The Journal of Open Source Software

Eyestream: An Open WebSocket-based Middleware for Serializing and Streaming Eye Tracker Event Data from Gazepoint GP3 HD Research Hardware

Matthew L. Hale[1]

1 School of Interdisciplinary Informatics, University of Nebraska at Omaha

DOI: 10.21105/joss.01620

Software

- Review
- Repository
- Archive

Editor: Christopher R. Madan

Reviewers:

- @RingoHHuang
- @adswa

Submitted: 25 May 2019
Published: 21 November 2019

Summary

Eye trackers enable a wide range of research and development activities in academia, medicine, and industry. Researchers and developers using eye trackers are able to collect rich user interaction data such as user fixation durations, points of gaze, and 3D models of user pupils that enable unique analysis and applications in areas such as user-interaction design, rehabilitative/restorative medicine, and marketing. Most eye trackers, including the two with the largest market share `Gazepoint GP3 HD` ("GP3 HD Eye Tracker 150Hz," 2018) and the `Tobii Pro X3-120` (Olsen, 2012), ship with proprietary desktop software packages that collect, house, and analyze captured data. This closed source delivery model makes it difficult for researchers and developers to directly interact with raw eye tracker data, which prevents custom analysis and hinders application development and integration efforts. Further, most closed-source eye tracker analysis tools do not support streaming and real-time analysis applications.

While the Gazepoint GP3 HD ("GP3 HD Eye Tracker 150Hz," 2018) is designed to work with Gazepoint's proprietary analysis software, it also provides an underutilized alternative low-level XML-based API called OpenGaze for extracting data from the hardware. We have created a tool on top of OpenGaze called `Eyestream`. `Eyestream` is capable of serializing hardware-captured eye tracker data as JSON and streaming it in real-time, at frequencies up to 150hz, to desktop, web, or mobile applications. `Eyestream` is an open source package designed to operate as middleware between the Gazepoint GP3 and applications that consume eye tracker data for analysis or visualization purposes. It was implemented using Python 2.7, a web socket server framework called `Django Channels` ("Django Channels," 2018), an in-memory database caching tool called `Redis` (Carlson, 2013), and a modified version of `PyOpenGaze` (Dalmaijer, 2017). `Eyestream` is containerized using `Docker` (Boettiger, 2015).

