Towards a Playful User Interface for Home Entertainment Systems

Florian Block[1,2], Albrecht Schmidt[1], Nicolas Villar[2], and Hans W. Gellersen[2]

[1] Embedded Interaction Group, University of Munich, Germany,
block@informatik.uni-muenchen.de,
albrecht.schmidt@ifi.lmu.de
http://www.medien.ifi.lmu.de/ei/
[2] Computing Department, Lancaster University, UK
{villar,hwg}@comp.lancs.ac.uk

Abstract. In this paper we propose a tangible cube as an input device for playfully changing between different TV-channels. First we consider several design approaches and compare them. Based on a cube that has embedded gravity sensing and wireless communication capabilities a prototype is implemented. A 3D graphical representation of the cube is shown on the television screen. On each face of the cube a TV stream is rendered. The motion of the cube on the screen is connected to the rotation the user performs using the real tangible cube. Our hypotheses is that users can use the cube to browse between channels and to zap intuitively and playfully gaining a improved user experience even if the efficiency is limited compared to a remote control. We report on initial user feedback testing our hypothesis in witch we found out that users can easily use the cube without instructions and, despite technical limitations, see it as an improvement of current systems. Finally we discuss the issues that emerged from user's feedback.

1 Introduction

Current home entertainment products show a wide range of user interfaces and interaction styles. In particular remote controls of users – if well designed – an efficient means to operate such devices. Recent trends in mobile devices and networking show a development towards universal remote controls based on web technology and mobile devices, such as PDAs and mobile phones [1,2]. These approaches can offer a very efficient way for controlling and manipulation functions, however considering the situations in which home entertainment is used efficiency is not the only goal. In [3] Bill Gaver explains the concept of the *Homo-Ludens* and that design has to take this into account. Humans are described as playful and creative explorers of their environment and hence good design has to address these basic needs, too – without compromising the effectiveness of a tool.

We looked at the domain of controlling a television set (TV). From informal observations and reports we can conclude that changing channels is the predominant actions users are taking. The goal when users who change channels are manifold:

P. Markopoulos et al. (Eds.): EUSAI 2004, LNCS 3295, pp. 207–217, 2004.

- switching to a specific channel
- browse the program, getting an overview of "what is on" at the moment
- zapping (looking at various channels one after another for a short time)

In the case where the users browse or zap the efficiency of selecting a specific channel is a minor concern. Here the overall experience in using the device is at the centre. There are further functions where efficiency plays a more dominant role, e.g. switching on and of the TV, controlling volume, and switching views (television and teletext). Most other functions are to our knowledge used rarely in daily use.
In this paper we report on a project where we explore a specific design option that introduces a playful element to controlling channels on a TV. Our focus was on providing means for changing a channel in a playful and pleasant, but hence effective way. Other functions could be included in the interface but this is not discussed further in this paper. In the next section we briefly explain the idea how we assessed the concept by talking to users. This is followed by a detailed description of the implementation. We discuss then our use experience and initial user feedback. Finally this is concluded with a discussion showing general issues for building tangible user interfaces for this domain.

2 Idea and Initial User Feedback

The basic idea for a new user interfaces for changing TV channels resulted of informal observations of people watching television. Browsing and zapping is with many people by now a common way of using these systems. Our experience form the UK and Germany suggests that breaks for advertising and a fairly large number of channels available are factors that enforce this way of using a TV. When browsing and zapping it can be observed that people use the "next-channel" and step through all programs rather than selecting a specific channel. These observations lead to the idea to create a playful interface for channel selection.

The basic concept is to use a handy cube that allows changing the channels on TV by physical movement in a 3D-space. More specific a virtual version of the cube is shown on the television screen. On each phase of the cube a TV stream is rendered. The motion of the cube on the screen is connected to the rotation the user performs using the real cube. The user now can rotate the real cube in order to see the different sides and the TV channels respectively. If the cube is put down and not moved anymore the TV channel currently facing the user on the virtual cube is enlarged to cover the full screen. As soon as the user picks the cube up again the currently showing channel is resized back to the facing site of the virtual cube. Other channels are shown on the other sides of the cube.

Before implementing the system we talked to potential users to get an initial feedback. In particular we were interested in the following two issues:

- Is the idea of the interface intuitive and easy to understand?
- Do people like the idea of a playful interface for a TV even if it reduces efficiency?

