

Lumino: Tangible Blocks for Tabletop Computers Based on Glass Fiber Bundles

Patrick Baudisch, Torsten Becker, and Frederik Rudeck

Hasso Plattner Institute
Potsdam, Germany

{Patrick.Baudisch, Torsten.Becker, and Frederik.Rudeck}@hpi.uni-potsdam.de

ABSTRACT

Tabletop computers based on diffuse illumination can track fiducial markers placed on the table’s surface. In this paper, we demonstrate how to do the same with objects arranged in a three-dimensional structure without modifying the table. We present *lumino*, a system of building blocks. In addition to a marker, each block contains a glass fiber bundle. The bundle optically guides the light reflected off markers in the higher levels down to the table surface, where the table’s built-in camera reads it. While guiding marker images down, the bundle optically scales and rearranges them. It thereby fits the images of an entire vertical arrangement of markers into the horizontal space usually occupied by a single 2D marker. We present six classes of blocks and matching marker designs, each of which is optimized for different requirements. We show three demo applications. One of them is a construction kit that logs and critiques constructions. The presented blocks are unpowered and maintenance-free, keeping larger numbers of blocks manageable.

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

Keywords: Tabletop, tangible, building blocks, glass fiber bundles, construction kit, stacking, markers.

General terms: Design, Human factors.

INTRODUCTION

Physical building blocks allow quick construction and manipulation of structures through two-handed tactile interaction [13]. Building blocks have been realized using a variety of tracking mechanisms including magnetic tracking (e.g., *bricks* [11]) and electric connectors between blocks (e.g., *self-describing building blocks* [2] and *ActiveCube* [22]).

Recently, tabletop computers based on diffuse illumination have emerged as a new platform for tangible computing. These tables can track 2D visual barcodes aka *fiducial markers* (Figure 3a) placed on their surface. Arbitrary objects and mechanical widgets (e.g., Slap widgets [34]) can be tracked by attaching markers to their bottoms.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CHI 2010, April 10–15, 2010, Atlanta, Georgia, USA.

Copyright 2010 ACM 978-1-60558-929-9/10/04...\$10.00.

One of the benefits of the table+marker approach is that it is particularly easy to maintain. Unlike other vision-based tracking methods [24], cameras are contained in the table, making the system self-contained, which allows it maintain its calibration. Unlike the aforementioned electric or magnetic mechanisms, fiducial markers are unpowered and thus require neither a tether nor maintaining batteries. Low maintenance matters, because it makes larger numbers of objects manageable—a prerequisite for complex tangible applications.

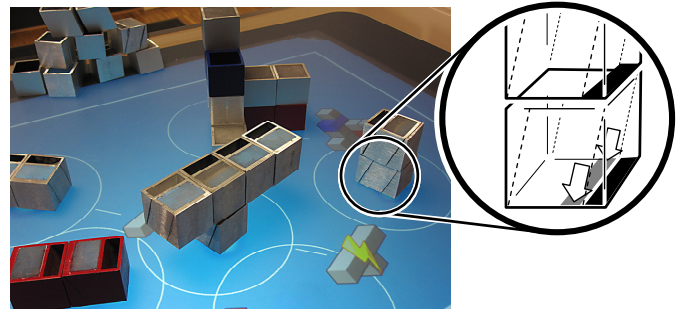


Figure 1: (a) *Luminos* are tangible building blocks that allow the underlying diffuse illumination table to track their 3D arrangement, e.g., to support users with suggestions (such as the “overhang” warning at the bottom right). The callout shows how it works: Each unit contains a glass fiber bundle that transports light reflected off higher-level markers down to where the table can see it. The fiber bundle also rearranges marker images into a horizontal layout, here by offsetting marker images.

Unfortunately, tabletop-based tracking does not translate to objects arranged in three-dimensional structures. The camera in the table cannot sense such structures, because objects closer to the surface occlude objects further away and because the table’s built-in diffuser prevents the camera from recognizing details unless the respective object in direct contact with the table surface.

In this paper, we demonstrate how to overcome this limitation and track objects arranged in 3D structures on an unmodified diffuse illumination table.

LUMINOS

Figure 1 shows one of our demo applications, the *lumino construction kit*. The user has arranged building blocks in three-dimensional structures on a diffuse illumination table, here Microsoft Surface. The key components are the blocks—they are designed to allow the table to recognize their three-dimensional arrangement. This allows the appli-

cation to log construction activities and to assist the user; here it displays a warning about the structural soundness of the simulated construction.

The callout in Figure 1 illustrates the structure of the blocks that enables this. Similar to traditional 2D marker objects, each of the blocks contains a fiducial marker, here shown abstractly as a thin black strip on the right. For blocks that touch the surface, the table sees this marker and extracts the block type from it, such as “red 2x1 building block”. In addition, however, each block also contains a glass fiber bundle that allows the table to see other markers further above the table surface.

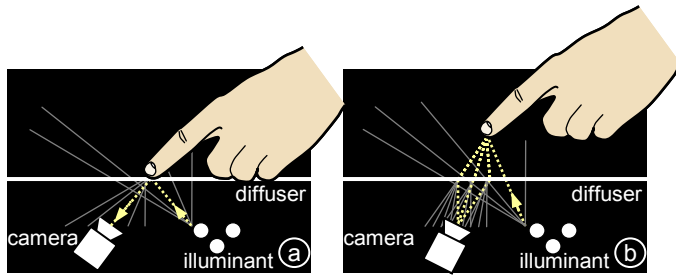


Figure 2: (a) Touch on a diffuse illumination table. (b) Light reflected off a distant hand is scattered before it hits the diffuser, thus it appears blurry to the camera.

To explain why this is necessary, we take another look at why diffuse illumination generally *cannot* sense objects above the surface. As illustrated by Figure 2, infrared illumination reflected off objects and hands returns to the diffuser where it is observed by a camera (Figure 2a). The further away an object is from the surface, the more the light reflected off the object spreads before it hits the diffuser, making its image on the diffuser appear blurry (Figure 2b). This is an important feature, because it allows the table to distinguish touch from non-touch by thresholding blurriness. Unfortunately, the same effect makes markers above the surface unreadable as shown in Figure 3 (see also [5]).

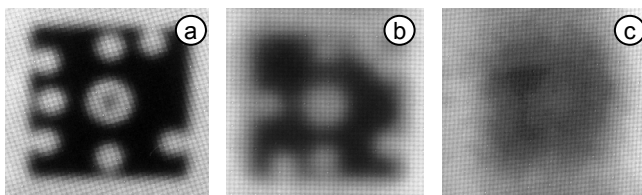


Figure 3: Three markers as seen by the internal camera of a Microsoft surface are (a) recognizable when touching the surface, (b) blurry at 5mm, and (c) unrecognizable when located 10mm above the surface.

