

Gesture-based Interface Reconfiguration

Christian Kray
Computing Department
Lancaster University
Lancaster, UK, LA1 4YR
+44 (0) 1524 592344

kray@comp.lancs.ac.uk

Martin Strohbach
Computing Department
Lancaster University
Lancaster, UK, LA1 4YR
+44 (0) 1524 593104

strohbach@comp.lancs.ac.uk

ABSTRACT

In this paper, we propose a novel approach to build an intuitive, reconfigurable tangible interface. Our approach is based on a small set of simple gestures using a set of arbitrary physical objects, and their recognition by a table equipped with weight sensors. In our model, physical tokens or everyday items such as coffee mugs or mobile phones are associated to virtual controls of the user interface by means of two-dimensional gestures. The resulting relationship between physical and virtual entities is then used to control the application. As the user is able to dynamically change the associations, the interface can be easily reconfigured and thus personalised to individual users.

Keywords

Tangible user interfaces, interface reconfiguration, spatio-temporal reasoning, ubiquitous computing, weight sensors, tangible objects.

1. INTRODUCTION

Semi-public displays such as TV screens in domestic environments increasingly evolve to central control centres for multimedia and home automation control. Several research projects currently investigate ways to facilitate user interaction with an increasingly complex set of (entertainment) devices (see, for example, [7]). In addition, a new generation of commercial products such as digital multimedia receivers or software such as Microsoft's Windows XP Media Center [9] is on the verge of entering the mass market, and will provide the infrastructure for new and interesting applications around the home.

However, standard input devices such as keyboard and mouse are not well suited for this environment, since they clutter living room space and are awkward to use in settings such as the kitchen or a bedroom. Similarly, remote controls are neither appropriate as they are often badly designed, bound to a specific device, not customisable for individual users and tend to get lost.

Tangible or graspable interfaces try to address this issue by designing new devices that seek to provide a more natural and intuitive way for application and device control. However, in a domestic setting they suffer from similar problems as traditional input devices such as remote controls: Dedicated control devices that have a fixed control functionality have to be present at the time and place a user wishes to interact with a system. This may seem obvious but if one considers a user moving around his house and interacting with various applications on his way, such devices would either have to be carried around or available at all locations.

In order to address this problem, we present a system that relies on mobile everyday items and an augmented table to enable user interaction with arbitrary services. In section 2, we provide a short overview over related work and basic issues. Our approach to address these problems is the main topic of section 3, where we introduce a set of basic two-dimensional gestures as well as their mapping to interface actions. In section 4, we review two example applications before presenting an outlook on future research in section 5. Section 6 summarises the main contributions of this paper.

2. RELATED WORK

While traditional interfaces such as graphical user interfaces (GUI) are well suited for a wide range of tasks, there are some issues that make them less desirable in a casual scenario such as the living room. On the one hand, they require some form of infrastructure, e.g. a display or a remote control. On the other hand, some are hard to learn. Consider for example the number of remote controls in an average living room and the number of (inconsistent) buttons across them. Hardly any user can access the full functionality of all their entertainment devices. Even if there is some form of a graphical interface, these are often awkward to use and highly complicated. In addition, customisation or adaptation of the interface is not supported, so that a user cannot tailor the interface to meet his specific preferences. [7].

Tangible interfaces (TUI) [2] have the potential to overcome these shortcomings. While such interfaces usually are very unobtrusive and provide a more intuitive way to access the functionality offered by a system, there are some challenges that need to be addressed. On the one hand, we have to find a mapping between tangible objects and gestures in the physical world and the corresponding entities and actions on the application side – i.e. in the virtual (or physical, e.g. home control) world. On the other hand there is a tendency that TUIs use either single artefacts for generic application control [11] or multiple physical devices that are designed for a specific application or application domain [14] [2]. The more generic tangible interfaces usually lack good integration into everyday environments and share many of the disadvantages of remote controls (e.g. loss, have to be carried around).

