BendDesk: Dragging Across the Curve

Malte Weiss *Simon Voelker* *Christine Sutter* *Jan Borchers*

RWTH Aachen University
52056 Aachen, Germany
{weiss, voelker, borchers}@cs.rwth-aachen.de
christine.sutter@psych.rwth-aachen.de

ABSTRACT
We present BendDesk, a hybrid interactive desk system that combines a horizontal and a vertical interactive surface via a curve. The system provides seamless touch input across its entire area. We explain scalable algorithms that provide graphical output and multi-touch input on a curved surface. In three tasks we investigate the performance of dragging gestures across the curve, as well as the virtual aiming at targets. Our main findings are: 1) Dragging across a curve is significantly slower than on flat surfaces. 2) The smaller the entrance angle when dragging across the curve, the longer the average trajectory and the higher the variance of trajectories across users. 3) The curved shape of the system impairs virtual aiming at targets.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Input Devices and Strategies.

General terms: Design, Human Factors

Keywords: Curved surface, desk environment, multi-touch, dragging, virtual aiming.

INTRODUCTION

A typical computer workplace integrates horizontal and vertical surfaces into a workspace. It encompasses at least one or more vertical displays that show digital content and a larger horizontal area, containing input devices, such as mouse and keyboard, paper-based documents, and everyday objects. Touch recognition technologies have combined the benefits of traditional input metaphors with digital documents [24]. Tablets allow high precision stylus input for graphic design; digital pens, such as Anoto[1], enable annotations on physical paper; and multi-touch gestures [4] provide an intuitive way to transform and modify digital data. However, despite all the advantages these interfaces have barely found their way into everyday workspaces yet.

[1]www.anoto.com

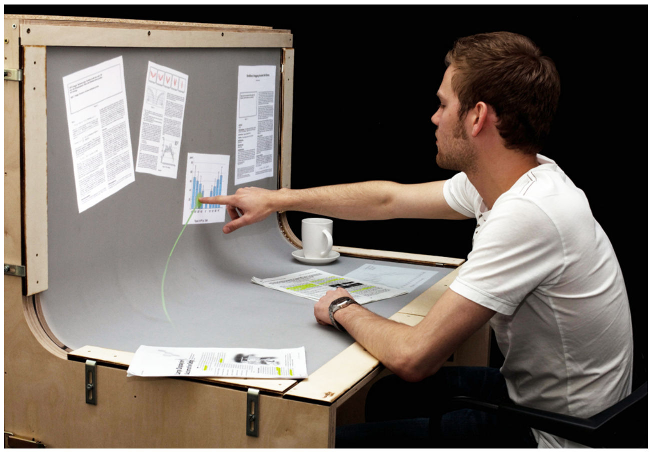
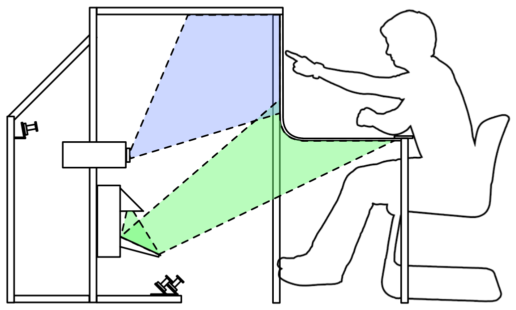


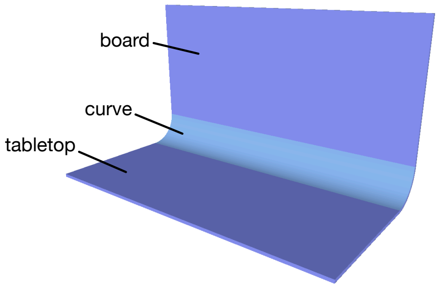
Figure 1: BendDesk seamlessly merges a horizontal and a vertical interactive surface with a curve.

Many systems have been proposed that use vertical and horizontal interactive surfaces within a single desk environment (e.g., [7, 16]). They provide a large interactive area and allow to move digital objects across multiple displays. However, those systems suffer from a lack of spatial continuity. According to the Gestalt Law of Closure [5], gaps between adjacent displays suggest isolated interactive areas. Other laws may be violated that are useful in screen design, e.g., the Law of Proximity, because objects belonging together may be separated across the gap. Furthermore, splitting objects across bezels impairs search accuracy and tunnel steering performance [2]. Finally, those setups limit the applicability of direct manipulation, as movement trajectories are interrupted when dragging a finger or pen from screen to screen.

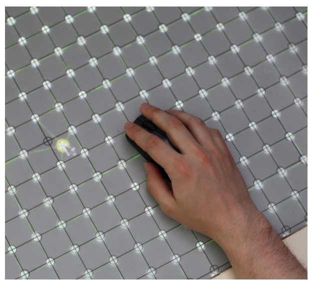
In this paper, we present BendDesk, a desk environment that merges a vertical and a horizontal multi-touch surface into one interactive surface using a curve (Figure 1). Our system provides a large interactive area within the user's reach and allows uninterrupted, seamless dragging gestures across the entire surface. The focus of this paper is to explore the effects of a curve between two orthogonal surfaces on one of the most basic gestures: dragging. Our results can inform the design of more complex gestures, as most of these can be subdivided into elementary dragging and pointing operations.



(a) Placing of projectors and cameras.



(b) Interactive areas of the BendDesk system.



(c) Manual screen calibration.

Figure 2: Hardware setup and screen calibration.

RELATED WORK

Our project was inspired by the Sun "Starfire" video prototype from 1994 that intended to predict a potential future workplace in 2004 [23]. The envisioned system featured a large, interactive area, different input modalities such as gestures and direct manipulation, and applications, such as remote collaboration.

In recent years, the specific characteristics of horizontal and vertical interactive surfaces have received great interest in the research community. According to Morris et al. [18], horizontal surfaces are more appropriate for annotation and pen-based note-taking, while vertical displays support reading and intensive writing tasks using keyboards. Since no display seems appropriate for all potential tasks, Morris et al. propose a hybrid system. In a later paper [17], they report on a field study involving multiple horizontal and vertical screens. Although participants were enthusiastic about the extra space, one problem reported was that the horizontal and vertical screens were perceived as isolated areas. Some studies [17, 19] indicate that interactive surfaces should allow tilting to increase comfort, such as the FLUX table [15]. However, Morris et al. also emphasize that desk environments should fit into the ecologies of objects. For example, a table should allow users to put down everyday objects. This coincides with observations in a long-term study by Wigdor et al. [26]. The authors point out the "dual use" of interactive tabletops as computing devices and as pieces of furniture. In their study, the participant tended to tilt the table at an angle that avoided objects to fall from the table.

