Classifying Interaction Methods to Support Intuitive Interaction Devices for Creating User-Centered-Systems

Dirk Burkhardt, Matthias Breyer, Christian Glaser,
Kawa Nazemi, and Arjan Kuijper

Fraunhofer Institute for Computer Graphics Research, Fraunhoferstraße 5,
64283 Darmstadt, Germany
{dirk.burkhardt,matthias.breyer,christian.glaser,
kawa.nazemi,arjan.kuijper}@igd.fraunhofer.de

Abstract. Nowadays a wide range of input devices are available to users of technical systems. Especially modern alternative interaction devices, which are known from game consoles etc., provide a more natural way of interaction. But the support in computer programs is currently a big challenge, because a high effort is to invest for developing an application that supports such alternative input devices. For this fact we made a concept for an interaction system, which supports the use of alternative interaction devices. The interaction-system consists as central element a server, which provides a simple access interface for application to support such devices. It is also possible to address an abstract device by its properties and the interaction-system overtakes the converting from a concrete device. For realizing this idea, we also defined a taxonomy for classifying interaction devices by its interaction method and in dependence to the required interaction results, like recognized gestures. Later, by using this system, it is generally possible to develop a user-centered system by integrating this interaction-system, because an adequate integration of alternative interaction devices provides a more natural and easy to understand form of interaction.

Keywords: Multimodal Interaction, Human-Centered Interfaces, Human-Computer-Interfaces, Gesture-based Interaction.

1 Introduction

Offering the use of modern interaction devices on technical systems is an actual trend for regarding customers' needs to provide an easy and intuitive interaction. Especially games are predominantly designed for playing them with other controllers than traditional once, like joystick, gamepad or the combination of mouse and keyboard. In 2006 the Nintendo Wii was published and had an amazing success because of its intuitively useable control paradigm by performing natural gestures. Going one step further Microsoft's Kinect introduce a full body gesture interaction by using only the own body as controller device. But not only systems for playing games try to utilize natural interactions, also systems like mobile phones using multi touch are successful like Apple's iPhone or other modern smartphones. All of these systems have one feature in common: they support a natural interaction by supporting gestures.

On the computer nowadays a gesture-based interaction is even not so successful. On these systems the traditional interaction devices mouse and keyboard are the most often used devices for the control of applications. Only multi-touch monitors are in some usage scenarios used for an easier interaction e.g. in public domains. Mostly the reason is the missing support in programs and applications. But also if alternative interaction devices are supported, their usage may not be adequate in all use case scenarios of a program. In different use cases, different interaction metaphors are needed to provide a useful interaction. For example by presenting a picture, abstract gestures are useful to instruct the viewer-program to zoom or rotate the picture. But if the user navigates through the menu of such a program only simple gestures are appropriate like pointing an entry or panning the display in a direction. Furthermore interaction devices often provide additional technical features e.g. the Nintendo WiiMote controller contains accelerometers, which are useful for supporting a gesture-based interaction. But the WiiMote also utilizes an infrared camera, which allows it to be used as pointing device. For an adequate support of modern interaction devices all the possible interaction methods and paradigms should be supported, but it is a challenge to address different kinds of interaction methods, if multiple modern interaction devices are used.

In this paper we introduce a taxonomy of possible interaction methods for existing interaction devices. In this taxonomy currently applied interaction methods are presented, based on nowadays available interaction systems and devices. In this paper we also describe a generic interaction analysis system we had developed, which handles different interaction devices and its supported forms of gestural interactions, basing on the defined taxonomy for interaction methods. Hence the system handles the technical aspects like detecting gestures, or adopting e.g. the pointing mechanism of the WiiMote to a physical display. By using this interaction analysis system, a developer does not need to spend effort in developing a gesture recognition system any more. Via an API the developer can also declare, which interaction method should be supported or be disallowed in the programs use case scenarios. The interaction analysis system automatically determines the supported devices and its technical aspects to use the relevant sensors. Of course the support of multiple interaction methods is provided, so different kinds of interaction methods are possible in a use case scenario.

