Physics based Tabletop interaction

Rasmus Höfer[1]

[1] Aalborg University Copenhagen, Department of Medialogy, Lautrupvang 15, 2750 Ballerup, Denmark
rhofer09@student.aau.dk

Abstract. In this paper a new technique for representing tabletop input within a game physics engine is explored. The technique allows users to manipulate digital objects in ways analogue to the manipulation of real objects. Vision-based input is represented directly within the games physics engine to utilize its capabilities of dynamic physical behavior. The technique is able to sense arbitrary input information, allowing manipulation by fingers, hands, arms or physical objects. A user study of the different aspects involved in the technique showed that more work is required to further enhance and explore its capabilities.

Keywords: Interactive surfaces, tabletops, graphical user interfaces, game physics engines.

1 Introduction

Interactive surface technologies are displays based on the concept of combining input with output, surfaces that sense and emits at the same time. In difference to the traditional graphical interfaces, where input is *indirect* and mediated through a pointing device (mouse), the concept affords *direct* input on the display surface. This form of direct input allows users to interact with the digital content by direct touch and manipulation, adding a tangible quality to the interaction [1]. Designers of interactive surface applications often base interactions on this tangible quality, designing virtual objects to exhibit a real-world or "*pseudo-physical"* behavior. The desired effect with this kind of real-world analogy or *metaphor* is, inter alia, to make the experience with the interface seem more natural and easier to grasp. By this interactive surfaces provide users with a familiar language, building on the conceptual understanding of objects behavior known from everyday life [2, 3].

The implementation of pseudo-physical behavior normally requires designers to define specific interaction logics or gestures based on the input of interactive surfaces [4]. Common examples (e.g. [23][24]) include the rotation and translation of objects, where users touch and manipulate an object analogue to e.g. moving a sheet of paper on a flat surface with one or more fingers [5]. In order to support diverse input devices (capacitive and light sensors), many applications treat the touch input from fingers as single discrete contact points. However, the vision based input technologies are often capable of sensing sophisticated input information, levering a thorough representation of the input (e.g. an image). In recent developments [6-8] this vision

based representation is used to further explore the pseudo-physical behavior on interactive surfaces; contours or contact shapes of touch input are used to preserve the full expressiveness of the input, allowing interactions by using the whole hand or even tangible objects on the surface.

In the work of Wilson et al. [7, 8] the input is further explored by using it within a games physics engine. The use of a physics engine which simulates Newtonian physics enables interaction with digital objects by modeling quantities such as force, mass, velocity and friction [7]. The aim of using the surface input within this simulation is to create a more realistic representation of the real-world behavior, without the need of defining specific interaction logics or pre-programming the behavior of objects; hiding the complexity normally required to program multi-touch behavior and handling it via the physics engine.

This paper further explores this intersection of interactive surfaces and physics by representing surface input within a physics simulation. The approach presented in this paper represent camera input from a tabletop as 3D meshes within the game engine, in order to allow users to manipulate digital objects in ways analogue to the manipulation of real objects. The aim is to include the depth of objects above the tabletop surface to further add to the realism afforded by such an approach. In this way a 2D-sensor (camera) is used to infer a 3D representation of the 2D input. Furthermore a requirement for the approach is to allow arbitrary input, allowing the manipulation of content by using fingers, hands, arms and physical objects.

2 Related Work

Over the past years the interest in interactive surfaces has grown steadily. With the release of commercial products like the Microsoft *Surface* or the Apple *iPhone*, much research has been focused on multi-touch and interactive surfaces. However, the ideas of direct input and touch have been around for almost 40 years [9]. Through advances in low-cost display and input technology a wide range of different form factors have been developed, such as horizontal tabletops, vertical displays or wall projections, hand-held devices or other more flexible setups.

The input provided by these technologies has in most research scenarios been limited to an abstraction of input to discrete input points, typically simulating pointing with mouse cursors. Many of these interactive surfaces use cameras for input, which provide rich information beyond single touch points. In ShapeTouch [6], an early abstraction to single points is avoided and the camera input is used to create a contact shape and motion field. This is used to explore physical based interactions, allowing users to use their whole hands as well as object to interact with the digital content. ShapeTouch infers a virtual contact force from the contact shape and motion field which provides pressing, colliding and friction force [6]. The pressing force is inferred by the amount of contact area upon the object, while friction is a combination of contact area and motion force. The colliding force in ShapeTouch is determined by the boundary of the contact shape. This use of virtual force is still an abstraction of real world physical behavior, as the downward force or pressure is determined by the area of contact. Also the implementation requires predefined interaction logics and

object behavior, which in turn makes it harder to implement for application developers.

The use of realistic dynamics simulation in computer graphics and animation have recently become feasible in real-time, through the development of sophisticated physics engines (e.g. PhysX [14], Havok [20], Newton [21] and ODE [22]). These can handle real-time collision detection and physical response as well as friction and more [10]. Although modern computer games and 3D modeling software often employ physics engines the use of such engines in the interface and interaction communities is sparse.