The glass fiber bundles in the proposed building blocks circumvent this problem by preventing the light reflected off a marker from spreading. Glass fiber bundles consist of a large number of very thin glass fibers arranged in parallel (Figure 4a). As illustrated by Figure 4b, light reflected off a marker is confined within the fibers and led down to the diffuser, preventing the marker image from blurring. We call this effect *deferred diffusion*.

In our design, this allows the table to see objects on top of the fiber bundle, such as the marker of another block. The sequence of glass fiber bundles in a stack of blocks allows the table to see across the entire vertical structure.

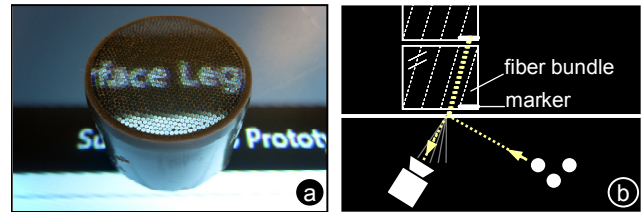


Figure 4: (a) Close-up of a hand-made glass fiber bundle, as contained in luminos. (b) Glass fiber bundles prevent the light reflected off markers from spreading on its way to the diffuser.

The other key characteristic of the glass fiber bundle is that it rearranges marker images as it hands them down to the table surface. In the particular design from Figure 1, glass fiber bundles are *slanted*, which introduces a horizontal offset. This makes each marker image come to rest *next* to the marker below it, rather than on top of it, which prevents markers from occluding each other. Applied recursively through the entire structure, the bundle transforms the vertical arrangement of markers into a horizontal layout on the table surface, where it is read by the table’s built-in camera system.

Resulting design challenge

The block-plus-marker design shown in Figure 1 is one of several designs we have created, each of which is optimized for a different purpose.

The main challenge is that the concept requires encoding the information from an entire vertical arrangement into the limited space normally available to a single 2D marker. At first glance, one might think that larger blocks or increased camera resolution can solve this. While these can help, we show that the space/resolution requirements of a naïve implementation of multi-layer markers grow faster than a constant increase in block size or camera resolution could compensate for. Instead, appropriately chosen mechanical constraints are key to making blocks work with current or future table hardware. The *offset block* design in Figure 1, for example, can support more layers if it mechanically prevents blocks from rotating and translating with respect to each other.

In the remainder of this paper, we present six types of blocks and four classes of markers to go with them. Each of them has its own strengths and limitations and we illustrate them with respective demo applications.

Benefits & contribution

The proposed glass fiber block-plus-marker approach extends the concept of fiducial markers to 3D, while preserving many of the original benefits. The proposed 3D blocks are unpowered, self-contained, require no calibration, and are thus easy to maintain. Luminos thereby help turning diffuse illumination tables into a platform for tangible interaction with 3D objects.

The primary contribution of this paper is (1) the concept of using glass fiber bundles to extend sensing beyond the diffuser and (2) the use of fiber optics to resolve occlusion between tangible blocks, as well as (3) the framework of blocks, markers, and mechanical constraints.

Our main contribution therefore lies in the *engineering* domain. We do not present a user study of block use, as the *user aspects* of tangible building blocks are well understood [15,36] (see also the discussion in [14]).

RELATED WORK

The work presented in this paper is related to tracking solutions for tangible computing, tabletop, and technologies based on glass fibers.

Tracking solutions for tangible computing

Many tangible physical building block systems sense connections between blocks in order to deduce the topology [13]. Electric connectors between blocks are particularly common, e.g., in *self-describing building blocks* [2], *triangles* [13], *tangible programming bricks* [23], and *ActiveCube* [22].

Other systems use a global reference frame, such as a magnetic reference point (e.g., used by *bricks* [11]), a dot pattern on the table (e.g., solutions based on *Anoto* [3] such as [16]) and ultrasonic localization [33]. Since these approaches are based on powered (“active”) components, they require batteries or a tether connecting them to power. This can make it difficult to manage large numbers of tangible objects.

Unpowered (“passive”) components, in contrast, use a centralized platform allowing individual components to be maintenance-free. Examples are magnetic tags [27] and resonance-based tracking [31], although they can track only a few objects and cannot determine object rotation. Acoustic tracking can localize objects, but cannot sense the object ID [8]. Adding RFID to a tracking system allows determining the ID [25].

Computer vision-based systems provide object location and orientation and work for many objects. Recognizing arbitrary objects reliably is a hard problem [4]. Reliability can be achieved with fiducial markers, such as those used by *AR-Toolkit* [9]. Retro-reflective markers are particularly accurate, but also visible and the illumination/camera system requires a comparably bulky setup [24]. Optical systems are generally subject to occlusion; the tangible search system by Fujii et al. alleviates the problem by extending the field of view of their camera using a mirror [12].

Tabletop

Back-projected tabletop computers use markers at the bottom of tangibles objects, e.g., *GeoTUI* [7], *metaDESK* [18], and *reacTable* [20]. Several projects have extended this concept to tangible *widgets*, such as *photo helix* [17], and *slap widgets* [34].

A selection of systems has tried to break away from the flatness of tabletop tangibles. *Second Light* can sense objects above the surface by using an electronically switchable diffuser [19]. *Tablescape Plus* complements

each tangible object on a table with a vertical projection surface [21].

On the input side, Wilson used a depth camera to transfer 3D objects placed on a tabletop system into the virtual world [35]. Bartindale and Harrison demonstrate how to overlay objects on a tabletop computer and how to determine ordering [5]; their objects must be flat and perfectly registered with each other though.

Internal reflection, glass fibers, and bundles

Luminos allow a table to sense at a distance; they thereby implement a *fiber optic sensor* [10]. Fiber optic sensors have been used in a variety of applications, such as *Fiberboard*, where it allows a side-looking camera to observe the backside of an LCD [6]. A similar concept is also explored by *Tactex*’ force sensing surfaces [32].

Fused glass fiber bundles are commonly used to adapt the focal range in night vision equipment (so-called *faceplates*) and as vision aids [29].

TYPES OF GLASS FIBER BUNDLES

The glass fiber bundle in a lumino defines how higher-level markers are “projected” onto the table surface. In Figure 1 we saw a preview of *offset blocks*. Overall, we have explored three types of square fiber bundles (Figure 5) and three types of round bundles (14), which we discuss in the following.

