UDC 681.5

Stephan Bergemann, M. Sc.,
Juergen Sieck, Prof. Dr.,
University of Applied Sciences, HTW-Berlin

PROTOCOL AND SYSTEM DESIGN FOR INTERACTIVE MEDIA FACADES

С. Бергеманн, Ю. Зік. **Розробка системи і протоколу для інтерактивних фасадів.** В останні десятиліття все більша кількість медіафасадів розробляється і застосовується в будівлях по всьому світу. Вони часто є тимчасовими і загалом виконують абсолютно пасивну роль, лише відтворюючи відео, фотографії та інший оплачений контент. В даний час чисельні науково-дослідні установи займаються інтерактивним застосуванням медіафасадів. Ця стаття надає різні варіанти і можливості того, як зробити медіафасади інтерактивно застосовними.

Ключові слова: медіафасад, розробка системи і протоколу.

С. Бергеманн, Ю. Зик. **Разработка системы и протокола для интерактивных фасадов.** В последние десятилетия все большее количество медиафасадов разрабатываются и применяются в зданиях по всему миру. Они часто являются временными и в основном выполняют абсолютно пассивную роль, всего лишь воспроизводя видео, фотографии и другой оплаченный контент. В настоящее время многие научно-исследовательские учреждения занимаются интерактивным применением медиафасадов. Эта статья предоставляет различные варианты и возможности того, как сделать медиафасады интерактивно применимыми.

Ключевые слова: медиафасад, протокол и конструкция системы.

S. Bergemann, J. Sieck. **Protocol and system design for interactive media facades.** In the past decades more and more media façades have been designed and applied to buildings all around the world. These are often temporary and mostly pure passive only playing back videos, pictures and other prerendered content. Interactive use cases for media façades are currently focused by many research institutions. This paper exposes different options and possibilities to make media façades interactively usable.

Keywords: media facades, protocol and system design.

Introduction. In the past few years, media façades received an increasing popularity. According to Häusler, they can be categorised into two main categories: Mechanical and electrical facades [1]. Further subdivision can be applied regarding to their types of displaying content. Electrical façades are either projection, illumination, or display façades or a combination of these techniques. Additionally, a subcategorisation can be done by projection, illumination or displaying technologies used.

Besides the technology, the content type to be displayed has also to be taken into consideration: Static images, moving images (videos) or realtime generated images on application runtime. Images as advertisements and other posters already had been used since a long time and also decorate façades. They are made up by smaller parts of the image that together form a bigger image, which might even cover a whole façade. Video installations are conventional and have also been used to transform façades into multimedia displays since the emergence of outdoor cinema.

Interactive media responds to a user's action and therefore its content is no longer static, but dynamic. For many temporary installations, such as fairs and exhibitions, but also on festivals, interactive multimedia installations using façades as projection/display area are increasingly used.

1. Related Work. Several installations have been described in the past allowing users to interact with media façades. In the *Tower of Winds* in Japan environment parameters like wind speed, direction and noise level are sampled and used to illuminate the tower accordingly [1].

Given the 20th anniversary of the "Chaos Computer Club" foundation, the *Haus des Lehrers* building in Berlin has been transformed into a monochrome dot-matrix display that could be remotely controlled by cell phones either sending text messages or calling a provided number and then pressing

number keys for interaction [3]. Also using text messages but in combination with a slighshot device, the *Spread.Gun* [4] and the *SMSlingshot* [5] projects were realised to virtually shoot messages at a projection façade. The *ARS Electronica Center* in Linz created an application to control their installed neon tube based media façade [6]. Using a smartphone and touch gestures useres were able to draw on the building. This was realised using camera tracking in the smartphone an mapping the touch events on the building. All these projects used different interaction methods and protocols for data interaction. These examples show the usage of a wide range of input data and methods. There exist message based systems such as *Spread.Gun* and *SMSlingshot* as well as button based input at the *Haus des Lehrers* and finally live video streaming and touch gesture based applications like the drawing application for the *Ars Electronica Center*.

