RWTH AACHEN UNIVERSITY

iWall: An Interactive Urban Display for Collaborative Mobile Applications

Bachelor's Thesis at the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

by
Simon Lukas Jakubowski

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr. Klaus Wehrle

Registration date: 19.11.2013
Submission date: 19.03.2014

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed, and that I have marked any citations accordingly.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Aachen, March 18th, 2014
Simon Lukas Jakubowski

Contents

List of Figures

Abstract

The iWall is a Media Facade installed at the historic computer science building of the RWTH Aachen University. With a resolution of 1920 pixels in width and only 10 pixels in height it will challenge designers and developers to create applications for this type of public display. To enable these people to access the iWall with their applications an easy to use software is needed. This software has to enable distributed access to the iWall by multiple applications on the one hand and the option to interact with them on the other.

In this Bachelor thesis I describe the developing process from the first prototype up to the iWall Collaboration software. Finally this allows developers to use a single framework to publish graphic content to the iWall. With a second framework it is also possible to control the written applications by one centralized mobile client.

Überblick

Die iWall ist eine “Media Facade” (Fassade für Medienanwendungen), installiert am denkmalgeschützten Gebäude des Informatikzentrums der RWTH Aachen. Mit einer Auflösung von 1920 Pixeln in der Breite und nur 10 Pixeln in der Höhe wird sie Designer wie auch Software-Entwickler herausfordern, für diese Art von öffentlichem Bildschirm Anwendungen zu schreiben. Um den Zugriff auf die iWall zu ermöglichen, benötigt es entsprechend einfach zu benutzende Software. Diese muss den verteilten Zugriff auf die iWall durch verschiedene Anwendungen erlauben, wie auch ermöglichen mit ihnen zu interagieren.

In dieser Bachelorarbeit beschreibe ich den Weg vom ersten Prototypen bis hin zur iWall Collaboration Software. Diese ermöglicht es Entwicklern letztendlich ein Framework zu verwenden um Grafikinhalte auf einen Teil der iWall zu bringen. Mit einem weiteren Framework ist es zudem möglich, die entwickelten Programme durch eine zentrale mobile Endanwendung steuern zu lassen.

Acknowledgements

First of all I like to thank Prof. Jan Borchers, who has the idea of creating the iWall and gives me the opportunity to write my Bachelor thesis about using the iWall for collaborative mobile applications. Then I like to thank all people working on the iWall project, especially Dr. Thorsten Karrer, who made every endeavor possible to bring forward the project and helps me to implement the iWall Video Player and the iWall Simulator. I also like to thank for his valuable input on this Bachelor thesis.

Last but not least I like to thank my family for supporting me during writing this thesis.

Conventions

Throughout this thesis I use the following conventions.

Text conventions

Definitions of technical terms or short excursus are set off in coloured boxes.

EXCURSUS:
Excursus are detailed discussions of a particular point in a book, usually in an appendix, or digressions in a written text.

Definition:
Excursus

Source code and implementation symbols are written in typewriter-style text.

`myClass`

The whole thesis is written in American English. Measurements are given in metric system. Floor numbering after American convention.

Chapter 1

Introduction

Large Public Displays in Aachen.

Large Public Displays are present in most urban areas displaying almost any kind of content to inform or entertain passing by users. In the city of Aachen you will find such displays e.g., at frequent bus stops (figure 1.1) to inform about delays or as advertisement boards. The number of these displays is growing and therefore challenging the developers to bring users to pay attention to their display. Making a display interactive seems like the most important step to achieve a higher commitment from the target audience. In the Aachen main hall two exhibits are using large public interactive displays. One is the "Karlspreis Stehlen" where you get information about each of the more than 64 laureate winning, the others are "Karlspreis" and the "Aachener Frieden", where visitor can interact with wooden blocks to get information about persons involved in the "Aachener Frieden".

Media Facades in Aachen.

If a Public Display is integrated into the facade of a building it is called a Media Facade. At the top of a large building, near the Aachen main station a weather measurement display (figure 1.2) can be found. Depending on the upcoming weather conditions the light elements are blinking or changing its color. The meaning of the blinking lights is not directly obvious and visitors has to search for these information at e.g., internet[1] or tourist information. Here it is

[1] http://www.stawag.de/unternehmen/engagement/wetter/

Figure 1.1: Time schedule of arriving buses at the Aachen main station.

hard for the visitors to distinguish between a facade, where lights are integrated because of artistic design like "Super C" or the "Toaster" (figure 1.3) and real Media Facades that are acting as an information display.

The iWall is the first interactive Media Facade in Aachen.

With the iWall the first interactive Media Facade appears in Aachen. This Media Facade is integrated into computer science building of the RWTH Aachen University. Inside each of 10 sun shades of the second floor, 1920 separate controllable lights are installed and emit their light to the lower blade. This creates a theoretical horizontal resolution of 1920 pixels and 10 vertical pixels.

Expecting computer science students as the main user group for the iWall.

Visitors of the building and potential users for the iWall will be students of the RWTH Aachen. Thus, it would be great if applications on the iWall help this group in their every day life by, e.g., displaying bus schedule or the Mensa menu. Making the iWall interactive by using the mobile phones of the students and all other users is one important key to getting them pay attention to the iWall. Another thing I am doing here is to allow writing own applications and therefore displaying own content on the wall to achieve a higher commitment to the iWall project. Because of the special form-factor with a hight of only 10 pixels, the iWall

Figure 1.2: Weather measurement display at the top of the left building near the Aachen main station.

Figure 1.3: "Super C" illuminated blue in the front and the "Toaster" with an orange illumination behind it.

Figure 1.4: The iWall is installed at the third floor inside the sun blades of the historical computer science building.

challenges possible developers and artists and expects their solutions.

It should be able to run multiple applications on the iWall.

For this, I need to provide users' a framework for creating applications running on the mobile devices on the one hand and creating applications displaying content on the iWall on the other. The mobile client should run on most mobile devices of the user and all applications running on the iWall should be accessible by one mobile client. Sharing the screen of the iWall will allow to run multiple applications at the same time. I must consider that not all application are always visible on the iWall. An application like the bus schedule would only need access to the iWall at a specific point in time for a fixed duration. This creates gaps on the iWall Display, and I have to think about a solution to solve this. To avoid disturbance of the audience, the iWall should not move content on the wall too often or show unexpected behavior.

Chapter 2

Related work

In this chapter I give an overview about related work and classify the iWall project into the research field of Large Public Display and Media Facade taxonomies. I identify problems that may occur with the iWall based on findings of current research. Finally I present possible interaction techniques and explain why a mobile client to enable interaction on the iWall is the best choice.

2.1 Large Public Displays

Müller et al. [2010] present a taxonomy of interactive Large Public Displays grouped by the four mental models

- Poster, where information is displayed to the audience like on a poster, but with the possibility to change the content dynamically by interaction,
- Windows, where the audience look into another location and enable interaction with other people or things at this location,
- Mirror, where the real world or the audience in front of the Large Public Display are mirrored to it and enable virtual reality interaction inside the mirrored content and

- Overlay, where the shape of the environment, the Large Public Display is projected on, is included into the displayed content and interaction of the audience,

and the interaction models

- Presence,
- Body Position,
- Body Posture,
- Facial Expression,
- Gaze,
- Speech,
- Gesture,
- Remote Control,
- Keys and
- Touch.

Only the Remote Control interaction model is suitable for the iWall.

Classifying the iWall into this taxonomy varies for the mental model by the content displayed. As interaction model only remote control is suitable here. Presence, Body Position, Body Posture, Facial Expression and Gaze as interaction models would need tracking the audience with a camera. This would be difficult to achieve, because of the bad lightning conditions when the iWall is running and the large area where people can see the iWall and may interact with it. For the same reason it would be hard to find a good spot to install an interface for the Speech, Keys and Touch interaction models.

2.1.1 Interacting with Large Public Displays

Michelis and Müller [2011] identifies the six interaction phases:

1. Passing by,
2. Viewing and Reacting,
3. Subtile Interaction,
4. Direct Interaction,
5. Multiple Interaction and
6. Follow up Actions

for Large Public Displays. Between each interaction phase exists a threshold. This shows how difficult it is to excites users to get attention from them and finally to get the motivation to interact with an installation like the iWall.

Social embarrassment is a major problem.

Brignull and Rogers [2003] identifies social embarrassment as a major problem with interaction on Large Public Displays. The possible proposed solutions are to lower the threshold to participate and creating a honey pot effect, if other people using the system. As the interaction model is fixed for the iWall, I will lower the threshold by running the mobile client without installing additional software on the smartphones of the visitors. With the iWall Collaboration Framework and the chance to create an own iWall App I expect the honey pot effect mentioned.

2.1.2 Designing Large Public Displays

People pay attention to public displays only for a few seconds.

Huang et al. [2008] point out, based on their findings, five recommendations to design a Large Public Display. The first finding called "Brevity of glance" says people are looking only for a few seconds at a display to perceive the needed information. The recommendation here is to avoid long sentences and too much information. For the iWall the textual information is limited by design. Another finding is called "Content format and dynamics" and the recommendations for this are to avoid abrupt content change and make the content dynamic, while giving the user the option to change the content by interaction. I will apply these recommendations while implementing iWall Collaboration. Three more findings on the form factor, size and

position of the display are mentioned but they can not be applied to the iWall.

SEMI-PUBLIC DISPLAYS:
In the field of HCI research the term "Large Public Display" can also be found as describing a shared display inside a workspace for collaboration like in i-Land by Streitz et al. [1999] or Liveboard by Elrod et al. [1992]. In some publications e.g. by Huang and Mynatt [2003] these displays are called more specific as semi-public displays, because these displays are not apparent to people outside of the (dedicated) workspace.