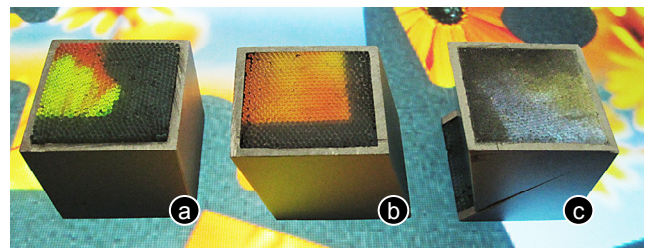


Figure 5: (a) Straight, (b) offset, and (c) demagnification block.

1. Straight blocks

The simplest possible blocks contain a straight vertical fiber bundle, as illustrated by Figure 6a. They “project” marker images straight down. As a result, these blocks can be used to add volume to overlaid flat fiducial markers, such as those demonstrated in [5]. In order to support multiple layers, we partition the block’s surface into k zones, such as the strips shown in Figure 6b. Straight fiber blocks need to line up; otherwise markers shift and may get occluded by other markers below.

The main limitation of straight fiber blocks is that we cannot stack two markers that use the same marker area, because the higher marker would be occluded. This means we cannot have more than one block of each type in a stack. Unless an application is willing to govern which blocks can be combined, this limits the overall number of blocks in the application to k —a very small marker set that will enable only comparably simple applications.

We address this shortcoming with *offset blocks*.

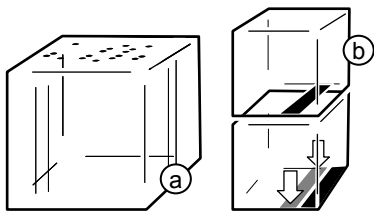


Figure 6: (a) Straight blocks consist of a single, vertically aligned fiber bundle. (b) Each block is required to use a different area for its marker (here shown as black strips).

2. Offset blocks

Offset blocks, first shown in Figure 1, shift the marker area of higher-level blocks on their way down (Figure 7a).

Unlike straight blocks, offset blocks *can* all use the same area for markers; any two blocks now work together. We can also create a large number of blocks with a limited number of marker types, because all blocks of the same *type* can now use the same marker. This is a substantial improvement over straight blocks because it allows for sets with an arbitrary number of blocks, as long as they fall into a manageable number of classes. In the construction kit from Figure 1, for example, all 1x1 blocks bear the same marker code, which makes ID marker codes short and helps fitting more marker codes into a given amount of space—an essential benefit given the severe space constraints faced by luminos.

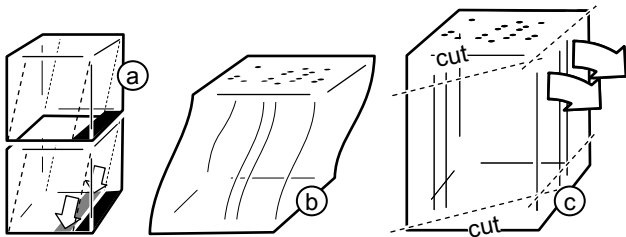


Figure 7: (a) Offset blocks shift the marker images of higher-level blocks. (b) Implementation based on a skewed bundle (c) This version based on a glass fiber bundle cut at an angle is easier to make, but transmits less light.

Figure 7b illustrates the best implementation of an offset block—a sheared fiber bundle. Top and bottom of the bundle are orthogonal to the plane, maximizing light acceptance and transmission to the next block [1]. Such a fiber bundle could be manufactured by heating up the center area of a straight fiber bundle and shearing it; a similar process is commonly used to produce tapered fiber bundles. Since such a process is difficult to accomplish in the lab, we instead used the implementation shown in Figure 7c, i.e., we *trimmed* a straight glass fiber bundle. We obtain a block of width w and height h that supports k layers by cutting the straight bundle at an angle of $\text{atan}(w/hk)$.

A limitation of both straight and offset blocks is that they need to be aligned carefully to assure that markers line up properly, rather than occluding each other. We address this with *demagnification blocks*.

3. Demagnification blocks

Demagnification blocks map the entire area on top of a block to a smaller area at the bottom of the block, as illustrated by Figure 8a.

Figure 8b shows a good implementation of a demagnification block: a block the fibers of which get thinner towards the bottom end. A similar design is available as a commercial product [30]—it is round though and magnifies in x and y , rather than just in one dimension. To achieve the one-dimensional demagnification, we have created the design shown in Figure 8c. Its stacks two shims, each of which is made from a *straight* glass fiber bundle, but with slightly different orientations. The top fiber bundle hands the maker image straight down; the bottom fiber bundle produces the one-dimensional demagnification.

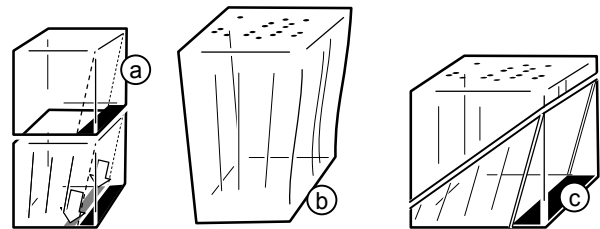


Figure 8: (a) Demagnification blocks map their entire top area to part of the bottom area. (b) Logically, they are based on a glass fiber bundle stretched in one dimension. (c) We obtain the same effect by stacking two appropriately cut straight fiber bundles, at the expense of light loss.

The table reads an arrangement of demagnification blocks by first parsing the markers of the blocks located directly on the table surface. It then *stretches* the remaining part of the camera image of each block across its marker area. This reconstructs the image of the next higher layer, which allows the table to repeat until all layers have been parsed.

The main advantage of demagnification blocks compared to our first two designs is that it allows the table’s camera to see the *entire* area on top of the block; magnifying blocks thereby effectively *sandbox* the higher layers. If combined with a marker code that tolerates clipping, demagnification blocks allow objects on higher layers to be moved and rotated, enabling a broad range of applications, such as tangible multi-layer widgets.

Limitations

The advantages of demagnification blocks come at a price though: marker images are de-magnified by every block they pass on their way down to the surface. As a result, the marker size seen by the camera decreases *exponentially*. This limits the maximum height of structures.

Figure 9 gives the exact numbers: a marker optimized for two-layer structure with magnifying blocks, for example, conveys only as much visual information as a marker designed for a four-layer offset block structure.