Consequently, there is a need for a 'light-weight' tangible user interface that is not only unobtrusive but also accessible in an intuitive way. Furthermore, it should support adaptation or reconfiguration of the interface according to the user's preferences and situation. The following section describes such an interface based on two-dimensional gestures and a single 'augmented object'.

3. A RECONFIGURABLE TANGIBLE INTERFACE

Our solution is based on a single augmented object, a weight table that is able to recognise a set of two-dimensional gestures performed with arbitrary objects. These gestures are used for configuring each individual user’s interface, i.e. assigning the user’s choice of physical objects to their virtual control element. Once this assignment has been made the gestures of individual objects can be recognised and control the application. We intend to use plan recognition as means for improving the efficiency of the interface and smart objects for persistent associations between personal objects and control elements. In total this approach allows the user to freely choose the tangible objects himself, thus improving the intuitiveness of the interface. On the application or service side, there is also a great flexibility: Depending on the type of application, different levels of the interface processing can be accessed. Figure 1 provides an overview of the major components of our tangible interface that will be described in the following subsections in more detail.

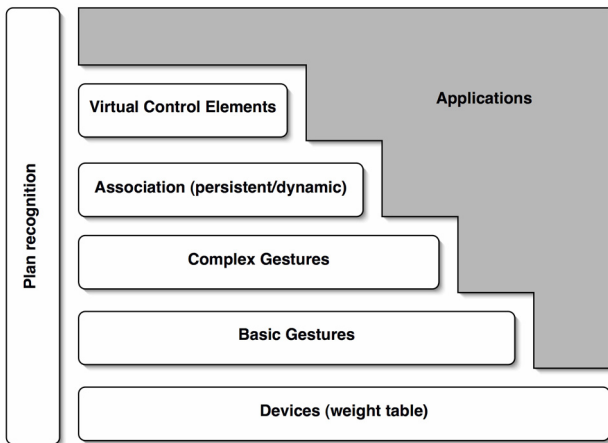


Figure 1: General architecture

3.1 The Weight Table

The basis of our tangible interface is a single augmented object, the weight table. The weight table is an off-the-shelf coffee table that is unobtrusively augmented with four industrial load cells at each corner. Each load cell can reliably measure a maximum weight of 1kg. A mechanical overload protection prevents damage to the load cells. The load cells are interfaced by a Smart-it, a special wireless sensor board [4] which is programmed to recognise the two-dimensional position of objects that are placed, moved and taken off the table, as well as their weight.

The accuracy for locating objects on the surface lays within 2cm for virtually all the objects. In [13] we found that the tracking accuracy of the current implementation is comparable to a serial mouse (i.e. on a standard coffee table).

The weight table can also detect the location of pressing and releasing events. A more detailed description of the table and its capabilities can be found in [5] and [6]. Figure 2 shows a current version of our weight table.

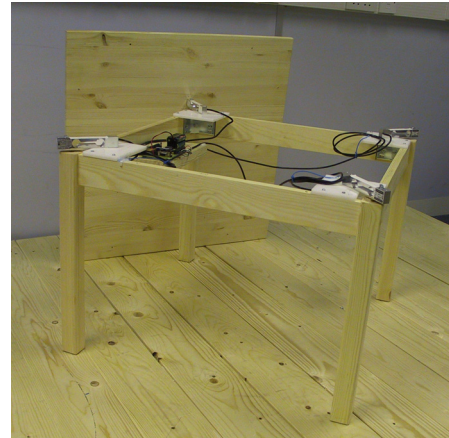


Figure 2: The weight table

3.2 Gesture recognition

The capabilities of the weight table can be used to recognise a basic set of two-dimensional gestures as shown in Table 1. The PUT_DOWN event occurs when the user puts an arbitrary object on the table. The corresponding event to that gesture provides information about the location on the table and the weight of the object. PICK_UP is the complementary event that is detected when an object is removed from the table. It also provides positional and weight information. As demonstrated in [1] the recognition of these gestures works highly reliably. We use PUT_DOWN and PICK_UP mainly for configuring the interfaces as described in section 3.4. The PRESS and RELEASE events correspond to actual user control gestures, i.e. an increase or decrease in pressure. This gesture does not discriminate whether an object is used or if the interaction with the table happens by just using the finger. However, if the position of an object is known (i.e. it was placed on the table at some point) we can relate a PRESS or RELEASE event to a specific object.

