# Design and Implementation of a Widget Set for Steerable Projector-Camera Units

Dennis Reiter
Computer Science
Saarland University
Saarbrücken, Germany

Andreas Butz
Media Informatics
University of Munich
Munich, Germany

## Abstract

We describe the design and implementation of graphical interaction widgets for use with a steerable projector-camera unit. The design of our widgets is adapted to provide the right visual cues when projected and they are controlled by the user's hand. The widgets' input regions are arranged in an ergonomic way and they use a simple but robust computer vision technique for interaction. The widget set has been implemented in our instrumented environment, and we demonstrate interactive buttons, checkboxes, sliders and selection boxes.

## 1 Introduction

Instrumented environments are one way of exploring ubiquitous computing scenarios within limited areas of space. Instrumentation can be achieved by various kinds of sensors and actuators, including cameras, projectors and conventional displays. The SUPIE (Saarland University Pervasive Instrumented Environment) is such an instrumented environment in the size of a regular office, and one of its central elements is a steerable projector-camera unit mounted on the ceiling in the middle of the room. Details of the setup can be found in [1]. One straightforward approach to make such an environment interactive is to transfer concepts from current desktop or WIMP (Windows, Icons, Menus, Pointing device) interfaces to it. Using known concepts, such as GUI widgets as interactive elements in the environment also has the advantage, that they will be intuitively understandable to people who had exposure to WIMP GUIs before. In a sense, they even come closer to the original idea of direct manipulation interfaces, since they are manipulated directly with the hand instead of indirectly via a pointing device.

## 2 Related Work

Kjeldsen and Hartmann [3] provide an overview of vision-based User Interfaces, the issues encountered there, and the general design space. They also discuss the use of widgets, but only at a very general level. From the techniques described there, we only use the statical pointing action, since this can be detected reliably even under difficult conditions, such as low resolution and frame rate, and high jitter and noise. Later work by the same authors [4] describes an architecture for reconfigurable interfaces which provide similar functionality in different situations, but the physical design of the widgets is not discussed. In [5], design issues of projective interfaces are also discussed, but only at a conceptual level. In our work, we observe these insights, but specifically contribute aspects of the physical design (i.e. the actual graphics) of robust projective widgets. Fails and Olsen present a way of providing interactivity in the physical environment in their work on light widgets [2]. While direct visual feedback is a core ingredient of Direct Manipulation Interfaces, their light widgets remain invisible to the user's eye and just provide data input. Feedback is only obtained indirectly via the triggered actions, such as volume control or station selection of a radio. In our work we aim to provide widgets which are much closer to the concept of widgets

in graphical UIs, i.e. clearly recognizable as interactive elements and providing immediate visual feedback. Using a steerable projector-camera unit to create the visual elements, our method also works with a single camera instead of two, since the camera and the projector almost share an optical axis, and interactivity is connected to projected elements rather than physical locations.

# 3 A Widget Set for Projection

Projected widgets differ from regular GUI widgets in several aspects. One limiting factor is the relatively low output resolution and the absence of pixel alignment in images rectified for arbitrary surface angles. Another limitation is the relatively low input resolution of a video camera. A more subtle difference is the fact, that we cannot project darkness. This means, that any surface can only become brighter by projection and that any dark area can only be dark in comparison to brighter surroundings. While this sounds trivial, it has substantial impact on the widgets' graphical design.

## 3.1 Creating Interactive Regions

To provide interactive elements with limited camera resolution and low frame rates, we refrained from true motion detection or feature tracking. Instead, we use interactive regions, in which we just detect the presence of large contrasting objects, such as the user's hand. These regions correspond to small rectangular areas in the video stream, for which we observe changes in their average color and brightness. If an interactive region has $w * h$ pixels, for each of the pixels we observe its red, green and blue values $0 \leq r, g, b \leq 255$. When the widget is created, its visual elements are projected and then a snapshot of the video stream in the interactive region is stored. This snapshot contains $r, g, b$ values for each pixel in the region, which now serve as the calibration values $r_0$, $g_0$, and $b_0$. Changes in the surface color, such as patterned or colored areas are thus accounted for. Slow changes in overall brightness, such as changing daylight, are accommodated for by the camera's automatic exposure control.

From then on, in each frame we compare the actual $r, g, b$ values to the calibration values and count the number of pixels $n_r$, for which $abs(r - r_0) > c$ for a constant $0 \leq c \leq 255$, similarly $n_g$ and $n_b$ for the other colors. If the number of pixels which deviate from the calibration value in this way is more than a certain part $p$ of the area in at least one color channel (e.g., $n_r > p * w * h$), this constitutes a significant change in the region. The inevitable pixel noise is filtered out by choosing the constant $c$ big enough. For practical matters, we have determined useful values of $c$ to be around $40$ for robust detection. Small objects, such as a single finger from a hand touching an adjacent widget, are filtered out by choosing the value of $p$ big enough. In our practical tests, values for $p$ around $0.5$ served our purposes well. Observing the three color channels separately accounts for situations, in which the projection surface has a similar overall brightness (grey value) as the user's hand. On a skin-colored background, of course, even this procedure fails. In general, however, the method above allows the creation of interactive regions, for which the touch of the user's hand (or any other large object) can be detected reliably. We do not attempt to distinguish an actual touch from an object hovering over the projection surface, so the widgets can also be operated without any physical contact.

