Text 2.0 and Eye Tracking Technology

A New Way to Enhance Reading and Information for Humanity
Overview of Presentation

- History of Reading
- How we read information today
- Text 2.0, what is it?
- Eye tracking technology
- Eye tracking algorithm
- Other applications
- The future
History of Reading

- Spoken language is a natural human form of communication that has been used for ~6 million years.

- Reading and text/symbols have only been around for 6,000 years
  - This is fairly new, we have not perfected it yet.
  - Text is a human invention.

- “Spoken words are the symbols of mental experience, and written words are the symbols of spoken words.” – Aristotle (first psychologist)
History of Reading

- People have always thought of ways to symbolize objects and thoughts.
- Sumerian Logographs from 4000 BC, the first known form of writing.
- This is a message that reads:

  - Quantities of the product:
    - c. 135,000 liters
  - Type of the product:
    - barley
  - Accounting period:
    - 37 months
In 2000 BC the Phoenicians developed the first methods to represent spoken language, an alphabet consisting of all consonants (no vowels).

Then, in 1000 BC the Greeks created an alphabet with vowels, we still use it today.
   This is considered one of humanities greatest inventions.

In 200 BC, punctuations were added to words.

In 700 AD, lowercase and uppercase letters were implemented by Medieval Scribes.

In 900 AD, spaces between words were added for the first time by Medieval Scribes. Before this, nobody could read silently except the great minds like Caesar and St. Ambrose, it was considered a highly complex skill.
   For the past 1000 years, text has remained virtually unchanged.
The first known written symbols of hominids are the pictures of animals on the walls of cavemen.

Since then, humans and other hominids have ‘written’ symbols in the dirt, on rocks/tablets, wood, scrolls (made from animal skin originally), all the way to the printing press and finally electronics.

Writing and printing is considered an art, today, we use text and symbology for just about everything.
Today, humans read and write in many different methods.

In the past 1000 years we've used written/printed text, now we use technology.

Some include:
- Computers
- Laptops
- eBooks
- Mobile phones
- iPad
- The list goes on...

Reading and writing have seen advancements in the past few years, but it has always remained quite similar.

In other words, we've always just either written what we wanted, or read what is provided. There has never been any ‘responsive’ system that reacts to us.

How can this change?
An idea to change reading forever: Text 2.0

- Text 2.0: text that knows its being read.
  - Interacts with user in real time

- Developed by Ralf Biedert from the German Research Center for Artificial Intelligence.

- How? With eye tracking technology, a system will “watch” a users eyes and perform automated responses to
  - What the users eyes are doing
  - Where the users eyes are
  - Pre programmed responses to user

- So, when a user is reading text, depending on the motion of the eyes, the text will respond accordingly.

- Some examples:
  - If a user looks at a word for a given amount of time, a definition will pop up.
  - If a user looks away from the page, an arrow will appear where the user left off.
  - When the user reaches the end of a page, the page will turn.
  - If a user starts to skim a page, only the most relevant words will appear, the rest of them dimmed out.

- http://www.youtube.com/watch?v=8QocWsWd7fc&feature=player_embedded
Overview of Hardware

- Tobii x120 eye tracker
  - Records users point of regard with 120 Hz
  - Outputs measured position in pixel coordinates for each eye
  - Then, take that data and process two independent median filters
  - Cluster spatially to discover fixations and cascades
Typical scenario:
- User tells search engine which results are relevant or not
- Feedback used to find an improved result

Similar to how eye tracking works, eye tracking uses an algorithm that detects and differentiates reading and skimming behaviors based on output from an algorithm.

They define the quality of a new eye movement

Then measure for determining the relevance of the documents

Identified correlations between eye movement measures and explicit relevance feedback are used for prediction methods
Algorithm main idea:
- Fixations are detected
- Transitions from one fixation to the next are classified resulting in ‘features’
- Scores associated with features are accumulated
- Thresholds for ‘reading’ and ‘skimming’ are determined
- If true, then the most plausible behavior is detected
Fixation Detection

- A new fixation is detected if 4 successive nearby gaze locations from the eye tracker are accumulated.

- For other gaze detections, they are checked to see if they fit in a 50x50 px rectangle with the previous detections.
  - If yes, then the new gaze location is assigned to the current fixation
  - If no, then it is either ignored or used as an indicator for a new fixation

- This functionality tolerates noise generated from random slight eye movements
Fixation Detection

- If there are at least 4 successive gaze locations that can be merged with the current fixation, the fixation has ended.

- Blinking of the eyes will end a fixation

- When a fixation ends, all previous gaze locations are averaged to one fixation coordinate
  - This can be used to keep a spot on a page when a user looks away
  - Also can be used to record the fixation for alternate uses and much more…
Classification of Fixation Transitions

- Each transition from one fixation to the next is classified according to length detection.