The combination of horizontal and vertical interactive surfaces has mostly been applied to two applications: collaborative workspaces and remote desks. While tabletops are suitable for face-to-face group work and provide awareness of each other's actions, interactive boards can provide an overview of information shared among groups. Accordingly, many systems have been developed that integrate vertical and horizontal interactive surfaces into collaborative workspaces in order to add digital capabilities [8, 13, 21, 25]. The incorporation of both surface types has also been applied to remote desk environments. For example, the Agora system [16] and DigiTable [7] provide an interactive horizontal surface for a private document space and a vertical surface displaying a remote person via a video conferencing system. However, the vertical surface is non-interactive in most of these systems.

Nearly all multi-touch systems are limited to one or more flat interactive devices. One exception is Sphere [1], a spherical multi-touch enabled display. Furthermore, the field of organic interfaces [6, 11] proposes interactive non-planar surfaces that can be freely deformed. Early examples of this vision are Paper Windows [12] and Gummi [22]. Recently, Curve [28] presented ergonomics and design considerations for building a curved multi-touch table.

DESIGN CONSIDERATIONS

We envisioned BendDesk as a multi-touch desk environment that supports interaction with digital documents but also respects the nature of traditional desks. Although there is evidence that tilted surfaces yield high acceptance for specific tasks (see above), we intentionally avoided them for two reasons: Firstly, we consider the support of the ecology of (everyday) objects as crucial. With the exception of special purpose desks, such as drawing tables, office desks are usually horizontal because people put physical objects on them. In contrast, the possibilities of placing objects onto a tilted surface, even at small angles, are limited. Secondly, tilting the vertical surface backwards would reduce its reachability at the top.

We also accounted for ergonomic requirements: the user should be able to sit in a comfortable position and to reach the entire input area without much effort. We applied ISO norm 9241-5 to choose the height of the table. Furthermore, we conducted preliminary user tests on an adjustable table prototype to find the depth for the vertical surface. In these tests, users perform pointing and dragging tasks where the depth of the vertical surface was varied.

HARDWARE SETUP

As illustrated in Figure 2, our interactive desk consists of one 104 cm $\times$ 104 cm acrylic surface that is bent to yield two orthogonal surfaces, seamlessly merged by a curve. The surface is mounted at 72 cm height[2], on a half-closed wooden box that contains all electronics, such as projectors for graphical output and cameras for touch input. The form factor of our setup separates the device into three interactive areas: the vertical *board* (100 cm $\times$ 43 cm), the *curve* (100 cm $\times$ 16 cm) with a radius of 10 cm, and the horizontal *tabletop* (100 cm $\times$ 40 cm). We choose a radius of 10 cm to provide a large

[2]following ISO 9241-5

planar interactive surface while allowing a comfortable dragging through the curve. Furthermore, we added a raised non-interactive strip in front of the board that fixes the acrylic. As a side effect, this provides an area for the user to rest her hands.

Two short-throw projectors behind the surface show the graphical user interface (GUI) on a Dura-Lar diffusor, each operating with a resolution of 1024 $\times$ 768 pixels. An Optoma EX525ST projector displays the GUI on the board, while a NEC WT615 projector shows the interface on the curve and the tabletop. Since the latter employs aspheric mirrors to project the graphics in a flat frustrum, the user can sit close to the table without occluding the projection.

We use Frustrated Total Internal Reflection (FTIR) [10] to detect touches on the surface. The acrylic is surrounded by a closed strip of 312 LEDs with a spacing of 1.2 cm that feed infrared (IR) light into the surface. Furthermore, we apply a thin silicone compliant layer between the acrylic and the diffusor. Three Point Grey FireFly MV cameras with attached IR filters track touches on the surface, each running at 60 fps and a resolution of 640 $\times$ 480 pixels.

VISUAL OUTPUT

Our framework provides a square 1024 $\times$ 1024 pixel GUI that maps isomorphically to the interactive area. The output resolution is approximately 26 dots per inch (DPI). However, the DPI can be increased by using projectors with a higher resolution or more projectors at shorter distances. The bottom left pixel $(0, 0)$ corresponds to the front left corner of the tabletop and the pixel $(1023, 1023)$ maps the top right corner of the board. We define the *upwards* direction on the table as a vector with a positive y-coordinate in GUI coordinates. The *downwards* direction is defined analogously.

Since our system involves a non-planar surface, we must compensate for substantial distortions when projecting the user interface. Hence, we render the entire GUI into an off-screen buffer first. Subsequently, each projector displays a part of this buffer on a bicubic spline patch that compensates the respective distortion. Special care is required to position each projector such that its projected *target area* is placed completely in its depth of field.

Projector calibration

We employ a manual calibration process to compute the spline patches for each projector. A paper calibration sheet with an imprinted uniform grid of 32×32 dots is placed onto the interactive area. Accordingly, each printed dot with index $(x, y) \in \{0, 1, ..., 31\}^2$ on the sheet maps to a pixel position $P(x, y)$ in the GUI space:

$$P : \{0, 1, ..., 31\}^2 \to [0, 1024)^2$$

The result of a successful calibration process is a *projected* dot pattern that exactly matches the nodes on the paper grid. That is,

$$D_i(x, y) = P(x, y) \quad \forall (x, y)\text{: frustrum}_i(x, y) = 1$$

where $D_i(x, y)$ is the mapping of projected grid dots to GUI coordinates for each projector $i \in \{1, 2\}$, defined analogously to $P(x, y)$. The function $\text{frustrum}_i(x, y)$ indicates whether the paper grid point (x, y) is inside the frustrum of projector i or not:

$$\text{frustrum}_i(x, y) = \begin{cases} 1 & P(x, y) \text{ in frustrum of projector } i \\ 0 & \text{otherwise} \end{cases}$$

Each projector is calibrated separately. When starting the calibration for projector i, it displays a 32×32 uniform grid that covers the entire screen space of the projector. Hence, each projected grid point is shown at a certain *screen coordinate* $S_i(x, y)$ with

$$S_i : \{0, 1, ..., 31\}^2 \to [0, 1024) \times [0, 768).$$

Thereafter, the user deselects all grid rows and columns that do no map to rows in the calibration sheet (defines $\text{frustrum}_i(x, y)$). In our case, this means that she deselects the bottom 18 rows for the top projector and the top 16 rows for the bottom projector. Then the user moves the projected grid dots until they fit with the corresponding points in the paper sheet ($D_i(x, y) = P(x, y)$). We implemented a set of transform tools to speed up this manual process.

