Designing Multimodal Interactive Systems Using EyesWeb XMI

Gualtiero Volpe
Paolo Alborno
Antonio Camurri
Paolo Coletta
Simone Ghisio
University of Genova
DIBRIS
Genova, Italy
gualtiero.volpe@unige.it
paoloalborno@gmail.com
antonio.camurri@unige.it
paolo.coletta@unige.it
simoneghisio@gmail.com

Maurizio Mancini
Alberto Massari
Radoslaw Niewiadomski
Stefano Piana
Roberto Sagoleo
University of Genova
DIBRIS
Genova, Italy
maurizio.mancini@unige.it
alby@infomus.org
radoslaw.niewiadomski@dibris.unige.it
stefano.piana@dist.unige.it
sax@infomus.org

Abstract

This paper introduces the EyesWeb XMI platform (for eXtended Multimodal Interaction) as a tool for fast prototyping of multimodal systems, including interconnection of multiple smart devices, e.g., smartphones. EyesWeb is endowed with a visual programming language enabling users to compose modules into applications. Modules are collected in several libraries and include support of many input devices (e.g., video, audio, motion capture, accelerometers, and physiological sensors), output devices (e.g., video, audio, 2D and 3D graphics), and synchronized multimodal data processing. Specific libraries are devoted to real-time analysis of nonverbal expressive motor and social behavior.
The EyesWeb platform encompasses further tools such EyesWeb Mobile supporting the development of customized Graphical User Interfaces for specific classes of users.
The paper will review the EyesWeb platform and its components, starting from its historical origins, and with a particular focus on the Human-Computer Interaction aspects.

Author Keywords

Multimodal interactive systems; visual programming languages; EyesWeb XMI

ACM Classification Keywords

H.5.2 [Information interfaces and presentation (HCI)]: User interfaces

Introduction

Summer 1999, Opening of Salzburg Festival, Austria: In the music theatre opera *Cronaca del Luogo* by Italian composer Luciano Berio, a major singer (David Moss) plays a schizophrenic character at times appearing wise and calm, while other times appearing crazy, with nervous and jerky movements. Some of his movement qualities are automatically extracted by using sensors embedded in his clothes and a flashing infrared light on his helmet, synchronized with video cameras positioned above the stage. This information is used to morph the singer's voice from profound (wise) to a harsh, sharp (crazy) timbre. The impact with such concrete real-world applications of multimodal analysis and mapping was of paramount importance to shape the requirements for our first publicly available version of *EyesWeb* [1].

In particular, the need for fast prototyping tools made us leave the concept of EyesWeb as a monolithic application, to be recompiled and rebuilt after any possible minor change, and made us move to a more flexible approach, which was being already adopted by other software platforms both in the tradition of computer music programming languages and tools and in other domains such as simulation tools for system engineering. EyesWeb was thus conceived as a modular software platform, where a user can assemble the single modules in an application by means of a visual programming language. As such, EyesWeb supports its users in designing and developing interactive multimodal systems is several ways, such as for example (i) by providing built-in input/output capabilities for a broad range of sensor and capture systems, (ii) by enabling to easily define and customize how data is processed and feedback is generated, and (iii) by offering tools for creating a wide palette of interfaces for different classes of users.

Since then EyesWeb was reworked, improved, and extended along years and went through five major versions, being always available for free[1]. Nowadays, it is employed in various application domains, going beyond the original area of computer music and performing arts, and including for example active experience of cultural heritage, exergaming, education and technology-enhanced learning, therapy and rehabilitation.

This paper is organized as follows: the next section presents some related work, i.e., other modular platforms endowed with a visual programming language with a particular reference to multimedia and multimodal systems; then, the major components of the EyesWeb platform are introduced; finally, the different classes of users for EyesWeb and the reasons that make it suitable for fast prototyping of applications including interconnection of smart objects are discussed under an HCI perspective.

