Bringing Physics to the Surface

Andrew D. Wilson[1], Shahram Izadi[2], Otmar Hilliges[2], Armando Garcia-Mendoza[2], David Kirk[2]

[1]Microsoft Research
One Microsoft Way
Redmond, WA 98052

[2]Microsoft Research Cambridge
7 JJ Thomson Avenue
Cambridge, CB3 0FB

{awilson, shahrami}@microsoft.com, otmar.hilliges@ifi.lmu.de, {armandog, dakirk}@microsoft.com

ABSTRACT
This paper explores the intersection of emerging surface technologies, capable of sensing multiple contacts and often shape information, and advanced games physics engines. We define a technique for modeling the data sensed from such surfaces as input within a physics simulation. This affords the user the ability to interact with digital objects in ways analogous to manipulation of real objects. Our technique is capable of modeling both multiple contact points and more sophisticated shape information, such as the entire hand or other physical objects, and of mapping this user input to contact forces due to friction and collisions within the physics simulation. This enables a variety of fine-grained and casual interactions, supporting finger-based, whole-hand, and tangible input. We demonstrate how our technique can be used to add real-world dynamics to interactive surfaces such as a vision-based tabletop, creating a fluid and natural experience. Our approach hides from application developers many of the complexities inherent in using physics engines, allowing the creation of applications without preprogrammed interaction behavior or gesture recognition.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Design, Human Factors, Algorithms

Keywords: Interactive surfaces, game physics engines

INTRODUCTION

Emerging interactive surface technologies allow users to interact with the digital world by directly touching and manipulating onscreen content. People who use such systems often comment that this ability to touch digital content adds a physical or tangible quality to the interaction, making the virtual feel more real. Many interactive surface-based applications attempt to further highlight this "pseudo-physicality" by carefully designing interface objects that exhibit a sense of real-world behavior. One example is the common rotate and translate behavior found in many tabletop applications, where the interaction is analogous to moving a sheet of paper on a flat surface with one or more fingers [17, 18]. Although such an interaction may feel realistic, it is very much preprogrammed. For example, the way in which a digital photo rotates or translates is defined by specific interaction logic. Such techniques thus possess an inherent scripted nature which may break down once the user interacts with the system in ways unanticipated by the developer.

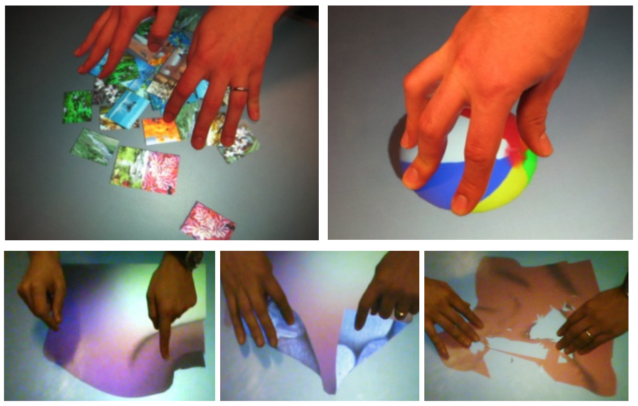
Figure 1. Some examples of physics-enabled interactions supported by our technique. Gathering and piling objects (top left), interacting with a ball using collisions and friction (top right), folding a cloth-like mesh (bottom left, middle), and tearing a mesh (bottom right).

In this paper we take a different approach for supporting pseudo-physical interactions on surface technologies—systems that are often capable of sensing multiple touch points and possibly even more sophisticated shape information. Specifically, we utilize physics engines used in computer games that simulate Newtonian physics, thus enabling interaction with digital objects by modeling quantities such as force, mass, velocity, and friction. Such engines allow the user to control and manipulate virtual objects through sets of parameters more analogous to those of real-world interactions.

While physics engines are comprehensive, they are also complex to master. Many coefficients and parameters are exposed to application developers, and controlling the simulation via user input is non-trivial, particularly when considering more than a single contact point. We present a simple yet powerful technique for modeling rich data, sensed from surface technologies, as input within the physics simulation. Our approach models one or more contact points, such as those sensed from a multi-touch surface [7,14], and also scales to represent more sophisticated shapes such as outlines of the whole hand or tangible objects on the surface [20,25]. This allows the user to interact with objects by friction forces and collisions, but avoids exposing users and application programmers to the complexities of the physics engine.

We demonstrate the applicability of the technique using a commercially available games physics engine and a prototype vision-based interactive surface. We highlight some of the interactions such an approach affords, for example gathering multiple digital objects or fine control of a virtual ball using friction and collision forces, as shown in Figure 1. Our system creates natural and fluid physics-based interactions "for free"—i.e., without the need either to explicitly program this behavior into the system or to recognize gestures. We also demonstrate the ability of advanced physics simulators to enable user interaction with more complex materials such as soft bodies and cloth, as shown in Figure 1.

We have experimented with various alternatives for simulating surface input within the physics world, the trade-offs of which are discussed in this paper. Our aim is to allow practitioners to understand the nuances of these alternatives so that they may further explore the intersection between interactive surfaces and physics. We also discuss both early experiences using our technique and some of its limitations.

RELATED WORK

Much work has recently been published on interactive surfaces, and particularly on direct-input tabletops [7,14,25]. Some of this work examines physics-like interactions. For example, Kruger et al. [17] explored simple notions of friction and motion to allow an object to be translated and rotated using a single point of contact. Other systems [12,22] have implemented gestures such as flicking, throwing, and pushing to add velocity and inertia to onscreen objects. Beyond tabletops, a number of interactive systems have explored the physicality of objects as a principle around which interaction is organized. Electronic files stack like real documents [19], move out of the way as if they have a solid form [23], peel back like paper [8], and move as if alive [5]. Rather than employing a full physics model, such systems use pseudo-physics in minimal ways to support subtle interaction possibilities. The present work looks to address a richer set of physics-enabled behavior, and focuses on problems related to input.

Various attempts to provide richer and more realistic 3D interactions have also been explored in the context of interactive tabletops [13]. Stahl et al. [24] describe a tabletop where objects float to the surface when accessed and sink back to the ground when no longer used. Hancock et al. present a set of methods to compensate for off-axis viewing [16] on multi-user tabletops, as well as techniques for shallow-depth interaction [15]. These systems allow basic manipulation of 3D objects but do not model interactions based on a physics simulation.

