tBox: A 3D Transformation Widget designed for Touch-screens

Aurélie Cohé, Fabrice Decle, Martin Hachet

tBox: A 3D Transformation Widget designed for Touch-screens

Aurélie Cohé
INRIA Bordeaux
Université de Bordeaux - CNRS (LaBRI)
aurelie.cohe@inria.fr

Fabrice Dècle
INRIA Bordeaux
Université de Bordeaux - CNRS (LaBRI)
fabrice.decle@inria.fr

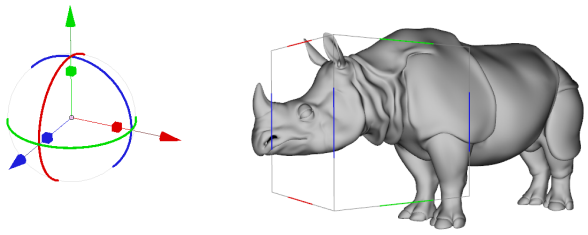
Martin Hachet
INRIA Bordeaux
Université de Bordeaux - CNRS (LaBRI)
martin.hachet@inria.fr

ABSTRACT
3D transformation widgets are commonly used in many 3D applications operated from mice and keyboards. These user interfaces allow independent control of translations, rotations, and scaling for manipulation of 3D objects. In this paper, we study how these widgets can be adapted to the tactile paradigm. We have explored an approach where users apply rotations by means of physically plausible gestures, and we have extended successful 2D tactile principles to the context of 3D interaction. These investigations led to the design of a new 3D transformation widget, tBox, that can been operated easily and efficiently from gestures on touch-screens.

Author Keywords
3D User Interface, 3D transformation widget, multi-touch

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: User Interfaces

INTRODUCTION
Since the introduction of *Skitters* and *Jacks* [1], 3D transformation widgets used in the manipulation of 3D objects have little evolved. These 3D user interfaces (UI) have been mainly designed for mouse-based systems where the user benefits from accurate pointing, distant interaction, an unobstructed view of the screen, and direct access to numerous buttons and keyboard shortcuts. Touch-screens have none of these qualities as noted by Moscovitch [6]. Consequently, 3D transformation widgets need to be reinvented to adapt to the tactile paradigm. An example of new 3D transformation widgets has been proposed by Schmidt et al. [8] where users invoke graphical control elements by using strokes. Others have explored multi-touch controls where several degrees-of-freedom (DOF) can be manipulated at the same time. In particular, Reisman et al. [7] have extended the well known Rotate-Scale-Translate (RST) multi-touch technique to 3D. Hancock et al. proposed techniques where users manipulate 3D objects with one, two, or three fingers in shallow depth [2], or gravity-based 3D environments [3]. Martinet et al. [5] have evaluated these interfaces for a 3D manipulation task, and they have proposed a technique based on the separation of the DOFs. Our approach is different, and complementary. We have designed a touch-based 3D transformation widget called tBox that favors the direct and independent control of 9 DOF; 3 translations, 3 rotations, and 3 scalings.



(a) (b)

Figure 1. A standard 3D transformation widget (a) and tBox (b) for the control of 9 DOF.

The design of this widget has been guided by initial observations of users interacting with standard 3D transformation widgets on touch-screens. The main conclusions of this preliminary study were that the selection of the DOF controls is difficult as soon as the graphical elements project close to each others on the screen. This is very frequent, in particular when all the DOF controls are displayed at the same time (Figure 1(a)). Moreover, we observed that users were sometimes perturbed by occlusion and ergonomic issues, and they had to think about how to position their hand. Finally, we noticed that it was difficult for them to apply successive fast and brief gestures, common in many tactile applications (e.g. mobile phone applications), and that multi-touch input was not exploited. Consequently, we have explored an alternative approach for the design of tBox, where precise and small mouse displacements are replaced by finger inputs that better fit the tactile paradigm. Such a touch-based approach may open 3D modeling to many users who are not 3D experts. For example, a user may want to create simple 3D shapes on his tablet (e.g. from 3D sketch-based techniques), and then arrange them together by way of tBox. This allows him to transform an idea he has in mind into a 3D scene. Our final goal is not to replace all the mouse/keyboard modeling tools by their touch-based counterparts. Instead, we believe that touch-screens represent a nice opportunity for enlarging the current expert desktop habits to new, broader usage and contexts (e.g. mobility and social communication).