The first question we approached showing people a sketch with a cube on a TV (which channel names on the sides) and a cube in a hand and asking how they think this UI works and how they would explore it. See figure 1 for a sample sketch. In this discussions we potential users we gained evidence that they could easily understand the concept and that they would do the right things to explore the interface.

After people understood the user interface we asked how they felt about drawbacks, especially with regard efficiency of direct channel select. For people this seems to be a minor concern as they generally do not find the task of selecting the channel time critical. One even mentioned that when zapping – the idea is to spend time – and hence playfulness is more important efficiency.

Based on this feedback we explored options for implementing such a system and to explore its usefulness and usability.

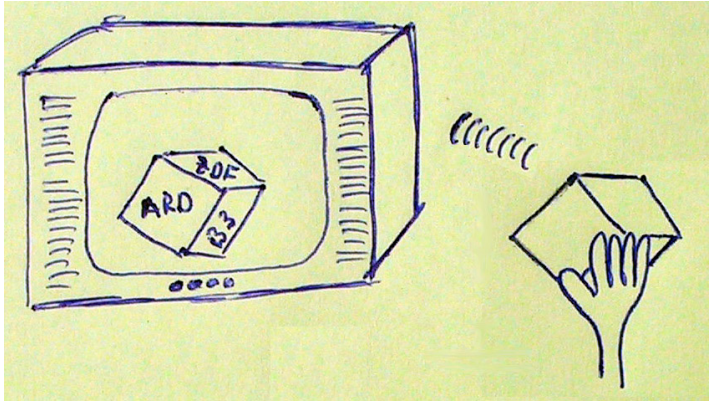

Fig. 1. To give users an idea what we would like to build we used a simple sketch that explains the basic concept. After people got the idea we asked initial user feedback.

3 Exploring the Design Space

Before implementing a full prototype we explored several design options. The first issue was related to finding a tangible object that is the physical manipulator and has a visual counterpart on the screen. A cube offered the affordances that proved to be most interesting. It can be easily handled by people, the manipulations (picking it up, putting it down, rotation and translation) are intuitive and it can be placed on any horizontal surface. Furthermore a cube offers clear faces to place the information (TV channels) on.

We explored further geometrical shapes, such as a hexagon, a cylinder, and a sphere. The hexagon seems an interesting object, this is as also explored by Butz et al. in [4], however we found the manipulations less intuitive. The hexagon also offers an easy way for presenting information on it. The sphere and the cylinder are very playful to use, however put them down a surface is a problem (the role away). Additionally visualization is discrete information units (such as separate TV streams) is more

difficult and there is no clear natural mapping. In figure 2 a sphere and a cube are shown next to the prototypical hardware that was used for sensing in the initial test. The prototyping tool is based on a Smart-Its hardware [5, 6]. Similar systems have been used in other projects, e.g. [7].

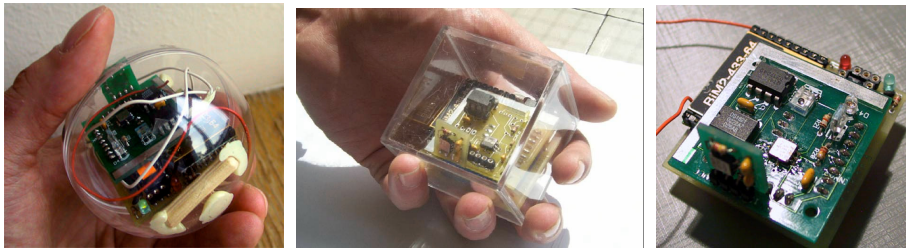

Fig. 2. In the initial design phase we explored different physical manipulation devices. The examples shown here are a sphere (left) and a cube (middle). These devices are built based on – Smart-Its - a platform for prototyping ubicomp applications.

For implementing the visualisation we chose to present a 3D model of the physical object manipulated by the user. For the representation of the TV channels we considered several options. The low fidelity would present only names and logos of the channels on the object. The high fidelity version should render continuously the current life TV stream on each face of the cube. An option in between is to have screen shots of all channels as representations. Given the aim to create a playful experience it appeared important to seek a high fidelity solution, even if this requires a number of receivers providing the TV feeds.

4 Implementation

To further explore this type of user interface we build a fully functional prototype of the system.