2.2 Media Facades

A Media Facade is a Larger Public Display integrated into the facade of a building.

Dalsgaard and Halskov [2010] define Media Facades as Large Public Displays integrated into buildings or street furniture. Based on an overview of some Media Facades the authors point out the following eight challenges for urban Media Facades:

1. New interface,
2. Integration into physical structures and surroundings,
3. Increased demand for robustness and stability,
4. Developing content to suit the medium,
5. Aligning stakeholders and balancing interests,
6. Diversity of situations,
7. Transforming social relations and
8. Emerging and unforeseen use of places and systems.

After the definition of the authors also the iWall is a Media Facade and therefore the eight challenges should apply also to the iWall. For the software implementation of the iWall I identify in particular the challenge four and seven as the most important. I will approach the challenge four later on by e.g. evaluate fonts suitable for the iWall and challenge

seven by enable other computer science students to allow writing their own Apps for the iWall.

The number of Media Facades are growing continuously. On the website of the Media Architecture Institute[1] appears a new report about a new interactive Media Facade around every week. Mignonneau and Sommerer [2008] represent an overview about many successful interactive Media Facades and concludes their benefits for the public, designers and developers.

2.2.1 iRiS

Create better interaction for Media Facades by prototyping.

Wiethoff and Gehring [2012] explained the development process of an interaction system for a Media Facade called iRiS. They conclude that designing interfaces for Media Facades differ from designing regular graphical user interfaces. Focus on a user centered design process and working with prototypes to get more design cycles and a better result is the advice by the authors. Prototypes for the iWall can be tested on the iWall Simulator.

2.3 Interaction with Large Public Displays

2.3.1 Touch Projector

Interacting through a live video stream of the smartphone.

Boring et al. [2011] propose using Touch Projector as a multi-user interaction on Media Facades. Based on a live video stream of their smartphones the users can interact with the content displayed on a facade. With an overlay (above the video on the smartphone display) individual content and user interface controls can be added. A dedicated server is processing the video stream and touch input of the mobile devices connected through wireless LAN.

For the test setup the authors use the facade of the ARS Electronic Center with 1087 uniquely illuminated windows

[1] http://www.mediaarchitecture.org/

and gives out three Apple iPhones to interested users. In this test setup it looks like they only use the front facade of the main building and there only the inner part of it, because they need to display a white border for the image processing. This limits the pixel resolution to 8 x 16 pixels in this case.

The authors mentioned, to detect the frame in the video live stream of the smartphone, they need a white frame on the Media Facade. For the iWall this would lower the resolution to only eight pixels in height. As the video stream is processed on the server, it would limit the number of parallel interacting users dramatically, due to bandwidth limitation.

2.3.2 Sweep and Point & Shoot

Using a mouse pointer on Large Public Displays.

Ballagas et al. [2005] propose using two techniques to interact with mobile phones on Large Public Displays. The "Sweep" technique tracks the relative movement of a mobil phone and map the position to a mouse pointer on the screen. For the "Point & Shoot" technique QR-Codes are overplayed about the content on the public display and the user is selecting an object on the screen by taking a picture of the nearest QR-Code with the mobile phone.

For the iWall the Sweep and Point gesture would be not practicable, because a user may fight for only one mouse cursor with others or would not identify her cursor displayed on the iWall. For Point & Shoot the resolution of the iWall is not high enough.

2.3.3 Urban Pixels

Flashlights as input devices.

Urban Pixels by Seitinger et al. [2009] are LED modules connected by a wireless network and can be placed on a building to create a Media Facade. For these Urban Pixels the authors also present three possible interaction techniques. The first is to use a stationary PC near the location

of the Media Facade to control the LED modules. Another interaction technique they present is to send a SMS with a mobile phone for interaction. And as an IR sensor is included in every Urban Pixel the LED modules can also controlled by a flashlight.

As in 2.1 discussed a stationery input is difficult to install for the iWall. Sending a SMS to the iWall is an option to enable also user without a smartphone but a mobile phone to interact with the iWall. I am looking forward to implement an example App for iWallCollaboration to make this interaction model possible. Interaction with a flashlight is difficult at the iWall, because of the high installation point above the ground, but may possible with bright flashlights if IR detection is installed at the facade.

2.3.4 Spread.gun and SMSlingShot

Use a sling shot to fire messages to a Media Facade

Fischer and Hornecker [2012] implement and evaluate the Spread.gun and SMSlingShot as an interaction with Media Facades. Both devices can input messages by the user and if finished the result can fired or sling shoot to the Media Facade. As in 2.1 discussed a stationary installation is difficult for the iWall, but a Sling Shot device could be build in the FabLab[2] of the Media Computing Group and tested on the iWall.

[2] http://hci.rwth-aachen.de/fablab

Chapter 3

Hardware and Software Setup

3.1 Location

The iWall is installed inside of ten sun blades.

The computer science building of the RWTH Aachen University is located in Aachen West and includes a historic building and three extensions named E1, E2 and E3. At the historic building eleven sun blades are installed on both sides of the building for the first, second and third floor. The iWall is installed inside the back of the sun blades (like shown in figure 3.1) of the third floor, facing the Halifax street and the Halifax student dormitory on the other side of the street.

3.2 iWall Hardware

The iWall Display has a resolution of only 10 pixels in height.

The iWall Hardware is designed to be invisible to visitors of the building at daylight to respect the facade of the historic building. Only in the evening the 19200 unique controllable RGB LEDs, installed in the back of the ten sun blades, getting active and emitting their light to the lower blade to present it as the iWall Display to passing by people. The iWall Display has a width of 60 meters and a height

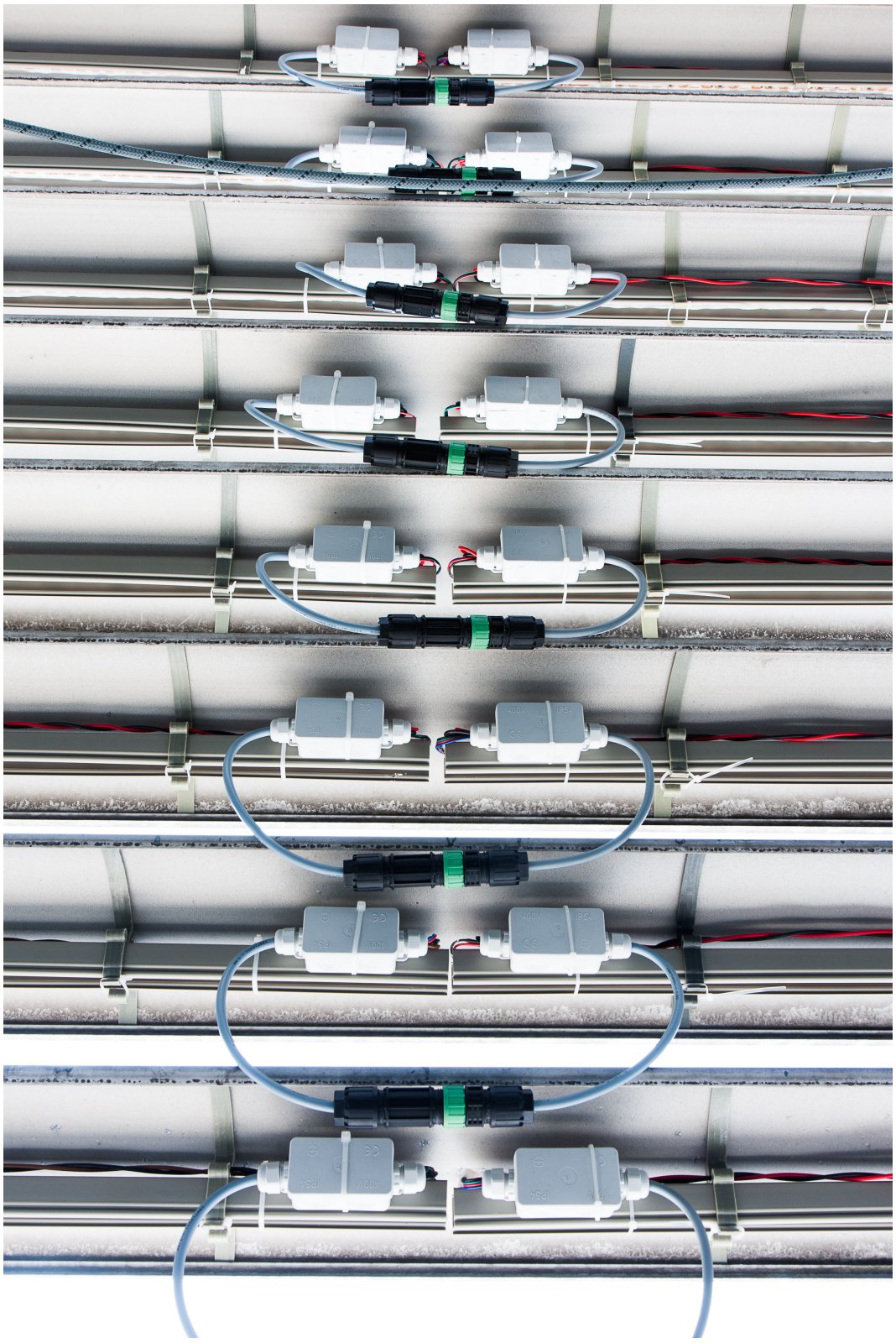


Figure 3.1: LED strips installed inside the sun blades of the computer science building.