Related work

Whereas general-purpose tools such as, for example, Mathworks' Simulink exists since long time, platforms especially devoted to (possible real-time) analysis of multimodal signals are far less common. *Max* [9] is a platform and a visual programming language for music and multimedia, originally conceived by Miller Puckette at IRCAM, Paris, and nowadays developed and maintained by Cycling '74. Born for sound and music processing in interactive computer music, it is also endowed with packages for real-time video, 3D graphics, and matrix processing. *Pd* (*Pure Data*) [10] is similar in scope and design to Max. It also includes a visual programming language and it is intended to support development of interactive sound and music computing applications. The addition of GEM (Graphics Environment for

[1] http://www.casapaganini.org
The actual released version is EyesWeb 5.6.0.0.

Multimedia) enables real-time generation and processing of video, OpenGL graphics, images, and so on. Moreover, Pd is natively designed to enable live collaboration across networks or the Internet. *vvvv* [13] is a hybrid graphical/textual programming environment for easy prototyping and development. It has a special focus on real-time video synthesis and it is designed to facilitate the handling of large media environments with physical interfaces, real-time motion graphics, audio and video that can interact with many users simultaneously. *Isadora* [6] is an interactive media presentation tool created by composer and media-artist Mark Coniglio. It mainly includes video generation, processing, and effects and is intended to support artists in developing interactive performances. In the same field of performing arts, *Eyecon* [4] aims at facilitating interactive performances and installations in which the motion of human bodies is used to trigger or control various other media, e.g., music, sounds, photos, films, lighting changes, and so on. The *Social Signal Interpretation* framework (*SSI*) [14] offers tools to record, analyze and recognize human behavior in real-time, such as gestures, mimics, head nods, and emotional speech. Following a patch-based design, pipelines are set up from autonomic components and allow the parallel and synchronized processing of sensor data from multiple input devices.

Whereas Max and Pd are especially devoted to audio processing, and vvvv to video processing, EyesWeb has a special focus on higher-level nonverbal communication, i.e., EyesWeb provides modules to automatically compute features describing the expressive, emotional, and affective content multimodal signals convey, with particular reference to full-body movement and gesture. EyesWeb also has a somewhat wider scope with respect to Isadora, which especially addresses interactive artistic performances, and to SSI, which is particularly suited for analysis of social interaction. Finally, with respect to its previous versions, the current version of EyesWeb XMI encompasses enhanced synchronization mechanisms, improved management and analysis of time-series (e.g., with novel modules for analysis of synchronization and coupling), extended scripting capabilities (e.g., a module whose behavior can be controlled through Python scripts), and a reorganization of the EyesWeb libraries including novel supported I/O devices (e.g., Kinect V2) and modules for expressive gesture processing.

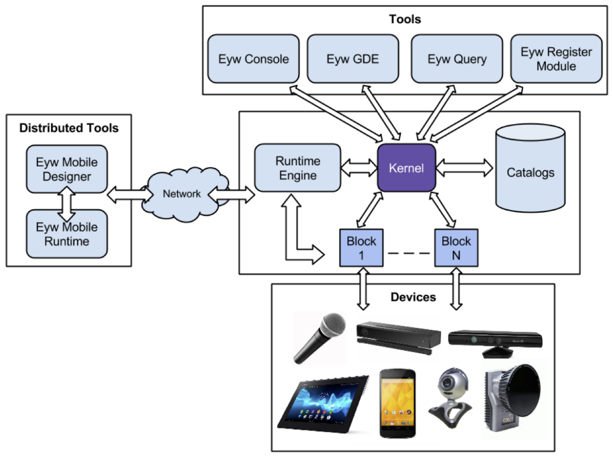

Figure 1: The overall architecture of the EyesWeb XMI platform

EyesWeb kernel, tools, libraries, and devices

Figure 1 shows the overall architecture of EyesWeb (the current version is named XMI, i.e., for eXtended Multimodal Interaction). The *EyesWeb Graphic Development Environment tool (GDE)*, shown in Figure 2, manages the interaction with the user and supports the design of applications (*patches*). An EyesWeb patch consists of interconnected modules (*blocks*). A patch can be defined as a structured