- These relate to features that occur during reading or skimming.

- Possible features:
  - Font size plays a major role here, it effects how fast our eyes will move.
  - Transition classification is based on letter space differences, not absolute pixel differences.

<table>
<thead>
<tr>
<th>Distance and direction in letter spaces</th>
<th>Feature</th>
<th>Reading detector score ( s_r )</th>
<th>Skimming detector score ( s_s )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 &lt; ( x ) &lt;= 11</td>
<td>Read forward</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>11 &lt; ( x ) &lt;= 21</td>
<td>Skim forward</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>21 &lt; ( x ) &lt;= 30</td>
<td>Long skim jump</td>
<td>-5</td>
<td>8</td>
</tr>
<tr>
<td>-6 &lt;= ( x ) &lt; 0</td>
<td>Short regression</td>
<td>-8</td>
<td>-8</td>
</tr>
<tr>
<td>-16 &lt;= ( x ) &lt; -6</td>
<td>Long regression</td>
<td>-5</td>
<td>-3</td>
</tr>
<tr>
<td>( x ) &lt; -16 and ( y ) according to line spacing</td>
<td>Reset jump</td>
<td>5 and line delimiter</td>
<td>5 and line delimiter</td>
</tr>
<tr>
<td>All other movements</td>
<td>Unrelated move</td>
<td>Line delimiter</td>
<td></td>
</tr>
</tbody>
</table>
The detection of certain reading actions is done with formulas just like the ones shown in the last slide.

In order to differentiate between reading and skimming, two independent detectors analyze sequences of features and accumulate associated scores.

\[ \sum_{f \in DF} s_d(f) > t_d \] : formula used for detection data.

r: reading detector uses different scores \( s_r \)

Skimming detector uses \( s_s \)

DF: multiset of contained features

each detector \( d \in \{r, s\} \) is tested for whether there is enough evidence for reading or skimming behavior.
Reading and Skimming Detection

- To detect for reading or skimming behavior, they compare the accumulated scores to detect for specific thresholds $t_d$ which tests for each $d$.

- If only one detector has enough evidence, an appropriate behavior is detected.

- If both have enough evidence for a text row, a more specific behavior is detected by comparing the accumulated scores from each detector.

- Then we can use thresholds from image in previous slides.
Reading and Skimming Detection

- This is the result of the skimming detection algorithm on text.
- Circles represent fixations, the diameter represents fixation duration.
- Classification of fixation transitions (detected features) shown by abbreviations on connecting lines:
  - R: read forward
  - S: skim forward
  - L: short regression
  - Reset: reset jump

In 1935, the findings of collaborators Frederic Gibbs, James Lennox, and Hallowell Davis from Harvard on the use of EEG in epilepsy was published. Since EEG results no pain or side effects, it is broadly included as a medium for identifying brain irregularities. The EEG is abnormalities. Persons who suffer from grand mal epilepsy while those with petit mal epilepsy have arch-shaped chemical stimuli. For instance, the use of drugs will make us asleep, the waves' pattern changes a few times have high frequency but low amplitude.
The eye tracking researchers tested many readers and recorded the systems responses.

They could then fine tune their software to meet the average requirements for people.
Eye tracking technology that responds to a user's actions instead of having a user input actions is **natural**, it flows and feels effortless.

Eye tracking devices are getting so small that they can be implemented anywhere.

**Issue:** will the user define the actions based on his preferences, or will there be pre programmed responses?

**Augmented text:** the creator of the text must define the actions (like HTML or JavaScript).

**Augmented reading:** based on gaze data and background history from users.
- No upper temporal boundary for assists
- Uses a second dimension, reaction time, that defines gazes and their corresponding reactions
- Consider short and long term gazes
Why another framework?

- GUI applications in C / C# / Java ... :-(
- proper (‘realistic’) layout of text difficult
- even simple changes can be hard if application is hardwired
- isn’t there a way to separate structure from layout from logic, even for gaze responsive applications?
Yes, HTML!

- est. standards: HTML, CSS, JS!
- 20 yrs. exp. in text
- Lots of developers
- perfect for rich documents
- let’s make it gaze!
The Browser Plug In

- best option for flexibility
- not bound to any specific browser (sort of)
- we benefit from their advancements
- clear separation of gaze logic from rest of rendering and layout
- but also a number of problems (we’ll discuss them later)
Intuitive integration of gaze into the browser

- stick to tags similar to mouse events
- use JavaScript for global things
- be high level
Understanding Gaze MarkUp

- HTML tags correspond to screen areas
- Tag attributes make areas active
- on {Fixation, Gaze-Over, GazeOut, Perusal, Read}
onFixation

- uses filtered data and computes fixations
- executed every time a fixation occurs inside tagged area