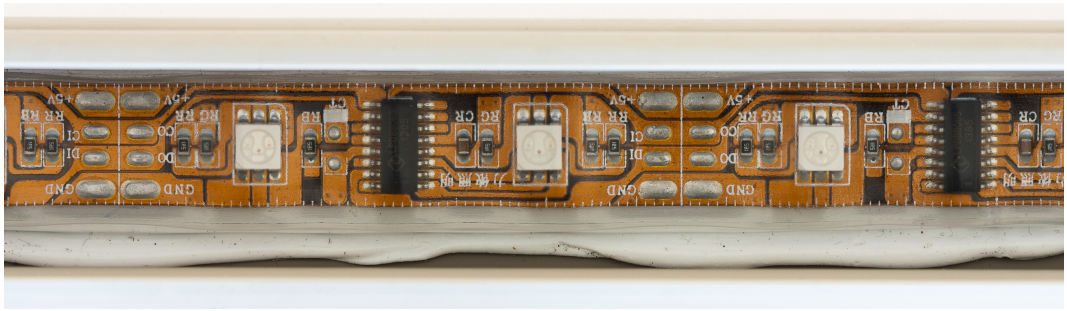


Figure 3.2: LED ribbon with LPD8806 chipset glued into a cable conduit.

of 2 meters. In each of the ten sun blades 1920 LEDs are installed and create a theoretical resolution of 1920 x 10 pixels. The light spots emitted to the blades have a diameter of 20 cm and the distance between the center of each light spot is only 3 cm in the horizontal resolution. Thus the light spots are overlapping and create a horizontal blur in the final image of the iWall Display.

3.2.1 LED strips

The iWall has 60 LED strips as interchangeable parts for maintaining the iWall easily. Each LED strip includes a prefabricated waterproofed LED ribbon of five meters length containing 320 LEDs glued into cable conduits like shown in figure 3.2. At the back of each cable conduit a drilled cable for 48 V power supply is mounted and at its end two little rainproofed white boxes. The white boxes includes a voltage step down controller (figure 3.3) feeding the LED ribbon with 5 V from both ends. Data channels and the 48 V power supply are daisy-chained by a connector to the next LED strip.

Interchangeable parts makes it easy to maintain the iWall.

3.2.2 Power Supply

To minimize the power drop in the power supply cables for the LED strips with 5 V several things are done. The iWall is power supplied from the middle of the wall to get shorter cable length and therefore a lower resistance for the cable.

Power drop is noticeable high for the basic voltage of the LEDs.

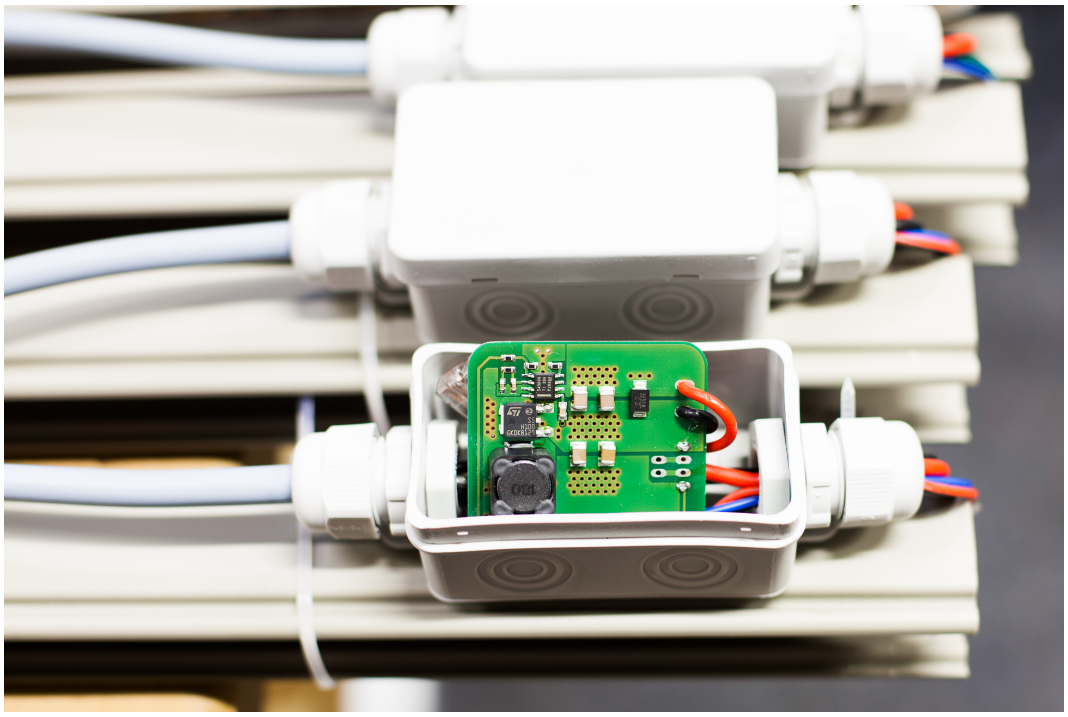


Figure 3.3: Step down controller from 48 V to 5 V installed in a rainproofed box.

The same effect is used by supplying the LED strips from both ends with power. The supply voltage from the power-supply unit to the step down controllers of the LEDs is 48 V. For the same wattage a higher voltage decreases the current and the power drop.

3.2.3 Brain Box

The micro controller operate the iWall are installed in a Brain Box.

The so-called "Brain Box" is a rainproofed box installed outside the building in the middle of the iWall. It includes a BeagleBone Black and a STM32F4 (figure 3.4). The BeagleBone receives pixel data from the iWallDriver through a TCP/IP Stream and send this to the STM to multiplex it into 20 data streams. Each of this 20 data streams will feed a half row of the iWall Display with pixel data from the middle of the building.

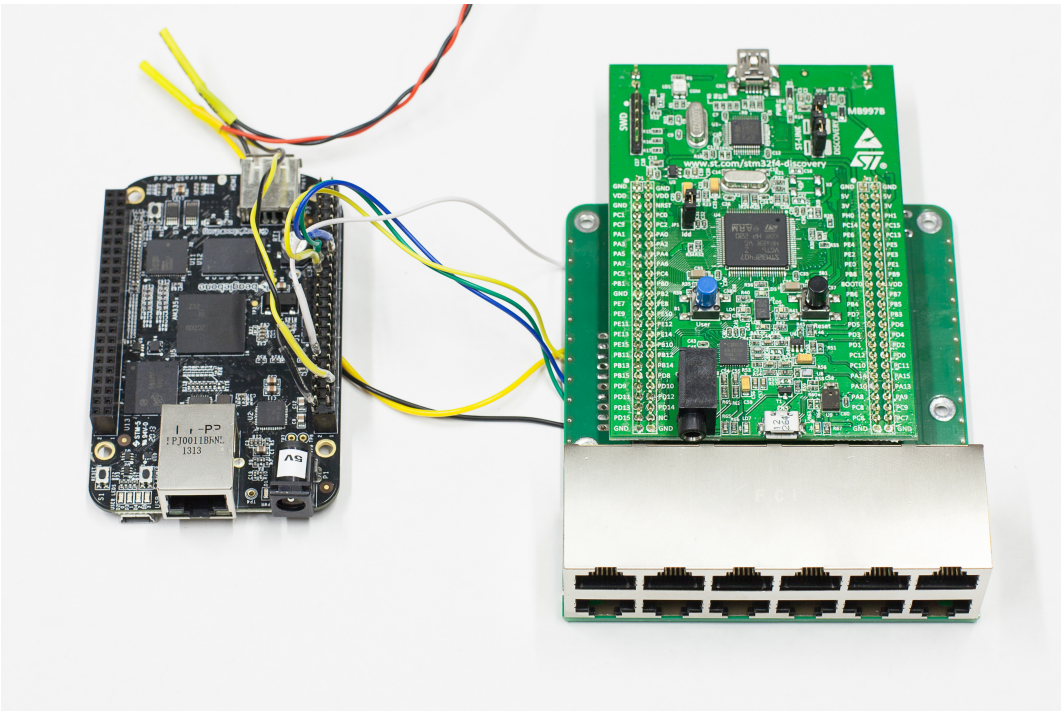


Figure 3.4: On the left the BeagleBone Black connected to the STM32F4 on the right. Each of ten network ports will be connected to a row of LED strips.

3.3 iWall Server Software

The iWall Server Software is running on a Mac.

As iWall server software I describe all the programs running on one dedicated macintosh machine (Mac), located in the server room of the Media Computing Group, to generate content and communicate with the Beagle Bone. All parts of the server software are written in Objective C and collected in one Xcode Workspace including the

- iWallDriver,
- iWallFramework,
- iWallScreenMirror,
- iWallVideoPlayer and
- iWallSimulator.

In the next sections I am going to explain the single software parts in more detail. All iWall Server Software where mainly written by Thorsten Karrer, who is a research assistant at the Media Computing Group.

3.3.1 iWall Driver

The iWall Driver alter pixel data into the correct format for the BeagleBone.

The iWall Driver takes an IOSurface[1] with pixel data of 1920x10 pixels size and 32 bit depth for all 4 channels (RGBA). As in 3.2.3 described the LED strips are feed from the middle of the iWall with pixel data. Thus the pixel of the final image needs to be sorted from the middle to the most outer before sending to the BeagleBone. The iWall Driver splits the pixel data of the IOSurface in two half frames, doing the bit swizzling on this data to fit the correct endianness and send this with TCP/IP to the BeagleBone.