Finally, when the user confirms the calibration, a sub-grid is extracted that contains all grid dots inside the frustrum of the projector ($\text{frustrum}_i(x, y) = 1$). The corresponding screen coordinates $S_i(x, y)$ then represent the interpolation points of the bicubic spline patch, whereas the values $D_i(x, y)$ are used as texture coordinates to render this part of the interface on the table. This technique easily scales up to setups with more than two projectors, while the process has to be performed only once for each configuration. Figure 2(c) illustrates the manual screen calibration.

Rendering pipeline

When launching a BendDesk application, our software framework first creates a 1024×1024 GUI texture. It then reads the spline patch for each projector and extracts a high resolution quad patch with texture coordinates that map into the GUI texture space. As the geometry is static, it can be rendered efficiently, e.g., by using vertex buffers. In each frame, the GUI is rendered into a texture first and then distributed to the projectors, which output the texture on the respective spline patches using the coordinates from the calibration process. We hide this pipeline in the background, i.e., the application designer addresses the GUI coordinate space, without having to pay attention to calibration issues or projector setups.

TRACKING

Our camera setup detects touches on the entire interactive surface, with each camera covering a specific area and sensing FTIR spots independently. We employed a simple detection algorithm based on a connected component analysis after background subtraction. After detecting spots for all cameras, their coordinates are transformed from camera to GUI coordinates. Similar to the screen calibration, we use a bicubic spline patch for this mapping, as described below. Finally, the transformed spots are sent to the application as touch events in GUI coordinates. Note that all camera fields of vision overlap to ensure continuous tracking between the areas. If multiple spots are mapped to nearly the same GUI

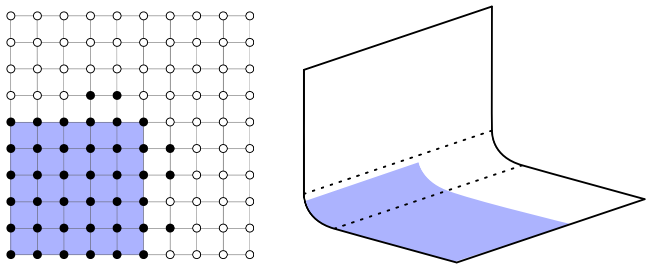


Figure 3: Extraction of spline patch to map from GUI to camera space. Left: Largest rectangle containing visible dots is extracted. Right: Corresponding area on table.

position, they are merged into a single touch event by averaging their coordinates.

A predictive tracking algorithm ensures the registration of touch events between successive frames, even if the user quickly changes speed or direction of a finger on the surface. That is, for each touch T at position p, we track its velocity and acceleration and extrapolate p to its anticipated position p' in the subsequent frame. If there is a touch close to p' in the next frame, we assume that it is a translated version of T. In practice, the use of predictive tracking strongly improves the touch registration on our system and reliably avoids that users "lose" dragged or transformed objects.

Camera calibration

For each camera $j \in \{1, 2, 3\}$, our calibration process creates a mapping from camera coordinates to global GUI coordinates. When starting the calibration, our software displays an $N \times M$ uniform grid with GUI coordinates $G(x, y)$ that covers the interactive surface. Note that this requires a correct projector calibration.

In the first step, the calibration creates a mapping C_j from GUI grid point indices to camera pixels:

$$C_j : \{0, 1, ..., N\} \times \{0, 1, ..., M\} \rightarrow [0, 640) \times [0, 480)$$

where N and M denote the grid resolution. The calibration intends to find the camera pixels that match the GUI grid points. Accordingly, we need to find values $C_j(x, y)$ for all (x, y) with $\text{visible}_j(x, y) = 1$, where

$$\text{visible}_j(x, y) = \begin{cases} 1 & G(x, y) \text{ visible from camera } j \\ 0 & \text{otherwise.} \end{cases}$$

All cameras are calibrated at the same time. The system successively highlights each grid point. For each highlighted dot $(\bar{x}, \bar{y})$, the user touches the surface at that position and then confirms with a button click on a wireless control. Now, our algorithm stores which cameras detected the resulting FTIR spot, i.e., $\text{visible}_j(\bar{x}, \bar{y})$, and at which position, $C_j(\bar{x}, \bar{y})$.

As illustrated in Figure 3, this manual process yields a visibility map, visible_j, for each camera. We extract the largest rectangle that only contains visible spots by solving the *Maximum Empty Rectangle problem* [20]. Similar to the screen calibration, the extracted point indices together with $G(x, y)$ and $C_j(x, y)$ represent the interpolation points for a bicubic spline patch $\mathcal{P}$ that maps from GUI to camera coordinates for camera j.

However, we need the inverse mapping to detect which positions on the GUI are touched. Our algorithm computes the map C_j^*

$$C_j^* : \{0, 1, ..., 639\} \times \{0, 1, ..., 479\} \rightarrow [0, 1024)^2$$

by uniformly evaluating the patch $\mathcal{P}$ with a high sampling rate. For each sample, the source GUI position is stored at the target camera position in C_j^*. This yields a discrete inverse map for camera j. Afterwards, if a spot is visible in camera j, we can read its GUI position from C_j^*. In order to avoid jitter, we employ bilinear interpolation for this lookup.

Although the calibration involves manual user interaction, it does not require more than five minutes in practice. Furthermore, it only has to be repeated when the camera setup is changed. In our case, a 20×20 grid was sufficient to calibrate all cameras.

EVALUATION

Dragging is a simple and one of the most frequently used gestures on interactive tabletops. However, dragging performance on curved surfaces is still a mostly unexplored topic. In this section, we present several user tests that investigate the dragging performance on the different areas of the table. We furthermore test the virtual aiming across curved surfaces, which is, e.g., important for flinging gestures.

Participants A total of 18 participants (16 males), aged between 24 and 32 years (mean age 27 years) took part in the study. They did not receive any compensation, but we raffled a \$25 gift coupon among them. 15 participants were computer scientists, two were school teachers, and one was a mechanical engineer.

General procedure The study was carried out in a dimly lit room, where participants sat in front of the BendDesk. Participants worked throughout three different interaction tasks: two dragging tasks and a virtual aiming task. Each task type was introduced by a test trial to familiarize participants with the new task. The task instructions were standardized and it was emphasized to solve the tasks as fast and accurate as possible. In total the experiment lasted about 40 to 60 minutes.