2 Related Works

For classifying interaction devices to specify existing input and output devices, different approaches are existing. Especially for input devices classifications exist, to provide the possibility to group devices in dependence of similar properties. In the following section we give a small overview of existing classifications for input devices.

One of our goals is to use the classification for conceptualizing and developing an interaction-system, for this fact we also introduce in some existing interaction systems, which provide the feature for an intuitive usage.

2.1 Classifications for Interaction Devices

In the past some effort was done in defining classifications of input devices, mainly with the goal of better understanding of the devices, so that optimal devices could be found for specific tasks or similarities in devices to replace a device by another adequate one [1].

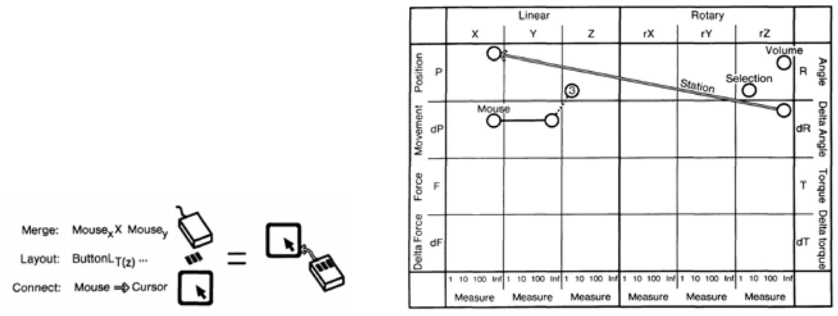
One of the first classifications was a taxonomy, which was defined by Foley et. al. [2]. The classification was focused on graphical user interface applications and its typical tasks that can be performed. Tasks were for example selecting a graphical object, orienting the mouse etc. Foleys taxonomy differentiates between devices that can perform these tasks and in what way they can do it, especially if its active principle is directly or indirectly.

Buxton et. al. [3] made the observation that there is a major difference in the way devices can produce the same output. Buxton called these differences pragmatics and divided the devices further into the way they produce the output (position, motion, pressure, etc.) and ended up with a taxonomy, which could represent simple devices.

To finally be able to classify all input devices, even virtual ones, Mackinglay [4] took Buxton's taxonomy and build a formal approach, which can be used to classify the most input devices. Mackinglay describes an input device with a 6-tuple:

$$T = \langle M, In, S, R, Out, W \rangle$$

M…	is an operation manipulator that describes, which value is changed by the device e.g. Rz would mean rotation around the z-axis
In…	is the input range in which the device can be manipulated e.g. a touchscreen would have a input range of [0, touchscreensize]
S…	is the actual state of the device
R…	defines a function that maps between In and Out
Out…	is the output range which results from In and function R
W…	describes the inner function of the device



(a) Merged operations (b) combined devices in Buxton's taxonomy

Fig. 1. Mackinglay's Taxonomy of input devices [4]

These tuples can be combined by 3 different merge operations to form more complex input devices:

- Connect: connects one device from *Out* with a fitting input device from *In*
- Layout composition: indicates the position for two simple devices at a superordinate device
- Merged composition: similar to layout composition, but the merged values defining a new complex data type

It is hard to insert devices for video or audio-based recognition in these taxonomies. In an alternative approach Krauss [5] divides devices on the super level in coordinate-based and non-coordinate-based interaction devices. So its classification (see Fig. 2) is driven by the output values of the devices.

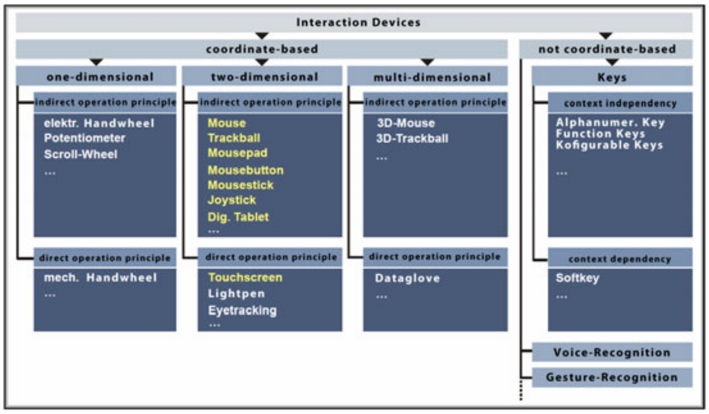


Fig. 2. Classification of input devices by Krauss [5]

