LTouchIt: LEGO Modeling on Multi-Touch Surfaces

Daniel Mendes
IST/Technical University of Lisbon
danielmendes@ist.utl.pt

October 2011

Abstract

Presently, multi-touch interactive surfaces have widespread adoption. Taking advantage of such devices, we present an interactive LEGO application, developed accordingly to an adaptation of building block metaphors and direct multi-touch manipulation. Although touch-based interaction has evolved in the past years, manipulation of 3D objects can not be used without know-how of CAD and 3D modeling software. In this work, we explore how 3D object manipulation can be simplified and made available for most users. Our solution (LTouchIt) allows users to create 3D models on a tabletop surface. To prove the validity of our approach, we compared LTouchIt with two LEGO applications, conducting a user study with 20 participants. The results suggest that our touch-based application can compete with existing mouse-based applications.

1 Introduction

Multi-touch surfaces introduced new interaction paradigms, different from those provided by traditional interfaces which rely on Windows, Icons, Menus and Pointing devices, denoted as WIMP interfaces [15]. Although touch-based interaction has evolved in the past years, manipulation of 3D objects can not be used without know-how of CAD and 3D modeling software, making them unsuitable for entertainment. In this work, we explore how 3D object manipulation can be simplified and made available for most users. Therefore, we chose the LEGO scenario, which is familiar to users of all ages. Nevertheless, we expect our findings to be of value to general applications that require direct manipulation of 3D content.

For most people, LEGO stands for more than a toy manufacturer. Many recognize it as the playful plastic blocks (bricks) that echo onto our childhood imagination. We envision that multi-touch interaction can surpass existing mouse-based LEGO applications for educational and entertaining purposes, namely for exhibition, public display and museums. We present an interactive LEGO application, that supports bimanual multi-touch input to create 3D models on the tabletop surface without expertise on 3D or CAD software.

Throughout the following sections we present an overview of existing virtual LEGO applications and research on 3D object manipulation for multi-touch surfaces. We follow with a detailed description of our solution, one that we believe can address 3D manipulation and still remain fun for building virtual LEGO models. Then, an evaluation methodology is described where our prototype is compared against two existing applications, which is followed by an analysis and discussion of user evaluation sessions. Lastly, we draw conclusions from our research and pinpoint possible ways to improve upon this work.

2 Related Work

As multi-touch surfaces become more available for exhibition and learning contexts, we propose an interactive LEGO application that takes 3D manipulation in consideration and remain coherent with actions available in other virtual LEGO solutions. Therefore, we review existing LEGO applications from a critical perspective, drawing conclusions for our proposal.

To build LEGO models on a multi-touch tabletop, the manipulation of virtual 3D objects challenge must be addressed. This challenge has been subject of research in Human Computer Interaction. In this section, we also present an overview of the field, focusing on metaphors that provide bimanual manipulation supported by multi-touch surfaces.

2.1 LEGO Applications

Following the technological advances we have been witnessing, some toys have been ported to the digital world. LEGO construction bricks are no exception and, currently, several applications allow to create virtual LEGO models. However, the majority of these applications are mouse-based, something we believe that diminishes the fun-factor in such contexts. We present a survey on digital LEGO applications, focused on both popular applications and innovative examples.

LEGO Digital Designer (LDD)[1], is a proprietary application of the LEGO Company. Models are created in a 3D environment represented through a visible auxiliary grid, resembling a traditional LEGO

[1] ldd.lego.com, last accessed on October 13, 2011.

Figure 1: LEGO Digital Designer.

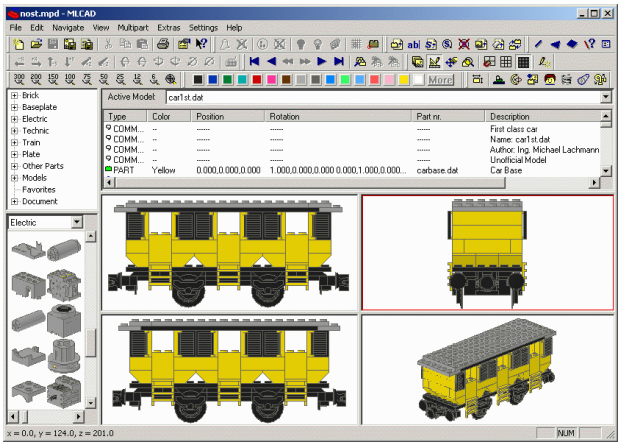

Figure 2: Mike's LEGO CAD.

baseplate, as depicted in Figure 1. LEGO bricks are displayed, through their previews, in a browsable list. Interaction with on-screen widgets allow the user to orbit the camera around a point, without tilting (no roll); zoom also relies on widgets, while panning requires a mouse movement combined with keyboard shortcut. Manipulation of bricks relies on an efficient system of connecting parts, based on the grid concept. Therefore, the translation of a brick is performed exclusively in the grid plane, to which it adapts. If the user translates a brick onto an existing one, the new brick is placed above. Rotation is restricted along two axes, which are perspective-dependent; thus, it requires reasoning regarding which camera position is best for the desired rotation effect.

Mike's LEGO CAD (MLCad)[2] is a Computer Aided-Design (CAD) system for building virtual LEGO models. It utilizes four viewports, each providing a different view of the 3D model, as depicted in Figure 2. Both perspective (non-editable, only for visualisation) and orthogonal views (for editing purposes) are supported. As expected, being CAD-based, it does not suit all users, in particular those that are not acquainted with the paradigms of typical CAD software. Thus, as our tests will show, it hardens learning curve for building virtual LEGO models. It integrates the LDraw[3] open-source parts library, which is widely used by the community of LEGO aficionados. Bricks are displayed in two browsable panels, a list of brick names (textual) and another with their previews (graphical). Unlike most LEGO applications, the environment of MLCad does not provide an auxiliary grid, and there is no restriction for the position of the bricks. Regarding manipulation of bricks, translation occurs in a plane parallel to the selected orthogonal view; while rotation can be executed in any of the three axes (roll, pitch and yaw).