Event(EventData)	Gesture Description
PUT_DOWN(x,y,w)	User puts an object on the table.
PICK_UP(x,y,w)	User removes an object from the table.
PRESS(x,y,p)	User increases the pressure on the table.
RELEASE(x,y,p)	User decreases the pressure on the table.

Table 1: Basic gestures and their events

Similar to PUT_DOWN and PICK_UP both PRESS and RELEASE provide positional information. However, the pressure value is not an actual weight but an indicator of the constant pressure the user performs on the table. Technically it is a filter over the sampled load cell data that is still varying. The varying weight is also how we distinguish for example the PUT-DOWN event from the PRESS event. The pressure information can be used for discrete control e.g. pressing a button in a GUI or for continuous operations like volume control.

More complex gestures can be derived by combining several of the above-mentioned primitives, e.g. for detecting the movement across the table surface. Table 2 lists the corresponding events.

Event(EventData)	Uses	Gesture Description
TRACE_START(trajjectory,p)	PRESS	User starts moving (an object) across the table.
TRACE_STOP()	RELEASE	User stops moving (an object) across the table.

Table 2: Complex gestures and their correlation to basic gestures

The TRACE_START event is detected by a succession of PRESS events with changing coordinates. After the event occurs the trajectory is streamed to the other components of the interface (Control Association, Plan Recognition, Controls). The stream consists of the coordinates provided by the PRESS event and the corresponding pressure value. The streaming stops when a RELEASE event occurs which is then communicated as TRACE_STOP event.

It should be noticed that the PRESS, RELEASE and TRACE events do actually not rely on using physical objects. The users could also use their fingers or whole hand to perform these gestures. However, in an exploratory study we found that users prefer to use objects when interacting with the table since they feel more comfortable that way [13]. Furthermore, in the interface presented in this paper we exploit the use of different objects as symbols to the user for the different virtual controls as demonstrated in the example applications (see Section 4).

3.3 Virtual Control elements

We assume that our applications can be controlled by a set of standardised software components. We will refer to these components as virtual *control elements*, or *controls* for short. Each control provides a restricted vocabulary to control parts of the application, e.g. BUTTON_PRESSED as the vocabulary of a widget in a GUI or VOLUME_UP as part of the vocabulary of a volume control. These examples show that controls may have different levels of abstractions. In fact, there is a tendency towards higher abstraction levels, e.g. the XUL interface description language [15] or Microsoft’s XEEL [10].

3.4 Dynamic associations

As we have illustrated in the section 3.2, the weight table supports a number of basic and more complex gestures for arbitrary objects. While these already allow for the mapping to some basic interface actions (such as pressing a button by pressing on the table), we still require some intuitive means to associate a physical object to a virtual control element and to modify or cancel such an association. We propose to use the basic gestures shown in Table 1 as an ‘alphabet’ from which more complex ‘words’ can be constructed that correspond to more complex actions such as the establishment of a new association. The only addition to this alphabet is a predicate that encapsulates a ‘non-event’ i.e. the state of the table or an object does not change over a specified time. The following three complex gestures can then be used to handle associations:

- **PUT-DOWN, NO-EVENT(long)**
This gesture establishes an association between the

physical object being put down at a location and a control element of the virtual interface.

- **PICK-UP, NO-EVENT(long)**
In order to cancel an association, the user can pick up an object that has been assigned to a control element, and then keep it off the table for a longer period of time.
- **PICK-UP, NO-EVENT(short), PUT-DOWN**
This gesture allows for the reconfiguration of the (virtual) interface. When the physical object is placed at a different location, so is the virtual control element within the virtual interface.