4.1 System Architecture

The overall system consists of two main components. The cube as tangible user interface is one component and the visual representation of the cube on the screen is the other. This is very similar to the sketch in figure 1. To make the system functional further components are required: the RF-receiver linking the tangible UI to the system, the driver code converting the sensor information into meaningful geometrical information, and the video source that can be mapped to the faces of the cube. In the following we explain these components in more detail.

4.2 Tangible UI and Receiver Unit

The hardware customized for the project is based on two Smart-Its that are wirelessly linked. The communication is via short range RF (up to 30m, FM) using a simple broadcast protocol. Full technical details can be found at [8].

Attached to the Smart-It included in the cube is a sensor board which is equipped with two accelerometers that are orthogonally adjusted. Each accelerometer includes 2 orthogonal adjusted sensors. The sensors used are accelerometers by Analog Devices (ADXL311) offering a measuring range of +/-2g. By using this arrangement, also depicted in figure 3, we get acceleration in three dimensions (X-Y and X-Z). The four resulting raw-data streams are constantly transmitted at a rate of about 30Hz per channel. The raw values are received by a second Smart-It which forwards the data over a standard RS-232 serial line to the computer running the visualization. Concepts for optimizing the design of a three-axis accelerometer are discussed in [9].

The raw values are collected by the input handling of the driver unit and passed to the data analysing unit. As one of the original four data streams is redundant only three of them are taken into account. To deal with inaccuracies several values are recorded over a short time and smoothened by applying a running average. The preprocessing causes a delay of about 50ms, but improves the data significantly.

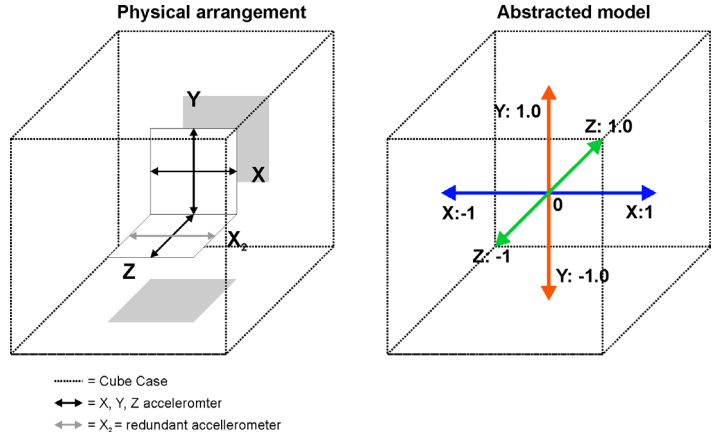

Fig. 3. On the left the physical arrangement of the two sensor devices inside the cube is depicted. On the right the abstracted view is shown.

Afterwards for each sensor orientation values are linearly converted to a floating point in the interval [-1.0, 1.0]. The minimum and maximum values are calibrated to the accelerometer's two extreme values caused by gravity (see Figure 3). The final resolution achieved is sensor-dependent approximately limited to resolution of 0.02 consequently to a total amount of 100 steps over the total calibrated range ([-1.0;+1.0]). The three converted values are paired in three different groups each with two elements (X and Y, X and Z, Y and Z). For each group it is now possible to calculate the angle between the gravity vector and one of the vectors in the group. This is achieved by applying the arctan function on the individual group quotients:

$$\alpha = \arctan\frac{x}{y}, \beta = \arctan\frac{x}{z}, \gamma = \arctan\frac{y}{z}$$

As x, y and z are values between -1.0 and 1.0 the arctan delivers angles between -180° and 180° giving a full 360° range, see Figure 4. In theory this results in 3 angles between gravity and one vector in each group.

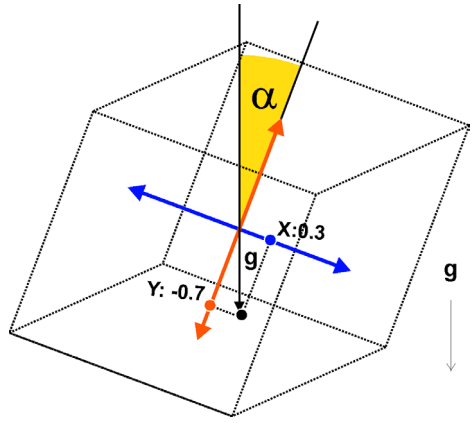

Fig. 4. If the components of the gravity vector are known the angle of the cube related to the gravity vector can be calculated.

