DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics: Games Engineering

Smartphone-assisted Virtual Reality using Ubi-Interact

Michael Alexander Lohr

DEPARTMENT OF INFORMATICS

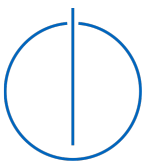
TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics: Games Engineering

Smartphone-assisted Virtual Reality using Ubi-Interact

Smartphone-gestützte Virtuelle Realität mit Ubi-Interact

Author:	Michael Alexander Lohr
Supervisor:	Prof. Gudrun Johanna Klinker, Ph.D.
Advisor:	Sandro Weber, M.Sc.
Submission Date:	October 15, 2019



I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

Munich, October 15, 2019 Michael Alexander Lohr

Acknowledgments

Abstract

Contents

Abbreviations

IMU Inertial Measurement Unit

UBII UBI-Interact

VR Virtual Reality

1 Introduction

Hi, this is my thesis, and I'm going to be the introduction.

2 Implementation

2.1 Ubi-Interact

UBI-Interact (UBII) is a framework for distributed applications, which enables to connect all kinds of different devices together. A centralized server is used to manage the system in a local network. It is currently developed and maintained by by Sandro Weber, who is also the advisor for this thesis. The abstraction into devices, topics and interactions allows to decouple the implementation of a software from device specific environments.

2.1.1 Architecture

The main components of the UBII framework are:

Clients describe a basic network participant. For every client registered in the system, exists is one network socket adress. So clients are an abstraction of a physical network device. They are defined by a unique identifier.

Devices can be registered by clients. A device is an abstraction for a virtual device, which groups different input and output devices together. It is defined by a unique identifier and a list of components.

Components contain the topic name, message format for input/output devices and wether it publishes input or receives output data. A data source for such an input device, could be any sensor for example a button or an Inertial Measurement Unit (IMU). Data output examples for input devices are lamps or displays.

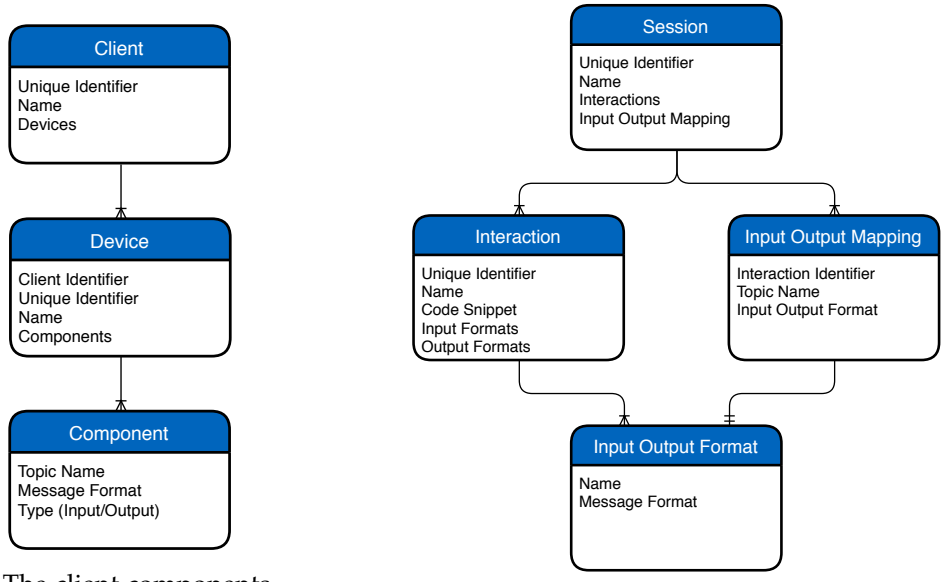
Message Formats are strings which identify a certain data format. Most common data types are available, like for example *Vector4×4*, *Vector2* or *Vector3*.

Topics are data channels which are addressed by a name. Clients can publish messages to topics, which are registered by a device. Clients are also able to receive messages, after subscribing to a topic. Such messages (also called “topic data”)

are formatted as JSON[1]-string, whose structure is defined by the device.

Sessions operate on the server, but can be specified by the client. They are defined by a unique identifier as well as a list of interactions and **input/output mappings**. The mappings are defined by a message format and topic name.

Interactions are reactive components. They operate on topics and are defined by a source code snippet[2]. Interactions are executed in a fixed interval on the UBII server. They can subscribe to topics and use the the received topic data as input, given an input/output mapping description. The output of the interaction is published into another topic. It is also possible to keep data to use in future executions (persistent state).



(a) The client components.

(b) The session components.

Figure 2.1: Relationships of the core components in an entity relationship diagram.