Dragging across the curve

We first investigated dragging performance across the different interactive areas of BendDesk and compared dragging performance across the curve to dragging on the planar board and tabletop area.

Task design and procedure The experimental task and the conditions are depicted in Figure 4. The system displayed the source, a white colored square with a side length of 50 px (4.88 cm), and the target, a white frame of the same size. Both were vertically arranged with a distance of 150 px (14.64 cm). The participant had to drag the source quad onto the target using her index finger. After successfully matching source and target (we allowed a tolerance of 10 px, or 0.98

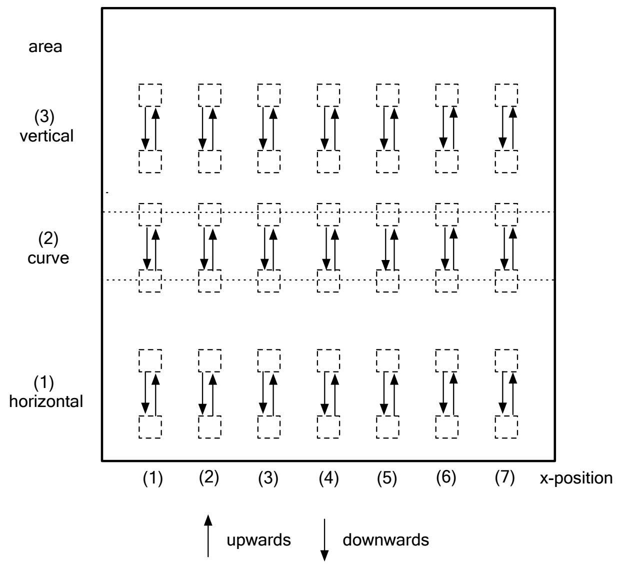


Figure 4: Experimental design of vertical dragging task.

cm), the interactive area went blank and the next trial was displayed. They appeared in three different areas (in the horizontal plane, the curve, or the vertical plane), and dragging direction from source to target was either upwards or downwards. This resulted in 3 (area) $\times$ 2 (dragging direction) experimental conditions. We further controlled the distribution of trials across the surface by presenting trials on seven different x-positions with two repetitions each. The order of trials was randomized. Participants worked throughout 84 trials with their dominant hand and throughout another 84 trials with their non-dominant hand. This yielded a total number of 168 dragging operations per participant. Dragging duration was defined as the interval from touching the source until correctly releasing it when the source was placed in the target (given in ms). Dragging trajectory covered the observed length of the finger's movement path, again from touching the source until correctly releasing it when the source was placed in the target (given in px).

We hypothesized the following outcomes:

- H1 *(horizontal vs. vertical)*: Dragging (a) duration and (b) trajectory are shorter on the horizontal surface than on the vertical one.
- H2 *(planar vs. curve)*: Dragging (a) duration and (b) trajectory are shorter on planar surfaces than on the curved area.
- H3 *(down vs. up)*: Dragging (a) duration and (b) trajectory are shorter when moving upwards in GUI coordinates than when moving downwards.

Results The data were analyzed for each of the dependent variables with 3 $\times$ 2 analyses of variances (ANOVAs) with the within-subject factors *area* and *direction*. Dragging durations are depicted in Figure 5. The ANOVA revealed a significant main effect of the factor *area* $(F(2, 34) = 14.20; p < 0.01)$. Dragging durations inside the curve (mean 1166 ms) were 14% (150 ms) longer than the dragging durations on

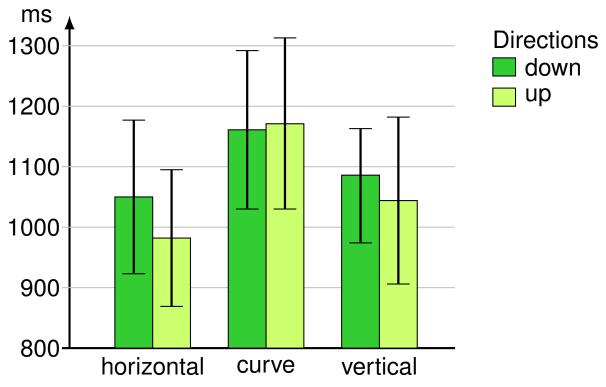


Figure 5: Dragging duration depending on area and direction. Whiskers denote 95% confidence interval.

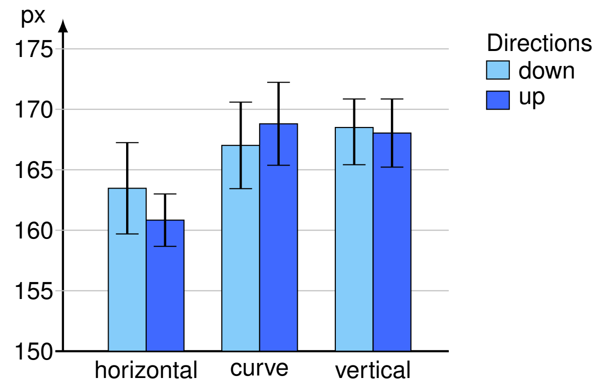


Figure 6: Length of dragging trajectory depending on area and direction.

the horizontal area (mean 1016 ms) and 10% (110 ms) longer than the dragging durations on the vertical area (mean 1056 ms). Other main effects and the interaction were not significant.

Figure 6 illustrates the length of dragging trajectories. The ANOVA showed a significant main effect of the factor *area* $(F(2, 34) = 28.84; p < 0.01)$. The dragging trajectories inside the curve (mean 167 px) were 3% (5 px) longer than the dragging trajectories on the horizontal area (mean 162 px). But dragging through the curve was equally long compared to vertical dragging (mean 168 px). Furthermore, for the horizontal plane, but not for the other areas, upward dragging was significantly shorter than downward dragging. This yielded a significant interaction $(F(2, 34) = 4.73; p < 0.05)$. The main effect of the factor *direction* alone was not significant.