2.2 Gesture-Based Interaction Systems

Some effort was invested to support a gesture-based interaction to normal computers and into application for allowing an easier control. In particular, the use of interaction devices from game consoles becoming also interesting for the use on normal computers. The reason is that they are well known and next to its easy usage, they provide different methods for interacting with a system. Early interaction devices like old data gloves were designed for only one interaction scenario. But in difference, a modern controller like the Wiimote, which is an input device developed by Nintendo, can recognize acceleration in 3 directions and it therefore can be used for gesture-recognition, next to use the infrared camera for the use as pointing device or use the buttons directly to control an application. To use only the WiiMote on a computer, different implementations are available. One of the most flexible programs for using the WiiMote as gesture-based device is WiiGee[1]. The recognition is implemented by a statistical comparison of the acceleration inputs and comparing them with previously trained acceleration data. WiiGee can classify the observed sequence as gestures [6][7].

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

There are also video-based applications that recognize faces [8] and some early approaches to recognize human gestures in video-based systems [9]. A more advances approach is Microsoft's Kinect[2] which uses stereo images to recognize gestures.

Speech recognition is a field, which gets more and more involved and works well for limited vocabulary. Many systems like cellphones, car navigation systems, operation systems and computer games support speech recognition.

3 Classification for Interaction Methods

In a first idea for creating a gesture-interaction system (see Fig. 3) that supports different kinds of alternative interaction devices, we focused on supporting different kinds of devices [10]. During this work we recognized that many devices can be used for different interaction methods, for instance the WiiMote is an accelerometer-based device, which can be used for gestures. Next to this it utilizes an infrared camera, which allows the use as pointing-device and it is possible to use it as key-based controller, because of the also provided buttons on the top of the controller.

Furthermore for some interaction methods various other devices do exists, which allows the same form of interaction. Representatives for 3-dimensional accelerometer-based devices are the WiiMote, Playstation Move Controller and most of the currently existing smartphones on the market.

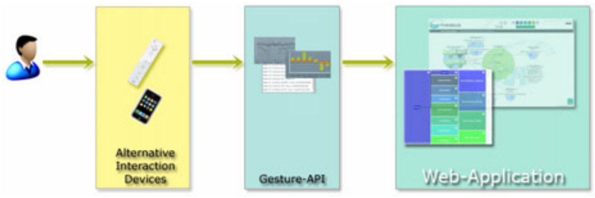


Fig. 3. Architecture of our Gesture-Interaction-System [10]

In default, for every supported controller another implementation is necessary, also for modules, which are able to e.g. recognize gestures. This is not an adequate procedure, so that we created another approach, which we are going to describe in the following sections.

For creating an effective tool to support also new upcoming alternative interaction devices, we need a concept for organizing them. Also to allow the use of modules for multiple times, like for gesture recognition module. In fact of this, we designed a multi-dimensional classification of interaction devices, grouped by the interaction method and the interaction result.

3.1 Classification of Interaction Methods

In the past times interaction devices are designed for a single use scenario, e.g. a 3D-Mouse was designed for using it to interact directly within a virtual 3-dimensional

[2] http://www.xbox.com/de-de/kinect

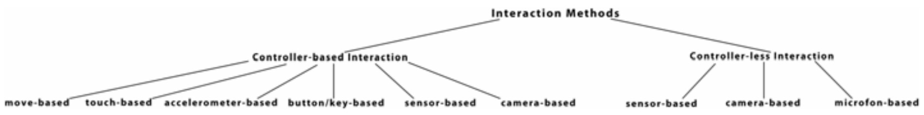


Fig. 4. Classification taxonomy for the possible interaction methods

world. In difference to these devices, the modern interaction devices for modern game consoles etc. utilize multiple technologies. The Playstation Move controller integrates so accelerometers, a gyrocopter, a compass and buttons, which in sum allows a manifold interaction style.