Preliminary version

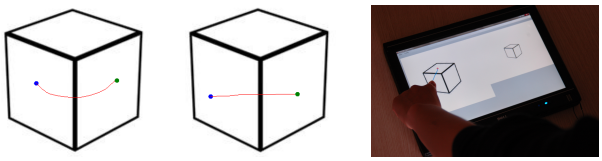
TBOX DESIGN

The tBox widget appears as a wireframe box, with face culling enabled (see Figure 1(b)). Rotations are performed from physically plausible gestures, translations rely on the selection of the box edges, and scaling benefits from dual-touch input. This design favors a direct access to all DOFs, while keeping a good separation of the actions to be applied. Similarly to standard 3D transformation widgets, tBox is always visible, the widget being displayed on top of rendered objects. The transformations applied to tBox are directly applied to the transformation matrix of the object being manipulated. The box-shaped form-factor has been chosen for two main reasons. First, it is an elementary 3D shape that provides good visual affordances for making it spin from "natural" gestures. Second, this form-factor enhances axes selection from finger inputs, as we will discuss later in this paper. Box-shaped widgets have already been used previously for standard desktop contexts (e.g. [4] and Maya's universal manipulator). In our approach, we have introduced new mechanisms aiming at enhancing interaction when users interact directly on touch-screens, from finger gestures.

Rotation

We have observed in our preliminary study that one of the main difficulty when using standard widgets on touch-screens came with the control of rotations. Thus, we initially focused our investigations on these DOFs. Because motions with physical behaviors appear to be compelling on tactile screens, we investigated an approach where users rotate the widget as they would do with a real cube. Our intuition was that the 2D gestures used to make a box spin around one of its primary axes can be characterized. We conducted a pilot study to investigate this assumption. We used a *guessability* study methodology, as proposed in [9]. The idea is to present the effects of gestures to participants, and elicits the causes meant to invoke them. The study was performed on a Dell Latitude XT Tablet PC. A video corresponding to a spinning cube and a static image of the same cube were displayed (see Figure 2(c)). The subjects were asked: *"Which gesture would you draw on the static cube to obtain the movements of the moving one?"*. In addition, we asked them to assess their gesture by answering the questions *"the gesture I did is a good match for its intended purpose"* (Q1) and *"the gesture I did is easy to perform"* (Q2), as done in [9]. We used two Likert scales with ordinal responses from $1 =$ *strongly disagree* to $7 =$ *strongly agree*. The subjects completed the task with thirty videos corresponding to the rotations in both directions around the three primary axes, for five viewpoints. Ten subjects aged from 22 to 50 (three women and seven men) participated in this experiment. Three were left-handed and seven were right-handed. We recruited subjects with no experience in 3D modeling in order to avoid gestures guided by previous habit.

Interestingly, we obtain very comparable gestures from one subject to another, and from one view to another. From these results, we made two main observations. First, when they exist, the inner edges appear as the visual references from which users draw their gestures. Consequently, these edges should have priority to the other edges in the rotation algorithm. Second, we observed that two strategies were used to make the cube spin. In some cases, subjects followed the 3D orientation of the touched face, as if they were *grabbing* the cube (Figure 2(a)). In other cases, they follow a direction that is tangent to the rotation, at the targeted edge. This corresponds to a *push* strategy where energy is minimized (Figure 2(b)). In both cases, the gestures rely on physically plausible behaviours, as we had expected. During the experiments, 3 subjects used the *push* strategy only, 3 used the *grab* strategy only, and 4 used both strategies. The mean scores about the pertinence and the easiness of the gestures applied by the subjects are very high (between 5.5 and 6.8), which tend to show that an approach based on such gestures could be easily understood and used.

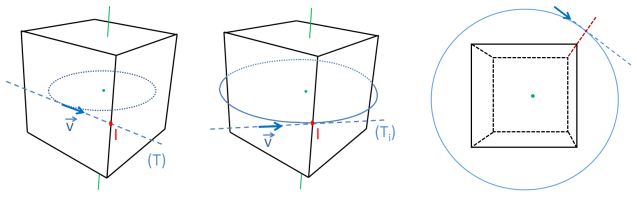


(a) (b) (c)

Figure 2. Example of physically plausible behaviour gestures: *grab* (a) and *push* (b). The setup of the experiment (c).