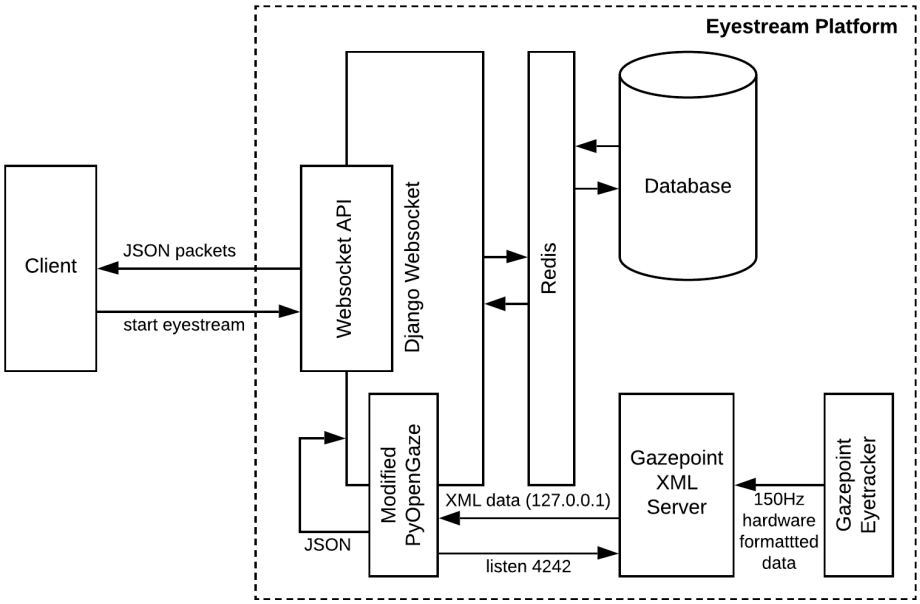

Figure 1: System-level architecture of eyestream

Eyestream and Its Uses

`Eyestream` provides a real-time streaming interface to other software applications that enable a wide-variety of applications. The streaming interface is built using websockets. Any application on any platform or written in any language can interact with `Eyestream` provided that it can establish a websocket. All major languages (C/C++/C#, Python, Java, JavaScript, Ruby, Go, etc) have direct or library support for websockets, making `Eyestream`'s approach widely open. Figures 2, 3, and 4 show the `Eyestream` platform from three perspectives: 2) The `Eyestream` server running in the command line, 3) Eyestream's invocation of the underlying Gazepoint eye monitoring software, and 4) a console, in Google Chrome Developer Tools, printing streaming eye data it is receiving from the server.

```
Performing system checks...

System check identified no issues (0 silenced).
March 28, 2019 - 15:16:12
Django version 1.11.12, using settings 'cybertrustgazepoint.settings'
Starting Channels development server at http://127.0.0.1:8888/
Channel layer default (asgi_redis.core.RedisChannelLayer)
Quit the server with CTRL-BREAK.
2019-03-28 15:16:12,007 - INFO - worker - Listening on channels http.request, websocket.connect, websocket.disconnect, websocket.receive
2019-03-28 15:16:12,006 - INFO - worker - Listening on channels http.request, websocket.connect, websocket.disconnect, websocket.receive
2019-03-28 15:16:12,009 - INFO - worker - Listening on channels http.request, websocket.connect, websocket.disconnect, websocket.receive
2019-03-28 15:16:12,009 - INFO - worker - Listening on channels http.request, websocket.connect, websocket.disconnect, websocket.receive
2019-03-28 15:16:12,013 - INFO - server - HTTP/2 support not enabled (install the http2 and tls Twisted extras)
2019-03-28 15:16:12,013 - INFO - server - Using busy-loop synchronous mode on channel layer
2019-03-28 15:16:12,015 - INFO - server - Listening on endpoint tcp:port=8888:interface=127.0.0.1
[2019/03/28 15:17:12] HTTP GET / 200 [0.04, 127.0.0.1:54618]
Not Found: /favicon.ico
[2019/03/28 15:17:12] HTTP GET /favicon.ico 404 [0.07, 127.0.0.1:54618]
[2019/03/28 15:17:15] WebSocket HANDSHAKING /gazepoint/ [127.0.0.1:54627]
[2019/03/28 15:17:15] WebSocket CONNECT /gazepoint/ [127.0.0.1:54627]
6520
Gazepoint controller starting...
Starting Gazepoint data capture...
```