It is worth mentioning that the first two complex gestures already allow for a simple form of reconfiguration – namely the re-association of an object to a different control. If the user picks up an object that is associated with one control, and after some time puts it down at a different place, the association is first cancelled, and then a new association to a different object is established. The third complex gesture, however, maintains the association between the physical object and a control element while modifying the virtual interface.

These complex gestures require a mapping function that maps physical locations on the weight table to locations in the virtual interface. In the case of a GUI, for example, a simple coordinate transformation from the table surface coordinates to the screen coordinates could be used. The proposed approach not only provides a very simple and intuitive means to handle dynamic associations but also leaves the basic gestures to basic interface actions. For example, pressing down an object can be mapped to the activation of a control element, and moving an object can result in a gradual change of the controlled concept.

3.5 Accidental association

As the weight table should also serve its original purpose as a piece of furniture, we have to avoid establishing associations between objects and controls that were not intended by user. The greatest problem may result from objects that are put accidentally on a location on the table, which corresponds to an unassigned control. E.g. putting a book on this location would associate it with the volume control. Moving the book on the table could then result in undesired effects on the application such as increasing the volume of background music.

One way to avoid accidental gestures is to perform specific gestures or to use dedicated objects that distinguish between a *configuration* and *control mode*. Section 4.2 demonstrates this approach for universal multimedia control application. A second approach requires a second gesture for confirming the association, e.g. by pressing the object for two seconds on the table. Another approach could try to infer the users’ intentions. See Section 3.7 on how plan recognition can help avoiding accidental associations.

Furthermore it is important to provide feedback mechanisms to the user to inform him about the establishment, cancellation and modification of associations. In the case of a GUI, a highlighted icon representing the control could signal the establishment of an association while a greyed out icon would indicate the cancellation of an association. When the interface is being reconfigured, the corresponding control elements could be

rearranged on the display. In connection with special gestures these feedback mechanisms could then allow the user to disassociate links to unintended control elements.

Finally, it should be noted that for objects that are associated with a control, the user can be assumed to be aware of their control functionality as the association has been explicitly established by the users. We therefore do not expect them to accidentally perform gestures that are not intended to control the application. However, the object can still be used for its original purpose by lifting it from the table as explained earlier in this section (see also Section 4).

3.6 Persistent associations

The described gestures are well suited for arbitrary objects – users can compose interfaces using arbitrary physical objects. However, it is possible that one user may want to associate his coffee cup with the scrolling control of his browser whereas another wants to use it as a back and forward button. However, there are personal items such as mobile phones, coffee cups, key fobs and wallets that can be associated with a control element permanently to avoid the configuration at each time before usage.

Although distinction of objects as in [8] would be theoretically possible, it is not practically feasible in our setting. There would be increased requirements to the accuracy of the load cells to distinguish a potentially unlimited set of objects. Furthermore, many objects such as drinking glasses change their weight.

Therefore, we can facilitate persistent associations by using smart objects - objects that are augmented wireless sensor nodes. The objects themselves need to recognise that they have been put on the table. In [1] we have demonstrated how smart objects such as glasses and jugs augmented with smart-its and pressure sensors can interact with the weight table.

Event(EventData)	Gesture Description
PUT_DOWN(objID,x,y,w)	User puts an object on the table.
PICK_UP(objID,x,y,w)	User removes an object from the table.
PRESS(objID,x,y,p)	User increases the pressure on the table with a dedicated object.
RELEASE(objID,x,y,p)	User decreases the pressure on the table with a dedicated object.
TRACE_START(objID,trajectory,p)	User starts moving a dedicated object across the table surface.
TRACE_STOP(objID)	User stops moving a dedicated object across the table surface.

Table 3: Basic gesture events for Smart Objects

The virtual control elements maintain the association to the smart object. If smart objects are used, the gesture events also contain an object identifier that allows the controls to determine if an association is already established or not. Table 3 shows the modified gestures if smart objects are used.

Once the user has established a persistent connection he can use his personal smart objects in connection with the table right away for controlling the application. However, at any point in time users are able to re-associate different controls to the smart object. Section 4.2 demonstrates the use of Smart Objects in our tangible interface.