Realistic dynamics simulation has a long history in graphics and animation communities. Baraff [3] gives an overview of the main concepts for calculation of rigid-body dynamics. More recently, real-time collision detection and response with stable friction calculations became feasible [4]. Erleben et al. [9] provide an overview of current techniques and advancements that led to the development of sophisticated physics engines for simulation and gaming, such as PhysX, Havok, Newton, and ODE. However, these techniques have yet to reach the user interface and interaction research communities broadly.

One notable exception is BumpTop [1], which uses a physics engine to add real-world dynamics to the Tablet PC desktop. It supports notions such as collisions, mass, and inertia, and higher level constructs such as piling. The work nicely demonstrates some of the capabilities of a modern physics engine. However, the approach is based on a single point of input and menu-based selection. Compared to the rich means available for manipulating physical objects, this single-point input model can be limiting.

Multiple inputs are considered in the application of an early physics engine by Baraff with the Responsive Workbench [10]. This system simulates a 3D bimanual assembly task using two 6 DOF input devices and a stereoscopic display. An object may be manipulated with one hand by placing eight springs connected to the corners of a virtual cube rigidly attached to the user's hand. Bimanual interactions are supported by the superposition of forces from both hands. Thus, when one hand is released, the object snaps to the position and orientation of the other.

Interactive surfaces, particularly vision-based systems [14,20,25,26], allow the capture of rich sensor data that not only includes multi-touch data but also detailed shape information, such as images of the users' entire hands or other tangible objects near the surface. We present a technique for modeling this rich and diverse sensor input, representing these effectively as friction and contact forces in the physics simulation. These capabilities augment and extend the single contact model utilized in BumpTop and the bimanual approach in Responsive Workbench, supporting single-touch, multi-touch, contour-based, and tangible input using a single technique.

INTERACTIVE SURFACE INPUT

A contact on an interactive surface (e.g., a fingertip touching the surface) is most easily represented as a discrete 2D point. In the case of vision-based interactive surfaces, neighboring sensor pixels are usually grouped into contiguous regions or *connected components* [6], with the idea that each component corresponds to a contact. The center of the component is then easily calculated. This approach thus reduces each contact to a point-based representation, regardless of shape.

This point representation of contacts allows application developers to think in terms of familiar point-rectangle hit-testing algorithms typical of traditional cursor-based systems, but it imposes significant limits on interaction. First, point-based hit testing may fail to catch the user touching a virtual object if the contact is not compact, as when the user places a large part of their hand on surface. In this case, the center point may lie outside the target object. Secondly, tracking point-based contacts to deduce motion can lead to difficult problems related to correspondence. For example, consider two fingers that move so near to each other that they now appear as a single contact. The choice of which of the original contacts to eliminate can result in very different motion interpretations. Finally, reducing contacts to points prevents users from drawing on the full spectrum of manipulation styles found in everyday life. Consider the multiple grasping strategies illustrated in Figure 2, for example. Each gives a different feeling of control in the real world. Ultimately, it seems that point-based systems encourage the exclusive use of index fingers on interactive surfaces.

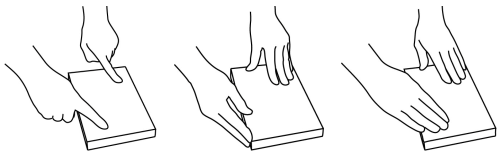

Figure 2. Multiple grasping strategies for rotating a resting object.

One approach for preserving more information about contact shape is to determine the *bounding box* of the contact, or the major and minor axes of an *ellipse model* that approximately fits the shape. These approaches work well for compact contacts (e.g., fingertips) and certain hand poses [27], but less so for complex shapes and their motion. Alternatively, the shape may be represented more precisely as a *polygon mesh* by calculating each contact's contour, represented as a closed path [11]. Another technique is to take pixels lying near the contour by computing the *spatial gradient* using a Sobel filter [11].

These approaches allow us to support even more sophisticated representations of user input. We would like to combine this broad fidelity of input with advanced physics simulations to expand the vocabulary with which we can manipulate digital objects. Our aim is to make manipulation of digital objects less scripted, using rich and varied interaction techniques and strategies.

PHYSICS SIMULATIONS

Today's physics engines enable the creation of real-world mechanics and behavior in graphical applications while hiding computational complexity. They employ many physics concepts such as acceleration, momentum, forces, friction, and collisions. In addition to rigid bodies, many systems model particles (for smoke, dust, and so forth), fluids, hair, and clothes. Virtual joints and springs give "rag doll" characters and vehicles appropriate articulation, and materials can be programmed with specific properties—so that ice is slick, for example. The present work primarily concerns *contact forces,* such as those due to collisions and friction between simulated bodies.

The handling of collisions is typically divided into *collision detection*, the determination of whether two rigid bodies are in contact, and *collision response*, the application of appropriate forces if they are in contact. For example, the collision of a cube falling on the floor may be detected by considering the intersection of the faces defining the cube with those of the floor. The change in motion of the cube as a result (the response) is a function of mass, inertia, velocity, the point of contact with the floor, and other factors.

Friction forces resist motion when the surface of one body stays in contact with the surface of another. If two surfaces are moving with respect to each other, *kinetic friction* opposes the forces moving the bodies. If two surfaces are at rest relative to each other, *static friction* opposes forces that would otherwise lead to the motion of one of the bodies.

SURFACE INPUT WITHIN A PHYSICS SIMULATION

In order to interact appropriately with virtual objects in a physics engine, surface contacts must be represented within the simulation. These engines have enormous potential and flexibility. Accordingly, there are many strategies for modeling surface input in the physics world. We briefly describe these strategies here, and give more detail later.

- *Direct force*: A force is applied where a contact point touches a virtual object. The force direction and magnitude is calculated from the contact's velocity and size if available.
- *Virtual joints and springs*: Each contact is connected to the virtual object it touches by a rigid link or spring, so that the object is dragged along with the contact.
- *Proxy objects*: Contact points are represented as rigid bodies such as cubes or spheres. These bodies are an approximation of the contacts, and interact with other virtual objects by collisions and friction forces.
- *Particles*: Where additional information about a contact's shape is available, multiple rigid bodies—or particles—are combined to approximate the shape and motion of the contact more accurately. This allows for better modeling of interaction with the whole hand or other contacts such as tangible objects.
- *Deformable 2D/3D mesh*: Another approach for modeling more sophisticated shapes is to construct 2D or even 3D meshes if appropriate sensors are available.