Here is the math. To maximize the usable marker area for the top layer we optimize the marker area m of the marker at the top-level k . Its image is scaled down by $k-1$ demagnification steps that together account for a demagnification of $(1-m)^{k-1}$. The projected marker image size is therefore only

$x^{k-1}(1-x)$. It has a global maximum at $x = 1/k$, so interestingly we get the same optimum marker sizes as for straight and offset blocks: for k layers, the marker takes up $1/k$ of the surface area. Unlike straight and offset blocks, however, the camera sees this image through up to $k-1$ demagnification blocks, resulting in a *visible* marker area v of $m(1-m)^{(k-1)}$. With $m = 1/k$ we obtain $v = 1/k(1-1/k)^{k-1}$ —the function plotted in Figure 9.

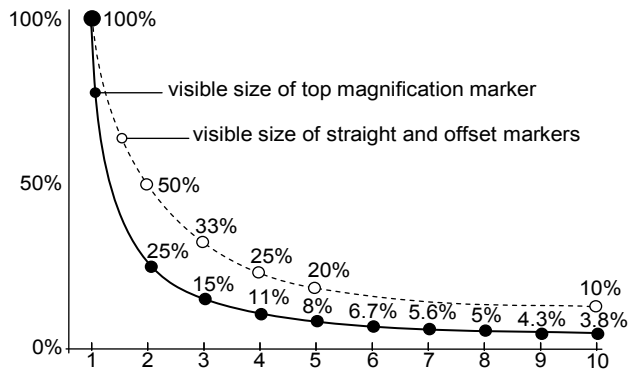


Figure 9: Visible size of top of k markers for demagnification blocks vs. straight and offset blocks.

On square blocks, any manipulation will lead to a certain amount of clipping, including block rotation. Since rotation plays a special role for many tangible widgets, we have created specialized versions of our three blocks for rotation.

Round versions of blocks to allow for rotation

In some cases it is more effective to handle rotation with round blocks organized in rings and sectors. Marker elements are here addressed as *radius* and *azimuth* (r, α), instead of the Cartesian (x, y).

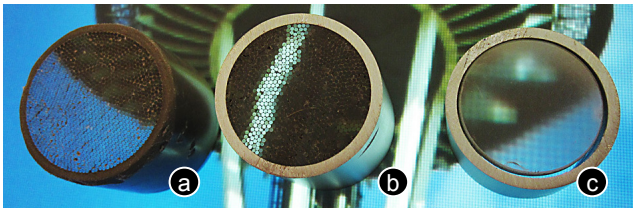


Figure 10: Round blocks: (a) the *straight roll*. (b) This *twister* makes the vertical line in the background appears rotated. (c) This commercially fabricated *taper* magnifies the image by 2.5:1, revealing individual pixels.

We can obtain such blocks by transforming our earlier “Cartesian” designs with polar coordinate transform. We obtain the three blocks shown in Figure 11.

The *straight roll* (Figure 11a) remains straight, but we organize markers in rings and/or sectors now. Ring-shaped markers offer a less favorable aspect ratio than rectangular markers. However, ring-shaped markers stay separated during block rotation. The block maintains the limitations of its Cartesian counterpart, though, namely that it allows for only k different non-overlapping marker designs.

The *twister* (Figure 11b) is the rotational counterpart to the offset fiber. Stacking identical blocks with sector-shaped markers lays the markers out in a circle. Twisters can be

fabricated by twisting a straight roll before fusing it (Figure 10b).

The *taper* (Figure 11c) corresponds to the demagnification block. Like its Cartesian counterpart it virtualizes the next layer. Demagnification now maps outer rings to inner rings, again causing an exponential loss of visible size, but maintaining aspect marker ratio. Tapers were initially developed as vision aids [29] and are therefore available commercially [30] (Figure 10c).

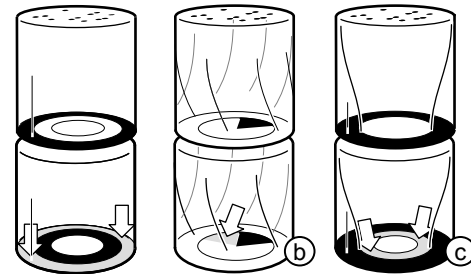


Figure 11: (a) *straight roll*, (b) *twister*, & (c) *taper*

PHYSICS AND LIMITATION OF GLASS FIBER BUNDLES

The strength of our approach stems from the physical characteristics of glass fiber. At the same time, the use of glass fiber bundles introduces the following three limitations.

Light loss is the *main* limiting factor of luminosity. While light transmission inside a fiber is largely lossless, light is lost whenever light *enters* a lumino.

The majority of light is lost with a *single* lumino already. The reason is that glass fibers accept light only if it is oriented roughly along the direction of the fiber (*acceptance cone* [1]). When light passes through a diffuser before entering a lumino light is lost, because the diffuse light is oriented in a broad range of angles. On its round trip (Figure 4b), infrared light passes through two diffuser-to-lumino transitions: (1) when it enters luminosity from below through the table’s diffuser and then (2) when it is reflected off the marker, entering luminosity from above. In our experience, these two transitions eliminate an estimated 80% to 90% of all infrared light. The remainder, however, is sufficient for recognition by the built-in cameras in a Microsoft Surface.

How *high* luminosity can be stacked depends on how far light loss *between* luminosity can be eliminated. We have achieved five layers with commercially produced 40mm glass fiber bundles from [10]. Since the surface of these bundles is *highly* polished, light leaves bundles only at angles within the acceptance cone, making it perfect for adoption by the next lumino. Commercially made bundles can also offer good conductivity of infrared light and space-filling hexagonal fiber shapes [29].

Hand-made plastic fiber bundles, in contrast, are useful primarily in prototyping. While they can hardly reach the required level of polish, they are easy to handle, which makes it easy to explore block types and shapes (see Section “Making Custom Glass Fiber Bundles”). The same holds for blocks the fibers of which meet the diffuser at an angle (Figure 7c and Figure 8c).

Resampling: Each glass fiber in a bundle transports one “pixel” of light (Figure 4b). On its path from the marker to the camera, the marker image therefore gets “resampled” every time it enters a fiber bundle. Repeated resampling can reduce image quality. In order to minimize quality loss, we use glass fiber bundles that exceed the resolution of the camera. Commercially available glass fiber bundles use 25-100 μ m fibers (e.g., Figure 10c), which supersedes the 1.1mm camera resolution in Microsoft Surface by a factor of 10-40. Hand-made plastic fiber bundles with 0.75mm to 1mm fibers work fine for prototyping (all fiber bundles in Figure 5 and Figure 10a and b).