The work of Wilson et al. [7] focuses on the use of modeling vision input into a physics engine. The input is treated in two different ways, either as a pointer approach from calculating connected components and using the center of the components as touch points, or as whole contour information based on a sobel image [16]. In order to interact with virtual objects this input has to be represented in the physics engine. Wilson et al. [7] presents and discusses five different strategies for representing such input, *direct force, virtual joints and springs, proxy objects, particles and deformable 2D/3D meshes*. The *particle proxy* technique (many small rigid bodies) presented is highlighted as the main contribution of their work, as it retains most of the benefits of mesh-based representations resulting in a high fidelity of interaction, while being easier for application developers to implement, than the other strategies. This technique is further developed in [8] to an implementation that also allows grasping behavior. Altogether these projects use shape and contour information which allows direct interaction by direct contact with the interactive surface. In contrast interactive surfaces which also support interactions above the surface, has been investigated by Hilliges et al. [11] and Wilson [12]. Range sensing cameras are used to directly measure the depth of an image, and to measure the 3D finger position to enable interactions in mid-air above or even between surfaces. While the use of range sensing hardware poses interesting opportunities, the focus of this work is on enabling interactions on and in the close proximity above the surface by using a regular camera setup.

2.1 Multi-Touch tabletop input

A number of hardware setup techniques have been developed for interactive surfaces. Depending on the hardware setup the input will vary. In the case of vision based surfaces there are typically a few techniques which are developed to give a consistent image of touch points on the surface. One such technique is the FTIR (frustrated total internal reflection) made popular by J. Han [13]. This technique uses infrared light mounted on the sides of a transparent surface (typically acrylic) where touching the surface will cause the infrared light being refracted from the points of touch (typical fingertips). Additionally a diffused surface on top or below is used to project the output image upon, as well as a compliant surface to enhance the contact points with the surface and allow scattering of light. A few other techniques similar to FTIR are DI (diffused illumination), which have the infrared light shining directly from below or top of the surface; DSI (diffused surface illumination), uses similar setup as FTIR

but with a special acrylic surface (endlighten) that have the ability to let the light reflect out of the surface. Depending on the setup and calibration, the DI and DSI techniques illuminate above the surface, together with the diffuser used for projection resulting in the possibility to capture hands or objects in a limited range (setup dependant, typically ~1-15cm) close to the surface (Fig. 1).

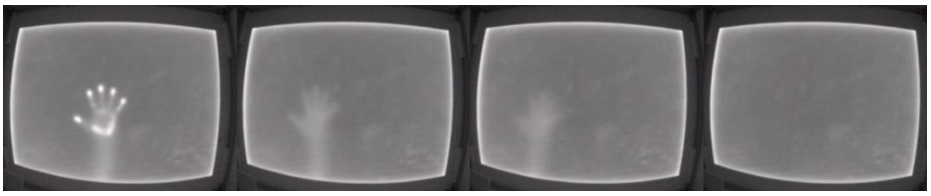


Fig. 1. Illustrates the raw input from of a flat hand and arm on top of a DSI setup with respectively direct touch on the surface, 2cm and 8cm range above the surface and input without touch (left to right).

The simplest representation of this input is as discrete points of surface contacts. Usually the pixels from camera input are grouped into connected components [16], and the center of each component (*blob*) is used as the point for interaction. This reduction of information can lead to problems, such as correspondence and it is preventing the full spectrum of interactions found in everyday life, encouraging the exclusive use of fingers for manipulation [7]. More realistic representation includes computing each contacts contour, as used in [6-8]. Using this contour information within a physics engine allows for more flexible and realistic manipulation possibilities as when only using single contact points. Besides contour information the input contains changing pixel information depending on the distance from the surface, which can be useful for creating a 3D representation of the input, as well as inferring limited pressure on the surface. It is the intention of this work to include this information within the physics simulation, to enable a more nuanced representation of the input, as when using contours only.

3 Strategies for manipulating within physics simulations

Modern physics engines simulate the mechanics and behavior of real-world physics while hiding much of the computational complexity. In this work Nvidia's PhysX [15] engine is used to handle the behavior between virtual objects. The engine makes use of physics concepts such as velocity, acceleration, momentum, forces, rotational motion, energy, friction, impulse, collisions, constraints and more [14]. Besides the realistic simulation of contact forces the engine also includes more advanced features such as simulation of soft bodies, cloth and fluids. To fully utilize these features the input from the interactive surface must be represented within the simulation. This representation grows in complexity together with the different complexity levels of the input. Thus, the simplest representation of input as single contact points can be represented within the physics world as simple primitive objects or virtual joints, whereas the more complex shape information needs a more advanced representation.

This increasing complexity of input allows more coherent and natural interactions, while it limits the choice of input technology and form factor.

The simple *pointer based input* has so far been represented within a physics simulation by calculating the direct force resulting from input movement (the most common way), or by using virtual joints. Wilson et al. [7] introduced the use of *proxy objects,* which represents the input within the physics simulation by rigid bodies of various forms (Fig. 2. left). For the more advanced contour input Wilson et al. used *particle proxies* - many rigid bodies either stacked [8] or ray cast [7] into the scene lying on the inputs contour shape (Fig. 2. right). These strategies have been investigated and the particle proxy technique found to be useful for more advanced interactions including the grasping of objects.

In contrast the representation by constructing 3D meshes of the input has not been investigated, which leaves the key question of this work: How can the interactive surface input be used for constructing 3D meshes enabling interaction with a physics simulation? In this paper the experiences with implementing and evaluating a mesh-based approach are described.