3.7 Knowledge-based reasoning

In Figure 1, we also listed a plan recognition component as being part of the architecture. In fact, comparing observed or inferred/abstracted events to a set of plausible plans within the context of an application may help to address several issues. On the one hand, it may help to disambiguate between several gestures that can be inferred from sensor measurements. For example, if the data from the load sensors provides support for either a PRESS or a TRACE_START event, knowing that the associated control does not afford an action corresponding to the TRACE_START may help to select PRESS as the most likely gesture. Similarly, if the user seems to be executing a complex plan such as scanning a web page for relevant information, certain gestures may be more likely than others even though measurement provides evidence for both.

On the other hand, a knowledge-based system may also support the process of recognising certain objects or to disambiguate between them. Consider, for example, two objects of similar weight, which have been persistently associated to two different functions. In this case, the current plan of the user may provide strong evidence for one of them, when a PUT_DOWN event with a similar weight is detected. Furthermore, plan recognition in general may enable a system to predict future actions of the user, which can either improve recognition or allow for proactive support of the user’s task.

Finally, plan recognition provides a means to address the problem of *accidental associations*, e.g. when the user does not want to access any service at all and uses the table just for putting things on it. In that case, he could accidentally associate one of those objects to a control or activate a function by using objects in the physical context. Instead of relying on dedicated modi (i.e. interface turned on/off) or special gestures or areas to control the general association behaviour, the system could resort to a plan library. The stored plans can then help the system to decide whether the actions it observes are more likely to be interface actions or artefacts of everyday use of the corresponding objects. In the latter case, the system can deactivate the interface function until it observes actions or gestures that can be mapped to a plan incorporating interface control (see also Section 5).

4. EXAMPLE APPLICATIONS

In order to illustrate the feasibility of the proposed approach, we will discuss two example applications: a simple web browser and a universal multimedia control. In both cases, we make some basic assumptions: There is a weight table along with several arbitrary objects, and we assume a direct mapping between the screen and the table surface. A direct mapping consists of a linear scaling of X/Y coordinates from table to screen and vice versa. For example, a mug placed in the top right corner of the table could establish an association with a control in the top right corner of the screen. More sophisticated mappings (e.g., using qualitative spatial relations) are discussed in Section 5.

4.1 Controlling a web browser

The process of ‘surfing’ the Internet is one of the most common tasks users perform on their computers. However, browsing the Web in the living room either requires a laptop or dedicated input means such as a mouse or a keyboard. While traditional devices are well suited for this task, we have already pointed out the disadvantages of using them in the living room. Hence, we introduce an alternative based on the weight table and the proposed event model.



Figure 3: A simple Web browser – green buttons are currently associated with a physical object

Figure 3 shows a simple Web browser with standard buttons such as ‘back’ or ‘forward’ as well as a scroll bar. When the user puts down his glass of water on the table at the location corresponding to the ‘back’ button, an association between the glass and the button is established. As the user should be informed about a successful association, the GUI should provide feedback, e.g. by flashing the border of the button and then highlighting it (to signal that the button is associated to a physical object). Then, the user can go back to the last visited site by pressing the glass on the table. Similarly, he can move a scroll bar by moving the object that was successfully associated to the scroll bar.

Adapting the interface to his current preferences is also straightforward. For example, if he uses the ‘reload’ button very frequently to refresh a news page about a sports event, he might want to have it closer to himself rather than on the distant side of the table. Assuming that it is assigned to his mobile phone, he can reconfigure the interface by simply picking up the phone, and placing it at the desired location. The GUI will reflect the change by moving the ‘reload’ button to the corresponding location on screen.

If the phone rings and the user picks it up to answer the call, and then talks for a while, he does not trigger any unwanted interface actions. The only thing that happens is the cancellation of the association between the mobile phone and the ‘reload’ button. When he finishes the call, he can put it down at the same location to effortlessly re-establish the original association. Thus, the interface does not interfere with the regular use of objects used for interface control, and can even support the parallel use, e.g. reading a book while using it as a button.