For grouping interaction devices, we conceptualized a taxonomy to group the controller by its basic underlying technology (see Fig. 4). Of course, every interaction device can be ordered in multiple groups – depending to the required technical feature.

This classification is necessary to provide the possibility that a developer etc. can address which kind of interaction method he wants to support or what kinds of interaction devices he wants to allow interacting in his application.

The important difference between this characterization and the existing approaches for classifying interaction device is that we separate the devices by its technical use method and what kind of sensors etc. are responsible for this method.

3.2 Classification of the Interaction Results

In a practical use a user or furthermore a developer has a specific requirement on the results by the usage of an interaction device. So if an interaction in a graphical user-interface is planned, the result from the used controllers must be coordinates etc. This can only be ensured, if the taxonomy provides the feature of filtering the devices by its generated result data.

By interacting through a graphical interface the user can use different kinds of interaction devices, in dependence to the needed results e.g. coordinates for a 3-dimensional environment. Today only 1, 2 or 3-dimensional environments are known. But in fact of further research approaches or the use of e.g. multi-dimensional

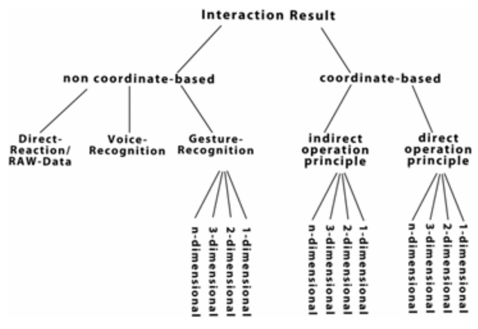


Fig. 5. Classification taxonomy for addressing the interaction result by previously defined interaction method

combinations of multiple 3D environments, we regard also n-dimensional interaction results, but of course today we found no real n-dimensional (n>3) interaction device or an environment, which provides the need of such an interaction device.

3.2 Usage of Interaction Devices by Classifying Interaction Devices

Our concept of the taxonomy for classifying interaction methods to support intuitive interaction devices is a combination of the two described sub-classifications. The result of coupling both classifications is an array in which every interaction device (restricted to input devices) can be classified (see array in Fig. 5).

With this array it is possible to determine interaction device with similar features. Next to the feature of addressing interaction devices that supports a required interaction method and generates a preferred interaction results, like complete coordinates within a graphical user-interface.

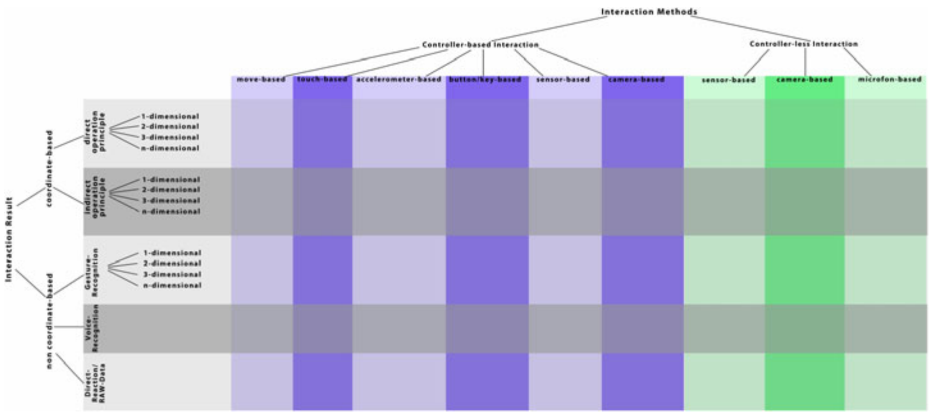


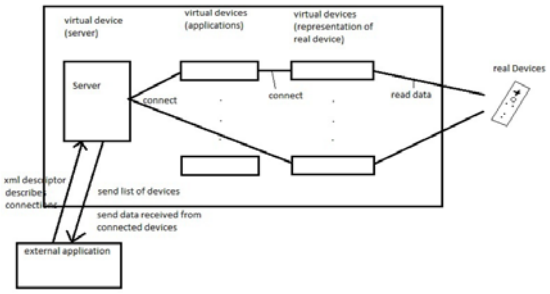
Fig. 6. Entire Classification Taxonomy for classifying interaction methods to support intuitive interaction devices, in which every input device can be arranged