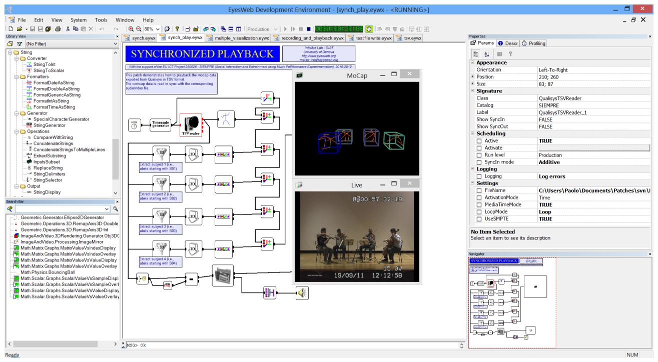
EyesWeb Development Environment - [synch_play.eywx - <RUNNING>]
SYNCHRONIZED PLAYBACK
MoCap
Live

Figure 2: A view of the EyesWeb Graphic Development Environment (GDE), with a sample patch for synchronizing motion capture, audio, and video recordings of the music performance of a string quartet.

network of blocks that channels and manipulates a digital input dataflow resulting in a desired output dataflow. Data manipulation can be either done automatically or through real-time interaction with a user. For example, to create a simple video processing patch, an EyesWeb developer would drag and drop an input video block (e.g., to capture the video stream from a webcam), a processing block (e.g., a video effect), and an output block (e.g., a video display). The developer would then connect the blocks and she may also include some interaction with the user, e.g., by adding a block that computes the energy of the user's movement and connecting it to the video effect block to control the amount of effect to be applied.

The build-up of an EyesWeb patch in many ways resembles that of an object oriented program (a network of clusters, classes, and objects). A patch could therefore be interpreted as "a small program or application". The sample patch displayed in Figure 2 was built for performing synchronized playback of multimodal data. In particular, this example displays motion capture (MoCap) data obtained from a MoCap system (Qualisys). The recorded MoCap data is synchronized with the audio and video tracks of the same music performance (a string quartet performance).

The EyesWeb kernel

The core of EyesWeb is represented by its *kernel*. It manages the execution of the patches by scheduling each block, it handles data flow, it notifies events to the user interface, and it is responsible of enumerating and organizing blocks in *catalogs* including a set of coherent *libraries*. The kernel works as a finite state machine consisting of two major states: design-time and run-time. At design-time users can design and develop their patches, which are then executed at run-time. Patches are internally represented by a graph whose nodes are the blocks and whose edges are the links between the blocks. In the transition between design and run time, the kernel performs a topological sort of the graph associated to the patch to be started and establishes the order for scheduling the execution of each block.

The EyesWeb libraries

EyesWeb libraries include modules for image and video processing, for audio processing, for mathematical operations on scalar and matrices, for string processing, for time-series analysis, and for machine learning (e.g., SVMs, clustering, neural networks, Kohonen maps, and so on). They also implement basic data structures (e.g., lists, queues, and labeled sets) and enable to connect with the operating systems (e.g., for launching processes or operating on the filesystem). Particularly relevant in EyesWeb are the libraries for real-time analysis of nonverbal full-body movement and expressive gesture of single and multiple users [2], and the libraries for the analysis of nonverbal social interaction within groups [12]. The former include modules for computing features describing human full-body movement both at a local temporal granularity (e.g., kinematics, energy, postural contraction and symmetry, smoothness, and so on) and at the level of entire movement units (e.g., directness, lightness, suddenness, impulsivity, equilibrium, fluidity, coordination, and so on). The latter implements techniques that have been employed for analyzing features of the social behavior of a group such as physical and affective entrainment and leadership. Techniques include, e.g., Recurrence Quantification Analysis [8], Event Synchronization [11], SPIKE Synchronization [7], and nonlinear asymmetric measures of interdependence between time-series.