IOSURFACE FRAMEWORK:
The IOSurface framework is provided by Apple and allows to create a frame buffer object to store e.g. pixel data in it. Creating an IOSurface with `IOSurfaceRef IOSurfaceCreate(CFDictionaryRef properties);` and setting the key `kIOSurfaceIsGlobal` to `kCFBooleanTrue` will also allow other processes to find and access the frame buffer object by an associated ID. An IOSurface must be locked for read and write access, because the frame buffer object could also be stored in the graphics memory. The IOSurface framework is used by the iWall Framework to pass pixel data to the iWall Driver, because it is written as a XPC Service running in a different process.

3.3.2 iWall Framework

The iWall Framework transforms frames of different pixel format and size to a standardized one.

The iWall Framework can be include by developers to send pixel data to the iWall. The framework takes a pixel buffer like CVPixelBuffer or IOSurface and is adjusting them to an iWall Driver compatible format by resizing and changing the pixel format. This allows to send a frame with a cor-

[1] `https://developer.apple.com/library/mac/documentation/Miscellaneous/Reference/IOSurfaceAPI_header_reference/Reference/reference.html`

rect aspect ratio of 1920 x 70 pixels to the iWall. Only one application is able to use the iWall Framework at a time.

3.3.3 iWall ScreenMirror

The iWall ScreenMirror as a bridge to transfer data from other applications.

The iWall ScreenMirror was written as an example application on how to use the iWall Framework. This application captures an area of 1920x70 pixels on the lower right corner of the main screen. This is done with the help of AVFoundation classes to generate a CVPixelBuffer for each frame. The CVPixelBuffer is backed by an IOSurface, which is then processed by the iWall Framework. With the iWall ScreenMirror it is possible to capture the graphic output from other applications running on the same system and send them to the iWall. Example applications I tested are a slide-cast and a video player.

3.3.4 iWall VideoPlayer

With iWall VideoPlayer also content from different locations can be streamed to the iWall.

The iWall VideoPlayer is using the AVFoundation to read almost every video files or streams and sending the CVPixelBuffer directly to the iWall Framework. This allows to provide content from outside of the Mac to the iWall. As the iWall Framework will adjust the frame size of the PixelBuffer it is possible to use video streams in all formats. For the correct aspect ratio the video should have the size of 1920 x 70 pixels.

3.3.5 iWall Simulator

The iWall Simulator helps developing applications for the iWall.

The iWall Simulator takes the output of the iWall Driver, swizzle it back to correct endianness and displays the result. It is written to test the correct output of applications using the iWall Framework and can also be used for prototyping while the physical iWall is not running.

Chapter 4

iWall Collaboration

iWall Collaboration is a software collection to enable collaborative iWall Apps.

Running multiple applications on the iWall and interact with them through a single mobile client is the goal of iWall Collaboration. Developers of applications for the iWall (iWall Apps) will be able to include two Frameworks into their Xcode Project to achieve this. Including the iWallCollaboration Framework into a Xcode Project allows to display content to a part of the iWall Display and including the iWallMobile Framework will enable to control the iWall App by mobile clients. In this chapter I like to explain the developing process from the first prototype to the final version of iWall Collaboration. The developing process is done in three steps by designing the application based on the requirements, then implementing it and finally evaluate the application to design the next iteration. This developing process is also known as a DIA Cycle (Design, Implement, Analysis).

4.1 First Prototype

The first prototype is a game running on the iWall.

It was planned to make the iWall available to the public at December the 06th of 2013 for the first time. On this day the TDI ("Tag der Informatik"[1]) event took place. Visitors of this yearly event are usually graduated computer science

[1] http://hci.rwth-aachen.de/public/TDI2013/

students and their dependent, because the graduation ceremony is the main part of it. These visitors were supposed to interact with the iWall. I implemented a game as a part of the iWall Server Software, controlled by mobile devices of the visitors.

4.1.1 Requirements First Prototype

Reaching a maximum number of visitors is important.

As discussed in chapter 2 the mobile devices of the visitors should be used to interact with the iWall. According to kantar wordpanel[2] most smartphones, bought 2012/2013 in Germany, are running Android or iOS. I assume a similar distribution of the Smartphone OS for the group of people, who like to play a game on the iWall (iWall Game). Thus I like to support both platforms to reach a maximum number of visitors with our mobile application. The user interface of the mobile client has to be available in english and german language, as I expected users understanding only one of both. Like every year the event is attended by many visitors and therefore many users, who may interact with the iWall Game. For this I need to implement a queuing system managing the users waiting for a free spot to play. Also every user should not play longer than one minute. It was predictable that not all lighting elements of the iWall would have been finished at the TDI event. Thus the horizontal position of the game, displayed on the iWall, should be easy adjustable. Due to security concerns the iWall Server, located in the server room of the Media Computing Group, is not reachable from outside of the RWTH network. So a separate located web server is needed to provide the content for the mobile devices of the visitors. The connection between them must be initialized by the iWall Server from a port not blocked by the RWTH internal firewall.

[2] http://uk.kantar.com/media/615033/kantar_worldpanel_comtech_smartphone_os_barometer_06_01_14.pdf

4.1.2 Implementation Idea

Using a web application as the mobile client.

Especially for the iOS platform it would be easy to develop and test mobile applications that can connect to our iWall Server, because of Apples vertical integration strategy of the system and software. But distributing these "native" Apps to the visitors at the TDI event would be time-consuming for the visitors and for us, as we did not had enough human resources to help installing the software. So I decided to implement the mobile client for the visitors as a web application (Web App), running in the standard web browser of most smartphones without the need to install additional software.

A web application leads to some more requirements.

By writing a Web App I need to consider some more requirements for the mobile application. I am not able to use the standard "click" event, like it is triggered if someone tapped on a link in the mobile browser. This would otherwise lead to unacceptable timing delays, because the browser software needs time to distinguish between a single tap, double tap and a scrolling gesture. Next I need to prevent scrolling of the website displaying the user interface (UI), to hold the buttons stationary and therefore help the user to hit the buttons while they are looking at the iWall. And finally I should consider what happened if the user reload the website at any stage of the mobile Web App. I decided to implement the Backend of the mobile Web App with the help of the Django Web Framework, because of my experience with this system. The Django Web Framework[3] includes a translation module, a Session Manager I can use to schedule waiting users and Dajax[4] to receive UI events (e.g. touching a button) from a user.

4.1.3 The Game Idea

Adjusting a game idea to fit the requirements of the iWall.

For the game running on the iWall I was inspired by the game SubwaySurfers[5], where a character is jumping between rails avoiding to hit trains. Here I see the sun blades of the computer science building (where the iWall is projected on) as analogous to the rails in this game. Whenever

[3] https://www.djangoproject.com/
[4] http://www.dajaxproject.com/
[5] http://kiloo.com/games/subway-surfers

a train (presented by white dots) appears, the user has to jump with her character (presented by a colored dot), to the next sun blade. As the character jumps only in vertical direction over the sun blades, it allows me to add multiple players next to each other in the horizontal space of the iWall. I decided to put ten players up and place the associated colored dots next to each other, rather than scattering them over the iWall, because, like in 4.1.1 discussed, I did not know the final width of working elements at the iWall. The iWall Game I wrote is named iWall Racer.

4.1.4 User Interface Mobile Client

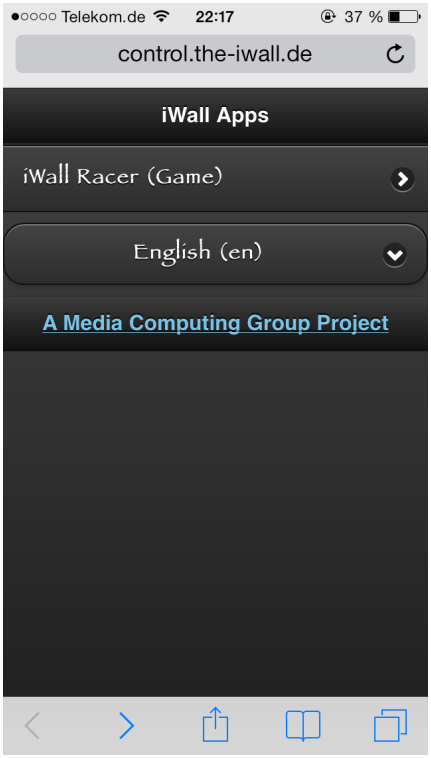
The WebApp presenting the mobile client has three pages with different user interfaces.

The user Interface of the mobile client consists of three pages. The first page (figure 4.1a) with the title "iWall Apps" represent a mental model of a dashboard, where the user can select a running iWall App. Also the UI language can be changed here. By selecting the iWall App "iWall Racer (Game)" the next page (figure 4.1b) with an explanation of the game appears. If the user tap on the "Start the Game" button, the third page (figure 4.1c) with the deactivated game UI and a countdown indicating the waiting time appears. After the countdown finished the UI will get active 4.1d and the user will see a mark indicating the assigned game character, represented by a colored dot. After one minute the browser will redirect back to the first page with the dashboard of iWall Apps.

