output trace file name? : test2.tra
3. Type an identifying string to print out:<enter>
4. What is maximum number of fixations on an item:
   100
5. Type name of file containing control info:
   test.ctl
6. type  <enter>
7. Name of file that lists data files ? : mb1.lst
8. Any exceptions file? y/n : n
9. Control (cnt) file name? : expling.cnt


Analysis Menu:
   b. Choose a number for an analysis: 1

First Fixation:
      d. 1, r, n, n, ff, - , - , - , -, n
      e. insert output file: tested2.txt
      f. Do you want a typeout of the item by item
         data? n
```

[Duration, Analysis with Eyedry: 47.7 sec]

4.2 Results

Two different people were trained in both programs before evaluation to en-sure a comparable result. This could be also done untrained, but I wanted to keep it simple for the subjects.

Eyedry	Subject 1	Subject 2	**EyeSimplify**
File Set 1	49.6 sec	**20.2 sec**	
File Set 2	**16.6 sec**	47.7 sec	

The table shows the values for EyeSimplify and Eyedry. EyeSimplify is more than 2-3x faster with both file sets in processing a first fixation analysis.

Additionally, a comparison in the no. of steps to perform for an analysis represents also the advantages of EyeSimplify: ca. 13 steps for Eyedry whereas EyeSimplify only needs max. 7 steps.

Evaluation passed.

5. Learned Lessions

This chapter highlights the learned lessions during a programming software project like this one.

5.1 The Project itself

In contrast to other projects, this project has been very time-consuming. The most time-consuming factor was the lack of information about how to compute eye-tracking measurements.

The motivation of the project is still given, when there are milestones within the project. This is also ensures that everything is in time.

All problems and changes and “to-dos” were written to a project log. This helped to keep an overview of the whole project.

5.2 The Development

The developer made several experiences during the development.

One very time-consuming factor was to refresh programming skills in Python, especially in totally unknown chapters like PMW etc.

During the development of GUI it is important that minor functionality is given and you can check the usability afterwards(!). If you want to re-design some parts of the GUI, you will need much more time as expected. For example, it could last one day to implement an additional feature.

Code must be thrown away, in the very first moment you don’t need it anymore. According to the principles of Extreme Programming, you won’t need your code again!

GUI Programming with Windows XP: first error.
GUI Programming with Tkinter and Windows XP: second error.

Don’t forget any backup! Fortunately the developer forgot nothing to backup, but after a week of 10 hours per day of programming, it would be very annoying to loosing time.

7. Plan for Version 2.0

As already mentioned above within part of the software, the following is planned for version 2.0.

7.1 Final Tests – Usability

The usability testing should be done with the current version and the version with a new GUI.

It can be tested with several persons with different background knowledge. The main user is defined as having knowledge in computers, psychology and eye-tracking measurements.

7.2 Debugging

7.2.1 Problems with Tkinter and Windows XP, and PMW

Vanishing widgets seems to be in relation to the TK – SaveAs-Dialogue. This can be solved by porting the GUI to wxPython – maybe.

Estimated Time Complexity: 1 month

7.2.2 The output from ES differs from the output of ED

This needs to be tested for each different figure per analysis.
The debugging would be a trial-and-error – development which can be very time-consuming, if you don't know why a figure is included in the measurements and why not.

The final regression test showed that EyeSimplify removes too many fixations from the measurements.

That debugging needs to be done in closer cooperation with Chuck Clifton and his team, because it needs to be checked whether there is a bug in ED or the fixations are needed or ES makes something wrong and why.

Estimated Time Complexity: 2 -3 weeks with 10h per day per analysis type. (with Chuck Clifton) 1.5 – 2 months (without Chuck Clifton)

7.3 Adding Features

Add Second Pass and First Pass Regressions Out to the analysis:
Implementation of SP and Rout should be done within 2 weeks, but with additionally 1 month per analysis for testing and debugging.

Further smaller features, as already mentioned above: 1 month.

Thank You

Dr. Anke Holler and Dr. Markus Demleitner for supervising the project,
their support and help in Python problems.

Dr. Lisa Irmen for giving me an insight in experimental psychological methods
and her support regarding all psychological related information.

Charles Clifton for being always a help. And digging into the problems with Eyedry and investing so much time at a point where I really was lost.
As well as Jeffrey D Kinsey and Patrick Sturt for their support.