## 3.2 Visual Cues and Feedback

In Desktop GUIs, interactive elements are usually given a 3D appearance as a visual cue for their interactivity and as a means to provide visual feedback. Buttons, for example, stand out by default and seem to sink in when they are pushed, imitating a physical behavior. The 3D appearance is provided by a virtual light source from the upper left corner of the screen (light from upper left corners has a long tradition in the visual arts), which makes the top and left edge of the button appear brighter than its surroundings and the lower and right edge darker. The background color is therefore never set to black, but mostly middle tone values. If the button is pushed, the edge coloring is reversed and since it is perceptually implausible that a light source changes so suddenly, and since this would also be perceptually inconsistent with all the other buttons remaining unchanged, we perceive that the pushed button sinks in (see figure 1 left).

If we try to project the same design without its surrounding grey area, this whole mechanism doesn't quite work anymore. While the regular button can still be seen as an object standing out from a dark environment, its

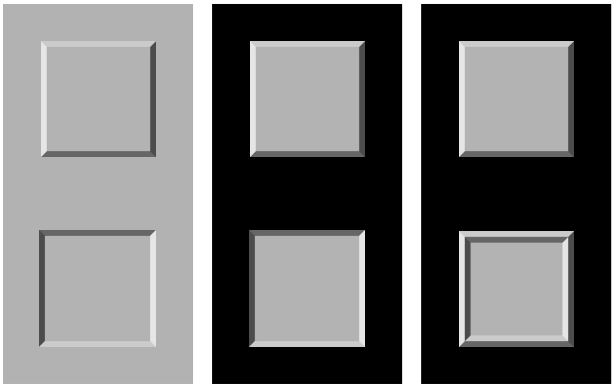

Figure 1: Left: a push button as known from Desktop GUIs; middle: the same design when projected; right: a button design which is better suited for projection

pushed counterpart appears like a well-lit cavity in a dark surrounding, which strongly contradicts our experience with the physical world. Cavities are usually darker than their environment (given a light source from the side), and so the button rather appears as standing out, but lit from the other side (see fig. 1 middle). In order to restore visual plausibility, a plausible visual context has to be provided. Instead of embedding the widget in a lit area, this can also be achieved by a design which doesn't make the whole button sink in, but only its middle part, leaving a ridge around the edge. This arrangement is more consistent with our experience, since the whole button is still perceived as standing out, and can therefore plausibly be well lit in a dark environment (see figure 1 right). Finally, in traditional GUIs, widgets can be "greyed out" which signifies that they are not functional at this point but will become functional in other situations. This effect can easily be achieved by darkening the widget, which makes it appear weaker in the projection.

### 3.3 Arrangement of Widget Elements

For a combo box, a logical arrangement of the interactive elements would be to place the button to show or hide the list of alternatives next to the current selection and the buttons to move the list up or down above respectively below it. If we use such an arrangement projected onto the

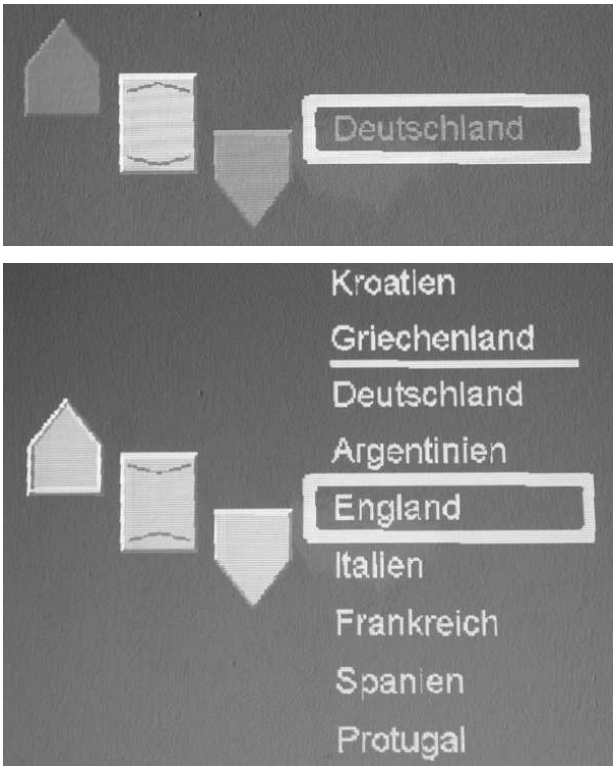


Figure 2: A projected combo box widget in its closed state (top) and while operated (bottom)