As of the characteristics of buildings and the diversity of display technologies for buildings, it is not always possible to cover the whole façades as a single projection area by one output device. In general multiple output devices have to be combined to cover the whole façade where each one provides a part of the whole image to be displayed (segmentation). For images and even more videos, synchronisation is a crucial requirement where its segments have to be played back at the same time. This can be accomplished with different approaches: Real time systems or synchronisation of timestamps via a network are possible solutions to this problem.

Serveral developments have been accomplished for real time graphics rendering via a network. Transmitting OpenGL library calls encoded via a network, *WireGL* [7], *Chromium* [8], *BroadcastGL* [9] and *ClusterGL2* [10] demonstrated possible solutions. In these implementations the application runs on a single machine, their calls to the graphics library (OpenGL in this case) get caught and a code is transmitted via network to distributed machines running renderers that transform these codes back to library calls and execute them. While the first systems are not under active development, the last solution (*ClusterGL2*) still receives updates and therefore seems to be the best choice. One main advantage of this approach of having only one application instance running is that any interaction data that is influencing the rendering process is implicitly the same on all rendering machines.

Other systems like *Quartz Composer*[1] take a different approach: Each system runs the same application and a master node keeps the all running instances of the application synchronic. This implies that all running application instances have to receive any interaction data input to respond accordingly on their displayed part. In the example of *Quartz Composer* however there is not implemented such a mechanism – the developer has to implement it himself.

2. Motivation. Most of the above mentioned installations where temporary and only designed for their specific context. Each façade only ran that single specific application. This fits perfectly for many temporary installations, but on a long-term basis, a façade may offer different methods of interaction for different kinds of applications. No common standards or best practise examples are yet given how different applications with various interaction interfaces may be covered by one media façade.

Each of those applications used only one specific form of interaction and therefore only one type of interaction data that had to be handled and transmitted. However, this approach may not be suitable for every other imaginable application that could run on that façade.

At the *Forschungs- und Weiterbildungszentrum Kultur und Informatik* (FKI) located at Wilhelminenhof campus of HTW Berlin (University of Applied Sciences), which is currently under construction, such a media façade will be installed besides a photovoltaic system. This media façade — schematically depicted in Figure 1 — is not yet completely built, but it will not be temporary and not designed displaying only a single application. It was designed for students at the university to develop their applications and get in contact with interactive media façade development. One special feature of this media facade is that it does not consist of one single screen area, but combines different technologies to gain a bigger area: The different areas are inhomogenous in regard to resolution, pixel density, and presentation capabilities. An area equipped with LED modules allows low resolution

[1] https://developer.apple.com/technologies/mac/graphics-and-animation.html

rendering. The whole window width of the first and second floor at the top border of the LED panel is equipped with rear projection allowing high resolution projection at many light conditions. Surrounding LED light bars frame the installation with ambient light.

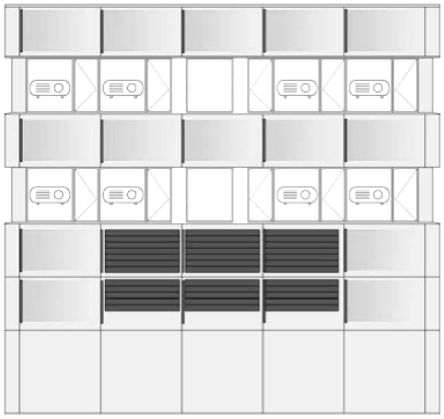

Fig. 1. Combination of different display technologies used at the media façade of the FKI

As the facade will not only display prerendered content, but also live and interactive media, a *ClusterGL2* based system has been implemented to solve synchronization, scaling, and transition challenges.

On that basis, this paper focuses on the approach to implement an easy to use interface and protocol that is suitable for many different applications and with it forms of interaction.