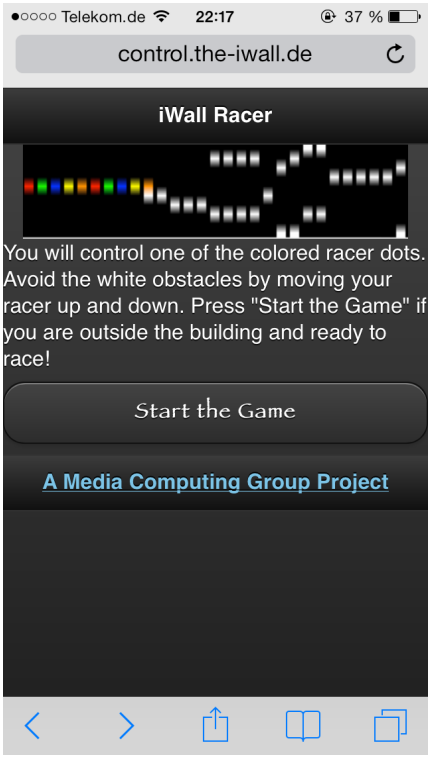
4.1.5 Implementation Details Mobile Client

Only the Web Backend knows the connections between a players and mobile clients.

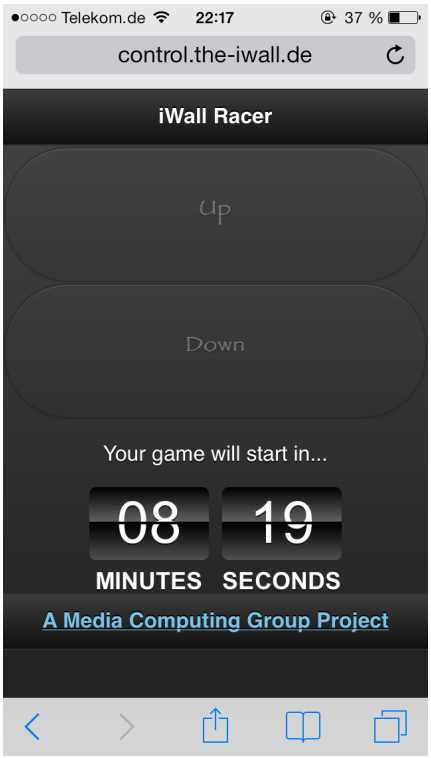
After the user press the "Start the Game" button the next page with the game UI is loaded. At this point a session for this user is created and added to a queue by the web server backend (Web Backend). Even if the user reloads the page she keeps the assigned session. This avoids blocking free game spots by reloading the page. The Web Backend is written in Python with the help of the Django Web Framework and manage the connection between characters in the game and mobile clients. This connection and the queue



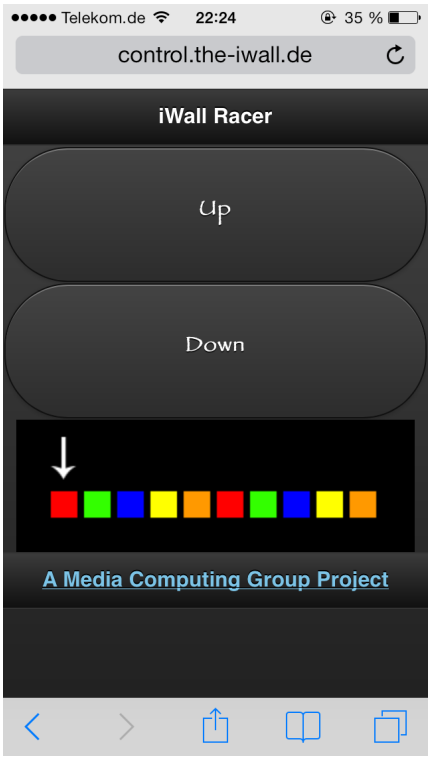
(a) Dashboard like UI with iWall Apps



(b) iWall Racer explanation (text courtesy of Prof. Jan Borchers)



(c) iWall Racer UI deactivated



(d) iWall Racer UI activated

Figure 4.1: The mobile UI of the first prototype.

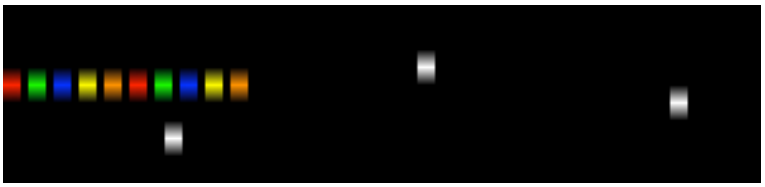


Figure 4.2: The first level of iWall Racer (screenshot from the iWall Simulator).

is unknown to the game running on the iWall server. The touches on the "UP" and "DOWN" button are recognized by the `ontouchstart` javascript function, because as discussed on PhoneGap Tips[6], this will avoid a 300 milliseconds delay by the browser of the mobile client. Scrolling is disabled by preventing the default behavior of the `touchmove` event by calling `evt.preventDefault();` function on this event. After recognizing an event, it is send to the web-server as an ajax request. The web-server will now send the player number and the command through a socket connection to the iWall Server. Each minute the iWall Racer Game, running on the iWall Server, will send a "RESET" command to the Web Backend. The Web Backend will use this information to synchronize the timing for waiting mobile clients and to open the game for the next ten players from the queue.

4.1.6 Interface iWall Racer

The iWall Racer has three levels.

The game consist of three levels and will run always one minute for all ten players. In the first level (figure 4.2), single dots with a fixed horizontal space of 100 pixels and a random vertical position will appear. Then in the second level (figure 4.3), the white dots appear like forming a street. The third level (figure 4.4) is placing the white dots randomly closer together. If one of the white moving dots hits a colored dot, the colored dot will disappear. After each level the colored dots will be shown again to enable users to play all levels.

[6] `http://phonegap-tips.com/articles/fast-touch-event-handling-eliminate-click-delay.html`

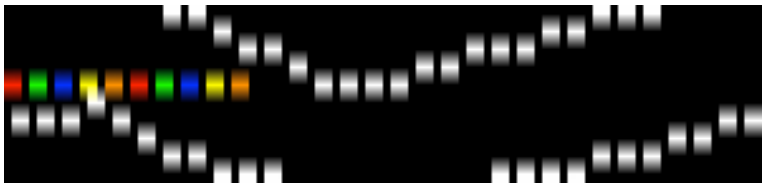


Figure 4.3: The second level of iWall Racer (screenshot from the iWall Simulator).

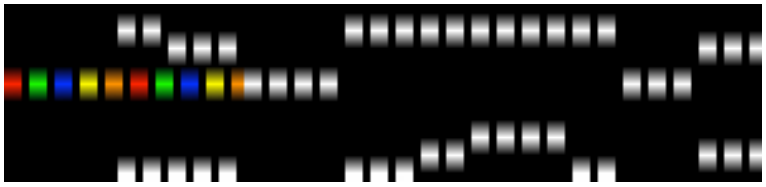


Figure 4.4: The third level of iWall Racer (screenshot from the iWall Simulator).

4.1.7 Implementation Details iWall Racer

The iWall Racer game is written in Objectiv C as a part of the iWall Server Software and is using the iWall Framework to submit a `CGBitmapContext` with a resolution of 1920 x 10 pixels to the iWall. A timer called `gameTimer` is firing 20 times per seconds to render a frame into a `CGBitmapContext` for the iWall Display. Another timer called `resetTimer` fires every 60 seconds and resets the game by informing the web server for the mobile clients, creating new random positions for the white dots and resetting the position of the colored dots. The communication between iWall Racer and the Web Backend is established by using SocketRocket[7] and some helper classes.

4.1.8 Implementation Details of the Socket Connection

Like shown in figure 4.5 the socket connection between iWall Racer and the Django Web Framework is tunneled by