LeoCAD[4], much like the previous system, also utilizes the CAD paradigm and LDraw library, but allows manipulation in both perspective and orthogonal views. Unlike the remainder applications, searching for bricks is accomplished through a text-based list, since the graphical preview only becomes available once a brick has been selected. From an interaction perspective, LeoCAD requires mode switching via interface buttons, e.g., the user has to swap explicitly from rotation to translation. By default, brick translation is performed in a horizontal plane; hence, if the user intends to move the object vertically, it is required to drag the corresponding axis (each object displays a 3D axis widget) instead of simply dragging the object. In order to rotate bricks, it follows the approach of Shoemake [14], displaying rotation handles. Once grabbed (via mouse) each of these three handles, restrict the rotation effect to a single plane. Furthermore, the camera can be rotated around its own axis, tilted (roll) or orbited around a point.

Moving away from WIMP metaphor, **LSketchIt** [13] is a calligraphic tool, that allows LEGO models to be built through sketching. The system is built upon LeoCAD, and shares many of its features, although the retrieval and selection of bricks is achieved by drawing a sketched-up version of the desired brick. Given the brick outline (sketch), the system presents a list of suggestions and allows the user to make modifications to the brick, refreshing the suggestions accordingly to that modification. The authors suggest that brick retrieval time can be reduced through sketching when compared to LEGO CAD applications.

Recently, boosted by the spread of handheld multi-touch devices (such as the iPad/iPhone), the **Blocks!!**[5] application was commercially released. In this application, camera manipulation is performed with one touch for pan and two for rotation and zoom (pinch zoom). New bricks can be added by selecting them from a list and, then, touching where the brick should be placed. Dragging an object moves it on the grid plane. When objects collide, the selected brick is stacked on top of the other. Further-

[2] `mlcad.lm-software.com`, last accessed on October 13, 2011.

[3] `www.ldraw.org`, last accessed on October 13, 2011.

[4] `www.leocad.org`, last accessed on October 13, 2011.

[5] `itunes.apple.com/us/app/blocks/id390088835`, last accessed on October 13, 2011.

more, bricks can be rotated using two fingers: the first fixes the object; and the second supplies the rotation direction. Multiple taps on a brick cause it to rotate, by 90 degrees intervals for each tap, and, when a full rotation is achieved, the brick is removed from the scene. Since this application is targeted at small multi-touch surfaces, it does not account for whole-hand and bi-manual interaction. A comparison against the remainder applications shows that Blocks!! is only aimed at simple LEGO models, since the brick selection and overall features are quite limited.

A critical perspective of the aforementioned applications allows us to conclude that, currently, there is no virtual LEGO application suited for large multi-touch tabletops, one that we believe will be more adequate for exhibition and educational purposes, while providing natural and pleasing interaction.

2.2 Tabletop 3D Object Manipulation

Research on object manipulation for tabletops started by studying several techniques for rotation and translation of 2D objects, using multi-touch gestures [5]. One of these techniques, the two-point rotation and translation, became the *de facto* standard, and is now popular among several multi-touch applications, even commercial ones. This technique uses two points of contact: the first translates the object; and the second rotates the object around the first touch. When combined with the variation of finger distance, it can be used to scale the object; also known as rotate-scale-translate (RST) or "pinch" when only concerning the scale or zoom effect [17].

As far as three-dimensional manipulation is concerned, various approaches have been proposed. Hancock et al. [3] suggested a set of guidelines to develop multi-touch interfaces, establishing a visual and physical link with the object and 3D visual feedback. Furthermore, these authors present a study regarding the manipulation of 3D objects using one, two and three touches simultaneously. Their results show that with one touch it is possible to extend the Rotate 'N Translate (RNT) algorithm [7] to the third dimension. This suggests that geometric transformations (rotate, scale or translate) should be relative to the touched point and not to the center of the object. Also, in the single touch approach, the rotation can be performed around any of the three axes, while translation occurs in a plane parallel to the visualization plane. The two-touch approach uses the first contact point to apply the RNT algorithm exclusively in the two dimensions of the viewing plane, while the second touch allows rotation around the two remaining axes and translation in the third dimension (the one orthogonal to the viewing plane). Lastly, the three-touch approach (which the authors denote as Sticky Fingers [4] - illustrated in Figure 3), showed optimistic results, faring better than the other two. In this approach, by applying the two-point rotation and translation technique for two-dimensional objects, it is possible to translate the object in two dimensions and rotate around one axis. The scale is replaced by the translation in depth relatively to the camera. For rotation, two touches on the object define a rotation axis, while a third touch, denominated Opposable Thumb, allows the rotation around that axis, providing six DOFs (degrees of freedom).

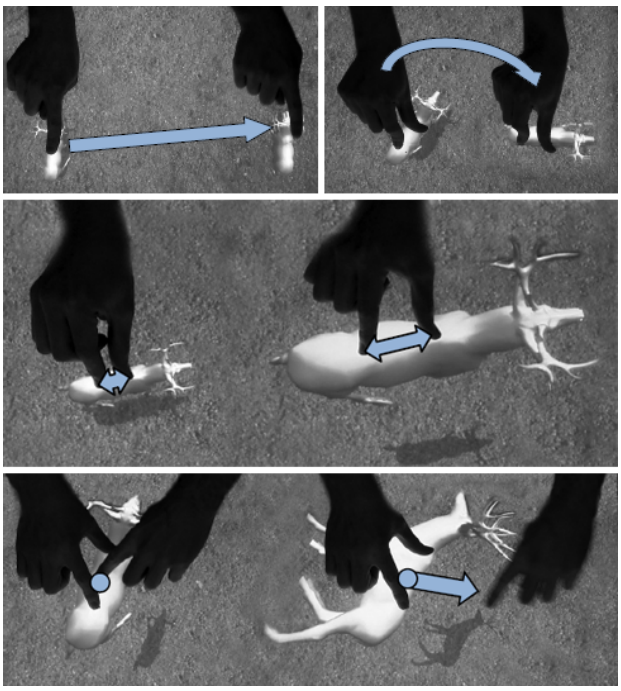

Figure 3: Sticky Fingers and the Opposable Thumb. From [4].