It would seem that a deformable 3D mesh of the hand would achieve the highest degree of fidelity. But a number of difficulties exist with this approach. First, most interactive surfaces provide sensing at or near the surface only, not full 3D shape. Similarly, because the manipulated object exists only on the (flat) display surface, the 3D shape of the hand, if captured, would not conform to the object and so would not reflect the shape of a real hand grasping a real object. Finally, constructing such an animated mesh is difficult, requiring robust tracking of features and accurate deformation of the 3D object.

That leaves us with a key question that motivates this paper: How does one best use surface input to interact with advanced physics simulations in useful ways? We describe our rationale and experiences in implementing and evaluating the above alternatives. The main contribution of this paper is a novel *Particle Proxy* technique that retains most of the benefits of mesh-based representations—in particular, a high fidelity of interaction—but is considerably easier for application programmers to implement.

Applying External Forces Directly

A typical strategy for moving an object on an interactive surface in response to touch is to continually update its position to match the touching contacts' position. Generally we will refer to this manner of moving objects by setting its position and orientation directly generally as *kinematic control*.

Within a physics simulation, however, the most common way for an application to control the movement of a rigid body is to apply one or more forces. For example, a spaceship in a game might have thrusters on either side of its body. The ship may be propelled forward by applying forward force at the location of both thrusters. If one of the forces is applied in the opposite direction, the ship will turn. Rotation is the by-product of *torque*, which occurs when forces are applied off-center (of mass) because different "parts" of the body are moving at different speeds.

From a programmer's point of view, this approach is very different than moving the ship by setting its position. To effect kinematic control within a physics simulation, we must calculate the precise force and torque required to move the object into its target position. This method of positioning an object ensures correct collision response with other bodies in the simulation. In comparison, directly setting the position of the body within a simulation can lead to unstable and unpredictable results. Absolute positioning might be analogous to teleporting a real object from one location to another. Issues such as interpenetration whereby objects become partially embedded in each other, can arise.

A natural strategy for moving an object to follow a contact on an interactive surface is therefore to consider that each contact imparts a friction force to the body it touches according to the contact's motion (and presumed mass). These multiple friction forces may be applied to the body, as in the example of the spaceship. Unfortunately, to calculate the forces necessary to match a contact's movement, all other external forces acting on the body must be taken into account and counteracted. These may include friction forces and collision responses that are difficult or impossible for application developers to obtain.

This difficulty extends to considering forces corresponding to surface contacts. In the case of multiple contacts, the correct friction forces corresponding to each contact must be determined simultaneously. Consider the case where one or more of the contacts exhibits static friction. Recall that static friction exerts a force that counteracts forces that would otherwise lead to a body's motion. For example, if one contact "pins" an object so that it will rotate due to the motion of another contact (e.g, Figure 2, left), the application of correct friction force due to one of the contacts requires knowing the friction force due to the other.

In fact, at the heart of any physics engine is a sophisticated constraint solver that addresses this very problem. Without essentially constructing a new solver within the physics-engine, or without access to internals of the existing solver, it would seem impossible to correctly apply contact forces directly. Even if it were possible to change the solver or embed another, such an approach would go against the spirit of the present work, wherein an existing full-featured physics engine is leveraged rather than built from scratch.

One possible solution is to treat all frictions as kinetic. But this poses a problem in the "pinning" example. Because kinematic friction forces only act in the presence of relative motion, the counteracting force that keeps the "pinned" part of the object stationary must first move. Thus, this approach results in a somewhat viscous and slightly unpredictable feel when moving objects.

Connecting to Objects with Joints and Springs

Another kinematic approach, used in systems such as BumpTop [1], is to connect virtual objects and an input contact using a *joint*. Think of this as an invisible piece of rope of predefined length that is tied to the object at a particular anchor point. The object is then pulled along using this rope.

By attaching a joint off-center, the object is subject to both force and torque—allowing the object to move and rotate using a single connection. In our earlier pinning example, one joint attaching a stationary contact point to one corner of the object would serve as a pivot point. A second joint attaching a second moving contact point to an opposing corner would cause the object to spin around the first contact point.

This approach is not well suited for multiple simultaneous contact points, particularly those pulling in opposite directions. Whereas in the real world, multiple contacts pulling in opposite directions on an object would result in the fingers sliding, or the object deforming or tearing, neither behavior is supported by joint constraints on a rigid body. It is thus easy for multiple rigid constraints to overconstrain the simulation, resulting in numerical instability and unpredictable behavior.

Springs can in part alleviate some of these issues by providing more flex in the connection. However, a trade-off exists between the elasticity of the spring and how responsive the connected object is to contact motion (springs should be fairly short and rigid to allow for a faster response). Problems of numerical stability and uncontrolled oscillations are likely [10]. Another approach is to allow the joint or spring to break in these situations, but this can easily lead to situations where objects fly out of the user's reach.

SETTING THE SCENE FOR A NEW TECHNIQUE

We have so far described two techniques that one would typically employ in single-point physics-enabled applications, and discussed the limitations of both in terms of modeling multiple contacts. The modeling of such input is challenging but only part of the story with respect to the limitations of these approaches.

First, as we described earlier, contacts are not always discrete 2D points, and it may be desirable to match the shape of the contact input closely. It is unclear how one would model more sophisticated shapes and contours with either of these initial approaches. Second, the above techniques address the case where the user touches the object directly, thereby moving the object by friction forces. Neither of these approaches addresses the movement of objects by collision forces, i.e., from contact forces applied to the side of the object (as in Figure 2, middle).

The next section presents a technique which handles friction and collision forces in the same framework and is easily extended to handle shapes of arbitrary contour. In doing so, it addresses many of the difficulties of the previous techniques.

Proxy Objects

The idea of *proxy objects* is to incorporate into the physics simulation a rigid body for each surface contact. These bodies are kinematically controlled to match the position of the surface contacts and can be thought of as incarnations of contact points within the physics simulation. Because they are regular rigid bodies, they may interact with other rigid bodies in the usual ways: either by collision or friction.

