Dirk P. Janssen Licia Calvi Stefano Gualeni Michelle Westerlaken Marcello Gómez Maureira

A Framework for Biometric Playtesting of Games

Abstract

A framework is described that can assist game developers in using biometric (psychophysiological) methods while playtesting. Biometric methods can give developers a valuable additional window on the playtester's experience.

The Use of Biometrics

When developing a game (or any other media product), it is important to collect as much feedback from users as possible. After all, users will determine the success or failure of the game and their decision to play, buy and recommend a game depends partially on having a smooth and satisfying play experience.

Several methods for collecting data during playtests have been developed, with interviews and observation still being the most popular ones. Newer methods include logging game metrics and collecting data about the physical state of the player, called biometrics or psychophysiology.

Both game metrics and biometrics add objective measurements to the subjective results from interviews and player observation. The industry has widely recognized the role of game metrics in playtesting and in the continuous evaluation of a game after it has gone live.

Game metrics are often used for marketing (conversion rate), but it feed into user experience analysis [5]. In a large number of studies over the past years, the value of the biometric approach has been demonstrated, see reviews in [1] and [2], but the methods is infrequently used in practice.

We think the low rate of adoption of the biometrics method has to do with a number of factors, the most important one being lack of a widely adopted and generic framework for measuring this data. For game metrics, several commercial frameworks exist (*playtomic.com*, *mochibot.com*, *flurry.com*) and it is relatively easy to program a proprietary system.

Although expertise is required for the successful in-depth analysis of game metrics, a number of widely accepted measures can be automatically computed by the existing frameworks (conversion rate, time on game, heatmap of player deaths, etc.). A framework has been developed for emotion

Requirements

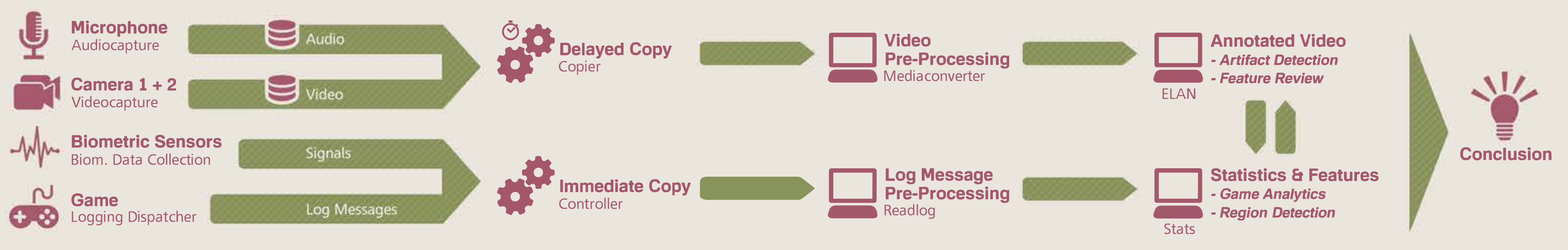
Our long-term goal is to make the use of biometrics just as straightforward as the use of game metrics. In this paper, we will focus on the cornerstone of all data acquisition: **the data collection framework.**

A biometric data collection framework for games should, to our mind, have the following properties:

- Simple and straightforward interface for programmers
- As much overlap between game metrics logging and biometrics logging as possible
- Functioning truly cross platform
- Independent of the biometrics hardware provider
- Usable in a multiplayer setting
- Automated test running and data aggregation
- Automatic generation of aggregated data over levels, maps, or whichever unit of analysis is indicated
- Easily reusable output in the form of PDF figures and spreadsheet files
- Using a widely accepted data storage format

Implementation

In essence, our framework is a collection of small programs that each do one dedicated task related to data collection. An overview of the various building blocks is given in the figure below. The assignment of these blocks to one or many computers is completely up to the user.



The four most important blocks acquire a type of data (biometrics, audio, video, logging) and either store this data locally or transmit it to the server. We commonly work with local storage of video and audio data and server storage of logging and biometrics data. Nightly jobs ensure that all data ends up on the server eventually. A small extension that we are working on now will allow low-res video to be sent to the network directly (for monitoring), while using delayed copy for the full resolution.

The biometric data collection program is responsible for interfacing with the hardware. This is the only part of the framework which is hardware dependent: it will initialize the biometrics hardware, pull and transmit data to the server.

