An informatics system for exploring eye movements in reading

by

Siliang Tang BSc.

A thesis presented in fulfillment of the requirements for the Degree of

Doctor of Philosophy

Supervisor: Prof. Ronan Reilly

Department of Computer Science

National University of Ireland, Maynooth

Maynooth, Co. Kildare, Ireland

October, 2011

ACKNOWLEDGEMENTS

It was already four years since I arrived in Ireland. In this beautiful island, I received many friendships, many helps, many happy times as well as many memorable stories. I still clearly remembered the days when first time I took the long flight, first night in Maynooth, my first lab demonstration, and etc. These four years give me so many irreplaceable experiences that became the most important and memorable phases in my life. I would like to give my genuine thanks to all my friends and everyone who helped me during these years, although many of them have already left this place for good, they are still in my heart and will be there forever.

Firstly my deepest gratitude must go to my supervisors, Professor Ronan Reilly of NUI Maynooth, and Professor Ralph Radach of Florida State University for their continued encouragement and invaluable suggestions during every stage of this research. Without their knowledge and assistance this study would not have been successful.

I would also like to express my gratitude to Professor Zhaohui Wu, Professor Yue Chen and Dr, Jianhua Yang in Zhejiang University who had influence on my decision to devote myself for science, look for researcher career.

I gratefully acknowledge the funding sources that made my Ph.D. work possible. I was funded by the John and Pat Hume Fellowship for my first three years and was then partial financial by the Reading Center of Florida State University, Tallahassee and RWTH Aachen University for my forth year. Especially thanks to Dr. Christian Vorstius and Matthew Solomon of Florida State University for their help on font experiment data collection and for entertaining me in Tallahassee.

I am deeply indebted to all the staff and students in the Computer Science building, especially to my close friends and colleagues Fangli Ying, Hao Wu, Inthraporn Aranyanak, Jian Wang, Jianghua Zheng, Niall Murphy, Patrick Hynes, Yanpeng Cao and their friends Tina, Lei Pan. Thanks for their abundantly support and consideration on my life as well as this project.

I would like to thanks Ashley, Chryistal, Cory, David, Huang Gang, Jialing Long, Smell, Xin Jin, Yu Dong and their friends Bo Chu, Kai Shong, Lei Yu. They are my house mates in 37 Moyglare Village, who provide me a pleasant living place which makes me feel like real home.

Finally I would like to give my thanks to my family members, and my fiancée Yifei Fu for supporting and encouraging my decision to pursue this degree. Their understanding and endless love, through the duration of my studies, without them this thesis would not have been possible at all.

Thank you all!

ABSTRACT

Eye tracking techniques have been widely used in many research areas including cognitive science, psychology, human-computer interaction, marketing research, medical research etc. Many computer programs have emerged to help these researchers to design experiments, present visual stimuli and process the large quantity of numerical data produced by the eye tracker. However, most applications, especially commercial products, are designed for a particular tracking device and tend to be general purpose. Few of them are designed specifically for reading research. This can be inconvenient when dealing with complex experimental design, multi-source data collection, and text based data analysis, including almost every aspect of a reading study lifecycle.

A flexible and powerful system that manages the lifecycle of different reading studies is required to fulfill these demands. Therefore, we created an informatics system with two major software suites: Experiment Executor and EyeMap. It is a system designed specifically for reading research. Experiment Executor helps reading researchers build complex experimental environments, which can rapidly present display changes and support the co-registration of eye tracking information with other data collection devices such as EEG (electroencephalography) amplifiers. The EyeMap component helps researchers visualize and analysis a wide range of writing systems including spaced and unspaced scripts, which can be presented in proportional or non-proportional font types. The aim of the system is to accelerate the life cycle of a reading experiment from design through analysis.

Several experiments were conducted on this system. These experiments confirmed the effectiveness and the capability of the system with several new reading research findings from the visual information processing stages of reading.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Researchers have been using eye trackers to explore aspects of eye-movement in reading for a very long time (Huey, 1901, 1908; Javal, 1879). Reading research with eye tracking now is entering its fourth era (Rayner, 1998). Many computer applications have emerged to help reading researchers to design experiments, present text stimuli and process the large quantity of numerical data produced by the eye tracker. Figure 1-1 depicts the typical reading experiment life cycle, from the theory or model through to the result that helps refine it. As shown in Figure 1-1, computer applications are involved in almost every step of a reading experiment. However, there are still many gaps between the outputs of the applications and what reading researchers want. Researchers therefore often have to develop their own tools to fill the gaps.

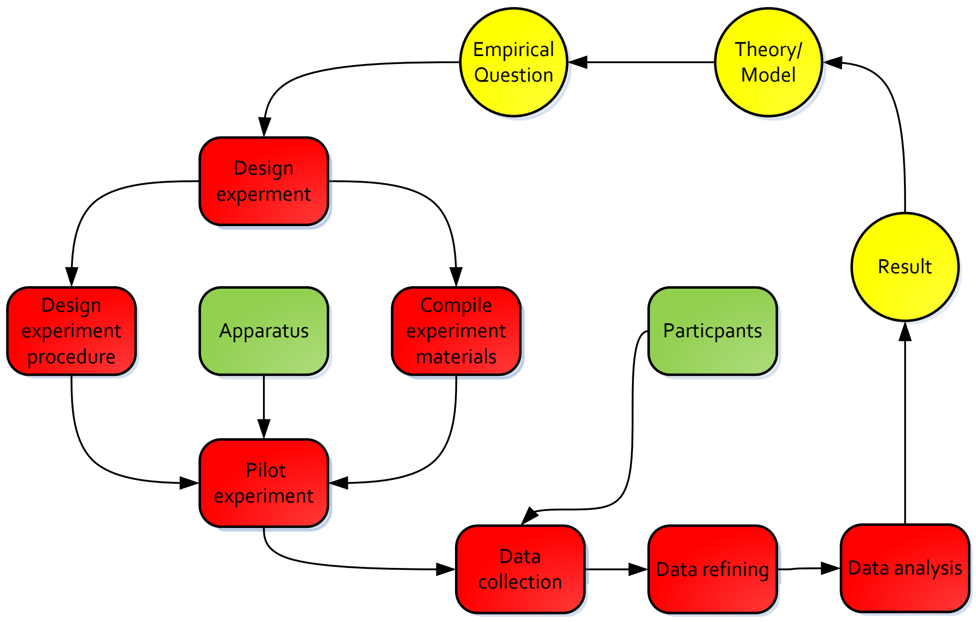


Figure 1-1: Reading experiment life cycle

1.1. Focus of Research

This thesis describes an informatics system which was designed specifically for reading research. It is supported by two major software suites: an experiment executor (ExpExec) and a data visualisation tool (EyeMap). ExpExec helps reading researchers to build an experimental environment, which can rapidly present a variety of display changes and can support the co-registration of eye tracking data with other data collection devices such as an EEG amplifier. EyeMap helps researchers visualise and analyse a wide range of writing systems including spaced and unspaced scripts, which can also be presented in proportional or non-proportional font types. The goal of the system is to accelerate the lifecycle of a reading experiment from the design stage through to the data analysis stage. Several experiments were conducted on this system and will be described in later chapters of the thesis.

1.2. The Progress of the System Development

When the system was first proposed, we were interested in the effects of luminance variation of text on the reading process. The experiment involved the rapid random modulation of the luminance of the text, and measuring the impact, if any, this might have on EEG recordings from the subject. Although there are many off-the-shelf commercial software products, none were applicable to this experimental design, because of two major challenges. The first challenge was rapidly presenting experiment stimuli text at different luminance levels. The second was the precise synchronization of two different data streams, one from the eye tracker and another from the EEG device. We therefore decided to implement an extensible software system with an experiment presenter (the Experiment Executor, or ExpExec) and multi-source data collectors designed specifically for reading and especially its low level vision aspects.

The first implemented part of this system was the stimuli presenter, which used assembly-language-like scripts to control the experiment sequence. This version was used to conduct the so-called “shimmer” experiments. The experimental data from shimmer experiment was first analyzed using SR DataViewer (SR Research Ltd., 2011c). However, during the analysis we found some word segmentation issues with the DataViewer and realized the importance of a reliable eye movement data analyzer in the life cycle of experiment data management. The problem with most commercial eye movement data analysis software, such as DataViewer is that they are designed to be general purpose. Therefore, low-level fixation events are at the center of their analysis. However, for a reading researcher, the most interesting story is mainly at the word level. They need to extract information from fixation reports and their experimental materials about words, such as word length, word frequency, first fixation on word, and gaze duration on the word. Clearly there is a gap between what these general purpose analyzers provide and what researchers need. The analyzers do not fully utilize the properties of the experimental materials. For a reading experiment, they cannot incorporate information such as font type, font size, and text into their analyses. Researchers have to develop their own tools to fill in this gap or sometimes do it manually.

Therefore we started to develop EyeMap in the second year of the project. The final goal of EyeMap was to seamlessly integrate reading materials with fixation data for visualization and export for further analysis. EyeMap can now do many things that were originally handled manually by reading researchers. For example, it automatically detects word boundaries and generates many variables that directly relate to reading research. The implementation of EyeMap started with the creation of a device-independent XML data format to store the eye movement data. The definition of experimental text material is described in a HTML file, which can be viewed in any browser. Then a data parser was created to convert EyeLink (SR-Research Ltd., 2011) ASCII outputs into our XML format. The visualizations showing fixations, saccades, and eye movement playback were then implemented. Development continued with variable export, fixation report generation, and finally word report generation. The development of EyeMap is on-going and will continue for the foreseeable future. It is been used by many reading research groups including the Reading Center at Florida State University, Tallahassee, RWTH Aachen University, MARCS Auditory Labs at the University of Western Sydney, the Psychology Department at the University of Sydney, the Linguistics Program of Northeastern University.

At the same time, in order to make ExpExec more usable, we developed a high-level language extension for the original assembly language system. A C scripting engine was designed and implemented to convert a C-like scripts into our original assembly scripts. Meanwhile, Data Collector, another important part of the system, was designed. The Client (Data Agent) Server (Multi-Thread Collector) model was used in the Data Collector design. The Data Agent was used to extract and normalize the data from data source, while the Multi-Thread Collector received data connections from the Data Agents, and finally synchronizes data from multiple sources. Some C++/Delphi code templates for the Data Agent and Multi-Thread Collector were completed. We then created several applications based on the templates to support Eye trackers including SR EyeLink (SR-Research Ltd., 2011) and Tobii (Tobii Technology, 2011); as well as several EEG devices including BrainVision (Brain Vision LLC., 2011), MindSet 24R (Biofeedback Instrument Corporation, 2011), NeuroScan (Compumedics Neuroscan, 2011), and ActiveTwo (BioSemi Instrumentation, 2011). A preliminary experiment that combined eye movement recording with an EOG (electro-oculographic) signal was successfully preformed.

So far, improvements and bug fixes of the system are still in progress. I plan to continue updating the system by stabilizing existing modules and creating new functions and further promote this system by conducting and publishing more experiments using it.

1.3. Outline and Contribution of the Dissertation

This dissertation consists of three parts.

Part I. Background

The first part of the thesis starts with the discussion of the three indispensible components of a reading experiment: the Eye tracking device, the experiment control software, and the data analysis and visualization tool. Part I introduces eye trackers designed for use in reading and is followed by a critical review of several candidate commercial trackers that can be used in reading experiments. Then I enumerate existing software packages for reading experiment design and eye movement data visualization and analysis, providing more detail on the more well-known tools on the market.

Chapter 2: Eye Tracking Devices for Reading Research

Chapter 3: Reading Experiment Design and Execution Applications

Chapter 4: Reading Data Analysis and Visualization Tools

Part II. System Framework and Implementation

The second part describes the overall architecture of the informatics system. The system covers almost every aspect of the software demands of a reading experiment from experiment building, stimuli presentation, data collection, data visualization and analysis. This part also includes implementation details and observations on each feature of the two major software suites in the system: Experiment Executor and EyeMap.

Chapter 5: Experiment Executor: A Reading Experiment Builder

Chapter 6: EyeMap: An Analysis and Visualization Tool for Eye Movement Data in

Reading

Part III. Experiments using the system

The final part provides a description of several experiments designed and executed within our system. They comprise a reading experiment using our shimmer paradigm, and a comparison study of reading serif and san-serif fonts
Chapter 7: Unspaced English and Chinese Reading with Shimmer as Boundary
Chapter 8: Font Study

The last chapter summarizes the contributions of the dissertation and provides conclusions drawn on the basis of the work.

The most essential contributions of this dissertation can be summarized in two parts. First, we created an informatics system, with solutions to solve several existing problems in conducting reading research. The system covers almost every aspect of the software demands of a reading experiment. Second, several new reading experiments were performed using novel technologies which were developed as part of this thesis, such as the shimmer paradigm, and co-registration of eye Movements and EEG. Several interesting results in the early stages of visual information processing in reading were produced, which it is hoped will contribute to the development of current reading models. The last two parts of the thesis represent an iterative synergy between software development and its testing through the development of various empirical experiments.

Part I. Background

Chapter 2

Eye Tracking Devices for Reading Research

2.1. Basics of Eye Movements in Reading

While reading, the typical reader of an English text makes a series of discrete jumps (or saccades) across the line of text of about 8 character widths in amplitude, pausing to fixate for around 220 milliseconds. About 90% of eye movements are from left to right, with the remainder moving against the flow of the text in what are called regressive movements. These parameters will vary depending on the writing system. The most significant change is to the average saccade amplitude. In Thai, for example, it tends to be around 5 characters and for Chinese between 2 and 3. These changes reflect changes in the mean word length in these writing systems. Interestingly, other parameters such as the proportion of regressions and mean fixation duration tend to be very similar.

The main research tool for studying reading behaviour is the computer-based eye-tracker. Using current eye tracking technology it is possible to measure the location of a fixation to a fraction of a character width and its duration to within 0.5 of a millisecond. Probably the most powerful paradigm afforded by this technology is the eye-movement contingent display change. It is possible to make changes to what is being read whilst the eye is in flight, which is typically for about 20 milliseconds. Note that during a saccade, no useful textual information can be picked up from the display. Thus, we can alter what the reader is able to preview in the parafovea prior to launching a saccade towards a given word and measure what effect this has on subsequent reading times on that word.

The programme of research into modelling eye movement control in reading over the last 20 years has been remarkably successful (Engbert, Nuthmann, Richter, & Kliegl, 2005; Reichle, Rayner, & Pollatsek, 2003; Reilly & O'Regan, 1998; Reilly & Radach, 2003, 2006). This is in large part because factors that control eye movements in reading are, on the whole, relatively low-level. Oculomotor factors predominate and the higher-level factors that do play a role (e.g., word frequency, word predictability) are readily quantifiable and easily incorporated into a computational model.

The family of models so far developed has successfully accounted for a range of reading data. To varying degrees they can predict the normal distribution of landing sites on words of varying length; the systematic “range error” that causes these distributions to shift as a function of how near to the word the eye is launched (more of which later); the robust effect of word frequency whereby words that occur frequently in the language are recognised more rapidly and fixated for a briefer durations; spill-over effects, where processing of word n effects n+1 and vice versa (depending on the theory motivating the model); and skipping of certain classes of word.

2.2. Eye Tracking Techniques

Reading experiments require tracking the reader’s eye with high temporal and spatial resolution. Many commercial eye-tracking devices were developed over the last 30 years. They were constructed to measure the eye's rotational position in many different ways. Pupil diameter and head position are also measured as additional variables by some tracker systems. However, not all the eye trackers on the market are suitable for reading research. An ideal reading eye tracker should be able to obtain data samples at a high frequency (250Hz and more) with real-time response (i.e., delays in the data stream of the order of less than 1ms). It must have very good spatial resolution and high accuracy (both less than 0.1 degree). Finally, the data recording device should be unobtrusive making the subject comfortable enough for long recording sessions.

In order to achieve the above-mentioned objectives, the video oculography (VOG) tracking techniques are introduced by tracker manufacturers. The techniques are based on reflected light using a high-speed camera to capture the movements of readers' eyes.

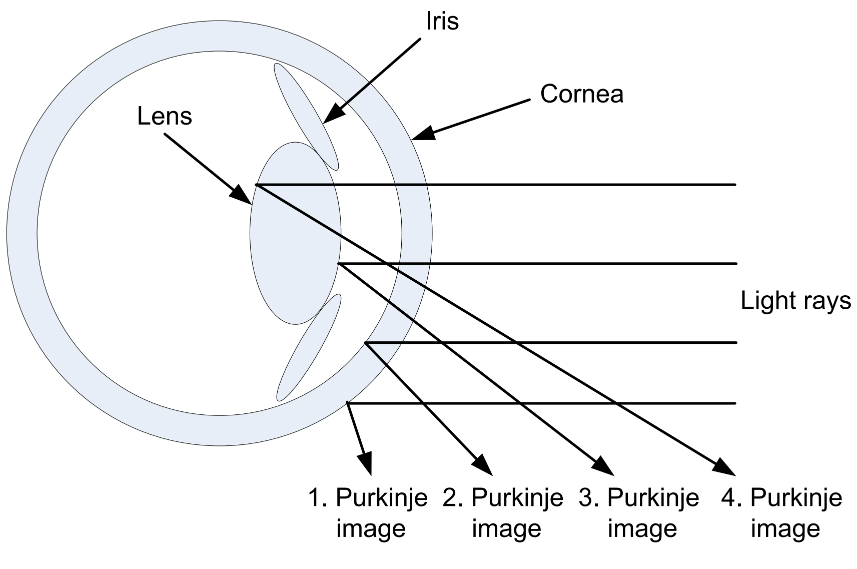


Figure 2-1: The four Purkinje images are reflections of incoming light on the boundaries of the lens and cornea (Glenstrup & Engell-Nielsen, 1995)

As shown in Figure 2-1, when light (usually infrared light) is shone into the user's eye, four reflections, called Purkinje images, occur at the boundaries of the lens and cornea. Of the four Purkinje images, the two brightest reflections are the first and the fourth. VOG eye trackers typically use the first Purkinje image and the centre of the pupil as features to track over time. A more sensitive type of eye tracker, the dual-Purkinje eye tracker (Crane & Steele, 1985) uses reflections from the front of the cornea (first Purkinje image) and the back of the lens (fourth Purkinje image) as features to track. VOG trackers are currently the most popular tracking devices for reading research. In most cases, two or more sets of infrared LEDs are placed near the eye to generate corneal reflections.

2.3. Commercial Eye Trackers in Reading Research

Many commercial eye trackers have been released in recent years. Some are designed especially for reading research, such as SR Research EyeLink series (SR Research Ltd., 2011c) and SMI iViewX series (SensoMotoric Instruments GmbH, 2011a). Some are designed for neuroscience research, such as Chromos Vision (Thomas RECORDING GmbH, 2011), while most of the others are built for usability research, such as the Tobii series (Tobii Technology, 2011) and ASL series. We will discuss further some commercial eye tracking products which can be used in reading research area in this section. This review is based on information available on official web pages of products and (for some devices) from their manuals. Therefore, not all claims could be checked for their accuracy.

2.3.1. SR Research EyeLink Series

SR Research produces high speed (up to 2000Hz sampling rate), high resolution (around 0.01°) eye tracking equipment for a wide variety of research applications. Its trackers are the most widely used among reading research groups. Its head mounted product is EyeLink II which has two cameras mounted on a helmet to capture both of the user's eyes. It provides a 250 Hz (both eyes tracked) or 500 Hz (one eye tracked) sampling rate. The measurement accuracy is 0.5°, and the resolution is 0.01°. EyeLink 1000 is its remote solution, with a head support, EyeLink 1000 has the same measurement accuracy and resolution as the head-mounted EyeLink II. However, it provides 2000Hz (monocular) or 1000Hz (binocular) sampling rate and very simple participant setup. The EyeLink system uses a combined pupil and corneal reflection eye movement detection algorithm. The outputs are the time-stamped eye position of X–Y screen coordinates with mean pupil diameter. Saccades, fixations, and blinks events are detected automatically.

2.3.2. SMI: iViewX Series

SMI produces desktop mounted solutions and its products are also quite suitable for most reading psychology studies. Desktop mounted RED 500 has a binocular sampling rate up to 500Hz, and the iViewX Hi-Speed solution (with head support) can track monocular at 1250 or 500Hz. The measurement accuracy of iViewX Hi-Speed is 0.25° to 0.5°, and its spatial resolution is less than 0.01°. The system uses a combined pupil and corneal reflection eye movement detection algorithm. SMI iViewX system uses nine dots in the calibration as default. The outputs generated consist of a timestamp, eye position in X–Y screen coordinates and pupil size. Saccades, fixations, and blink events are also detected automatically. Updated data for events with continuing fixation can be sent at set intervals.

2.3.3. Tobii TX300

Tobii (Tobii Technology, 2011) is an eye tracking company is mainly focused on market research and usability applications. Therefore, most of Tobii's products use relatively low-speed cameras (less than 200Hz). However, its latest product TX300 is designed for reading psychology. TX300 is a desktop- mounted tracker which has a 300 Hz sampling rate (binocular). The measurement accuracy of TX300 is around 0.5°, and the spatial resolution is about 0.05°. The system uses an embedded eye tracking processing unit. The outputs generated consist of timestamp, eye position of X–Y screen coordinates and pupil diameter.

2.3.4. Fourward Technologies: DPI Eye tracker Gen 6

The Dual-Purkinje-Image (DPI) eye tracker Gen 6 from (Fourward Technologies, 2011) is a good example of a tracker that uses two Purkinje image (first and fourth). The DPI Eye Gen 6 tracker has a 400Hz sampling rate and results in good measurement accuracy (less than 0.01°) and resolution (less than 0.01°).

2.4. Conclusion

Although all of the above-mentioned trackers are suitable enough for most reading research purposes, our recommendation is, for these experiments needing a high sampling rate, trackers like EyeLink 1000 which has a 2000 Hz sampling rate may satisfy their requirements. While Dual-Purkinje-Image trackers like DPI Eye tracker Gen 6 could offer a better spatial resolution for some studies. Finally for these studies needing a high tolerance for head movement, they may benefit from the use of the Tobii tracker series.

Chapter 3

Reading Experiment Design and Execution Applications

In last section we discussed several commercial eye trackers and described their technical specifications. In order to make appropriate and effective use of these eye-tracking devices in a psychology study, many research groups and tracker manufacturers have developed their own experimental stimuli presentation applications for the above-mentioned trackers. Many of them are designed for special purposes and are not publically available. In the following section, we will investigate and discuss several experimental design tools which can be used in conjunction with these trackers.

3.1. General Experiment Design Tools in Reading

To create a general tool for convenient experimental design requires a deep understanding of both tracking device control as well as experimental psychology procedures, therefore most experiment design tools are commercial products which are created by tracker manufacturers or third party companies. Compared to applications designed only for a specific area, these tools can create and manage many experimental projects and design and modify them in a convenient way. Some integrate experimental design, data collection, and data analysis functions together into one software suite. Some present experiment stimuli on a timeline and arrange them in user-defined order. Some even use a visual drag-and-drop interface that allows users to manipulate experimental elements graphically and create their own experimental procedure and logic with loops and condition controls using an icon-based language.

In the following sections, several general presentation application-creation solutions used in reading research will be introduced. The software reviewed here are: Tobii Studio (Tobii Technology, 2008), Experiment Center from SMI (SensoMotoric Instruments GmbH, 2011b), Experiment Builder (SR Research Ltd., 2011a), E-Prime (Psychology Software Tools, 2011a), Presentation (Neurobehavioral Systems, 2012), the MATLAB-based Psychophysics Toolbox (Cornelissen, Peters, & Palmer, 2002) and etc. As mentioned before, some observations are from product manuals therefore not all claims could be checked for accuracy.

3.1.1. Commercial Solutions from Tracker Manufacturers

As described in section 2.3, the main tracker manufacturers in reading research area are Tobii, SMI and SR Research. Their corresponding commercial solutions for general experiment creation are Tobii Studio (Tobii Technology, 2008), Experiment Center (SensoMotoric Instruments GmbH, 2011b) and Experiment Builder (SR Research Ltd., 2011a).

Tobii Studio integrates experimental design, data recording, data replay, and data visualization functions. It is intuitive in stimulus arrangement and diversified in the stimuli types that can be used. As shown in Figure 3-1, stimuli can be placed on a timeline based on order of their appearance. Users can insert or rearrange them by drag-and-drop. PDF Element and Questionnaire are also implemented as a type of stimulus. Another good feature of Tobii Studio is its project and participant management. The project management feature allows users to import, export, backup, and merge experimental projects, while participant management allows users to record, import and export participant information such as gender, age range for further data classification. Tobii Studio can be used for many psychological experiments based on videos and/or pictures. However it has very limited support for reading studies, since it can only present text within a small range of font type, font size, and color in the center of the screen. Otherwise, you must convert your text stimuli into PDF elements. There is no other support for text stimulus in data analysis.

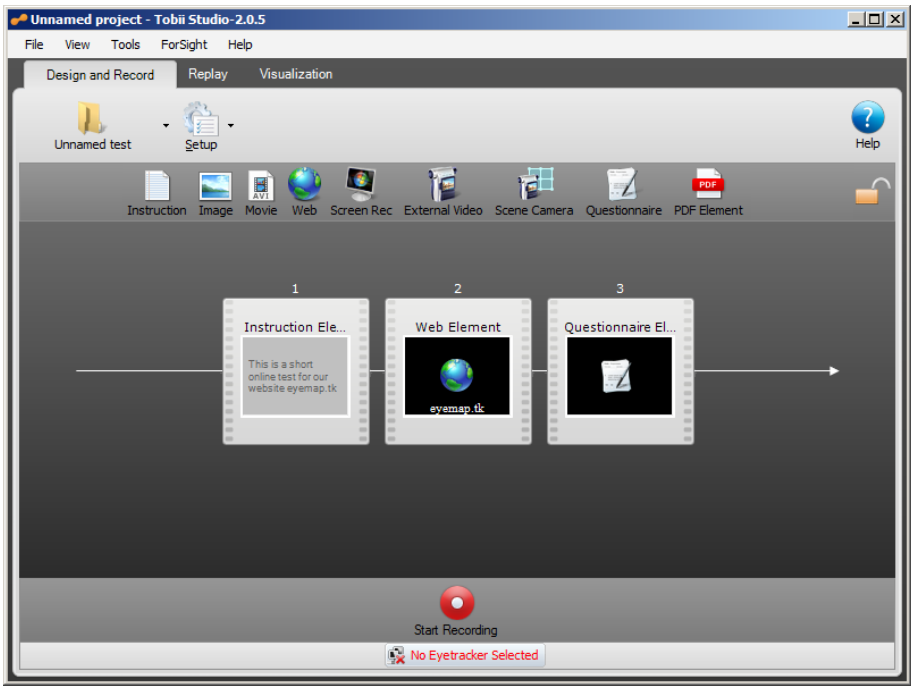


Figure 3-1: Sample experiment by Tobii Studio 2.0.5

Experiment Center (SensoMotoric Instruments GmbH, 2011b) is another commercial experiment design and data collection tool produced by SMI for their trackers. It has similar experimental design functions as Tobii Studio. Experiment Center supports rich text presentation as a basic stimulus type. As shown in Figure 3-2, it provides a what-you-see-is-what-you-get ("WYSIWYG") text editor window which allows users to create their text stimuli. Moreover, areas of interest (AoIs) for words, sentences and paragraphs in created text stimuli can be automatically generated by SMI analysis software BeGaze.

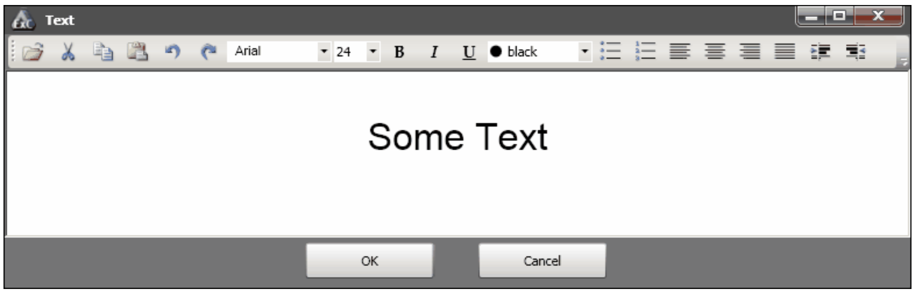
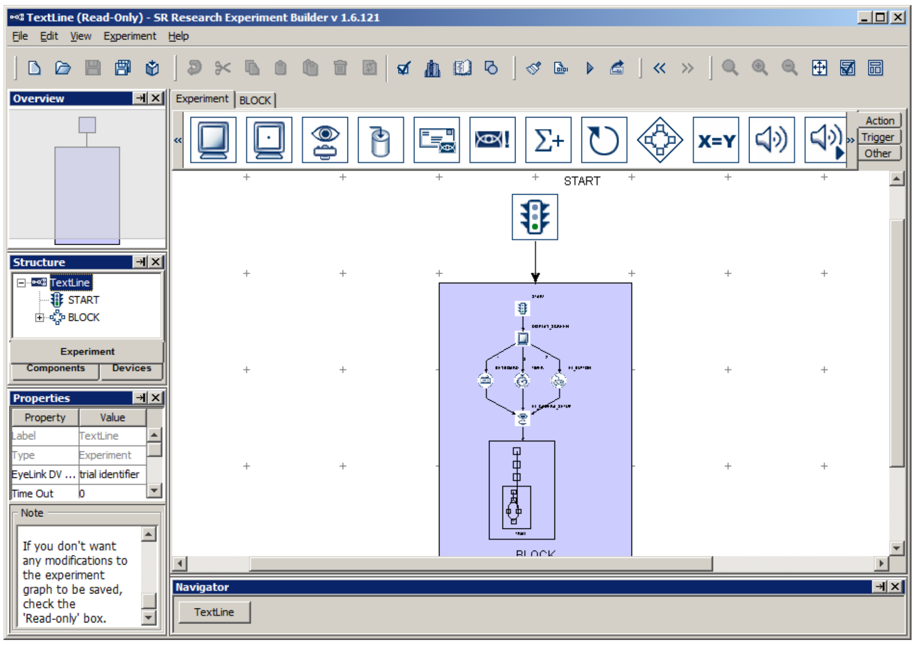


Figure 3-2: Text editor window provided by Experiment Center

The most widely used high-level eye tracking experiment creation tool for reading is SR Research Experiment Builder (SR Research Ltd., 2011a) which works in conjunction with SR eye trackers. A supposed advantage of Experiment Builder is its icon-based language. This technique is also called visual programming or diagrammatic programming (Bragg & Driskill, 1994). As shown in Figure 3-3, experimental procedures are separated and encapsulated into many actions and triggers. Action components instruct the computer to do something. While trigger components are conditions which control the experiment flow.

One or multiple actions and triggers can be added to a sequence block, which can be repeated several times. An external data source is attached to each sequence block as different parameters. A sequence is also an action component, therefore it can be nested. This kind of recursive definition of the sequence creates a manageable hierarchical organization of experiments. Furthermore, during the experimental design, sequences can be exported or imported between different projects. This also facilitates and simplifies the experiment creation.



Property	Value
Label	TextLine
Type	Experiment
EyeLink DV ...	trial identifier
Time Out	0

Note

If you don't want any modifications to the experiment graph to be saved, check the 'Read-only' box.

Figure 3-3: An experiment designed by Experiment Builder with visual programming technology

In addition to programming through the visual programming Interface, Experiment Builder also allows users to create custom visual components using the Python (van Rossum, 2012) script. The custom component (class) supports attributes and methods definition, and allows bi-directional data exchange between other visual components. Therefore Experiment Builder can do more sophisticated flow control and calculations by this Python extension.

For reading experiments, Experiment Builder not only supports flexible experiment procedure design, but also provides good support for text stimuli presentation. Compared to the text editor in SMI Experiment Center, a similar but more powerful WYSIWYG text editing environment, Screen Builder, is provided by Experiment Builder. Screen Builder behaves like a drawing board onto which text stimuli and various types of other graphic resources (e.g., images, drawings, or videos) can be added. It also allows users to specify the movement pattern of these graphic resources. Additional utilities (e.g., interest areas, drawing grid) are provided for the ease of editing and further data analysis.

With those commercial experiment creation tools offered by tracker manufacturers, researchers can design and perform their reading experiments in a convenient and intuitive way. However their disadvantages are also obvious. First, most of these tools are quite expensive, and they are usually designed for particular tracker series from the same company. Second, their functions are only focused on limited research areas. For example, the mainstream products of tracker companies like Tobii and SMI are focused on web usability and market research experiments and thus have limited scope for interaction with participants. Therefore their experimental design tools cannot produce interactive experiments that use gaze-contingent paradigm or any similar interactive design.

3.1.2. Commercial Solutions from Third Party

In addition to those manufacturer solutions, there are also many other third party commercial products, which help researchers to design their experimental psychology procedures and present visual stimuli. Most of them are professional psychological experiment presenters, such as, E-Prime (Psychology Software Tools, 2011a) and Presentation (Neurobehavioral Systems, 2012).

E-Prime is a software applications suite for conducting psychological and neuro-scientific experiments, developed by (Psychology Software Tools, 2011a). Although E-prime is not designed particularly for reading research, it has good support for wide range of reading eye trackers including EyeLink, SMI and Tobii trackers. It has also opened its interface to third party tracker manufactures such as Tobii (Psychology Software Tools, 2011b), SMI (Vicente-Grabovetsky, 2010), and EyeLink (E-Prime sample can be found in EyeLink Software Development Kit) (2011b).

E-Prime also has a powerful text stimuli presentation component which is well suited for many reading experiments.

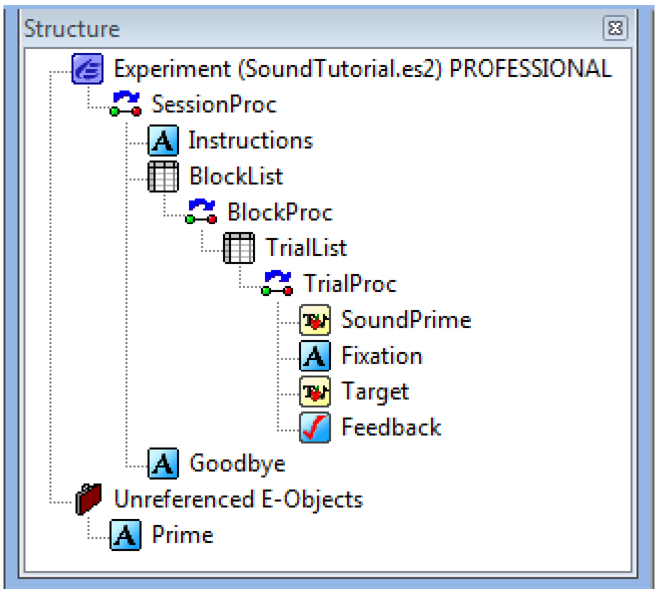


Figure 3-4: A sample experiment structure.

In experimental design, E-prime allows user to create and define a Block Procedure, which have a similar function as the sequence block in Experiment Builder. In Block Procedure, drag-and-drop experimental components can be placed on a time line, and repeated according to different criteria based on the Block Procedure definition. Similarly, Block Procedures can be nested. Therefore E-prime can cover many aspects of experimental paradigm creation. Figure 3-4 shows an experiment structure with Block Procedures. Another good feature of E-Prime is that it is temporally accurate to within a few milliseconds (average error <= 0.5ms). It also has a series of applications for experimental data management including data merge, filter and recovery.

Another application, which can be used for reading material presentation, is called Presentation (Neurobehavioral Systems, 2012). It is a stimulus delivery and experimental control program designed for behavioral and physiological experiments, especially for neuroscience. Presentation can present stimuli including sounds, images, videos, and even 3D stimuli. Compare to above-mentioned experiment presentation tools, Presentation is more low level. First, the experiment scenario of Presentation is described in its own descriptive language (Scenario Description Language). Moreover, a Presentation Control Language (PCL, one interpreted programming language) is used to control experimental procedures. It can be interfaced with eye tracking hardware using its own eye tracker extensions. The eye tracker extension is also defined by PCL as an eye_tracker type. Theoretically, Presentation can support any tracker that implements its IEyeTracker2 COM interface. So far, SMI and EyeLink tracker extensions for Presentation are already publicly available.

For these third party tools, they are usually designed for wider research applications, and support more than one series of trackers. However, tools like Presentation are not that intuitive as manufacturers' products.