To sum up, when comparing horizontal and vertical dragging (H1) the results clearly showed shorter trajectories for operations in the horizontal plane. This is in accordance with H1. However, dragging duration was comparable for both planes. On a first glance this is not further surprising, since finger amplitude and target size remained constant over the task. So, in accordance with Fitts' Law [9], movement durations should be constant as well. However, on a second glance the results also show that the movement plane seemed to have no

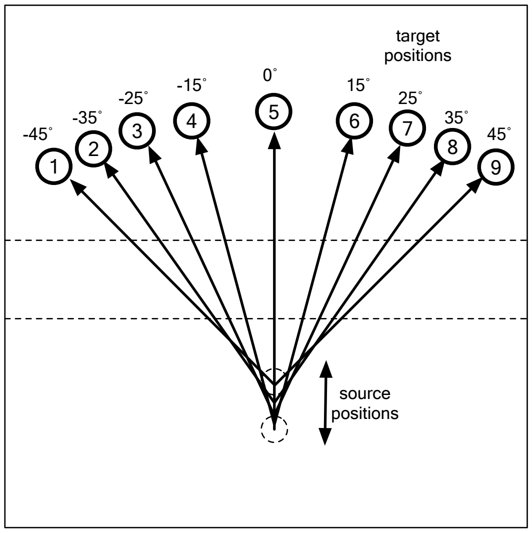


Figure 7: Experimental design of cross-dragging performance task for upward condition.

further effect on movement durations. The main finding is, that movement execution was optimized along the observed movement path. This optimization did not lead to any further improvement of movement duration, probably caused by a bottom effect—durations were very short and seemed to be already at a minimum for the given distance. Second, our results support H2: dragging on a planar surface is indeed more efficient (in terms of durations) than dragging across the curve. Concerning Fitts' Law [9] this is a rather unexpected finding, as with a constant index of difficulty one would have expected constant movement durations over all areas. As this is clearly not the case findings suggest that motor control across the curve is more complex and therefore takes longer. Considering the movement path, only horizontal but not vertical dragging was superior to dragging in the curved area. This might indicate that motor control is more difficult for dragging in a curved or vertical area than in the horizontal area. Finally, we hypothesized more efficient upward than downward movements (H3). Although performance data in both planar surfaces slightly hint at an advantage for upward movements, this was only significant for horizontal finger trajectories. Thus, overall the data did not confirm H3.

Cross-dragging performance depending on angle

With the second task we explored dragging performance not within an area (as in Task 1) but across the whole BendDesk surface and we compared if dragging performance depended on the angle of approach.

Task design and procedure The experimental task is depicted in Figure 7. Our system displayed the source, a white colored circle and the target, a black colored circle inside a white ring. Both circles had a diameter of 60 px (5.82 cm) and the thickness of the target ring amounts to 20 px (1.94 cm). The distance between source and target was 600 px (58.20 cm). As in Task 1, participants had to drag the source onto the target using the index finger. After successfully matching source and target (within a tolerance of 10 px, or 0.98 cm), the interactive area went blank and the next trial appeared.

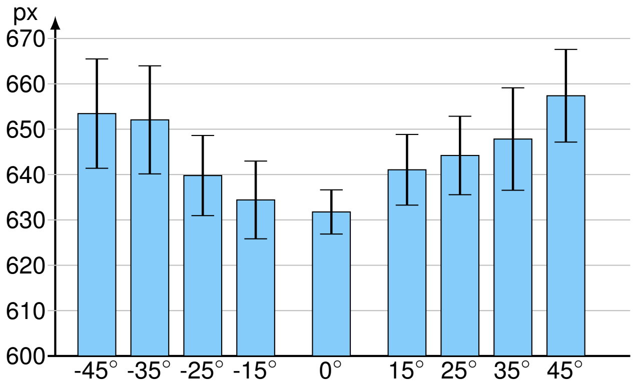


Figure 8: Length of dragging trajectory depending on angle.

Trials appeared in 9 different movement directions (with two repetitions each): (1) $45°$, (2) $35°$, (3) $25°$, (4) $15°$ to the left, (5) $0°$ (vertical line), and (6) $15°$, (7) $25°$, (8) $35°$, (9) $45°$ to the right. The movement started either in the horizontal area (upward) or the vertical area (downward). The order of trials was randomized. Participants worked throughout 36 trials with their dominant hand and throughout another 36 trials with their non-dominant hand. A total of 72 dragging operations were presented. Dependent variables were the same as described in Task 1.

We assumed that a larger angle yields a lower dragging performance and higher deviation from the ideal dragging line:

- H4: The dragging (a) duration and (b) trajectory increases with larger dragging angles.
- H5: The deviation of trajectories increases with larger dragging angles, thus showing more variance in movement paths.

Results Data were analyzed for each of the dependent variables with one-factorial analyses of variances (ANOVAs) with the within-subject factors *angle*. For dragging durations the ANOVA revealed a significant main effect for the factor *angle* $(F(8, 128) = 2.65; p < 0.05)$. Dragging durations varied between 1306 and 3806 ms. However, the differences across angles were too small to show statistical significance in post-hoc comparisons.

The mean length of dragging trajectories are depicted in Figure 8. We found a significant main effect of the factor *angle* $(F(8, 128) = 8.94; p < 0.01)$. Post-hoc comparison showed that dragging trajectories for targets $45°$ to the left or to the right of the source (mean 652 px) were significantly longer when compared to targets vertically presented to the source (mean 631 px).

Furthermore, deviation of movement trajectories from the direct connection between source and target were analyzed (Figure 9 and Figure 10). The ANOVA showed significant main effects of the factor *angle* for the maximum $(F(8, 128) = 11.66; p < 0.01)$ as well as the average $(F(8, 128) = 10.51;$

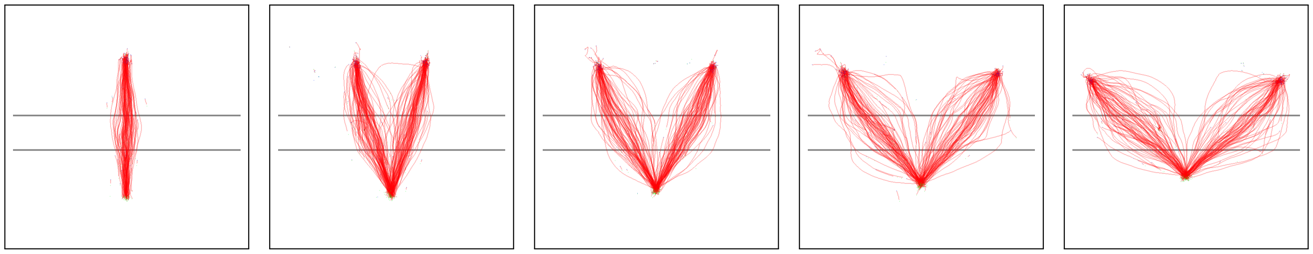


Figure 9: Dragging trajectories for upward dragging across the curve for different angles. Variance significantly increases with higher angles.