The proxy approach carries various benefits such as hiding the complexity of force calculations (in fact, hiding almost all physics aspects) from the programmer, while avoiding the difficulties of the previously described approaches. It leverages collision as well as friction forces (both static and kinetic) to model rich interactions such as pushing, grabbing, pinching, and dragging. Proxy objects interact with other objects in the simulation through the means provided by the engine. Finally, this approach avoids unnecessary strain on the solver (e.g., inserting extreme force values) and resulting unstable simulation states.

Proxy objects are created and positioned for each point of contact. Most simply, a single shape primitive such as a cube or sphere may be used for each contact. When a contact initially appears, a ray casting calculation is performed to determine the 3D position of the proxy so that it touches the underlying object, as shown in Figure 3. An interesting alternative to using a sphere or cube as a proxy shape is to create a thin capsule, box, or cylinder which stretches from the 3D camera near plane to the surface of the touched object (see Figure 4). This kind of proxy will collide not only with objects resting on the same plane as the touched object (or “floor”), but also objects that are in mid-air, or stacked on other objects. Such behavior may correspond more closely to user expectations.

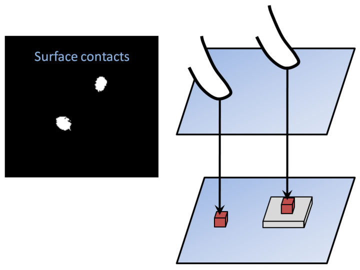

Figure 3. Proxy objects (red) are placed in the 3D scene, one per contact (left image), by ray casting.

As the sensing system provides updates to a contact position, the corresponding proxy object is kinematically controlled to match the updated position. This is done, as described earlier, by applying the necessary forces to bring the proxy object (of known mass) to the updated position of the contact. This scheme allows users to leverage collision forces to push objects around or grab objects by touching them from two opposing sides.

A small change in the kinematic control enables the proxy object to exert friction forces when it lies on top of another rigid body (as when the user touches the top surface of a virtual object, for example). In particular, only forces tangential to the touched object are applied to match the contact position. As with regular dynamic bodies, gravity is still included as an external force. In the case where gravity is directed into the surface, the proxies thus exert downward force onto other objects and cause friction forces. This hybrid kinematic-dynamic control of the object can be implemented by direct addition of forces to the proxy rigid body, or by a prismatic joint constraint on the body’s motion. The simulated weight of the finger on the object may be adjusted by changing the mass of the proxy object, while the material properties of the virtual objects may be adjusted by changing static and kinetic friction coefficients.

The main advantage of the proxy object technique is that it leverages the built-in capability of the physics engine to simulate both friction and collision contact forces. Most significantly, because the calculation of contact forces is handled entirely by the built-in physics engine solver, the combined effect of simultaneous static and kinetic friction forces due to multiple proxy objects is handled correctly. These friction forces enable users to translate and rotate objects (through opposing forces) that they touch directly.

From Points to Contours: Particle Proxies

Thus far we have approximated each touch point as a single proxy object. This permits a simple, fast implementation, and lends itself to sensing systems that report only contact position and no shape information, as well as applications that favor interaction with the fingertip or stylus.

Some interactive surfaces provide shape information, such as an oriented ellipse, bounding box, or full polygonal contour. The idea of the *particle proxy* is to model the contact shape with a multitude of proxy objects (“particles”) placed along the contour of the contact (see Figure 4). Particles are added and removed as contours change size and shape. A practical implementation involves creating a new set of proxy objects for the contour at the beginning of each simulation frame, and destroying all proxy objects after the physics simulation has been updated. Even though the proxies will be destroyed after the physics update, each enacts collision and friction forces during the update.

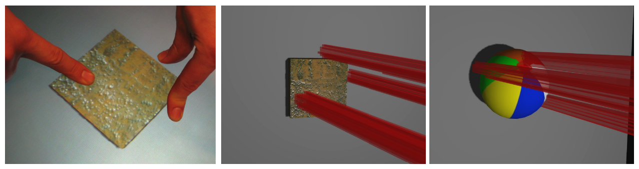

Figure 4. Particle proxies approximate the shape of multiple contacts. Left: applying friction from the top, and collision from the side to grip a block. Middle: Long proxy particle objects (red) illustrated. Right: Particle proxies accommodate non-flat objects. Here we model each proxy as a long capsule.

The advantage of the particle proxy approach is twofold. First, collisions appear to be more correct because they more closely follow the shape of the contact. This is particularly important when using the flat or side of the hand, tangible objects, or generally any contacts other than fingertips (see Figure 5). Similarly, the distribution and magnitude of friction forces on the top of an object are more accurately modeled. For example, the flat of the hand may exert more friction than the tip of a finger (Figure 2, right) by virtue of having more particles assigned to it. Likewise, a single contact turning in place can exert friction forces to rotate an object. Unlike the single proxy model, each particle is placed (ray cast) separately, so that a contact can conform to irregularly-shaped 3D virtual objects (Figure 4).

As in the single proxy object model, each particle is kinematically controlled to match the movement of the contact to which it belongs. Generally, the velocity of a point on the contour can be computed by examining the contact's contour in the previous frame. This calculation may be simple, as with an ellipse model, or more complex, as with a polygonal contour.

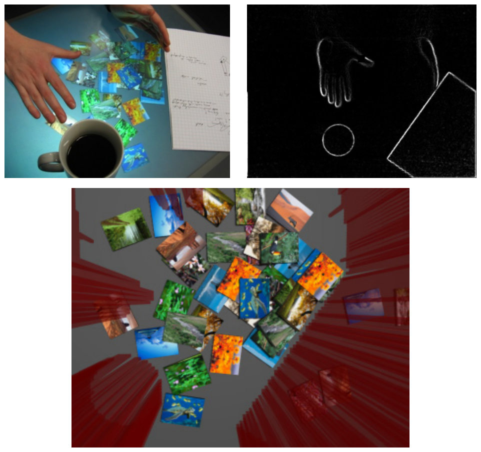

Figure 5. Top left: Particle proxies approximate the shape of multiple contacts and physical objects (cup, notepad). Top right: Sobel image shows contact contours. Bottom: Particles (red) as they project into the 3D scene.