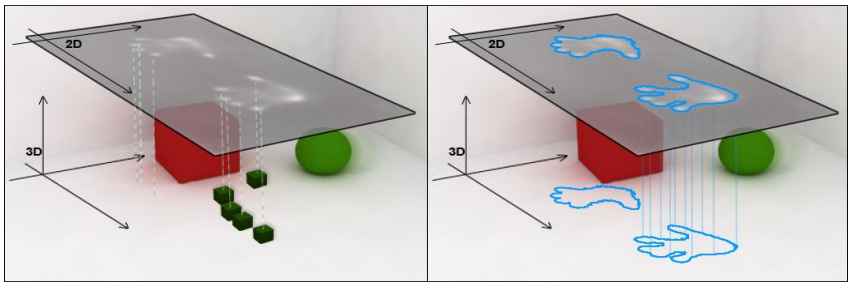


Fig. 2. Left image shows concept of single contacts to rigid bodies mapping, right image shows concept of using contour information by ray casting into the scene.

4 System implementation

The implementation proposed by this work uses input from a diffuse surface illuminated (DSI) tabletop, as seen in Fig. 1. Several image processing steps are applied to this input in order to use it within the 3D physics game engine. The game engine used is the Unity3D engine [15] which includes an easy to use API for the creation of interactive 3D content, allowing fast prototyping. It uses the PhysX engine [14] for physics simulation.

The aim of the implementation is to generate a 3D representation of arbitrary input within the game engine which should be able to interact with the physics engine in real time, consequently allowing a wide variety of manipulation styles. Given the requirements of real time capability, allowing arbitrary input (e.g. objects of any kind besides hands or fingers on the surface) and the use of a single camera, the strategy used will be simplistic without tracking or recognizing 3D features. In short the strategy followed in the implementation is to first apply image segmentation, then all pixels lying inside each segments contour, are normalized and used as a kind of

displacement map for creating different mesh representations. A detailed discussion follows.

4.1 Image processing

The strategy for the image segmentation used in this implementation is to obtain the contour through connected components. One of the problems with connected components is that regions may be subdivided, due to holes, into smaller components although belonging together. Given the stable condition of the hardware setup this strategy may be adequate for segmentation although a range of other strategies may give a more stable segmentation, e.g. snakes boundary detection [16].

Prior to the image segmentation a few steps are applied to correct for lens distortion, level normalization and positioning of the input image [16]. The image segmentation is based on subtracting a captured background frame from the current frame. As the camera and lighting conditions are almost stable given by the closed box of the tabletop DSI system, a robust image of the foreground is obtained without the need of more advanced background subtraction techniques. In order to eliminate camera noise and allow calibration of the remaining foreground a blur and a high-pass filter is applied to the image. Normally the attributes of the blur and high-pass filter are set to further remove features of the foreground, so that only direct contacts (containing a high pixel intensity) with the surface are remaining, as shown in Fig. 3 (lower middle). In this approach the filters are set to only remove camera noise, in order to leave foreground regions almost intact, as show in Fig. 3 (top middle). After this the image is threshold to binary and segmented into connected components.

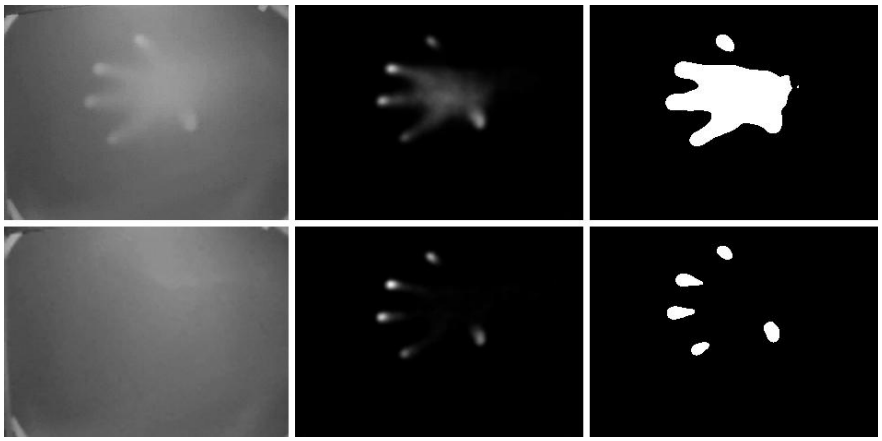


Fig. 3. (top left) raw input (lower left) captured background (top middle) high-pass and background removed as used in this approach (lower middle) high-pass and background removed as in regular scenarios (top and lower right) resulting segmented binary image.

Based on the contour of the segmented regions the boundaries are used to define a uniform grid for which the pixels lying inside the contour and on the grid points (referred to as control points) are used for creating the mesh. A grid is used in order to simplify the data used for mesh creation and more importantly to allow for changing the resolution of the mesh by adjusting the density of the grid. Consequently changing the grid to a very high density will take every pixel lying inside the contour for mesh creation while a low density will only take a few points inside the contour (see Fig. 4). This variable grid is used as control points for creating the 3D mesh, which is explained in detail later. Given this grid the pixels intensity is used to define the control points depth or (z-value). With a high density grid each control point corresponds subsequently with a pixel intensity value. When using a lower resolution the average pixel intensity of the surrounding pixels (based on the spacing of each grid cell) is used to define the control point's depth value.

Finally the data obtained through this approach is transferred to the game engine in order to create a 3D mesh, which is used as a rigid body for interacting with the physics engine.