[7] `https://github.com/square/SocketRocket`

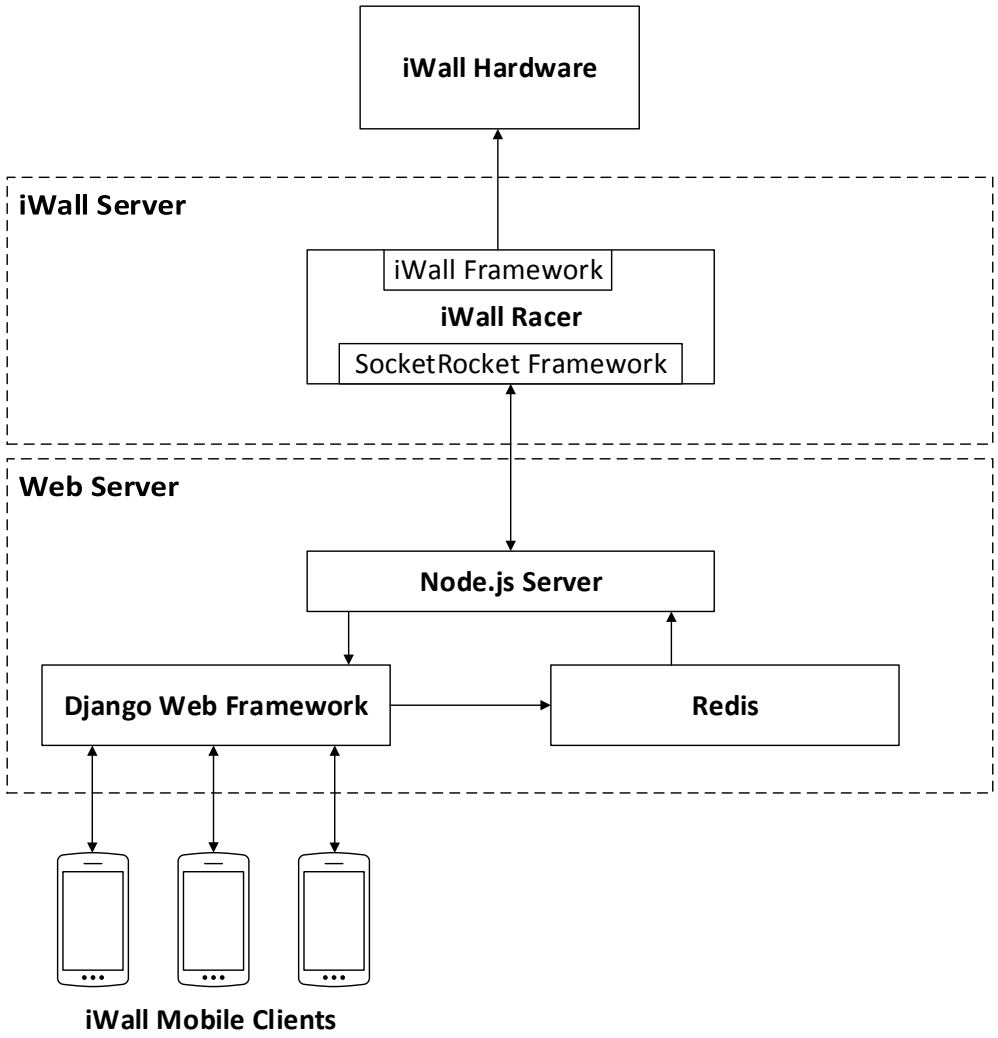


Figure 4.5: First Prototype: Overview about the server structure

a Node.js Server and on the back channel additionally by a Redis database server. The Django Web Framework is deployed by an Apache web server and has a limited runtime. This is the reason why I choose an additional server to handle the incoming socket connection and keep it open for a long runtime. Sending data from the Django Web Framework to the Node.js Server is done by using the messaging system of the Redis database server while sending data back can be done by simple HTTP calls.

4.1.9 Evaluation

The feedback by some students was beneficial.

Because of technical hardware problems, the iWall Game was not running on the TDI event. Nevertheless I showed the game running on the iWall Simulator to some students at this day. The feedback here was beneficial to me. All users found the timing between touch and reaction on the iWall Simulator acceptable, even if the test system was connected through WIFI to the web server. One user reports, the Mobile WebApp does not work on a Windows Phone. I identified this as a problem with some javascript functions, not available on the Windows Phone platform. The same problem appears on a desktop system with mouse input, where the `ontouchstart` javascript function is not available. Some users considered that it would be better to know the assigned colored dot before the countdown finished, one other the buttons could be larger for her device. I noticed, if the window of the iWall Game application is in background or overlapped by other application windows, the `gameTimer` will get called irregular and drop frames. This behavior is called App Nap and can be disabled.

APP NAP:
Apple introduces App Nap with OS X Mavericks. Applications minimized to the dock or overlapped by other windows are putting to sleep. This is the reason why the render timer for my frist prototype get called irregular. App Nap can be disabled by creating an activity with `[[NSProcessInfo processInfo] beginActivityWithOptions:NSActivityIdleSystemSleepDisabled reason:@"Good Reason\];`.

4.1.10 Conclusion

The iWall Game is the first iteration towards iWall Collaboration.

With the iWall Game I implemented the first interactive application for the iWall. The next iteration in my developing process is the implementation of the iWall Collaboration Software. Based on the findings from implementing and presenting the iWall Racer, now I can define the requirements for the iWall Collaboration Framework.

4.2 Requirements for iWall Collaboration

Running multiple iWall Apps on the iWall is the main requirement.

The iWall Collaboration software should provide shared access to the iWall on the one hand and interaction with mobile clients on the other. It needs to merge the drawings created by the connected clients (called "iWall Apps") to one frame that fits the size of the iWall. Adding a new application to the frame displayed on the iWall should be done by an animated transition to satisfy the users like mentioned in section 2.1.2. The frame-rate should be controlled by the iWallCollaboration Server, because the iWall hardware can only handle 20 frames per second. Frame drops like in 4.1.9 described must be avoided. If an iWall App disconnects from the iWall Collaboration Server and therefore creates a gap on the iWall display, a strategy how to fill this gap should be defined. Developers of iWall Apps should be able to use predefined mobile user interfaces to avoid an additional deployment step to a web server like it was necessary for the first prototype. These mobile user interfaces should work on most platforms and allow also mouse clicks like in 4.1.9 discussed. Changing the user interface while an iWallApp is running should be possible. Different to the first prototype also the iWall Apps should know about all connected mobile clients.

4.3 Implementation Details of iWall Collaboration

The iWall Collaboration Software has two separated frameworks.

I decide to create two separated frameworks called "iWall-Collaboration Framework" and "iWall Mobile Framework" for iWall Collaboration. Adding these two frameworks to a Xcode project allows the iWall App to connect to their server counterparts called "iWall Collaboration Server" and "iWall Mobile Server" like shown in figure 4.6. The iWall Mobile Framework is separated from the iWallCollaboration Framework, because applications for the iWall may not need mobile interaction and like discussed in 4.1.1 the iWall Mobile Server may not be running on the same machine as the iWallCollaboration Server. With this solution

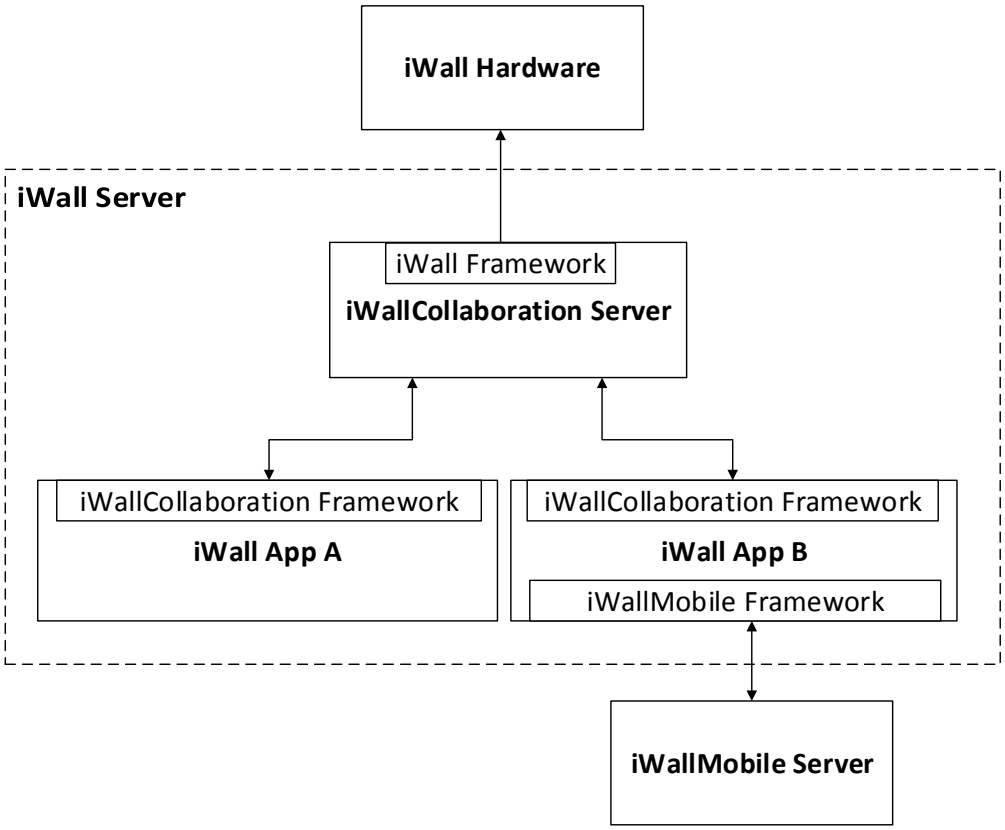


Figure 4.6: iWall Collaboration: Overview about the server structure

I avoid unnecessary tunneling. In the next two sections I explain both server implementation in more detail.

4.3.1 Implementation of the iWall Collaboration Server

The iWall Collaboration Server use the IOSurface framework to receive pixel data from its clients.

To establish a socket connection between the iWallCollaboration server and its framework I use the "ThoMoNetworking Framework"[8]. The clients (iWall Apps) on this connections create drawings with a resolution of 1920x70 pixels, with 32 bit with 20 frames per second. This would be a data rate about 10 MB every second and it would waste

[8] hci.rwth-aachen.de/thomonet

bandwidth if sending this data over the socket connection. As the iWall Apps are running on the same machine as the iWall Collaboration Server, I use the IOSurface framework to create shared frame-buffer objects. Then I use the socket connection to trigger the drawing process of the clients every 1/20 second.

PLACING A DYNAMIC FRAMEWORK INSIDE ANOTHER:
I include the ThoMoNetworking framework inside the iWall Collaboration Framework. If an iWall App includes the iWall Collaboration Framework, the dynamic loader would only find the ThoMoNetworking framework if it is located at `/Library/Frameworks` or in the run-path search paths of the iWall App itself. To avoid this problem I set the installation path of the ThoMoNetworking Framework to `@rpath` (Run-Path Dependent) and therefore include it also in the run-path search paths of the iWall App. In my opinion it should be avoided to create such umbrella frameworks, because frameworks are typically provided from different resources and are updated differently. But here I like to keep it simple for developers by include a single framework in the iWall Apps.

4.3.2 Scheduling the iWall Apps

Scheduling the iWall Apps to avoid gaps on the iWall Display while not disturbing the users.