Clipping: Recognizing a marker requires an optical path between marker and table. If blocks are not aligned, a part of a marker hanging over the edge will get *clipped*, making it invisible to the camera. An application can cope with partial clipping by encoding markers with redundant, self-correcting codes. We can enable larger overhang by inserting an *extension block*, which is a specialized type of demagnification blocks (Figure 12a). The same concept allows creating overpasses (Figure 12b).

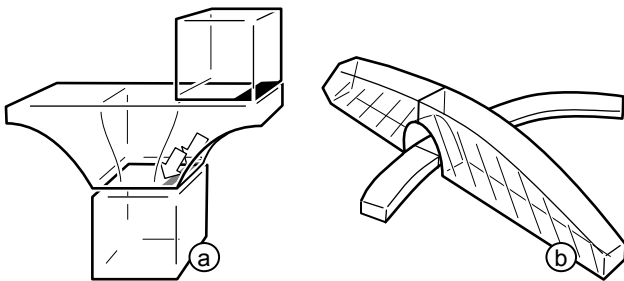


Figure 12: (a) Extension bricks allow sensing above a void and (b) building overpasses that simultaneously sense above and below.

Glass fiber blocks can be combined with a variety of markers. We first present three concrete demo applications; then we use these as the basis for discussing markers.

APPLICATION EXAMPLES

We have created three demo applications: *lumino checkers*, *multi dials*, and the *lumino construction kit*.

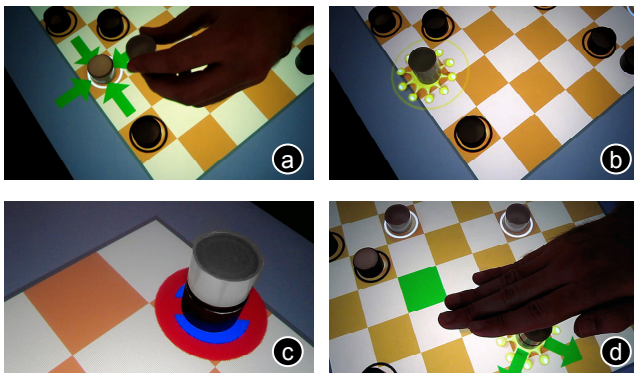


Figure 13: (a) The table requests this piece be made a king (b) Adding another piece causes the table to display the king symbol (c) the table senses incorrect stacking (d) as well as hover.

Checkers

The checkers demo allows users to move and take pieces. As illustrated by Figure 13 (a) When a piece reaches the opposite baseline the table requests another piece be stacked onto it (b) which causes the table to display the king symbol. (c) The table senses incorrect stacking, as well as (d) hover, which it uses to give hints.

Lumino checkers is based on blocks made from tapers. Figure 14 shows the underlying marker system. Both types of pieces contain black and white marker code, making them visually distinct from background and hover. Blocks can be stacked in any orientation and the use of large marker elements makes these blocks tolerant against moderate amounts of translation.

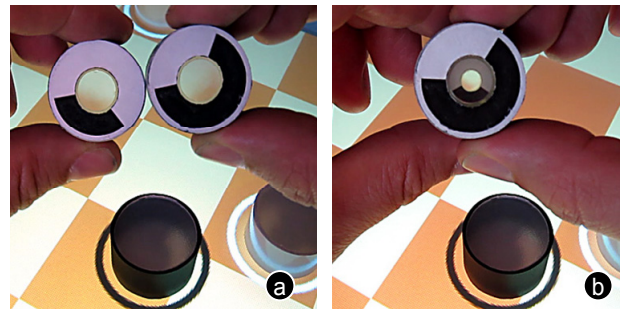


Figure 14: (a) The marker side of a white and a black checkers lumino. (b) Both pieces stacked.

Multi dials

Luminos allow making tangible tabletop widgets, similar to *slap widgets* [34] or the dial control in *photo helix* [17]), but multi-layer.

Figure 15 shows an example, a multi-dial. The user places a color correction dial on a low contrast photo with a reddish color cast. The user spins the dial until the color seems correct. Adding a contrast dial on top of the first dial allows adjusting contrast.

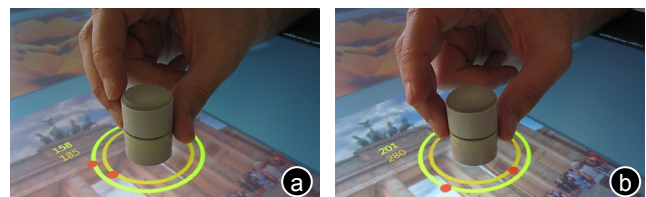


Figure 15: (a) Spinning the bottom dial of this multi-dial allows users to remove a color cast, the top dial adjusts contrast.

The user now picks up the multi-dial as a whole and uses it as a “stamp” to apply the same correction to other images (Figure 16). The settings of the multi-dial are preserved during stamping because they are “stored” in the spatial relationship between blocks.

Unlike multiple dials next to each other, a multi-dial occupies less space and can be carried or dragged onto another object without changing its setting.

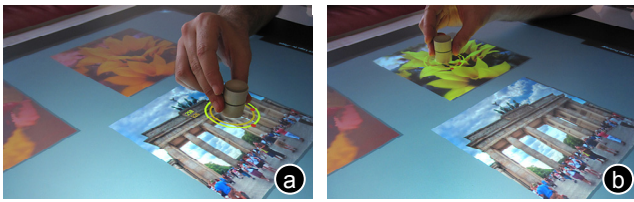


Figure 16: (a) Once a good color correction has been found, (b) the entire correction can be applied to other pictures by stamping them with the multi dial

We implemented multi dials using the same luminos as checkers. Dial luminos are application agnostic, so that the same dial components could be used to adjust band-wise audio volume or browse the day, month, and years of a time machine.

Lumino construction kit

The lumino construction kit (Figure 17) allows users to put together simple block constructions. The system automatically logs construction activities, checks the soundness of the hypothetical building, and informs the user about potential flaws and construction alternatives.

Figure 17a and b: The application recognizes when two or more objects are combined and starts logging and labeling them as a unit. Here the system recognizes when two structures have been connected using a bridge. (c) The system warns users about unsound structures and suggests improvements.