Figure 2: Eyestream's websocket server running on console

Figure 3: Gazepoint real-time eye monitor invoked by Eyestream

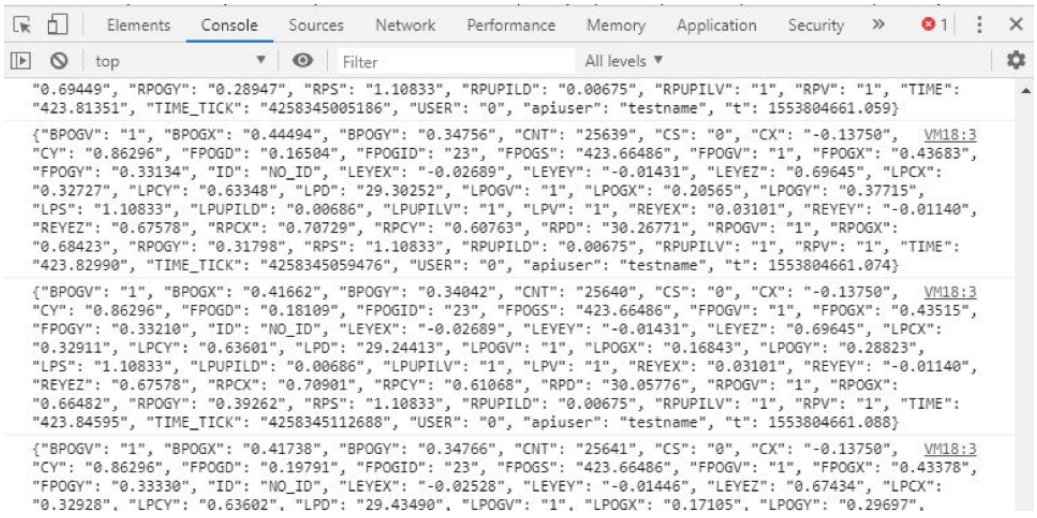

Figure 4: Data streaming to a browser is printed in the console

One common use for real-time eyetracker data is the generation of real-time heatmaps showing the temporal progression of user eye movements. An example application using `Eyestream` to build real-time heatmaps is shown in Figure 5. This application, called `Cybertrust`, is a research and training platform that helps users identify phishing attempts. In this example, as the user's gaze travels across the screen (depicted as the black line), the heatmap overlay gradually changes color to reflect the amount of time spent fixating on a particular area. Heatmap data is rendered using a D3 plot so that users and trainers can see what they are

focusing on within phishing content.

Other possible applications for `Eyestream` include medical tools for restorative eye or stroke care (Kasten, Bunzenthal, & Sabel, 2006), examining areas of interest within a page for UI design or marketing (Goldberg, Stimson, Lewenstein, Scott, & Wichansky, 2002), and as an interaction modality for video games (Corcoran, Nanu, Petrescu, & Bigioi, 2012) or virtual reality systems.