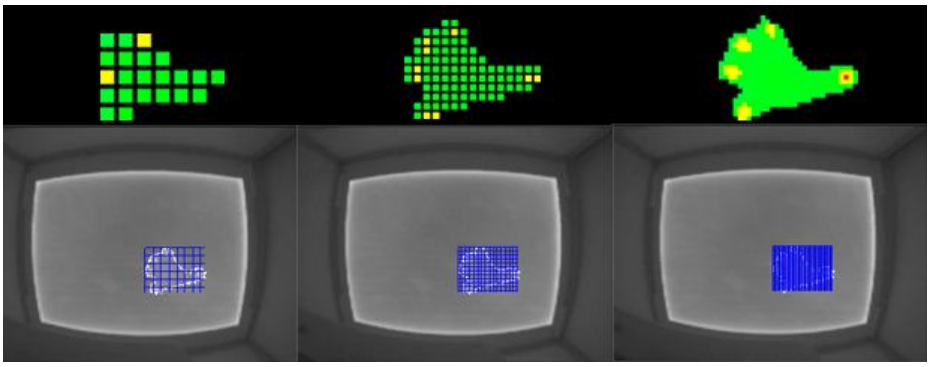


Fig. 4. Color coded depth maps (control points) resulting from three different grid densities (top) with corresponding input images (bottom) and overlaid grids.

4.2 Mesh creation and physics engine interaction

The aim with creating a mesh of the data obtained from image processing is to use it as representation of the surface input for interacting with the physics engine. For this aim the best strategy of using the data needs to be explored. Three different ways are investigated: a *particle* based approach, a *mesh* based approach and a *b-spline* based approach. Each of these approaches has different effects on the behavior of the physics engine, particularly the *collision* contact model, and may require more or less calculation time.

The particle based approach uses the data to create many small rigid bodies on the grid points obtained from image processing; it resembles the strategy of particle proxies as used in [7], with the difference that particles are not only created on the contour but on all control points. The depth value of control points is used to define

each particles mass, while each particle is placed in the scene by finding the nearest point of intersection with either virtual objects or the scene ground by ray casting (see Fig. 5 left for an example). Using different shaped particles, such as cubes and spheres, together with the varying mass of the particles give different collision responses in the physics engine. This approach is simpler than creating a full mesh of the data, but as it uses many particles it may also slow down the simulation significantly, depending on the particle count.

The mesh based approach uses the grid to create a 3D mesh of the depth data. First the normalized data obtained from image processing is transformed to match the size of the 3D world. Then a mesh is created, where each control point is defining a vertex position. As the data comes in an ordered grid form, the triangulation and polygon creation is simple opposed to non-ordered data; 2 triangles are created for every 4 vertices defining a rectangular polygon. The mesh created in this way is used as a kinematic rigid body collider, which in the physics engine is able to influence other rigid bodies, by controlling its position. The physics engine calculates motion and collision forces based on the movement of the colliders. Updating the mesh created in this approach forces for each update to recalculate the collider, which internally needs to update all surface normals of the mesh. The tradeoff of using a low density grid opposed to a high density unfolds in calculation time and representation fidelity; a high density grid may take longer time to calculate but also gives a better representation of the input from the surface.

In the b-spline based approach the grid is used as control points for defining a nonrational basis-spline (short b-spline) surface [17]. The aim of using a b-spline surface is to further enhance the input representations fidelity and accuracy. A b-spline surface is defined by the following equation (1).

$$\mathrm{p}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} \mathrm{N_{i,p}}(\mathrm{u})\,\mathrm{N_{j,q}}(\mathrm{v})\mathrm{p_{i,j}}\,. \qquad \textbf{(1)}$$

In the equation $\mathrm{N_{i,\,p}}(\mathrm{u})$ and $\mathrm{N_{j,\,q}}(\mathrm{v})$ are b-spline basis functions of degree p and q, which product serve as coefficient for the control point $\mathrm{p_{i,\,j}}$. Using the data obtained from image processing as input for defining the control points of the b-spline surface, gives a much smoother and natural representation of the surface input. The smoothness of this representation can be modified by adjusting the resolution of the output grid $\mathrm{p}(u, v)$ and by setting the degrees of the respective basis functions (see Appendix and Fig. 5. for some results). Using de Boor's algorithm [18] for calculating the basis function gives $\mathrm{O}(\mathrm{n}^2)$ operations, depending on the degree (n) for each basis function. The advantage of using a b-spline surface for input representation is the smoother and more natural representation. It has to be evaluated if this representation is feasible for real-time purposes, as the surface normals together with the b-spline surface needs to be recalculating for each update similarly as described in the mesh strategy.

The following summarizes the implementation algorithm, where the different steps of image processing and mesh creation are taking place for each frame. Note that the meshes are scaled to match the size of the virtual scene, where the maximum depth

value of the control points correspond to the scenes ground plane, in difference to the ray cast determined vertical position of the particles:

Compute connected components from tabletop input
For each connected components contour
    Set grid of size *n* * *m* with cell spacing *f*
    Obtain pixel intensity of the grids control points
    Create B-spline Mesh, Mesh based on control points*
    Mesh is scaled to match the size of the virtual scenario*
    Collider and rigid body are updated based on Mesh*
      *particles are spawned for all control points as rigid bodies instead
    Rigid body is moved (based on contours center) and forces updated
Physics simulation is updated
All rigid bodies are destroyed