3. Concept for a generalized interface. Different types of interaction interfaces usually contain different kinds of sensors. Gestures may be recorded by other sensors than audio or video signals. However, after its processing both can be used to interact with applications. When designing a generalised interface, all these stream-based (audio and video) and event-based (any form of control data like information about pressed or released buttons or touch events of connected devices) data types must be taken into consideration.

Also the ongoing growth of mobile networks using GPRS, EGPRS and 3G has to be taken into account as well as the increasing popularity of smartphones and devices. Therefore, a network based solution seems reasonable.

MIDI (Musical Instrument Data Interface), DMX, OSC[2] (Open Sound Control) and RTP (Real-Time Transport Protocol) are some existing interfaces and protocols, which have been investigated during our research.

OSC was implemented to solve common problems (speed, extendability and accuracy) of MIDI and extending its area of applications. OSC is not exactly a protocol, but a data format specification. It does not enforce the usage of a specific hardware interface. However, most implementations (e.g. WOscLib[3], Oscpack[4], and liblo[5]) are UDP and thereby network based. Using human readable addresses for parameters in form of URIs (e.g. /pong/background/colour) and extended set of combinable basic data types (e.g. int, string, float), it provides powerful, easy to control, and – for applications – easy to implement features to handle different types of event-based data. When used in network environment, OSC works with the client-server principle.

RTP[6] is an extensible protocol for live conferencing tools and situations using UDP as underlying protocol layer. It provides several predefined data types for audio and video streaming data (codecs) as

[2] http://opensoundcontrol.org/
[3] http://wosclib.sourceforge.net/
[4] http://code.google.com/p/oscpack/
[5] http://liblo.sourceforge.net/
[6] http://tools.ietf.org/html/rfc3550

well as offers mechanisms to deal with limits in bandwidth, delays in transmission, and synchronisation of event sources and sinks. Therefore it is ideal for transmitting stream based data. It allows multipoint connections, but may not be mistaken with the P2P principle as with monitor, mixer, and translator it defines some more mechanisms that are not common in the P2P principle.

Our goal is to combine OSC and RTP by defining a new data type encoding OSC messages. These messages are to be sent it in another stream besides possibly video and audio data streams. This provides a maximum in flexibility concerning data types and therefore methods of transmitting interaction data. It would be possible to transmit video streams, audio streams as well as touch events, message strings, and various other representations of interaction data.

4. Implementing OSC via RTP. Several open source libraries were extended to support the combination of OSC and RTP. The open source liblo library (lightweight OSC implementation) was used for OSC support. It provides functionalities to encode, decode, transmit, and dispatch OSC messages. With user-defined callback methods it is easily possible to extend existing or implement new interactive applications with OSC.

The EMIPLIB[7] library, base on the JRTPLIB[8], was used for RTP messages. Both libraries are open source and EMIPLIB extends the basic JRTPLIB by several codecs for video and audio streams. While JRTPLIB deals with all RTP traffic, synchronisation, source management and handling of the different streams, EMIPLIB is responsible for handling the different data types sent via RTP. As a result, we extended EMIPLIB to also deal with OSC streams.

Another focus of our implementation was to keep the OSC functionality besides the option of using OSC via RTP. This design decision was made with respect to embedded devices that might not be capable of implementing the whole RTP stack, but however might implement OSC. Therefore a RTP server always runs besides an OSC server. Both servers dispatch received OSC messages the same way. The OSC messages received via RTP are decoded and handed over to liblo to be dispatched and the incoming OSC messages via the OSC server are directly dispatched by liblo.

5. Test. As a proof of concept three applications where implemented:

— a Pong game using only OSC and some self constructed embedded controllers as input devices — each with a fader to control the paddle and sending the interaction data to the application;

— a painting application using OSC via RTP and sole OSC where multiple users can draw on the façade in selectable colors;

— a conferencing application displaying incoming video streams using OSC via RTP for positioning each incoming video on the screen, varying its width, height and opacity and RTP for streaming video data that has been captured by webcams on connected mobile devices.