Rotation algorithm

From the results described above, we have designed the rotation algorithm as follow. As soon as a finger motion is detected (i.e. 50 mm finger displacement), we compute the finger's movement vector $\vec{v}$ in screen space. If the continuation of $\vec{v}$ intersects with the screen-space projection of an inner edge at point I, we see if $\vec{v}$'s 3D projection on the cube is tangent to the corresponding rotation (Figure 3(a)), or if $\vec{v}$ is aligned with the tangent line (T_I) at point I, projected in 2D (Figure 3(b)). If one of these conditions is verified, then a corresponding rotation should start. Otherwise, we see if similar conditions are verified with a crossed exterior edge. Note that we increase the edge length when computing intersections. Hence, users do not need to cross the cube edges exactly. When the box is almost aligned with the viewing direction, we also consider the depth-oriented edges that are not visible, as illustrated in Figure 3(c).



(a) (b) (c)

Figure 3. Axes of rotation are inferred from early finger movements. *Grab* (a) and *Push* (b) strategies. Invisible edges are taken into account for orientations closely aligned with the view (c).

To make the cube spin, users "push" or "grab" it by way of straight gestures. Technically, a linear mapping is applied between users' gestures and tBox rotations. Fast and brief inputs correspond to flicking gestures. The rotation continues with a given inertia, and it stops after a short time of decreasing speed or when the user stops the movement with a *tap* input. Such gestures allow users to quickly specify

coarse cube orientations. Slower and continuous gestures allow the rotations to be refined. This approach is a 3D extension of what is currently done on touch-based mobile devices (e.g. scrolling a long list of phone numbers). Moreover, by using their two hands on both sides of the cube, users can adjust rotation angles from successive inputs while keeping a good visualization of the manipulated object. Figure 4 illustrates tBox being rotated.

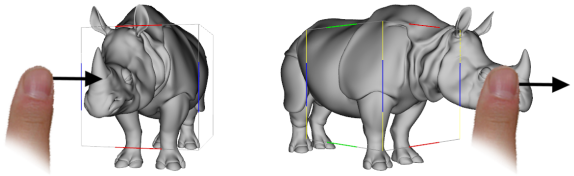
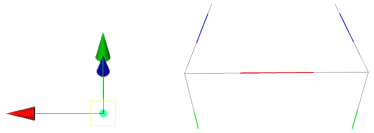


Figure 4. The tBox widget being rotated.

Translation

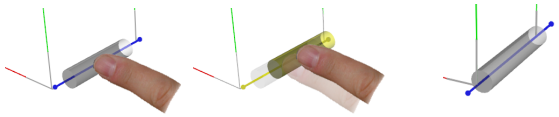
In order to well separate it from rotations, we have based the activation of translations on the (coarse) selection of tBox edges. To apply a translation along a primary axis, the user selects a colored segment that is centered on one of the corresponding edges of the tBox widget. Compared to standard 3D transformation widgets based on triplets of arrows, a box-shape form-factor is valuable when used on touch-screens. Indeed, the visual components to be caught are spread around the focus area, which limits the selection issues linked to the fat finger problem (see Figure 5). Moreover, two to three box edges per direction can be easily selected at any time (except when the cube is facing the viewing direction). Consequently, users can choose the best edge to be caught in order to avoid ergonomic and occlusion issues. Beyond single axial translations, the design of the tBox widget also favours multi-finger input. First, by chaining translation operations with two or three fingers, users can quickly reach 3D locations with successive refinements along the primary axes x, y, z. Then, the simultaneous input of two fingers on axis edges allows translations on planes. A *double-tap* input on an edge with a second finger, attaches the first finger movements to the corresponding translation. Hence, the *translation-in-plane* mode can be controlled from a single finger input.



(a) (b)

Figure 5. Axis selection may be difficult to perform with a standard translation widget (a). A box-shaped widget make it easier (b).