2.1.2 Interactions

An powerful but optional core feature of UBII are so called interactions. As explained in the component overview (see 2.1.1), they are reactive components, which operate on

[1]JSON is a standardized data exchange format, that uses human-readable text. It is often used for web-based data communication.

[2]Currently only JavaScript is supported as a script language.

topics and regularly execute given code snippets. They can be used to outsource logic and calculations to the UBII server. This abstraction introduces the possibility to reuse logic in other applications in similar context.

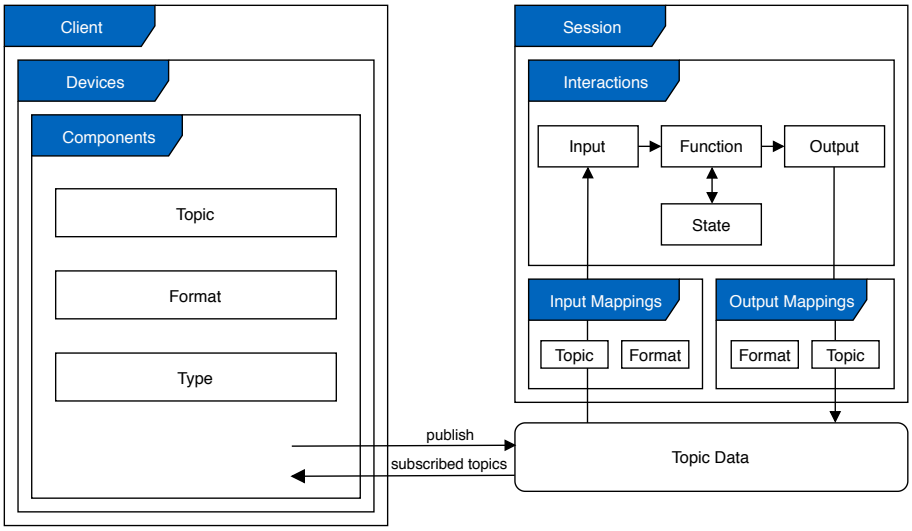


Figure 2.2: Interaction processing overview. This graphic gives a rough overview of the dataflow when using an interaction. Conceptualized with the help of Sandro Weber.

They are also useful, if you want to connect two topics with different formats. Imagine an application, which consumes a rotation given in euler angles in radians. But some devices publish euler angles in degrees. We could implement an interaction, which takes euler angles in degrees from one topic, and publishes euler angles in radians to another one, which then is used by the application.

The code snippet has to define a function, which accepts three parameters: `inputs` is a collection of values, which were published into a topic. The topic which was used, is defined by the input mappings of the session. `outputs` is an empty collection, where values can be added. Those values are then published into a topic, defined by the output mappings of the session. `state` stores a persistent collection of values, which can be used in later executions of the same interaction.

```
1  // detect intentional movement by comparing the current position with a previous one
2  function (inputs, outputs, state) {
3    const threshold = 0.05;
4
5    if (state.position) {
6      const vector = {
7        x: inputs.position.x - state.lastPosition.x,
8        y: inputs.position.y - state.lastPosition.y,
9      };
10
11     const squaredDistance = Math.pow(vector.x, 2) + Math.pow(vector.y, 2);
12
13     outputs.moved = squaredDistance < threshold;
14   } else {
15     outputs.moved = true;
16   }
17
18   state.lastPosition = inputs.position;
19 }
```

Figure 2.3: An example for an interaction written in JavaScript. This interaction calculates the squared distance of two points. One of the points comes from the input, while the other one is stored in the state. To achieve this, the euclidean vector norm of the subtraction of both vectors without the square root is calculated and compared with a threshold constant. The result is then written into the output as a boolean data type.

3 Introduction

3.1 Section

3.1.1 Cite Tests

Citation test [Afo+17] vs [Afo+17].
More [pre Afo+17, post]
Also possible[1]

Acronym test Virtual Reality (VR) cool.
And a second time VR awesome.

3.1.2 Draw Test

3.1.3 Other Tests

See Table 3.1, Figure 3.2, Figure 3.3, Figure 3.4.

Table 3.1: An example for a simple table.

A	B	C	D
1	2	1	2
2	3	2	3

[1] Afo+17.