All applications were using the same library and the proposed combination of OSC and RTP met all needs.

As loosing messages in OSC may be a significant flaw we also tested the reliability of our interface by implementing two test scenarios to evaluate if mechanisms for error correction would be needed:

a) using a laptop connected to Wifi network that is connected to the application machine via Ethernet;

b) using a laptop connected via Wifi to the Internet (1 Mbit/s upstream, 5 Mbit/s downstream) and the application machine being connected to the Internet via Ethernet.

We then sent 100 times 10000 random OSC messages (approx. 20 Bytes each) via RTP using several different delays between each messages to test different message rates.

[7] http://research.edm.uhasselt.be/emiplib/emiplib.html

[8] http://research.edm.uhasselt.be/~jori/page/index.php?n=CS.Jrtplib

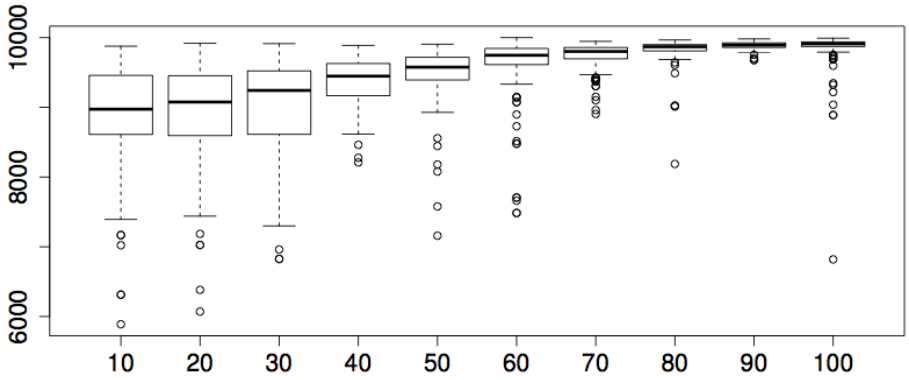

Fig. 2. OSC message loss on application machine in scenario a)
(x: delay between each message in µs; y: Number of received OSC messages from 10000)

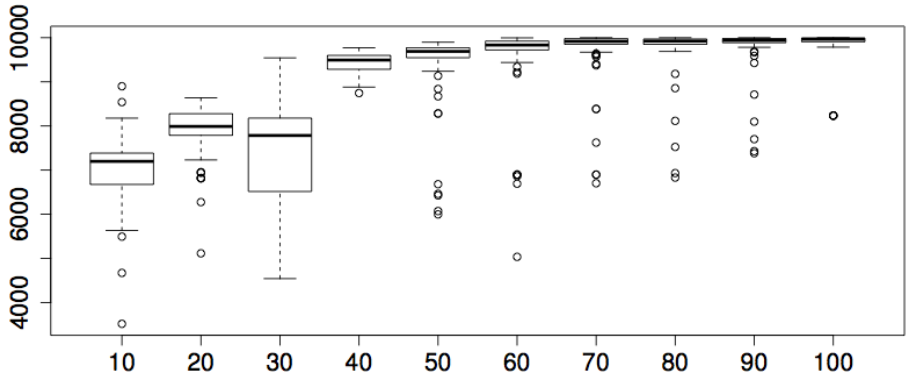

Fig. 3. OSC message loss on application machine in scenario b)
(x: delay between each message in µs; y: Number of received OSC messages from 10000)

As our tests have shown, the mean OSC message loss is below 2 % with a delay of 80 µs and more. No further error correction mechanisms were implemented as they would have increased the traffic drastically than the expected benefit would have justified.