3.1.3. Free or Open Source Solutions

Many free or open-source solutions emerged recently. Because of their openness and transparency, they are now increasingly used in labs of reading research community. The most representative one among those tools is Psychophysics Toolbox (Brainard, 1997).

Psychophysics Toolbox is a free set of MATLAB (MathWorks, 1989) and GNU/Octave (Eaton, 2012) functions for vision research, which makes it easy to synthesize and show accurately controlled visual and auditory stimuli as well as interact with the subjects. To design a reading experiment, it requires MATLAB or Octave programming skills as well as knowledge of the eye tracking toolbox, such as EyeLink Toolbox (Cornelissen, et al., 2002) or Talk2Tobii toolbox (Deligianni, 2009). Therefore it is not as intuitive as other high-level experiment design tools. However, Psychophysics Toolbox has many visual stimuli presentation and control functions which definitely facilitate complex experiment implementation. With the powerful MATLAB/Octave numerical computing support, Psychophysics Toolbox applications are especially suitable for pilot studies, which require more flexible procedure control and complex stimuli presentation.

There are also many Python-based (van Rossum, 2012) open source presentation solutions available, such as Vision EEG (Straw, 2008), and PsychoPy (Peirce, 2007, 2008). The same as Presentation, they are mostly created for building neuroscience experiments. Although, they can be used for reading research, they cannot talk to eye trackers directly. Raw tracker API Python wraps such as PyLink (SR Research Ltd., 2012) could be used to solve this problem.

Compare to commercial solutions that we enumerated in previous sections, open-source tools has many advantages, because they are free to use, open for extension, and could be more flexible than commercial solutions. However, their disadvantages are also obvious. To make fully use their advantages, users have to learn at least one interpreted language (MATLAB, Octave, or Python). This introduces a programming threshold for most psychology researchers.

3.2. Development based on Raw APIs

General experiment design tools are not always capable of covering all the experimental specification. Some researchers therefore create their own eye tracking presentation applications directly from raw tracker APIs.

A raw tracker API is a programming interface to receive eye movement data and events as well as to control the tracker. They are provided by tracker manufacturers within their tracker SDK (software development kit). Many eye tracker manufacturers encapsulate their tracker APIs into a Component Object Model (COM) (Rogerson, 1997) object and can thus be used by a large range of programming languages.

However, tracker companies frequently design their APIs differently due to the different device capacities and potential application area. This presents problems for developers who work with different trackers. They may have to learn new APIs and deal with technical differences when they are migrating their presentation application to another tracker platform.

Chapter 4

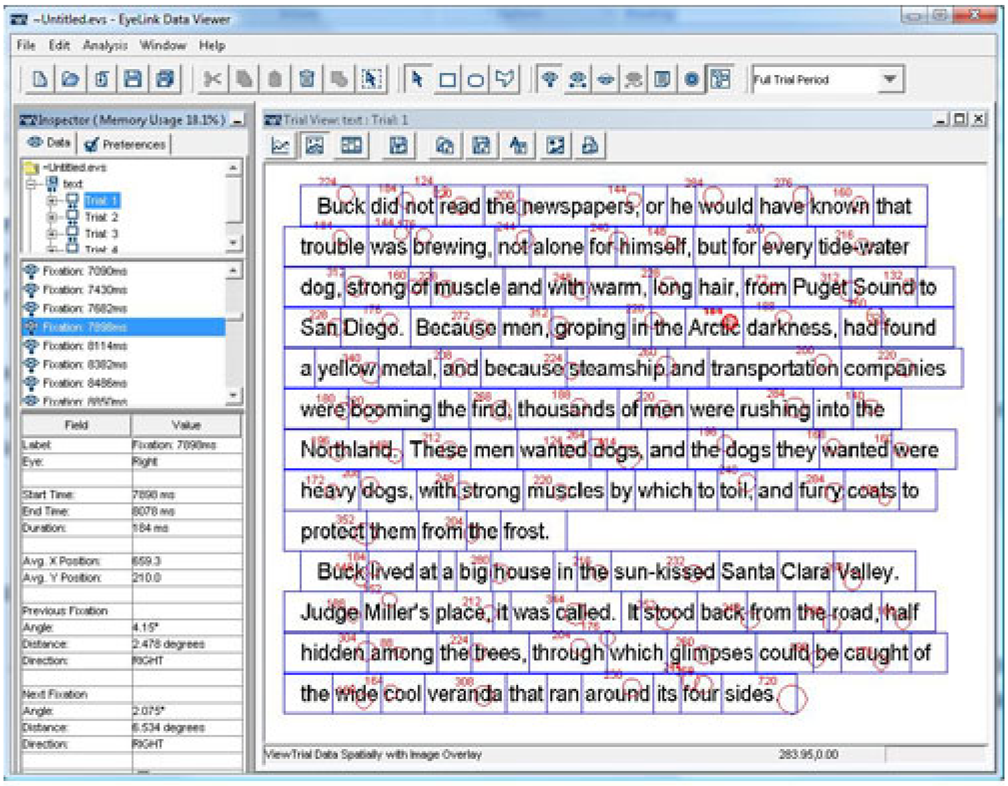
Reading Data Analysis and Visualization Tools

In much of vision research, it has been suggested that visualization tools are necessary for facilitating the understanding of large volumes of data (Stellmach, Nacke, Dachselt, & Lindley, 2010; Vicente-Grabovetsky, 2010) because the meaning of individual eye movements is impossible to interpret according to a fixed set of rules, as there is very little evidence from eye movement behavior *per se* that can help distinguish between an unintentional fixation and a deliberate or purposeful one (Posner, 1980) . However, in studies of reading, words as the object of study tend to constrain eye movements, thus increasing the overall proportion of what we might refer to as "meaningful" fixations. The task of reading is sufficiently different from other visual tasks that quantitative data analysis is possible in reading. Several successful computational models of eye movement control have been developed (Engbert, et al., 2005; Reichle, et al., 2003; Reilly & O'Regan, 1998; Reilly & Radach, 2003, 2006) and serve as theoretical motivation for many empirical reading studies.

4.1. Eye Movement Data Analysis Tools in Reading

In recent years, there has been a steady stream of improvements in the visualisation and analysis of eye tracker data for static 2D stimuli such as text or still pictures (Lankford, 2000; Wills, 1996; Wooding, 2002). However, these tools are usually highly dependent on the tracker system, because of the raw data format and the data-collection method. Furthermore, few tools have the ability to calculate reading related variables for quantitative analysis. For example Tobii Studio can generate statistical charts and visualize eye movement data in both static and animated gaze plots and heat maps. However it does not have any reading-related variable output options. The same problem exists with many other general analysis tools such as BeGaze, GazeTracker, and NYAN (Interactive minds GmbH, 2010). These have more powerful visualization functions than they do quantitative data analysis abilities. Some can only export very basic eye movement events such fixations, saccades and blinks from specific tracker but few of them are designed particularly for reading research. Thus, it is often the case that gaze data visualization software and additional quantitative data analysis tools need to be developed from the ground up for eye trackers used in many laboratories. This state of affairs usually leads to duplication of effort in software development.

One of the most widely used reading data analysis is SR DataViewer. It is commercial software produced by (SR Research Ltd., 2011c) as an important component of its software tool chain. DataViewer allows the user to view, filter, and process EyeLink gaze data. For data analysis, it can generate variables including interest area dwell time, first pass and regressive fixation measures, and so on. Overall, DataViewer is a powerful and professionally developed tool for reading research. For visualization, as shown in Figure 4-1, it provides several different data viewing modes such as eye-event position and scan path visualization, temporal playback of recording with gaze position overlay.



Field	Value
Label:	Fixation: 7898ms
Eye:	Right
Start Time:	7898 ms
End Time:	8078 ms
Duration:	184 ms
Avg. X Position:	659.3
Avg. Y Position:	210.0
Previous Fixation	
Angle:	4.15°
Distance:	2.478 degrees
Direction:	RIGHT
Next Fixation	
Angle:	2.075°
Distance:	6.534 degrees
Direction:	RIGHT

Buck did not read the newspapers, or he would have known that trouble was brewing, not alone for himself, but for every tide-water dog, strong of muscle and with warm, long hair, from Puget Sound to San Diego. Because men, groping in the Arctic darkness, had found a yellow metal, and because steamship and transportation companies were booming the find, thousands of men were rushing into the Northland. These men wanted dogs, and the dogs they wanted were heavy dogs, with strong muscles by which to toil, and furry coats to protect them from the frost.

Buck lived at a big house in the sun-kissed Santa Clara Valley. Judge Miller's place, it was called. It stood back from the road, half hidden among the trees, through which glimpses could be caught of the wide cool veranda that ran around its four sides.

Figure 4-1: EyeLink Data Viewer on viewing a paragraph reading experiment data

Another software EDAS II (Eye movement Data Analysis System II) produced by (Entroware Communication, 2009) is one of the few tools designed for reading research. It provides text and image (for Unicode and multiline text) based reading eye movement data analysis functions, which can be used for fixation sequence analysis and object-based analysis. The analysis of reading data in many word-specific movement measures such as the size of saccades to words, saccade duration, landing position, skipping rate, regression probability, number of passes through a word, total number of fixations, and various fixation-dependent measures (including first fixation duration, second fixation duration, gaze duration, total viewing duration, and more). One of the good features of EDAS II is that it corrects vertical drift for multiline reading. It also supports batch processing for experimental data files and generates an Excel spreadsheet as output. Finally it uses XML for experiment definition, however currently the data source can only be extracted from SR EyeLink proprietary EDF data files.

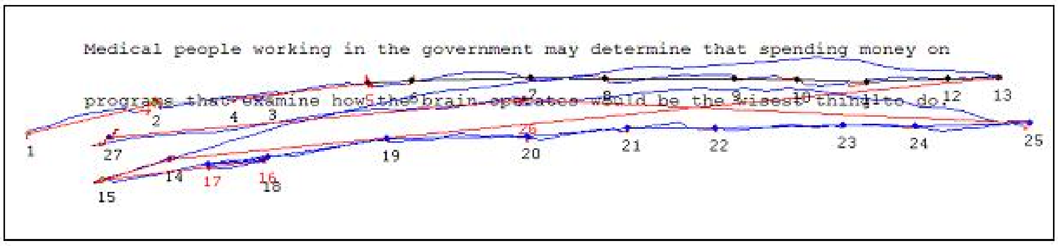


Figure 4-2: Multiple line text analysis

For visualization, as shown in Figure 4-2, EDAS II can plot raw eye movement curves to overlap with the text stimuli. Fixations are plotted as dots with a number indicating their time sequence.

Overall, there are four problems for these reading data analysis tools. First, they tend to have few functions for processing reading data and generating useful variables. Second, Most of them have limited options for text presentation and analysis. Some do not readily support the editing of eye movement data. Third, they are hardware specific and the data are stored in a proprietary format. Finally, they are not freely available, difficult to extend. Purchase and support can be quite expensive.

4.2. Eye Movement Classification Algorithms

Although there are many different data visualization and analysis methods, the eye movement classification algorithms, which determine the saccade and fixation events, are quite limited. In reading data analysis, the performances of eye movement classification algorithms have a significant influence on the data analysis results. There are five major classification algorithms, which are widely used:

1) Velocity-Threshold Identification (I-VT) (Salvucci & Goldberg, 2000)
2) Hidden Markov Model Identification (I-HMM) (Salvucci & Goldberg, 2000)
3) Kalman Filter Identification (Komogortsev & Khan, 2007; Sauter, Martin, Di Renzo, & Vomscheid, 1991)
4) Minimum Spanning Tree Identification (I-MST) (Salvucci & Goldberg, 2000)
5) Dispersion-Threshold Identification (I-DT) (Duchowski, 2007; Salvucci & Goldberg, 2000)

Komogortsev, *et al.* (2010) evaluated and compared the performance of these five algorithms to classify specific oculomotor behaviour. They presented a white single 'jumping point' on the black background with vertical coordinates fixed to the middle region of the screen. They found that the performance of these algorithms varied dramatically even in such simple stimulus evoked task involving a single, common threshold value. Therefore they created a standardized scores system, which selects reasonable threshold value for any eye movement classification algorithms.

However, for most data analysis software, the event detection algorithms were unknown to the public, non-extensible or quite limited in function. For example, Clearview (Tobii Technology, 2004) use I-VT based fixation filter(Larsson, 2010), and its successor Tobii Studio offers two fixation filters: original I-VT '*ClearView*' and another *'Tobii Fixation Filter'*, which is a mixture of I-DT and I-VT (Olsson, 2007). There is no saccade detection mechanism for Tobii analysis software (a saccade is merged into the succeeding fixation. This may be due to the fact that most Tobii trackers use a relatively low sampling rate. While SR DataViewer uses the results from online saccade/fixation classification functions, which is carried out in real time during eye movement recording.

Part II. System Framework and Implementation

Chapter 5

Experiment Executor: A Reading Experiment Builder

As mentioned in Chapter 4, there are two major ways of implementing a stimulus presentation application for reading experiment. The first, which most psychology researchers will choose, is to use general experimental design software. These programs basically do not require users to be skilled programmers. By selecting pre-defined experimental paradigms and using a step-by-step creation wizard, they allow users to create different experiments combined with multiple sources of stimuli such as texts, pictures, audios and even videos quite simply and easily. However the price of this kind of high-level software is a reduction in flexibility. Advanced experimental paradigms with complex experimental procedures cannot be readily deployed by these programs. To overcome this drawback, some software tools have been developed (Psychology Software Tools, 2011a; SR Research Ltd., 2011a; Tobii Technology, 2008) using a visual drag-and-drop interface, with which users can manipulate experimental elements graphically and create their own experimental procedure using an icon-based language. However sometimes, due to the functional limitation of the experimental elements themselves, complex reading experiments, especially low-level vision experiments in which lots of display changes are involved, are not feasible within this paradigm. To fulfill the requirements of more flexibility control, a user has to create presentation applications in standard development tools (such as Microsoft Visual Studio, Eclipse (Eclipse, 2010) or Matlab (MathWorks, 1989) and program them using low-level eye tracker APIs. Although all the eye tracker manufacturers provide programming interfaces for users to access the full functionality of the tracker, these interfaces are encapsulated in many different ways, such as import libraries (LIB), dynamically linked libraries (DLL), component object models (COM) etc. All of these APIs are named differently, thus every time users want to deploy their experiment to a new tracker they have to learn its APIs, which obviously costs time and effort. On the other hand, to implement a new experimental paradigm from a low-level interface, without the help of any high level experimental elements, the user has to learn not only the tracker APIs, but also the graphic APIs and system APIs such as those that control display changes, communicate to input devices, output experimental data to storage and so on. This introduces a very high programming threshold for most psychology researchers. After considering the advantages and disadvantages of both approaches, we decided to create the Experiment Executor (ExpExec), an intermediate way to fill the gap between the high and low levels.

5.1. Development principles of Experiment Executor

ExpExec consists of a scripting engine, a virtual machine, and an embedded function module. It supports C language scripts. User scripts are compiled by the scripting engine, and then executed on the virtual machine. All the functions implemented are encapsulated into the embedded function module and therefore can be called from the scripts directly, while the virtual machine shields the user from the differences between eye-tracker APIs. For example, it supports EyeLink, Tobii trackers and a mouse-based tracker simulator. This means that users can run the same experimental scripts on these tracker systems without changing a line of code. Compared to the high level approach, the ExpExec C script gives users the same flexibility as visual programming. Although lacking the ease of use of a graphical interface, the coding tends to be more efficient for experienced users. While on the other hand, compared to low level API programming, users who have certain programming background or want to learn some programming can work with our script language quite easily without knowing any details of the eye-tracker APIs, graphic APIs, or system APIs. Thus the ExpExec is a balanced solution for a wide range of reading experiments and it works particularly well for exploring low-level vision aspects of reading.

5.2. System Architecture

The scripting engine and virtual machine design gives us a balanced solution for constructing an eye tracking reading experiment with different methods. However, our final goal is to implement a platform for performing various reading experiments, especially to explore the area of low-level reading, and ultimately to cooperate with a range of other data collection devices such as EEG systems.

We have built an experiment building environment, which consists of ExpExec, Data Collector and Data Exporter. As shown in Figure 5-1, all the elements of the environment are presented in a rectangular box. A blue box indicates a software module, a red box represents a physical device, and a yellow box represents an input or output file. The dashed box on the left of the Figure 5-1 is an input data source. Experimental materials are experiment stimuli such as text and pictures. The C scripts file contains the experiment logic and control of the experiment process.

The Eye Tracking Experiment Executor is the key component of our experiment presentation environment. It runs on Windows platform, which consists of the scripting engine, virtual machine, and embedded function modules.

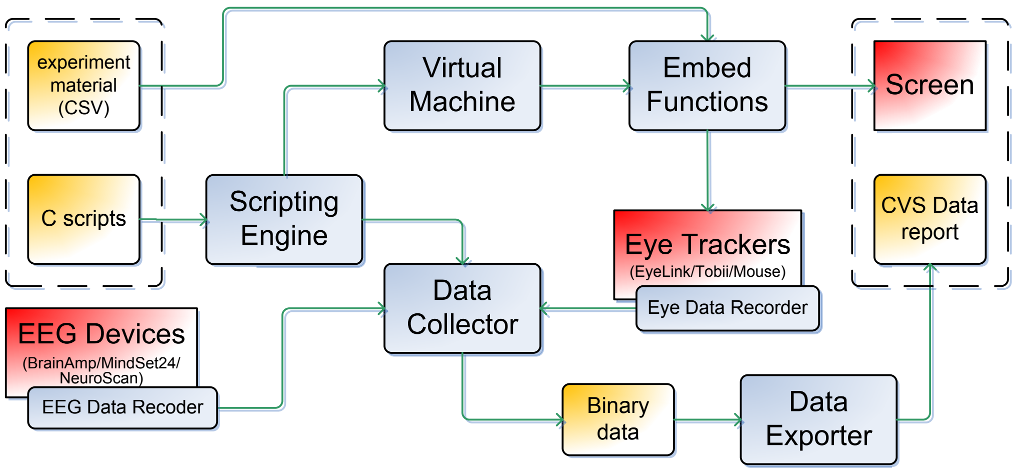


Figure 5-1: Framework of experiment building environment

5.3. Scripting Engine

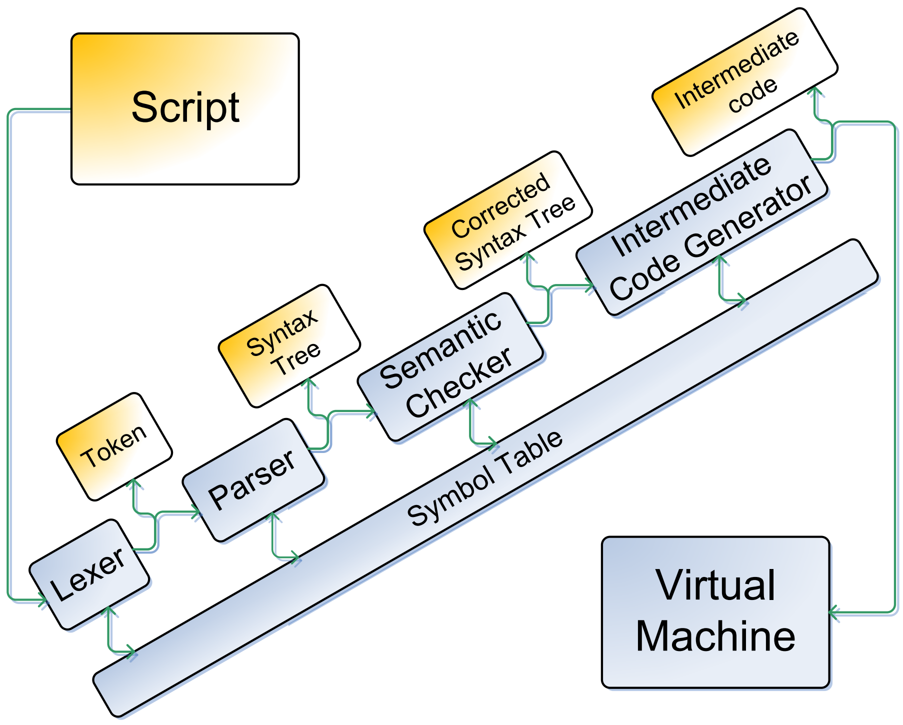


Figure 5-2: Scripting Engine

The ExpExec uses a third party C scripting engine, (see attached DVD folder Chapter 5\ Script engine). It uses a language called min-C, which is a subset of the standard C language (Kernighan, Ritchie, & Ejeklint, 1988). The reason we choose C as the host language is because C is probably one of the most popular programming languages of all time and it has greatly influenced the design of many other modern programming languages. Most researchers doing a science major learn a C-related programming language, so a C scripting engine may re-move some of the burden of learning a new language for most users. Another advantage of using C is that we can take advantage of long-standing experience in building C compilers, which will help speed up our development process.

The min-C scripting engine takes a source script as the input and converts it into intermediate code as output. During compilation, there are four major phases. As shown in Figure 5-2, the source script is firstly passed through a lexical analyzer. It converts the character stream into a token stream by scanning through the input characters sequentially and recognizing "words" in the stream. Then the tokens are sent to the parser. The output of the Parser is a syntax tree which can be used for further correction and optimization. After the scripting engine generates the syntax tree, semantic checking is performed immediately to make sure that not only is the syntax of the program correct, but the statements also make sense. The number of parameters supplied to a function, for example, should be the number this function expects. The major part of semantic checking is type checking: determining the types of expressions and reporting any inconsistencies, such as trying to compare a Boolean to a string or passing the wrong type of argument to a function.

5.4. Embedded Functions

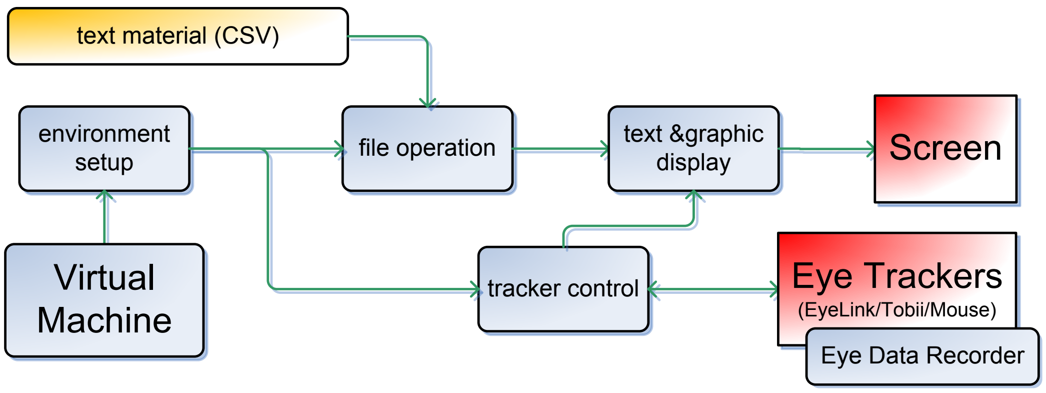


Figure 5-3: Embedded functions

Embedded Functions are an intermediate level between the virtual machine and the actual experiment presentation operations. They are predefined functions for tracker control and graphic display, which shield the device differences from the user and simplify the presentation operations. These functions can be classified into four categories:

1. Functions for Environment setup:

   The experimental environment including the presentation device (such as presenting information in a picture file, or a real monitor) and its parameters (such as full screen display, vertical synchronization, background color, foreground color); eye tracker models (e.g., EyeLink, Tobii, or mouse[1]); and Text presentation environment (e.g., font type, font color and font luminance). These environment parameters are defined as different environment types and can simply be set by our *setenv* function.

Table 5-1: Supported environment types

Environment categories	**Type**	**Function**
Screen related environment variables	DISPLAY 0x000003	Set screen display (SCR(0x000011)/IMG(0x000012))
	FULLSCR 0x000010	Set window displayed in full screen or not (NO/OFF)
	VERTSYN 0x000007	Set display vertical synchronization (NO/OFF)
	BGCOLOR 0x000002	Set background color (RGB in hex)

[1] For testing reasons, ExpExec has a mouse simulation module, which allows the user to run scripts without a real eye tracker. In this case, the mouse is used to simulate the eye.

Tracker related environment variables	TRACKER 0x000009	Set tracker type (EYELINK(0x00000a)/TOBII(0x00000b)/(MOUSE 0x00000c))
Text display related environment variables	FONTSIZE 0x000004	Set font size for current displayed text
	FONTCOLOR 0x00000f	Set font color for current displayed text
	LUMINANCE 0x000005	Set luminance for current displayed text
System related environment variables	REALTIME 0x000006	Set program running in real-time priority or not (NO/OFF)

2. Functions for file operations:

   As shown in the Figure 5-3, file operation functions help users load and manage text materials for further processing. They also help users save their experiment outputs such as button press events to local disk. For example, function *int file_load(char** filename*, char delimiter)* loads a text file from local disk and preserves it in memory as a matrix. While function *char* file_read(int filehandle, int index)* will return the data from the loaded matrix in position *index*.

3. Functions for presentation operations:

   For a reading experiment, most experimental stimuli are text or pictures. The biggest advantage of ExpExec is that it can render these stimuli at a very high speed. With the power of Microsoft DirectX, display changes such as swapping experimental stimuli, or changing the text luminance can be controlled by these presentation operations in a flexible and appropriate way.

4. Functions for tracker control:

   Tracker control is one of the most important parts of the experiment process. Some operations such as drift correction are performed quite frequently in order to get a precise starting gaze location. On the other hand, although ExpExec is only presentation software, we may still need real time eye position information to feed back to display control functions in many display change experiments (such as the boundary-paradigm Rayner, 1975). The main difficulty with tracker control is there are many different trackers on the market and they all have different protocols for controlling the tracker status and getting at the real-time data. The current version of ExpExec now has three independent modules (SR EyeLink, Tobii and mouse tracker) for tracker control. These include the functions for setting up tracker status (e.g., open connection, calibration, validation, drift correction and etc.), and getting real-time eye movement data. Other tracker modules can be plugged into our system quite easily, as long as they implement all the functions that mentioned in our Experiment Executor Function Guide (see attached DVD, folder Chapter 5\User Manual). Figure 5-4 presents some sample code of ExpExec.

```
/* Experment on Font Study
 *
 * author: siliang tang
 * email: stang@cs.nuim.ie */

#include "windows.h"
//all windows api function is supported
#include "Expsim2.h"
//read the function guide for detail
#define TRIAL_NUM 192

void main() {
    //check edf data file
    long FileData;

    if (FindFirstFile("*.edf", &FileData) != INVALID_HANDLE_VALUE) {
        MessageBox(0, "EDF file already collected!", "Error", MB_OK);
        return;
    }

    //select tracker
    if (MessageBox(0, "EyeLink(OK) or Mouse(Cancel) ?", "Eye tracker
select", MB_OKCANCEL
         | MB_DEFBUTTON1) == IDOK) {
        setenv(TRACKER,EYELINK);
    } else setenv(TRACKER,MOUSE);

    //load data sentences
    int fh_sentence;
    fh_sentence = file_load("sentences.csv",',');

    //initialize the environment
    setenv(FULLSCR,ON);
    setenv(VERTSYN,ON);
    setenv(DISPLAY,SCR);
    setenv(T_VOLUME,OFF);
    setenv(BGCOLOR,0xffffff);

    setenv(FONTCOLOR,0);
    setfont("Arial");
    setenv(FONTSIZE,19);

    text_align(CENTER,"Waiting for connection...");
    disp_flip(1);

    tracker_connect(ON);
    tracker_setup();
```

Figure 5-4: Sample code of ExpExec

5.5. Data Collector

The main difficulty with multiple source data collection is data synchronization.

In the data collection process, there might be, for example, two types of data streams that need to be synchronized, one from the EEG device, another from Eye tracker. However, different EEG and eye tracking devices have different application programming interfaces and they usually work with different sampling rate. This requires our data collection framework to be flexible and extensible enough to support various experiment devices and setups.

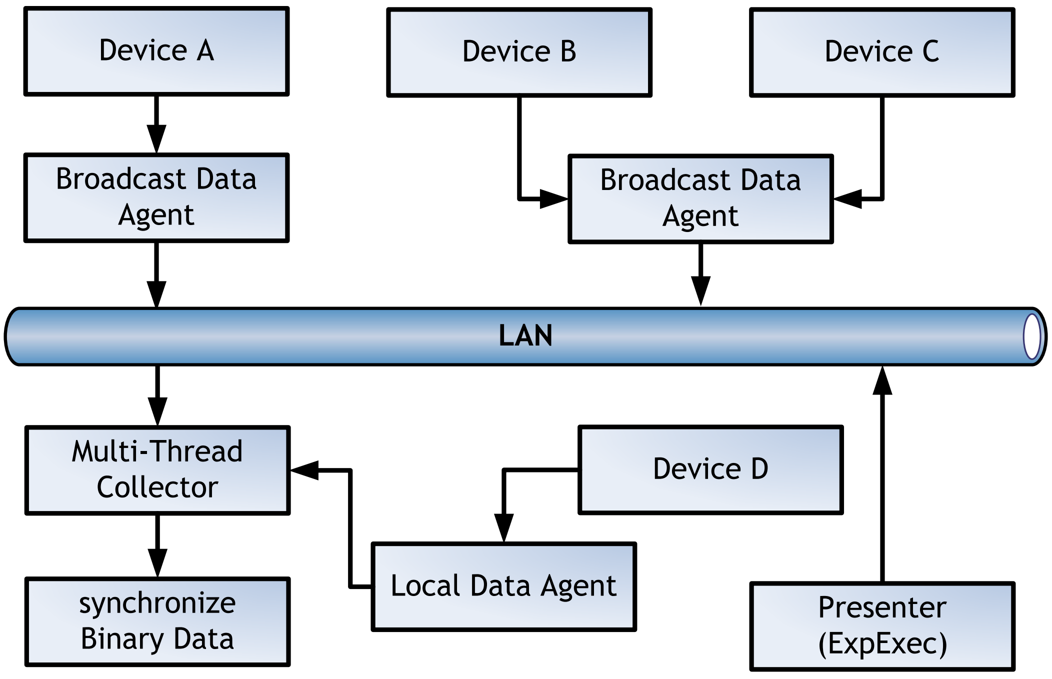


Figure 5-5: Data collection framework

To achieve this goal, the framework is designed as follows: The Framework contains three different layers, the top layer comprises experimental devices with different raw data structures and recording speeds (sampling rate). The middle layer is the Data Agent. The function of this layer is to plug into the device API, get the real-time data and broadcast it to the LAN (Local Area Network). After this layer, we come to our button layer, which receives data packages from LAN and merges them into synchronized data file.

The full scheme of the Data Collection Framework is shown in Figure 5-5. Multiple data stream sources are collected by the Data Agent separately. Each Agent is implemented on the local device programming interface, which corresponds to a particular experimental device. These agents may have different functionality and behavior, but follow the same data transfer protocol. Therefore, these data can be received from any Multi-Thread Collector side on the same LAN, or locally (as shown in Figure 5-5 Device D). Local connecting will shorten the development process and give rise to better performance. However some devices only work under exclusive mode, which means they only accept one connection. Local connecting may bring about the problem that other part of the system (except Multi-Thread Collector) may lose control of data collection procedure, and of course Presenter could not receive any feedback from local connected data sources. To speed up development time, we created Data Agent templates in many different languages, including Pascal, C, C++, and Java. The templates facilitate implementation of new agents with a standardised data acquisition interface and data transfer protocol. Moreover, the source code of new agents can be compiled on different platforms. We can therefore neutralise platform differences for all the experimental devices in this layer. On the other hand, in a real data collection session, in order to make sure the system is load-balanced and reduces the chance of a single point of failure more than one Multi-thread Collector may be employed in the last layer.

Inside the Multi-thread Collector, more than one type of data stream has to be synchronized. We also designed a Multi-source Data Synch Algorithm to reduce the conflict and make the collection proceed smoothly.

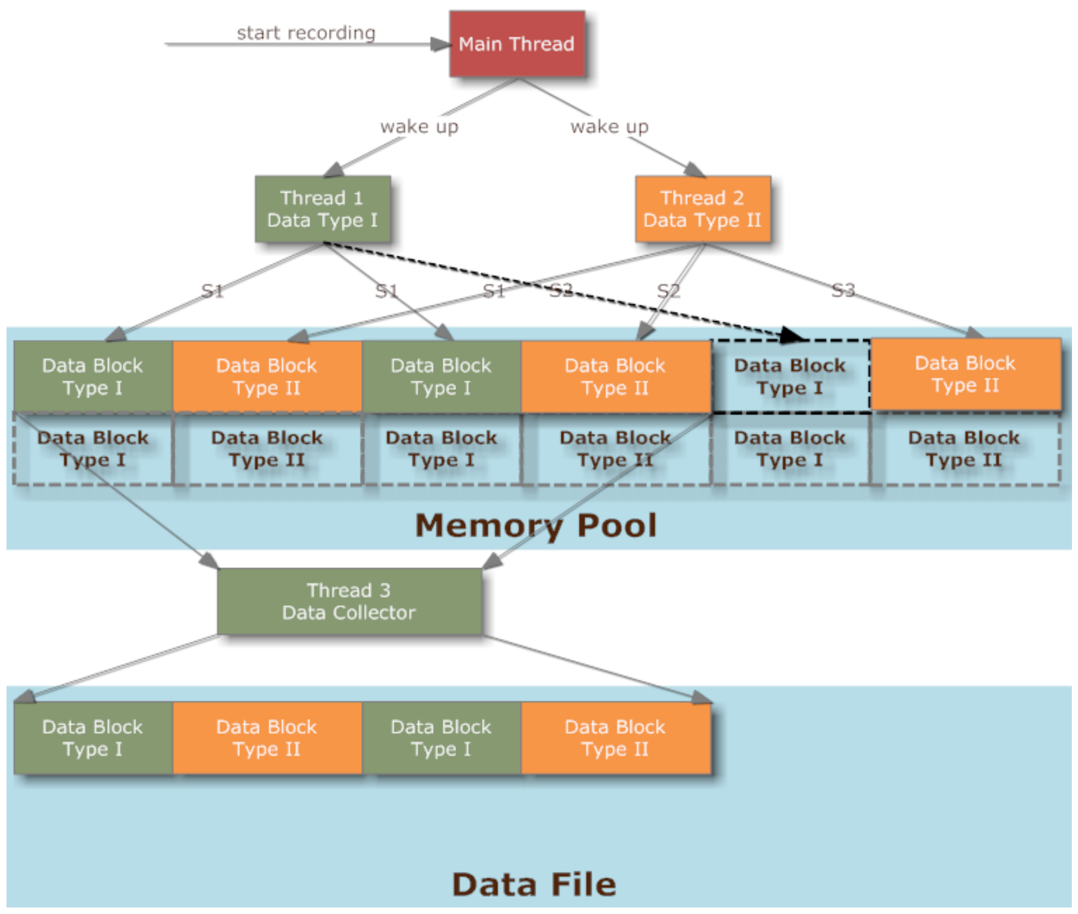


Figure 5-6: Multi-source data synch algorithm

As shown in Figure 5-6, a working memory pool which contains many data slots is created based on a configuration file. Each slot is a minimal unit for data recording and each slot has its own address. When the main thread receives the start recording event, it will wake up the data-collection threads. Each data-collection thread retrieves a real-time data sample package from the LAN and calculates the memory pool address based on a hash of the time stamp of the current sample. These data- collection threads work independently, since they align on the same starting time stamp, each block will be placed in time order in the memory pool. On the other hand, since threads collecting data at different speed, every skipped block will be filled with last valid sample and this re-samples data automatically. The memory pool is designed as a round-robin queue. Collected data will be flushed back to the binary data file once every 10 seconds. To stop recording, a stop recording event will be sent to the main thread. The main thread then reverses a Boolean semaphore, and tells all the data-collection threads to finish recording and stop the corresponding devices.

With the help of the three-layer framework, we created a platform- independent solution for most data synchronization.

5.6. Universal Data Exporter

In data storage, as we mentioned above, an ASCII data file is always bigger than a binary one, especially for numerical data. In order to ensure lower system load and fewer disk write operations on the recorder side, binary format are used during data collection. However, binary data needs further conversion to make data sharing convenient. Therefore, we need a conversion tool for ASCII export in our tool chain.