Unlike the angles related to gravity, the rotation around the worlds Y-Axis (that is perpendicular to the floor) can not be measured by the sensors used in the prototype and is therefore simulated by a state machine. Once calibrated it keeps track of the according Y-rotation and can consequently decide which side of the real cube is currently facing the user. Based on that information the current world's Y Angle can be set in 90° steps. For details on the state machine algorithm see [10]. Analysing the gravity-related values gives the result that, depending on the side that is up, always one angle seems to have a very low resolution and in the worst case flicker in steps of up to 180°. Logically this is no problem as only 2 gravity related angles are needed. However, practically it has to be decided which of the three angles we employ in the final transformation and also to avoid using the flickering angle. Before defining an efficient selection strategy we need to explain the observed behaviour.

The effect is due to a simple mathematical relation of the x, y and z values. As long as the cube is not being translated, the three components together always form the g-vector which is, like mentioned earlier, calibrated to the absolute value of 1.0 ($x^2 + y^2 + z^2 = 1$). Looking at $x^2 + y^2 = 1 - z^2$ the geometrically interpretation would be that all x-y value pairs, plotted in a Cartesian coordinate system, lie on a circle with radius $1 - z^2$. The closer the value for z is to the absolute value of 1 the smaller the radius for the value pairs of the X-Y group becomes. As sensor data has a limited resolution and is not optimal this cases problems. Consequently this means the resolution decreases to 0 with z increasing to the absolute value of 1. Figure 5 shows

an illustration for an example resolution. Considering the angles – a lower resolution in values causes a lower resolution in angles and this explains the formerly mentioned flickering as the resolution is nearly 0.

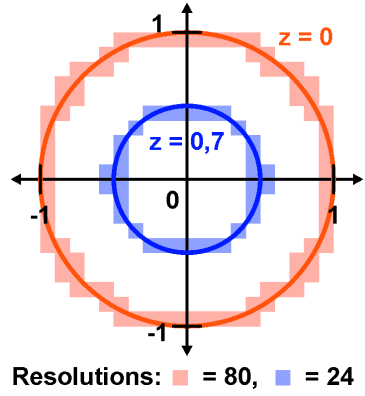

Fig. 5. The XY resolution decreases to 0 with z increasing to the absolute value of 1. The image shows an example with a decrease of resolution with a base resolution of 10 by 10.

For finding a strategy to employ the calculated angles correctly, we look at two different resolutions during a transition. We can approximate the resolution of a group with the current radius r by using the formula for the circumference:

$$res_{approx} = 2r\pi \cdot res_v$$, where res_v is the value resolution (in our case $res_v = 100$).

Looking at the approximated resolutions of the XY and XZ group in dependency of the z value (this corresponds to a rotation around the X-Axis, see fig 3 for illustration) gives the following progression of resolution, depicted in figure 6.:

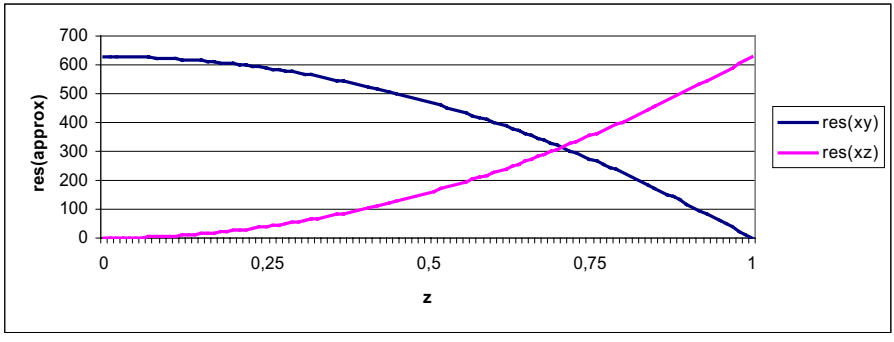

Fig. 6. Plotting the resolution of the of XY and XZ groups in dependency of the z-values.