For handling 6 DOFs simultaneously, Reisman et al. [12] introduced a method to extend the RST into 3D. Using only direct touches on the object, a constraint solver calculates the new position and orientation of the object, maintaining the correspondence between the positions of the fingers in the 2D screen and the screen-space target positions, i.e., the 3D points where the user touched the object.

Wilson [16] followed a different approach, in which there is no need to define a specific interaction through gesture recognition. His solution consists in virtual proxies of the user's contact zone with the surface. The interaction between the proxies and the virtual objects follows a physical model, which allows the manipulation of objects in the scene, e.g., translating can be achieved by pushing the object from one side. However, this approach does not address depth manipulation, since the objects are placed in a horizontal plane.

Martinet et al. [8] proposes two techniques to translate 3D objects. The first extends the viewport concept found in many CAD applications (four viewports, each displaying an different view of the model). Touching and dragging the object within one of the viewports translates the object in a plane parallel to that view. Manipulating the object, with a second touch, in a different viewport modifies depth relatively to the first touch. For the second method,

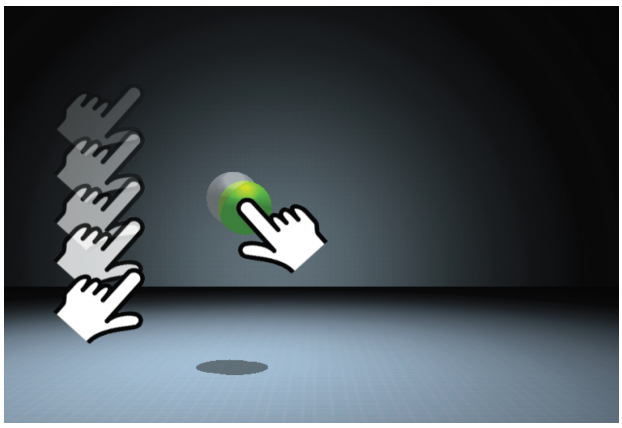

Figure 4: Illustration of the Z-Technique. From [8].

denoted as Z-technique (Figure 4), only one view of the scene is employed. In this technique, the first touch moves the object in the plane parallel to the view, while the backward-forward motion of a second touch controls the depth relatively to the camera position. The authors preliminary evaluation suggests that users prefer the Z-technique.

Improving upon the Z-Technique, Martinet et al. [9] added the constraint solver described in [12] to perform rotations, separating the control of translations from rotations. With one direct touch on the object the user can move it on the camera plane, and through a simultaneous indirect touch its depth can be manipulated. Also, with two or more direct touches the object can be rotated. A comparison of this approach to Sticky Fingers and Screen-Space [12] showed that separation of translation and rotation can lead to better performance.

Cohe et al. [1] propose a widget for indirect manipulation of the object in 9 DOFs (three for translations, three for rotations and three for scales) separately. It consists of a wireframe box which the user can interact with, instead of the actual object itself. The object is translated by dragging the edge of the box corresponding to the desired axis. Dragging two opposite edges scales the object. Rotation is defined by the direction of user motion, once the user starts dragging a face of the box.

The aforementioned work covers interaction with 3D objects on multi-touch surfaces, but most are targeted at scenarios where the camera does not move, or at least does not rotate. Moreover, scenes with different cameras, such as an isometric perspective, will require additional cognitive effort from the user in order to understand the plane in which the objects are moving. It might be also difficult for a user to comprehend which gestures are needed to translate an object to a desired position, e.g., to move the object in a horizontal plane. Besides, as stated in [9], its difficult to interact with small objects with multi-finger approaches, since there is no sufficient space for multiple direct touches on the same object.

Our work addresses these issues, since they are common challenges in 3D building block applications, such as the LEGO scenario. We propose an application, with free camera manipulation, that provides easy navigation and understandable 3D object interaction (6 DOFs).

3 LTouchIt

We developed an application (**LTouchIt**) to build virtual LEGO models on multi-touch sensitive surfaces, based on the analysis of existing virtual LEGO applications and the state of the art regarding object manipulation on tabletops, as previously described.

To start the development cycle, we conducted an evaluation with 21 users in order to identify the advantages and drawbacks on three of the most commonly used LEGO applications: LDD (LEGO Digital Designer), MLCad and LeoCAD (which are covered in our related work). The results of this evaluation, as described in [10], served as guidelines throughout the development of our solution, one that we hope can surpass the problems identified in existing solutions, and also suit entertainment and educational purposes.