The audiocapture and videocapture programs do just what their names imply. Both programs create two types of output: A common audio file (wav format) or video file (avi format), and a stream of synchronization messages that are sent to the server.

The logging dispatcher is a DLL that is loaded by the game under test. The DLL will take care of synchronizing with the server and sending the messages over the wire. We have alternative implementations for scenarios in which a DLL cannot be used, and which use ports or HTTP-GET requests (but provide less accurate timing).

A logging message is a simple text string, adhering to this standard: Words are separated by spaces; the first word is the command. (*COLLECT, JUMP, FIRE,* etc.); each of the following words is a key-value pair, written with an equal sign (*object=coin, x=40, bullet=1,* etc.).

Once the data arrives on the server, they are stored in the Hierarchical Data Format (version 5) [4]. This format, which was developed by NCSA and NASA, is specifically designed for storing numerical data in matrix format. All data files from one player are kept together to facilitate archiving.

Two pre-processing steps are required before the data can be analyzed: A first program will align the data onsets of the streams by trimming data. It will also convert the video data from its raw avi format to compressed mp4, adjusting it to be exactly in sync with the other streams as it is processed.

The second program will extend the event-based logging data as received from the game. We require game-state information to answer analysis questions like ''which actions do players use in this area''. The logging pre-processor adds such information, marks regions of interest, and introduces new logging messages to facilitate analysis.

To check the integrity of the data and to manually mark stretches of time for exclusion, we turn to a multimedia annotation tool ELAN [6]. This program will show multiple video tracks, the audio track, and annotation tiers. We can include a biometric signal in the display too. ELAN is used to give the analyst feedback on when episodes, levels, and events occurred, and it is used to mark artifacts (sneezing, coughing, etc.) for exclusion.

Finally, a statistics module will aggregate metrics and biometrics data over the requested regions, time spans, levels, or any combination of the above. We use Python as a flexible and accessible query language to formulate such aggregations. A number of standard graphs and tables are produced. The metric and biometrics data is also written to a CSV spreadsheet format, so it can be further processed and analyzed.

Limitations

A limitation of the current implementation is that the statistics module produces mainly descriptive statistics, the inferential statistics are still under development. We do not have found the need for machine learning algorithms, although interesting approaches exist and we will certainly consider including these in the future [7]. This would also open the framework to more extensive player modelling [8].

Conclusion

In conclusion, our framework covers most of the requirements for running biometrics analysis on playtesters. We will soon be making it available for interested commercial parties.

NHTV Breda University of Applied Sciences

[1] R. Mandryk. Physiological measures for game evaluation. In K. Isbister and N. Shaffer", editors, Game Usability: } Advice from the Experts for Advancing the Player Experience. Morgan Kaufmann, 2008.
[2] M. Seif El-Nasr, A. Drachen, and A. Canossa, editors. "Game Analytics: Maximizing the Value of Player Data". Springer, 2013.
[3] J. Wagner, F. Lingenfelser, B. Nikolaus and E. André. Social Signal Interpretation (SSI) - A Framework for Real-time Sensing of Affective and Social Signals. Künstliche Intelligenz, Springer, 2011, 25.
[4] The HDF Group. The HDF5 User's Guide, November 2012. www.hdfgroup.org/HDF5/doc/UG/index.html.
[5] B. Weber, M. Mateas, and A. Jhala. Using data mining to model player experience. In FDG Workshop on Evaluating Player Experience (EPEX 2011), Bordeaux, France, 2011.
[6] P. Wittenburg, H. Brugman, A. Russel, A. Klassmann, and H. Sloetjes. Elan: a professional framework for multimodality research. In Proceedings of the Fifth International Conference on Language Resources and Evaluation, 2006. Produced by the Max Planck Institute, tla.mpi.nl/tools/tla-tools/elan.
[7] G. N. Yannakakis, H. P. Martinez, and A. Jhala. Towards Affective Camera Control in Games User Modeling and User-Adapted Interaction, Springer, 2010.
[8] N. Shaker, G. N. Yannakakis and J. Togelius. Towards Player-Driven Procedural Content Generation. In Proceedings of ACM Computing Frontiers Conference, pp. 237-240, 2012.