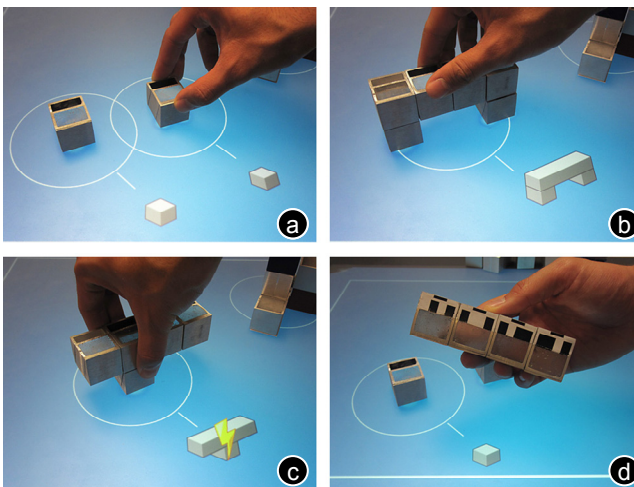


Figure 17: The lumino construction kit application (a) recognizes and labels individual components. (b) When two components are merged with a bridge the construction kit labels them as one. (c) Here it warns users about an unsound structure. (d) The markers we used.

The construction kit demo is implemented based on offset blocks as shown in Figure 17d. It offers three types of building block (1x1, 1x2, and 1x4) as well as decorative blocks, such as figurines, roof tiles, etc.

Each unit of the 1x2 and 1x4 building blocks contains its own distinct marker; this allows the table to tolerate clipping resulting from overhanging blocks. Top pieces, such as shingles, have a slanted shape to physically prevent users

from trying to place objects on them; they do not contain glass fiber bundles.

The lumino construction kit is a simple envisionment of how a future application might support architects. Luminos allow the table to keep track of the prototypes and design explorations on the table as the user is creating them. This allows the table to take on the role of an assistant, here a civil engineer. The table might critique the construction and point out construction flaws, display piece lists, running totals of construction cost, and the availability of the work materials contained in the current designs, snapshot the 3D structure of designs to render and print or replay and juxtapose structure later as part of a design critique.

Implementation

All demo applications were implemented on *Microsoft Surface* in C# and C using the Open Computer Vision Toolkit *OpenCV* [26].

LUMINO MARKERS AND CONSTRAINTS

In the demo section we showed two specific marker designs. In this section, we discuss the wider design space of lumino markers in order to help application designers optimize for their respective requirements.

Given that luminos need to convey k -times more information than traditional 2D markers, there is no single block-plus-marker design that covers all scenarios. Instead we optimize either for size of the block set (e.g., for use in construction kits) or for flexibility of individual blocks (e.g., for use in 3D tangible widgets).

Different block designs afford different markers; we therefore organize our discussion by block type.

1. Making straight blocks flexible is exponential

Straight bundles are the most limited design. As mentioned earlier, straight blocks generally require blocks to line up in order to assure that markers do not occlude. Within limits, we can relax these constraints though.

We can implement a multi dial based on straight rolls if each block uses a different ring for its marker (Figure 18a). A feature on each ring allows the table to track the rotation of each block, enabling us to create a multi-dial control.

The recursively generated marker sets Figure 18b-d allow multiple blocks to share the same ring. The marker sets in Figure 18e-g allow determining their rotation as well.

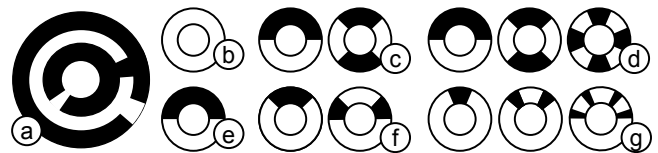


Figure 18: (a) Straight rolls allow for rotation if each block uses a different ring. (b-d) Marker sets that allows testing for the presence of up to 1, 2, and 3 markers, respectively. (e-g) Marker sets that allow determining their rotation as well.

The marker set in Figure 19a applies the same general concept to create a marker set that allows for unconstrained rotation & translation. No $n-1$ subset of the markers is able to fully occlude the remaining marker (Figure 19b).

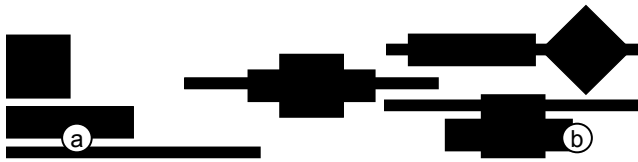


Figure 19: (a) Each element of this marker kit is twice as long and half as high as the previous. (b) No $n-1$ elements can fully obscure the remaining one.

While we can use these marker codes to add flexibility to straight blocks, the resulting designs tend to be inferior to the demagnification-based designs. A multi-dial based on straight bundles is less flexible than the demagnification-based design demonstrated earlier. More flexible designs require an exponential amount of resolution in the number of involved blocks, but with even higher factors than demagnification blocks (e.g., 16^n for the set shown in Figure 19). We will therefore generally choose demagnification blocks over straight blocks.

Compared to offset blocks, straight blocks require encoding layer information explicitly, e.g., using the method described by Bartindale and Harrison [5], which requires n choose $2 = O(n^2)$ marker bits. As a result, we will generally choose offset blocks over straight blocks.

2. Demagnification blocks offer maximum flexibility

Their exponential resolution requirement (Figure 9) inherently limits the stacking height of demagnification blocks. Within these constraints, however, demagnification blocks offer the higher possible flexibility. Since they virtualize the higher layers, they inherently allow users to rotate and translate blocks. This makes them well-suited for tangible input controls, such as multi-layer versions of *tangible widgets*, such as the multi-dial presented earlier. Figure 20 shows an example of how to make markers for demagnification blocks immune against moderate amounts of clipping.



Figure 20: (a) The Markers of the checkers game (b) By adding redundancy we can make markers immune against translation.

3. Offset blocks offer maximum marker capacity

We used offset blocks for the lumino construction kit demo applications, because they support a large number of blocks and a comparably large stacking height. Large numbers of blocks imply long block IDs and thus require a lot of information per marker strip. Large stacking height requires many strips to be packed into a single marker space.

Offset blocks require blocks to align to assure that marker strips are properly mapped to the bottom layer. However, they maximize the information density of their markers better than straight blocks or demagnification blocks. In the following we discuss how we fine-tuned offset blocks to maximize marker capacity.

Saving on registration information: Since offset blocks are constrained anyways, they allow us to cut down on the use

of *registration* marks. Registration marks are the part of the marker that allows the system to determine the position and the rotation of a block and thus locate the remaining part of its marker. If the spatial relationship between blocks is constrained, registration marks are only required for the bottom-level block. We accomplish this by attaching registration marks to the casing, *not* the marker area (Figure 21). This way, the table can see the registration for the bottom-most block, but the registration marks of the higher-level blocks remain invisible to the table and do not use up valuable space on the projected marker image.