From Tracking to Flow

One difficulty in basing velocity calculations on tracked contacts is that tracking can fail, particularly when the user is using less constrained grasping postures such as the edge or flat of a hand rather than the more cursor-like index finger. In these cases, components can split and merge in ways that do not correspond to how we see the physical input, leading to erroneous velocity calculations, and ultimately in the case of our physics simulation to unpredictable motion.

An alternative approach is to calculate the motion of the particle independently of any tracked contact information. For example, local features of the image may instead be tracked from the previous frame to calculate velocity. Simple block matching of the sort used in optical flow [2] is one such technique (see Figure 6).

When using local motion estimates, the tracking of discrete contact objects and exact contours may then be avoided altogether by placing proxy particles at image locations with high spatial gradient (e.g., Sobel filter [11]). These pixels will lie on contact contours. The particle proxy technique is summarized as:

```
compute Sobel image from surface input
for each pixel with high spatial gradient:
    ray cast into scene to determine initial particle position
    add particle rigid body to physics simulation
    compute contact motion at particle (e.g., from flow)
    compute corresponding tangential motion in scene
    apply force to particle to match scene motion
    apply downward force (gravity) to particle
update physics simulation
destroy all particle rigid bodies
```

The instantaneous, piecewise nature of the shape and motion calculations of the flow-based particle proxy method possesses important advantages. First, the friction and contact forces lead to more stable physics simulation results than if shape and motion were calculated from discrete tracked objects. Second, because the technique makes few assumptions regarding the shape or movement of contacts, it imposes few limits on the manipulations a user may perform, whether leading to collisions, friction forces, or combination thereof.

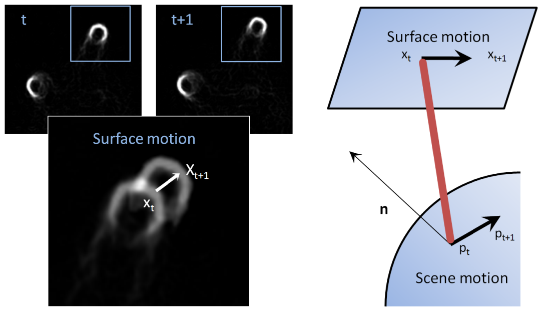

Figure 6. Computing flow of a particle. Surface motion at a point $\mathbf{x}_t$ is computed by comparing successive edge images. Corresponding tangential motion in the scene is computed by ray casting image point $\mathbf{x}_t$ into the 3D scene, to obtain 3D point $\mathbf{p}_t$. Point $\mathbf{p}_{t+1}$ is found by projecting image point $\mathbf{x}_{t+1}$ onto the tangent plane formed by $\mathbf{p}_t$ and object normal $\mathbf{n}$.

NEW PHYSICS-BASED INTERACTIONS

Our goal in introducing more detailed surface input types into a physics simulation is to enable a wide variety of manipulation styles drawn from real-world experience. While we have only begun to explore some of the possibilities that these techniques afford, here we consider a few which we believe are noteworthy.

Manipulation Fidelity

The ability to exploit detailed shape and motion information has broad consequences when considering the manipulation of even the simplest objects. Free-moving virtual objects can be moved by any one of a variety of strategies that combine collisions against the contours of hands and fingers with static and kinetic frictions. Because all three kinds of forces can be employed simultaneously, the overall impression is one of unusually high fidelity. An interesting example is the manipulation of a ball that is free to roll on the surface: it may be compelled to roll, spin, stop, or bounce in a surprisingly precise fashion, using a single light touch, multiple touches, or the flat of the hand for stopping power (Figure 1). Physical objects can also be integrated at no cost, allowing a variety of interesting tangible behaviors (see Figure 7 for some examples).

Gathering

The ability to sense and process contours, as well as distribute friction forces piecewise across the virtual space, enables the manipulation of many objects at once, much as one might move a group of small objects spread across a table (see Figure 1, top left). Users may use the edges of their hands (or even arms) to collide against many objects at once, or use the flats of multiple hands to apply friction forces. For interactive surfaces able to sense physical objects, an interesting possibility is to use a ruler to move and align multiple objects.

Manipulating Objects in 3D

Modeling virtual objects and input in 3D enables interesting yet familiar interactions. For example, a flat object resting on a larger flat object may be moved by tapping its side or applying friction. Depending on the masses and frictions involved, it may be necessary to hold the larger object in place. It is thus important for the designer to tune masses, frictions, and appearances to match user expectations.

If the interaction is limited to collision forces from the side and friction forces from the top, however, the manner in which a user may place the smaller object on top of another is unclear. Ramps, see-saws, and other constructions are possible, if somewhat contrived. In certain cases it may be possible to flip one object onto another through the application of sufficient friction forces to one side of the object.

When the objects to be stacked are thin, such as cards representing documents [23, 1], one approach is to give the top and bottom surfaces of each object a cambered shape that allows the user to raise one side by pressing down on the other. The user may then move another like-sized card under the tilted card (Figure 7, top left). This behavior corresponds to our awareness that in the real world even “flat” objects such as cards and paper have some 3D shape that is often intuitively exploited to manipulate them.

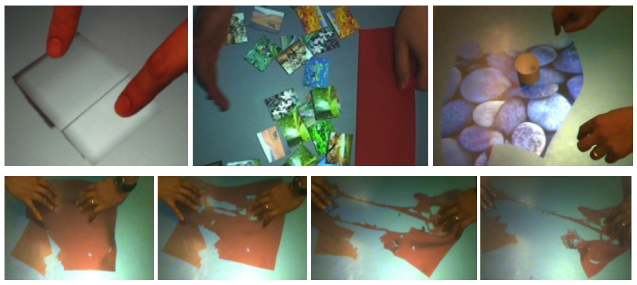
Figure 7. A cambered card is slid on top of another by pushing one side while holding the bottom card in place (top left); tangible objects interact with the digital: e.g., the side of a physical piece of card used to gather up objects (top middle), or a real cup used to pin down a cloth (top right); tearing a mesh in two by applying opposing forces (bottom).

Cloth and Soft Bodies

We have used rigid bodies such as boxes and spheres to explain our interaction techniques. However, in the real world many objects are not rigid but are instead soft, malleable, and can deform or dissolve when forces are exerted on them. Examples include rubber, clothes, and paper.