4 Concept for an Interaction-System

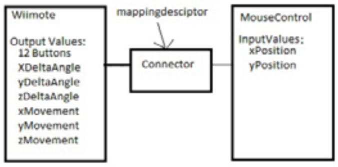
To create an extensible interaction system, we conceptualized a system, which regards the presented taxonomy. One of the most important features of the conceptualized system will be the possibility to address devices with a specific interaction metaphor and with a special return value. This enables the chance that later further applications can be developed, which will use the interaction-system and over this they can select the kind of the preferred way to interact within the application. On this way the developer of an application can for instance define, that every kind of accelerometer-based devices will be supported for a gesture-based interaction. The consumer has now the choice to use such a device, but it does not matter, if it will be a WiiMote, a Playstation Move controller or a Smartphone with accelerometers.

To achieve such a system and also to support the aspect of recognizing gestures with a device, every field of the array equates to an abstract device. This will be achieved in modeling every device as a virtual device. The treatment of the data e.g. the final coordinates or the recognition of a performed gesture will fully be overtaken by the interaction-system. Over an API the application is able to commit information to the server like the enabled kind of interaction devices or further supported gestures. The application gets as return value the coordinates or the final performed gestures. So the interaction-system is saving a lot of effort for developers of applications, who wants to enable a gesture-based interaction within his application or program.

Fig. 7 shows the overall concept of our system. Another application can communicate with our system by sending xml messages over the network layer. Over the connection the system sends an xml message that lists all available devices, and expecting after this an xml message, which describes the devices that should be operated and how they should be connected. Every device is implemented as a plugin, which the server can automatically load. This makes our system generically to support further and also new upcoming devices like Microsoft Kinect.


virtual device (server)
virtual devices (applications)
virtual devices (representation of real device)
real Devices
Server
connect
connect
read data
xml descriptor describe connections
send list of devices
send data received from connected devices
external application

(a) Architecture illustration for the interaction-system



(b) mapping of the device data to the screen

Fig. 7. Architecture and data mapping of interaction devices by the Gesture Interaction System

The system consists in general of 3 components. The central and most important component is the server of the interaction-system, which organizes all available interaction devices and also the modules for the supported interaction methods. Next to this central system an API library is available, so that a developer of a program is able to use the interaction-system. The API is planned for C# (.NET) and for Adobe Flex, which allows also web-application to support modern devices. The third component will be a system for providing the possibility to configure new gestures. So this Learning-Tool is only designed for modes where a gesture-interaction should be used. So developers etc. can train new gestures for controlling their application. The Learning-Tool generates an XML-configuration file, which can later be committed by the API to the server of the interaction-system. On this way the server will know the new gestures and is able to recognize them.

5 Discussion

This interaction-system is an additional system to support alternative interaction devices. Our main research scope lay in an adaptive visualization framework for visualizing semantics data. To enable an intelligent user-interface we have to support intuitive information visualization on the hand, and on the other hand we have to provide an easy form for interacting through visualizations from the input side. For this circumstance we need an interaction-system to delimit a specific method of interaction (and a specific type of result value, like a performed gesture or coordinates) and the used interaction method e.g. by accelerometers. This is especially necessary, if another kind of navigation should be supported then only a coordinate-based once. For example, how is it possible to provide a gesture-based interaction through graphs? The challenge is that the navigation to its direction is harmful, because if e.g. 5 nodes are positioned on top right, it is hard to determine which node has to be selected, if a gesture to the top, right direction is performed. So an alternative approach is required for providing an adequate gesture-based interaction through graphs. Similar to this kind of visualization, it is also hard to find a gesture-based approach for timeline and geographical visualizations.