Calculating the corresponding YZ-angle at the point of intersection delivers a value of exactly 45°. Repeating this procedure with all different values and cube states gives the significant transition angles of -135°, -45°, 45° and 135°. In the example depicted we exchange the XY-angle with the XZ angle, as soon as the YZ-angle crosses one of the given values.

This results in the following algorithm for handling the validity of the three angles. Assuming that there are three groups g1, g2, g3. Then the following two rules apply:

I) Exactly 2 groups must always be valid.

II) If a valid group's angle crosses -135°, -45°, 45° or 135° switch the validity state of the other two groups.

Now you always have two valid angles to mount on the worlds X and Z rotations and a Y angle given by the state machine. The state machine now sets up a basic transformation matrix bringing the cube its current state. After that the state machines decides on how to additionally apply the current valid angels on the worlds X and Z rotations. This matrix, applied to an arbitrary geometry would now transform the structure according to the real cube's rotation state. Eventually the final transformation matrix, the underlying raw and the average arrays are offered via the driver interface. Additional an event engine analyses the average history and calculates noise-level over a range of one hundred values. By also considering the calibrated sensor data it can decide, if the cube is lying on a plain object and fire the according events when it enters or leaves this state.

4.3 Screen Interface and Visualization

The Screen Interface generally consists of three software parts and a screen device. The input module access the data provided by the driver and forwards it to the program logic. Depending on the occurred events it chooses either full screen or zapping mode and additionally updates a global transformation matrix that is later used by the graphics engine to calculate the virtual cube. The program logic also assigns the six video sources that are for now just static.

A parallel thread renders the videos in the background and derives ready textures to be used in the final scene. To perform the final drawing we rely on the graphics engines 3D capabilities. It uses the transformation matrix, updated by the data acquisition thread, to calculate the virtual cube's geometry. Finally the different textures are accordingly projected onto the different sides. This is rendered into a three-dimensional environment covered by a two-dimensional descriptive layer containing instructional information.

Additional, each sign of the cube is numbered according to the real cubes enumeration. This is a helpful tool for recalibration as it shows the Y-rotation the state machine is currently assuming. An aberration of the real cube and the virtual cube thus indicates a mis-calibration.

The graphics engine uses Microsoft's DirectX because it supports existing hardware accelerators as well as various types of audio and video formats. Possible video-sources are all Windows-Media supported local or streamed file types like .AVI,

.MPEG, .ASF or .WMV. To provide TV life feed we have one stream available for each channel. Depending on the used graphics hardware, the final presentation can be output to all supported display types and resolutions.

4.4 Usage of the System

The user interaction with the interface is quite simple: The TV is turned on and as usual a channel is showing. The cube is lying next to the device and can be picked up at any time. As soon as this happens, the currently showing channel is scaled down revealing the three-dimensional environment including the virtual cube and afterwards snaps on the side that it is assigned to. Now the screen of Figure 7 is showing and the cube is reaction on the user's input. As the user rotates the real cube to a new position new live streams are shown. By this she is able to see up to three videos at the same time. The user is has the option to preview the adjoining sides by navigating to the next destination. Assuming that the user's goal is to bring her favoured media to the front position (which is the best visible one) the program-logic considers it as the user's selection. There is also no change of that side if the user lies the cube down, consequently the selected media stays in focus. Based on that assumption the application chooses the front media to be enlarged to full screen as soon as the user puts the cube back on a surface. The user can repeat they this procedure as often as she wants.

Fig. 7. A user exploring the cube interface by interacting with a large screen television.

5 User Feedback on the Prototype and Discussion

After finalizing the prototype we asked 6 users to casually explore the system. In this phase we were in particular interested in potential problems that occur when using the system.

One problem that became quickly obvious is the missing capability to sense the third angle due to the sensor system used and due to the fact that the sensing is related to gravity. The experience for the user is significantly different for motions in different direction. In two directions the reaction of the virtual counterpart is very direct and immediate, whereas the coupling in the third dimension is very rough (as described above). This different behaviour conflicts with the user's intuitive understanding of the connection between the real and the virtual cube.

However having one dimension that does not respond immediate also introduced a certain degree of freedom to the usage. In our sessions we recognized that users facilitate this as a feature. By relying on the system to ignore a limited Y roation the user can reset a position – this is similar to lifting a mouse for readjusting it into a more comfortable place without manipulation in the virtual space.