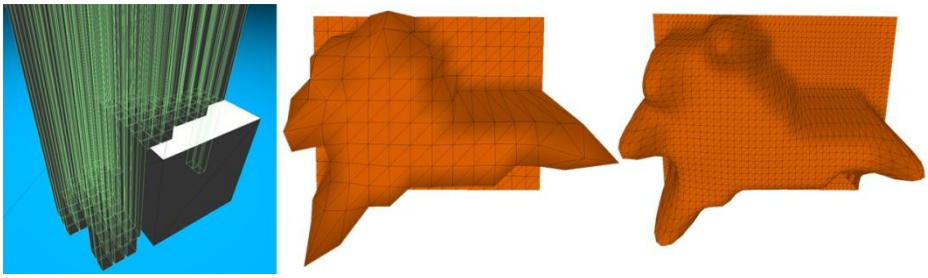


Fig. 5. (left) Cube shaped rigid body particles with different mass ray cast on scenes ground plane and cube (middle) Mesh generated of control points (right) B-spline mesh generated of control points. All representations are created of the tabletop input as seen in figure 4 middle.

5 Initial evaluation

To better understand how people interact with and use the proposed system, an experimental qualitative study was conducted. The focus of this study was to evaluate the speed, accuracy and quality of the three different strategies (particle, mesh and b-spline) used in the implementation. Key questions for this study addressed the usability of the implementation, such as: Is the interaction afforded by the implementation understandable and usable? Is it of sufficient accuracy? Is it predictable? Can users without knowledge of multi-touch discover the usage?

In this study 8 participants (5 male 3 female: 21-29 years age) were exposed to 3 simple tasks (Fig. 6) involving different aspects of the interaction possibilities afforded by the implementation. The participants had different engineering backgrounds and all had normal or corrected to normal vision. Prior to the tasks all participants were asked to evaluate their personal experiences with touch devices, were 2 of the participants had no prior experience, 4 had experience with multi-touch enabled smart phones, and 2 had experience with tabletops. The study was conducted using a DSI-tabletop system with the following specifications in order from top to bottom: An array of 150 infrared leds (850~nm) mounted on the side of a light emissive

transparent acrylic sheet (PLEXIGLAS EndLighten Clear 0N002 XL) for illuminating objects on top of the surface, a sheet of translucent acrylic (PLEXIGLAS RP Grey 7D006) for rear-projection of the output image, a web camera (Playstation 3) with infrared band pass filter (850~nm) for image acquisition and a short throw projector for projecting the output image. The system was tested using one computer with 2.8GHz Intel i7 CPU, 4 GB RAM and a ATI Radeon HD 5800 graphics card. The image processing part was programmed in C++ using openFrameworks and openCV. The data obtained was sent to Unity v3 by UDP using the OSC protocol.

The intended study was based on a 3x2x2 repeated measures within-subjects design, which was changed to a 3x2 design caused by issues with the implementation (see next section). All participants worked through the 3 tasks under 2 conditions, one with added feedback and one without feedback. This feedback was displayed as the actual contour of the tabletop input, providing users indications of how they can interact with the virtual contents. All participants were first exposed to a practice task, were they had as much time as they liked for getting used to and explore the interaction style on their own. No introduction of how to properly use the system was given. After this all participants worked through the 3 tasks, first without feedback (including practice), followed by the same 3 tasks with feedback. For all participants the time to complete each task was measured.

The three tasks participants worked through were a positioning, a sorting and a navigation task. In the positioning task participants were asked to move 4 objects with 2 different shapes (sphere and cross) into their designated positions. In the sorting task participants were asked to sort 16 objects (8 spheres and 8 cubes) by their two different colors. This involved moving 8 objects with orange color and 8 objects with blue color to their corresponding sides (see Fig. 6 middle). In the navigation task participants were asked to move one object along a path, were moving the object out of the paths boundary would result in resetting the objects position to the start of the path. All participants were presented to the tasks in the same order, positioning, sorting and navigation. During these tasks observations were noted regarding the participants style of interaction, particularly it was observed if participants used one finger, more fingers or the whole hand, bimanual or single handed.

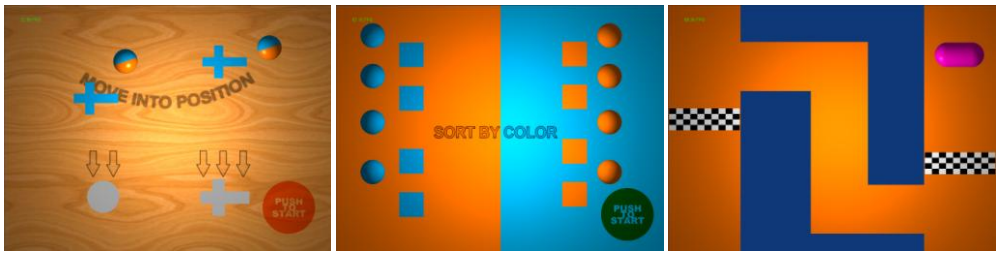


Fig. 6. Showing the three tasks (left) positioning task (middle) sorting task (right) navigation task

After completing all tasks in both conditions (with and without feedback) participants were interviewed informally using a variation of the Microsoft Desirability Toolkit [19]. This interview was started by letting participants choose 3 words from a list of

total 86 words (see Appendix for list), with the instruction of selecting the 3 words that best described their experience with using the system and completing the tasks. For each word chosen the participants were interviewed for the reasons of choosing that word. For example if a participant chose the word “confusing” he/she was asked what their motive and reason was for choosing that specific word.