All iWall Apps render into an IOSurface with the full resolution of the iWall and must define a minimum width `n` in pixels. This ensures the first `n` pixel columns of the IOSurface is rendered at the iWall Display. If a new iWall App appears on the iWall Display it slides in with an animation from the right and will resize an existing iWall App to its minimum size. Finishing an iWall App creates a gap on the iWall Display. While an iWall App, like the iWall Racer game, should keep its assigned spot on the iWall Display to avoid disturbance of users playing the game, an iWall App displaying only text could also move on the iWall Display to fill these gaps. Thus iWall Apps can define that they content may move on the iWall Display. Otherwise these gaps are filled whenever a new iWall App fits into it.

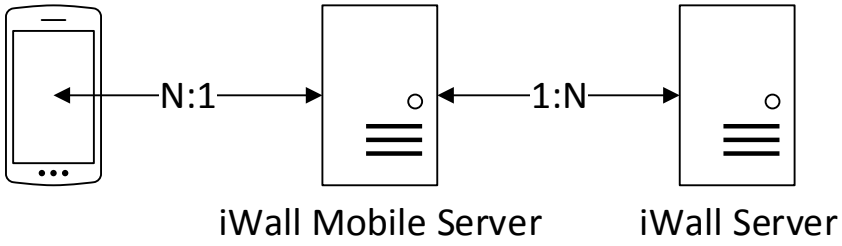


Figure 4.7: iWall Mobile Server: Overview about the server structure

4.3.3 Implementation of the iWall Mobile Server

The iWall Mobile Server has two different network ports for server and client applications.

The iWall Mobile Server is a Node.js[9] server written in JavaScript with the socket.io[10] add-on. It is listening on two ports, one for the mobile clients and the other for iWallApps. This also allows to receive connections from multiple iWall Servers (e.g. if another iWall is installed) like shown in figure 4.7. For the mobile client a web server like Apache [11] or Nginx[12] provides one static HTML page. The dynamic content is emitted from the iWall Mobile Server by connecting to it. The UI on this page is created with the help of the jQuery Mobile Framework[13]. I am using the `vmousedown` function[14] from jQuery Mobile to receive touches and mouse clicks as one single event with best timing.

4.4 Example iWall Apps

To test the expect behavior of the iWall Collaboration software I implement several example iWall Apps I like to present in this section.

[9] `http://nodejs.org/`
[10] `http://socket.io/`
[11] `http://httpd.apache.org/`
[12] `http://nginx.org/`
[13] `http://jquerymobile.com/`
[14] `http://api.jquerymobile.com/vmousedown/`

4.4.1 iWall Example Game Client

This application is adapted from the iWall Racer Game to run with the iWall Collaboration framework.

4.4.2 iWall Example Font Tester Client

To identify the best font for the iWall Display I wrote this applications. It loads all fonts available at the current system in an array and display them one after another. The fonts can be switched by using the UI shown in figure B.1a.

4.4.3 iWall Example Color Picker Client

I wrote this color picker application to test the correct transition animations between several iWall Apps. It displays a solid color over the full frame of the application on the iWall. Changing the color can be done by using the color picker in the application window or by using the mobile UI shown in figure B.1b.

4.4.4 iWall Example Bus Client

This example iWall App fetches live data from the ASEAG (company operating the bus connections in Aachen) server and displays the latest bus arriving on the iWall (figure C.1).

Chapter 5

Summary and future work

With the first prototype I created an iWall App and run through several problems. Other developers may experience the same if writing new iWall Apps from scratch. I implemented the iWall Collaboration Software to avoid implementation mistakes and make it easier for developers to create new interactive iWall Apps.

5.1 Summary and contributions

With iWall Collaboration I presented a software system for developers to create iWall Apps for Collaboration. It is now possible to run multiple iWall Apps at the same time by sharing the screen of the iWall Display. All iWall Apps can be controlled by a single mobile client, running in most browsers. The iWall Collaboration Software creates the user interface for the mobile clients and manage the space, each iWall App gets on the iWall Display. New iWall Apps can be deployed on the iWall Server in a single step without affecting the iWall Mobile Server.

5.2 Future work

During my Implementation of iWall Collaboration I had many ideas for improving my work but not enough time to implement them. In this section I like to sum up these ideas.

5.2.1 Platform Independent iWall Apps

All iWall Apps are only working with the iWall Collaboration Server if they are running on the same Mac. The iWall Collaboration Server should be extended to allow also iWall Apps providing content from another, platform independent location.

5.2.2 Multiple iWalls

If another iWall or similar interactive Media Facade is installed on the computer science building it would make sense to control it also with the same mobile client. As I abstracted away the iWall Mobile Server from the iWall Collaboration Server it is possible to use the iWall mobile client also for another Media Facade. For this purpose the iWall mobile client should be extended to show only the application available on the Media Facade selected by the user.

5.2.3 Slingshot Device

Like in 2.3.4 explained it would be possible to interact with a Slingshot Device at the iWall. Due to lack of time I was not able to finish a prototype and test it.

5.2.4 iWall as a Tool

The iWall could also work as a tool for students integrated in their every day life. For Large Public Displays Greenberg and Rounding [2001] present the Notification Collage and Huang et al. [2004] present IM Here to do this. These ideas could be adapted for the iWall Display.

5.2.5 Evaluation iWall

If the iWall is installed completely and running continuously an evaluation of iWall Collaboration can be done by asking passing by or interacting users. Tracking the users by a camera is complicated, because the iWall facade is illuminated only at evening or night. With this evaluation the expected honey pot effect could be proven and it would be able to find out if the threshold to interact with the iWall is low enough.

Appendix A

Create a new iWall App

Creating a new iWall App with Xcode and the iWall Collaboration software can be done by the following steps:

1. Create a new Xcode Project with a "Cocoa Application" and choose a project name.

2. Under "Linked Frameworks and Libraries" (figure A.1) add `iWallCollaboration.framework`, `IOSurface.framework` and if your application should be supported by the iWall Mobile Client also `iWallMobile.framework`.

3. Include the iWall Collaboration Framework (and the iWall Mobile framework) by adding

   ```
   #import <iWallCollaboration/iWallCollaboration.h>
   #import <iWallMobile/iWallMobile.h>
   ```

   to your class.

4. Create the properties

   ```
   @property (nonatomic, strong) iWallCollaboration *myiWall;
   @property (nonatomic, strong) iWallMobile *myMobileServer;
   ```

   and add the protocols `iWallCollaborationDelegate` (and `iWallMobileDelegate`) to your class.

5. Initialize and connect to the iWall Collaboration Server (and iWall Mobile Server) by adding

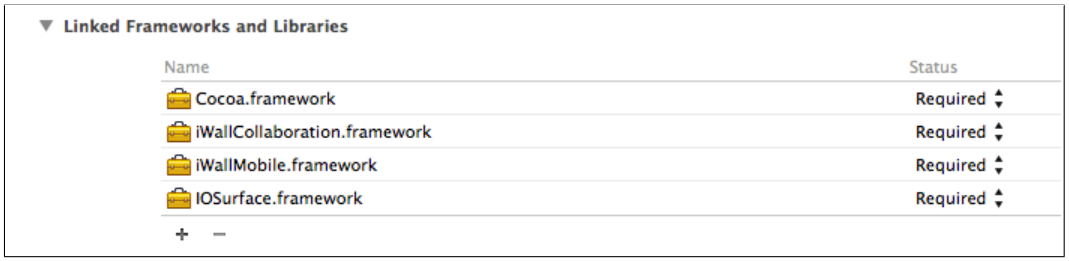


Figure A.1: Screenshot after adding iWallCollaboration.framework, IOSurface.framework and iWallMobile.framework to the Xcode Project.

```
self.myiWall = [[iWallCollaboration alloc]
                                initWithiWallIdentifier:@"iWall"];
[self.myiWall setDelegate:self];
[self.myiWall connectToiWall];

self.myMobileServer = [[iWallMobile alloc] init];
[self.myMobileServer setDelegate:self];
[self.myMobileServer connectToMobileServer];
```

to your sourcecode.

6. Implement the required delegate methods

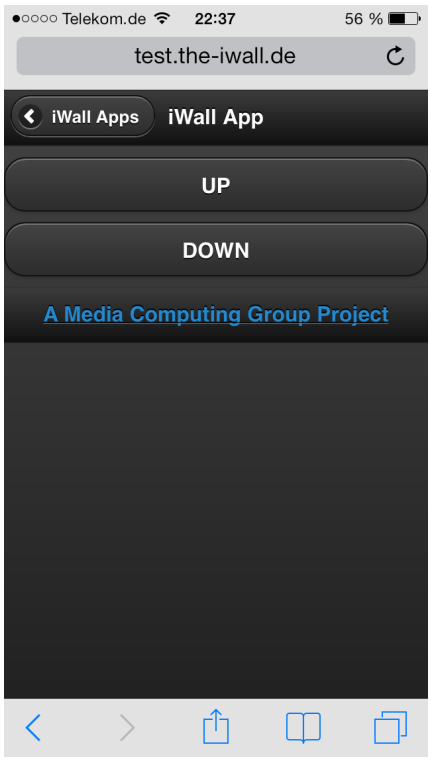
```
// returns the minimum width of the iWall App in pixels
- (int) minWidth;
// draw content into the given IOSurface
- (void) drawInIOSurface: (IOSurfaceRef) surfaceRef;

// returns title displayed at the
// dashboard of the mobile client
- (NSString*) titleForTabBar;
// returns the unique identifier for the app
- (NSString*) appIdentifier;
// returns the max number of clients
// that can connect use this app
- (NSNumber*) maxNumberOfClients;
// returns the InterfaceType
// displayed on all mobile clients
- (IntefaceType) interfaceType;
```