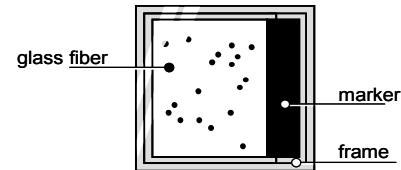


Figure 21: The reflective gray aluminum frame on this block also serves as a registration mark.

Removing registration can lead to substantial savings of the marker code. In *Microsoft Surface* markers, for example, registration accounts for close to 50% of the marker surface (In Figure 3a, the registration portion consists of the large dot in the center, the three white dots North, West, and South, and the black area East. The remaining up to eight dots form the marker ID).

Another reason for using offset blocks is that they encode which layer each marker is located in implicitly through the relative position of the respective marker strip (see Figure 23a).

Encoding block identity: Once we have removed the registration information, *all* remaining marker area can now be use for information about the block, such as a block ID. Figure 22a shows the code based on space-filling squares we use for that purpose.

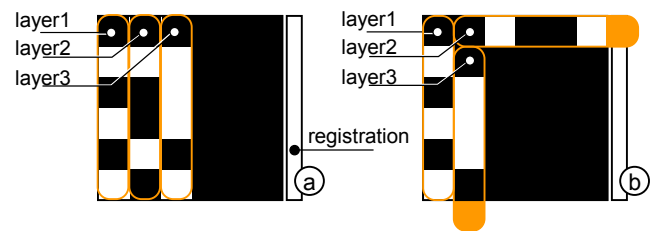


Figure 22: (a) Offsets blocks: the marker strip from each layer is mapped to its own column, here starting at the very left.

(Orange outlines for illustration only) (b) The same three blocks with the second one being rotated by 90° causes lowest bits to be clipped (highlighted in orange).

We reserve the code combination “00...0”, because an all-black marker would be indistinguishable from background. We also reserve “11...1” as it allows us to recognize when a solid reflective object is placed on top of the 3D structure, such as the user’s hand. We have used this to implement the recognition of touch and, by fine-tuning sensitivity, hover as shown in Figure 13.

All other codes are used to identify objects. For some applications we use a unique code for each object, which allows us to identify individual objects. For other applications, objects fall into categories within which pieces are interchangeable, such as a *roof tile* object in a construction kit application. When using offset blocks, markers never occlude each other, even when identical. Rather than assigning a different marker to every *object*, this allows us to assign a marker to an entire class, so that two objects from the same class bear the same ID. This allows us to use n bits of marker space for $(2^n - 2)$ different object *classes*. For the construction kit application, for example, we used 4 marker bits to encode 14 classes of objects.

Allowing 90° rotations: In general, offset blocks have to be fully rotation constrained—it is a limitation of this approach. That said, for some applications it is desirable to be able to place blocks in different orientations. In order to allow blocks in a construction kit to be oriented either “down” or “across” we register rotation using the encoding scheme shown in Figure 22b. All codes here are prefixed with a black-white sequence. The table decodes marker strips by starting at the marker’s origin, here the top left, which is always black. If the pixel below is white the first marker is oriented vertically; if the pixel on the right is white the first marker is oriented horizontally. The table removes recognized codes and repeats with the remaining data until all markers are parsed.

This encoding scheme can cause bits at the end of a marker strip to be clipped, as illustrated by Figure 23b. We therefore use these bits to encode non-critical information, such as block color.

Constraints

If necessary, we can assure that the constraints are met by adding mechanical constraints. Figure 23 and Figure 24 show a selection of mechanical constraints for round or square blocks. Figure 23c-g and Figure 24c-g are custom designed to be mounted to the casing of round and square blocks respectively. This eliminates the need to modify the glass fiber bundle and prevents constraints mechanics from occluding the bundle.

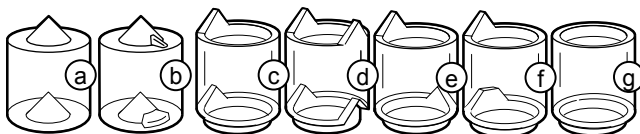


Figure 23: Examples of rotational constraints on round blocks: (a) unconstrained (b) range 45°. Constraints attached to casing: (c) fully constrained, (d) 180°, (e) forced rotation, (f) range 45°, (g) unconstrained. All are translation constrained.

Summary of block-plus-marker designs

Each of the presented glass fiber blocks has its specific strengths and limitations. Which block works best therefore depends on the requirements of the application at hand. Offset blocks allow for the largest sets of blocks. Demagnification-based blocks allow for maximum flexibility. And straight blocks, while generally inferior to the other two basic types, allow for the easiest fabrication.

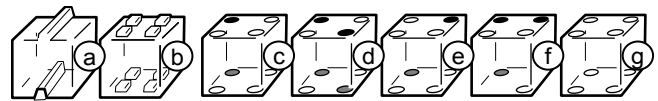


Figure 24: Constraints for square blocks: (a) 1D-translation, (b) grid. Constraints based on magnets attached to casing (black = north, white = south): (c) Fully constrained, (d) Allow 180° rotations, (e) Force 90° rotation, (f) Allow 0° or 90° rotation, (g) Unconstrained.

MAKING CUSTOM GLASS FIBER BUNDLES

The ability to prototype unusual block designs quickly has been essential to our ability to explore the design space. They enabled us to try out styles beyond mass fabricated fibers (e.g., *Plastecs* [30]), such as the slanted fiber bundles in offset blocks.

Here is the process we use to make fiber bundles. (1) We start with 30 meters of 0.75mm IR-conducting plastic optical fiber as raw material (actual glass fibers are hard to process with normal tools). (2) We cut the fibers into 30mm pieces by feeding them through a paper cutter. A stop assures equal length. (3) We fill fibers into a piece of round or rectangular aluminum tube. Shaking the block repeatedly for a few seconds causes fibers to align. (4) If we want a solid block this is the moment to inject slow-binding epoxy. We have made better experienced without though—instead we stuff fibers so tightly that they hold together by friction. (5) We verify fiber alignment against a calibration pattern. (6) Once the fibers are too tight to move, we stop shaking and instead twist the bundle back and forth to optimize fiber alignment further. (7) We sharpen the last fibers and force them in. (8) We prevent fibers from moving using a stop on the opposite side and polish the bundle on a belt sander or sandpaper. The ends of plastic fibers melt during sanding. This causes fibers to fuse together with their neighbors, making the bundle more robust. For the final polish, we use copy machine paper as super fine-grained “sand paper”. (9) Finally, we install the bundle and the marker in the aluminum casing.