wall in front of us at eye height, we cannot reach the middle or top area without covering up the bottom one, since our hand comes into the image from below. This problem is referred to as the Midas touch problem: every interactive region we cover is triggered. A horizontal alignment of the interactive regions would solve this problem, but would contradict the desired spatial mapping of the buttons (up moves up and down moves down). One possible solution is to position interactive regions on a diagonal line, which is perpendicular to the expected direction of the forearm when using the widget. Figure 2 thus shows a projected combo box widget for use by the left hand at eye height. In our widget set, combo boxes can be created in different variations for use by the left or right hand and at different heights in the environment, which results in different arrangements of the interactive elements.

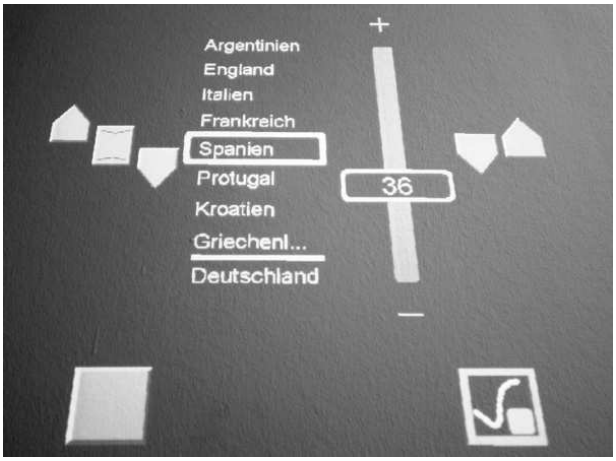


Figure 3: Four widget types: a combo box (top left), a slider (top right), a check box (bottom right) and a simple push button (bottom left)

The other three widget types we implemented are a slider, a checkbox and a simple push button (Figure 3). The slider is operated similarly to the combo box in its open state with the difference, that the axis of possible values doesn't move, but the boxed value on it changes and slides up and down with the push of the up/down buttons. While this design is slower than a slider operated by a moving hand, it can be built with the robust detection algorithm discussed above and will therefore also work under difficult conditions. The checkbox widget consists of a frame which either contains a check mark or is empty (in analogy to screen GUIs) and a small button in the lower right corner which can be touched to toggle the check mark. The design of the simple push button itself (bottom left) is obvious from the discussion above. These four widgets were implemented in Java and they form the basis of our projective widget set. They can be instantiated with different parameters regarding their content, range or spatial arrangement and a position in the environment, just as regular GUI widgets on a 2D screen. The position, of course, is supposed to lie on a suitable projection surface in our 3D model of the environment. Currently, this 3D model and the set of display surfaces are still constructed manually, but we will work towards an automatic acquisition in the future.

# 4 Summary and Future Work

We have presented the design and implementation of widgets for use with a steerable projector-camera unit. These widgets have a graphical design which is adapted to the fact that they are projected without a visual context. They use a simple but robust recognition method which allows their use under difficult conditions, such as low camera resolution and uncontrolled lighting. The spatial arrangement of the widget elements accounts for a general problem with this kind of algorithm (Midas touch problem). Four interactive GUI widgets have been implemented as a widget set which can be used by programmers similarly to regular widgets in screen interfaces, in order to create interactive elements in the physical environment.

Using widgets in an instrumented environment will make interactive elements immediately recognizable to users who have seen GUI widgets before. In WIMP GUIs there is a single mouse pointer and all interaction is done through it. In instrumented environments, this doesn't hold, since users can operate widgets with both hands, and several users can interact in a cooperative or concurring way. While the widgets presented here have a very direct relation to the GUI world, we will explore other widget concepts to account for this situation as well.

# References

[1] A. Butz, M. Schneider, and M. Spassova. Searchlight - a lightweight search function for pervasive environments. In *Proceedings of Pervasive 2004*, LNCS. Springer, 2004.

[2] J. Fails and D. Olsen. LightWidgets: Interacting in everyday spaces. In *Proceedings of IUI '02 (San Francisco CA, January 2002)*, 2002.

[3] R. Kjeldsen and J. Hartman. Design issues for vision-based computer interaction systems. In *Workshop on Perceptive User Interfaces*. ACM Digital Library, November 2001. ISBN 1-58113-448-7.

[4] R. Kjeldsen, A. Levas, and C. Pinhanez. Dynamically reconfigurable vision-based user interfaces. In *Proceedings of 3rd International Conference on Computer Vision Systems*, pages 323–332, 2003.

[5] N. Sukaviriya, M. Podlaseck, R. Kjeldsen, A. Levas, G. Pingali, and C. Pinhanez. Embedding interactions in a retail store environment: The design and lessons learned. In *Proc. of INTERACT'03, Zurich, Switzerland*, 2003.