A further issue that we investigated was the fact that we had to decide what information (channels) to put on the limited number of faces on the cube. There was not final conclusion but several options seem appropriate depending on the user and the environment. One approach is to give the user the opportunity to freely assign the sites with her favourite channels. Another strategy would be to randomly assign TV-Channels to the sides and exchanging them on the fly as soon as they are rotated to a non-visible state. This is more related to the idea of a playful interaction.

Similar to the feature image-in-image the cube shows more than one stream at the time. This is similar to a feature more and more TV-devices have namely the possibility to add a little window of another channel in one of the full screen's corners and be able to quickly switch between them. In this respect the cube interface offers more options as there are up to 3 channels on at once and further 3 are quickly available.

The overall user feedback was that the user interface provides a intuitive and more playful user interface to a TV, even with the limitations mentioned above.

6 Conclusion

In this paper we presented the idea, the concept, implementation, and initial user feedback on a new interface for a TV. We explored different types of potential devices that can be used as a tangible UI and decided to use a cube because of the affordances given and the clear mapping to visualization.

The implementation consists of several parts: a tangible UI that is wirelessly connected to the system on which the visualization is realized. To acquire information about the user's manipulation of the interface a unit sensing four axis of acceleration is used. The acceleration values are converted and used to move the visual representation of the screen.

Initial user feedback suggests that having user interfaces that are effective and playful can create a good user experience. In cases such as watching TV providing a

playful experience can be more important than pure efficiency. Currently we are improving the sensor system based on the initial feedback and plan for a larger user study.

Acknowledgement. The research presented in this paper is joint work between the Embedded Interaction Group at University of Munich and the Computing Department at Lancaster University. The work at Munich is funded by the German Research Foundation (DFG). The Lancaster contribution is supported by the EPSRC as part of the Equator IRC.

References

1. T. Lashina, F. Vignoli, V. Buil, S. van de Wijdeven, G. Hollemans, J. Hoonhout. The Context Aware Personal Remote Control: A Case Study on Context Awareness. 23rd International Conference on Distributed Computing Systems Workshops (ICDCSW'03).
2. Jeffrey Nichols, Brad A. Myers, Michael Higgins, Joe Hughes, Thomas K. Harris, Roni Rosenfeld, Mathilde Pignol. "Generating Remote Control Interfaces for Complex Appliances." CHI Letters: ACM Symposium on User Interface Software and Technology, UIST'02, 27-30 Oct. 2002, Paris, France. pp. 161-170.
3. Bill Gaver. Designing for Ludic Aspects of Everyday Life. Special Issue on Ambient Intelligence. ERCIM News No.47, October 2001.
4. Andreas Butz, Markus Groß and Antonio Krüger: "TUISTER: a Tangible UI for Hierarchical Structures", in Proceedings of IUI 2004, January 13-16, 2004, Madeira, Funchal, Portugal.
5. H. Gellersen, G. Kortuem, M. Beigl and A. Schmidt. Physical Prototyping with Smart-Its. IEEE Pervasive Computing Magazin (2004, accepted for publication).
6. L. E. Holmquist, H.-W. Gellersen, A. Schmidt, M. Strohbach, G. Kortuem, S. Antifakos, F. Michahelles, B. Schiele, M. Beigl, R. Mazé. Building Intelligent Environments with Smart-Its. IEEE Computer Graphics & Applications. January/February 2004 (Vol. 24, No. 1), pp. 56-64.
7. MIT Media lab. Responsive Environments Group. 3X Inertial Measurement Unit Project. http://www.media.mit.edu/resenv/imu/ (page visited May 2004).
8. Lancaster University. Smart-Its technical information. Wiki-Web. Page visited May 2005. http://ubicomp.lancs.ac.uk/twiki/bin/viewauth/Smartits/WebHome (anonymous login as user TWikiGuest with password guest).
9. H. Rodjegard, G. Andersson. Design optimization of three-axis accelerometers based on four seismic masses. Proceedings of IEEE Sensors, 2002. Vol 2. Pages: 1099- 1104
10. K. Van Laerhoven, N. Villar, A. Schmidt, G. Kortuem and H.-W. Gellersen. Using an Autonomous Cube for Basic Navigation and Input. ICMI/PUI 2003. ACM Press. Vancouver, Canada. November 2003, pp. 203-211.