3.1 Architecture Overview

Our application was developed accordingly to the architecture illustrated in Figure 5. While the blue modules consist of existing solutions we adopted, the green modules were developed for our application. The first and most relevant is the Interaction Controller. This module analyses the movements of the fingers on the surface, which are tracked using CCV

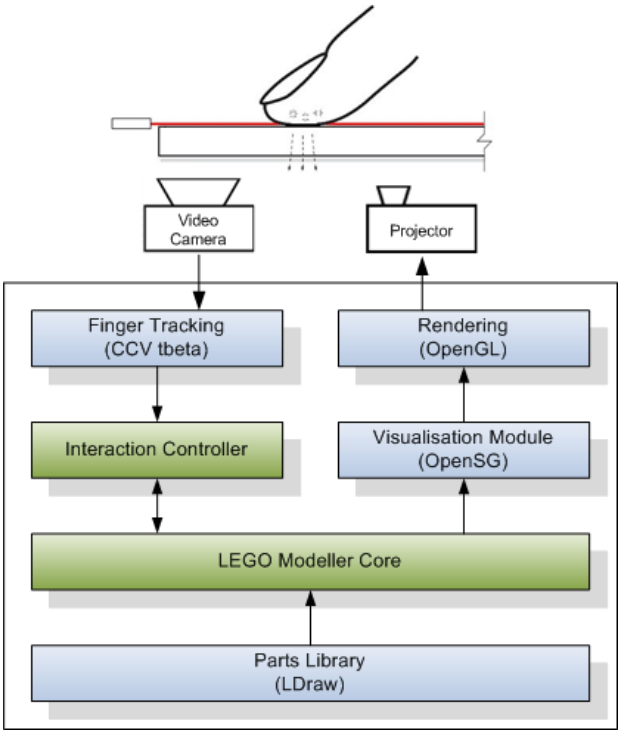

Figure 5: LTouchIt architecture.

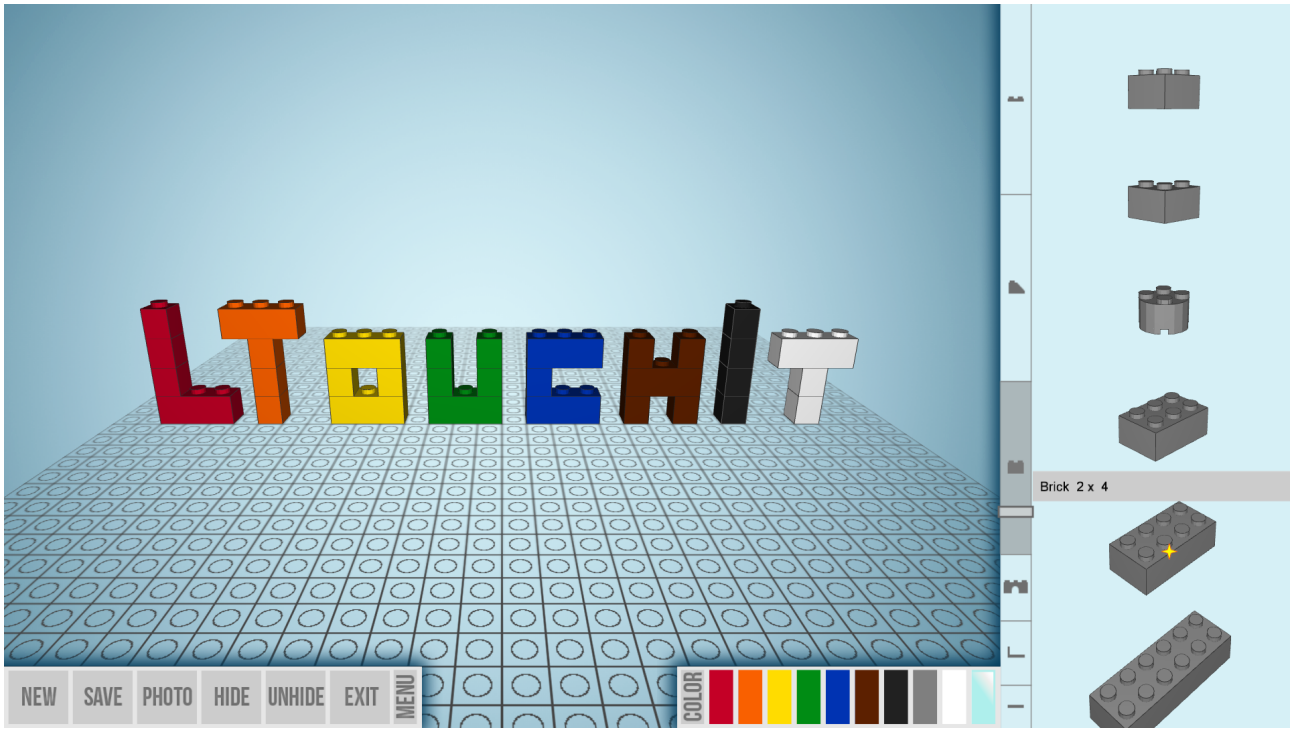

Figure 6: The LTouchIt interface.

tbeta[6], in order to identify the gestures made and the corresponding actions. These actions are then transmitted to the other developed module, the LEGO Modeller. This is the core of the application and it is where all of its logic is implemented. It also contains information about the LEGO bricks, the model being built and the camera.

All the data related to the scene, including the representation of the bricks, their position, orientation and details of the visualization, is stored using OpenSG[7]. This open source graph system undertakes the conversion to OpenGL primitives and the rendering process. The information regarding the geometry of the bricks is gathered from the LDraw parts library. Using the LDraw standard, **LTouchIt** can open construction files saved from other LDraw compatible applications and vice-versa.

3.2 Interface

The interface of **LTouchIt** is depicted in Figure 6. The construction takes place in the center area on a visible grid. The grid adapts if necessary, expanding and contracting as bricks are spread throughout the construction area. It resembles a traditional LEGO baseplate, helping users to fit the bricks, which move in intervals (snapped) of one unit.

The right-side vertical panel holds the available bricks. The user can search through the list by moving it up or down. Within the panel, bricks are organized into groups, making it easier to be retrieved. Also, a scroll bar indicator allows quick access of brick sub-groups by touching on the visual identification (an icon that depicts what type of bricks can be found in the group).

To use a brick, the user can pick it from the list and drag it to the desired position. Conversely, to remove a brick from the construction site, the user can drag it back to the panel. Also, holding the finger on a brick in the panel will display more information, such as the brick name and dimensions.