5.1 Early issues

The intention with the study was to test all three strategies, particles, mesh and b-spline mesh. This could however not be achieved, as both the mesh and the b-spline strategy showed too much instability for being used in a study. This instability is caused by several reason, which are described and possible improvements given in the discussion section. For the study only the particle strategy was tested, as it showed less instability.

5.2 Results

The quantitative results obtained from measuring task completion times were compared using a paired t-test (repeated measures) for conditions with and without using feedback. Mean completion times for each task and each condition can be seen in Fig. 7. All three tasks showed significant difference ($p < .01$) between the two conditions. Another statistical analysis was made to compare users who solved the tasks bimanually, using both hands, with users who only used single hands. The use of hands was noted during all tasks, were 4 of the participants mainly used one hand and the other 4 mainly used both hands. In the positioning and sorting task without feedback a significant difference ($p < .01$) was found between the use of one and two hands. The navigation task showed no difference together with all tasks including feedback.

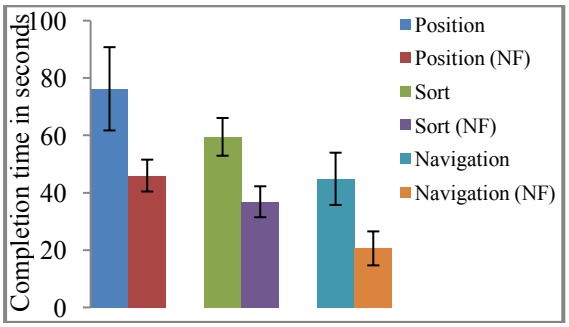


Fig. 7. Mean completion times for positioning, sorting and navigation task. Paired columns showing tasks with feedback left and without feedback right. (NF = no feedback)

Though the qualitative results from interviewing participants are not quantifiable a picture of the most frequently chosen words can be given. This is displayed as a word cloud in Fig. 8. The words chosen by participants were counted and the word cloud displays the frequency as size and intensity, where the more often a word was chosen

the size of the displayed word was increased along with an increased intensity in grey level. Since these words only give meaning with an explanation of why they have been chosen, a short sum-up of the most frequent words follows. The word "fun" was chosen 6 times. Most participants commented that they thought it was fun to use the prototype and it was different with how they normally operate interfaces.

Fig. 8. Word cloud of the words chosen by participants using the desirability toolkit. Larger and darker words were chosen more often than smaller lighter words.

The word "fast" was chosen 4 times. All participants related this word to the response of the virtual objects, which in their opinion moved too fast when touching. Some comments were given such as: "you have to be really careful and have a steady hand.", or "the object moved to fast when touching it." The word "unpredictable" was chosen 3 times. Participants related this word to the behavior of virtual objects when touching them, where most participants commented that they did not feel in control of the objects. This was also observed with some participants who had problems solving the tasks and manipulating the virtual objects. Other words such as "frustrating" and "inconsistent" were chosen for similar reasons. A few participants commented this unpredictability as: "once I got used to it, which takes a little practice, I can use it."

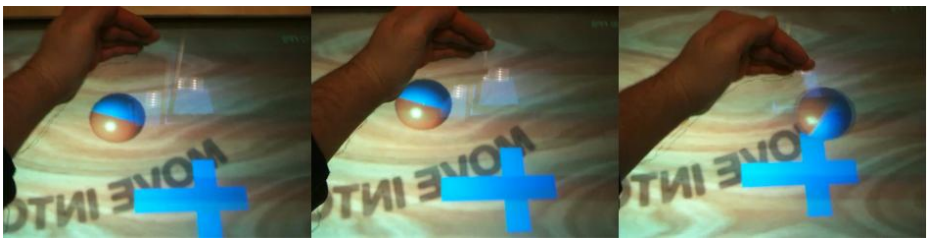


Fig. 9. Example of moving an object by using the arm.

Altogether the interview and the observations showed indications of most users having trouble with using the system for accurate control, with a few exceptions. However, the observations also showed that many participants used manipulation strategies, which in a regular kinematic controlled tabletop system (where objects are moved directly upon touch and "stick" to the fingers) are not possible. Manipulation styles observed included: moving many objects simultaneously by using the whole hand or arm (Fig. 9.), moving an object with one hand and stopping it with the other, pushing objects from the side, turning and rotating objects by pushing from sides,

controlling the speed of objects by pushing from side with one finger and holding on top with another. Another manipulation style observed was that many participants tried to move objects by directly touching them from top, which worked for spherical objects, but not for cubical.

Possible reasons for the lack of control and suggestions for enhancing all implementation strategies are given in the next section.

6 Discussion

The results of the study showed that the physical behavior and the used particle strategy have to be revised. Although all participants solved all three tasks, the intended easy of controlling objects with the implemented strategy was not present. The reasons of this can mostly be found in the implementation, particularly the image processing needs to be discussed and improved together with the physics simulation. The study also showed that feedback needs to be provided for easier comprehension of the system. Though most users found it hard to accurately control objects, many found the physical manipulation styles fun to use and the observations showed that users were able to discover and utilize new manipulation styles on their own. In the following improvements to the implementation are discussed, with regards to all three strategies (particle, mesh and b-spline).