The biggest problem with binary data is that the data structure varies for different experimental setups. Different eye tracker models (e.g., Tobii and EyeLink have totally different raw data structure) as well as different EEG setups (e.g., same EEG device running on different channels and sampling rate) all affect the final data structure. Thus, to extract the information from a data block, we may need to create different data parsers. To solve this problem we create a universal Data Exporter to convert any block of repeated binary data into ASCII format

(CSV file). In Data Exporter all the conversions are based on a configuration file which has the description of experimental data format.

Table 5-2: An example of Data Exporter configuration file

```
time_sample          unsigned__int32      4
SAMPLE_TYPE          __int16              2
contents_flags       unsigned__int16      2
l_pupil_x            float                4
r_pupil_x            float                4
l_pupil_y            float                4
r_pupil_y            float                4
l_headref_x          float                4
r_headref_x          float                4
l_headref_y          float                4
r_headref_y          float                4
l_pupil_size         float                4
r_pupil_size         float                4
l_gaze_x             float                4
r_gaze_x             float                4
l_gaze_y             float                4
r_gaze_y             float                4
x_pixelsPerDegree    float                4
y_pixelsPerDegree    float                4
status_flags         unsigned__int16      2
input                unsigned__int16      2
buttons              unsigned__int16      2
tracker_data_type    __int16              2
tracker_data         x__int16            16
padding              xxx                172
eeg_channel_1        __int16              2
         …                    …           …
eeg_channel_66       __int16              2
event_channel        __int16              2
```

The above Table 5-2 is a Data Exporter configuration file, which is taken from our real eye tracking and EEG co-registration experiment.

Table 5-3: Item description

Element Title	Data Type	Data Length
time_sample	unsigned__int32	4

In the configuration file, each line has three components. They are: Element Title, Data Type, and Data Length. As shown in Table 5-3, Element Title is the header of data column in exported CSV file. Data Type is the export type of current data element. Data Exporter supports 20 different data types, and they are listed in the following Table 5-4:

Table 5-4: Data types supported by Data Exporter

Category	Data Types	
Byte	char	bool
Signed Integer	short	_int8
	int	_int16
	long	_int32
	long_long	_int64
Unsigned Integer	unsigned_short	unsigned__int8
	unsigned_int	unsigned__int16
	unsigned_long	unsigned__int32
	unsigned_long_long	unsigned__int64
Double	float	double

Only supported data type will be exported. The Data Length indicates the length occupied by the current data element. With the help of the configuration file, users can convert any experimental data into ASCII format and export interested data elements columns only.

5.7. Use Case 1: Shimmer Experiments

As the first use case of the experiment building environment, and part of a programme of research into the co-registration of eye movements and EEG, a series of preliminary experiments were carried out on ExpExec. They involved the reading of texts where the luminance of each word was varied according to a different random value drawn from a Gaussian distribution. The intended effect of the luminance variation (shimmer) was inspired by the VESPA (visual-evoked spread spectrum analysis) technology (Lalor, Kelly, Pearlmutter, Reilly, & Foxe, 2007; Lalor, Pearlmutter, Reilly, McDarby, & Foxe, 2006), where fast and continuous display changes on stimuli were involved. The VESPA technique was used to examine the electrophysiological effects of visual spatial attention to multiple, continuously presented stimuli. Compare to an earlier study by (McMains & Somers, 2005), where split-spotlight attention mechanisms were used, VESPA had the advantage that it allowed researchers to disentangle the distribution of attention on multiple, continuously presented stimuli such as text. Therefore, by shimmering each word in the sentence separately (asynchronously) it may allow us to “tag” the words to permit the disentangling of multiple sources of influence on the resulting EEG signal.

A number of reading experiments were planned to ensure that the flicker paradigm did not interfere with the basic properties of the reading process. A standard and robust effect in natural reading is the already-described preview benefit effect (Rayner, 1998). The Experiment was designed to verify that the standard preview effect was unaffected by the flicker manipulation.

5.7.1. Parafoveal preview benefit

Parafoveal preview benefit in reading is the phenomenon whereby information from a word in the right parafovea enhances the processing of that word when it is subsequently foveated. The amount of benefit is a relative speed-up of approximately 30 milliseconds in viewing time when the reader is given a valid preview. Although small in absolute terms, the phenomenon is robust and well attested in numerous reading studies over the last 30 years (Rayner, 1998).

5.7.2. Shimmer technique

The shimmer technique involves the rapid random modulation of the luminance of the text. Our interest in the effects of luminance flicker stemmed initially from our research into the co-registration of EEG and eye movements in reading, there has been an interest in the effects of flicker on reading over the last few decades. An early impetus for such research was the desire to determine what effect, if any, different CRT refresh rates had on the normal reading process. A series of studies by Kennedy and colleagues (Kennedy, 1996; Kennedy, Brysbaert, & Murray, 1998) systematically explored the effects of various refresh rates on reading parameters such as fixation duration and saccade length and found small but significant effects on saccade length. Lower refresh rates led to shorter than average saccades, which in turn tended to increase the rate of refixation (Radach, Heller, & Hofmeister, 1997)

O'Regan (1990) initiated a debate on the impact of flicker on the interpretability of results from eye-movement contingent display change paradigms. He argued that the response times of computer-based eye tracking equipment at the time were such that subjects could be aware on a significant number of trials of the unmasking of text in, say, a boundary paradigm (Rayner, 1978). This would then lead to disturbances in fixation durations and saccade targeting; phenomena that until then had been attributed to the curtailment of preview information to the reader. O'Regan argued that boundary effects, on the contrary, were an artifact of the flicker produced by the display-change paradigm. However, in a response to this position, Inhoff *et al.* (1998) in a comprehensive study of the impact of different refresh rates on preview found no discernible effects of refresh rate on preview benefit (Inhoff, et al., 1998).

In summary, the literature on the impact of flicker on reading suggests that it may impact on saccade length and that varying sensitivity to flicker may play a role in reading impairment.

5.7.3. Participants

Twenty one subjects from the Florida State University community participated in this experiment. All had either normal or corrected vision and were native speakers of English.

5.7.4. Materials & Design

The sentences used in the experiment were English translations of the materials used in (Radach, Huestegge, & Reilly, 2008) and comprised 192 sentences. They contained a target word of 7 or 8 letters in length that either remained the same during the identical preview condition or was changed from an non-word mask (matched on length and the first letter) when the reader's eye crossed an invisible boundary to the left of the target (Rayner, 1975). Fifty percent of the trials involved identical preview of the target and 50% involved an unrelated non-word mask preview.

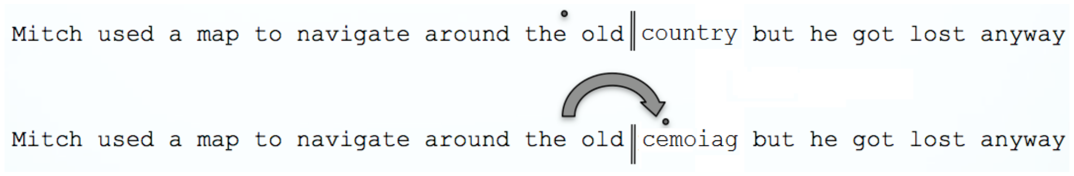

Mitch used a map to navigate around the old country but he got lost anyway
Mitch used a map to navigate around the old cemoiag but he got lost anyway

Figure 5-7: An example of the unrelated preview conditions.

As shown in Figure 5-7, the • indicates the subject’s fixation location, and the vertical bar represents the invisible boundary, which when crossed causes the target word just after the bar to change to an unrelated non-word mask. In the identical preview condition, no word change occurred. Note that each word was shimmered according to a shimmer pattern. There are two patterns of shimmer. The first pattern is which we called “asynchronous” shimmer. In “asynchronous” pattern each word in the sentence gets a new and different luminance value every time before the monitor refreshes itself, which results that each word shimmered separately. While in another shimmer pattern, which we called “synchronous” shimmer, each word in the sentence gets a new but same luminance value before every refresh frame, in which the sentence as a whole was shimmered synchronously. For both patterns the refresh rate of the monitor is 200 Hz. In addition, for 50% of the sentences, each word was shimmered asynchronously to the rest of the sentence. Thus, the experiment was a 2x2 factorial design, where the two factors were pattern of shimmer (synchronous vs. asynchronous) and preview type (identical vs. unrelated).

Text was displayed as white on a black background. In order to manipulate luminance values we choose to represent colour values using the HSL system (Joblove & Greenberg, 1978), an additive colour system based on the attributes of colour (hue), percentage of white (saturation), and brightness (lightness). The HSL system describes brightness relationships more accurately than the more common RGB system, while remaining computationally simple to use. The luminance values of each word were changed to random values drawn from a Gaussian distribution (A random value is first generated by using a standard Gaussian distribution with zero mean and unit standard deviation, then we set a threshold between –255*4 and +255*4, which means if the value is bigger than 1020, it will stick to 1020, and if the value is smaller than - 1020, it will turn itself to - 1020. Finally, the new value is rescaled by dividing 8, and then shifted by adding 128. Therefore, the final value is between 0 and 255). Gaussian distribution makes the varying luminance patterns less annoying to the reader by avoiding abrupt changes. The overall effect was to give the text a shimmering rather than flickering appearance, with the luminance of each word being altered by a different sequence of random variation (see http://www.eyemap.tk/shimmer /sample1.avi for a sample video of the “asynchronous” shimmer stimuli).

5.7.5. Apparatus

Experimental stimuli were presented on a 200Hz monochrome green M21LMAX CRT monitor from Image Systems Corporation as our stimuli presenter. The spatial resolution of the monitor was 1024 x 768 pixels and the screen subtended 27.8° (40.6 cm) horizontally and 21.5° (30.5 cm) vertically. This resulted in 36.4 pixels per deg at a viewing distance of 82 cm.

Eye movements were recorded using an EyeLink 1000 (SR-Research Ltd., 2011) eye tracker, which has a maximum sampling rate of 1000 Hz. It was calibrated for each subject by instructing them to fixate a series of there dots (0.55° diameter) that were laid out as a line on the screen. Subjects were seated with their heads stabilised by a chin-rest. They viewed the display binocularly, though only their right eye was calibrated. Stimuli were presented and controlled by ExpExec 2.0 on a display PC and eye movement data were collected using a separate recorder PC.

5.7.6. Procedure

Prior to the start of the experiment, subjects read a short set of instructions in which they were told to read some English sentences and that the text would appear to shimmer. After some sentences, they would be presented with a pair of words and their task was to indicate by pressing a button on a game pad which of the pair occurred in the preceding sentence. They were asked to read at their normal rate.

Each trial started with a fixation dot (0.83° diameter) that appeared on the left of the screen, next to where the first character of the stimulus sentence would be presented. Subjects initiated each trial by pressing an assigned button on the game pad. The EyeLink 1000 system then performed a drift correction to correct for possible shifting of the head-mounted tracking system. When the drift correction was made, the fixation dot disappeared and the stimulus sentence would appear on the screen. The luminance of the words changed at the rate of 200Hz. The change was synchronised with the screen refresh rate so that no frames were lost during the trial. All text was displayed in a 17pt Consolas mono-spaced font, with each character 13 pixels in width.

Subjects completed six practice trials before proceeding to the main experiment. In the main experiment, 192 sentences were shown in random order.

For 50% of the experimental sentences, the target-word location was occupied by a length- and frequency-matched word that was inappropriate for the context. For the remainder of the sentences, the target-word location was filled by a context-appropriate word. On reading the experimental sentences when the subject's eye crossed an invisible boundary at the end of the pre-target word, one of two events occurred: (1) in cases where the target location was occupied by an inappropriate word, this word was replaced by the correct word; (2) where the location already contained the correct word, no change occurred.

5.7.7. Results

The main focus of this experiment was on the viewing times for the target word. Fixation durations for all subjects were filtered, with fixations less than 80 ms or greater than 800ms excluded from analysis. A mixed-effects model (D. Bates & Sarkar, 2007) was used to analyse the results of the experiment. In this case, the fixed effects were preview type and shimmer synchrony, while the random effects were subject and sentence.

Table 5-5 enumerated the fixed effects on the first fixation duration (FFD) and gaze duration (GD) of target words. A preview effect was evident for both FFDs and GDs ($|t| > 2$ in both cases). The effect of shimmer pattern on both measurements was also significant ($|t| > 2$). First fixation durations and gaze durations were longer in asynchronous compared to synchronous mode by about 7 ms. However, most crucially, there was no significant interaction between preview type and shimmer pattern for both measures ($|t| < 2$).

Table 5-5: Analysis of target words

Model	Fixed effects contrasts	β	SE	t
First fixation duration (FFD)	Preview (unrelated - identical)	7.72	3.43	**2.25**
	Shimmer (synch - async)	-7.08	3.44	**-2.06**
	Preview x Shimmer	2.55	6.85	0.37
Gaze duration (GD)	Preview (unrelated - identical)	8.30	3.43	**2.42**
	Shimmer (synch - async)	-7.05	3.43	**-2.05**
	Preview x Shimmer	2.56	6.85	0.37

5.7.8. Discussion

The results of the shimmer experiment showed that there was a small advantage in reading under synchronous shimmer mode compared to asynchronous mode. The results not only indicated that there was no interaction between the shimmer patterns and preview, but also that there were small but significant preview effects for both FFD and GD.

5.8. Use Case 2: Co-Registration of Eye Movements and EEG

As mentioned in the last section, the major motivation for the shimmer experiments was to help develop a technique for the co-registration of eye movements and EEG. A series of preliminary experiments were carried out involving recording eye movement data and EEG signal synchronously. These preliminary experiments were deployed under different environments with different eye trackers including SR EyeLink series (SR-Research Ltd., 2011) and Tobii 1750 (Tobii Technology, 2011); as well as several EEG devices including BrainVision (Brain Vision LLC., 2011), MindSet 24R (Biofeedback Instrument Corporation, 2011), NeuroScan (Compumedics Neuroscan, 2011), and ActiveTwo (BioSemi Instrumentation, 2011).

5.8.1. Eye Tracker Data Agent

As described before, the eye tracker data agent was usually implemented as a broadcast data agent (see Section 5.5 for more details about the data agent). This is because eye trackers are usually implemented as client–server models, since high speed data recording consumes too much computing resource and can potentially slow down stimulus presentation. This creates a convenient and fast path to create the data collector using a broadcast technique. Two separate broadcast examples were implemented in C++ for the SR EyeLink series and the Tobii series of platforms. The source code is available in our supporting materials (see attached DVD folder Chapter 5\Co-Registration Data Recorder) and the code is a partially modified version of the sample code supplied in the EyeLink and Tobii SDKs. The broadcast agent connects to the tracker and broadcasts its raw data as well as control events such as start or stop recording event to the multi-thread data collector. It works independently from the experimental stimulus presenter (such as Experiment Builder or our ExpExec), therefore no matter what kind of presenter and experiment procedure is being used, the eye tracker data agent can always work without the need to change a line of code.

5.8.2. EEG Data Agent

We implemented at least four data agents for four different EEG devices. Half of them were local data agents (see Section 5.5 for more details about the data agent). This is because there is no interaction between EEG data and presenters.

A local data agent is a better choice for quick development and low latency data transfer. However, there is a danger that a local data agent may consume to many resources, which may slow down the multi-thread data collector.

The source code for the EEG Data Agents was written in C and C++. They are also available in our supported materials (see attached DVD folder Chapter 5\Co-Registration Data Recorder). Some code is partially modified from the sample code of the EEG SDKs or modified from BCI2000 (Schalk, McFarland, Hinterberger, Birbaumer, & Wolpaw, 2004). BCI2000 is a general-purpose system for brain-computer interface (BCI) research. It is an open source project, which contains many EEG data acquisition modules.

5.8.3. First Prototype

The first data collection was performed by a prototype system which was designed to combine data sources from a SR EyeLink II and a BioSemi ActiveTwo. BioSemi ActiveTwo is 264 channels low noise DC EEG Amplifier (256 EEG channels plus 8 touchproofs for EOG, EMG, and ECG). It is fast (16 kHz sampling rate per channel on recording of 32 EEG channels), and easy to operate. It provides reliable measurements without the need for skin preparation. The electrodes are shielded and easy to clean. However, on the other hand, the SR EyeLink II system is not an optimal choice of tracker since it is head mounted. This combination requires participant to put both EEG cap and tracker on their head. Therefore for this preliminary test, we only collected 4 EOG channel from participant. As shown in Figure 5-8, the participant wore the EEG cap underneath the SR EyeLink II tracker. Four EOG and two ground electrodes were connect to the BioSemi ActiveTwo AD box. While the AD box is connect to the local recording machine (the laptop) though the USB2 receiver.

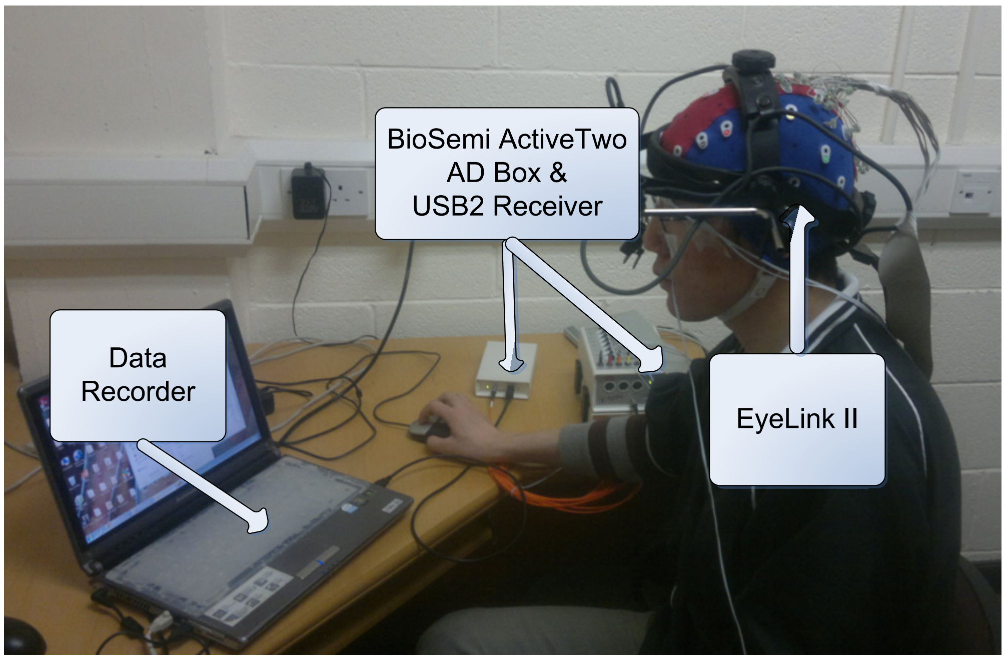


Figure 5-8: Experimental setup

The architecture of prototype system was shown in

Figure 5-9. Three computers (Display machine, Data Collector, and EyeLink Recorder) were connected by a private local area network. The BioSemi ActiveTwo amplify was actually working at 2048 Hz, and the EOG signals were resampled at 512 Hz (we used the mean value of every 4 samples from the same channel as the new sample value). The rest of the setup was the same as we described in previous shimmer experiment. The participant was seated in front of a 21-inch Samsung SyncMaster 1100MB CRT monitor with their head stabilised by a chin-rest. The monitor operated at 100Hz and the eye tracker (SR EyeLink II) at 500Hz but was resampled to 512Hz (the resample process was based on the time stamp of the collected samples, one sample is duplicated in approximately

every 42 samples) with only right eye recorded.

Broadcasted eye movement events and local collected EOG data were received by the Multi-thread Collector (Running at Data Collector in

Figure 5-9), resample to 512 Hz, and merged into binary raw data packages (see Figure 5-10) using our multi-source data synch algorithm (Figure 5-6) . The control messages such as start or stop recording were sent by Presenter (e.g., ExpExec) to EyeLink Recorder, and finally delivered to Data Collector. Before each trial started, both EEG and eye tracker were reset, in order to clean the previous collected data before start moment.

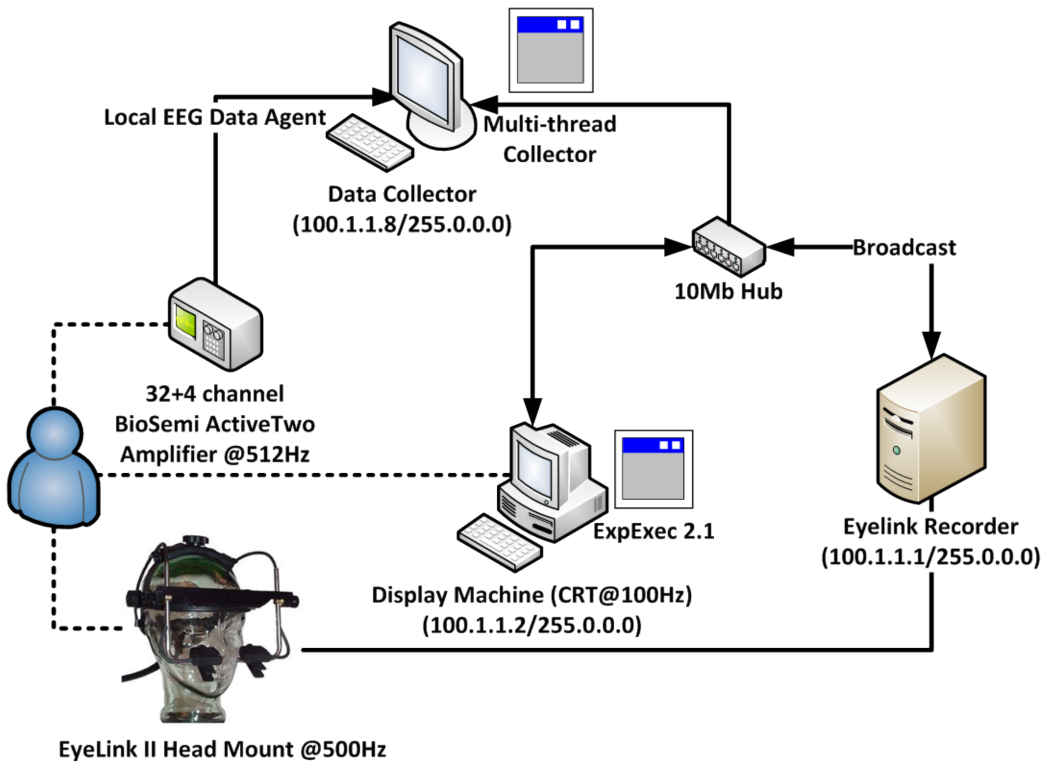

Local EEG Data Agent
Multi-thread Collector
Data Collector (100.1.1.8/255.0.0.0)
10Mb Hub
Broadcast
32+4 channel BioSemi ActiveTwo Amplifier @512Hz
ExpExec 2.1
Eyelink Recorder (100.1.1.1/255.0.0.0)
Display Machine (CRT@100Hz) (100.1.1.2/255.0.0.0)
EyeLink II Head Mount @500Hz

Figure 5-9: Setup of the prototype system which combines the EyeLink II with BioSemi ActiveTwo amplifier

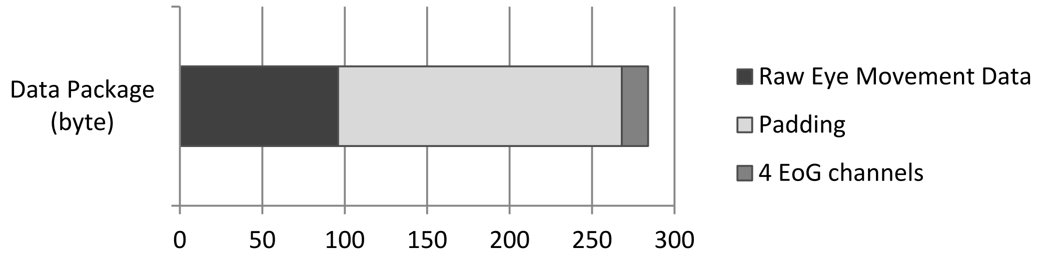


Figure 5-10: Collected raw data package

5.8.4. Discussion and Future work

The prototype system demonstrated the feasibility of simultaneously collecting multiple data sources using our system. However, it also exposed some issues, which should be considered or solved in future work. First, due to the head mounted tracker we used, no real EEG data but only EOG data were collected in this preliminary experiment. Besides, the infrared light sources on the tracker seem to generate electronic interference with the electrodes (we observed a strong 15Hz interferences from environment, which were possibly a synchronisation signals to the LED light sources). To fulfill the goal of co-registration of eye movements and EEG, the real data collection should be performed on remote trackers such as EyeLink 1000, or Tobii TX300. In addition, local data agents may cause I/O pressure on Data Collector when we recording at a high speed (for example over 1000Hz).

Therefore, a further experiment will be considered and prepared to run on a remote EyeLink 1000 with NeuroScan Amplifier under the help of MARCS Auditory Labs at the University of Western Sydney. The system setup is shown in

Figure 5-11. The new setup will solve the above mentioned problems. As shown in the Figure, the original EEG and eye movement data will be recorded

separately on a NeuroScan Recorder and an EyeLink Recorder, and both recorders will broadcast their events to Data Collector. The source code of new Multi-thread Collector in this setup and some preparation documents can be found in our support materials (see attached DVD, folder Chapter 5\ Co-Registration Data Recorder).

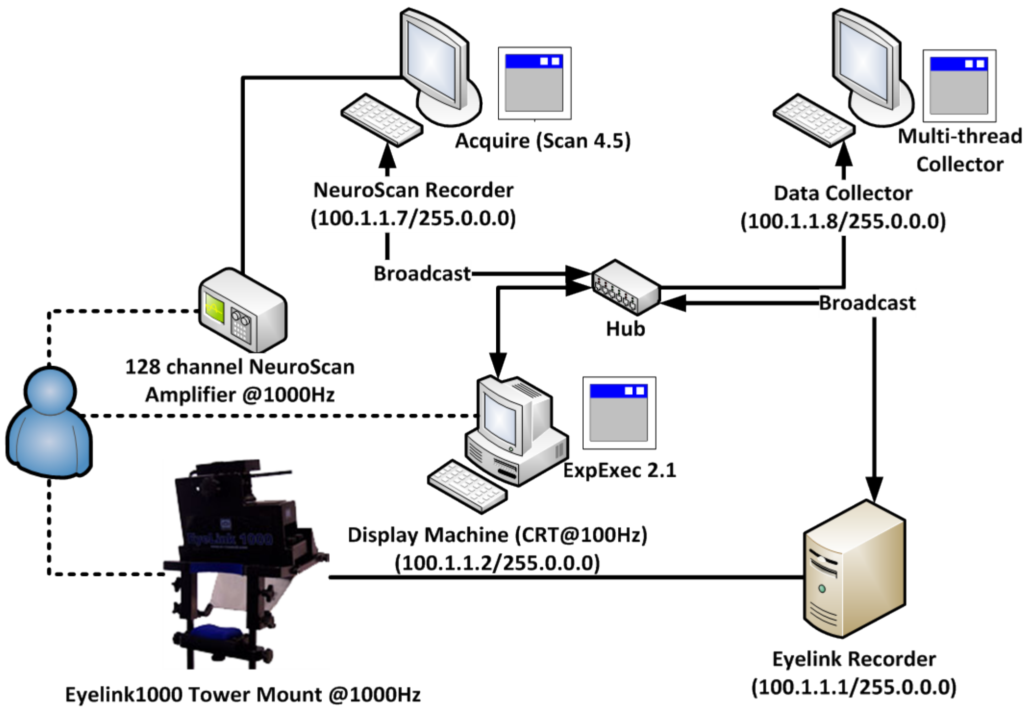


Figure 5-11: System setup which combines the EyeLink 1000 with NeuroScan amplifier

Chapter 6

EyeMap: An Analysis and Visualization Tool for Eye Movement Data in Reading

The following sections provide a brief overview of the EyeMap eye movement data analysis and visualization tool. EyeMap is a freely available software system for analyzing and visualizing eye movement data specifically in the area of reading research. Compared to similar systems, including commercial ones, EyeMap has more advanced features for text stimulus presentation, interest area extraction, eye movement data visualization, and experimental variable calculation. It supports binocular data analysis for Unicode, proportional and non-proportional fonts, spaced and unspaced scripts. Consequently it is well suited for research on a wide range of writing systems. To date it has been used with English, German, Thai, Korean, and Chinese. EyeMap is platform independent and can also work on mobile devices. An important contribution of the EyeMap project is a device-independent XML data format to describe data from a wide range of reading experiments. An online version of EyeMap allows researchers to analyse and visualize reading data through a standard web browser. This facility could, for example, serve as a font-end for online eye movement data corpora.

6.1. General Issues in Reading Data Analysis and Visualization

As mentioned in Chapter 5, most of the data analysis software supplied with trackers is proprietary. Their major disadvantages for reading research applications are:

They tend to have few functions for processing reading data and generating useful variables (measurements on eye movement events in specific areas of interest such as letters and words);

Most have limited options for text presentation and analysis. Some cannot deal directly with text materials (users have first to convert the text to an image). Some do not support Unicode and therefore have problems dealing with writing systems other than Roman-based ones. Some do not readily support the editing of eye movement data. For example SR DataViewer, supports only manual drift correction for each text line and sometimes has problems identifying word boundaries;

The software is hardware specific and the data are stored in a proprietary format. Most cannot work across platforms, which causes problems with data sharing and result comparison;

They are not freely available, are difficult to extend, have limited visualization capabilities, and can handle only a limited range of experimental paradigms. As a result, researchers often have to develop their own analysis software for specific experiments;

Most solutions are commercial software packages and purchase and support can be quite expensive for academic researchers.

6.2. Design Principles of EyeMap

In order to resolve the above-mentioned problems and produce a more effective tool that allows psychologists to gain deeper insight into their eye movement reading data, we developed EyeMap, a freely available software system for visualizing and analyzing eye movement data specifically for reading research.

EyeMap supports dynamical calculation and exporting of more than 150 different types of reading related variables (see Appendix A for the current list of variables). It allows researchers to load reading materials from a HTML file, which is convenient for creating, presenting, modifying, and organizing different experimental text materials. Furthermore, Unicode text can also be handled, which means all writing systems, even un-spaced scripts such as Thai and Chinese, can be presented and segmented properly. Moreover, as far as we are aware, it is the first data analysis software that can automatically and reliably find word boundary for proportional and non-proportional fonts. During the analysis, these boundaries are marked as areas of interest (AoI) based on font metrics. Another advantage of EyeMap is that it can work on any platform, even mobile devices. It also contributes an XML (eXtensible Markup Language) based device-independent data format for structuring eye movement reading data. Another important contribution is that EyeMap provides a powerful user interface for visualizing binocular eye movement data. Direct data manipulation is also supported in the visualization mode. Furthermore, an EyeMap online version allows users to analyze their eye movement data through a standard web browser, which is in keeping with the trend towards the use of cloud-based computing environments.

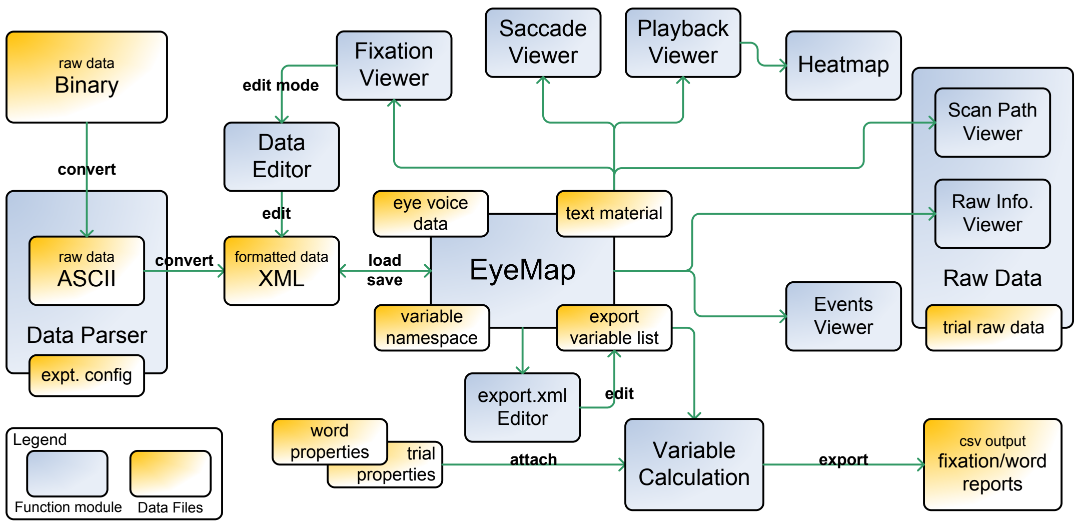


Figure 6-1: System overview

Figure 6-1 presents all of the functional modules and associated data files involved in the EyeMap environment. In Figure 6-2, below, we present an informal use-case analysis of the process of going from the binary formatted eye tracker data (top of figure) through to an exported CSV file (bottom of figure).

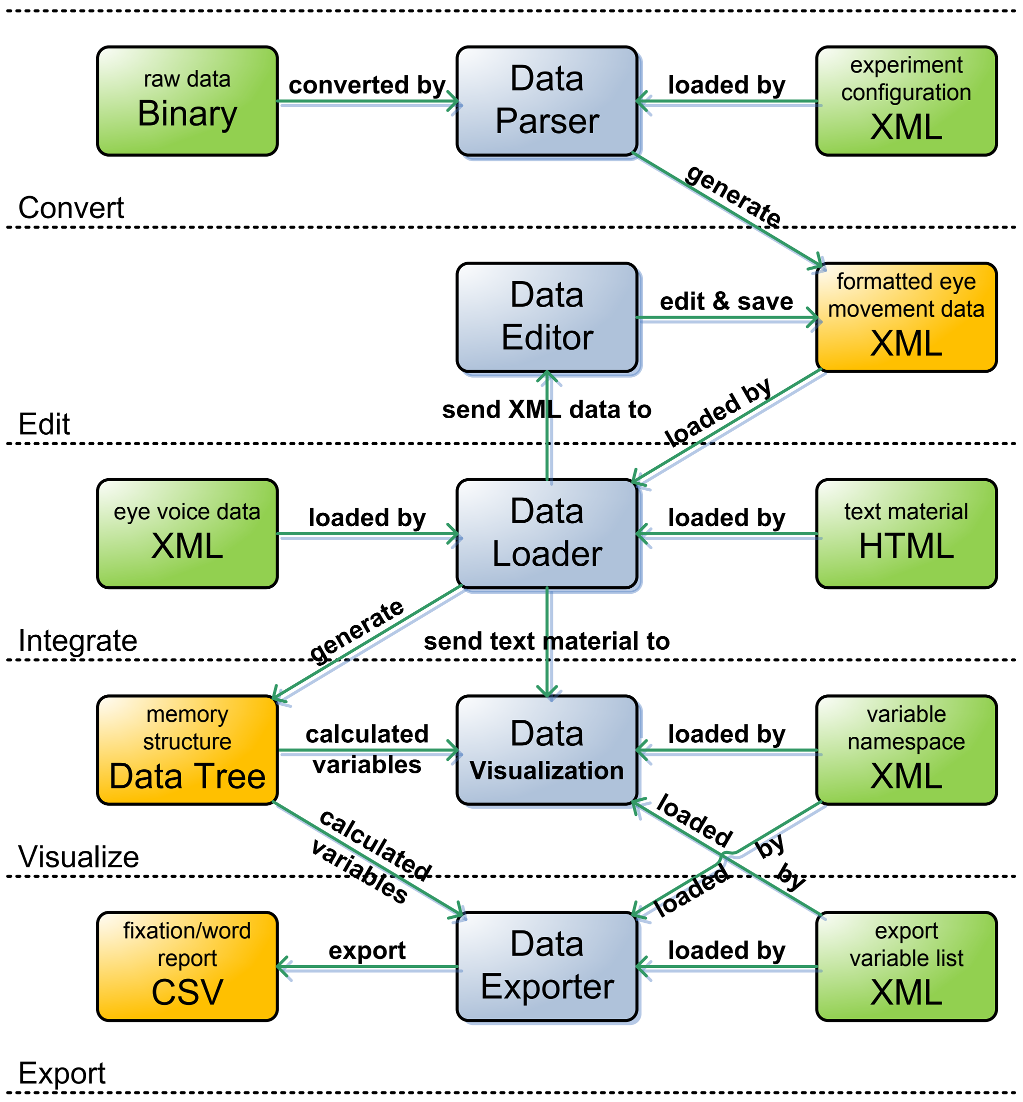


Figure 6-2: Use-case analysis of EyeMap

The blue rectangles in Figure 6-2 represent processes carried out by EyeMap, the yellow ones represent EyeMap-generated data, and the green ones represent user-supplied data sources. Note that in the current version of the system, the Data Parser is a standalone module, separate from the main EyeMap software.