**Diagram:**
- Multicolored circles indicating fixations:
  - Blue = ignored fixation
  - Green = relevant fixation

**Legend:**
- First Fixation
- Last Fixation

**Area on screen w. handler**
onGazeOver / onGazeOut

- works like onFixation
- but executed only once respectively

Diagram:
- First Fixation
- Last Fixation
- Area on screen w. handler
- = ignored fixation
- = relevant fixation
onPerusal / onRead

- not triggered upon single fixations but on reading behavior
- onPerusal (skimming / reading / ...)
- onRead (reading only; currently same as onPerusal)
JavaScript Gaze API

- required for glue code
- applications can tweak setup (e.g., enable logging, tracking device, ...)
- register global (application wide) listeners
- register „high performance“ extensions (i.e., computationally intensive algorithms) and add-ons like speech IO, ...
Other Features

- session recording & replay
- write once! (evaluation code also works in realtime)
- online monitoring & diagnosis
The Tracking Server feeds the Plugin

- remote server provides gaze to browser plugin
- not in client because of restrictions (loading drivers, startup speed)
The Tracking Server feeds the Plugin

- Remote server provides gaze to browser plugin
- Not in client because of restrictions (loading drivers, startup speed)
Background Services

- provide services that can't be included directly because of size and startup speed
- discovered automatically
Problems of Integration

- Browsers do not provide ...
  - ... word positions
  - ... a fast geometry API for plugins
- we need to do some ‘magic’
  - pseudo-renderer
  - updating elements in background
Text 2.0

"It is true that on that you can't have come from very far away ..."<br>
And he sank into a reverie, which lasted a long time. Then, taking my sheep out of <span onRead="eventHandler.playAudio('papercrumple.wav');">his pocket</span>, the contemplation of his treasure.<br> You can imagine how my curiosity was aroused by "My little man, where do you come from? What is this 'where I live,' of which you speak?"

- Augmented text

- Most prototypes that use this kind of technology use the Text 2.0 framework
  - Browser plug in that handles gazes and enables web pages with JavaScript to react to gazes
  - Similar to a mouse and keyboard

- Most low level event handling is performed inside the browser plug in
  - User interface responses are defined in HTML and JavaScript
  - Gaze effects triggered by fixations on rendering position of given elements
Projects

- Many projects have been started and completed that use this technology.
- eyeBook
- PEEP (Processing Easy Eye Tracker Plugin)
- FCC leaks of Apple patents
- Open source applications
- Many more…
eyeBook

- Original Text 2.0 project
- Uses augmented text (manual control of actions by coder)
- Had many actions in response to users eyes:
  - Sound effects
  - Images that pop up
  - Translations: popped up above word
  - Intelligent footnotes
  - Speech IO: here, the user asks a question such as “what does this mean?” or “can I have more information about that?”
    - An action response would then appear

- QuickSkim: all words given a class from 0 (most frequent) to 9 (least frequent)
  - Each class is numerically assigned an opacity value for its relevance
  - The user now skims through the words, it detects skimming, then blurs words
PEEP (Processing Easy Eye Tracker Plugin)

- “The probably easiest way on earth to develop stunningly looking eye tracking applications (and one of our most eye-appealing by-products we’ve created so far... ;-)

- [http://text20.net/node/14](http://text20.net/node/14)

- Open source program from the makers of Text 2.0 where you can develop anything that you want to use eye tracking tech. for

- There are many sample applications on the web page

- Code example

- There are some bugs...
Apple's patent application includes technology that "determine(s) the location of gaze relative to display screen" in order to modify onscreen objects (image via AppleInsider, info from wired.com).
Apple will probably be using this technology as a crazy update to the iPad, yet again putting it ahead of all other eReaders available.

According to rumors, this could be deployed well within 5 years, perhaps even much sooner.

This is exciting news, and developers, writers, and corporations are developing items like interactive children's books, interactive web pages, operating systems and much much more...
Alternative uses for Eye Tracking

- Use the eye tracking technology as a mouse for your computer
- [http://www.youtube.com/watch?v=RmBi7WYveYo](http://www.youtube.com/watch?v=RmBi7WYveYo)
- Training purposes
- Medical
- Vehicular interactions
- Biological research
- Creating models for AI systems in order for them to see like us
- Advertisement placement: use the gaze data from lots of people to determine the BEST way to structure your ad.
- The list goes on...
Me!

- I have been in contact with Ralf Biedert about jobs and internship opportunities.
- Unfortunately it’s in Germany... I don’t know any german...
Sources

- Text20.net
  - Here I found most of the information including papers and slides
  - Slides: the Text 2.0 framework
- Papers:
  - The Text 2.0 framework
  - Eye relevance
  - Gaze query expansion
Thanks!

- Thank you for listening!
- I hope it was informative!