Available Tools
Referring to Figure 1, in addition to EyesWeb GDE that was already described above, further available tools are:

- *EyesWeb Mobile*: an external tool supporting the design of Graphic User Interfaces linked to an EyesWeb patch. EyesWeb Mobile is composed of a designer and a runtime component. The former is used to design the visual layout of the interfaces; the latter runs together with an EyesWeb patch and communicates via network with the EyesWeb kernel for receiving results and controlling the patch remotely.
- *EyesWeb Console*: a tool allowing to runs patches from the command line to reduce the GDE overhead. On Windows, it runs both as a standard application or as a Windows service and additionally it can be used to run patches in Linux and OS X.
- *EyesWeb Query*: a tool to automatically generate documentation from a specific EyesWeb installation, including icon, description, input and output datatypes of each block. Documentation can be generated in latex (pdf), text, and MySql.
- *EyesWeb Register Module*: a tool allowing to add new blocks (provided in a dll) to an existing EyesWeb installation, and to use them to make and run patches. The new blocks extend the platform and may be developed and distributed by third parties.

Supported devices
EyesWeb supports a broad range of input and output devices, which are managed by a dedicated layer (*devices*) located in between the kernel and the operating system. In addition to usual computer peripherals (mouse, keyboard, joystick and game controllers, and so on), input devices include audio (from low-cost mother-board-integrated to professional audio cards), video (from low-cost webcams to professional video cameras), motion capture systems (to get with high accuracy 3D coordinates of markers in environments endowed with a fairly controlled setup), RGB-D sensors (e.g., Kinect for X-Box One, also known as Kinect V2, extracting 2D and 3D coordinates of relevant body joints, and capturing RGB image, grayscale depth image, and infrared image of the scene), Leap Motion (a sensor device capturing hand and finger movement), accelerometers (e.g., onboard of Android smarthpones and connected via network, by means of the *Mobiles to EyesWeb* app, see Figure 3; X-OSC sensors, and so on), Arduino, Nintendo Wiimote, RFID (Radio-Frequency Identification) devices (e.g., used to detect the presence of a user in a specific area), and biometric sensors (e.g., respiration, hearth rate, skin conductivity, and so on). Output includes audio, video, 2D and 3D graphics, and possible control signals to actuator devices (e.g., haptic devices, robots). Moreover, EyesWeb implements standard networking protocols such as, for example, TCP, UDP, OSC (Open Sound Control), and ActiveMQ. In such a way, it can receive and send data from/to any device, including smart objects, endowed with network communication capabilities.

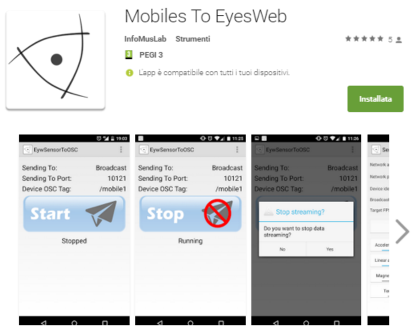

Figure 3: A view of the *Mobiles to EyesWeb* Android application on the Google PlayStore

Classes of users and interfaces

Users of EyesWeb can be ideally collected in several classes.

EyesWeb end users usually do not directly deal with the platform, but rather experience the multimodal interactive systems that are implemented using the platform. Their interface is therefore a natural interface they can operate by means, e.g., of their expressive movement and gesture to act upon multimedia content. In such a way, end users can also interact with smart devices they may wear or hold (e.g., smartphones). Smart devices can either be used to directly operate on content, or the data they collect can be presented back to end users, e.g., using data sonification [5] and visualization technologies.