On the bottom there are two toolbars, which can be expanded and closed. The left toolbar holds all the usual functionalities, such as: create a new model, save the current model and quit the application. Furthermore, it also allows to temporarily hide bricks or to show previously hidden bricks. The right toolbar is a color palette, that can be used to change the color of a brick or the whole brick list.

A final analysis of our interface shows that tabletop design guidelines have been taken into consideration, accounting for: maximizing the interactive area (77% of the area is dedicated for building the models); dominant hand is taken into consideration (as it provides more ergonomic comfort throughout the interaction); and graphical widgets have an appropriate size (enough to be touched and remain partially visible, thus minimizing occlusion).

3.3 Interactions

To manipulate bricks, we aimed at a natural and familiar scenario for LEGO users, moving further from the traditional selection concept - which is to drag objects with a single touch. Instead, we used a “pick” metaphor to grab an object and then, without releas-

[6] `ccv.nuigroup.com`, last accessed on October 13, 2011.
[7] `www.opensg.org`, last accessed on October 13, 2011.

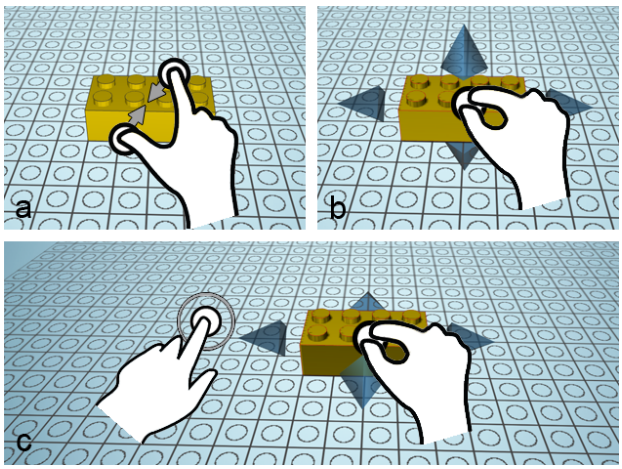

Figure 7: (a) "Pick" gesture; (b) Arrows identifying a vertical translation plane; (c) Changing to a horizontal translation plane.

ing, move it, as depicted in Figure 7. This gesture is resemblant of picking up a real LEGO brick and placing it on the desired location. As our test subjects denoted in their commentaries, this gesture was found to be very familiar and understandable.

In order to assess the best approach for 3D object manipulation on multi-touch tabletops, with unconstrained viewpoints, we carried out a user evaluation with 20 users comparing several approaches for translation and rotation. In the following sections we briefly describe the comparison of these techniques and draw conclusions upon the selected approach. Complementing the object manipulation, we present our camera manipulation and cloning metaphor.

3.3.1 Object Translation

We developed three translation techniques: **Orthogonal** uses a translation plane in which the normal direction is closer to the view vector and orthogonal to one of the scene axes; **Horizontal-Z** moves the object accordingly to a horizontal plane and, by scrolling a second touch, changes the object depth relatively to the camera (similarly to the Z-Technique); and **Plane-Switch** uses a horizontal translation plane which, by tapping a second finger, changes to a vertical plane. For all techniques, dragging the object will move it in the current translation plane; thus, the user manipulates no more than two DOFs with one finger, which maintains a strict relation between the two-dimensional input and the two-dimensional translation.

We compared these approaches amongst each other and with the Z-Technique. Test results showed that users experienced difficulties when translating objects in a plane parallel to the view, suggesting that an orthogonal (to one of the scene axes) translation plane is desirable. Also, we found that manipulating object depth with the scrolling touch was sometimes misunderstood as a scaling gesture. Result analysis denoted **Orthogonal** and **Plane-Switch** as the more efficient

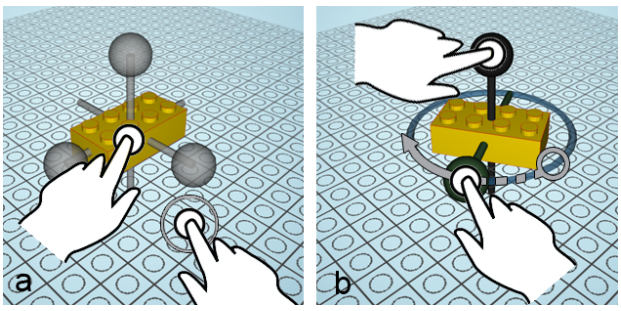

Figure 8: (a) Showing rotation handles; (b) Rotating a brick.

approaches, although without statistically significant differences between them, which suggests that a combination of the two might result in the desired approach (since they are combinable).

Our solution, combines the aforementioned approaches into one (**Grab'N Translate**), as depicted in Figure 7. After picking up the object (Figure 7.a), the user translates it in the plane defined by the camera (Figure 7.b), allowing to take the most of the current view. If the user intends to change the translation plane, no camera movements are required; instead, a tap with a second finger alternates between a horizontal and a vertical plane (Figure 7.c). Visual feedback is provided by four arrows and the shadow of the object. The arrows depict the possible directions in which the object can move, which describes the current translation plane. When the object intersects the construction grid the arrows become red.

3.3.2 Object Rotation

Concerning the rotation of objects, we developed three techniques: **Camera-Defined-Axis** uses the RST applied to the current translation plane, i.e., while moving the object, a second touch rotates the object around the first; **User-Defined-Axis** is similar to the Opposable Thumb, but the rotation axis is one of the scene axes; **Rotation-Handles** uses virtual handles (similar to object handles in [2]), the first touch selects the rotation axis and the second touch rotates the object. These techniques only allow to control rotation over one DOF at a time, something we found desirable in our evaluation of LEGO applications.