7. Check the iWall Example Apps in Appendix D for example implementation.

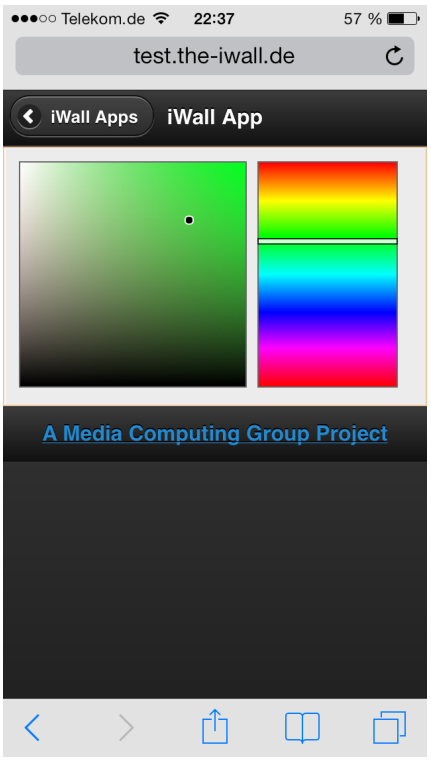
Appendix B

Predefined Mobile UIs

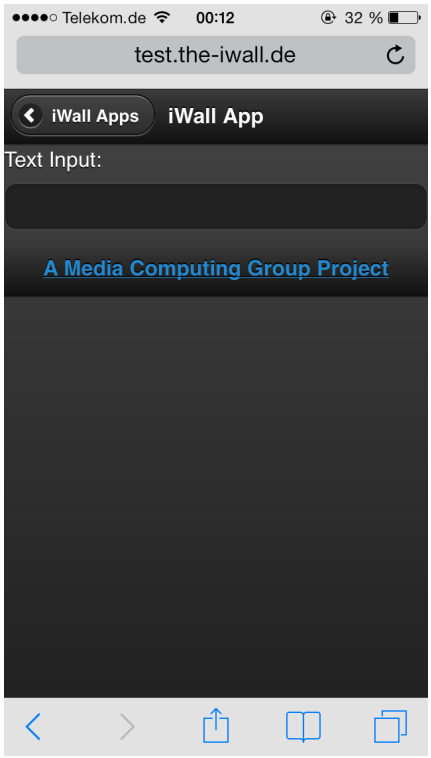
In this appendix I sum up the predefined mobile UIs (figure B.1) available for developers of iWall Apps.



(a) mobile UI with Up and Down Button



(b) mobile UI with color picker



(c) mobile UI with text input

Figure B.1: The predefined mobile UIs available at typeset of this document.

Appendix C

iWall Test Drive

- On December 6th, 2013 at the TDI event the iWall Display was controlled only by the STM and fades between different colors.
- On February 26th, 2014 at the dorkbot[1] meeting the finished part of the iWall was running with the iWall Video Player showing the dorkbot logo.
- On March 10th, 2014 the iWall was running with the iWall Collaboration Framework the first time (figure C.1 and C.2).

[1] `https://hci.rwth-aachen.de/dorkbot`

Figure C.1: The iWall running the Example Color Picker Client on the left and the iWall Bus Client next to it (March 10th, 2014).

Figure C.2: Two columns with LED strips are still missing and one row on the left side is not working.

Appendix D

DVD with Sourcecode

The attached DVD includes all source files to run the iWall Collaboration Server, the iWall Mobile Server and all example applications. In the folder `iWallMobileServer` the file `README.TXT` explains how to install and start the iWall Mobile Server. The folder `iWall` contains the Xcode Workspace for the iWall Server Software and can be opened with the file `iWall.xcworkspace`. It also includes the iWall Simulator, iWall VideoPlayer, iWall ScreenMirror, iWall Framework and the ImageSwizzler projects, that are all property of the Media Computing Group. The project written by me for this thesis are the projects iWall Example Font Tester Client, iWall Example Bus Client, iWall Example Game Client, iWall Example Color Picker Client, iWall Mobile Framework, iWall Collaboration Server and iWall Racer. To run the iWall Collaboration Server first start the server then the clients.

Bibliography

Rafael Ballagas, Michael Rohs, and Jennifer G. Sheridan. Sweep and point and shoot: Phonecam-based interactions for large public displays. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '05, pages 1200–1203, New York, NY, USA, 2005. ACM. ISBN 1-59593-002-7. doi: 10.1145/1056808.1056876. URL http://doi.acm.org/10.1145/1056808.1056876.

Sebastian Boring, Sven Gehring, Alexander Wiethoff, Anna Magdalena Blöckner, Johannes Schöning, and Andreas Butz. Multi-user interaction on media facades through live video on mobile devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2721–2724, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0228-9. doi: 10.1145/1978942.1979342. URL http://doi.acm.org/10.1145/1978942.1979342.

Harry Brignull and Yvonne Rogers. Enticing people to interact with large public displays in public spaces. In *In Proceedings of the IFIP International Conference on Human-Computer Interaction (INTERACT 2003*, pages 17–24, 2003.

Peter Dalsgaard and Kim Halskov. Designing urban media façades: Cases and challenges. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 2277–2286, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-929-9. doi: 10.1145/1753326.1753670. URL http://doi.acm.org/10.1145/1753326.1753670.

Scott Elrod, Richard Bruce, Rich Gold, David Goldberg, Frank Halasz, William Janssen, David Lee, Kim McCall,

Elin Pedersen, Ken Pier, John Tang, and Brent Welch. Liveboard: A large interactive display supporting group meetings, presentations, and remote collaboration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '92, pages 599–607, New York, NY, USA, 1992. ACM. ISBN 0-89791-513-5. doi: 10.1145/142750.143052. URL http://doi.acm.org/10.1145/142750.143052.

Patrick Tobias Fischer and Eva Hornecker. Urban hci: Spatial aspects in the design of shared encounters for media facades. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 307–316, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1015-4. doi: 10.1145/2207676.2207719. URL http://doi.acm.org/10.1145/2207676.2207719.

Saul Greenberg and Michael Rounding. The notification collage: Posting information to public and personal displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '01, pages 514–521, New York, NY, USA, 2001. ACM. ISBN 1-58113-327-8. doi: 10.1145/365024.365339. URL http://doi.acm.org/10.1145/365024.365339.

Elaine M. Huang and Elizabeth D. Mynatt. Semi-public displays for small, co-located groups. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pages 49–56, New York, NY, USA, 2003. ACM. ISBN 1-58113-630-7. doi: 10.1145/642611.642622. URL http://doi.acm.org/10.1145/642611.642622.

Elaine M. Huang, Daniel M. Russell, and Alison E. Sue. Im here: Public instant messaging on large, shared displays for workgroup interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 279–286, New York, NY, USA, 2004. ACM. ISBN 1-58113-702-8. doi: 10.1145/985692.985728. URL http://doi.acm.org/10.1145/985692.985728.

ElaineM. Huang, Anna Koster, and Jan Borchers. Overcoming assumptions and uncovering practices: When does the public really look at public displays? In Jadwiga Indulska, DonaldJ. Patterson, Tom Rodden, and

Max Ott, editors, *Pervasive Computing*, volume 5013 of *Lecture Notes in Computer Science*, pages 228–243. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-79575-9. doi: 10.1007/978-3-540-79576-6_14. URL http://dx.doi.org/10.1007/978-3-540-79576-6_14.

Daniel Michelis and Jörg Müller. The audience funnel: Observations of gesture based interaction with multiple large displays in a city center. *International Journal of Human-Computer Interaction*, 27(6):562–579, 2011. doi: 10.1080/10447318.2011.555299. URL http://www.tandfonline.com/doi/abs/10.1080/10447318.2011.555299.

Laurent Mignonneau and Christa Sommerer. Media facades as architectural interfaces. In Christa Sommerer, LakhmiC. Jain, and Laurent Mignonneau, editors, *The Art and Science of Interface and Interaction Design*, volume 141 of *Studies in Computational Intelligence*, pages 93–104. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-79869-9. doi: 10.1007/978-3-540-79870-5_6. URL http://dx.doi.org/10.1007/978-3-540-79870-5_6.

Jörg Müller, Florian Alt, Daniel Michelis, and Albrecht Schmidt. Requirements and design space for interactive public displays. In *Proceedings of the International Conference on Multimedia*, MM '10, pages 1285–1294, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-933-6. doi: 10.1145/1873951.1874203. URL http://doi.acm.org/10.1145/1873951.1874203.

Susanne Seitinger, Daniel S. Perry, and William J. Mitchell. Urban pixels: Painting the city with light. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 839–848, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-246-7. doi: 10.1145/1518701.1518829. URL http://doi.acm.org/10.1145/1518701.1518829.

Norbert A. Streitz, Jörg Geißler, Torsten Holmer, Shin'ichi Konomi, Christian Müller-Tomfelde, Wolfgang Reischl, Petra Rexroth, Peter Seitz, and Ralf Steinmetz. i-land: An interactive landscape for creativity and innovation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '99, pages 120–127, New

York, NY, USA, 1999. ACM. ISBN 0-201-48559-1. doi: 10.1145/302979.303010. URL http://doi.acm.org/10.1145/302979.303010.

Alexander Wiethoff and Sven Gehring. Designing interaction with media façades: A case study. In *Proceedings of the Designing Interactive Systems Conference*, DIS '12, pages 308–317, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1210-3. doi: 10.1145/2317956.2318004. URL http://doi.acm.org/10.1145/2317956.2318004.

Index

Typeset March 18, 2014