This permits its replacement when parsing data files from different manufacturers.

It should also be noted that currently EyeMap uses the results from the online saccade/fixation detection mechanism used by the specific eye tracker manufacturer. However, in principle, it should be possible to use a potentially more accurate post-processing algorithm on the raw eye position samples extracted from offline eye movement data source, such as the EyeLink EDF file and Tobii TSV exports.

Isomorphic with the sequence of actions illustrated in Figure 6-2, the workflow in EyeMap follows roughly four phases: data conversion, data integration, data visualization and editing, and data export and sharing. In the following sections we will provide some more detail of these different phases.

6.3. Eye Movement Data Conversion in EyeMap

Data organization of the tracking device is always the first issue for eye movement data analysis software developers. Eye movement data are organized in many different ways and each tracker manufacturer has their own data descriptions. In high-speed online trackers, which are deployed in most reading experiments, the raw data are stored in binary format, due to the requirement for fast data retrieval and storage. The lack of standardization of data format, however, can be an impediment to data exchange between different research groups. Some tracker manufacturers provide APIs or tools to export the binary data into ASCII, which is a convenient general format for data sharing, but not sufficient enough for data processing, since the data in this format lack any structural or relational information. Some researchers have tried to develop device-independent data formats for a range of trackers. The first public device-independent data format was developed by Halverson and colleagues (Halverson & Hornof, 2002). It has an XML-like structure in which information such as experiment description, area-of-interest information, and internal/external events are stored together. XML enforces a more formal and strict data format that facilitates further data processing. Another XML-based generic data storage format was published as a COGAIN deliverable (R. Bates, Istance, & Spakov, 2005) and has become the *de-facto* standard for gaze-based communication systems. However, since it is designed for gaze-based communication it focuses mainly on hardware aspects so as to be compliant with as many trackers as possible, which is not necessarily a helpful option for reading research. This is because only a limited number of types of tracker can be used in modern reading research (see the discussion in section 2.3). On the other hand, reading psychologists are more interested in the cognitive processing occurring during the fixation/saccade/blink/pupil events, which is at a finer level of detail than simple gaze-level analyses afforded by low-temporal resolution trackers. Therefore, COGAIN carries with it a considerable overhead, while still not being good enough for reading research. For these reasons, we developed another XML-based device-independent data format designed specifically for reading experiments to facilitate the calculation of eye movement measures that are specific to reading.

A reading experiment is always trial based, and in each trial, subjects are required to read some pre-determined text. To describe a reading experiment with all the related events that can occur during the experiment, we need at least the following elements in our XML description:

1. <root></root>: All the information of the experiment is inside the root pair. It contains <trial> elements. And each data file only has one <root> block;
2. <trial></trial>: Trial block contains all the information for a single experiment trial. Each block has <fix> and <sacc> elements. Each trial block has a unique number called “id”, which starts from 0, indicating the number of the trial in the current experiment. Each trial also has an <invalid> tag, which represents whether the current trial is valid or not;
3. <fix></fix>: Fix block contains all the information for a fixation, each fixation block may have the following tags:
   a) <eye>: the eye with which the current fixation event is associated;
   b) <st>: start time-stamp in milliseconds;
   c) <et>: end time-stamp in milliseconds;
   d) <dur>: duration (in milliseconds) of the current fixation;
   e) <x>/<y>: average X, Y coordinates of the current fixation;
   f) <pupil>: average pupil size (diameter) in arbitrary units for the current fixation;
   g) <blink>*: duration (in milliseconds) of a blink if it occurred prior to the current fixation;
   h) <raw>*: the distance in pixels of the start (sx; sy) and end (ex; ey) X, Y coordinates from the average X, Y coordinates of the current fixation;
   i) <id>: a unique number starting from 0 (based on temporal order, where even numbers are used for the right eye and odd for the left

eye);

4. <sacc> block contains all the information for a saccade, each saccade block will have the following tags:
    a) <eye>: the eye with which the current fixation event is associated;
    b) <st>: start time stamp in milliseconds;
    c) <et>: end time stamp in milliseconds;
    d) <dur>: duration (in milliseconds) of the fixation;
    e) <x>/<y>: X, Y coordinates of the start position;
    f) <tx>/<ty>: X, Y coordinates of the end position;
    g) <ampl>: the total visual angle traversed by the saccade is reported by <ampl>, which can be divided by (<dur>/1000) to obtain the average velocity;
    h) <pv>: the peak values of gaze velocity in degrees per second;
    i) <id>: a unique number starting from 0 (based on temporal order, even numbers for right eye, odd for left eye);

With respect to the odd/even ID coding scheme, this has proved especially useful for helping to synchronise data from the right and left eye for studies of binocularity effects. The SR DataViewer, for example, provides separate data exports for right and left eye, but the task of synchronizing these data streams is non-trivial. The use of odd/even ID numbers makes the synchronization process a lot more manageable.

An example of XML data file is show in Figure 6-3.

```
<root>
    <trial id="0">
        <fix>
            <eye>R</eye>
            <st>927307</st>
            <et>927470</et>
            <dur>164</dur>
            <x>100.1</x>
            <y>104.7</y>
            <pupil>1501</pupil>
            <raw sx="-2.7" sy="0.9" ex="2.4" ey="-0.4"/>
            <id>0</id>
        </fix>
        <sacc>
            <eye>R</eye>
            <st>927471</st>
            <et>927515</et>
            <dur>45</dur>
            <x>102.6</x>
            <y>104.1</y>
            <tx>508.6</tx>
            <ty>133.3</ty>
            <ampl>11.66</ampl>
            <pv>506</pv>
             <id>0</id>
         </sacc>
    </trail>
</root>
```

Figure 6-3: An example of eye movement data XML

Data parsers were developed to convert different data sources into the aforementioned XML format. The data parser is actually a small lexical analyzer generated by Lex and Yacc (Levine, Mason, & Brown, 1992). The parser works on lists of regular expressions that can identify useful components from the ASCII data file, and recognize them as tokens, then recombine the tokens into XML elements. Currently the parser works on EyeLink (SR-Research Ltd., 2011) and Tobii (Tobii Technology, 2008) exported ASCII data formats. However, it would be quite straightforward to create a new parser for other eye tracking devices, since users only need to create a new list of regular expressions based on the output specification of the new device.

6.4. A Data Structure for Reading

In a typical reading experiment, parameters of interest are either properties of the stimulus materials (words, characters, and their linguistic features) or measures of the ongoing oculomotor behavior during the dynamic process of acquiring this linguistic information. This produces two different perspectives on the experimental data, the text perspective and the event perspective. Moreover, reading researchers are usually only interested in some specific variables (or attributes) in any given experiment. Take some common variables in a text reading experiment as an example: the variable *launch distance* is an event-based variable, which describes the distance in character spaces from the launch site of an incoming saccade to the beginning of the current word. In contrast, the variable *gaze duration*, defined as the sum of all fixations on the target prior to any fixation on another word, is considered a text-based variable in this classification. In order to fulfill the data modification and variable calculation requirements of these two perspectives, the experimental data are organized as a tree structure containing six different levels. From the root to the leaves: experimental level, trial level, word level, gaze level, fixation level and saccade level. The basic component of the experimental data tree is an abstract node class. The node class has the basic functions of constructing and maintaining the relationships among the elements of the data tree, which is the key to all the variable calculations. For a node $N_{p'}$, if its i^{th} child node is N_i, then the node N_i can be formalized as the following set:

$$N_i = \left\{ i, i', N_p, N_{i-1}, N_{i+1}, A_{i_j} = \left\{ A_{i_1}, A_{i_2}, \cdots, A_{i_j} \right\}, T_{i_n} = \left(N_{i_1}, N_{i_2} \cdots, N_{i_n} \right) \right\}$$

($N_i \in T_{p_n}$ and $N_i = N_{p_i}$)

As shown in the formula above, the node N_i satisfies the following properties:

1. N_i has an ID *i* which indicates that it is the i^{th} child of its current parent.
2. N_i has another ID i' which indicates that it is the i'^{th} child of its old parent. If $i \neq i'$, this means current parent node N_p is not the original parent of N_i. In other words, Node N_i is created by another node, but adopted by N_p. Since the node class allows multiple parents to connect to one node, the ID i' can help a parent node to identify itself.
3. N_i has a reference to its current parent node N_p.
4. N_i has a reference to its predecessor (N_{i-1}) and successor (N_{i+1}).
5. N_i has a set of attributes (P_{i_j}) , which contains the specific information of node N_i.
6. N_i contains a *n*-tuple T_{i_n} which records an ordered list of child nodes.

From the vertical perspective, the node class builds a bi-directional link between a parent node and its children. While from the horizontal perspective, the node class retains bi-directional references to children at the same level. The vertical perspective can simplify variable calculations. For example, once you reach the gaze node G_i, the gaze duration on G_i can simply be calculated by following formula: $G_i \times A_{i_{gazeDur}} = \sum_{j=1}^{n} G_{i_j} \times A_{i_{j\,fixationDur}}$. This effectively involves the traversal of the gaze subtree and the accumulation of fixation durations in the process. A horizontal perspective, on the other hand, is extremely useful in generating word-level data matrices for output and analysis. Typically in a reading experiment the analysis focuses in turn on the fixation event level and on the objects of experimental interest, usually the words and sentences of the text. EyeMap allows the export of separate event level and word level data matrices.

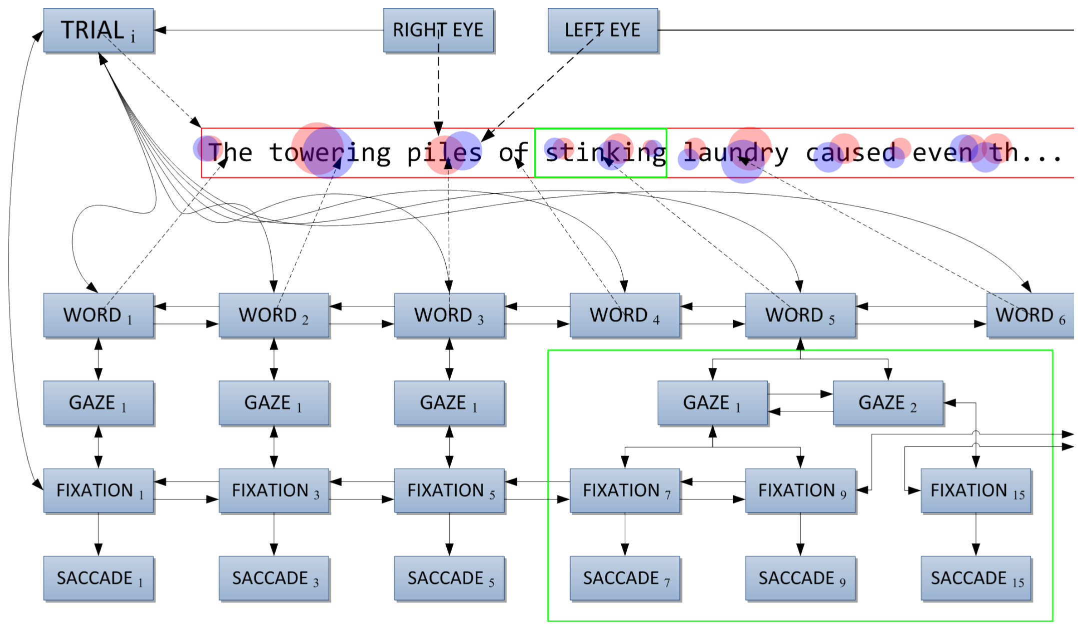


Figure 6-4: Nodes in memory

Figure 6-4 is a fragment of a dynamic memory data structure created for processing a trial of a single line of text reading data. Inside the green box, the word *stinking* has three right eye fixations and these fixations belong to two different gazes. Thus, as depicted in the Figure 6-4, the WORD5 node contains all the information (word boundary, word center, word frequency etc.) for the word *stinking*. It has two child GAZE nodes. Three FIXATION nodes are allocated to two GAZE nodes based on the gaze definition.

To optimize the loading speed, the data tree is not created at one time. Each time when a trial is presented/modified the corresponding trial branch is created and/or updated, thus the information in the visualization is always up to date.

6.5. Data Integration

In the current implementation of EyeMap there are two sources of data that can be used to augment the eye movement data. A mandatory data source is the file detailing the text being read (i.e., text.html). Clearly, any reading eye movement data is uninterpretable without information about the words being viewed. The text information is provided in a HTML formatted file which also provides font type and size information. The latter plays an important role in word and letter segmentation.

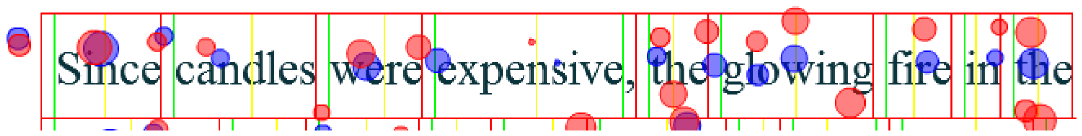
An additional, optional, facility is to synchronize speech data with eye movement data. Voice data is incorporated in two ways: as part of the visualization system and as part of the data export facility.

For visualization, the articulation start time and duration are extracted from a user-supplied *voice.xml* file for each word. When the user examines a word in the current trial, EyeMap will play the associated recording of that word.

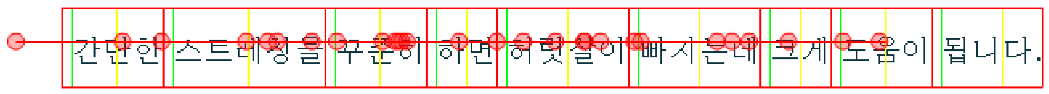
For data analysis, the articulation start time is synchronized with the experimental timeline. It is therefore possible to associate when word articulation starts or ends with which fixation was being made at that time and what word or letter was being fixated. In this way, EyeMap can readily provide information about the eye voice span.

6.6. Data Visualization and Editing

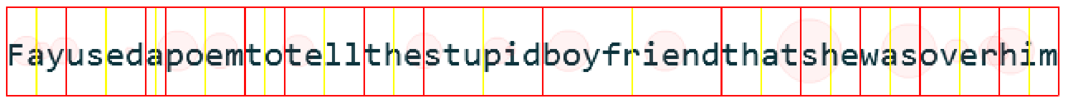
The main job of this phase is to represent the experimental stimulus and visualize the eye movement data in different ways, which facilitates the user in understanding and exploring the data from different perspectives. In a reading experiment, the primary stimulus is text. Figure 6-5 shows some sample experimental data using EyeMap with different combinations of Unicode, proportional and non-proportional fonts, and spaced and unspaced scripts. So far, EyeMap has been used for research on a wide range of writing systems, such as English, German, Thai, Korean, and Chinese.



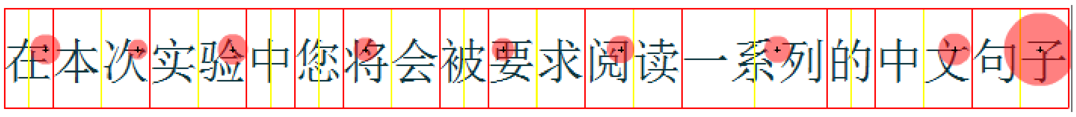
a) multiline spaced English text with proportional font and punctuation


간단한 스트레칭을 꾸준히 하면 허릿살이 빠지는데 크게 도움이 됩니다.

b) single-line spaced Korean text with non-proportional font



c) single-line unspaced English text with non-proportional font



d) single-line spaced Korean text with non-proportional font

Figure 6-5: Representing various texts with recorded eye movement data and word boundaries

As shown in Figure 6-5, EyeMap not only re-creates the experimental stimulus, but also uses font metrics to find the word center (yellow line) and word boundary automatically, with and without punctuation (red versus green boundary), and marks them as areas of interest. As far as we are aware, EyeMap is the first software capable of reliably segmenting proportional fonts in this way. For example, the SR Data Viewer has problems handling proportional fonts. Figure 6-6, below, gives a sample comparison with EyeMap.

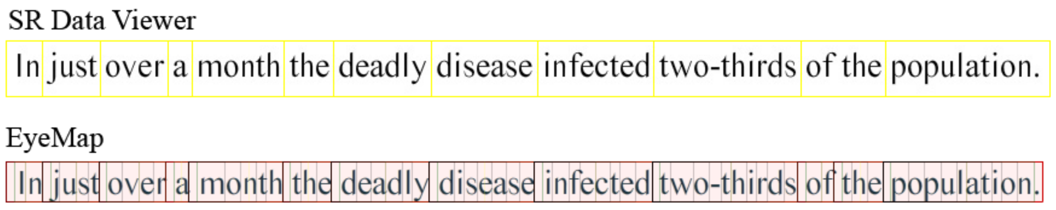
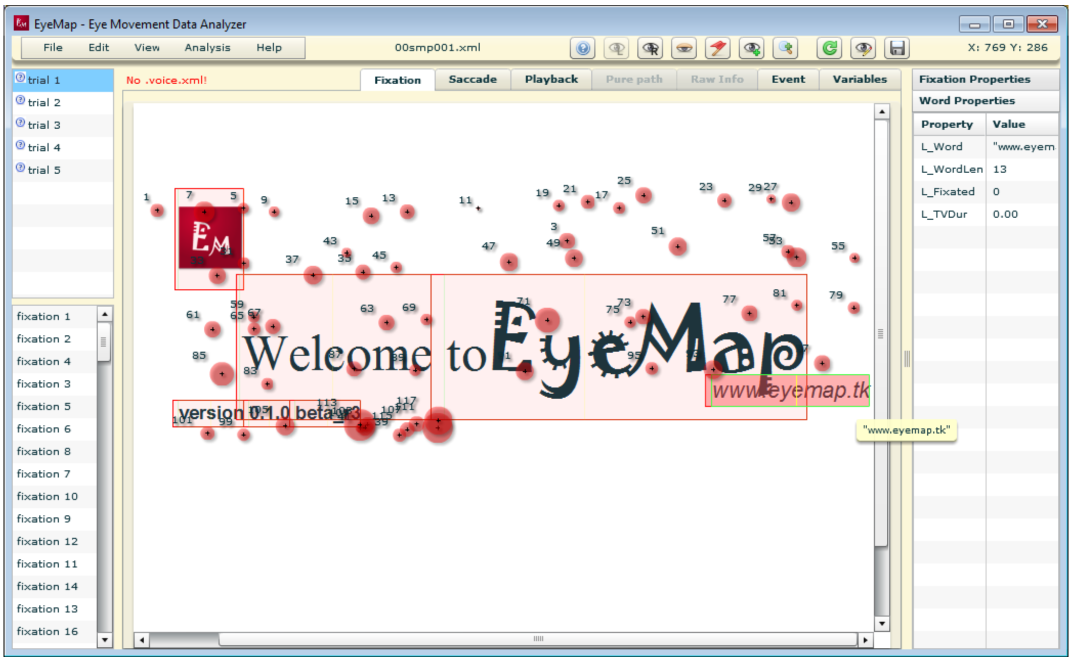

SR Data Viewer
In just over a month the deadly disease infected two-thirds of the population.
EyeMap
In just over a month the deadly disease infected two-thirds of the population.

Figure 6-6: Comparison of SR Data Viewer and EyeMap segmentation algorithms

There appear to be several problems with SR Data Viewer’s auto- segmentation algorithm. In Figure 6-6, it treats the two words “of the” as one. Performance deteriorates further when the algorithm has to work with colored texts. Also, the algorithm always draws the word boundary through the middle of the space between the two words, while reading researchers normally treat the space preceding a word as part of that word. Segmenting letters for other than mono-spaced fonts is also highly unreliable. EyeMap, on the other hand, can detect the boundary of every letter in the word, even for text materials comprising mixed proportional font types with letter kerning.

EyeMap achieves this superior performance by taking advantage of the Flex implementation platform. It provides an application programmer interface (API) to a wide range of functions for segmenting paragraphs into lines, lines into words, and words into characters. Flex does all this on the basis of font metrics rather than the analysis of an image, as is the case with Data Viewer. Note, however, that for word segmentation in writing systems that don’t explicitly indicate word boundaries, EyeMap must rely on a user-specified separator incorporated in the HTML text file.

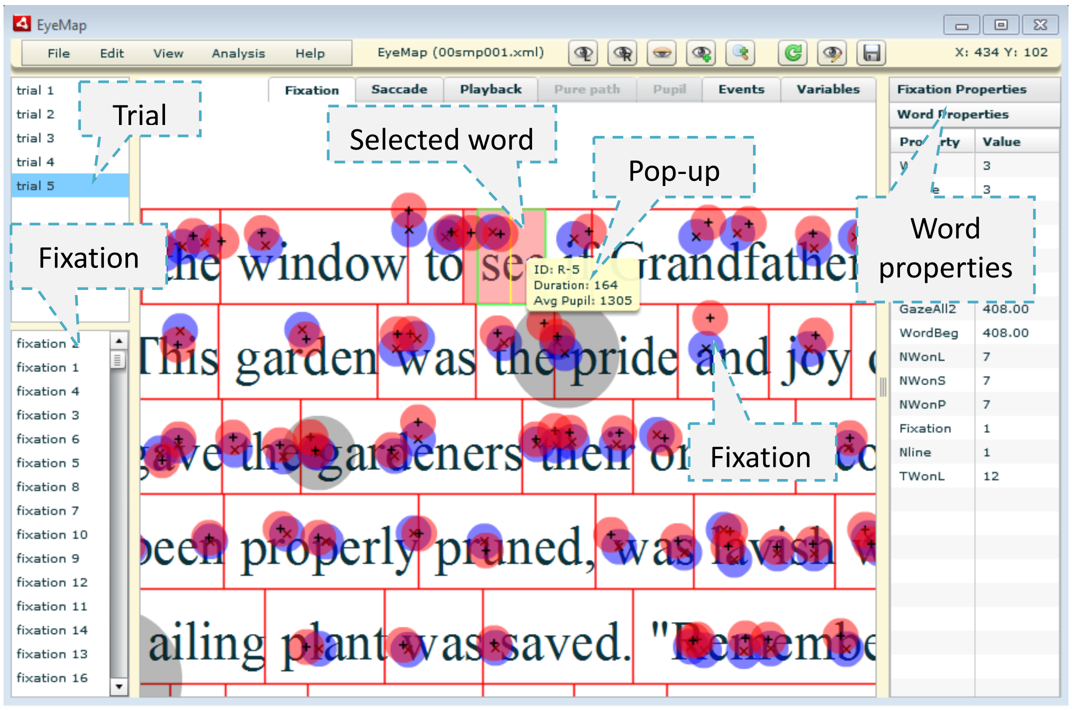
EyeMap also supports simple HTML presentation. Figure 6-7 illustrates how EyeMap can reproduce the web elements involving a mix of texts and pictures. The boundary of these elements can be detected by EyeMap automatically. This function can facilitate the data analysis of online text reading as well as the analysis of experiments where the materials contain different font types and font sizes.



Property	Value
L_Word	"www.eyem
L_WordLen	13
L_Fixated	0
L_TVDur	0.00

Figure 6-7: A text layout involving different font types, font sizes, and pictures

In addition, EyeMap offers one-dimensional, two-dimensional and dynamic data visualizations (replays), which allow for a wide range of visualization. These visualizations are implemented as 'viewers', representing a conceptual model of a particular visualization, with supporting panels. There are six different viewers, four of which are 2D-viewers. These are fixation viewer, saccade viewer, play-back viewer, and pure scan path viewer. The rest are 1D-charts, which comprise the pupil size viewer and an events viewer.

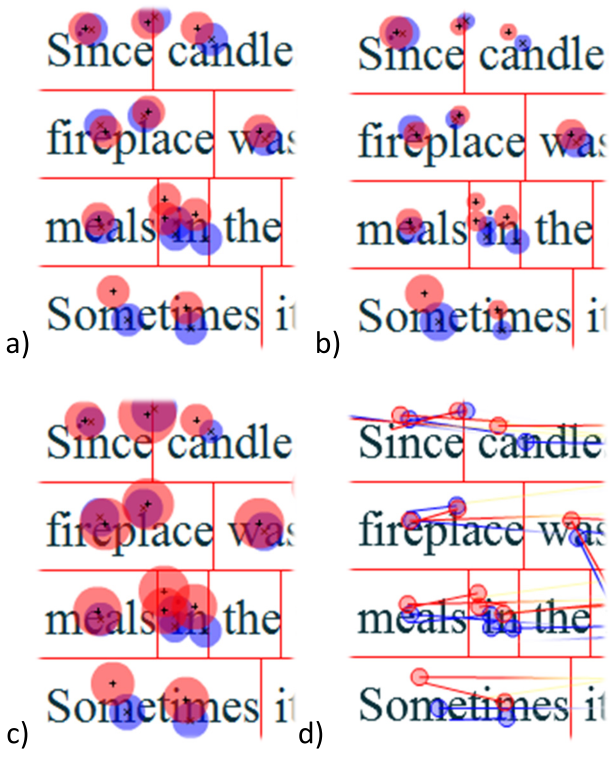


Pro rty	Value
W	3
e	3
GazeAll2	408.00
WordBeg	408.00
NWonL	7
NWonS	7
NWonP	7
Fixation	1
Nline	1
TWonL	12

Figure 6-8: A zoomed fixation viewer

An example of a two-dimensional viewer is the fixation viewer shown in the Figure 6-8. Its contents are dynamically changed according to the selected trial. It contains a text background that shows the stimulus page of text for the current trial. Fixations are represented by semi-transparent circles with different colors indicating the two eyes. When the user clicks the corresponding object (fixation dots, saccades, or words) inside the viewer, all the properties of that object are displayed. For example, when the word “see” is selected in Figure 6-8, the word level variables of “see” are calculated automatically and presented on the word properties list right of screen. The viewers also allow quick access to object data through pop-up tips. When the mouse cursor moves over an object, a tip with relevant information about the underlying object appears close to it.

Although the 2D viewers implemented in EyeMap have some features in common, they plot different information on the text stimulus. Figure 6-9 demonstrates six available visualizations from the same text stimulus page. The snapshots a) to c) are taken from the fixation viewer. The fixation dot size in b) and c) corresponds to different measures, with Figure 6-9b using fixation duration as the circle radius, while in Figure 6-9c the radius represents mean pupil size. Example d) is taken from the saccade viewer, which connects gaze point (mean fixation position) with a gradually colored line indicating saccade distance and directions. Furthermore, snapshot e) is a word heat map captured from a dynamic gaze replay animation in the playback viewer. f) depicts a pure scan path representing real saccade trajectories. Finally, Figure 6-9g is a fixation plot from the fixation viewer which uses color intensity to indicate fixation duration.



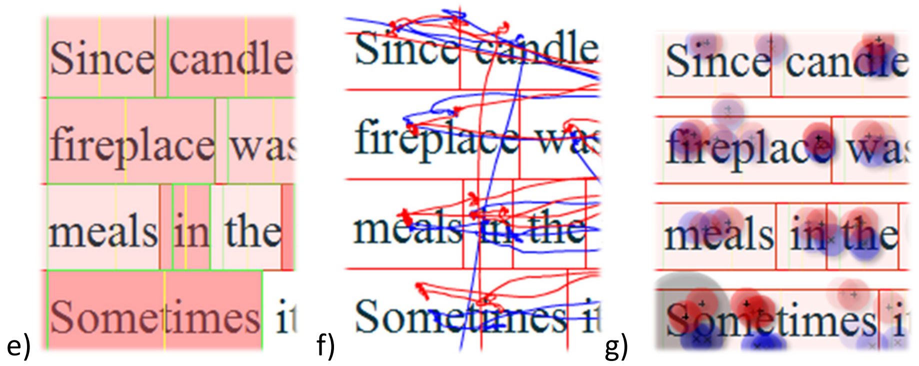


Figure 6-9: 2D eye movement plot

with: a) fixation dot with fixed circle radius, b) fixation durations as the circle radius, c) pupil size as the circle radius, d) gradated colored saccade lines with gaze point, e) word heatmap with word boundaries, f) pure scan path, g) color intensity indicating fixation duration

EyeMap also utilizes different visual effects to enhance each viewer. For example, a drop-shadow effect can be activated in fixation viewers, which can be useful in some situations. For example, Figure 9 shows some data collected from a patient with pure alexia using a letter-by-letter reading strategy (Ablinger, Huber, Schattka, & Radach, under revision), where the drop-shadow very clearly indicates the temporal unfolding and spatial overlap of multiple fixations on the same word.

EyeMap also has different visual effects to enhance each viewer. For example, a drop-shadow effect can be activated in fixation viewers which can be useful in some situations. For example, Figure 6-10 shows some data collected from an aphasic patient, where the drop-shadow shows the overlapped fixations more clearly.

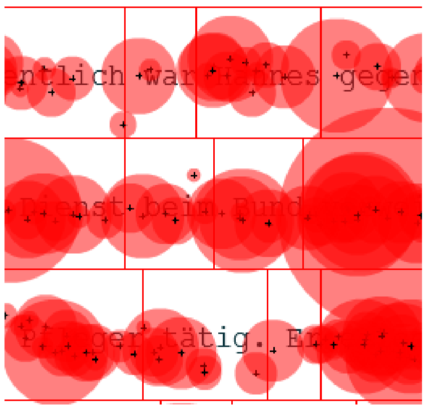
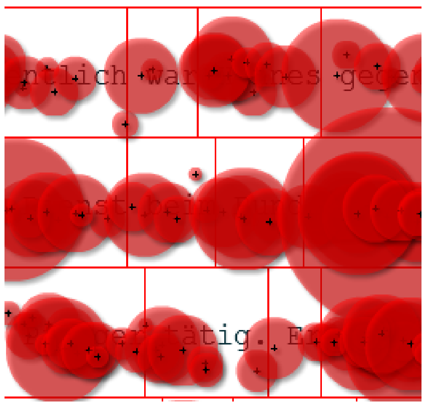
a)  b) 

Figure 6-10: Fixation dots with/without shadow drop

In the playback viewer, as can be seen from Figure 6-11, when gaze animation is performed over a static text background, the semi-transparent fixation dot has a ‘tail’ to highlight the most recent fixations. The playback viewer also supports exporting the eye movement animation to flash video (FLV) format with different play back speed.

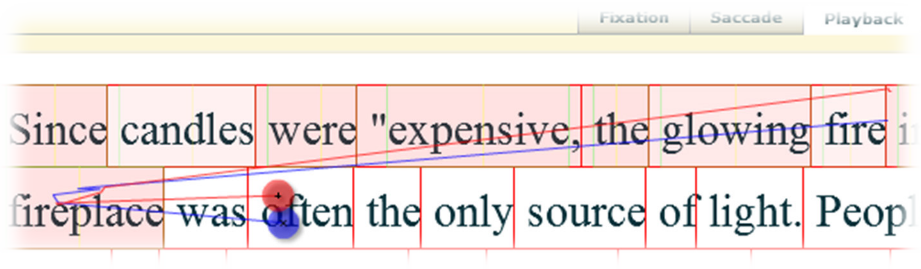
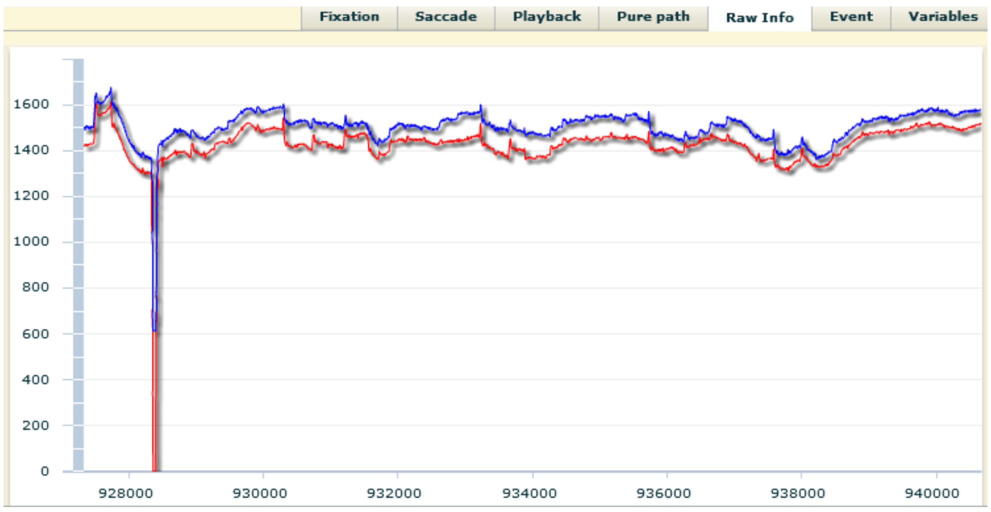
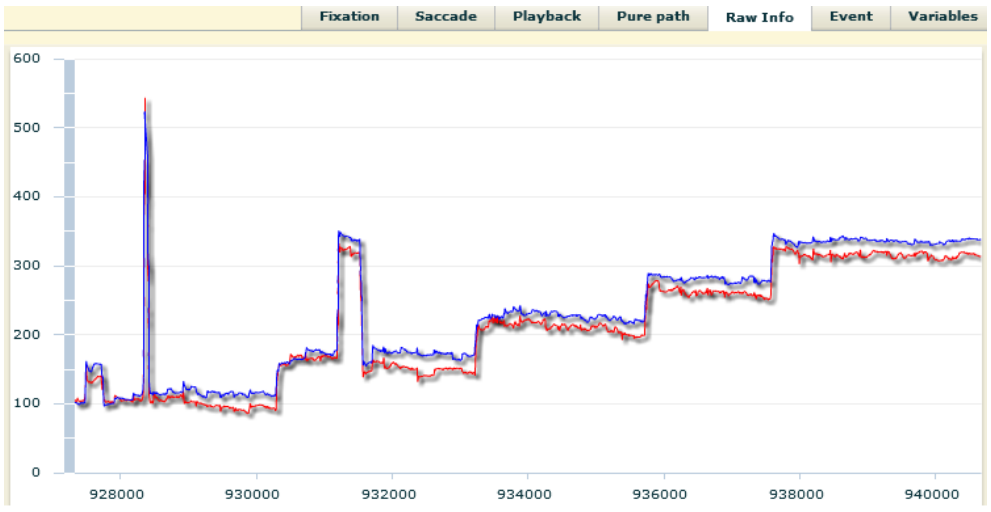

Fixation
Saccade
Playback
Since candles were "expensive, the glowing fire i
fireplace was often the only source of light. Peopl

Figure 6-11: Gaze playback with a virtual ‘tail’

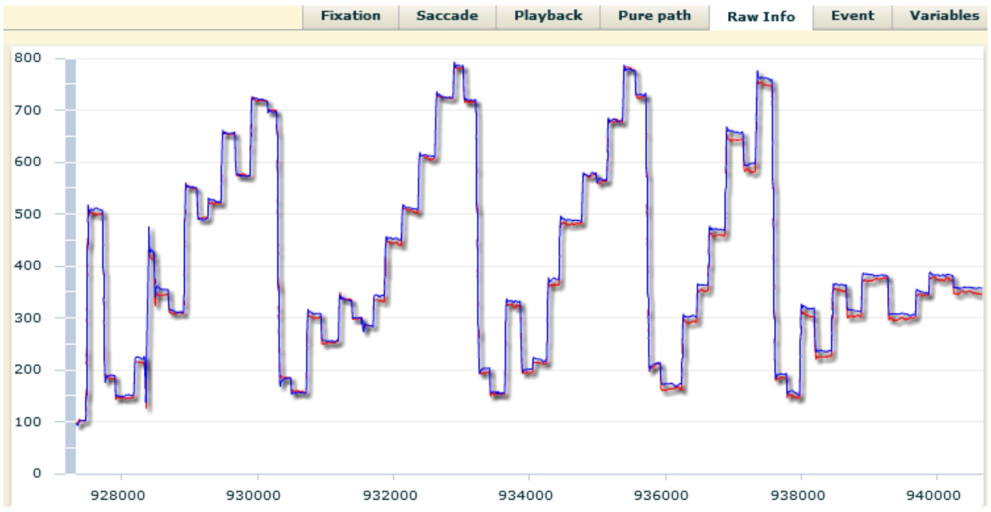
Time-correlated information, such as eye movement events, pupil dilation, mouse movements, keystrokes, stimuli changes, speech, etc., may be better represented in one-dimensional visualizations. Figure 6-12 and Figure 6-13 give an example plots of pupil dilation/eye movement x, y-coordinates against time as well as eye movement events vs. time in a one-dimensional chart, which we have implemented as a raw data information viewer and an event viewer. In the event viewer, time range is controlled by a time axis bar, the user can control the observation time by moving the slider on the time axis or dragging the mouse around chart.



a) Raw pupil size plot vs. time



b) Y coordinates plot vs. time



a) X coordinates plot vs. time

Figure 6-12: Eye movement raw data plots

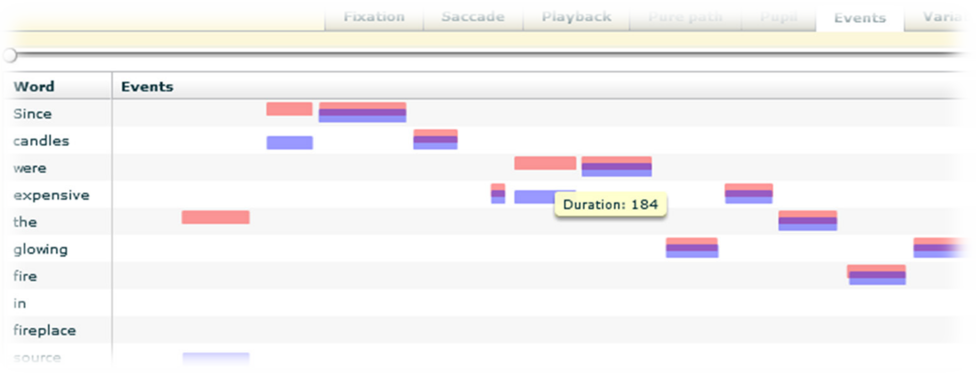


Figure 6-13: Fixation events on words as a function of time

In addition to visualizing reading data, direct data manipulation and modification through visualization is also supported by EyeMap. This function is extremely helpful in some cases, since fixation drift may occur during experimental trials despite good calibration, especially when working with children and patients. In a typical single-line reading experiment, a drift correction target dot may appear at the beginning of each trial and the spatial reference grid for the calibration would be shifted accordingly. However, in a multi-line text reading experiment, as shown in Figure 6-8, although the reading data is recorded by a relatively high quality tracking system (e.g., EyeLink II at 500Hz) with a drift correction before each trial, the data might not necessarily be clean, due to a small drift over time, especially in the vertical dimension. In such cases, manual drift correction can solve the problems of misalignment, when a few fixations cross over into the space assigned to the next line of text. As an illustration, Figure 6-14 presents an example of how a near-ideal scan path can be produced with little effort in offline drift correction. Note that we recommend using this feature only for small corrections in real research studies so that natural variability of eye movement patterns is not distorted.