EyesWeb developers make applications (patches) using the EyesWeb GDE and its visual programming language. This implements all the basic constructs of programming languages such as sequences (a chain of interconnected blocks), conditional instructions (e.g., by means of switch blocks that direct the flow of data to a specific part of a patch when a given condition is matched), iterative instructions (by means of a specific mechanism that allows to execute a given sub-patch repetitively), and subprograms (implemented as sub-patches). Because of the visual programming paradigm, EyesWeb developers do not need to be computer scientists or expert programmers. In our experience, EyesWeb patches were developed by artists, technological staff of artists (e.g., sound technicians), designers, content creators, students in performing arts and digital humanities, and so on. Still, some skills in usage of computer tools and, especially, in algorithmic thinking are required. EyesWeb developers can exploit EyesWeb as a tool for fast-prototyping of applications for smart objects: EyesWeb can receive data from such objects by means of its input devices and the task of the developer is to design and implement in a patch the control flow of the application.

Supervisors of EyesWeb patches are users who supervise and control the execution of an EyesWeb application. They can both set parameters to customize the execution of the patch before running it and act upon the patch (e.g., by changing the value of some parameters) while the patch is running. Consider, for example, a teacher who customizes an educational serious game for her pupils and operates on the game as a child is playing with it, or a therapist who sets e.g., target and difficulty level of an exergame for a patient. Supervisors of EyesWeb patches do not need any particular skill in computer programming and indeed they are usually a special kind of end user, with a specific expertise in the given application area. The EyesWeb Mobile tool allows for endowing them with traditional Graphical User Interfaces they can use for their task of customizing and controlling the execution of a patch. In such a way, they do not need to go into the details of the EyesWeb GDE and they work with an interaction paradigm, which is more familiar to them. Moreover, the patch is prevented from possible un-

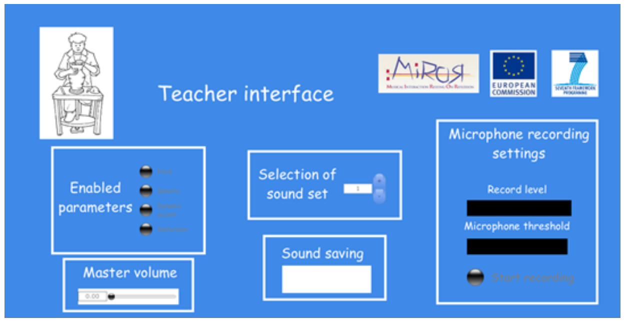

Figure 4: An EyesWeb Mobile interface developed for enabling a teacher to customize a serious game for children.

wanted modifications. The EyesWeb Mobile interfaces can also work on mobile devices (e.g., tablets) to facilitate operations when the operator cannot stay at the computer (e.g., a therapist who needs to participate in the therapy session). This feature makes EyesWeb Mobile a tool that also suits remote configuration of smart objects applications. Figure 4 shows an EyesWeb Mobile interface developed for enabling a teacher to customize a serious game for children.

EyesWeb programmers develop new software modules for the EyesWeb platform. They need to be skilled C++ programmers and are endowed with the *EyesWeb SDK*, which enables them to extend the platform with third-parties modules. In particular, the EyesWeb SDK enables including in the platform new modules for interfacing possible smart objects that e.g., do not communicate through standard networking protocols or are not supported by the platform yet.

Conclusion

EyesWeb is nowadays employed by thousands of users spanning over several application domains. It was adopted in both research and industrial projects by research centers, universities, and companies. A one-week tutorial, *the EyesWeb Week* is organized every two years at our research center. In our experience at Casa Paganini - InfoMus, EyesWeb was used in research projects that combine scientific research in information and communications technology (ICT) with artistic and humanistic research [3]. In this context, the platform provided the technological ground for artistic performances (e.g., in *Allegoria dell'opinione verbale* by R. Doati, *Medea* by A. Guarnieri, and *Invisible Line* by A. Cera, to name just some of them), for multimodal interactive systems supporting new ways of experiencing art and cultural heritage (e.g., *Museum of Bali* in Fano, Italy; *Museum of La Roche d'Oëtre* in Normandy, France; *Enrico Caruso Museum* near Florence, Italy), for serious games in educational settings (e.g., *The Potter* and *BeSound*), and for exergames for therapy and rehabilitation (e.g., in an ongoing collaboration with children hospital Giannina Gaslini in Genova, Italy). EyesWeb is being currently improved and extended in the framework of the EU-H2020-ICT DANCE Project, investigating how sound and music can express, represent, and analyze the affective and relational qualities of body movement.