In addition to rigid body dynamics, most available physics simulations offer some form of support for soft body, cloth, and fluid simulation. As all interactions in our model are conducted through collision or friction forces, the model can be applied to arbitrary virtual objects. For example, it is possible to crumple a piece of cloth with a grasping interaction using all the fingers of one hand. The crumpled cloth can then be straightened by pushing down with the flat hand. One can even tear paper-like objects apart by applying forces in opposing directions on two corners (Figure 7).

Another possible application would allow soft volumetric bodies to be squished so as to fit into cavities or compressed so as to slide underneath other objects. Soft materials could also be used for terrains; deformation could be triggered by users digging their fingers into the terrain, using their whole hands to form valleys, or using a cupping gesture to create elevations. More open-ended and free-form interactions with particle systems (e.g., simulating sand) and fluids can be imagined in a gaming context.

USER STUDY

To further understand and evaluate the utility of the techniques described in this paper, an exploratory experiment was performed. The following questions were addressed: Are users able to comprehend and exploit the openness of interaction that the physics model affords? Is the interaction of sufficient fidelity? Is it discoverable and predictable? Do users notice and value the added fidelity, or do they just expect kinematic control? Ultimately, how do users express their expectations in the physics-enabled manipulation?

The study exposed 6 participants to 3 simple physics-enabled tasks (as shown in Figure 8), and analyzed various behavioral and experiential aspects of interaction during task completion. The 3 male and 3 female participants came from a range of backgrounds, and all had normal (or corrected to normal) vision and no mobility impairments. The experiment was conducted on an early prototype of Microsoft Surface [20], using the Nvidia PhysX gaming physics engine [21]. This allowed sensing of multiple fingertips as well as outlines of hands and forearms.

The experiment utilized a 3x3x2 within-subjects (repeated measures) design. Each participant worked through the three puzzles in each of three interaction techniques: Joints, Proxies, and Particles. Pilot testing included a fourth condition, Direct Forces, but this was dropped during further testing (as explained below). For techniques that do not intrinsically support collisions, a simple collision model based on kinematic objects was applied, allowing interactions from the top and the sides of an object. In addition, we hypothesized that the presence of visual feedback showing users precisely where their input is applied might improve the discoverability of each technique. Visual feedback of Joints was represented as red lines drawn from the contact point to the anchor point on the object (these disappeared if the joint was broken); Proxies as red cubes at each center point where contact was sensed; and Particles as smaller red cubes per pixel in the contour image. We ran each technique with and without visual feedback as our third independent variable.

The task setup (see Figure 8) was as follows. In Task 1, each of four spheres and rectangles were placed exactly on matching targets; each object disappeared upon proper placement. In Task 2, an assortment of objects of different shapes, sizes, and masses were sorted onto the left or right portions of the screen depending on their color. In Task 3, a cylindrical object was steered from a set starting position (far top right of photo) to a target (shown in red) by passing several waypoints (shown in blue) without dropping the object from a platform (which caused the task to restart).

The tasks were presented in the same order to each participant, whereas the order of interaction techniques was counterbalanced across participants using a Latin-square design. Experimentation occurred in two main phases (with visual feedback of the input and without), presentation of which was, again, counterbalanced across participants. During the experiment, participants were not given any direct instruction, but had several attempts to try out each new puzzle. Participants performed each task twice (excluding any training), under experimental conditions, to provide an average completion time for each condition. Participants were interviewed informally after completing their session.

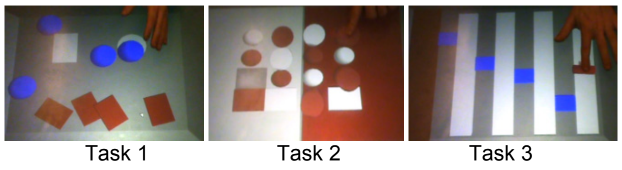

Figure 8. Task 1: Exact positioning of boxes and spheres. Task 2: Sorting by color. Task 3: Steering.

Early Issues with Direct Forces

Initially, the Direct Forces technique was implemented by applying a smooth velocity at a given contact point on the object, computed as a measure of the displacement between the contact's current and last positions (i.e., kinetic friction). This seemed a fair approximation for modeling surface input as direct forces. However, our pilots questioned the efficacy of this technique. Specifically, users found it difficult to complete tasks that involved accurately positioning objects; i.e., moving and then stopping an object at the target location. Moving the object could be performed reasonably, but to stop it the user needed to counteract the motion in the opposite direction. This often led to excessive velocity applied in the reverse direction, causing objects to "overshoot" the target. Consequently, performance with this technique was so poor that we felt it needed no further evaluation. Based on these issues and feedback from the pilots, we excluded this technique from analysis.

Initial Results and Observations

Although this was only an initial exploration, we observed many promising interactions and forms of gesture within the study. Users seemed aware of the potential of this new type of environment and exploited the physics-based system's facilitation of experimentation, and we observed many new interaction strategies.

Kinematic Control and the Curse of the Single Finger

Figure 9 shows the completion times for all tasks. Joints provide kinematic control that closely mimics drag-and-drop behavior, and thus facilitate easy positioning of objects. This is reflected in the results. After some experimentation, there was a moment when users discovered that the object was under familiar kinematic control. Users commented that *"my hands are like magnets"* or *"I can press hard and stick my fingers."* Of course, pressure and magnetism were not factors at play here (in fact, post-study interviews revealed that participants were unsure of the general principle behind the Joints technique). Nevertheless, users performed the task rapidly after discovering the object was somehow fixed to their fingers.

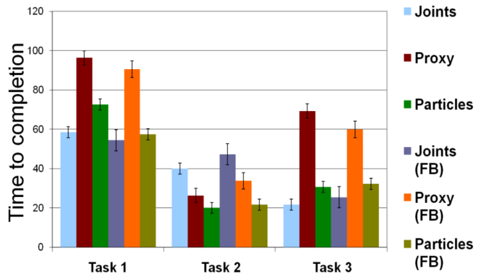

Figure 9. Task completion times. **FB** denotes conditions in which visual feedback of the user's input was provided.