Image processing

The deficits of allowing accurate control and manipulation of objects can partly be found in the image segmentation and contour retrieval routine. The contour obtained from using connected components was unstable, particularly when using a larger contact area as input, such as the whole hand or arm (Fig. 10. shows three examples of inaccurate contours). This instability appears through a flickering contour, which is caused by pixels on the contour changing and falling in or out of the binary threshold. As the grid used for all methods is based on this contour it results in a change of the actual representation (particles, mesh and b-spline) in the game engine. This in turn can result in a collision of the representation with a virtual object in the physics engine, thus moving the object without actually moving a finger or hand. The effects are less visible in the used particle strategy, as the rigid bodies created on the grid points are ray cast into the scene on every frame, thus not causing strong collision forces when not moving the finger or hand. Possible solutions to these problems could be to utilize either a different contour routine (e.g. snakes boundary) or smooth the contour with a variable time and motion factor. Such time factor could for example be to only update the contour when the input is moving more than a threshold, or to interpolate the data over time. Apart from the changing contour the pixel values for determining the control point's depth data need also to be stabilized. This is necessary for the mesh and b-spline strategy, as the depth data is used to create the meshes and therefore stability is required for not causing unwanted collisions. Time interpolation may also be used to stabilize the depth data. Altogether the instability of the image

processing is problematic when using a model-free approach. This also shows the limitations of such an approach, were a more stable way could for example be to track user's hands by pose estimation or skeleton tracking.
Another important observation was that the depth data obtained from mapping pixel values could not cover the full range in top of the tabletops surface, meaning that when holding a hand in top of the surface the pixel values flatten out with added height. Therefore the depth map was only accurate in close contact with the surface, whereas higher contacts resulted in an inaccurate representation. Solutions for better depth sensing would either be to change the illumination method of the tabletop, to e.g. include different illumination layers in top of the tabletop (which require major hardware change) or to switch to a depth sensing camera, which was not intended in this work.

Fig. 10. Three images showing the contour as a line for feedback. Instability and inaccuracy can result in unwanted collision forces.

Physics simulation and game engine

The models created by the three different strategies (particles, mesh and b-spline) had different effects on calculation times (FPS) and the physics simulation.

Experiments with the mesh created by the b-spline strategy showed that the mesh is fairly accurate and matches the input, depending on the density of the grid provided by image processing (see Appendix for an overview). The effect of calculating this mesh results in a significant decrease of processing power available for other calculations, thus resulting in lower frame rates. With small input sizes, such as for example one finger tip, this still is applicable in real-time, whereas larger inputs results in to heavy calculations and in unacceptable drop in frame rates (often FPS < 3). In order to use this strategy in a real-time application much of the calculations could be "outsourced" to the GPU using parallel programming (e.g. CUDA). The mesh strategy can however create matching results, when using a high density input grid. Calculation times for the mesh strategy all showed to be easy doable by the CPU and frame rates never dropped below 30. Calculation times for the particle approach only showed to decrease frame rates when using large contact areas. This is partly caused by a large amount of objects (particles) created at runtime and ray cast calculations for each particle. All strategies used in this prototype implementation were not optimized for run-time and possible better frame rates should be achievable.

Another part of the deficits of allowing accurate control with the particle strategy is caused by the physics simulation and the way force calculations are applied. Using

the contours center to move the rigid bodies (particles), result in collision forces which only relate to this single point. Better control could be enabled when including motion forces for all control points in the grid, which in turn require a different image processing approach, such as e.g. optical flow [16].

9 Conclusion and future work

In this paper a new technique for representing tabletop input within a real-time game physics engine was introduced. Requirement for the technique was to be input independent, allowing any kind of objects besides user's hands to be used as input. The technique utilized three different strategies for representing camera based tabletop input: a particle strategy, a mesh based and a b-spline based strategy.

The mesh strategy showed benefits in computation time, while other factors cannot be concluded at this time, since the strategy was too inaccurate for testing.

The b-spline based strategy showed to be computational heavy running on the CPU and not applicable for real-time, at least when using large input areas. Again other factors cannot be concluded since the strategy was not evaluated.

The particle strategy showed to require more accuracy, which could be obtained by changing or enhancing parts of the image processing routine and parts of the physics calculations. Users showed to like the physical based manipulation style and were able to discover and utilize manipulation strategies on their own.

Overall the strategies require more work in the image processing routine and in parts of the physics calculations to be reevaluated. Furthermore this work shows the difficulties of implementing an input independent system.

References

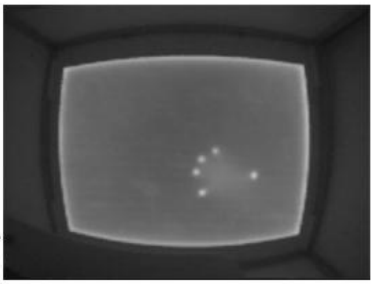
1. Terrenghi, L., Kirk, D., Sellen, A., Shahram, I.: Affordances for Manipulation of Physical versus Digital Media on Interactive Surfaces. In: Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '07), pp. 1157--1166, ACM, New York (2007)
2. Fishkin, P.K.: A taxonomy for and analysis of tangible interfaces. In: Personal and Ubiquitous Computing. vol. 8, issue 5, pp. 347--358, Springer, London (2004)
3. Fishkin, P.K., Moran, T.P., Harrison, B.L.: Embodied user interfaces: Towards invisible user interfaces. In: Proceedings of the IFIP TC2/TC13 WG2.7/WG13.4 Seventh Working Conference on Engineering for Human-Computer Interaction. pp. 1--18, Kluwer, B.V. (1999)
4. Wobbrock, J.O., Morris, M.R., Wilson, A.D.: User-defined gestures for surface computing. In: Proceedings of the 27th international conference on Human factors in computing systems (CHI '09). pp. 1083--1092, ACM, New York, NY, USA (2009)
5. Kruger, R., Carpendale, S., Scott, S.D., Tang, A.: Fluid integration of rotation and translation. In: Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '05). pp. 601--610, ACM, NY, USA (2005)
6. Cao, X., Wilson, A.D., Balakrishnan, R., Hinckley, K., Hudson, S.E.: ShapeTouch: Leveraging contact shape on interactive surfaces. In: IEEE International Workshop on