First experiments with tBox have shown that the rotation gestures tended to start unwanted translations, when the finger motion were involuntary started on a tBox edge. Consequently, we introduced a new mechanism to prevent from erroneous inputs. To start a translation along one of the x, y, or z axis, the user touches a corresponding colored segment. However, the translation mode is not activated directly. Instead, the colored segment is transformed into a 3D widget corresponding to a cylinder centered on the active edge, that can be slided along it. The translation mode and, consequently, the effective translation of the manipulated object starts when the cylinder collides with the other edges of the tBox widget as illustrated in Figure 6. This approach is inspired by *sliding widgets*, which are very well suited for tactile interaction [6]. A famous example is the unlock slider of the Apple iPhone where a sliding gesture is preferred compared to a simple *tap* input. The validation tests we performed have shown that these 3D sliding widgets make the rotation interface more robust. Note that when several translations are chained, the sliding widget is used to activate the first one only.



(a) (b) (c)

Figure 6. A sliding widget appears when a colored segment is touched (a). The translation starts after the slider collides with the other edges of the box (b). When the projected size of an edge is too small, the translation slider is adapted, so the gesture performed before starting a translation is similar for any situations (c).

Scaling

For scaling operations, tBox relies on the standard dual-touch metaphor that allows users to resize 2D objects with *pull apart* and *shrink* gestures. This 2D metaphor is very convincing as the semantic link between the user gesture and the resulting action works well. In our approach, a *pull apart* or *shrink* gesture inside or on both sides of the tBox widget implies the widget, and consequently the attached object, to be resized uniformly (see Figure 7). In addition to uniform scaling, bi-directional scaling along the three primary axes of the tBox widget can be controlled by selecting two opposite edges of a face, and moving them away from each other. Both fingers can be moved in opposite directions at the same time for symmetric scales. If one finger moves while the second one remains static, then the manipulated object is extended in the direction of the moving finger (see Figure 7).

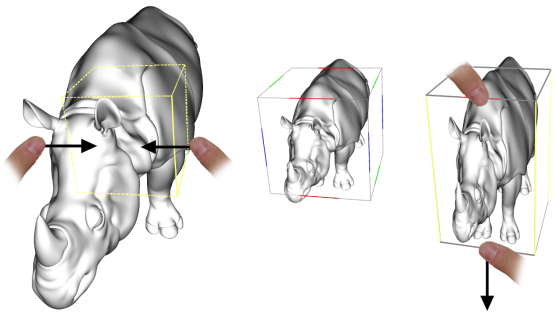


Figure 7. A shrink gesture on both sides of the tBox widget (left) decreases the size of the manipulated object (middle). Moving two opposite edges of a tBox face allows the control of directional scaling along one axis (right).

Additional features

For some widget orientations, some of the tBox edges may project close to each others, which may lead to difficult selections. Consequently, we dynamically compute the edges

that can be selected, and we highlight them with the colored segments, as currently done with many 3D transformation widgets. Concretely, when two edges have their middle point that project at a distance which is smaller than the finger contact area, we keep the one being closer to the observer only. Hence a large tolerance area can be maintained around the edges that can be selected, which ensures easy selection on touch-screens. Note that when an edge is disabled, its corresponding primary axis can still be accessed by another edge (e.g. red axis in Figure 8). Moreover, the edges whose projections are too small are disabled to prevent from inaccurate depth-oriented translations.

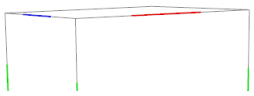


Figure 8. Back edges that project too close to other edges are disabled to avoid erroneous selections.

For selecting or deselecting an object, we use *double-tap* gestures. In addition, standard multi-touch camera controls are used to *zoom* (pull-apart gestures), *orbit* (one finger moving around a 3D point fixed by another finger), or *pan* the view (two joint fingers). Hence, users can directly access all the controls without menus or additional buttons, which favours fast and direct interaction on touch-screens.