EyeMap offers a convenient edit mode, which allows users do correct fixation locations without having to use the keyboard. In edit mode, users can select multiple dots by dragging a rectangle and align them by selecting a corresponding option from the right-click mouse context menu. To move the selected fixation dots up and down as one group, users can use the mouse wheel directly. A double click on the view will simply submit all the intended modifications.

It is also possible to align globally, in the horizontal axis, all fixations with the drift correction dot. This is only effective for displays involving single lines of text. Its advantage is that it operates on a complete data set instantaneously. However, if the experiment involves multi-line displays of text, alignment has to be done manually as described in the previous paragraph.

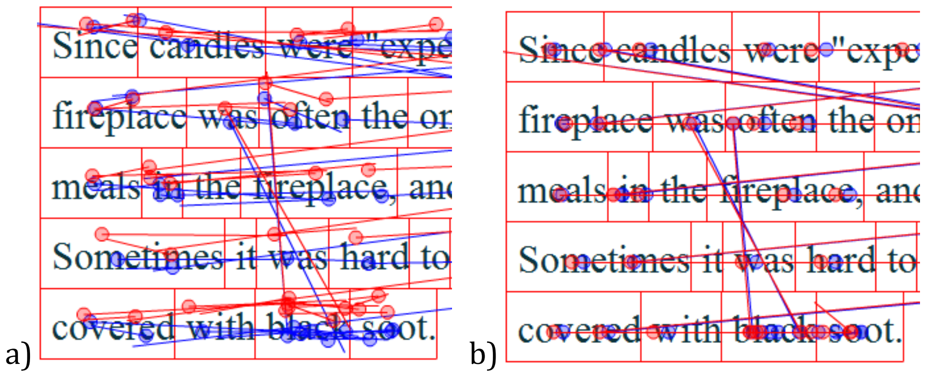


Figure 6-14: Eye scan path before and after the manual drift correction.

6.7. Data Export and Sharing

Ultimately, the whole point of a system like EyeMap is to generate a data matrix from an experiment that can be statistically analyzed. EyeMap provides an easy-to-use interface for the selection of appropriate variables at the fixation event and word level. Once a standard set of variables has been decided upon, these can be stored in the *export.xml* file and will be automatically loaded in future sessions. As already mentioned, EyeMap provides over 150 variables of relevance to the researcher (see Appendix A for the current list). The user can see the full list along with a brief description in EyeMap’s variable viewer (see Figure 6-15). The output data format is a CSV file with a header containing the exported variable names.

The fixation and word report output is dependent on the data tree module described in the data structure section. The exported variable names and their associated eye are listed in the user supplied *export.xml* file, which can also be created or modified in the EyeMap *export.xml* Editor (variables viewer). The *export.xml* file is loaded by EyeMap automatically. As shown in Figure 6-15, users can add variables to the export list by a drag-and-drop from the list of the available variables on the left. Variables can also be removed from the export list by similarly moving them to the Trash list on the right.

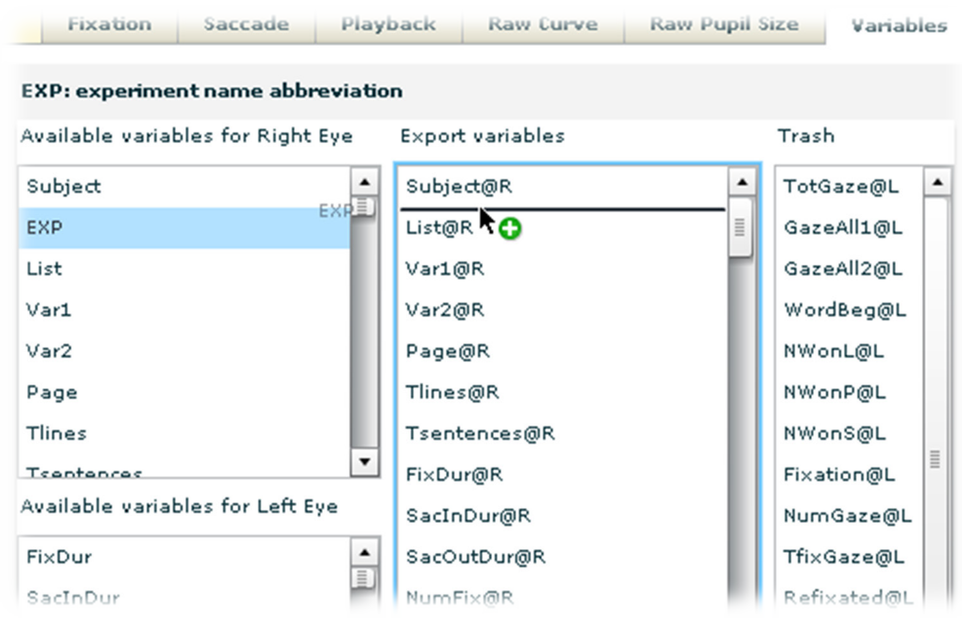
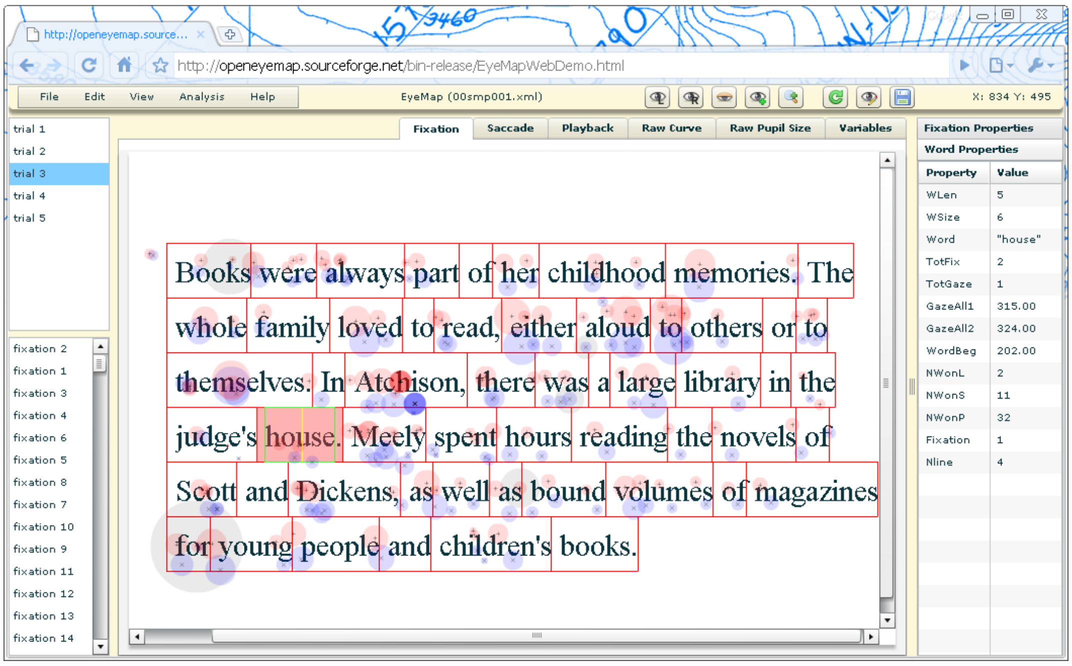


Figure 6-15: Creating an export variable list by drag-and-drop in the *export.xml* editor

Data sharing is important in all lines of research. However, the current situation with oculomotor reading research is that most research groups compile their text materials and collect eye movement data for their own purposes with limited data sharing. Large amounts of similar data are collected. In recent years, some research groups have started to create eye movement data corpora based on text reading studies that have been made available to fellow researchers (Kennedy & Pynte, 2005; Kliegl, Nuthmann, & Engbert, 2006). A long-term goal should be to develop online databases for all the researchers to share freely their experimental setups, methodologies, and data. As a significant step towards this goal we have created an online version of EyeMap at http://eyemaponline.tk. The application is presented in Figure 6-16 as running within a Chrome browser. Users can access the full functionality of EyeMap through all standard web browsers, which demonstrates that EyeMap can serve as a useful front-end tool for presenting text corpora and supporting the analysis of large sets of reading data. The current online version is not connected to any database as yet, so users currently cannot change the default data source. However, in the near future, the online system will allow users to upload and select their data source from a set of data corpora. This type of functionality is our ultimate goal and will motivate the design principle of future EyeMap versions.



Books were always part of her childhood memories. The whole family loved to read, either aloud to others or to themselves. In Atchison, there was a large library in the judge's house. Meely spent hours reading the novels of Scott and Dickens, as well as bound volumes of magazines for young people and children's books.

Property	Value
WLen	5
WSize	6
Word	"house"
TotFix	2
TotGaze	1
GazeAll1	315.00
GazeAll2	324.00
WordBeg	202.00
NWonL	2
NWonS	11
NWonP	32
Fixation	1
Nline	4

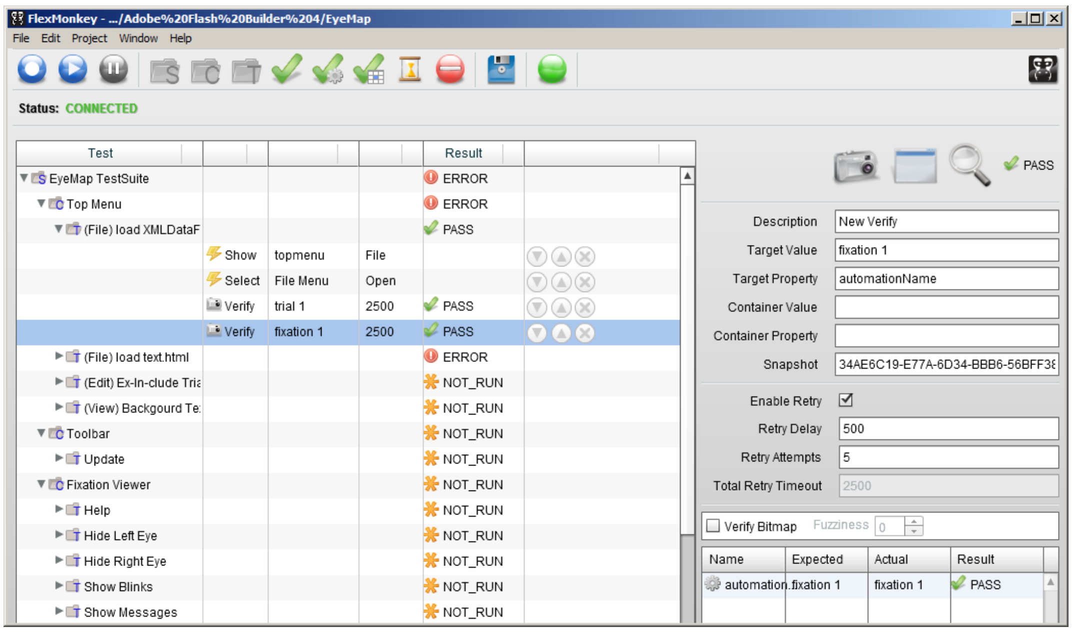
Figure 6-16: Online version of EyeMap running in a chrome browser

6.8. Implementation

To satisfy the requirement of platform independence, the system was built using several interpreted languages. Interpreted languages give applications some additional flexibility over compiled implementations, such as platform independence, dynamic typing, smaller executable program size, to list only a few. The data parser, which converts a given tracker's eye movement data into a device independent XML data format, is written in Java, built using JDK (Java Development Kit) 6 and packaged into an executable jar file. EyeMap itself is written in Adobe FLEX, and built using Adobe Flash builder 4.5.1. Adobe Flex is a software development kit (SDK) released by Adobe Systems for the development and deployment of cross-platform rich internet applications based on the Adobe Flash platform. Since Adobe Flash platform was originally designed to add animation, video, and interactivity to web pages, FLEX brings many advantages to developers wanting to create user-friendly applications with many different modes of data visualizations as well as integrated audio and animation, which fully satisfy the design demands of EyeMap. Although maintained by a commercial software company, a free Flex SDK is available for download, thus making source code written in Flex freely available.

6.9. Testing

Unit testing is an important step in software development. The Flex IDE Flash builder already embeds FlexUnit (TypeBased, 2011) as its unit testing framework. FlexUnit mimics the functionality of JUnit, a Java unit testing framework, and comes with a graphical test runner. It is designed for Flex and ActionScript 3.0 applications and libraries. However it is not easy to test EyeMap under FlexUnit, since many outputs of EyeMap are its visualizations.


FlexMonkey - .../Adobe%20Flash%20Builder%204/EyeMap
File Edit Project Window Help

Status: CONNECTED

Test				Result	
▼ EyeMap TestSuite				ERROR	
▼ Top Menu				ERROR	
▼ (File) load XMLDataF				✓ PASS	
	Show	topmenu	File		
	Select	File Menu	Open		
	Verify	trial 1	2500	✓ PASS	
	Verify	fixation 1	2500	✓ PASS	
▶ (File) load text.html				ERROR	
▶ (Edit) Ex-In-clude Tria				NOT_RUN	
▶ (View) Backgourd Te:				NOT_RUN	
▼ Toolbar				NOT_RUN	
▶ Update				NOT_RUN	
▼ Fixation Viewer				NOT_RUN	
▶ Help				NOT_RUN	
▶ Hide Left Eye				NOT_RUN	
▶ Hide Right Eye				NOT_RUN	
▶ Show Blinks				NOT_RUN	
▶ Show Messages				NOT_RUN	

✓ PASS

Description	New Verify
Target Value	fixation 1
Target Property	automationName
Container Value	
Container Property	
Snapshot	34AE6C19-E77A-6D34-BBB6-56BFF38
Enable Retry	✓
Retry Delay	500
Retry Attempts	5
Total Retry Timeout	2500

☐ Verify Bitmap Fuzziness 0

Name	Expected	Actual	Result
automation	fixation 1	fixation 1	✓ PASS

Figure 6-17: FlexMonkey UI testing interface

We therefore use FlexMonkey (Logic., 2011) as our testing tool. FlexMonkey is an Adobe AIR application released on open source GNU General Public License. It is used for testing Flex and AIR based applications, which provides the functionality to record, playback and verify Flex UI interactions. FlexMonkey is the de-facto standard for automated testing of Adobe FLEX and AIR applications (Kaushal & Kaur). Its test automation tool can generate Action Script-based testing scripts that users can easily include within their development environments. As shown in Figure 6-18, EyeMap user interface was tested by FlexMonkey.

6.10. Comparison and Conclusion

Prior to EyeMap, the best known and most popular eye movement data analysis tool for reading was probably EyeLink DataViewer, However, as a general-purpose eye movement data-processing software in the reading area, it still has some disadvantages. First, from the visualization point of view, EyeMap is fully compatible with the EyeLink data format and has almost comparable visualization modes as those provided by DataViewer. Moreover, EyeMap has more precise AoI extraction from text stimulus (DataViewer uses pixel-based AoI extraction, which cannot process colored words properly.) On the other hand, from an analysis aspect, although DataViewer provides many export variables, most of them are too general for a reading psychologist's purpose. Most importantly, it cannot combine binocular data; left eye and right eye have to be analyzed separately. In contrast, EyeMap can generate combined reports for binocular data. Finally, from a more general perspective, DataViewer is relatively expensive for academic researchers. It can also be inconvenient to use because running it requires a hardware "dongle" to be used on each machine on which the software is installed. It is also limited to running on Windows and Mac platforms. In contrast, EyeMap is free you can share your data with other researchers and do your data analysis on any platform. Further comparisons of the two software systems are provided in Appendix A.

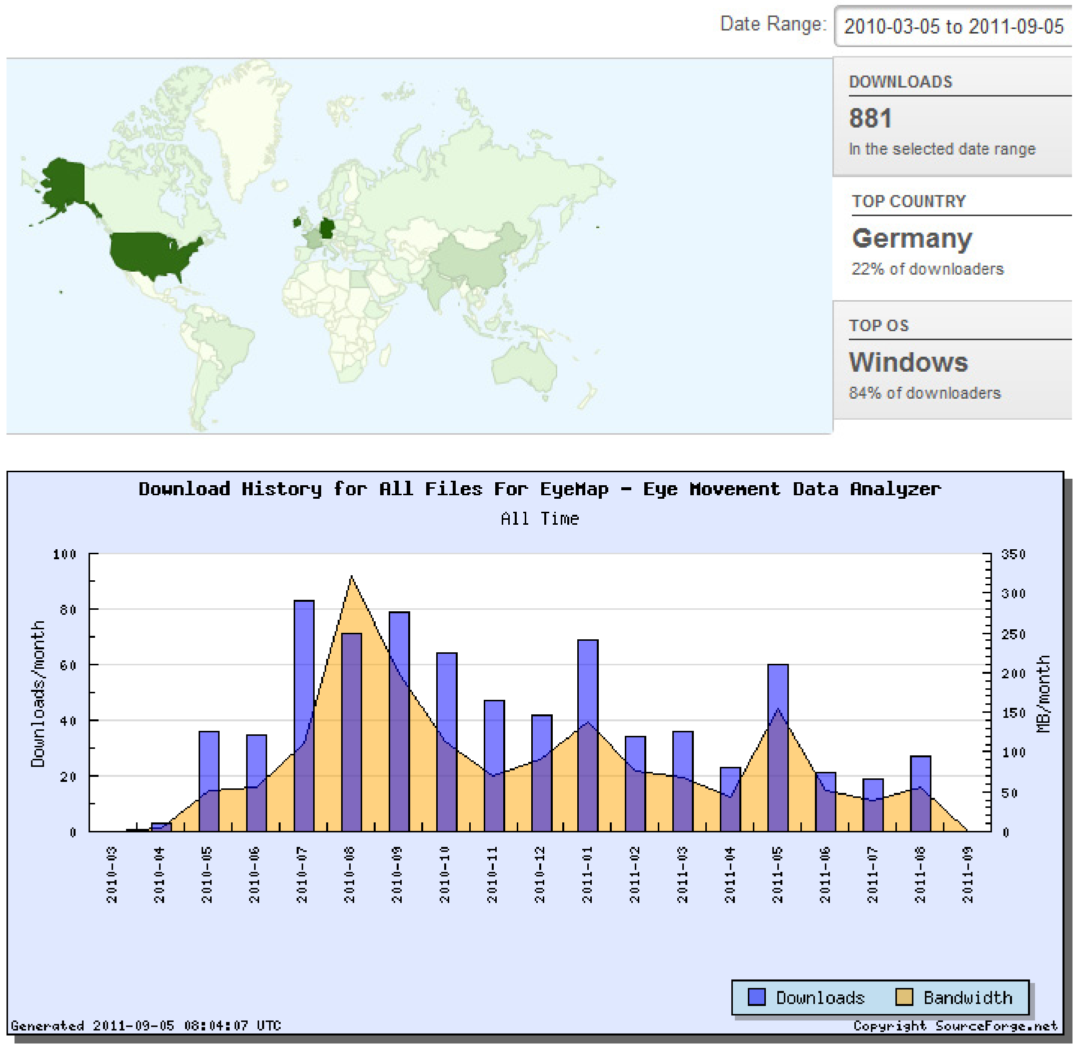
In addition to manually checking the mapping of fixation location to word and letter locations, we compared the variables created in EyeMap to the analysis output from DataViewer. For this validity check, we randomly picked a participant dataset from a current sentence reading study consisting of 132 trials. For the EyeMap analysis, the EDF output was converted into XML using EyeMap's edf-asc2xml converter. Word regions were created automatically, based on the information in the text.html file. Fixation and Word Reports were exported and the resulting CSV files were read into SPSS.

For the DataViewer analysis, EDF data were imported into DataViewer. As the auto-segmentation function in DataViewer did not create word boundaries immediately following the last pixel of the last letter in a word, interest areas had to be defined manually for each trial. Fixation, Saccade, and Interest Area Reports were exported and resulting Excel files were read into SPSS. Variables from both outputs were compared and showed exact matches. The table in Appendix A presents results for some key variables for different word lengths for this one subject.

The key innovation in EyeMap involves the development of a freely available analysis and visualization platform specifically focused on eye movement data from reading studies. This focus has allowed for the development of (1) an XML-based open standard for the representation of data from a range of eye tracking platforms; (2) the implementation of full Unicode support, significantly improving the ease with which different writing systems can be studied; (3) the development of robust techniques for text handling, such as the automatic segmentation of text in both proportional and non-proportional fonts into areas of interest; (4) the development of support for integrating a speech stream with the eye movement record; and (5) a facility for analyzing binocular data in a more integrated way than has heretofore been possible.

While the component technologies of the system are not new, their combination represents a unique and powerful reading analysis platform.

So far, as shown in the Figure 6-18, after we released the first EyeMap installation package through sourceforge.net (https://sourceforge.net/projects/openeyemap) for free download, the software has been downloaded over 800 times from 56 different countries and areas other than Ireland. The project has an average top 10% rank (which is calculated based on download and web traffic by sourceforge.net) among the 40,000 sourceforge.net projects.



Date Range: 2010-03-05 to 2011-09-05

DOWNLOADS
881
In the selected date range

TOP COUNTRY
Germany
22% of downloaders

TOP OS
Windows
84% of downloaders

Figure 6-18: Downloads and bandwidth statistics of EyeMap project

(From March 5, 2010 to September 5, 2011 by Sourceforge.net)

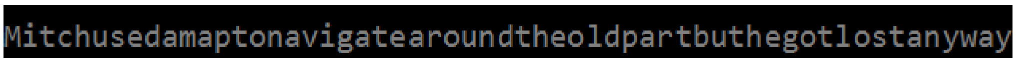
Part III. Experiments with the system

The following sections provide an overview of several experiments conducted using our software suite. The purpose of these experiments was two-fold. In the first place they were designed to answer scientific questions about the nature of the reading process. Second, they provided an opportunity to test the ability of the system to handle complex experimental design, fast stimuli presentation, data analysis and visualization. A sample walkthrough from experimental design to data analysis is given in Appendix A.

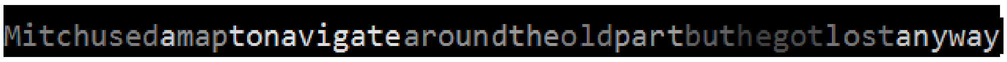
Chapter 7

Unspaced English and Chinese Reading with Shimmer as Boundary

During the two previous experiments, we noticed that asynchronous shimmer makes each word stand out from each other, thus serving as a word boundary. We therefore tried to use this new technique to explore the role of explicit word segmentation in reading two writing systems: English and Chinese. The new experiment involved shimmering the words in a line of text using a random pattern of luminance variation. The letters in a word are shimmered synchronously, while each word is shimmered asynchronously. The effect of this is to make the letters in a word appear as a unit and the words to appear visually distinct from each other. This has a somewhat similar effect to that observed by (Perea & Acha, 2009) who used alternating bold and normal fonts for adjacent unspaced English words, and (Bai, Yan, Liversedge, Zang, & Rayner, 2008) who used alternating white and grey backgrounds for adjacent unspaced Chinese words. As shown in Figure 7-1, our shimmer paradigm, however, has the advantage, of not introducing different visual weightings for words, which could affect saccade targeting. In the case of the shimmer paradigm all words have the same visual weight. In addition, with the non-space segmented Chinese writing system, we can measure the effects of explicit word segmentation in Chinese without having to introduce spaces. We therefore designed and preformed two experiments, one involving the reading of unspaced English and the other of Chinese under synchronous and asynchronous shimmer conditions. (See http://www.eyemap.tk/unspaced/english(5-25-2010).avi and http://www.eyemap.tk/ unspaced/chinese(5-25-2010).avi for sample videos of the two sets of stimuli).


Mitchusedamaptonavigatearoundtheoldpartbuthegotlostanyway

a). Synchronously shimmer of an unspaced English sentence


Mitchusedamaptonavigatearoundtheoldpartbuthegotlostanyway

b). Asynchronously shimmer of an unspaced English sentence

Figure 7-1: Example of synchronously/asynchronously sentence

7.1. Experimental Methods

The first experiment comprised 72 experimental sentences with spaces removed from the original text. The average length of the sentences was 57 characters. Each sentence was presented in Consolas at 19pt on a black background using the shimmer technique. The experimental equipment and setup were identical to the first shimmer experiment described in section 5.7.4 and 5.7.5. Four different counterbalancing lists were prepared. Half of the sentences were presented using asynchronous shimmer, while another half were shimmered synchronously. Asynchronous/synchronous sentences were blocked and presented in different orders. A horizontal three-point calibration was performed at the start of the experiment followed by a three-point calibration validation. A single-point drift correction was performed before each sentence was read. Throughout the reading of each sentence, the experimenter was able to view on a separate monitor the text the participant was reading, overlaid with a cursor corresponding to real-time gaze position. If the experimenter judged that tracking accuracy had declined, a full calibration was initiated before the next screen. Participants were instructed to read the sentences for comprehension. They had to complete six practice trials before proceeding to the main experiment. After reading each sentence, participants pressed a button to end the trial. On some trials, this button press could result in the presentation of a multiple choice question which the participant responded to by pressing one of four alternative buttons on a game controller. After answering the question, subjects proceeded to the next sentence. Eighteen questions were prepared, and were dispersed equally throughout the two blocks. This resulted in one question for every four sentences roughly. Twelve participants with either normal or corrected vision took part in the experiment. All were native speakers of English and under-graduate students from the NUI Maynooth community. None had participated in either of the previous shimmer experiments.

7.2. Results of Unspaced English Reading

Experimental data were filtered out in following condition:

1) Fixations less than 80ms or greater than 800ms duration;
2) Fixations preceded or followed by a blink;
3) Fixations on the first or last word in a line.

First fixation duration and gaze duration were calculated. We also analysed landing site effects, since (Perea & Acha, 2009) found that the reader’s eye tends to land nearer to the beginning of the word when reading unspaced English.

7.2.1. First fixation duration

First fixation duration as a function of shimmer was plotted in Figure 7-2, and a mixed-effects model (D. Bates & Sarkar, 2007) was used for this analysis. The results are shown in Table 7-1, and they suggest that the fixed effects on shimmer patterns were significant. For the first fixation, landing on async shimmering words costs about 23.5 ms longer than synch ones.

Table 7-1: Fixed effects on first fixation duration of unspaced English reading

Fixed effects contrasts	β (ms)	SE	t
Shimmer (synch - async)	-23.54	9.82	**-2.40**
Log word frequency	-3.70	1.75	**-2.11**
Shimmer x Log word frequency	3.94	2.24	1.78

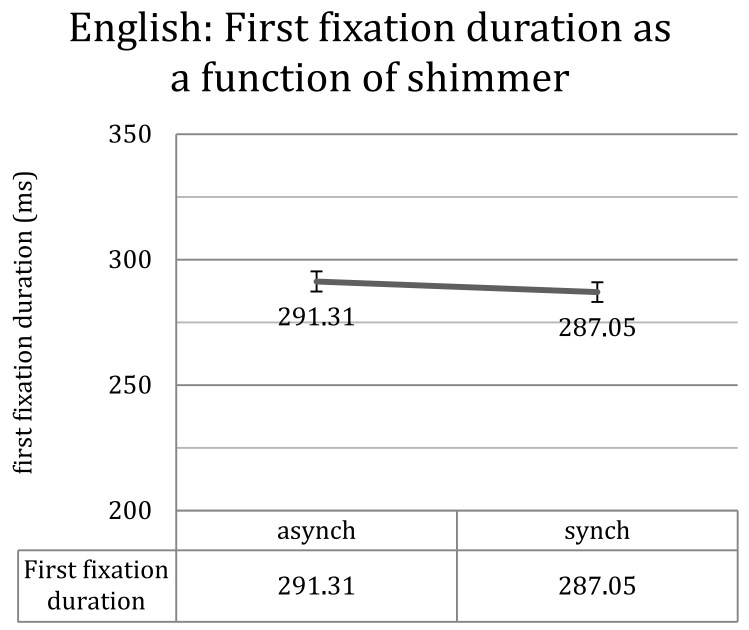


Figure 7-2: First fixation duration as a function of shimmer (unspaced English)

7.2.2. Gaze Duration

Again a mixed-effects model (D. Bates & Sarkar, 2007) was applied to this analysis. Table 7-2 presents several fixed effects on gaze duration. For both word frequency and shimmer pattern the effects were significant. Gaze duration as a function of shimmer was plotted in Figure 7-3. It shows that reading is much slower in synch mode (participants spend 57 ms longer on a word when they read unspaced text shimmered synchronously). This is most likely due to there being nothing to help their word segmentation in synchronised mode. Small but significant gaze duration by frequency interaction was also found in this analysis, and a further analysis of this effect is shown in Figure 7-4.

Table 7-2: Fixed effects on gaze duration of unspaced English reading

Fixed effects contrasts	β (ms)	SE	t
Shimmer (synch - async)	56.94	12.53	**4.54**
Log word frequency	-21.25	3.32	**-6.40**
Shimmer x Log word frequency	-5.90	2.98	**-1.98**

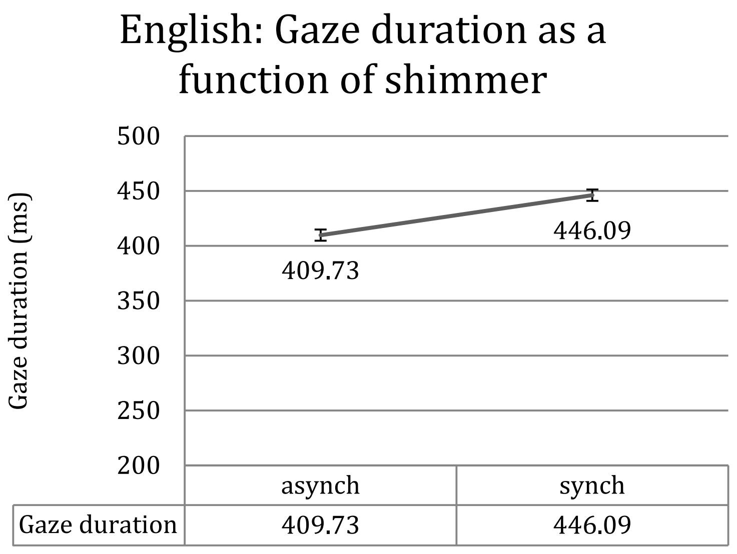
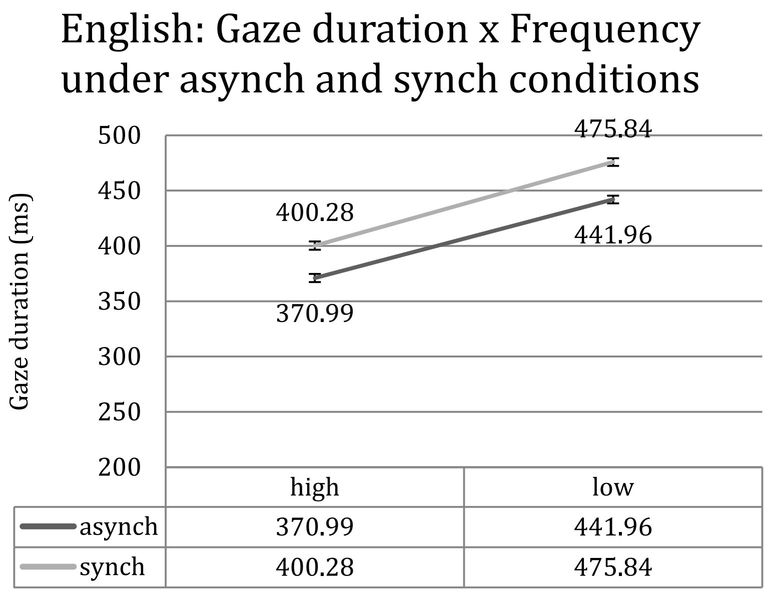
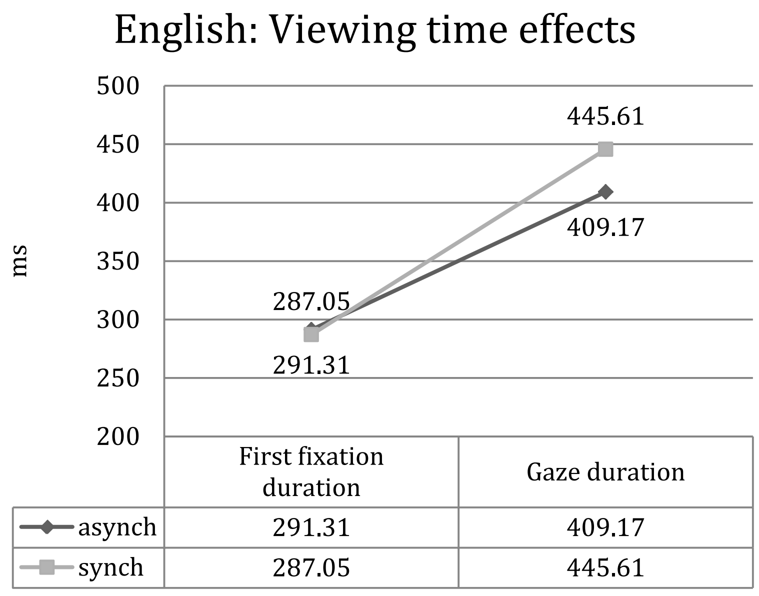


Figure 7-3: Gaze duration as a function of shimmer (unspaced English)



	high	low
asynch	370.99	441.96
synch	400.28	475.84

Figure 7-4: Gaze duration by frequency interaction for under async and synch conditions



	First fixation duration	Gaze duration
asynch	291.31	409.17
synch	287.05	445.61

Figure 7-5: Viewing time effects on unspaced English reading under async and synch conditions

7.2.3. Discussion on the viewing time effects

The findings on gaze duration are consistent with those from the alternating bold paradigm (Perea & Acha, 2009) and it appears that shimmer plays a similar role to alternating bold case. However, the results of first fixation duration (FFD) and gaze duration (GD) seem to conflict with each other. Figure 7-5 illustrated the average FFD and the average GD under different shimmer patterns. All the above-mentioned results suggest that compared to async mode, participants have shorter FFDs but longer GDs for synch mode. Dramatically decreasing of viewing time on gaze duration suggests that readers make less refixations on the same word. The corollary of this is that the reader makes more single fixations in async mode. As we know, single fixations are usually longer than the first fixa-tion of a gaze which contains more than one fixation. It has been argued (Rayner, Reichle, & Pollatsek, 1998) that this is because if the eye lands in a non-optimal location in a word, it will rapidly make a new refixation, thus creating a short first fixation. However, if it lands in an optimal location, it may only make one fixation, but this will be a longer fixation because more work is done.