However, the quantitative results tell only part of the story. During the study we also observed many limitations with the kinematic approach. The discovery of this type of essentially drag-and-drop behavior in the Joints condition led users to predominately interact with a single finger and with a single object at a time. Even rotations of an object were predominantly undertaken using a single finger [17].

Experimentation with multi-fingered or bimanual techniques was therefore rare in the Joints condition. During informal interviews, users commented that the condition was *"limited"* and *"less satisfying"* than the other techniques even though they performed the tasks rapidly. Although it is too preliminary to draw significant conclusions, it does suggest the need to measure more than task completion time when evaluating such physics-based techniques.

Users also had a poor understanding of how collisions were supported in a kinematic approach such as Joints. We observed many instances where accidental collisions caused by hit-testing on the side as opposed to the top of the object would cause an object to move away from the user and cause a great deal of confusion. This makes us revisit whether a kinematic-plus-collision model makes any sense to the user at all: Why indeed should an object only be sticky when you touch its top as opposed to its sides? This actually led some users to infer that objects were magnetized in a way that supported both attraction (when touching the top) and repulsion (when colliding with the sides).

Using Feedback to Go Beyond Kinematic Control

As shown in the results, feedback did not play a significant role in the Joints condition, as one might expect given the familiarity of the approach. Feedback played a more significant role for Particles in Task 1. After some training time, users discovered they could interact with more than just their fingertips. Bimanual "cupping" and "throwing and catching" techniques were devised to rapidly move objects to target positions (Figure 10). These strategies, and the general level of fine control, enabled users in the Particle condition to obtain completion times comparable to more kinematic approaches. During interviews, users reflected positively about the interactions Particles afforded.

However, these types of contour-based bimanual interactions could not be utilized with Proxy objects—although participants did try. In fact, in many cases, a hand gesture on the surface would be poorly approximated as a single proxy (the center of mass of the contact shape), causing objects to slip through a hand or causing other peculiar hit-testing behavior. Multiple fingers were used to reorient boxes effectively, but overall, bimanual control was rare.

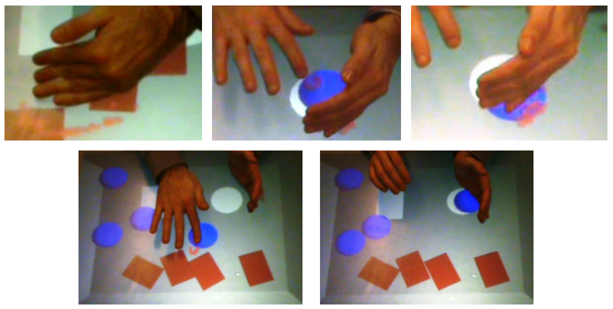
Figure 10. Using contours of the hand to move multiple boxes (top left); provide a barrier to change ball motion and position smoothly over target (top middle, right); throw and catch an object from a greater distance (bottom).

While the "drag and drop" nature of Task 1 clearly favored kinematic control such as that offered by the Joints approach, Task 2 offered a clear advantage to concurrent manipulation of multiple objects for rapid sorting. As might therefore be expected, use of both Proxy and Particles techniques, which seemingly promoted multi-touch interaction, led to faster completion times in this task (Figure 11).

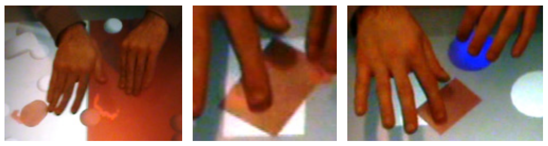
Figure 11. Two-handed and multi-fingered strategies adopted in the proxy and particle conditions. Coarsely moving objects using both hands (left), two-fingered rotation by applying torque to align a box (middle); fine-grained movement of two objects using a single finger of each hand (right).

Coming to Grips with Non-planar Objects

Another specific trade-off in our design was that the rigid body cubes in the Proxy condition only provided an effective means for interacting with flat objects. They provided little grip of spherical objects (or more complex 3D meshes). This was clearly evident in the final task where the Proxy cubes struggled to keep the cylindrical object under control, as shown in Figure 8. In this task, we found users often reverted to point-based interaction to control the small non-planar object; the use of contours was infrequent. However, our initial results suggest that Particles still outperform Proxy objects for these purely point-based interactions. This suggests that for scenarios where touch-only input is available, the Particle model subsumes the single Proxy object model.

DISCUSSION

The results of the user study and general experimentation suggest that while the more familiar kinematic approaches (somewhat inevitably) offer more predictable control in some situations, the particle proxy approach can offer comparable performance while providing new modes of interaction (such as cupping the ball in Figure 10). That our study participants were able to devise new manipulation strategies from limited feedback and training is encouraging. With more time, we expect users to further draw on their experience with real-world manipulations.

There are a number of ways in which our interactive surface simulation does not match the physics of the real world. In suggesting that we abandon familiar, kinematic point-based control in favor of strongly physics-based techniques, an important consideration is whether users are able to negotiate these differences.

First, while in the real world one might apply more or less force to control friction, our system has no sense of how hard the user is pressing. When using particle proxies, the amount of friction applied is instead proportional to the number of proxies applied to the object, which itself is related to the surface area of the touch. For example, a single finger moving across an object will apply less friction than multiple fingers. Not surprisingly, this distinction was generally lost on study participants, who often tried to press harder to bring an object under their control. Similarly, our users would sometimes apply multiple fingers to an object when they wanted precise movement. Because of the inevitable imprecision of the simulation, the object would move too unpredictably for very fine control. In many of these cases, it would have been better to use a light (small) touch rather than a full grip.

Second, grasping a virtual object by placing contacts on either side can be difficult if not impossible in many of our techniques. Such manipulations require persistent resting contacts to be placed on virtual objects. The particle-based approach, in which each proxy is created every frame, places the proxy corresponding to a grasping finger on top of the object, thus defeating attempts to grasp it. The single proxy object approach uses persistent proxies, and so allows grasping of an object resting on the floor. It may be possible to extend the particle approach to allow proxies to persist at a given depth when it seems appropriate, or to explore a hybrid approach in which both the particle and single proxy techniques are used simultaneously.