Figure 4: Multimedia user interface. Virtual control elements from left to right: device selector, channel selector, volume control, Play/Pause, Rewind, Forward.

4.2 Universal multimedia control

This application example describes how the weight table can be used as a reconfigurable tangible interface in a domestic setting for multimedia control. Figure 4 shows the reconfigurable GUI we use in this setting. The six coloured button display icons that represent the virtual control elements to the user. Initially the buttons are greyed out indicating that no physical object has been associated with the control yet. The table icon in the upper left corner of Figure 4 is used to activate the configuration mode in this example: if an object is present at that location, it indicates that the dynamic association is active.

The association between physical objects and virtual control elements is established by putting the object at the position on the table that corresponds to the position of the graphical representation of the widgets. As soon as the association is set up the icon is highlighted.

For smart objects the application reacts slightly differently. As the PUT_DOWN event also communicates an object identifier, the application checks if this object is already associated with a virtual control element. If there is an associated control element, the association is instantly active - independently from where the object has been put on the table. In fact, the GUI relocates the highlighted icon to a position on the screen matching the position on the table. If there is no associated control element, association works in the same way as with non-augmented objects. The only difference is that the user performs an additional gesture to confirm the persistent association. This gesture can be recognised by the table or by the smart object, e.g. pressing a button on the smart object, shaking it or moving it with a certain pattern on the table.

As described in the previous example and in Section 3.4 the objects can be instantly used for reconfiguring the interface (by relocating them on the table) as well as for their normal everyday use. Even if the application is still in configuration mode, relocating the object to a position of a non-associated control will not reconfigure the objects association. Instead the display will rearrange the graphical interface to match the new position and move the unassociated control to a new location.



Figure 5: Setting of the Table multimedia control

Finally, if the object on the configuration icon is removed the graphical representation of the interface fades out enabling the users to control its multimedia devices without graphical obstruction. They are also safe to use everyday objects, such as glasses and mugs on arbitrary positions on the table without causing the interface to reconfigure.

5. OUTLOOK

The proposed approach can be extended in several ways. One obvious way is the generalisation of the event mapping to a larger class of GUI control elements, and the definition of further events needed in the process. Furthermore, the application of the underlying principles to non-graphical user interfaces such as acoustic interfaces is a suitable extension.

In addition, we plan to conduct exploratory user studies to evaluate the intuitiveness and acceptance of the proposed approach. These studies may also provide information about the suitability of dedicated input devices or tokens as well as about the situations where persistent associations are helpful. Furthermore, such studies will enable us to evaluate how to properly discriminate and represent application control, configuration mode and non-controlling usage of the objects.

We also hope to strengthen the integration of reasoning techniques and the underlying model. In this context we also think of investigating how gestures can be mapped to the virtual control element vocabulary. E.g. the gestures could be learned to increase the intuitiveness for the individual users: the gesture for a volume control can be pressing, sliding or if a smart object is used even rotating the object.

A further important field is the investigation of accidental associations and gestures. Here, we plan to evaluate different means to address this problem, e.g. plan recognition, dedicated gestures for meta-control such as enabling/disabling the TUI as a whole, or the introduction of modi.

Additionally, we will investigate more sophisticated algorithms for the mapping between screen and table space. We are particularly interested in qualitative spatial relations [16] such as LEFT_OF or NEXT_TO, which promise to facilitate the task of identifying corresponding areas.

Finally, on a more technical level, the weight table provides means to measure load changes very precisely, thus supporting further gestures such as LIGHT_PRESS or SLOW_PUT_DOWN. We plan to investigate the benefits of an extended vocabulary in the near future.

6. CONCLUSIONS

In this paper, we proposed a novel approach for building an intuitive, reconfigurable tangible interface using mobile everyday items. We introduced the weight table – a standard table enhanced with load sensors – and defined a set of basic events that occur when using this table. Based on these basic events or gestures, we defined more complex gestures that are used to facilitate the control of an application and the (re)configuration of its user interface. We illustrated the feasibility of the approach by two example applications, a web browser and a universal multimedia control. In addition, we pointed out several future research opportunities based on the proposed approach.