Table 7-3: Fixed effects on the number of fixations on word under different shimmer condition

Fixed effects contrasts	**β (ms)**	**SE**	**t**
Shimmer (synch - async)	3.52	0.35	**10.16**

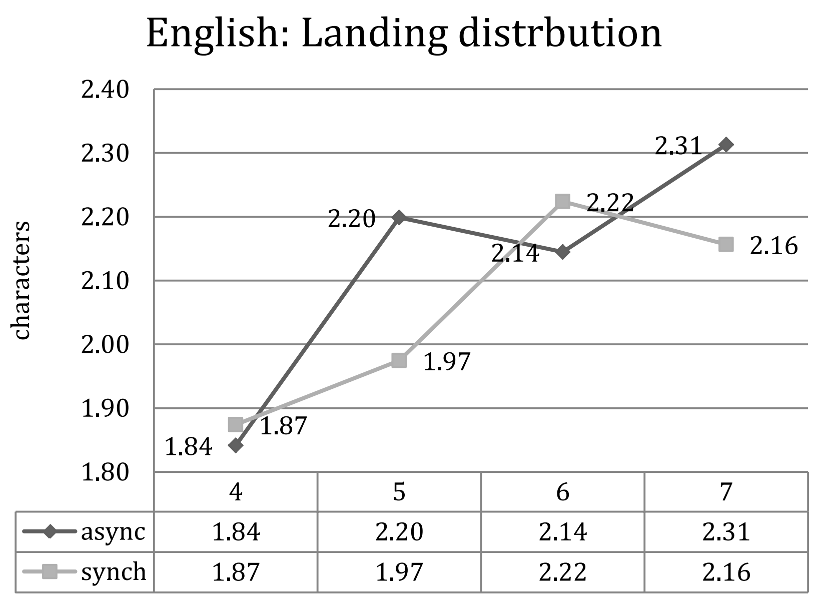
Further analysis illustrated in Table 7-3 showed that reader made about 3.5 more fixations on the word in synch mode than async mode. Moreover, another logistic mixed effects analysis result showed that shimmer to be highly significant with more single fixations for async than synch (z value = -4.805; Pr(>|z|) = 1.55e-06).

7.2.4. Landing Site effects

Similarly to Perea and Acha (2009) did, we also looked into the effect of initial landing position on subsequent processing of the target word. Factors of word length, initial landing position, and shimmer patterns were investigated. The results of all effects and their interactions are shown in Table 7-4. We can see an interesting but short-of-significance effect in landing site and as a function of shimmer patterns. Further analysis of this effect was plotted in Figure 7-6. In Figure 7-6, for word length 5 and 7, eye tends to land nearer to the beginning of the word, while for word length 4 and 6, there is no significant difference between async and synch modes. This phenomenon (eye tends to land nearer to the beginning of the word) was also consistent with findings of (Perea & Acha, 2009).

Table 7-4: Fixed effects on landing site of unspaced English reading

Fixed effects contrasts	β (ms)	SE	t
Shimmer (synch - async)	-0.40	0.22	-1.84
Word length	0.24	0.02	**12.28**
Launch distance	0.16	0.02	**7.20**
Shimmer x Word length	0.03	0.03	0.95
Shimmer x Launch	-0.04	0.04	-1.02
Word length x Launch	0.02	0.00	**6.44**
Shimmer x Word length x Launch	0.01	0.01	**1.68**



	4	5	6	7
async	1.84	2.20	2.14	2.31
synch	1.87	1.97	2.22	2.16

Figure 7-6: Landing distribution of unspaced English reading under async and synch conditions

7.2.5. Chinese Reading Experiment

Chinese is a logo-syllabic script, and there is no space between each word. Without explicit word boundary, how the Chinese reader deals with segmentation is still unclear. For this experiment, the reading materials were taken from the Beijing Sentence Corpus, which was used in (Yan, Kliegl, Richter, Nuthmann, & Shu, 2010) and were selected from the People's Daily, a Mandarin newspaper. One hundred and fifty experimental sentences were presented in the Song font at 24pt, and were blocked into two groups. The design was the same as the unspaced English reading. Half of the sentences were presented in asynchronous shimmer, while the other half were shimmered synchronously. Asynchronous/synchronous sentences were also blocked and presented in different order. Thus, the shimmer types as well as their presentation order created four different counterbalancing lists. Twenty-five percent of sentences were followed by a simple question. The same apparatus setup and experimental procedures were adopted as with the unspaced English reading. Fourteen subjects were involved in the Chinese phase of the study with either normal or corrected vision. They were native Chinese speakers and were post-graduate students from NUI Maynooth community. None of them had participated in previous experiments.

7.3. Results of Chinese Reading

Experimental data were filtered in the same way as the previous unspaced English reading data. The only difference in analysis is that saccade amplitudes were measured in half characters.

7.3.1. First fixation duration

First fixation duration as a function of shimmer in Chinese reading is plotted in Figure 7-7. Similar to the last experiment, a mixed-effects model was applied to this analysis. The results are shown in Table 7-5. They suggest that there was no significant effect for all the factors except word frequency. A further analysis on the first fixation duration as a function of shimmer (Figure 7-7) shows that asynchronous shimmer did not help Chinese reader. This finding differs from unspaced English reading, but is consistent with (Bai, et al., 2008) finding, where they found that alternately highlighting word background neither facilitates nor interferes with Chinese reading.

Table 7-5: Fixed effects on first fixation duration of Chinese reading

Fixed effects contrasts	β (ms)	SE	t
Shimmer (synch - async)	-5.30	5.79	-0.91
Log word frequency	-3.38	1.11	**-3.04**
Shimmer x Log word frequency	-0.42	1.46	-0.29

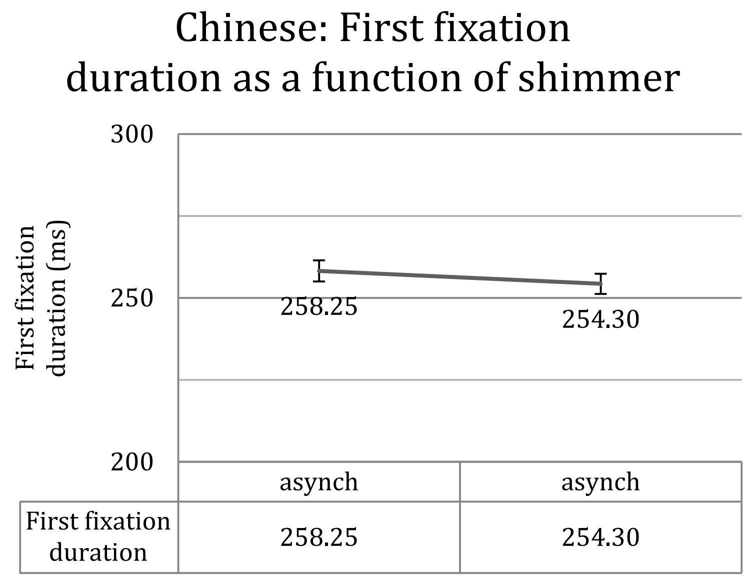


Figure 7-7: First fixation duration as a function of shimmer (Chinese)

7.3.2. Gaze Duration

Further analysis on gaze duration proposed the same results. A further analysis on the first fixation duration as a function of shimmer (Figure 7-7) showed that for gaze duration, there was no significant effect for all the factors except the word frequency. A further analysis on the first fixation duration as a function of shimmer (Figure 7-7) suggests a similar result.

Table 7-6: Fixed effects on gaze duration of Chinese reading

Fixed effects contrasts	β (ms)	SE	t
Shimmer (synch - async)	-7.60	7.56	-1.01
Log word frequency	-27.25	2.21	**-12.31**
Shimmer x Log word frequency	1.17	1.94	0.60

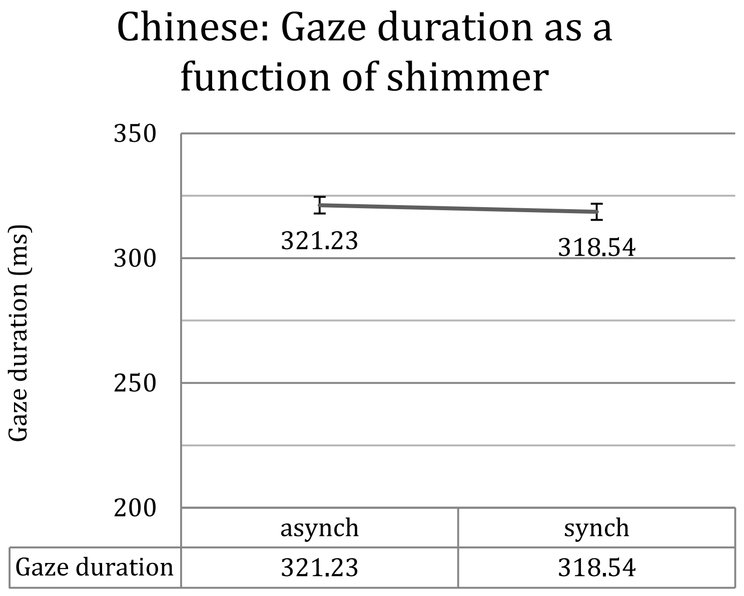
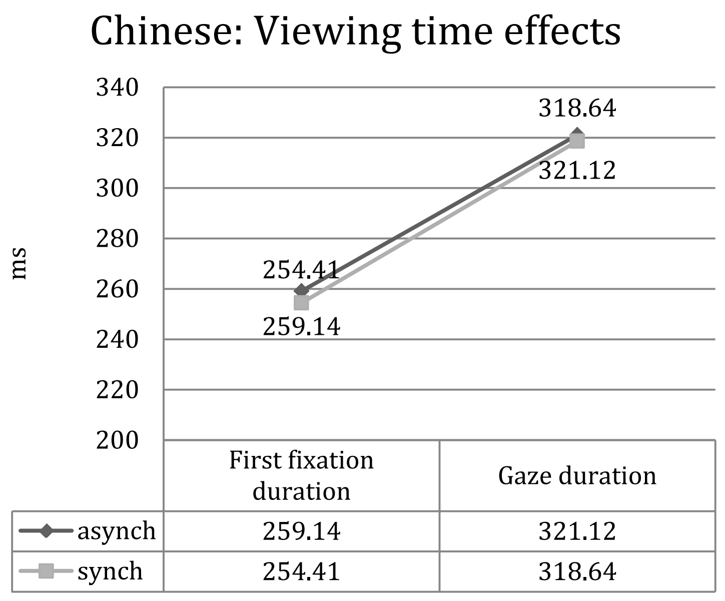


Figure 7-8: Gaze duration as a function of shimmer (Chinese)

7.3.3. Discussion on the viewing time effects

Two viewing time effects suggested that there was no significant difference between the different shimmer patterns. Figure 7-9 illustrated the average first fixation duration (FFD) and the average gaze duration (GD) under two shimmer patterns. Both FFDs and GDs were slightly longer in async mode, which is the same finding obtained in our second shimmer experiment, where the results suggested there was small advantage in reading under synchronous shimmer than asynchronous. The overall results are consistent with Bai, *et al.*'s (2008), where they found that alternately highlighting word background neither facilitates nor interferes with Chinese reading.



	First fixation duration	Gaze duration
asynch	259.14	321.12
synch	254.41	318.64

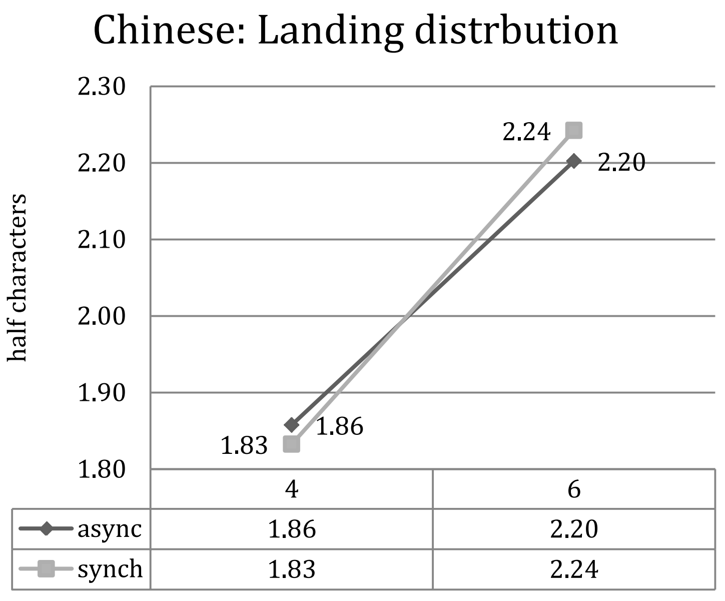
Figure 7-9: Viewing time effects on Chinese reading under async and synch conditions

7.3.4. Landing Site effects

Further analysis on landing site was also performed in order to make a comparison with unspaced English. Saccade amplitudes were measured in half characters, and only 2 and 3 character words were selected. The landing distribution of Chinese reading under async and synch conditions was shown in Figure 7-10. There is not much difference between 2 and 3 character words on landing. The fixed effects were also presented by Table 7-7, no significant landing site effects due to shimmer were found for Chinese readers.

Table 7-7: Fixed effects on landing site of Chinese reading

Fixed effects contrasts	β (ms)	SE	t
Shimmer (synch - async)	-0.03	0.19	-0.14
Word length	0.42	0.02	**17.52**
Launch distance	0.15	0.02	**8.65**
Shimmer x Word length	-0.03	0.04	-0.67
Shimmer x Launch	-0.00	0.03	-0.14
Word length x Launch	0.05	0.00	**11.38**
Shimmer x Word length x Launch	-0.01	0.01	-0.77



	4	6
◆ async	1.86	2.20
■ synch	1.83	2.24

Figure 7-10: Landing distribution of Chinese reading under async and synch conditions

7.4. Conclusion

This study proposed a new technique using shimmer as a segmentation tool to help readers to identity word boundaries. Two separate experiments on unspaced English and Chinese under different shimmer conditions were performed. For unspaced English reading, shorter fixation durations in synchronous compared to asynchronous mode is most likely due to misplaced initial landings followed by a rapid refixation, which in turn gives rise to longer gaze durations. Synch mode is therefore consistent with reading unspaced English. While in async mode, shimmer provides usable boundary information which greatly reduced the gaze duration on words. In contrast, for Chinese reading, there are no significant effects due to type of shimmer on viewing time as well as landing site, which accords with Bai, *et al.*'s (2008) findings.

The findings on Chinese reading again suggested that it seems quite unlikely that a lifetime of reading experience without spaces could be quickly overcome via the insertion of word segmentation evidence between words. While for English readers, our findings suggest that word segmentation evidence is very important to them. The overall result suggests that the shimmer paradigm has similar properties to the alternating bold and highlighting paradigms, but without the potential imbalance of visual weight, which may confound participant's attention of these two paradigms.

Chapter 8

Font Study

The second major study performed with our informatics system was exploring the influence of font legibility on reading. It is generally accepted that typefaces affect text readability. In the typographic literature, for example, features such as serifs are believed to have some positive impact on legibility (Burt, 1959; Rubinstein, 1988).

Serifs are semi-structural details on the ends of some of the strokes that make up letters and symbols. There has been considerable debate on the relative legibility of serif and sans serif typefaces since their creation. Some researchers consider that serifs give readers extra information about the letters which increases letter legibility and therefore helps reading (McLean, 1980; Rubinstein, 1988). Some argue that the extra information from serifs incurs a processing cost and therefore slows down reading, especially when reading in non-optimal viewing conditions, such as from a computer screen (Bernard, Chaparro, Mills, & Halcomb, 2003; Dillon, 1992), when the font size is very small (Morris, Aquilante, Yager, & Bigelow, 2002), when readers have reading difficulties or suffer from impaired vision (Russell-Minda et al., 2007), or when readers are learning to read (Coghill, 1979). While many others believe that there is no difference in legibility between serif and sans-serif typefaces (Arditi & Cho, 2005; De Lange, Esterhuizen, & Beatty, 1993; Moriarty & Scheiner, 1984; Tullis, Boynton, & Hersh, 1995; Zachrisson, 1965). However this latter research is primarily based on reaction time studies. There have been few if any studies involving modern eye tracking technology with the recent notable exception of Slattery and Rayner (2010).

In order to explore the effect of serifs on a typeface and compare the legibility of serif and sans serif typefaces, we selected the most representative examples of the two typeface categories: Times New Roman (TNR) and Arial. These two typefaces are so common that their legibility has already been studied separately or in contrast to other typefaces by many researchers (Bernard, et al., 2003; Bernard, Liao, & Mills, 2001; Hill & Scharff, 1997; Kingery & Furuta, 1997; Sheedy, Subbaram, Zimmerman, & Hayes, 2005). These studies have suggested that Arial is more readable and legible for on-screen reading and for reading under conditions of low contrast. However, it is unclear what typeface features give rise to Arial's enhanced legibility.

In order to eliminate the potential confounding effects of character features other than serifs on legibility, we created two typeface variations of TNR and Arial. As shown in Figure 8-1, we first removed serifs from TNR characters (46 letters were modified, the remaining characters were kept the same) and thus created a TNR variant we called Times New Roman Sans Serif (TNRSS). Then we attached these serifs to the Arial font on corresponding letters with corresponding font size, creating an Arial variant, which we called Arial Serif (AS).

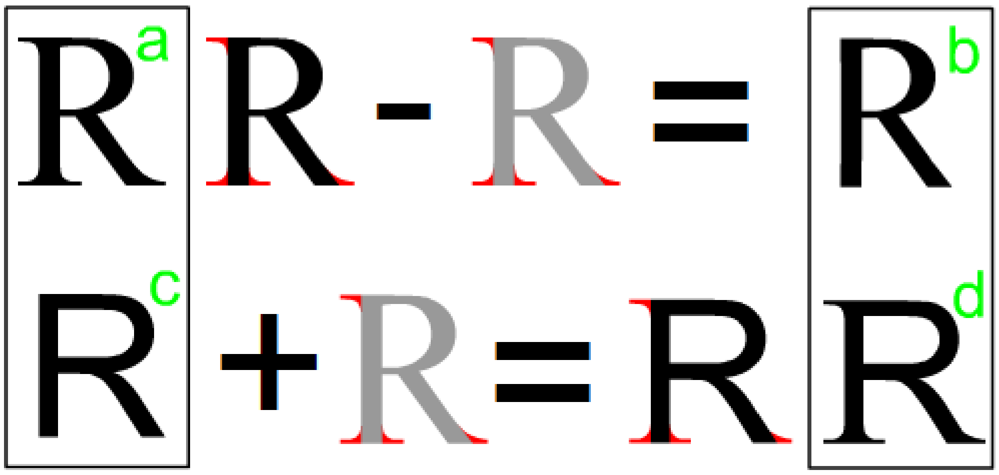


Figure 8-1: Serifs are removed from TNR and attached the Arial font

a) Times New Roman; b) Times New Roman Sans Serif; c) Arial; d) Arial Serif

A sentence reading experiment was designed and implemented involving a boundary paradigm using these pairs of typeface variant. All of them were proportional typefaces in that characters could have different widths. The traditional boundary paradigm non-word masks used for non-proportional font types are therefore no longer applicable. It is very different to find a perfect mask for both typefaces. Therefore, as illustrated in Figure 8-2, we used a new technique which actually stretches or shrinks the non-word mask to make it the same width as the target word. Anti-aliasing is also applied to the fonts to preserve their visual quality.

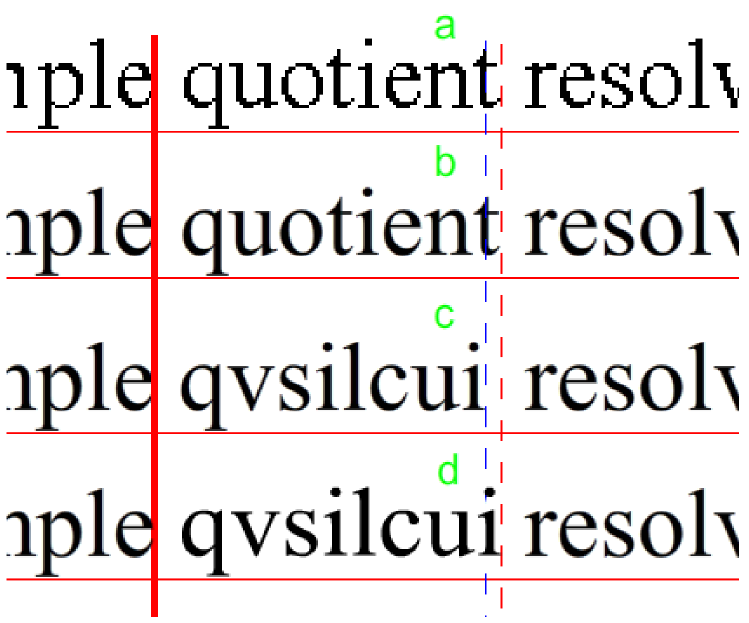


Figure 8-2: A sample sentence

a) Experimental sentence without anti-aliasing; b) Experimental sentence with anti-aliasing; c) Non-word mask is shorter than target word; d) Non-word mask is stretched to the same width as the target word;

However as shown in Figure 8-3, if the non-word mask is stretched or shrunk too much, it may allow participants to be aware of the change. We therefore created a mask generator, which can be found on the web page http://eyemap.tk/maskGen/MaskChecker.html, to generate appropriate masks. After careful selection, all the non-word masks in our experiment have a stretch/shrink rate (compared to their original width) of less than 10%.

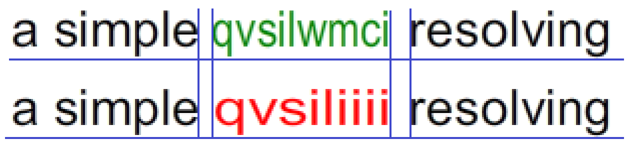


Figure 8-3: Unsuitable non-word mask

Moreover, in order to reduce the legibility of the text and simulate conditions of poor reading conditions, we created a “blur” condition. A Gaussian blur (Shapiro & Stockman, 2001) was applied to each stimulus sentence. This is a widely used transformation in graphics software and its effect is to create a smooth blur resembling that of viewing the image through a translucent screen.

8.1. Participants

Thirty seven subjects from the Florida State University, Tallahassee community participated in this experiment. All had either normal or corrected vision and were native speakers of English.

8.2. Materials & Design

The experimental materials were chosen from (Radach, et al., 2008). One hundred and ninety-two sentences were blocked by typeface, while preview and blur conditions were randomly assigned. This created 16 counterbalanced lists (two typefaces x two serif conditions x two preview conditions x two blur conditions). For half of the experimental sentences the target-word location was occupied by a non-word (mask) that was replaced by the correct word when a pre-target boundary was crossed. Thirty-two questions were prepared, and were distributed evenly among the typeface blocks. Participants were asked to fill in a questionnaire after the experiment. The questionnaire had 10 question and related to the popularity of serif and sans serif fonts in their daily reading, how often they read from computer screen, typeface preferences, their sensitivity to the blur condition etc.

8.3. Apparatus

One hundred and ninety two sentences as well as their corresponding questions were generated in the two Arial and two TNR variants and converted into JPEG pictures. This generated 1536 pictures, which were further processed by Adobe Photoshop CS3 (Adobe Systems Incorp., 2007) to create their blurred version. A Gaussian blur with radius of 1.5 pixels was applied to each picture. These pictures were then presented on a 21-inch Iiyama vision master 510 CRT monitor with a refresh rate of 100 Hz non-interlaced. The spatial resolution of the monitor was 1024 x 768 pixels and the screen subtended 42.7° (40.4 cm) horizontally and 32.0° (30.3 cm) vertically. This resulted in 38.5 pixels per deg at a viewing distance of 86 cm.

Eye movements were recorded using an EyeLink 1000 (SR-Research Ltd., 2011) desktop mounted, video-based eye tracker sampling at 1000Hz. It was calibrated for each subject by instructing them to fixate a series of three dots (0.52° diameter) that were laid out as a horizontal line on the screen. Subjects were seated with their heads stabilised by a chin-rest. They viewed the display binocularly, though only their right eye was calibrated. Stimuli were presented and controlled by experimenter from a display PC and eye movement data were collected using a separate recorder PC.

8.4. Procedure

Prior to the start of the experiment, the experimenter read out a short set of instructions to subjects in which they were told to read some English sentences at their normal rate.

Each trial started with a fixation dot that appeared on the left of the screen next to where the first character of the stimulus sentence would be presented. The subject fixated the dot and pressed a gamepad button to proceed. The fixation dot disappeared and the anti-aliased stimulus sentences would appear at screen position (x = 50, y = 375). Sentences were presented on a white background with a black font colour. Arial and Serif Arial fonts were displayed at 25px (18.75pt), which equates to 13.48 screen pixels per letter on average for lower case letters plus space, while Sans Serif Times New Roman and Times New Roman were displayed at 28px (21pt), which equates to 12.63 pixels screen per letter on average (i.e., around 3 letters per degree).

When a question was presented, participants were asked to answer it orally. The answer was recorded by the experimenter and three horizontal recalibration dots were displayed after each question to prevent possible shifting caused by mouth movements.

8.5. Results

The data were first refined using the eye movement data analysis software, EyeMap (Tang, Reilly, & Vorstius, 2011). Inappropriate trials and fixations were removed, and the word boundaries were automatically extracted as areas of interest. EyeMap then exported a word-based matrix with relevant dependent and independent variables computed. The data matrix was analysed using a linear mixed-effect modelling (Baayen, Davidson, & Bates, 2008; D. Bates & Sarkar, 2007; Kliegl, 2007), which allows the integrated analysis of random effects for both items and subjects along with the standard experimental fixed effects.

The three experimental factors were typeface, serif, and blur. The data analysis was separated into two phases. In the first phase, we investigated three reading time measures on the corpus of words derived from all of the sentences read by the subjects, but excluding words where a display change occurred. The viewing time measures analysed were first fixation duration (FD), single fixation duration (SFD) and gaze duration (GD). In the second analysis phase, we focused on just the target words in order to explore more fully the interaction between preview and the various experimental factors.

8.5.1. Corpus Analysis

Data were filtered according to the following criteria:

1) Fixations on the first and last words in the sentence were excluded;
2) Word lengths between 3 and 10 letters inclusive were selected;
3) Words involving a preview change were excluded;
4) Words in which a blink occurred were excluded;
5) Fixation durations smaller than 80ms or greater than 800ms in duration were excluded.

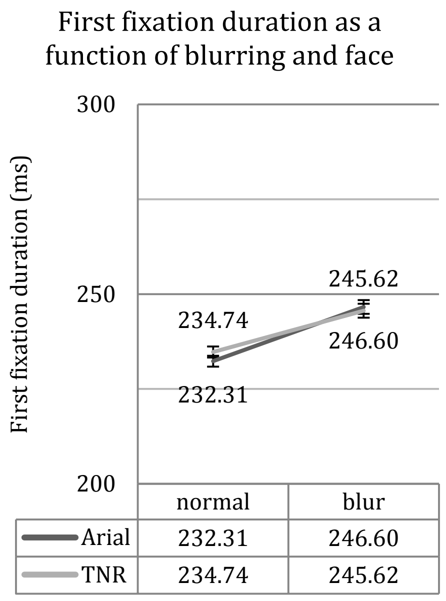
Table 8-1: Analysis of sentence corpus

Model	**term**	**β**	**SE**	**t**
First fixation duration (FFD)	intercept	241.90	3.83	**63.13**
	Log word frequency	-2.64	0.35	**-7.48**
	Word length	3.12	0.53	**5.91**
	Blurred (yes – no)	17.03	0.92	**18.52**
	Typeface (TNR - Arial)	0.57	0.79	0.73
	Serif (yes – no)	-0.47	0.79	-0.59

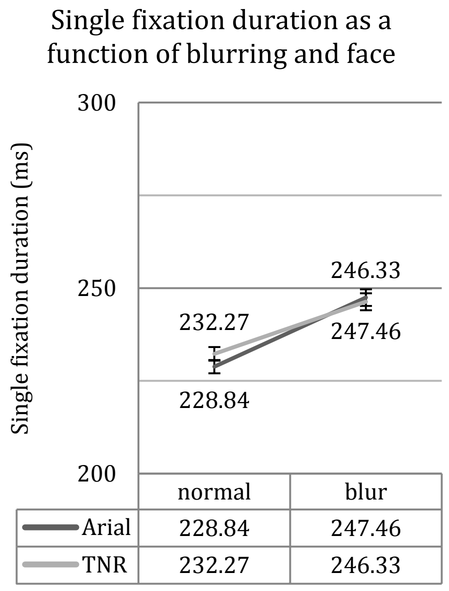
	Blur x Typeface	-4.41	1.58	**-2.79**
	Blur x Word length	0.93	0.41	**2.29**
Single fixation duration (SFD)	intercept	245.41	4.33	**56.73**
	Log word frequency	-3.02	0.44	**-6.84**
	Word length	4.25	0.65	**6.54**
	Blur (yes – no)	19.62	1.17	**16.82**
	Typeface (TNR - Arial)	1.55	0.98	1.58
	Serif (yes – no)	-0.86	1.00	-0.86
	Frequency x Serif	0.61	0.29	**2.14**
	Blur x Typeface	-4.81	1.95	**-2.46**
Gaze duration (GD)	intercept	271.85	6.12	**44.41**
	Log word frequency	-6.43	0.67	**-9.58**
	Word length	11.42	1.00	**11.42**
	Blurred (yes – no)	16.22	1.39	**11.70**
	Typeface (TNR - Arial)	4.79	1.18	**4.07**
	Serif (yes – no)	-1.68	1.19	-1.42
	Frequency x Word length	-1.73	0.31	**-5.56**
	Frequency x Blur	1.11	0.35	**3.16**
	Word length x Typeface	2.67	0.61	**4.39**

In this sentence corpus, the fixation probability is 79.68% and the probability of single fixation for corpus is 48.96%. Table 8-1 is a summary of the three final models used to fit the reading time measures. The models comprise five common terms (log word frequency, word length, blur, typeface, serif), plus any additional significant interaction terms. Word frequency and word length were both included because they have a significant impact on viewing times. Note that the covariates (log word frequency and word length) are centred to make the intercept take on the value of the overall mean of the dependent measure. Random effects in the model were subject and word.

Across all of the dependent measures, word length, word frequency, and blur proved highly significant. Typeface was only significant in the case of gaze duration, where TNR gave rise to longer gaze durations than Arial. However, typeface was involved in significant interactions with the blur condition for FFD and SFD (cf. Figure 8-4) and with word length for GD. The only significant effect involving serif was an interaction with word length for SFD.



a)

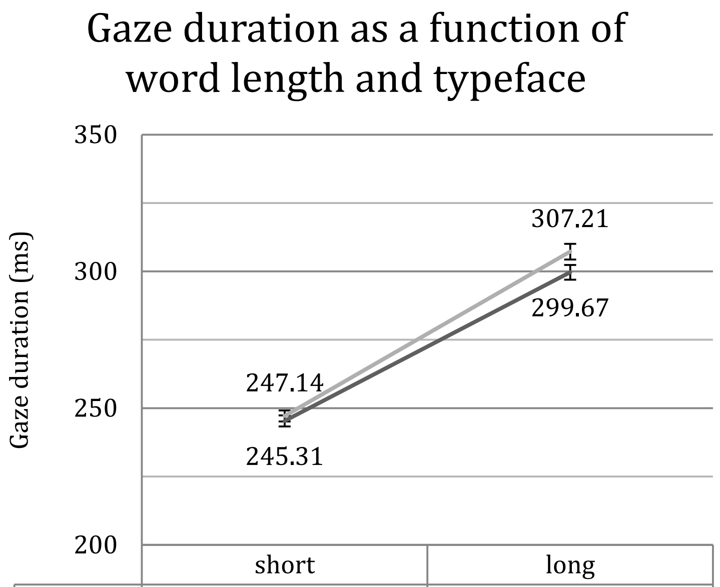


b)

Figure 8-4: Interactions between blur and typeface for (a) first fixation and (b) single fixation durations from the word corpus data set.

(Note that the error bars represent a 95% confidence interval.)

As can be seen from Figure 8-4, the main source of the interaction between blur and typeface is the disappearance, and indeed a hint of a reversal, of the Arial viewing time advantage for blurred compared to non-blurred words.



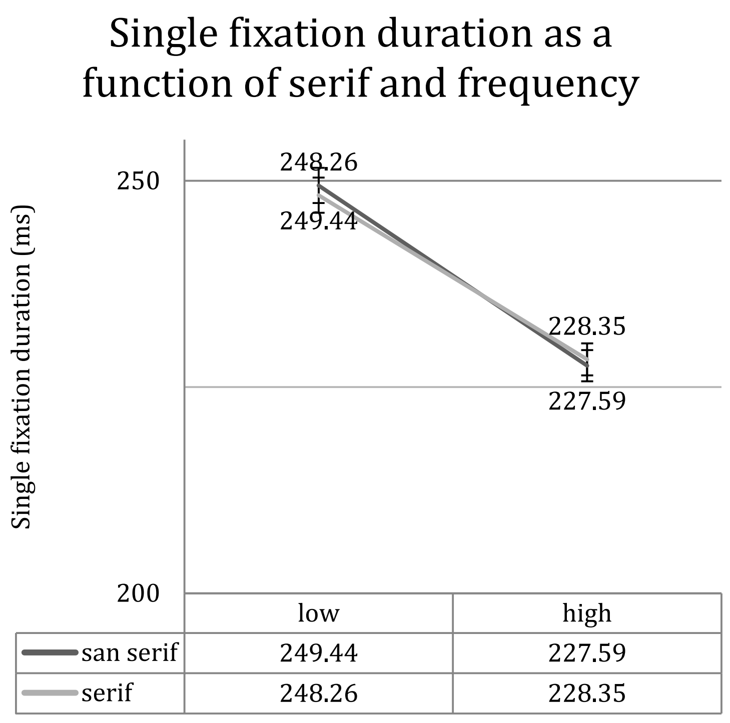
	short	long
Arial	245.31	299.67
TNR	247.14	307.21

Figure 8-5: Interaction between word length and typeface for gaze durations from the word corpus data set.

(Note that the error bars represent a 95% confidence interval.)

In Figure 8-5, we see that the source of the typeface x length interaction is that the Arial advantage in gaze duration only obtains for long words (words > 5.7 chars in length, the average word length in the sentences used).

Finally, Figure 8-6 shows the one serif effect, an interaction between word frequency and serif for single fixation durations.



	low	high
▬ san serif	249.44	227.59
▬ serif	248.26	228.35

Figure 8-6: Interaction between serif and word frequency for single fixation durations from the word corpus data set.

(Note that the error bars represent a 95% confidence interval.)

The main source of the interaction appears to be the small advantage that serifs provide in the processing of low-frequency words.