CONCLUSIONS

In this paper, we demonstrated how to sense 3D arrangements of building blocks with a regular, unmodified diffuse illumination table. The main idea was (1) to use glass fiber bundles to move the visual focus of the table to the required location in the structure and (2) to use that same glass fiber bundle to rearrange marker images into a 2D configuration the table can read. We presented several designs for blocks and markers, and demonstrated the concept using three demo applications.

On a more abstract level, we introduced glass fiber elements as a means that allows tabletop cameras to see beyond the diffuser. We think that this concept has the potential to extend the application of diffuse illumination display systems—beyond the specific application presented in this paper. As future work, we plan on exploring touch input based on tangible glass fiber components.

ACKNOWLEDGMENTS

We thank everyone who helped us learn about glass fiber optics, especially Mr. Raitza from Loptek in Berlin, Jim

Basinet from Plastecs, MA, and Juergen Freitag from Schott in Mainz. We also thank Chris Harrison for sharing a pre-print of his paper and Christian Holz and Sean Gustafson for their comments on earlier drafts of this paper.

REFERENCES

1. Acceptance Cone wikipedia.org/wiki/Acceptance_cone
2. Anderson, D., Frankel, J., Marks, J., Leigh, D., Sullivan, E., Yedidia, J., and Ryall, K. (1999). Building virtual structures with physical blocks. In *Proc. UIST '99*, 71–72.
3. Anoto Pen. <http://www.anoto.com>
4. Azad, P., Gockel, T., Dillmann R. (2008). *Computer Vision - Principles and Practice*. Elektor International Media BV
5. Bartindale, T. and Harrison, C. Stacks on the Surface: Resolving Physical Order Using Fiducial Markers With Structured Transparency. In *Proc. Tabletop '09*, 4 pages.
6. Bartindale, T. Fiberboard. tom.bartindale.com/fiberboard
7. Couture, N., Rivière, G., and Reuter, P. GeoTUI: a tangible user interface for geoscience. In *Proc. TEI '08*, 89–96.
8. Crevoisier, A. and Polotti, P. Tangible acoustic interfaces and their applications for the design of new musical instruments. In *Proc. NIME '05*, 97–100.
9. Fiala, M. ARTag, a fiducial marker system using digital techniques. In *Proc. CVPR '05*, 20–25.
10. Fiber optics http://en.wikipedia.org/wiki/Fiber_optics
11. Fitzmaurice, W., Ishii, H., and Buxton, W. Bricks: Laying the Foundations for Graspable User Interfaces. In *Proc CHI '95*, 442–449.
12. Fujii, K., Shimamura, J., Arakawa, K., and Arikawa, T. Tangible search for stacked objects. In *Proc. CHI '03*, 848–849.
13. Gorbet, M., Orth, M., and Ishii, H. Triangles: tangible interface for manipulation and exploration of digital information topography. In *Proc. CHI '98*, 49–56.
14. Greenberg, S. and Buxton, B. Usability evaluation considered harmful (some of the time). In *Proc. CHI '08*, 111–120.
15. Guo, C., and Sharlin, E. Exploring the Use of Tangible User Interfaces for Human-Robot Interaction: A Comparative Study. In *Proc. CHI '08*, 121–130
16. Haller, M., Brandl, P., Leithinger D., Leitner J., Seifried T., and Billinghamurst, M. Shared design space: sketching ideas using digital pens and a large augmented tabletop setup. In *Proc. ICAT '06*, 948–959.
17. Hilliges, O., Baur, D., and Butz, A. Photohelix: browsing, sorting and sharing digital photo collections. In *Proc. Tabletop '07*, 87–94.
18. Ishii, H., Ullmer, B. Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proc. CHI '97*, 234–241.
19. Izadi, S., Hodges, S., Taylor, S., Rosenfeld, D., Villar, N., Butler, A., and Westhues, J. Going beyond the display: a surface technology with an electronically switchable diffuser. In *Proc UIST '08*, 269–278.
20. Jordà, S., Geiger, G., Alonso, M., Kaltenbrunner, M. The reacTable: exploring the synergy between live music performance and tabletop tangible interfaces. In *Proc. TEI '07*, 139–146.
21. Kakehi, Y., Naemura, T., and Matsushita, M. Tablescape Plus: interactive small-sized vertical displays on a horizontal tabletop display. In *Proc. Tabletop '07*, 155–162.
22. Kitamura, Y., Itoh, Y., and Kishino, F. Real-time 3D interaction with activeCube. In *CHI Ext. Abstr. '01*, 355–356.
23. McNerney, T. (2004). From turtles to tangible programming bricks: explorations in physical language design. *Personal and Ubiquitous Computing* 8, 5, 326–337.
24. Moeslund, T. and Granum, E. A Survey of Computer Vision-Based Human Motion Capture. In *Proc. CVIU '01*, 231–268.
25. Olwal, A., and A. Wilson. SurfaceFusion: unobtrusive tracking of everyday objects in tangible user interfaces. In *Proc. GI '08*, 235–242.
26. Open CV <http://sourceforge.net/projects/opencvlibrary>
27. Paradiso, J. A., Hsiao, K., and Benbasat, A. Tangible music interfaces using passive magnetic tags. In *Proc. NIME '01*, 1–4.
28. Patten, J., Ishii, H., Hines, J., and Pangaro, G. Senseable: a wireless object tracking platform for tangible user interfaces. In *Proc. CHI '01*, 253–260.
29. Peli E, Siegmund WP. (1995) Fiber-optic reading magnifiers for the visually impaired. *J. Opt. Soc. Am. A*, 12, 10, 2274–2285.
30. Plastecs. http://www.kissolar.com/fiber_optics.html
31. Promethean. <http://www.prometheanweb.com>.
32. Tactex Controls Inc. Kinotex Multi-touch sensors. http://www.tactex.com/files/multitouch_overview.pdf
33. Ward, A., Jones, A., and Hopper, A. (1997). A new location technique for the active office. *IEEE Pers. Comm.* 4, 5, 42–47.
34. Weiss, M., Wagner, J., Jansen, Y., Jennings, R., Khoshabeh, R., Hollan, J.D., and Borchers, J. SLAP widgets: bridging the gap between virtual and physical controls on tabletops. In *Proc. CHI '09*, 481–490.
35. Wilson, A. Depth-sensing video cameras for 3D tangible tabletop interaction. In *Proc. Tabletop '07*, 201–204.
36. Xu, D., Read, J., Mazzone, E., Bron, M. Designing and Testing a Tangible Interface Prototype. In *Proc. IDC '07*, 25–28.