User evaluation showed that **Rotation-Handles** outperforms the remainder approaches. Users seem to be more pro-efficient with virtual handles since it provides more comprehensible feedback of the rotation axis, while reducing time to change rotation axis - because users do not need to change views, as observed in **Camera-Defined-Axis**, or define the axis manually, as required by **User-Defined-Axis**. **Rotation-Handles** is the technique used in our solution, complemented with a snap mechanism that automatically fits the brick to the nearest 90 degrees angle when the rotation handle is released. In order to make the the handles visible, the user taps some-

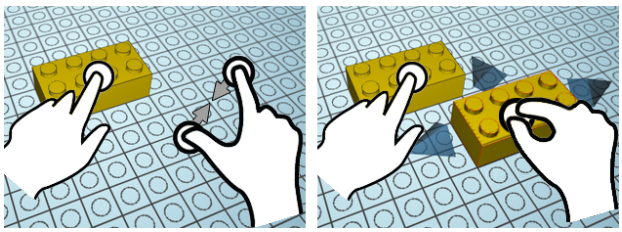

Figure 9: Gesture for cloning a brick.

where on the surface while holding the object with the other finger, as illustrated in Figure 8.

3.3.3 Camera Manipulation

To manipulate the camera, our solution accounts for orbit, zoom and pan operations. One touch rotates the camera around the model; two touches zoom in and out; and four or more touches in close proximity move the camera. It is possible to quickly re-center the camera by tapping the desired position while panning. Furthermore, throughout observation we found that some users prefer to rotate the camera with two touches, which was also included.

To provide a fluid interaction, we allow concurrent bimanual combination of camera rotation and object translation. Thus, the user can change viewing angle with one hand, while moving the picked object with the other. Moreover, the translation plane will change accordingly to the new perspective.

3.3.4 Brick Cloning

To duplicate (or clone) a brick, preserving its color and orientation, we developed a cloning metaphor: "touch and pick", which is illustrated in Figure 9. The gesture consists of pointing to the object that we want to duplicate with one finger of the non-dominant hand and, then, picking the same object with a pick gesture using the dominant hand.

Once the "pick" gesture is completed, the new brick is created and placed underneath the dominant hand, allowing the user to move it to the desired position. Since we found that it can be difficult to touch and pick small sized objects, we allow to pick outside the brick, as depicted in Figure 9. Furthermore, this enables users to create the cloned object directly on the desired position.

4 User Evaluation

To evaluate our prototype application (**LtouchIt**), we compared it with two existing LEGO applications based on the WIMP paradigm. We chose LEGO Digital Designer (**LDD**) and **MLCad**, since our preliminary evaluation [10] suggested they were the most pleasing for users. Furthermore, LDD is a proprietary LEGO tool, while MLCad is a community-driven effort.

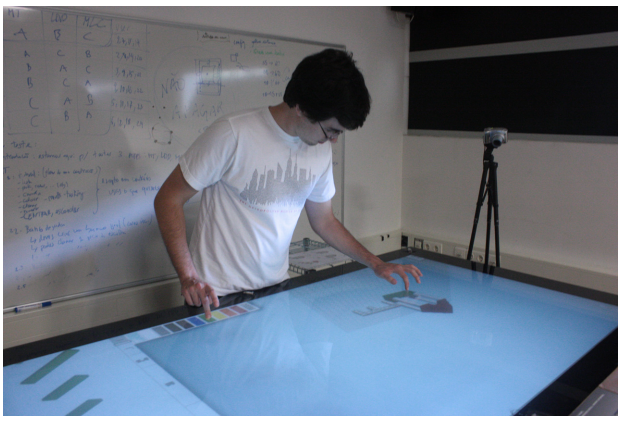

Figure 10: User interacting with LTouchIt.

4.1 Participants

Evaluation was carried out by 20 users (15 male and five female; all right handed), ranging from 11 to 26 years old (mean=23.5 years old). Concerning previous experience with CAD or 3D modelling applications, 10% of the users have some experience and 20% are very experienced with such software. Only 15% have never used a multi-touch device, while the remainder have experience with multi-touch smartphones. None of the test subjects had used a LEGO modelling application before.

4.2 Apparatus

Tests were conducted in a closed environment, featuring: a computer (with mouse and keyboard) running LDD and MLCad; and our multi-touch enabled surface (1.58x0.87m optical tabletop), which is depicted in Figure 10. With the users' permission, tests were videotaped and comments were transcribed from audio recording.

4.3 Test Description

Tests were structured in four stages: a pre-test questionnaire to establish user profile; briefing about test purpose and training session; three tasks (one for each application, each preceded by a demonstration of the application); followed by a questionnaire after completing the task in each application. To ensure even test distribution of the applications, we alternated the application order for each user.

Each test session was comprised of three situations: **LDD**, **MLCad** (both running on a desktop computer with a mouse device) and **LTouchIt** (on the tabletop). For each application the user was required to build the LEGO model depicted in Figure 11. Users were given a sheet of paper, describing the desired task (step-by-step building instructions), resemblant of a construction guide that come with real LEGO model kits. Users were encouraged to study the model beforehand, although they were free to consult it throughout the task.

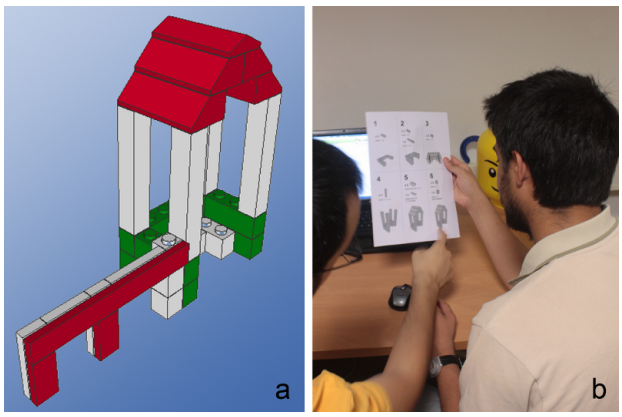

Figure 11: (a) The LEGO model; (b) Test briefing.