8.5.2. Target Words Analysis

Table 8-2: Target word analysis

Model	term	β	SE	t
First fixation duration (FFD)	intercept	301.56	5.64	**53.47**
	Log word frequency	-4.87	0.74	**-6.59**
	Word length	4.98	3.66	1.36
	Blur (yes – no)	4.27	3.36	1.27
	Typeface (TNR - Arial)	9.46	2.89	**3.28**
	Serif (yes – no)	-3.35	2.91	-1.15
	Preview (target - mask)	-64.95	2.92	**-22.25**
	Blur x Preview	13.52	5.85	**2.31**
Single fixation duration (SFD)	intercept	328.93	9.03	**36.41**
	Log word frequency	-7.11	1.10	**-6.47**
	Word length	9.90	5.39	1.84
	Blurred (yes – no)	20.37	4.85	**4.20**
	Typeface (TNR - Arial)	12.92	3.82	**3.39**
	Serif (yes – no)	0.43	3.87	0.11
	Preview (target - mask)	-84.47	3.85	**-21.95**
Gaze duration (GD)	intercept	392.59	14.45	**27.18**
	Log word frequency	-19.01	1.65	**-11.52**
	Word length	33.78	8.23	**4.10**
	Blurred (yes – no)	-11.02	5.79	-1.90
	Typeface (TNR - Arial)	11.88	4.92	**2.41**
	Serif (yes – no)	-6.26	4.98	-1.26

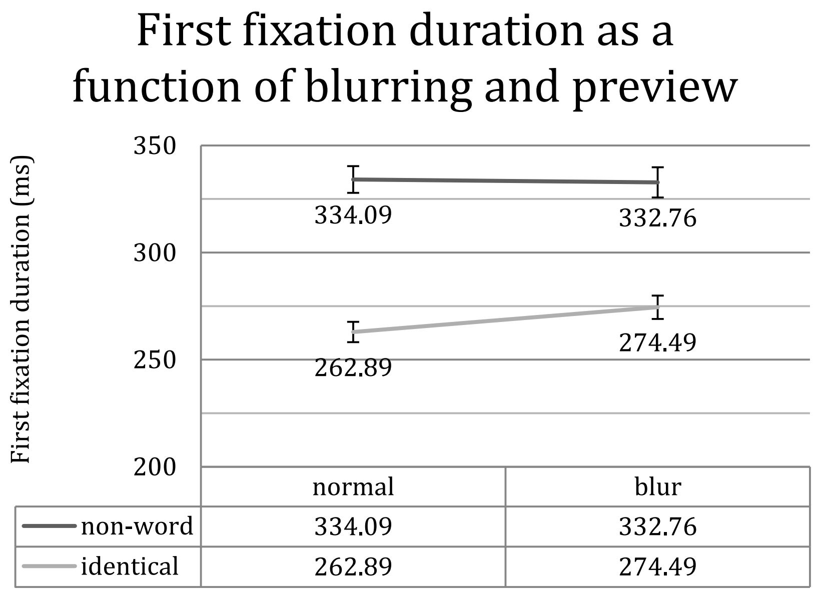
	Preview (target - mask)	-92.74	4.92	**-18.86**
	Frequency x Word length	-7.10	3.00	**-2.37**
	Frequency x Preview	-4.17	1.97	**-2.11**

The target word data was analysed in an approximately similar way (cf. Table 8-2). Compared to the overall word corpus, the target words had word lengths between 7 and 9 characters but were chosen to encompass high and low word frequency. For target words, the fixation probability is 98.24% and the probability of single fixation is 38.36%. Three models were constructed to account for the three viewing time measures of first fixation duration, single fixation duration, and gaze duration. The core terms in the model were identical to those used in the word corpus model with an additional term included for the display change manipulation (preview). As with the covariates in the previous set of models, they were centred so as to permit the interpretation of the intercept as the overall mean of the dependent measure.

The preview term was highly significant for all viewing time measures. More crucially to the study, typeface was a significant effect for all viewing times, with Arial giving rise to shorter viewing times than TNR.

There is no interaction between font type and preview benefit. This may implications that the subtle font features are blurred out in the parafovea. Therefore, font type does not have much effect on the preview.

Word frequency and word length proved significant for all viewing times and in the anticipated direction: high frequency words were fixated for less time as were shorter words. Surprisingly, the blur condition proved to have significantly less impact on viewing times compared to its effect for the corpus as a whole.

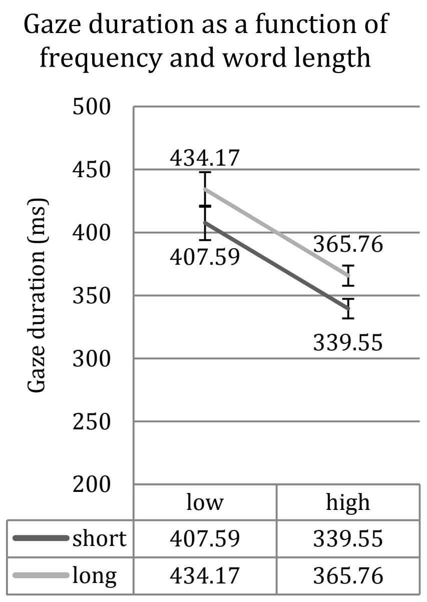


	normal	blur
non-word	334.09	332.76
identical	262.89	274.49

Figure 8-7: Interaction between blur and preview for first fixation durations from the target words data set.

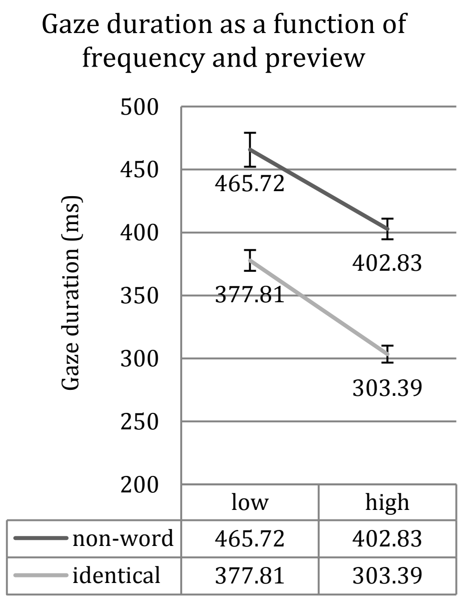
(Note that the error bars represent a 95% confidence interval.)

The main interactions were between blur and preview in the case of first fixation durations and between frequency and preview and frequency and word length in gaze durations. From Figure 8-7, the main source of the interaction appears to be the reduction in preview benefit in the blurred condition.



	low	high
short	407.59	339.55
long	434.17	365.76

a)



	low	high
non-word	465.72	402.83
identical	377.81	303.39

b)

Figure 8-8: Interaction between (a) word frequency and length and (b) word frequency and preview for gaze durations in the target words data set.

(Note that the error bars represent a 95% confidence interval.)

In Figure 8-8, we can see the source of the interaction in (a) is the more reliable difference between long and short high frequency words and in (b) it is the relative increase in preview benefit for high versus low frequency words.

8.5.3. Conclusion

The main results of the study described here are (1) that the Arial typeface, under the right circumstances, affords shorter viewing times than Times New Roman, and (2) serifs play little or no role in Arial's viewing time advantage.

The locus of Arial's advantage appears to be in the consistent thickness of its strokes compared to TNR. In the case of the corpus analysis, there are interactions between the blur condition and typeface, with blurring abolishing Arial's advantage over TNR (see Figure 8-4). This is exactly what one would expect if stroke thickness played a role, since blurring would have the effect of appearing to thicken the thinner stroke regions of TNR.

One paradoxical finding in the results is the significant reduction in the effect of blur on viewing times for the target word data. In the word corpus as a whole, the effect size of blur is consistently of the order of 17ms, but is much more variable for the target words, ranging from a low of 4 ms in FFDs to a high of 20 ms for SFDs. One possible explanation is that the overall viewing time on target words is greater than the corpus average because of the display change manipulation (e.g., 301 ms compared to 245 ms for target and corpus FFDs, respectively). Quite likely, the effect of extended viewing times such as those found in the target data is to reduce the impact of blur.

An important qualification on the finding of a typeface advantage for Arial is that the significant and consistent effects we find among the target words are significantly reduced when we look at the larger corpus of words in the study. A possible explanation is that elevated viewing times on the target words serves to enhance the typeface effect. This is supported by the fact that corpus GDs at around 272 ms are actually shorter than target FFDs at 301 ms. Since the only significant corpus-based typeface effect was found for GDs, it seems reasonable to assume this was because the GDs were close to the values at which a typeface effect was being detected in the target words.

Further circumstantial evidence implicating elevated viewing times in the typeface effect is in the typeface x word length interaction in Figure 8-5. Here we see that the elevated viewing times associated with longer words is also associated with an enhanced advantage for Arial.

Overall, therefore, the main finding is that Arial has a legibility advantage, but only in situations in which reading is slower than average. This might be the case with difficult text or when reading normal text in sub-optimal viewing conditions.

Chapter 10

Conclusions

This dissertation describes an informatics system as well as several reading experiments. As mentioned in the first chapter, the critical contributions of this dissertation are twofold. The first contribution is a software suite (Experiment Executor and EyeMap) designed specifically for the control and analysis of eye tracking experiments in reading. Experiment Executor (ExpExec) is a stimulus presenter which offers a balanced solution for a wide range of reading experiments and is particularly well-suited for exploring low-level vision aspects of reading. Although ExpExec was originally designed for fast display change experiments involved in the co-registration of eye movements and, it turned out to be a general-purpose light-weight presenter that allows rapid experiment preparation. The experiments mentioned in this dissertation were all preformed using ExpExec and the scripts (see attached DVD, folder Chapter 5\Sample Scripts) are all less than 200 lines. It was the first presentation software supporting the “shimmer” paradigm and its scripts can work on several different trackers without modification. Currently ExpExec is being used by other research groups, including the Department of Psychology, Tianjin Normal University and MARCS Auditory Laboratories at the University of Western Sydney.

The other important part of our system is the data visualization tool EyeMap which also uses a new hardware-independent XML data format. Compared to similar products on the market, EyeMap is one of the first eye movement data analyzers which automatically supports word/letter segmentation on proportional font types/unspaced texts, binocular eye movement analysis as well as eye voice span visualization and analysis. It also generates many unique reading-related variables and all supported variables (over 150 variables) can be accessed selectively and immediately after they are included in the analysis. In addition, users can create filters on any supported variables and the filtered results will be reflected in both data outputs and visualizations immediately. EyeMap is now also being used by many reading research groups including the Reading Center at Florida State University, Tallahassee; RWTH Aachen University, Germany; University of Wuppertal, Germany; MARCS Auditory Labs at the University of Western Sydney; the Psychology Department at the University of Sydney; and the Linguistics Program of Northeastern University.

The remaining parts of our system including data agent, multi-threaded data collector, and data exporter were created specifically for collecting data simultaneously from multiple sources and furthermore providing real-time feedback to stimulus presentation software based on these data streams. The reliability of our multi-source data synch algorithm, and feasibility of the system was tested by a preliminary experiment that combined reading with EEG recording.

The second part of the thesis contribution includes several reading experiments preformed using our system. Most of them were designed to explore the low-level vision aspects of reading. There were reading experiments using our shimmer paradigm and a comparison study involving the reading of serif and san-serif fonts under normal and blurred viewing conditions.

The shimmer paradigm involved the reading of texts where the luminance of each word was varied according to a different random value drawn from a Gaussian distribution. We develop a new paradigm using shimmer as a word segmentation tool for research on spaced or unspaced scripts. Two experiments on reading of Chinese and unspaced English under asynchronous and synchronous shimmer were carried out. The experimental results indicated that English readers can actually extract boundary information from asynchronous shimmer and we observed similar results to the alternating**bold** paradigm of Perea and Acha (2009). While for Chinese readers, asynchronous shimmer had no effect on viewing time or first fixation landing sites.

Another experiment we did on this platform dealt with font legibility. It was one of the few eye movement studies on font legibility and probably the first preview study on proportional fonts, where text stretch/shrink technology was used to create appropriate non-word masks. In this study, we focused on some key factors in font design, such as serif, and style of font family (stroke transformation or thickness of strokes). Four proportional font types, Arial, Serif Arial, Times New Roman and San Serif Time New Roman, were investigated under normal and blurred reading condition. The main finding of this study is that Arial has a legibility advantage, but only in situations in which reading is slower than average.

In summary, our system provided a series of tools for reading eye movement research, which can compete with commercial and other academic tools. It incorporates many valuable techniques for performing complex reading experiments as well as visualisation and analysis of collected data. However, as a completed system, it still has many issues that worth to be considered. First, there is no quality assurance or quality control scheme in data collection and later phase. We found that it may quite dangerous to leave the quality control to experimenters, especially for these experiments in which gaze-contingent paradigm or any similar interactive design were employed. We often detected the data irregularity during analyse stage, sometimes, the quality varies even for the same participant. As part of future development of the software, a quality control scheme should be developed to prevent, detect and eliminate data errors and irregularities as early as the data collection phase. Of course, data quality should also be controlled and re-evaluated at the analysis stage.

Another issue of our system is that its application domain is quite limited. Currently, the system is able to perform and analyse many reading experiments. It also provides options to support other trackers, and cooperate with other data sources. However, when we expand the research scope from reading to neighboring eye movement research areas such as scene perception, and visual search, where lots of graphic stimuli presentation tasks are required, the system has limitations. This is because it was specifically designed for reading research, of which the main focus is text. Currently, ExpExec can present graphic stimuli. However, its picture rendering function was mainly implemented as a supplement to text presenting. More flexible operations on graphic stimuli manipulation should be considered. Similarly, in analysis, EyeMap should allow users to create and modify irregular and/or nested areas of interest for analysing these graphic stimuli. Moreover, for those already implemented measurements (variables), in order to make them be used more widely between labs, we are thinking to document the algorithms used for the calculation of each measurement, and setting the standard in the definition of those measurements, or even introduce scripting capabilities to EyeMap in the future.

Although the system was created to accelerate the experiment life cycle, there are still some small efficiency issues that could be improved by us in the near further, such as batch processing tools for project data export and binary to XML data conversion; GUI tools for easy creation and modification of text.html file; auto gaze position drift correction for multi-line texts, and so on.

Reference

Ablinger, I., Huber, W., Schattka, K., & Radach, R. (under revision). Recovery in a letter-by-letter reader: More efficiency at the expense of normal reading strategy.

Adobe Systems Incorp. (2007). Adobe photoshop CS3 (Version CS3): Adobe Systems Incorp. Retrieved from http://www.adobe.com/products/photoshop/photoshop

Arditi, A., & Cho, J. (2005). Serifs and font legibility. *Vision research, 45*(23), 2926-2933.

Baayen, R. H., Davidson, D. J., & Bates, D. M. (2008). Mixed-effects modeling with crossed random effects for subjects and items. *Journal of memory and language, 59*(4), 390-412.

Bai, X., Yan, G., Liversedge, S. P., Zang, C., & Rayner, K. (2008). Reading spaced and unspaced Chinese text: Evidence from eye movements. *Journal of Experimental Psychology: Human Perception and Performance, 34*(5), 1277-1287. doi: 10.1037/0096-1523.34.5.1277

Bates, D., & Sarkar, D. (2007). lme4: Linear mixed-effects models using S4 classes.

Bates, R., Istance, H., & Spakov, O. (2005). D2. 2 Requirements for the Common Format of Eye Movement Data. *Communication by Gaze Interaction (COGAIN), IST-2003-511598: Deliverable, 2*.

Bernard, M., Chaparro, B. S., Mills, M., & Halcomb, C. G. (2003). Comparing the effects of text size and format on the readibility of computer-displayed Times New Roman and Arial text. *International Journal of Human-Computer Studies, 59*(6), 823-835.

Bernard, M., Liao, C. H., & Mills, M. (2001). *The effects of font type and size on the legibility and reading time of online text by older adults*.

Biofeedback Instrument Corporation. (2011). MindSet EEG Systems Retrieved 11/06, 2011, from http://www.biof.com/onlinestore/mindset.asp

BioSemi Instrumentation. (2011). ActiveTwo Retrieved 11/06, 2011, from http://www.biosemi.com/products.htm

Bragg, S. D., & Driskill, C. G. (1994). *Diagrammatic-graphical programming languages and DoD-STD-2167A.* Paper presented at the AUTOTESTCON '94. IEEE Systems Readiness Technology Conference. 'Cost Effective Support Into the Next Century', Conference Proceedings. , Anaheim, CA , USA

Brain Vision LLC. (2011). Brain Vision LLC - products Retrieved 11/06, 2011, from http://www.brainvision.com/products.htm

Brainard, D. H. (1997). The psychophysics toolbox. *Spatial vision, 10*(4), 433-436.

Burt, S. C. L. (1959). *A psychological study of typography*: University Press.

Coghill, V. (1979). Can children read familiar words set in unfamiliar type? *Information Design Journal, 1*(4), 254-260.

Compumedics Neuroscan. (2011). Neuroscan system Retrieved 11/06, 2011, from http://www.neuroscan.com/systems.cfm

Cornelissen, F. W., Peters, E. M., & Palmer, J. (2002). The Eyelink Toolbox: eye tracking with MATLAB and the Psychophysics Toolbox. *Behavior Research Methods, 34*(4), 613-617.

Crane, H. D., & Steele, C. M. (1985). Generation-V dual-Purkinje-image eyetracker. *Applied Optics*, 527-537.

De Lange, R. W., Esterhuizen, H. L., & Beatty, D. (1993). Performance differences between Times and Helvetica in a reading task. *ELECTRONIC PUBLISHING-CHICHESTER-, 6*, 241-241.

Deligianni, F. (2009). Talk2Tobii Toolbox (Version Beta): Centre for Brain and Cognitive Development. Retrieved from http://www.cbcd.bbk.ac.uk/people/affiliated/fani/talk2tobiiFiles/talk2tobii09_LP.mexmaci

Dillon, A. (1992). Reading from paper versus screens: A critical review of the empirical literature. *Ergonomics, 35*(10), 1297-1326.

Duchowski, A. T. (2007). *Eye tracking methodology: Theory and practice*: Springer-Verlag New York Inc.

Eaton, J. W. (2012). Octave: GNU Octave. Retrieved from http://www.gnu.org/software/octave/

Eclipse, I. (2010). Eclipse Foundation.

Engbert, R., Nuthmann, A., Richter, E. M., & Kliegl, R. (2005). SWIFT: a dynamical model of saccade generation during reading. *Psychological Review, 112*(4), 777.

Entroware Communication. (2009). EDAS II (Version 2): Entroware Communication,. Retrieved from www.entroware.com

Fourward Technologies, I. (2011). DPI Eyetracker Gen 6 Retrieved 11/06, 2011, from http://www.fourward.com/

Gitelman, D. R. (2002). ILAB: A prog ram for postexperimental eye movement analysis. *Behavior Research Methods, Instruments, & Computers, 34*(4), 605.

Glenstrup, A. J., & Engell-Nielsen, T. (1995). Eye controlled media: Present and future state.

Halverson, T., & Hornof, A. (2002). VizFix Software Requirements Specifications. *Computer and Information Science, University of Oregon. Available at< h ttp://www. cs. uoregon. edu/research/cm-hci/VizFix*.

Hill, A., & Scharff, L. (1997). *Readability of websites with various foreground/background color combinations, font types and word styles*.

Huey, E. B. (1901). On the psychology and physiology of reading. II. *The American Journal of Psychology*, 292-312.

Huey, E. B. (1908). *The psychology and pedagogy of reading*: The Macmillan Company.

Ihaka, R., & Gentleman, R. (1996). R: A language for data analysis and graphics. *Journal of* *computational and graphical statistics, 5*(3), 299-314.

Inhoff, A. W., Starr, M., Liu, W., & Wang, J. (1998). Eye-movement-contingent display changes are not compromised by flicker and phosphor persistence. *Psychonomic bulletin & review, 5*(1), 101-106.

Interactive minds GmbH. (2010). NYAN 2.0 TX Eye Tracking Data Analysis Suite (Version 2.0 XT): Interactive minds GmbH,. Retrieved from http://www.interactive-minds.com/en/eye-tracking-software/nyan-2

Javal, E. (1879). *Essai sur la physiologie de la lecture*.

Joblove, G. H., & Greenberg, D. (1978). Color spaces for computer graphics. *ACM SIGGRAPH Computer Graphics, 12*(3), 20-25.

Kaushal, N., & Kaur, R. Comparative Analysis of Various Automated Test Tools for Flex Application.

Kennedy, A. (1996). Eye movement control during the inspection of words under conditions of pulsating illumination. *Journal of Cognitive Psychology, 8*(4), 381-404.

Kennedy, A., Brysbaert, M., & Murray, W. S. (1998). The effects of intermittent illumination on a visual inspection task. *The Quarterly Journal of Experimental Psychology Section A, 51*(1), 135-151.

Kennedy, A., & Pynte, J. (2005). Parafoveal-on-foveal effects in normal reading. *Vision research, 45*(2), 153-168.

Kernighan, B. W., Ritchie, D. M., & Ejeklint, P. (1988). *The C programming language* (Vol. 78): Citeseer.

Kingery, D., & Furuta, R. (1997). Skimming electronic newspaper headlines: A study of typeface, point size, screen resolution, and monitor size. *Information processing & management, 33*(5), 685-696.

Kliegl, R. (2007). Linear mixed-effects distributed processing models for reading fixations. *Journal of Experimental Psychology: General, Online supplement*, 1-22.

Kliegl, R., Nuthmann, A., & Engbert, R. (2006). Tracking the mind during reading: The influence of past, present, and future words on fixation durations. *Journal of Experimental Psychology: General, 135*(1), 12.

Komogortsev, O. V., Jayarathna, S., Koh, D. H., & Gowda, S. M. (2010). *Qualitative and quantitative scoring and evaluation of the eye movement classification algorithms*.

Komogortsev, O. V., & Khan, J. I. (2007). *Kalman filtering in the design of eye-gaze-guided computer interfaces*.

Lalor, E. C., Kelly, S. P., Pearlmutter, B. A., Reilly, R. B., & Foxe, J. J. (2007). Isolating endogenous visuo-spatial attentional effects using the novel visual-evoked spread spectrum analysis (VESPA) technique. *European Journal of Neuroscience, 26*(12), 3536-3542.

Lalor, E. C., Pearlmutter, B. A., Reilly, R. B., McDarby, G., & Foxe, J. J. (2006). The VESPA: a method for the rapid estimation of a visual evoked potential. *Neuroimage, 32*(4), 1549-1561.

Lankford, C. (2000). *Gazetracker: software designed to facilitate eye movement analysis*.

Larsson, G. (2010). *Evaluation methodology of eye movement classificatio algorithms.* Master of Science, Skolan för datavetenskap och kommunikation, Kungliga Tekniska höskolan, Stockholm, Sweden.

Levine, J. R., Mason, T., & Brown, D. (1992). *Lex & yacc*: O'Reilly Media.

Logic., G. (2011). FlexMonkey (Version FlexMonkey 5 Beta): Gorilla Logic. Retrieved from http://www.gorillalogic.com/flexmonkey

MathWorks, M. U. G. (1989). MathWorks Inc. *South Natick MA*.

McLean, R. (1980). *The Thames and Hudson manual of typography*: Thames and Hudson.

McMains, S. A., & Somers, D. C. (2005). Processing efficiency of divided spatial attention mechanisms in human visual cortex. *The Journal of neuroscience, 25*(41), 9444.

Moriarty, S. E., & Scheiner, E. C. (1984). A study of close-set text type. *Journal of applied psychology, 69*(4), 700.

Morris, R. A., Aquilante, K., Yager, D., & Bigelow, C. (2002). *Serifs slow RSVP reading at very small sizes, but don't matter at larger sizes*.

Neurobehavioral Systems, I. (2012). Presentation. San Francisco, CA, USA: Neurobehavioral Systems. Retrieved from http://www.neurobs.com

O'Regan, J. K. (1990). Eye movements and reading. *Reviews of oculomotor research, 4*, 395.

Olsson, P. (2007). *Real-time and offline filters for eye tracking.* Master, Royal Institute of Technology.

Peirce, J. W. (2007). PsychoPy--Psychophysics software in Python. *Journal of neuroscience methods, 162*(1-2), 8-13.

Peirce, J. W. (2008). Generating stimuli for neuroscience using PsychoPy. *Frontiers in neuroinformatics, 2*.

Perea, M., & Acha, J. (2009). Space information is important for reading. *Vision research, 49*(15), 1994-2000.

Posner, M. I. (1980). Orienting of attention. *The Quarterly Journal of Experimental Psychology, 32*(1), 3-25.

Psychology Software Tools, I. (2011a). E-Prime application suite for psychology experiment design, implementation, and analysis   Retrieved 11/06, 2011, from http://www.pstnet.com/eprime.cfm

Psychology Software Tools, I. (2011b). E-Prime® Extensions for Tobii.

Radach, R., Heller, D., & Hofmeister, J. (1997). New evidence on the “locus” of screen-pulsation effects.   . *Paper to Fourth Scientific Meeting, BIOMED Project “Assessment and Remediation of Adverse Effects of Visual Display Units (VDUs) in the workplace”, Nice*.

Radach, R., Huestegge, L., & Reilly, R. (2008). The role of global top-down factors in local eye-movement control in reading. *Psychological research, 72*(6), 675-688.

Rayner, K. (1975). The perceptual span and peripheral cues in reading. *Cognitive Psychology*.

Rayner, K. (1978). Eye movements in reading and information processing. *Psychological bulletin, 85*(3), 618.

Rayner, K. (1998). Eye movements in reading and information processing: 20 years of research. *Psychol Bull, 124*(9849112), 372-422.

Rayner, K., Reichle, E. D., & Pollatsek, A. (1998). Eye movement control in reading: An overview and model. *Eye guidance in reading and scene perception*, 243¨C268.

Reichle, E. D., Rayner, K., & Pollatsek, A. (2003). The EZ Reader model of eye-movement control in reading: Comparisons to other models. *Behavioral and Brain Sciences, 26*(04), 445-476.

Reilly, R. G., & O'Regan, J. K. (1998). Eye movement control during reading: A simulation of some word-targeting strategies. *Vision research, 38*(2), 303-317.

Reilly, R. G., & Radach, R. (2003). Foundations of an interactive activation model of eye movement control in reading. *The mind's eye: Cognitive and applied aspects of eye movement research*, 429¨C455.

Reilly, R. G., & Radach, R. (2006). Some empirical tests of an interactive activation model of eye movement control in reading. *Cognitive Systems Research, 7*(1), 34-55.

Rogerson, D. (1997). *Inside COM: Microsoft's Component Object Model*: Microsoft Press.

Rubinstein, R. (1988). *Digital typography: an introduction to type and composition for computer system design*: Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

Russell-Minda, E., Jutai, J. W., Strong, J. G., Campbell, K. A., Gold, D., Pretty, L., & Wilmot, L. (2007). The legibility of typefaces for readers with low vision: A research review. *Journal of Visual Impairment and Blindness, 101*(7), 402.

Salvucci, D. D., & Goldberg, J. H. (2000). *Identifying fixations and saccades in eye-tracking protocols*.

Sauter, D., Martin, B., Di Renzo, N., & Vomscheid, C. (1991). Analysis of eye tracking movements using innovations generated by a Kalman filter. *Medical and biological Engineering and Computing, 29*(1), 63-69.

Schalk, G., McFarland, D. J., Hinterberger, T., Birbaumer, N., & Wolpaw, J. R. (2004). BCI2000: a general-purpose brain-computer interface (BCI) system. *Biomedical Engineering, IEEE Transactions on, 51*(6), 1034-1043.

SensoMotoric Instruments GmbH. (2011a). Gaze and eye tracking systems products overview Retrieved 11/06, 2011, from http://www.smivision.com/en/gaze-and-eye-tracking-systems/products/overview.html

SensoMotoric Instruments GmbH. (2011b). SMI Experiment Center™ Retrieved 11/06, 2011, from http://www.smivision.com/en/gaze-and-eye-tracking-systems/products/experiment-center-software.html

Shapiro, L. G., & Stockman, G. C. (2001). Computer Vision. 2001 (pp. 137, 150): Prentice Hall.

Sheedy, J. E., Subbaram, M. V., Zimmerman, A. B., & Hayes, J. R. (2005). Text legibility and the letter superiority effect. *Human Factors: The Journal of the Human Factors and Ergonomics Society, 47*(4), 797.

Slattery, T. J., & Rayner, K. (2010). The influence of text legibility on eye movements during reading. *Applied Cognitive Psychology, 24*(8), 1129-1148. doi: 10.1002/acp.1623

SR-Research Ltd. (2011). Products, Solutions, and Publications Retrieved 13/06, 2011, from http://www.sr-research.com

SR Research Ltd. (2011a). Experiment Builder (Version 1.10.1): SR Research Ltd.,. Retrieved from http://download.sr-support.com/ebrelease/sreb1.10.1.zip

SR Research Ltd. (2011b). EyeLink 1000 Core System.

SR Research Ltd. (2011c). SR Data Viewer (Version 1.10.123): SR Research Ltd.,. Retrieved from http://download.sr-support.com/dvrelease/EyeLinkDV_1.10.123.zip

SR Research Ltd. (2012). PyLink (Version 2.6): SR Research Ltd.,. Retrieved from https://www.sr-support.com/forums/showthread.php?t=14

Stellmach, S., Nacke, L. E., Dachselt, R., & Lindley, C. A. (2010). Trends and Techniques in Visual Gaze Analysis. *Arxiv preprint arXiv:1004.0258*.

Straw, A. D. (2008). Vision egg: an open-source library for realtime visual stimulus generation. *Frontiers in neuroinformatics, 2*.

Tang, S., Reilly, R. G., & Vorstius, C. (2011). EyeMap: A software system for presenting and analyzing eye-movement data in reading. *Behavior Research Methods*.

Thomas RECORDING GmbH. (2011). Chromos Vision Retrieved 11/06, 2011, from http://www.thomasrecording.de/

Tobii Technology. (2004). ClearView (Version 2.5.1): Tobii Technology,.

Tobii Technology. (2008). Tobii Studio (Version 2.0.5): Tobii Technology AB. Retrieved from http://www.tobii.com/en/analysis-and-research/global/products/software/tobii-studio-analysis-software/

Tobii Technology. (2011). Psycholinguistics and reading Eye tracking Retrieved 11/06, 2011, from http://www.tobii.com/en/analysis-and-research/global/research/linguistics/

Tullis, T. S., Boynton, J. L., & Hersh, H. (1995). *Readability of fonts in the windows environment*.

TypeBased. (2011). FlexUnit (Version FlexUnit 4.1 Beta 2): TypeBased. Retrieved from http://flexunit.org/

van Rossum, G. (2012). Python language website, from http://www.python.org

Vicente-Grabovetsky, A. (2010, 2010-07-01 12:38:18). Eye Tracking with Eprime Retrieved 27/07, 2011, from http://imaging.mrc-cbu.cam.ac.uk/meg/EyeTrackingWithEprime

Wills, G. J. (1996). *Selection: 524,288 ways to say*.

Wooding, D. S. (2002). *Fixation maps: quantifying eye-movement traces*.

Yan, M., Kliegl, R., Richter, E. M., Nuthmann, A., & Shu, H. (2010). Flexible saccade-target selection in Chinese reading. *The Quarterly Journal of Experimental Psychology, 63*(4), 705-725.

Zachrisson, B. (1965). *Studies in the legibility of printed text*: Almqvist & Wiksell.

Appendix A. Function Comparison of EyeMap and SR DataViewer

Table A-1: Function comparison of EyeMap and SR DataViewer

Name	**EyeMap**	**SR DataViewer**
Most recent version	0.3.6	1.11.1
Last update	June 3, 2011	March 4, 2011
	Visualization	
Fixation	Semi-transparent dot with shadow. Red indicates right eye and Blue indicates left eye. Number of fixation displays on the top-left side of the dot.	Colored circle. For binocular data, only one eye can be visualized. Fixation duration displays on the top-left side of the circle
Fixation Duration	dot radius/color intensity	circle radius
Raw Information of fixation	on mouse over the fixation dot	click fixation circle
Fixation Information	on click, calculate selected fixation variables (140+)	**N.A.**
Fixation Heatmap	**N.A.**	Yes
AoI Information	On click, calculate selected word variables (60+). If audio is available play the audio of AoI on click.	AoI dimensions/Fixation Count/Dwell time
AoI Heatmap	Yes	**N.A.**
Saccade	Gradated coloured lines with a small circle indicating the start position	Numbered lines with arrow indicating its direction
Pupil Size	Semi-transparent dot with shadow	**N.A.**
Message	Yellow circle	Green box

Blink	Semi-transparent grey dot	Red line
Drift Correct position	Green circle in saccade viewer	**N.A.**
Raw Data	X/Y/Pupil Size vs. time	**N.A.**
2D Temporal Graph	Fixation/AoI vs. time	Fixation/Saccade vs. time
Chart Save	Yes	Yes
	Playback control	
Animation	Semi-transparent fixation dots growing by time with a 'tail' to show the most recent fixations.	Moving dots
Playback with voice audio	Red progress bar filling the word when audio is play-ing.	**N.A.**
Eye voice span	Yes	**N.A.**
Play/Pause/Stop/Forward	Yes	Yes
Control playback speed	Yes	Yes
Export animation to video	FLV format	AVI/MOV format
	Editing	
Drift Correct Fixations	Yes	Yes
Fixation Merge/Split/Delete	**N.A.**	Yes
Auto Vertical Align Fixa-tion	Reset all fixation Y posi-tion to drift correction po-sition Y	**N.A**
	Data Analysis	
AoI Creation	Find word begin/end/center without pronunciation based on font metrics, displayed in semi-transparent red box.	Automatically find (based on image pro-cessing, not reliable) or user defined. Displayed in yellow rectangle. User can create irregular AoI by mouse drag or

		click.
Data Filters	Any number of data filters on any supported variables	Duration filter for fixation/ Distance filter for saccade
Binocular Analysis	Synch left right eye automatically, output data sheet with both eyes.	**N.A**
Variables on Fixation	43 variable types on Fixation level 9 variable types on Gaze level	40+ variable types, user can access current, next, and previous fixation form fixation/saccade report.
Variables on Saccade	37 variable types on Saccade level	40+ variable types, user can access current, next, and previous saccade form fixation/saccade report.
Variables on AoI	21+ variable types on Word level. Get the properties of fixation and gaze on word (AoI) through variable "GxFy"(return all variables of yth fixation of xth gaze on the word in fixation list)	8 variable types on 1st, 2nd, 3rd and last Fixation of the AoI; 5 variable types on 1st, 2nd, 3rd and last Run of the AoI.
Variables on Trial	5 variable types on Experiment level; 5 variable types on Trial level	40+ variable types
Variables on Message	3 variable types	30+ variable types
Variables on Eye Voice	24 variable types	**N.A**
Landing Site Analysis	22 variables types for proportional and monospaced fonts	**N.A**

Appendix B. EyeMap Variables

Table B-1: EyeMap variable list

No.	Variable Name	Level	Description
1	EXP	Expt	Exptal name abbreviation, taken from 3rd-5th digit of filename.
2	List	Expt	optional, taken from 6th digit of filename.
3	Subject	Expt	Subject code, taken from first two digits of filename.
4	Var1	Expt	optional, taken from 7th digit of filename.
5	Var2	Expt	optional, taken from 8th digit of filename.
6	LineCount_T	Trial	Total number of lines for the Trial.
7	SentenceCount_T	Trial	Total number of sentences for the Trial.
8	TrialNum	Trial	Trial Number.
9	TrialProperty	Trial	Properties defined in the trial.csv file.
10	WordCount_T	Trial	Total number of words for the Trial.
11	Fixated	Word	1 if the word was fixated, 0 if not.
12	FixCount_W	Word	Total number of fixations on the word.
13	GazeCount_W	Word	Total number of Gazes (passes) on the word.
14	LineNum_T	Word	Line number in the current Trial.
15	BlinkCount_W	Word	Number of blinks on the word.
16	SentenceNum_T	Word	Sentence number in the current Trial.
17	TVDur_sac	Word	Total Viewing Time (TVD), or the sum of all fixations on the word + Saccades.
18	TVDur	Word	Total Viewing Time (TVD), or the sum of all fixations on the word.
19	Word	Word	Word, without punctuation.
20	WordBlinkDur	Word	Duration of the blinks in the word.
21	WordCount_L	Word	Total number of words on the Line.
22	WordCount_S	Word	Total number of words in the Sentence.
23	WordLen_punct	Word	Word Length, in letter, including punctuation.
24	WordLen	Word	Word length, in letters.
25	WordLocX	Word	x-pixel location of the word (upper left corner).
26	WordLocY	Word	y-pixel location of the word (upper left corner).