Grasping may be difficult to replicate for more fundamental reasons. Virtual objects exert no counteracting force against the fingers, so it is difficult to know how “hard” the fingers are pressing on an object. Grasping an object in order to lift it out of the plane may be challenging to depict directly on an interactive surface with a 2D display. Similarly, the sensation of moving the hand across an interactive display surface is the same regardless of the simulated material, and whether the contact exerts static or kinetic friction.

Some of these problems may be addressed by improving the basic sensing techniques. For example, our system may be able to sense pressure by interpreting the gray levels of the input image as light touches or heavy touches. For proxy-based techniques, this could be implemented by changing the mass of the proxy. New range-sensing cameras [26] providing per-pixel depth information may also be appropriate. Per-pixel depth might be used to construct a rich 3D model of the hand, opening new opportunities in modeling grasping behavior. It might also assist in grasping an object in order to lift it up and place it on another object.

Clearly one need not completely replicate the physics of object manipulation in order to construct useful applications exhibiting physics-inspired behavior. The appropriate degree to which the techniques in this paper are applied depends on the application. A game might naturally exploit detailed physics throughout, while a graphical layout application might be very selective. Joint constraints provided by physics engines may be used to constrain motion, for example, to ease alignment tasks. While joints can be used to simulate the real-world counterparts of traditional GUI sliders, dials, buttons, and the like (as suggested by [10]), some aspects of traditional interactions do not naturally lend themselves to a physics implementation. Changing the size of an object dynamically, for example, does not lend itself to rigid-body simulation.

CONCLUSION

We have introduced a number of techniques to incorporate interactive surface input primitives into a real-time physics simulation. Our techniques take advantage of the fidelity of sensing provided by vision-based interactive surfaces, with the goal of enabling in a virtual domain the range of object manipulation strategies available to us in the real world.

ACKNOWLEDGMENTS

We thank Nic Villar for producing the accompanying video.

REFERENCES

1. Agarawala, A. and Balakrishnan, R. 2006. Keepin' it real: pushing the desktop metaphor with physics, piles and the pen. *CHI 2006*, 1283-1292.
2. Barron, J., Fleet, D., Beauchemin, S., and Burkitt, T. 1992. Performance of optical flow techniques. *Computer Vision and Pattern Recognition*, 236-242.
3. D. Baraff, 1989. Analytical methods for dynamic simulation of non-penetrating rigid bodies, *SIGGRAPH Computer Graphics*, 223-232.
4. D. Baraff. 1994. Fast contact force computation for nonpenetrating rigid bodies. *SIGGRAPH Computer Graphics*, 23-34.
5. Chang, B.-W., and Unger, D. 1993. Animation: from cartoons to the user interface. *UIST ’93*, 45-55.
6. Chang, F., Chen, C.-J., and Lu, C.-J. 2004. A linear-time component labeling algorithm using contour tracing technique, *Computer Vision and Image Understanding*, vol. 93, no. 2, 206-220.
7. Dietz, P. and Leigh, D. 2001. DiamondTouch: a multi-user touch technology. *UIST 2001*, 219-226.
8. Dragicevic, P. 2004. Combining crossing-based and paper-based interaction paradigms for dragging and dropping between overlapping windows. *UIST 2004*, 193-196.
9. Erleben, K., Sporring, J., Henriksen, K., and Dohlman, K. 2005. *Physics-Based Animation*. Charles River Media, Inc.
10. Fröhlich, B., Tramberend, H., Beers, A., Agrawala, M., and Baraff, D. 2000. Physically-based manipulation on the responsive workbench. *IEEE VR Conference 2000*, 5-11.
11. Gonzalez, R., and Woods, R. 2007. *Digital Image Processing: Third Edition*. Prentice Hall.
12. Geißler, J. 1998. Shuffle, throw or take it! working efficiently with an interactive wall. *CHI Ext. Abstracts,* 265-266.
13. Grossman, T., and Wigdor, D. 2007. Going deeper: a taxonomy of 3D on the tabletop. *Second IEEE Internat'ional Workshop on Horizontal Interactive Human-Computer Systems*, 137-144.
14. Han, J.Y. 2005. Low-cost multi-touch sensing through frustrated total internal reflection. *UIST 2005*, 115-118.
15. Hancock, M., Carpendale, S., and Cockburn, A. 2007. Shallow-depth 3d interaction: design and evaluation of one-, two- and three-touch techniques. *CHI 2007,* 1147-1156.
16. Hancock, M., Carpendale, S., Supporting Multiple Off-Axis Viewpoints at a Tabletop Display. 2007. *Second IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, 171-178.
17. Kruger, R., Carpendale, S., Scott, S., Tang, A. 2005. Fluid integration of rotation and translation, *CHI 2005*, 601-610.
18. Liu, J., Pinelle, D., Sallam, S., Subramanian, S., and Gutwin, C. 2006. TNT: improved rotation and translation on digital tables. *Graphics Interface 2006*, 25-32.
19. Mander, R., Salomon, G., and Wong, Y.Y. 1992 A ‘pile’ metaphor for supporting casual organization. *CHI ’92*, 627-634.
20. Microsoft Corporation. Microsoft Surface. http://ww.surface.com. 2007.
21. NVIDIA Corporation. NVIDIA PhysX. http://www.nvidia.com/object/nvidia_physx.html. 2008.
22. Reetz, A., Gutwin, C., Stach, T., Nacenta, M., and Subramanian, S. 2006. Superflick: a natural and efficient technique for long-distance object placement on digital tables. *Graphics Interface 2006*. 163-170.
23. Robertson, G., Czerwinski, M., Larson, K., Robbins, D., Thiel, D., and van Dantzich, M. 1998. Data mountain: using spatial memory for document management. *UIST ’98*, 153-162.
24. Ståhl, O., Wallberg, A., Söderberg, J., Humble, J., Fahlén, L. E., Bullock, A., and Lundberg, J. 2002. Information exploration using the pond. In *Proc. CVE*, 72–79.
25. Wilson, A. 2005. PlayAnywhere: a compact interactive tabletop projection-vision system. *UIST 2005*, 83-92.
26. Wilson, A. 2007. Depth-sensing video cameras for 3D Tangible Interaction. *Second IEEE International Workshop on Horizontal Interactive Human-Computer Systems,* 201-204.
27. Wu, M, and Balakrishnan, R. 2003. Multi-finger and whole hand gesture interaction techniques for multi-user tabletop displays. *UIST 2003*. 193-202.