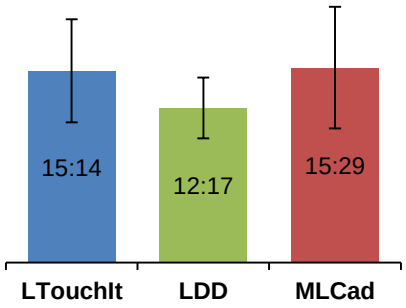

Figure 12: Average task completion time (mm:ss) per application.

The LEGO model (Figure 11) contains several bricks of different types and colours, thus users were required to search through different groups and paint with different colours. Also, the inclusion of repeated bricks in the model allowed us to understand if users prefer to pick the same brick from the list or to clone it. Finally, building this model requires rotating bricks around all three axes.

All participants received a LEGO model kit, as gratitude for the time spent in our evaluation.

4.4 Results

We present three different perspectives on the analysis of the results from our user study. Firstly, we present a quantitative analysis drawn from task completion times in each application; we follow with a qualitative analysis based on the questionnaires; and finally, we discuss several observations captured throughout the test sessions.

4.4.1 Task Completion Time

For each application, we measured the time required to complete the construction of the model. The results follow a normal distributed, accordingly to the Shapiro-Wilk test. Thus, we subjected our results to an One-Way ANOVA test, which suggested that statistically significant differences existed ($F_{2,57}$=4.123, p¡.05). A Post-hoc Tukey HSD multiple comparisons test revealed that the **LDD** was significantly different from **MLCad**.

	LTouchIt	LDD	MLCad
Fun*	3,0 (1)	3,0 (2)	2,0 (2)
Move*	4,0 (1)	3,0 (2)	4,0 (1)
Rotate*	4,0 (1)	2,0 (1)	3,0 (1)
Pan	4,0 (1)	4,0 (1)	3,0 (2)
Orbit	4,0 (1)	4,0 (0)	-
Zoom	4,0 (1)	4,0 (0)	4,0 (0)
Search*	4,0 (1)	3,0 (1)	2,0 (1)
Clone	4,0 (0)	4,0 (1)	4,0 (2)
Paint*	4,0 (0)	3,5 (1)	3,0 (1)

* indicates statistical significance

Table 1: Questionnaire results for each application (Median, Inter-quartile Range).

The average completion time for each application is depicted in Figure 12. **LDD**, due to inclusion of collision detection and brick adaptation systems, was the fastest. **MLCad**, although fast for some 3D expert users, was generally the slowest. **LTouchIt**, despite using an interaction paradigm which participants are not acquainted, and without the automatic brick fitting, was able to perform slightly faster than **MLCad**.

4.4.2 User Feedback

In the questionnaires, we asked users to classify, using a 4 points Likert scale (1 - none, 4 - very), each application regarding: how fun it was to use; how easy it was to manipulate bricks (translation and rotation); camera control (pan, orbit and zoom); search and retrieve bricks; clone bricks; and paint. The participants' ratings are shown in Table 1. The Wilcoxon Signed Rank Test was used to assess statistically significant differences.

Users strongly agreed that **MLCad** is less fun than **LTouchIt** and **LDD** (Z=-3.043, p=.002 and Z=-2.586, p=.010). Concerning brick manipulation, users strongly agreed that **MLCad** is easier to use than **LDD** (Z=-1.979, p=.048). The **LDD** brick adaptation system, although efficient in most cases, might give frustrating outcomes occasionally. For instance, when the user wants to place a brick side by side with another, depending on the perspective, the application might place it above or below the first, which is incorrect for the user. Also for brick manipulation, **LTouchIt** classification is very close to **MLCad**. Moreover, users strongly agreed that rotating bricks is more difficult in **LDD** than in **LTouchIt** or **MLCad** (Z=-3.125, p=.002 and Z=-2.045, p=.041). Participant commentaries suggest that the virtual handles of **LTouchIt** are preferred over the rotation buttons in the **MLCad** interface.

Regarding camera manipulation, none stood out.

The gestures used in **LTouchIt** were able to compete with other applications solutions with no significant differences in users' preference. Since **MLCad** does not allow camera rotation in the editable viewports, we did not include the orbit classification for this application.

To retrieve the desired bricks from list, users strongly agreed that **LTouchIt** was easier than **LDD** and **MLCad** (Z=-3.493, p¡.001 and Z=-3.697, p¡.001). The same is observed for the user preferences regarding colouring bricks (Z=-2.877, p=.004 and Z=-3.166, p=.002), with **LTouchIt** above the remainder. Our clone feature showed less deviation than **LDD** and **MLCad**, suggesting that the gesture was found understandable by most users, even when compared to the clone mode in **LDD** and the keyboard shortcuts used in **MLCad** to "copy and paste" bricks.

4.4.3 Observations

We observed throughout user evaluation that orthogonal viewports paradigm, which is employed in **MLCad**, is not natural for non CAD experts. Conversely, we witnessed that our solution simplifies 3D manipulations for non expert users.

In **MLCad** users shown no problems in rotating bricks, since viewports allow the user to maintain different views of the model. In this case, all interactions are 2D, because they take place on the viewports, rather than on a perspective view. In **LDD** users tend to have occasional problems with 3D rotations, because the rotation operation is perspective-dependent, and the available rotation axes are not identified, which most users found frustrating. Our solution goes one step further, by allowing 3D manipulation with free camera, but avoiding the aforementioned issues.