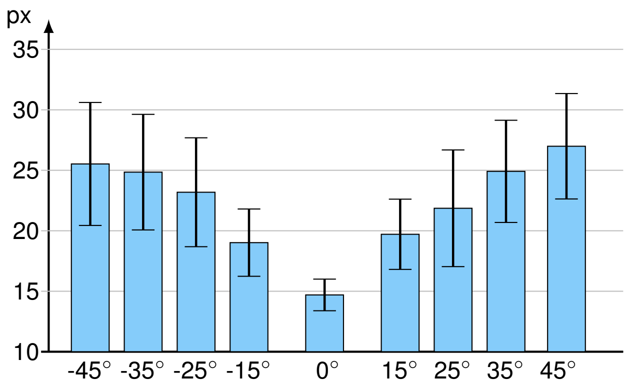


Figure 10: Average deviation from direct line between source and target depending on angle.

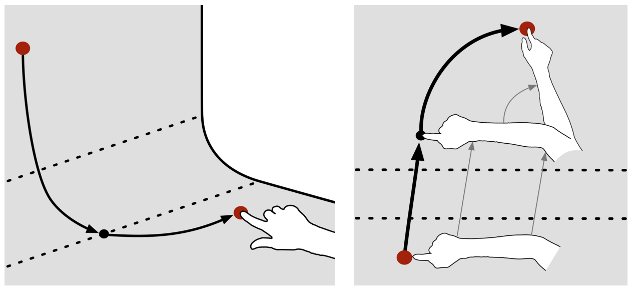


Figure 11: Observed dragging trajectories that reduce exertion.

$p < 0.01$) deviation between presented and observed amplitude (Figure 10). The deviation increased by 85% (12 px) for larger angles.

We summarize, in accordance with H4 and H5 trajectories and the variance in movement paths increased for extreme angles.

Virtual aiming at the target

With the last task, we investigated virtual aiming performance across the BendDesk surface. Other than in the previous tasks participants did not move their finger towards the target, but had to adjust both fingers inside the source area along a virtual aiming path to hit the target. We compared whether virtual aiming was supported with or without a grid displayed on the surface.

Task design and procedure The experimental task is depicted in Figure 12. The system displayed the source, a gray colored circle with a diameter of 200 px (19.5 cm) and the target, a white colored circle with a diameter of 30 px (2.9 cm). The distance between source and target was 800 px (78.1 cm). Participants had to position the left and right index finger inside the source area until an imagined line drawn through both finger tips would hit the target area. The system gave visual feedback by rendering circles beneath the touches. When participants felt that they would have hit the target they released both fingers and the system displayed a gray line through both touches towards the target area. Then, the interactive area went blank and the next trial appeared.

Trials appeared in ten different movement directions (with two repetitions each). Targets within the horizontal plane: (1) 90°, (2) 80°, (3) 70°; target within the curve: (4) 60°; and targets across the curve and in the vertical plane: (5) 50°, (6) 40°, (7) 30°, (8) 20°, (9) 10°, and (10) 0°. The order of trials was randomized. Participants worked throughout a block with a uniform grid on the system's surface (we displayed a 26 x 26 grid with a cell size of about 40 px $\times$ 40 px, or 3.9 cm $\times$ 3.9 cm) and throughout another block without a grid but a solid blue-colored surface. This resulted in 10 (angle) $\times$ 2 (background) experimental conditions. We further controlled the virtual aiming direction by presenting the source either in the right or left corner of the horizontal area (upward aiming), or in the right or left corner of the vertical area (downward aiming). This resulted in a total number of 160 virtual aiming operations. As dependent variable we measured the aiming error (Figure 13), i.e., the deviation between the virtual aiming path and the target area, or in other words the spatial misjudgment (given in px).

We hypothesized the following outcomes:

- H6: The aiming error is smaller for virtual aiming within one plane than across the curve and different planes.
- H7: The aiming error is smaller for virtual aiming with a grid displayed on the surface than without a grid.

Results The data were analyzed with a 10 $\times$ 2 analysis of variance (ANOVA) with the within-subject factor *angle* and *background*. Aiming errors are depicted in Figure 14. The ANOVA revealed a significant main effect of the factor *angle* ($F(9, 158) = 17.24; p < 0.01$). Aiming errors were smallest when source and target were within the same plane, i.e., virtual aiming at 90° (mean error 9 px) was significantly more accurate than at all other angles (mean error 43 px).

Furthermore, aiming at 90° and 0° was supported by the grid displayed on the surface: Aiming errors for the 90° angle were 65% (8 px) smaller with displayed grid (mean 5 px) than without the grid (mean 13 px). For the 0° angle the aiming errors were 57% (19 px) smaller with the grid (mean 14 px) than without the grid (mean 33 px). However, the background did not have any effect on the other angles, yielding a significant interaction $(F(9, 153) = 2.61; p < 0.01)$. The factor *background* alone did not show any significant effect on aiming errors.

Finally, the results from the virtual aiming task showed that virtual aiming is most accurate for the orthogonal angles (0°, 90°) when a grid is displayed on the surface. This is only partially in line with our hypotheses H6 and H7.

Exhaustion considerations

As pointed out earlier, users had to use their entire arm to drag objects across the curve and onto the vertical surface. We believed that these movements would lead to muscle fatigue in short time. Therefore, we conducted an informal test to gain a rough estimate when dragging movements become inconvenient. We repeated the cross-dragging task and let users drag objects across the curve in both directions without dropping the arm onto the surface. We asked the participants to stop the test as soon as they felt muscle fatigue. Furthermore, we asked users to express any signs of fatigue during the test.

Results In the first four minutes, no participant reported any signs of fatigue. After four minutes six participants expressed signs of fatigue in their upper arm. On average, each participant conducted this task for about 7:30 minutes. However, after 12 minutes two participants commented that they could do the test “the whole day”. Both stopped the test after about 15 minutes without any symptoms of fatigue.

Most participants (16/18) perceived the downward dragging as more comfortable than the reverse direction because of the inward rotation of the hand during the movement. For the downwards direction, they could almost let their arm fall down. This confirms our observations in the second user test.

Ten participants thought that their dragging speed on the curved area was much slower than on the other areas. Additionally, five of them thought that they had to use more pressure on the curve to drag the object. 13 participants stated that diagonal dragging through the curve was inconvenient.