27	WordNum_E	Word	Word number for the Expt.
28	WordNum_L	Word	Word number on the Line.
29	WordNum_S	Word	Word number in the Sentence.
30	WordNum_T	Word	Word number for the Trial.
31	WordProperty	Word	Properties defined in the word.csv file.
32	FixCount_G	Gaze	Total number of fixation for the current Gaze (Pass).
33	GazeBlinkDur	Gaze	Duration of the blinks in the gaze.
34	GazeDur_sac	Gaze	Gaze duration (GD) of the current Gaze (Pass), including internal saccades, plus outgoing.
35	GazeDur	Gaze	Gaze duration (GD) of the current Gaze (Pass).
36	GazeNum_W	Gaze	Number of the current Gaze (Pass) on the word.
37	GazePupil	Gaze	Mean Pupil diameter for the entire gaze.
38	BlinkCount_G	Gaze	Number of blinks on the gaze.
39	Refixated	Gaze	1, if FixNum_G
40	RefixCount	Gaze	Total number of refixations on the word in the current gaze
41	Blink	Fixation	If, -1, then blink before, if, 1, then blink after the current fixation.
42	BlinkDur	Fixation	Duration of the Blink.
43	FixDur	Fixation	Duration of the current Fixation.
44	FixLocX	Fixation	x-pixel Average fixation location of the current fixation.
45	FixLocXBeg	Fixation	x-pixel location of the current fixation at the beginning of that fixation.
46	FixLocXEnd	Fixation	x-pixel location of the current fixation at the end of that fixation.
47	FixLocY	Fixation	y-pixel Average fixation location of the current fixation.
48	FixLocYBeg	Fixation	y-pixel location of the current fixation at the beginning of that fixation.
49	FixLocYEnd	Fixation	y-pixel location of the current fixation at the end of that fixation.
50	FixNum_E	Fixation	Fixation number, in the Expt.
51	FixNum_G	Fixation	Number of the current fixation in the current Gaze (Pass).

52	FixNum_S	Fixation	Fixation number, in the Sentence.
53	FixNum_T	Fixation	Fixation number, for the Trial.
54	FixNum_W	Fixation	Number of the current fixation on the words.
55	FixPupil	Fixation	Mean Pupil diameter for the Fixation.
56	FixStartTime	Fixation	Time stamp of the start of the current Fixation (ms).
57	LandPos_NP	Fixation	(monospaced) Landing position in the word, described in letter units (space before word is 0).
58	LandPos	Fixation	Landing position in the word, described in letter units (space before word is 0).
59	LandPosDec_NP	Fixation	(monospaced) Landing position in the word, described in letter units including decimal places (space before word is 0).
60	LandPosDec	Fixation	Landing position in the word, described in letter units including decimal places.
61	LaunchDistBeg_NP	Fixation	(monospaced) Launch distance from the beginning of the current word.
62	LaunchDistBeg	Fixation	Launch distance from the beginning of the current word starting at letter zero.
63	LaunchDistBegDec_NP	Fixation	(monospaced) Launch distance from the beginning of the current word including decimal places.
64	LaunchDistBegDec	Fixation	Launch distance from the beginning of the current word including decimal places.
65	LaunchDistCent_NP	Fixation	(monospaced)Launch distance from the center of the current word.
66	LaunchDistCent	Fixation	Launch distance from the center of the current word.
67	LaunchDistCentDec_NP	Fixation	(monospaced) Launch distance from the center of the current word including decimal places.
68	LaunchDistCentDec	Fixation	Launch distance from the center of the current word including decimal places.
69	LaunchDistEnd_NP	Fixation	(monospaced) Launch distance from the end of the current word.
70	LaunchDistEnd	Fixation	Launch distance from the end of the current word.
71	LaunchDistEndDec_NP	Fixation	(monospaced) Launch distance from the end of the current word including decimal places.

72	LaunchDistEndDec	Fixation	Launch distance from the end of the current word including decimal places.
73	LineSwitch	Fixation	If the current fixation is on a different line than the previous Fixation.
74	NxtFixDur	Fixation	Duration of the next Fixation.
75	NxtLandPos_NP	Fixation	(monospaced) Landing position of the next fixation in characters.
76	NxtLandPosDec_NP	Fixation	(monospaced) Landing position of the next fixation including decimal places.
77	NxtWordFix	Fixation	Outgoing Saccade Amplitude in Word Units, Zero if last fixation with the same word.
78	PreFixDur	Fixation	Duration of the previous Fixation.
79	PreLandPos_NP	Fixation	(monospaced) Landing position of the previous Fixa-tion in characters.
80	PreLandPosDec_NP	Fixation	(monospaced) Landing position of the previous Fixa-tion including decimal places.
81	PreWordFix	Fixation	Incoming Saccade Amplitude in Word Units, Zero if last fixation with the same word.
82	Refixation	Fixation	If 1, then multiple fixations on the word, if 0, only 1 or none).
83	RepairTime	Fixation	Total re-reading time on word.
84	ReReading	Fixation	1 if there was a prior fixation on the line to the right of the current word, else 0.
85	G1F1	Fixation	The first fixation in the 1st gaze on the word.
86	G1Fn	Fixation	The last fixation in the 1st gaze on the word.
87	GnFn	Fixation	The last fixation in the last gaze on the word.
88	SacInAmp	Saccade	Amplitude of the incoming saccade, in letters.
89	SacInAmpX	Saccade	Amplitude of the incoming saccade, x-axis.
90	SacInAmpY	Saccade	Amplitude of the incoming saccade, y-axis.
91	SacInDur	Saccade	Duration of the incoming Saccade.
92	SacInInter	Saccade	If 0, then intra-word saccade. If 1, then inter-word saccade.
93	SacInLocBegX	Saccade	Beginning location of the incoming Saccade, x-axis.
94	SacInLocBegY	Saccade	Beginning location of the incoming Saccade, y-axis.
95	SacInLocEndX	Saccade	End location of the outgoing Saccade, x-axis.

96	SacInLocEndY	Saccade	End location of the outgoing Saccade, y-axis.
97	SacInNWonP	Saccade	Target Number of Word on Page of incoming saccade
98	SacInProg	Saccade	If 1, then incoming saccade is progressive, if 0, then regressive.
99	SacInStartTime	Saccade	Time stamp of the start of the incoming Saccade (ms).
100	SacInVel	Saccade	Mean incoming saccade velocity, x-coordinate
101	SacInVel_max	Saccade	Maximum incoming saccade velocity, x-coordinate
102	SacInVelX	Saccade	Average incoming saccade velocity, x-axis.
103	SacInVelY	Saccade	Average incoming saccade velocity, y-axis.
104	SacInWord	Saccade	Word from which the current saccade has exited.
105	SacInXY	Saccade	Saccade Amplitude of the incoming saccade, in Eu-clidian units.
106	SacOutAmp	Saccade	Amplitude of outgoing saccade
107	SacOutAmpX	Saccade	Amplitude of the outgoing saccade, x-plane.
108	SacOutAmpY	Saccade	Amplitude of the outgoing saccade, y-plane.
109	SacOutDur	Saccade	Duration of the outgoing Saccade.
110	SacOutInter	Saccade	If 0, then intra-word saccade. If 1, then inter-word saccade.
111	SacOutLocBegX	Saccade	Beginning location of the outgoing Saccade, x-axis.
112	SacOutLocBegY	Saccade	Beginning location of the outgoing Saccade, y- axis.
113	SacOutLocEndX	Saccade	End location of the outgoing Saccade, x- axis.
114	SacOutLocEndY	Saccade	End location of the outgoing Saccade, y- axis.
115	SacOutNWonP	Saccade	Target Number of Word on Page of outgoing saccade
116	SacOutProg	Saccade	If 1, then outgoing saccade is progressive, if 0, then regressive.
117	SacOutStartTime	Saccade	Time stamp of the start of the outgoing Saccade (ms).
118	SacOutVel	Saccade	Mean outgoing saccade velocity, x-coordinate
119	SacOutVel_max	Saccade	Maximal outgoing saccade velocity, x-coordinate
120	SacOutVelX	Saccade	Mean outgoing saccade velocity, x-coordinate
121	SacOutVelY	Saccade	Mean outgoing saccade velocity, x-coordinate
122	SacOutWord	Saccade	Word to which the current saccade is directed.
123	SacOutXY	Saccade	Saccade Amplitude of the outgoing saccade, in Eu-clidian units.

124	ArticDur	Voice	Articulation duration of the current word.
125	VLauoff10	Voice	Distance from voice offset fixation to the first character of the word that is being spoken.
126	VLauon10	Voice	Distance from voice onset fixation to the first character of the word that is being spoken.
127	VoffFix	Voice	Voice offset fixation.
128	VoiceOffPos_NP	Voice	(monospaced) Letter position at time of the current word's voice offset.
129	VoiceOffPosDec_NP	Voice	(monospaced) Letter position at time of the current word's voice offset, including decimal places.
130	VoiceOnPos_NP	Voice	(monospaced) Letter position at time of the current word's voice onset.
131	VoiceOnPosDec	Voice	Letter position at time of the current word's voice onset, including decimal places.
132	VoiceOnPosDec_NP	Voice	(monospaced) Letter position at time of the current word's voice onset, including decimal places.
133	VoiceOffPos	Voice	Letter position at time of the current word's voice offset.
134	VoiceOffPosDec	Voice	Letter position at time of the current word's voice offset, including decimal places.
135	VoiceOnPos	Voice	Letter position of the current word's voice onset.
136	VoiceOffTrialTime	Voice	Trial based time of the current word's voice offset (ms).
137	VoiceOnTrialTime	Voice	Trial based time of the current word's voice onset (ms).
138	VoiceOffWord	Voice	Fixated word at the time of the current word's voiced offset.
139	VoiceOnWord	Voice	Fixated word at the time of the current word's voiced onset.
140	VonFix	Voice	Voice onset fixation.
141	FalseStartFix	Voice	False start fixation
142	FalseStartTrialTime	Voice	Trial based time of the current word's voice false start (ms).
143	FalseStartWord	Voice	Fixated word at the time of the current word's voiced false start.

144	FalseStartPos_NP	Voice	(monospaced) Letter position of the current word's voice false start.
145	FalseStartPos	Voice	Letter position of the current word's voice false start.
146	FalseStartPosDec_NP	Voice	(monospaced) Letter position at time of the current word's voice false start, including decimal places.
147	FalseStartPosDec	Voice	Letter position of the current word's voice false start, including decimal places.
148	MSGxInc	Message	Increment number of a Message.
149	MSGxName	Message	Message Name.
150	MSGxStartTime	Message	Time Stamp (EDF time) of the Message.

Appendix C. Comparison of Variable Values from SR DataViewer and EyeMap

The table below displays a sample of typical reading variables for a randomly selected subject broken down by word length. See Appendix A for a definition of the variables

Table C-1: Comparison of variable values from SR DataViewer and EyeMap

R_WordLen		TVDur		FixDurG1F1		GazeDurG1F1		FixCount_GG1F1		SacInAmp_G1F1	
		EM	DV	EM	DV	EM	DV	EM	DV	EM	DV
1	Mean	169.8750	169.88	169.88	169.88	169.88	169.88	1.00	1.00	2.1800	2.1800
	N	8	8	8	8	8	8	8	8	8	8
	Std. Deviation	55.43964	55.440	55.440	55.440	55.440	55.440	.000	.000	.60444	.60444
	Minimum	92.00	92	92	92	92	92	1	1	1.15	1.15
	Maximum	274.00	274	274	274	274	274	1	1	2.95	2.95
2	Mean	233.8624	233.86	175.16	175.16	182.79	182.79	1.05	1.05	2.2926	2.2926
	N	109	109	109	109	109	109	109	109	84	84
	Std. Deviation	123.80574	123.806	51.712	51.712	59.337	59.337	.210	.210	2.52116	2.52116
	Minimum	55.00	55	55	55	55	55	1	1	.02	.02
	Maximum	779.00	779	348	348	396	396	2	2	20.11	20.11
3	Mean	246.8095	246.89	177.77	178.04	183.12	183.38	1.04	1.04	2.3706	2.3790
	N	357	358	357	358	357	358	357	358	318	319
	Std. Deviation	134.29925	134.120	62.533	62.661	71.094	71.164	.202	.202	1.97661	1.96595
	Minimum	41.00	41	35	35	35	35	1	1	.00	.02
	Maximum	792.00	792	568	568	648	648	3	3	20.84	20.84
4	Mean	327.0598	327.06	186.34	186.34	219.65	219.65	1.18	1.18	2.3732	2.3732
	N	184	184	184	184	184	184	184	184	170	170
	Std. Deviation	183.97019	183.970	66.425	66.425	97.558	97.558	.416	.416	2.87832	2.87832
	Minimum	36.00	36	36	36	36	36	1	1	.06	.06
	Maximum	1064.00	1064	435	435	660	660	3	3	34.34	34.34
5	Mean	355.2072	355.21	188.04	188.04	230.80	230.80	1.22	1.22	2.1608	2.1608
	N	251	251	251	251	251	251	251	251	242	242
	Std. Deviation	198.98327	198.983	73.162	73.162	116.029	116.029	.471	.471	.95467	.95467

	Minimum	41.00	41	41	41	41	41	1	1	.02	.02
	Maximum	1238.00	1238	582	582	773	773	4	4	11.02	11.02
6	Mean	381.8077	381.81	202.38	202.38	243.40	243.40	1.27	1.27	2.1655	2.1655
	N	208	208	208	208	208	208	208	208	195	195
	Std. Deviation	232.77356	232.774	66.675	66.675	117.727	117.727	.571	.571	.79746	.79746
	Minimum	72.00	72	69	69	69	69	1	1	.01	.01
	Maximum	1424.00	1424	576	576	953	953	5	5	5.92	5.92
7	Mean	384.5181	384.52	189.67	189.67	220.17	220.17	1.23	1.23	2.2822	2.2822
	N	193	193	193	193	193	193	193	193	188	188
	Std. Deviation	261.16047	261.160	63.041	63.041	88.451	88.451	.481	.481	1.34691	1.34691
	Minimum	103.00	103	36	36	43	43	1	1	.03	.03
	Maximum	2294.00	2294	386	386	697	697	4	4	12.22	12.22
8	Mean	439.4660	439.47	206.14	206.14	268.72	268.72	1.36	1.36	2.3063	2.3063
	N	103	103	103	103	103	103	103	103	102	102
	Std. Deviation	296.67407	296.674	70.985	70.985	185.240	185.240	.827	.827	.76345	.76345
	Minimum	49.00	49	49	49	49	49	1	1	.08	.08
	Maximum	1705.00	1705	404	404	1203	1203	6	6	5.15	5.15
9	Mean	420.9273	420.93	211.49	211.49	249.13	249.13	1.24	1.24	2.2500	2.2500
	N	55	55	55	55	55	55	55	55	54	54
	Std. Deviation	236.45623	236.456	82.988	82.988	125.783	125.783	.508	.508	.79442	.79442
	Minimum	65.00	65	62	62	65	65	1	1	.28	.28
	Maximum	1006.00	1006	510	510	701	701	3	3	4.63	4.63
10	Mean	535.1212	535.12	197.76	197.76	331.45	331.45	1.73	1.73	2.2447	2.2447
	N	33	33	33	33	33	33	33	33	30	30
	Std. Deviation	308.24268	308.243	84.904	84.904	220.611	220.611	1.039	1.039	1.20611	1.20611
	Minimum	151.00	151	41	41	41	41	1	1	.05	.05
	Maximum	1242.00	1242	445	445	967	967	5	5	7.54	7.54
11	Mean	625.7333	625.73	217.53	217.53	353.60	353.60	1.67	1.67	2.3673	2.3673
	N	15	15	15	15	15	15	15	15	15	15
	Std. Deviation	430.32238	430.322	58.300	58.300	281.706	281.706	.976	.976	1.07526	1.07526
	Minimum	191.00	191	118	118	118	118	1	1	.08	.08
	Maximum	1474.00	1474	378	378	1231	1231	4	4	4.21	4.21
12	Mean	467.7778	467.78	201.33	201.33	291.56	291.56	1.56	1.56	2.2956	2.2956
	N	9	9	9	9	9	9	9	9	9	9
	Std. Deviation	234.58835	234.588	81.185	81.185	177.291	177.291	.726	.726	.79128	.79128
	Minimum	143.00	143	79	79	79	79	1	1	1.52	1.52
	Maximum	910.00	910	357	357	606	606	3	3	3.81	3.81
13	Mean	1209.5000	1209.50	166.50	166.50	867.00	867.00	5.00	5.00		
	N	2	2	2	2	2	2	2	2		

	Std. Deviation	89.80256	89.803	48.790	48.790	4.243	4.243	1.414	1.414		
	Minimum	1146.00	1146	132	132	864	864	4	4		
	Maximum	1273.00	1273	201	201	870	870	6	6		
	Mean	340.5468	340.50	189.19	189.25	222.73	222.76	1.20	1.20	2.2769	2.2788
	N	1527	1528	1527	1528	1527	1528	1527	1528	1415	1416
Total	Std. Deviation	225.34808	225.280	67.405	67.420	118.958	118.927	.522	.522	1.68552	1.68305
	Minimum	36.00	36	35	35	35	35	1	1	.00	.01
	Maximum	2294.00	2294	582	582	1231	1231	6	6	34.34	34.34

Appendix D. Experiment Walkthrough (Shimmer Experiment)

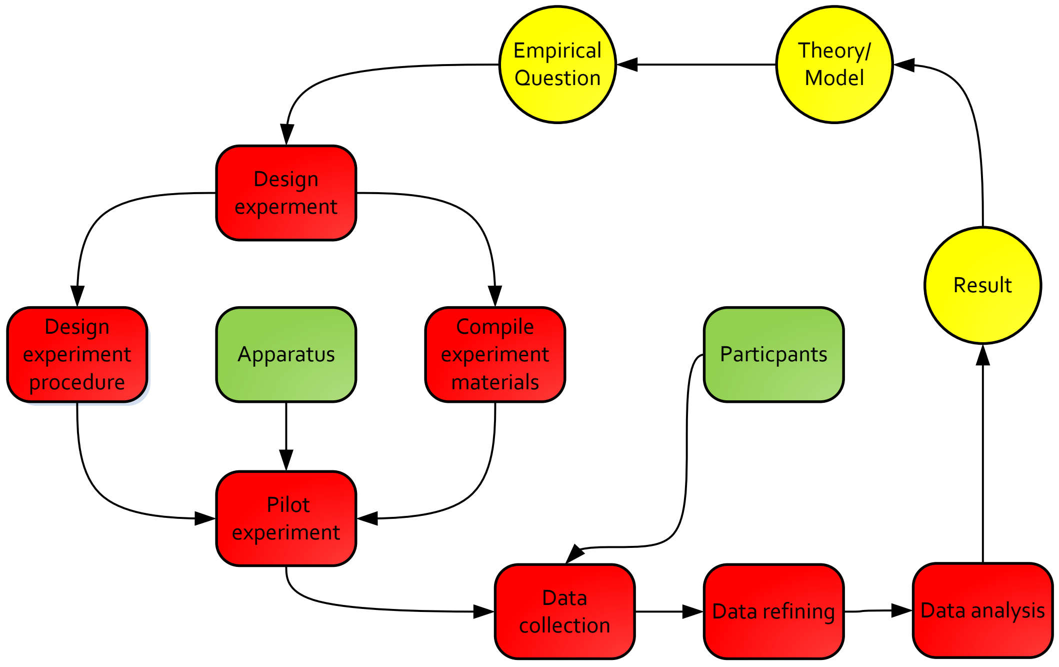


Figure D-1: Reading experiment life cycle

In order to demonstrate the work path of a reading experiment on our informatics system, we selected our Shimmer Experiment as an example to help readers to walkthrough the usage of every component of our system. Shimmer Experiment was the first reading experiment that was fully deployed by the system. The following sections are organized according to the sequential steps of experiment life cycle, which was shown in Figure D-1.

Design experiment

The goal of Shimmer Experiment was to investigate and compare two different shimmer patterns (i.e., asynchronous and synchronous) under identical and unrelated preview conditions. Therefore the experiment adopts 2 x 2 factorial design. The two factors are preview conditions and shimmer patterns.

Design experiment procedure

The steps of the experiment procedure are visualized in Figure D-2.

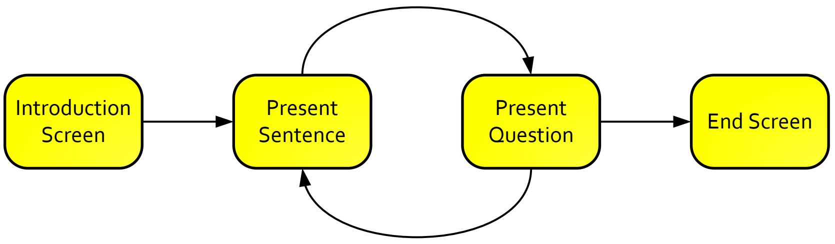


Figure D-2: Experiment procedures

Compile experiment materials

Experiment materials are usually single sentences. For this experiment, it was planned to present these sentences to readers with synchronous or asynchronous shimmer and for some sentences, preview changes in target words would be triggered when the eye across an invisible boundary prior to the target. Therefore, to facilitate presentation control, we need a list of variables not only related to the experimental sentences themselves but also to other information such as whether a word is a target word, a non-word mask, question, the type of shimmer, preview type and etc.. Some essential information is listed in Table D-1.

Table D-1: Experiment control variables

Variables	Sample Value
sequence number	1
non-word mask	qvsilcui
sentence	The scientists came up with a simple quotient resolving all doubts about their model
target word	quotient
question	What resolved the doubts about the model?
choice	quotient or masquerade
shimmer type	synchronous
preview type	identical
target word position	8

In this experiment, 192 sentences were selected and saved into a CSV data file together with information in above table. Since it is 2 x 2 factorial design, 4 CSV files with different shimmer type and preview type are predefined.

Pilot experiment

To run a pilot experiment, we should first compose our experiment script. The task of the script is to present experimental materials based on the designed procedure. To accomplish this task, some extra functions need to be implemented in order to connect each step in Figure D-2. An extended version of Figure D-2 is shown in Figure D-3, where the red box indicates these potential function modules that need to be implemented.

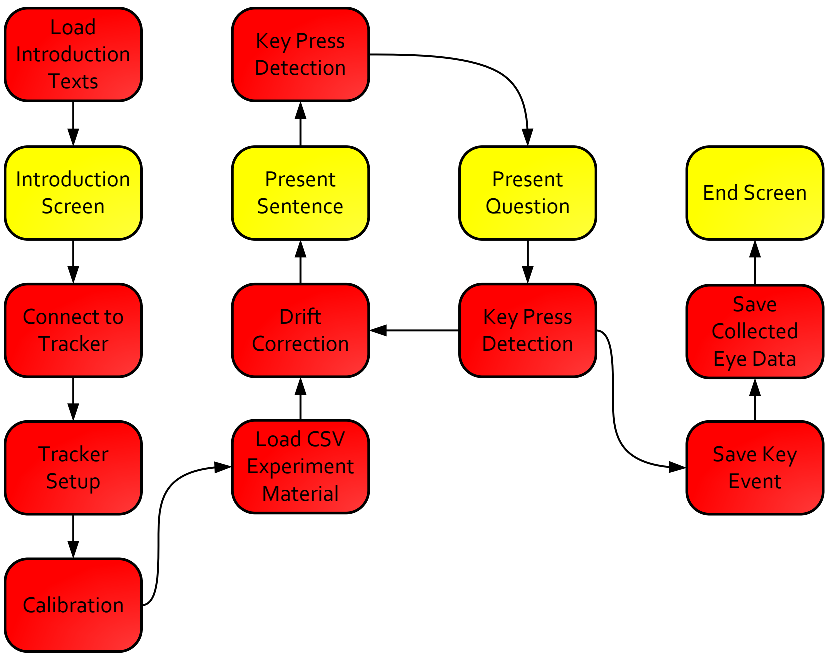


Figure D-3: Experiment procedures (extended)

In the next stage, these function modules are then were replaced by ExpExec APIs. Pseudo-code descriptions of these modules are shown in Figure D-4.

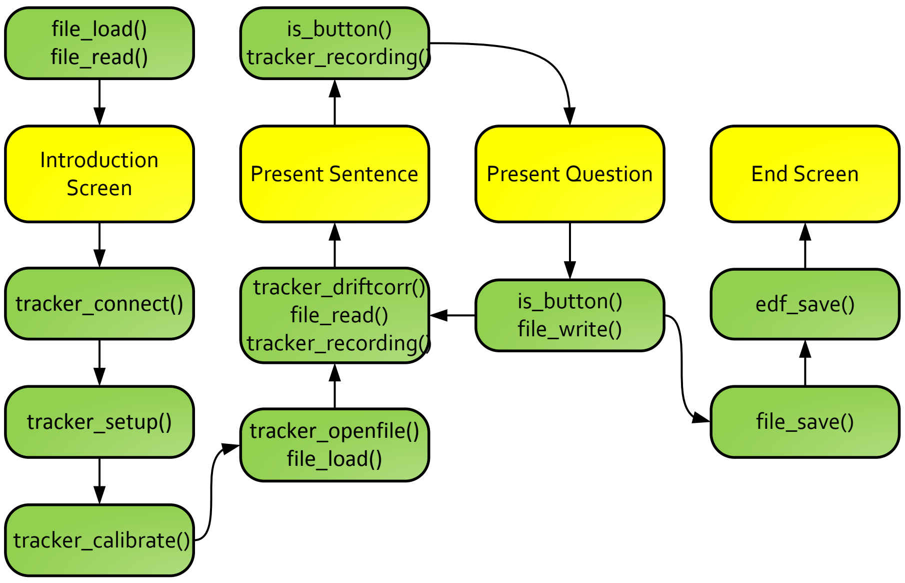


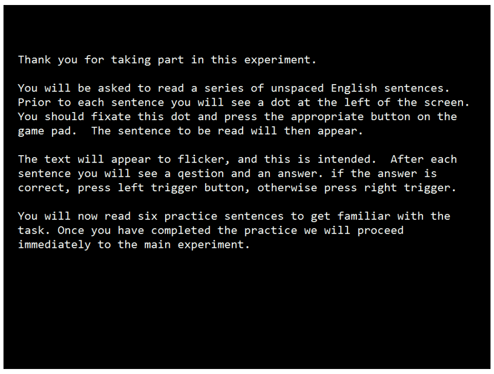
Figure D-4: Experiment procedures (pseudo-code)

At this stage, the ExpExec mouse simulation module can be used for testing scripts without needing to use an actual tracker. After script was fully tested, they were then packaged (with CSV experiment materials) and delivered to display machine, where ExpExec was pre-installed.

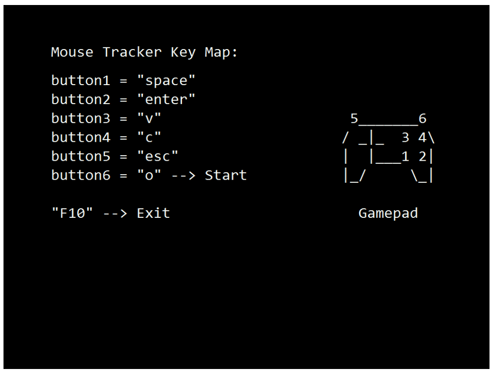
Data collection

During the data collection, each participant has their own working directory, where experiment script and other supplements (such as CSV files, text files, or image stimuli) were located. Recorded eye movement data would also be stored in this directory. These directories should be created by the experimenter before running the participant. The experimenter should also fill in the participant form and record participant observations during the data collection. If any inappropriate behavior is discovered, the experimenter should break the experiment by pressing ESC button on display machine and do a re-calibration.

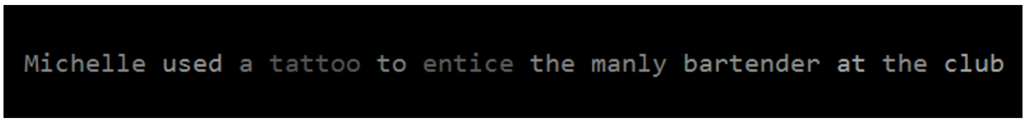
Some ExpExec running screen snapshots during the data collection phase were captured and shown in Figure D-5.

a) 

```
Thank you for taking part in this experiment.

You will be asked to read a series of unspaced English sentences.
Prior to each sentence you will see a dot at the left of the screen.
You should fixate this dot and press the appropriate button on the
game pad.  The sentence to be read will then appear.

The text will appear to flicker, and this is intended.  After each
sentence you will see a qestion and an answer. if the answer is
correct, press left trigger button, otherwise press right trigger.

You will now read six practice sentences to get familiar with the
task. Once you have completed the practice we will proceed
immediately to the main experiment.
```

b) 

```
Mouse Tracker Key Map:

button1 = "space"
button2 = "enter"
button3 = "v"                      5_______6
button4 = "c"                     / _|_  3 4\
button5 = "esc"                   |  |___1 2|
button6 = "o" --> Start           |_/      \_|

"F10" --> Exit                      Gamepad
```

c) 

```
Michelle used a tattoo to entice the manly bartender at the club
```

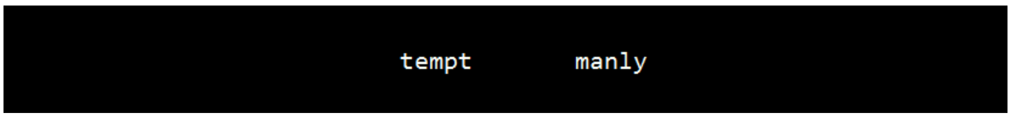
d) 

Figure D-5: ExpExec running screen snapshots.

a) Introduction screen; b) EyeLink tracker setup screen; c) Experiment sentence; d) Question (which word is appeared in the preceding sentence?)

Data refining

Before data analysis, collected eye movement data files (EyeLink EDF files in this case), are first converted to EyeMap XML format by EyeMap data parser. A config.xml can be used, if users want to control the data extraction process by using user-defined trial stop/start messages (messages that are inserted during the data collection to indicate start or stop of the experimental trial), or if they want to view the raw data plots in EyeMap. A config.xml sample used in this experiment is shown in Figure D-6.

```
<config>
    <samplerate>1000</samplerate>
    <rawout>false</rawout>
    <begin>PAGE</begin>
    <end>RECORD</end>
</config>
```

Figure D-6: A sample config.xml

After the data conversion, XML data files were loaded into EyeMap for refining. As shown in Figure D-7, practice trials marked as invalid and therefore will be removed from further outputs. We also selected 'Gaze-Y to Driftcorr Y' option to align all fixation dots on the line of screen center.

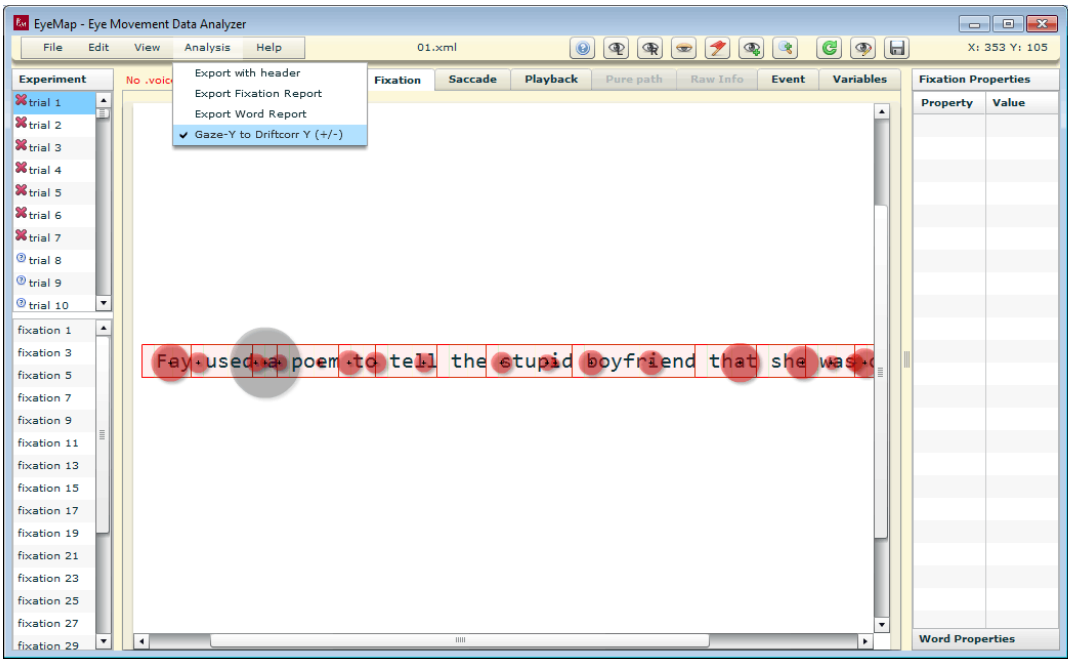
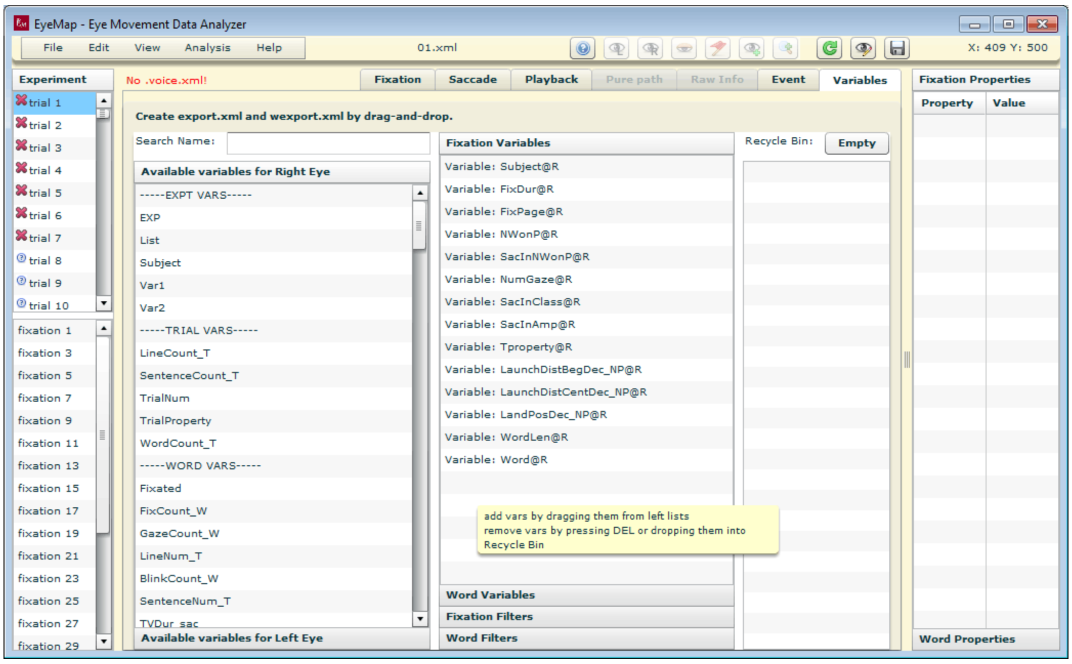


Figure D-7: EyeMap data preprocessing

Data analysis

When refining is over, we then started creating output variable list at EyeMap's "Variables" Tab. For this experiment, we need a fixation based variable report in CSV format, which includes fixation variables, such first fixation duration, saccade direction, landing distance and etc.



No .voice.xml!

Create export.xml and wexport.xml by drag-and-drop.

trial 1
trial 2
trial 3
trial 4
trial 5
trial 6
trial 7
trial 8
trial 9
trial 10

fixation 1
fixation 3
fixation 5
fixation 7
fixation 9
fixation 11
fixation 13
fixation 15
fixation 17
fixation 19
fixation 21
fixation 23
fixation 25
fixation 27
fixation 29

Available variables for Right Eye

-----EXPT VARS-----
EXP
List
Subject
Var1
Var2
-----TRIAL VARS-----
LineCount_T
SentenceCount_T
TrialNum
TrialProperty
WordCount_T
-----WORD VARS-----
Fixated
FixCount_W
GazeCount_W
LineNum_T
BlinkCount_W
SentenceNum_T
TVDur_sac

Available variables for Left Eye

Fixation Variables

Variable: Subject@R
Variable: FixDur@R
Variable: FixPage@R
Variable: NWonP@R
Variable: SacInNWonP@R
Variable: NumGaze@R
Variable: SacInClass@R
Variable: SacInAmp@R
Variable: Tproperty@R
Variable: LaunchDistBegDec_NP@R
Variable: LaunchDistCentDec_NP@R
Variable: LandPosDec_NP@R
Variable: WordLen@R
Variable: Word@R

add vars by dragging them from left lists
remove vars by pressing DEL or dropping them into
Recycle Bin

Figure D-8: Creating export variable list

This output can then be used as input to a wide range of statistical analysis software (e.g., R, Ihaka & Gentleman, 1996).