FIRST USER FEEDBACK AND CONCLUSION

We conducted an informal user study with eight participants. Four of them had no familiarity with 3D modeling, and four were 3D experts. We asked the participants to play with the interface, with no precise goal. The experimental scene was composed of several objects that can be assembled together for the creation of a character (see Figure 9). The participants were encouraged to "think aloud". We informed them that they were able to control translations, rotations, and scaling as well as the camera view from [multi-]touch gestures, but we did not explained them how to proceed. We observed that the participants discovered almost all the functionalities by themselves very quickly. This confirmed that the affordances provided by the widget allow a good understanding of its functionalities. Moreover, the multi-touch gestures appear to be known as standard gestures by the participants, even for those who are not familiar with multi-touch systems. Note that none of the participants discovered the *translation-in-plane* mode, which appeared as an advanced feature. Both expert and novice users managed to assemble the character as they wanted. The participants reported that they liked the rotation mechanism based on physical behaviors. They indicated that it worked well, and that they managed to apply the rotation they wanted easily. We observed that the activation of translations and scaling was performed without any difficulty. None of the subjects appeared to be disturbed by the sliding widget. This is very beneficial as this mechanism prevents from erroneous activations.

3D transformation widgets have shown undeniable benefits for mouse and keyboard systems. Unfortunately, these UIs are of limited usability when used on touch-screens, as input spaces largely differ. This excludes numerous 3D applications that rely on these UIs from the global tactile evolution of interactive systems. In this paper, we have presented an approach that adapts 3D transformation widgets to the tactile paradigm, inspired from what has been learned these past few years in the scope of 2D touch-based interaction. We have conceived tBox, a new UI that favors the independent control of 9 DOF, and that is complementary to other 3D multi-touch techniques (e.g., [7][2]). Preliminary experiments have shown that tBox can be used easily both by 3D expert and novice users for 3D manipulation tasks on a touch-screen. Further user studies need to be conducted to better understand user performance, in particular for precise and fully-controlled 3D tasks. One research direction that also seems interesting is to continue investigating the relation between 3D shapes perception and user actions, as we started to study with the cube rotation task. We hope that our work will inspire new research in the scope of interaction with 3D content on touch-screens, with the final goal of making rich interactive 3D applications more accessible to everyone.

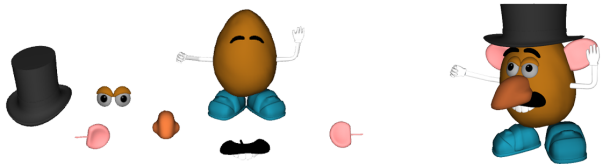


Figure 9. The experimental environment used during the experiment.

ACKNOWLEDGMENTS

This work was supported by the ANR project ANR-09-CORD-013 InSTInCT - http://anr-instinct.cap-sciences.net.

REFERENCES

1. E. A. Bier. Skitters and jacks: interactive 3d positioning tools. In *I3D '86: Proceedings of the 1986 workshop on Interactive 3D graphics*, pages 183–196. ACM, 1987.
2. M. Hancock, S. Carpendale, A. Cockburn, and NZ. Shallow-depth 3d interaction: design and evaluation of one-, two-and three-touch techniques. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1147–1156. ACM, 2007.
3. M. Hancock, T. ten Cate, and S. Carpendale. Sticky tools: Full 6DOF force-based interaction for multi-touch tables. In *Proc. ITS*, pages 145–152, 2009.
4. S. Houde. Iterative design of an interface for easy 3-d direct manipulation. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 135–142. ACM, 1992.
5. A. Martinet, G. Casiez, and L. Grisoni. The effect of dof separation in 3d manipulation tasks with multi-touch displays. In *Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology*, VRST '10, pages 111–118, New York, NY, USA, 2010. ACM.
6. T. Moscovich. Contact area interaction with sliding widgets. In *UIST '09: Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 13–22. ACM, 2009.
7. J. L. Reisman, P. L. Davidson, and J. Y. Han. A screen-space formulation for 2d and 3d direct manipulation. In *UIST '09: Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 69–78. ACM, 2009.
8. R. Schmidt, K. Singh, and R. Balakrishnan. Sketching and composing widgets for 3d manipulation. *Computer Graphics Forum*, 27(2):301–310, 2008. Proceedings of Eurographics 2008.
9. J. O. Wobbrock, M. R. Morris, and A. D. Wilson. User-defined gestures for surface computing. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 1083–1092. ACM, 2009.