Conclusion and future work. Media façades provide a way to transport art, information and multimedia content. Even interactive installations can be integrated right into the façade using plenty of different input devices and provide enhanced possibilities to turn a regular building into a memorisable one. However, current research only has investigated in different single application approaches that are usually implemented only for one specific purpose, e.g. festival or trade fair, and that are usually temporary installations.

Using existing open source libraries and protocols OSC and RTP as well as combining them, we developed a generalised approach for transmitting interaction data to media façades. The provided solution is both reliable and extendable. It provides all options to implement the same functionality as in other existing applications, but uses the same protocols and interfaces for each application.

With the proposed combination of OSC and RTP it is possible to have a series of very different applications running on a media façade where each may be controlled by the same input devices or enable different input devices to control one parameter. Also combining OSC and RTP it is possible to use one Interface for various input data types. Tests have shown that the newly added option of sending OSC messages via RTP works well via local area networks but also via the Internet. As RTP was designed for conference systems, it is also possible to implement collaborative applications.

The provided solution is also applicable for other use cases: For example, multimedia applications might implement live collaboration between multiple users in different locations using

OSC via RTP.

Our future work besides implementing a wide range of different applications exploiting this new generalised interface for our façade will include the development of a web-based platform to upload applications, specify their OSC addresses and RTP capabilities to provide this information to users that want to use the applications on the façade with their own controllers.

Acknowledgment. This paper describes the work undertaken in the context of the project SIGNAL hosted by the research group Information and Communication Systems INKA that is generously funded by the European Regional Development Fund (ERDF).

References

1. Hank, M. Häusler. Media façades: history, technology, content. avedition, [Ludwigsburg, Germany], 2009. isbn: 9783899861075 3899861078.
2. Bergemann, S. Medienfassade — Möglichkeiten von Visualisierung und Interaktion / S. Bergemann, R. Schlegel. — 2012. — http://inka.htw-berlin.de/Sieck/Stud_Projekte/bergemann_schlegel_fp2.pdf.
3. Blinkenlights.net / Project Blinkenlights. url: http://blinkenlights.net. — 10.12.2012.
4. Fischer, P.T. VR/Urban: Spread.gun-design process and challenges in developing a shared encounter for media façades / P.T. Fischer, C. Zöllner, E. Hornecker // In: Proceedings of the 24th BCS Interaction Specialist Group Conference. — BCS '10. — Dundee, United Kingdom: British Computer Society. — 2010. — P. 289 — 298. — isbn: 978-1-78017-130-2.
5. Fischer, P.T. Urban HCI: spatial aspects in the design of shared encounters for media façades. In: Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems / P.T. Fischer, E. Hornecker // CHI '12. — Austin, Texas, USA.
6. Boring, S. Multi-user interaction on media façades through live video on mobile devices / S. Boring // In: Proceedings of the 2011 annual conference on Human factors in computing systems. – CHI '11. – Vancouver, BC, Canada. — ACM. — 2011.
7. Humphreys, G. WireGL: A Scalable Graphics System for Clusters / G. Humphreys, M. Eldridge, I. Buck, G. Stoll, M. Everett, P. Hanrahan // SIGGRAPH. — 2001.
8. Humphreys, G. Chromium: A Stream-Processing Framework for Interactive Rendering on Clusters / G. Humphreys, M. Houston, R. Ng, R. Frank, S. Ahern, P.D. Kirchner // SIGGRAPH. — 2002.
9. Ilmonen, T. Broadcast GL: An Alternative Method for Distributing OpenGL API Calls to Multiple Rendering Slaves / T. Ilmonen, M. Reunanen, P. Kontio // The Journal of WSCG. — 2005.
10. Neal, B. Distributed OpenGL Rendering in Network Bandwidth Constrained Environments / B. Neal, P. Hunkin, A. McGregor. — 2011.

Reviewer Ph. D., prof. Odessa nat. polytechnic univ. Brovkov V.G.

Received December 15, 2012.