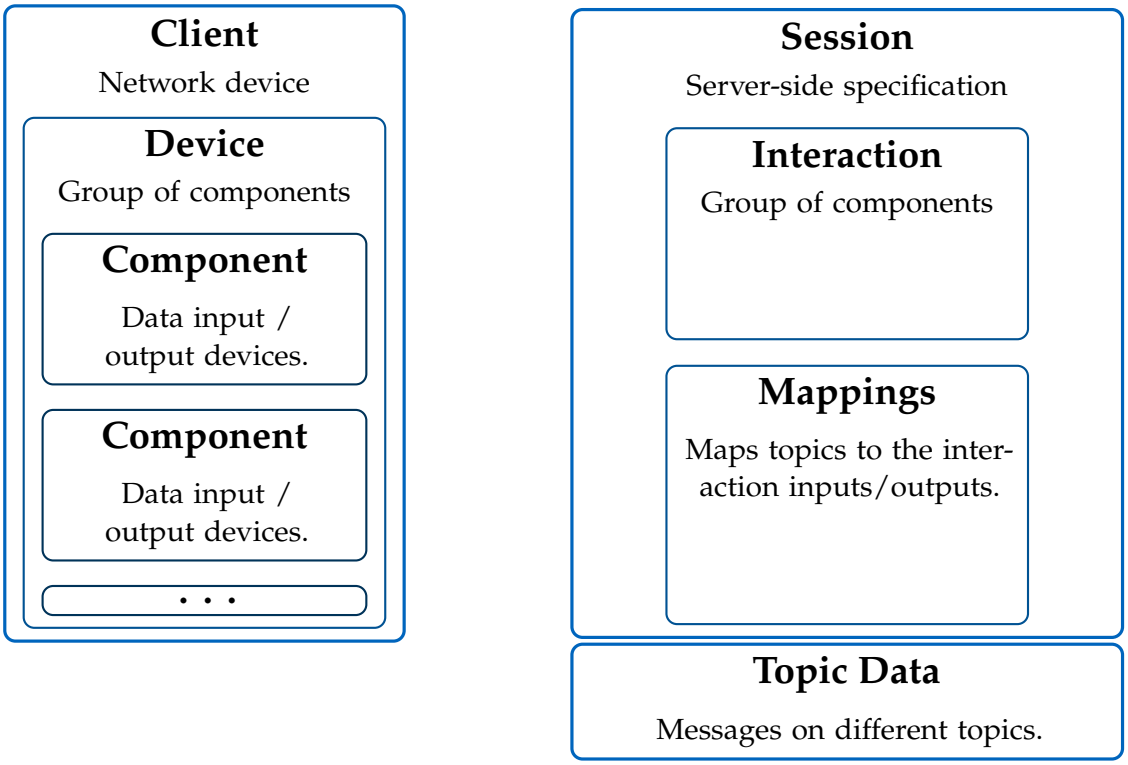


Figure 3.1: Do not forget!

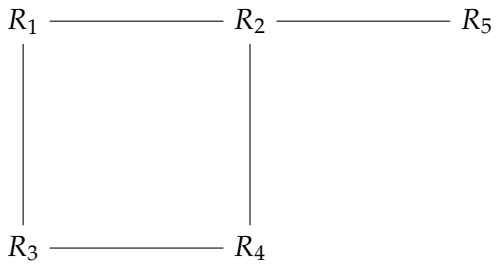


Figure 3.2: An example for a simple drawing.

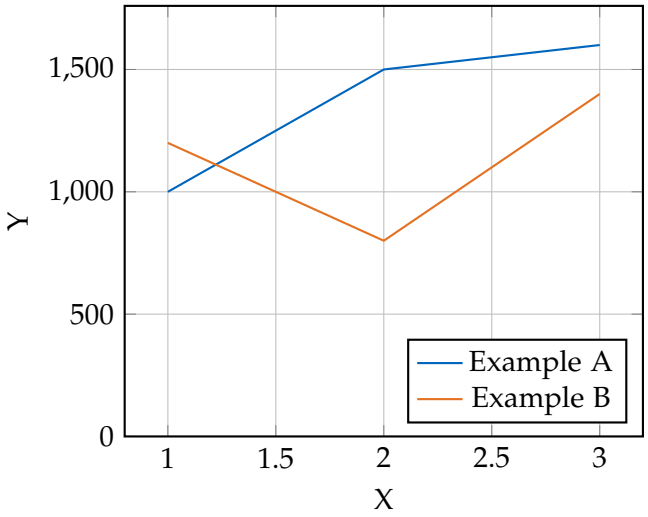


Figure 3.3: An example for a simple plot.

```
1  SELECT * FROM tbl WHERE tbl.str = "str"
```

Figure 3.4: An example for a source code listing.

List of Figures

List of Tables

Bibliography

[Afo+17] L. Afonso, P. Dias, C. Ferreira, and B. S. Santos. "Effect of hand-avatar in a selection task using a tablet as input device in an immersive virtual environment." In: *2017 IEEE Symposium on 3D User Interfaces (3DUI)*. Piscataway, NJ: IEEE, 2017, pp. 247–248. ISBN: 978-1-5090-6716-9. DOI: 10.1109/3DUI.2017.7893364.