Furthermore, user comments suggest that the **LTouchIt** pick metaphor tends to be more natural than the brick selection in **LDD**. In **LTouchIt**, once a brick is picked, the user inherently knows that it is being held, since the two fingers are in contact and the pick and move gestures are fluidly merged. Conversely, **LDD** requires the user to click on the desired brick and, once again, click for placing it in the target position, thus, not providing tactile feedback that the brick was held. This observation also conforms with the known advantages of direct over indirect manipulation [6].

We observed that most users found our clone gesture to be a fast technique for speeding up constructions that often have repeated bricks, such as walls.

5 Conclusions

In this paper we presented a system for virtual LEGO modelling, specifically designed for multi-touch tabletop devices. These devices offer new interaction possibilities, but also several challenges that must be surpassed. We focused primarily on multi-touch manipulation of 3D objects on a large surface, developing a set of gestures that can be applied into a variety of research domains, outside the LEGO scope.

We conducted and evaluation with 20 users, comparing our proposal with two LEGO applications. Results suggest that our interactive application is in tune with its competition, yet it provides a hands-on experience, one that we believe to be more adequate for entertainment and educational purposes than existing LEGO applications. Furthermore, from user commentaries and questionnaire analysis, we can conclude that most participants found it easy to manipulate 3D objects without expertise in CAD or 3D software.

Our findings are generalizable towards research on interaction with 3D objects for multi-touch tabletops. Our tests showed that users are comfortable with separation of rotation and translation into atomic actions (as stated in [9]) rather than allowing combined actions. Also, these manipulations are separated in DOFs, which we found helpful for novice users. Translation can be performed on plane (two DOFs) which maps directly to the device affordance, since tabletops offer bi-dimensional input (X-Y for the touch coordinates). Whereas for rotation only one degree is considered at a time, since users found multiple DOFs to be more cognitively demanding. This seems to corroborate with [1, 9] as user agreeability over gestures for 3D rotation lean towards actions that manipulate one DOF at a time.

Furthermore, our scenario addressed manipulation of small objects on a low resolution input device (since not all tabletops are able to distinguish very close finger touches). Methods such as Sticky Fingers [4] require that three fingers are placed on an object, which for small objects can be problematic since the object size must be at least wide enough to accommodate all fingers in a comfortable position. The usage of handles overcomes this situation, since the manipulation is no longer directly on the object, as proposed for 2D and 3D multi-touch interactions by [11, 1].

6 Future Work

We believe that our system can evolve from the current prototype to a fully fledged tool for LEGO fans of all ages. To achieve this vision, new features can be developed, such as: collision detection (fitting bricks together automatically); improving upon brick retrieval (one that will fit into larger sets of data); support all standard LEGO bricks; group selection (manipulate multiple bricks at once); and finally, multi-user collaboration around the same tabletop. These functionalities raise a whole new set of challenges to be tackled in the future.

References

[1] A. Coh´e, F. Decle, and M. Hachet. tBox: A 3D Transformation Widget designed for Touch-

screens. In *ACM CHI Conference on Human Factors in Computing Systems*, Vancouver, Canada, May 2011.

[2] B. Conner, S. Snibbe, K. Herndon, D. Robbins, R. Zeleznik, and A. Van Dam. Three-dimensional widgets. In *Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 183–188. ACM, 1992.

[3] M. Hancock, S. Carpendale, and A. Cockburn. Shallow-depth 3d interaction: Design and evaluation of one-, two-and three-touch techniques. In *Proc. CHI 2007*, pages 1147–1156, New York, NY, USA, 2007. ACM Press.

[4] M. Hancock, T. ten Cate, and S. Carpendale. Sticky tools: Full 6DOF force-based interaction for multi-touchtables. In *Proc. ITS*, pages 145–152, 2009.

[5] M. S. Hancock, F. Vernier, D. Wigdor, S. Carpendale, and C. Shen. Rotation and translation mechanisms for tabletop interaction. In *Proc. Tabletop*, pages 79–86. IEEE Press, 2006.

[6] K. Kin, M. Agrawala, and T. DeRose. Determining the benefits of direct-touch, bimanual, and multifinger input on a multitouch workstation. In *Proceedings of Graphics Interface 2009*, pages 119–124. Canadian Information Processing Society, 2009.

[7] R. Kruger, S. Carpendale, S. Scott, and A. Tang. Fluid integration of rotation and translation. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 601–610. ACM, 2005.

[8] A. Martinet, G. Casiez, and L. Grisoni. The design and evaluation of 3d positioning techniques for multi-touch displays. In *3D User Interfaces (3DUI), 2010 IEEE Symposium on*, pages 115–118. IEEE, 2010.

[9] 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*, pages 111–118. ACM, 2010.

[10] D. Mendes and A. Ferreira. Virtual lego modelling on multi-touch surfaces. In *WSCG'2011 Full Papers Proceedings*, pages 73–80, 2011.

[11] M. Nacenta, P. Baudisch, H. Benko, and A. Wilson. Separability of spatial manipulations in multi-touch interfaces. In *Proceedings of Graphics Interface 2009*, pages 175–182. Canadian Information Processing Society, 2009.

[12] J. Reisman, P. Davidson, and J. Han. A screen-space formulation for 2d and 3d direct manipulation. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 69–78. ACM, 2009.

[13] T. Santos, A. Ferreira, F. Dias, and M. J. Fonseca. Using sketches and retrieval to create lego models. *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling*, 2008.

[14] K. Shoemake. Arcball: a user interface for specifying three-dimensional orientation using a mouse. In *Graphics Interface*, volume 92, pages 151–156, 1992.

[15] A. Van Dam. Post-wimp user interfaces. *Communications of the ACM*, 40(2):63–67, 1997.

[16] A. Wilson. Simulating grasping behavior on an imaging interactive surface. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 125–132. ACM, 2009.

[17] J. O. Wobbrock, M. R. Morris, and A. D. Wilson. User-defined gestures for surface computing. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pages 1083–1092, New York, NY, USA, 2009. ACM.