Another point for discussions is the completeness of the taxonomy. We currently applied it on actual existing input devices especially from game consoles. We also try to regard most kinds of smartphones. From all these devices we determined its utilized features for differentiate its interaction methods and form. Then we tried to determine the possibly results, which are required in external applications. In the presented form, we cannot ensure that all kinds of interaction are regarded. We supported all common kinds of interaction, but it is possible that also new approaches have to be regarded and so an extension of the taxonomy has to be taken.

6 Conclusion

In this paper we introduced into our defined taxonomy for classifying interaction devices in dependence of its interaction metaphor and its interaction results. In difference to other existing classifications, our approach is primary driven by the usage in adaptive semantic visualizations and therefor to select specific kinds of interaction devices or devices with specific forms of interaction. Next to this, the taxonomy groups devices with similar forms of interaction, which helps to abstract these devices and its significant properties and provide so the possibility to develop e.g. gesture-recognition modules for such an abstract device and finally also all related devices which are grouped under this abstract device.

This idea for classifying interaction devices is used for our interaction system, which allows the use of alternative interaction devices like WiiMote or the Playstation Move controller. With this system every currently available input device can be classified by its utilized interaction method and in dependence to its interaction result e.g. coordinates or an identified gesture. The interaction-system consists of two parts, the central server, which organizes the devices and provides an interface for external applications. And also the API for the use within application that communicates with the server and gets the results of interactions from the connected devices.

The conceptualized system can be extended by further and also new upcoming input devices and also for the determined input results further modules can be developed to support e.g. different kinds of gesture-recognition. This can be useful if an accurate recognition is required for normal interactions in application as well as a less accurate real-time recognition like it is needed in games.

But in general it is possible to develop a user-centered system by integrating this interaction-system, because an adequate integration of alternative interaction devices provides a more natural and easy to understand form of interaction.

Acknowledgements. This work was supported in part by the German Federal Ministry of Economics and Technology as part of the THESEUS Research Program. For more information please see http://www.semavis.com.

References

1. Jacob, R.J.K., Sibert, L.E., McFarlane, D.C., Mullen Jr., M.P.: Integrality and separability of input devices. ACM, New York (1994)
2. Foley, J.D., Wallace, V.L., Chan, P.: The human factors of computer graphics interaction techniques. Prentice Hall Press, Englewood Cliffs (1990)
3. Buxton, W.: Lexical and pragmatic considerations of input structures. ACM, New York (1983)
4. Mackinlay, J., Card, S.K., Robertson, G.G.: A semantic analysis of the design space of input devices. L. Erlbaum Associates Inc., Mahwah (1990)
5. Krau, L.: Entwicklung und Evaluation einer Methodik zur Untersuchung von Interaktionsgerten für Maschinen- und Prozessbediensysteme mit graphischen Benutzungsoberflächen. Universität Kaiserslautern (2003)
6. Poppinga, B.: Beschleunigungsbasierte 3D-Gestenerkennungmit dem Wii-Controller. University of Oldenburg (2007)
7. Thomas Schlomer, N.H.S.B., Poppinga, B.: Gesture Recognition with a Wii Controller. In: 2nd International Conference on Tangible and Embedded Interaction (2008)
8. Zhao, W., Chellappa, R., Phillips, P.J., Rosenfeld, A.: Face recognition: A literature survey. ACM Comput. Surv. 35, 399–458 (2003)
9. Mitra, S., Acharya, T.: Gesture Recognition: A Survey. IEEE Systems Man and Cybernetics Society (2007)
10. Burkhardt, D., Nazemi, K., Bhatti, N., Hornung, C.: Technology Support for Analyzing User Interactions to Create User-Centered Interactions. In: Stephanidis, C. (ed.) UAHCI 2009. LNCS, vol. 5614, pp. 3–12. Springer, Heidelberg (2009)
11. Nazemi, K., Burkhardt, D., Breyer, M., Stab, C., Fellner, D.W.: Semantic Visualization Cockpit: Adaptable Composition of Semantics-Visualization Techniques for Knowledge-Exploration. In: International Association of Online Engineering (IAOE): International Conference Interactive Computer Aided Learning 2010 (ICL 2010), pp. 163–173. University Press, Kassel (2010)