Horizontal Interactive Human Computer Systems (TABLETOP '08). pp. 129--136, IEEE Computer Society, Washington, DC, USA (2008)
7. Wilson, A.D., Izadi, S., Hilliges, O., Garcia-Mendoza, A., Kirk, D.: Bringing physics to the surface. In: Proceedings of the 21st annual ACM symposium on User interface software and technology (UIST '08). pp. 67--76, ACM, NY, USA (2008)
8. Wilson, A.D.: Simulating grasping behavior on an imaging interactive surface. In: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS '09), pp. 125--132, ACM, NY, USA (2009)
9. Buxton, B.: A Touching Story: A Personal Perspective on the History of Touch Interfaces Past and Future. In: Society for Information Display (SID) Symposium Digest of Technical Papers. vol. 41, issue 1, pp. 444--448, SID (2010)
10. Hasegawa, S., Sato, M.: Real-time Rigid Body Simulation for Haptic Interactions Based on Contact Volume of Polygonal Objects. In: Computer Graphics Forum. vol. 23, issue 3, pp. 529--538, Blackwell Publishing Inc. (2004)
11. Hilliges, O., Izadi, S., Wilson, A.D., Hodges, S., Garcia-Mendoza, A., Butz, A.: Interactions in the Air: Adding Further Depth to Interactive Tabletops. In: Proceedings of the 22nd annual ACM symposium on User interface software and technology (UIST '09). pp. 139--148, ACM, NY, USA (2009)
12. Wilson, A.D., Benko, H.: Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In: Proceedings of the 23nd annual ACM symposium on User interface software and technology (UIST '10). pp. 273--282, ACM, NY, USA (2010)
13. Han, J.Y.: Low-cost multi-touch sensing through frustrated total internal reflection. In: Proceedings of the 18th annual ACM symposium on User interface software and technology (UIST '05). pp. 115-118, ACM, NY, USA (2005)
14. Nvidia PhysX, http://developer.nvidia.com/object/physx_features.html (online)
15. Unity 3D, http://unity3d.com/unity/engine/physics (online)
16. Pratt, W.K.: Digital Image Processing: PIKS inside. (3rd ed.), Wiley-Interscience (2001)
17. Rogers, D.F.: An Introduction to NURBS: With historical perspective. Elsevier (2000)
18. Boor, C.: On calculating with B-Splines. In: Journal of Approximation Theory, vol. 6, issue 1, pp. 50--62, July (1972)
19. Benedek, J., Miner, T.: Measuring Desirability: New Methods for Evaluating Desirability in a Usability Lab setting. Redmond, WA.
(Word document): http://www.microsoft.com/usability/UEPostings/DesirabilityToolkit.doc
Developed by and © 2002 Microsoft Corporation. All rights reserved.
20. Havok Physics engine, http://www.havok.com/index.php?page=havok-physics (online)
21. Newton Physics engine, http://newtondynamics.com (online)
22. ODE Physics engine, http://www.ode.org (online)
23. Apted, T., Kay, J., Quigley, A.: Tabletop sharing of digital photographs for the elderly. In: Proceedings of the SIGCHI conference on Human Factors in computing systems (CHI '06). pp. 781-790, ACM, NY, USA (2006)
24. Hancock, M.S., Vernier, F.D., Wigdor, D., Carpendale, S., Shen, C.: Rotation and translation mechanisms for tabletop interaction. In: First IEEE international Workshop on Horizontal Interactive Human Computer Systems (TABLETOP '06). pp. 79--88, IEEE Computer Society, Washington, DC, USA (2006)

Appendix: Desirability toolkit wordlist (as used in the interview)

Accessible	Controllable	Fast	Poor quality	Understandable
Advanced	Convenient	Flexible	Powerful	Unpredictable
Ambiguous	Counter-intuitive	Frustrating	Predictable	Unrefined
Annoying	Creative	Fun	Relevant	Usable
Appealing	Cutting edge	Hard to Use	Reliable	Useful
Approachable	Desirable	High quality	Responsive	Vague
Attractive	Difficult	Illogical	Rigid	
Awkward	Distracting	Impressive	Satisfying	
Boring	Dull	Inadequate	Simple	
Bright	Easy to use	Incomprehensible	Simplistic	
Busy	Effective	Inconsistent	Slow	
Clean	Efficient	Ineffective	Sophisticated	
Clear	Effortless	Innovative	Stable	
Cluttered	Empowering	Insecure	Straightforward	
Compelling	Energetic	Intuitive	Stressful	
Complex	Engaging	Irrelevant	Time-consuming	
Comprehensive	Entertaining	Meaningful	Time-saving	
Confusing	Exciting	Misleading	Too technical	
Consistent	Expected	Motivating	Unattractive	
Contradictory	Familiar	Overwhelming	Unconventional	

Figure showing effect of different grid spacings using mesh and B-spline generation.



Input image
(Hand with sprawled fingers)