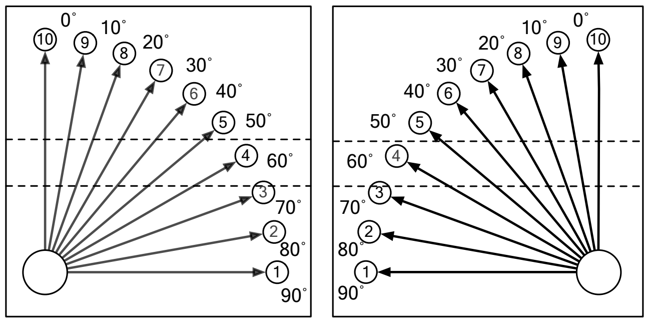


Figure 12: Experimental design for virtual aiming task.

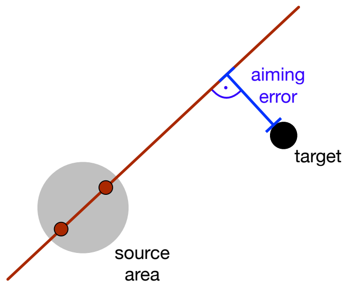


Figure 13: Two finger touches in the source area define a straight line towards the target. The aiming error is the deviation between this virtual aiming path and the target area.

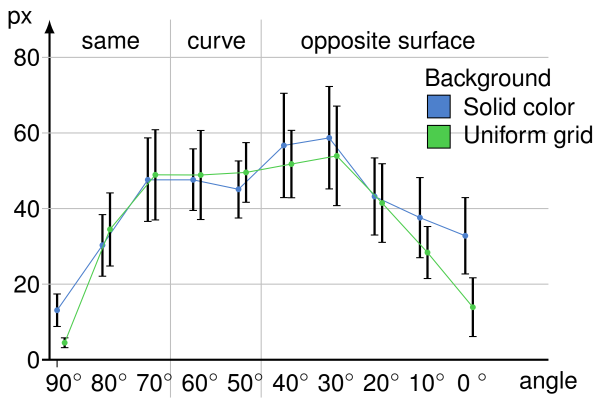


Figure 14: Distance from target depending on angle.

Our user population is not representative for an ergonomic analysis of the BendDesk system as most users were male and between 24 and 32 years old. Nevertheless, in contrast to our assumptions, all users were able to perform the dragging task for a rather long period without any fatigue. In future work, we will explore the ergonomic aspects of interaction gestures in more detail.

DISCUSSION

The evaluation of BendDesk revealed three main findings: First, dragging on a planar surface is faster and straighter than dragging across the curve, although the distances were constant for all dragging tasks. This is a rather unexpected finding, as Fitts’ Law [9] would have predicted constant movement durations over all areas. The increased movement durations across the curve went along with a higher curvature in hand paths. From a cognitive point of view, the curved hand path is similar to motor behavior observed when avoiding obstacles. Jax and Rosenbaum [14] found in their study that the anticipation of obstacles led to more curved hand paths, even when the obstacle was not present. This suggests that our participants perceived the curve as a kind of obstacle, which they tended to avoid. Considering the motor behavior, we assume that the more curved hand paths in the curve also results from the more complex motor activity involved in curve dragging: The participants performed horizontal dragging basically by pushing or pulling the hand backwards or forwards. Analogously, they dragged the target on the vertical surface by lifting or lowering the arm. In contrast, when moving across the curve, users tended to turn in the entire hand while moving it upwards and downwards, which yields a more complex movement. One person stated afterwards that the tendon in his index finger hurt if he did not turn the hand, while another person reported that he was afraid of drilling his index finger into the surface and, thus, turned the hand. Furthermore, four users wanted to change from the index to the middle finger when they unintentionally released an object during the dragging because they considered the middle finger as stronger and more stable.

Second, the angle had nearly no effect on the duration of dragging operations. However, we noticed a significant increase in trajectory length when the angle is increased. In order to gain more insights into the causes for this effect, we plotted out the trajectories for each angle (Figure 9). Two effects become apparent: First, at higher angles participants tended to minimize the dragging distance on the curve. Some users even separated the movement into a short path across the curve and a long path for the remaining movement. Second, the higher the angle the higher the spreading of trajectories beside the direct line. This also matches Figure 10 that indicates an increased variance for higher angles. Furthermore, our observations revealed that most users optimized their dragging operations to reduce muscle exertion. We noticed that some users dragged downwards by letting the arm quickly fall straight downwards and across the curve before dragging the object to the target, as shown in Figure 11(a). Another frequent movement was an upward dragging, where the user firstly dragged the object across the curve using a stiff bent arm and finished the dragging by turning hand and lower arm with the upper arm as rotation axis, as shown in Figure 11(b). Moreover, two users reported that approaching the curve in a flat angle feels uncomfortable. In general, we noticed that users tried to create a convenient movement trajectory despite the task to acquire the target as fast as possible.

Finally, the virtual aiming task revealed the complexity of imagined instead of manual aiming gestures. Participants severely misjudged the spatial relations towards the target. However, at orthogonal angles (0°, 90°) the aiming error was lowest, probably because the table borders provided alignment guides. This is further supported by the observed improvement of virtual aiming at 0° and 90° when a grid was present. In this case participants could easily touch the grid lines to hit the target. We assume that the complexity of virtual aiming depends on the required cognitive mapping between the 3D and the GUI space. If the user aims at a target on the same surface, she has to compensate for the perspective distortion of the plane, where, according to [27], those effects are stronger on horizontal surfaces. If the target is placed on the opposite area, the user has to perform a three-dimensional non-linear transformation of the table shape to the rectangular control space.

CONCLUSION

We presented an interactive desk system that merges a vertical and a horizontal display with a curve into a spatially cohesive surface. This provides a large interactive area that users can reach in a comfortable sitting position. We introduced a technique to project on the curved surface, as well as algorithms for multi-touch detection under strong distortions. The system enables seamless dragging gestures across all areas. Nevertheless, our user studies suggest that the curve represents a slight but noticeable physical barrier. It leads to longer interaction times when crossing it and some users tend to minimize the dragging distances in that area when approaching it with a flat angle. Furthermore, it impairs the user's spatial perception.

For application designers, this means that the three areas should not be considered as a single interactive surface. Users will more likely reduce the number of interactions across the curve, and the user interface should not require cross-dragging with flat angles. Instead, the characteristics of the curve must be taken into account and can even be exploited to divide the surface into logical units. For example, an application could use the horizontal display to create content that is stored in the curve before it is assembled at the vertical area. That is, the three areas would represent steps in a workflow. Another scenario is remote collaboration, where the vertical space represents a public space showing content visible to all co-workers, while the horizontal area is a private space for individual content. In these cases, the curve could act as an intermediate storage, or as a "dock" or "taskbar".