7. REFERENCES

- [1] Holmquist, L.E., Antifakos, S., Schiele, B., Michahelles, F., Beigl, M., Gaye, L., Gellersen, H.-W., Schmidt, A. and Strohbach, M.: Building Intelligent Environments with Smart-Its. SIGGRAPH 2003, Emerging Technologies exhibition.
- [2] Ishii, H. and Ullmer B. Tangible Bits: Towards Seamless Interface to Access Digital Information. In Extended Abstracts of Conference on Human Factors in Computing Systems (CHI '01), Seattle, Washington, USA, March 31 – April 5, 2001, ACM Press, pp.187-188.
- [3] Smart-its Webpage. <http://www.smart-its.org>
- [4] Schmidt, A. Ubiquitous Computing – Computing in Context. PhD Thesis, 2003.
- [5] Schmidt, A., Strohbach, M., Van Laerhoven, K. and Gellersen, H.-W. Ubiquitous interaction - Using surfaces in everyday environments as pointing devices. In Proceedings of UI4ALL, Lecture Notes in Computer Science (LNCS), Volume 2615, N. Carbonell & C. Stephanidis (Eds.). Springer Verlag, 2002, pp. 263-279.
- [6] Schmidt, A., Strohbach, M., Van Laerhoven, K., Friday A. and Gellersen, H.-W. Context Acquisition based on Load Sensing. In Proceedings of Ubicomp 2002, G. Boriello and L.E. Holmquist (Eds). Lecture Notes in Computer Science, Vol 2498, ISBN 3-540-44267-7; Springer Verlag, Gothenburg, Sweden, September 2002, pp. 333 - 351.

- [7] Kirste, T., Herfet, H. and Schnaider, M. EMBASSI: multimodal assistance for universal access to infotainment and service infrastructure. In Proceedings of the Workshop on Universal Accessibility of Ubiquitous Computing: providing for the elderly, R. Heller (Ed.). ACM Press, New York, NY, USA, 2001, pp. 40 – 51.
- [8] Konomi, S., Muller-Tomfelde, C., Streitz, N.A. Passage: Physical Transportation of Digital Information in Cooperative Buildings. In: Streitz, N., Siegel, J. Hartkopf, V., Konomi, S. (Eds): Cooperative Buildings – Integrating Information, Organizations, and Architecture. Proceedings of the Second International Workshop (CoBuild'99). LNCS 1670, Heidelberg, Germany, Springer, 1999, pp.45-54.
- [9] Microsoft Windows XP Media Center Homepage. <http://www.microsoft.com/windowsxp/mediacenter/>
- [10] Microsoft XEEL announcement at WinHEC 2003. Microsoft Press Release. <http://www.microsoft.com/presspass/press/2003/may03/05-06winhec2003keynotepr.asp>
- [11] Rekimoto, J. and E. Sciammarella. ToolStone: Effective Use of the Physical Manipulation Vocabularies of Input Devices. In Proceedings of UIST 2000, 2000.
- [12] Rekimoto J., Ullmer B., and Oba H. DataTiles: A Modular Platform for Mixed Physical and Graphical Interactions, CHI2001, 2001.
- [13] Strohbach, M. and Sheridan, J. Exploratory user study investigating acceptance and use of the weight table as pointing device. In preparation, 2003.
- [14] Ullmer, B., Ishii, H. and Glas, D. mediaBlocks: Physical Containers, Transports, and Controls for Online Media. In Proceedings of SIGGRAPH '98, Orlando, Florida USA, July 19-24, 1998, ACM Press, pp. 379-386.
- [15] The Mozilla Project. XUL – XML-based user interface language. <http://www.mozilla.org/projects/xul/xul.html>
- [16] Cohn, A.G., Calculi for Qualitative Spatial Reasoning. In: J. Calmet, J.A. Campbell, and J. Pfalzgraf (Eds.), Artificial Intelligence and Symbolic Mathematical Computation. Berlin, Heidelberg, New York, Springer, 1996, pp. 124-143.