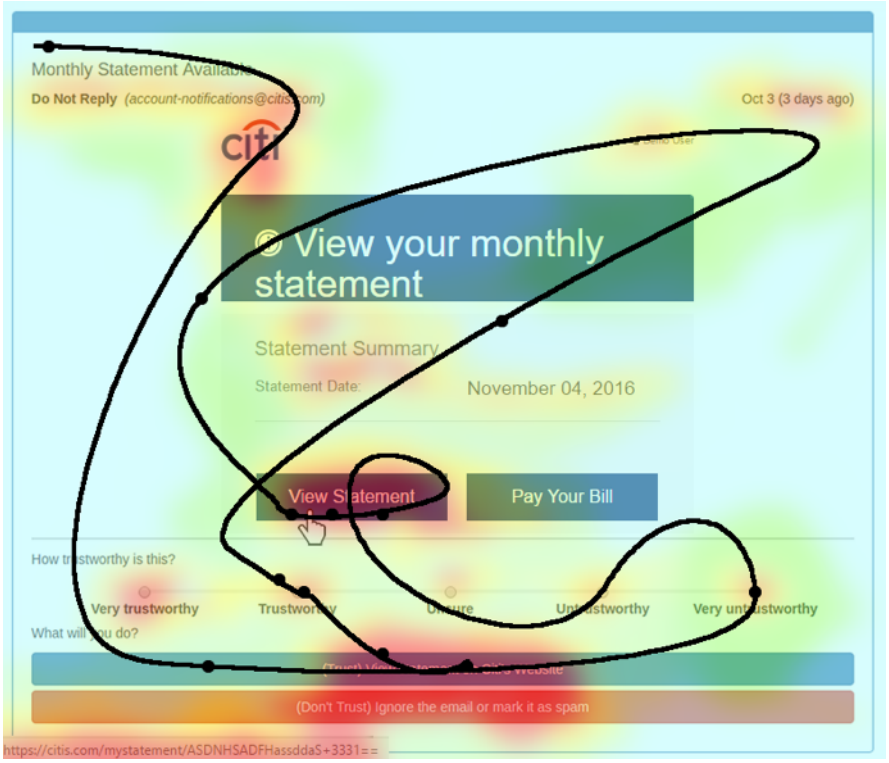

Figure 5: Heatmap of eye movements as viewed in a phishing training app

On-going Research Projects Using Eyestream

Eyestream is currently in use within the `Cybertrust` phishing research platform (M. L. Hale et al., 2015; M. L. Hale, Gamble, et al., 2015; Hale, Walter, Lin, & Gamble, 2016; Hefley, Wethor, & Hale, 2018). `Cybertrust` is a gamified experimentation platform used to identify factors related to phishing victimization.

License

Eyestream is licensed under the GNU General Public License and can be found on the following GitHub repository: https://github.com/MLHale/eyestream.

Acknowledgements

We acknowledge contributions by Gabi Wethor (gewethor on GitHub) for her work in testing the installation and usage instructions.

References

Boettiger, C. (2015). An Introduction to Docker for Reproducible Research. *ACM SIGOPS Operating Systems Review*, *49*(1), 71–79. doi:10.1145/2723872.2723882

Carlson, J. L. (2013). *Redis in Action*. Manning Publications Co.

Corcoran, P. M., Nanu, F., Petrescu, S., & Bigioi, P. (2012). Real-time Eye Gaze Tracking for Gaming Design and Consumer Electronics Systems. *IEEE Transactions on Consumer Electronics*, *58*(2), 347–355. doi:10.1109/TCE.2012.6227433

Dalmaijer, E. (2017, May). Esdalmaijer/PyOpenGaze. *GitHub*. Retrieved from https://github.com/esdalmaijer/PyOpenGaze

Django Channels. (2018).*Django Channels - 2.1.7 documentation*. Retrieved from https://channels.readthedocs.io/en/latest/

Goldberg, J. H., Stimson, M. J., Lewenstein, M., Scott, N., & Wichansky, A. M. (2002). Eye Tracking in Web Search Tasks: Design Implications. In *Proceedings of the 2002 symposium on eye tracking research & applications* (pp. 51–58). ACM. doi:10.1145/507079.507082

GP3 HD Eye Tracker 150Hz. (2018).*Gazepoint*. Retrieved from https://www.gazept.com/product/gp3hd/

Hale, M. L., Gamble, R. F., & Gamble, P. (2015). CyberPhishing: A Game-based Platform for Phishing Awareness Testing. In *2015 48th Hawaii International Conference on System Sciences* (pp. 5260–5269). IEEE. doi:10.1109/HICSS.2015.670

Hale, M. L., Gamble, R., Hale, J., Haney, M., Lin, J., & Walter, C. (2015). Measuring the Potential for Victimization in Malicious Content. In *2015 IEEE International Conference on Web Services* (pp. 305–312). IEEE. doi:10.1109/ICWS.2015.49

Hale, M. L., Walter, C., Lin, J., & Gamble, R. F. (2016). Apriori Prediction of Phishing Victimization Based on Structural Content Factors. *International Journal of Services Computing*, 1–13. doi:https://doi.org/10.29268/stsc.2017.5.1.1

Hefley, M., Wethor, G., & Hale, M. L. (2018). Multimodal Data Fusion and Behavioral Analysis Tooling for Exploring trust, Trust-propensity, and Phishing Victimization in Online Environments, 862–871. doi:10.24251/HICSS.2018.108

Kasten, E., Bunzenthal, U., & Sabel, B. A. (2006). Visual Field Recovery After Vision Restoration Therapy (VRT) is Independent of Eye Movements: An Eye Tracker Study. *Behavioural brain research*, *175*(1), 18–26. doi:10.1016/j.bbr.2006.07.024

Olsen, A. (2012). The Tobii I-VT fixation filter. *Tobii Technology*, 1–21.