FUTURE WORK

In future work, we will investigate the factors influencing the dragging performance in further detail. We plan to conduct tests with different curve radii and varying angles between the large areas. Furthermore, we want to explore the application domain of BendDesk. We intend to identify the desk tasks that are suitable for the transfer to our system and determine the role each area plays in particular applications. Moreover, we plan to involve additional input modalities. For example, our diffusor layer can be replaced with an Anoto pattern that allows both touch and precise pen input [3, 15]. Additionally, we plan to find the limitations of such a system opposed to a common desk setup. Finally, we want to fathom to which extent the vision of a multi-touch based desk environment in the shape of BendDesk is practicable.

ACKNOWLEDGMENTS

This work was funded in part by the German B-IT Foundation. We thank Christian Remy for his helpful comments on this paper.

REFERENCES

1. Benko, H., Wilson, A.D., and Balakrishnan, R. Sphere: Multi-touch interactions on a spherical display. In *Proc. of UIST '08*, ACM, 2008, pp. 77–86.

2. Bi, X., Bae, S.-H., and Balakrishnan, R. Effects of interior bezels of tiled-monitor large displays on visual search, tunnel steering, and target selection. In *Proc. of CHI '10*, ACM, 2010, pp. 65–74.

3. Brandl, P., Forlines, C., Wigdor, D., Haller, M., and Shen, C. Combining and measuring the benefits of bimanual pen and direct-touch interaction on horizontal interfaces. In *Proc. of AVI '08*, ACM, 2008, pp. 154–161.

4. Cao, X., Wilson, A., and Balakrishnan, R. ShapeTouch: leveraging contact shape on interactive surfaces. In *Proc. of TABLETOP '08*, 2008, pp. 129–136.

5. Chang, D., Dooley, L., and Tuovinen, J.E. Gestalt theory in visual screen design: a new look at an old subject. In *Proc. of WCCE '01*, Australian Computer Society, Inc., 2002, pp. 5–12.

6. Co, E. and Pashenkov, N. Emerging display technologies for organic user interfaces. *Communications of the ACM*, 2008, 51(6):45–47.

7. Coldefy, F. and Louis-dit-Picard, S. Digitable: An interactive multiuser table for collocated and remote collaboration enabling remote gesture visualization. In *Proc. of TABLETOP '07*, 2007, pp. 1–8.

8. Everitt, K., Shen, C., Ryall, K., and Forlines, C. MultiSpace: enabling electronic document micro-mobility in table-centric, multi-device environments. In *Proc. of TABLETOP '06*, 2006, pp. 27–34.

9. Fitts, P.M. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 1954, 47:381–391.

10. Han, J. Low-cost multi-touch sensing through frustrated total internal reflection. In *Proc. of UIST '05*, 2005, pp. 115–118.

11. Holman, D. and Vertegaal, R. Organic user interfaces: Designing computers in any way, shape, or form. *Communications of the ACM*, 2008, 51(6):48–55.

12. Holman, D., Vertegaal, R., Altosaar, M., Troje, N., and Johns, D. Paper windows: interaction techniques for digital paper. In *Proc. of CHI '05*, ACM, 2005, pp. 591–599.

13. Izadi, S., Brignull, H., Rodden, T., Rogers, Y., and Underwood, M. Dynamo: a public interactive surface supporting the cooperative sharing and exchange of media. In *Proc. of UIST '03*, ACM, 2003, pp. 159–168.

14. Jax, S.A. and Rosenbaum, D.A. Hand path priming in manual obstacle avoidance: Evidence that the dorsal stream does not only control visually guided actions in real time. *Journal of Experimental Psychology: Human Perception and Performance*, 2007, 33:425–441.

15. Leitner, J., Powell, J., Brandl, P., Seifried, T., Haller, M., Dorray, B., and To, P. FLUX: a tilting multi-touch and pen based surface. In *Proc. of CHI '09*, ACM, 2009, pp. 3211–3216.

16. Luff, P., Heath, C., Kuzuoka, H., Yamazaki, K., and Yamashita, J. Handling documents and discriminating objects in hybrid spaces. In *Proc. of CHI '06*, ACM, 2006, pp. 561–570.

17. Morris, M., Brush, A.J.B., and Meyers, B. A field study of knowledge workers' use of interactive horizontal displays. In *Proc. of TABLETOP '08*, 2008, pp. 105–112.

18. Morris, M.R., Brush, A.J.B., and Meyers, B.R. Reading revisited: Evaluating the usability of digital display surfaces for active reading tasks. In *Proc. of TABLETOP '07*, 2007, pp. 79–86.

19. Müller-Tomfelde, C., Wessels, A., and Schremmer, C. Tilted tabletops: In between horizontal and vertical workspaces. In *Proc. of TABLETOP '08*, 2008, pp. 49–56.

20. Naamad, A. and Lee, W.L. On the maximum empty rectangle problem. *Discrete Applied Mathematics*, 1984, 8(3):267–277.

21. Rekimoto, J. and Saitoh, M. Augmented surfaces: A spatially continuous work space for hybrid computing environments. In *Proc. of CHI '99*. ACM, 1999, pp. 378–385.

22. Schwesig, C., Poupyrev, I., and Mori, E. Gummi: a bendable computer. In *Proc. of CHI '04*, ACM, 2004, pp. 263–270.

23. Tognazzini, B. The "Starfire" video prototype project: A case history. In *Proc. of CHI '94*, ACM, 1994, pp. 99–105.

24. Wellner, P. The DigitalDesk Calculator: Tangible Manipulation on a Desk Top Display. In *Proc. of UIST '91*, ACM, 1991, pp. 27–33.

25. Wigdor, D., Jiang, H., Forlines, C., Borkin, M., and Shen, C. WeSpace: the design development and deployment of a walk-up and share multi-surface visual collaboration system. In *Proc. of CHI '09*, ACM, 2009, pp. 1237–1246.

26. Wigdor, D., Perm, G., Ryall, K., Esenther, A., and Shen, C. Living with a tabletop: Analysis and observations of long term office use of a multi-touch table. In *Proc. of TABLETOP '07*, 2007, pp. 60–67.

27. Wigdor, D., Shen, C., Forlines, C., and Balakrishnan, R. Perception of elementary graphical elements in tabletop and multi-surface environments. In *Proc. of CHI '07*, ACM, 2007, pp. 473–482.

28. Wimmer, R., Hennecke, F., Schulz, F., Boring, S., Butz, A., and Hußmann, H. Curve: Revisiting the Digital Desk. In *Proc. of NordiCHI '10*, ACM, 2010, pp. 561–570.