The need of supporting such a broad range of application domains required to make a trade-off between implementing general-purpose mechanisms and exploiting domain-specific knowledge. For example, on the one hand, support to generality sometimes encompasses a somewhat reduced learnability and usability of the platform by lowly-skilled EyesWeb developers due to the increased complexity of the implemented mechanisms. On the other hand, some EyesWeb modules developed on purpose for specific application scenarios have a limited scope and are difficult to reuse. Future directions and open challenges include increased cross-platform interoperability and a tighter integration with cloud services and storage technologies.

Acknowledgements
This research has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 645553 (H2020-ICT Project DANCE, http://dance.dibris.unige.it). DANCE investigates how affective and relational qualities of body movement can be expressed, represented, and analyzed by the auditory channel.

REFERENCES

1. Antonio Camurri, Shuji Hashimoto, Matteo Ricchetti, Andrea Ricci, Kenji Suzuki, Riccardo Trocca, and Gualtiero Volpe. 2000. EyesWeb: Toward Gesture and Affect Recognition in Interactive Dance and Music Systems. *Computer Music Journal* 24, 1 (April 2000), 57–69.
2. Antonio Camurri, Barbara Mazzarino, and Gualtiero Volpe. 2004. Analysis of expressive gesture: The EyesWeb expressive gesture processing library. In *Gesture-based communication in human-computer interaction*. Springer, 460–467.
3. Antonio Camurri and Gualtiero Volpe. 2016. The Intersection of Art and Technology. *IEEE MultiMedia* 23, 1 (Jan 2016), 10–17.
4. Eyecon. 2008. `http://eyecon.palindrome.de/`. (2008). Accessed: 2016-03-25.
5. Thomas Hermann. 2008. Taxonomy and Definitions for Sonification and Auditory Display. In *Proceedings of the 14th International Conference on Auditory Display (ICAD 2008)*, Patrick Susini and Olivier Warusfel (Eds.). IRCAM.
6. Isadora. 2002. `http://troikatronix.com/`. (2002). Accessed: 2016-03-25.
7. Thomas Kreuz, Daniel Chicharro, Conor Houghton, Ralph G Andrzejak, and Florian Mormann. 2012. Monitoring spike train synchrony. *Journal of Neurophysiology* (2012).
8. Norbert Marwan, M. Carmen Romano, Marco Thiel, and Jürgen Kurths. 2007. Recurrence plots for the analysis of complex systems. *Physics Reports* 438, 5–6 (2007), 237–329.
9. Max. 1988. `http://cycling74.com/products/max/`. (1988). Accessed: 2016-03-25.
10. Miller Puckette. 1996. Pure Data. In *Proceedings of the International Computer Music Conference*. 224–227.
11. Rodrigo Quian Quiroga, Thomas Kreuz, and Peter Grassberger. 2002. Event synchronization: A simple and fast method to measure synchronicity and time delay patterns. *Physical Review E* 66, 4 (Oct 2002), 041904.
12. Giovanna Varni, Gualtiero Volpe, and Antonio Camurri. 2010. A system for real-time multimodal analysis of nonverbal affective social interaction in user-centric media. *IEEE Transactions on Multimedia* 12, 6 (2010), 576–590.
13. vvvv. 1998. `http://vvvv.org/`. (1998). Accessed: 2016-03-25.
14. Johannes Wagner, Florian Lingenfelser, Tobias Baur, Ionut Damian, Felix Kistler, and Elisabeth André. 2013. The Social Signal Interpretation (SSI) Framework: Multimodal Signal Processing and Recognition in Real-time. In *Proceedings of the 21st ACM International Conference on Multimedia (MM '13)*. ACM, New York, NY, USA, 831–834.