Virtual LEGO Modelling on Multi-Touch Surfaces

Daniel Mendes
DEI/IST/Technical University of Lisbon
Av Rovisco Pais, 1
1049-001 Lisboa
Portugal
danielmendes@ist.utl.pt

Alfredo Ferreira
INESC-ID/IST/Technical University of Lisbon
Rua Alves Redol, 9
1000-029 Lisboa
Portugal
alfredo.ferreira@inesc-id.pt

ABSTRACT

Construction of LEGO models is a popular hobby, not only among children and young teenagers, but also for adults of all ages. Following the technological evolution and the integration of computers into the everyday life, several applications for virtual LEGO modelling have been created. However, these applications generally have interfaces based on windows, icons, menus and pointing devices, the so-called WIMP interfaces, thus being unnatural and hard-to-use for many users. Taking advantage of new trends in of interaction paradigms we developed an innovative solution for virtual LEGO modelling using a horizontal multi-touch surface. To achieve better results, we selected the most common virtual LEGO applications and performed a comparative study, identifying advantages and disadvantages of each one. In this paper we briefly present that study and describe the application developed upon it.

Keywords: 3D Modelling, 3D Object Manipulation, Multi-touch Interaction, LEGO Models

1 INTRODUCTION

With the popularization of multi-touch technology and the decreasing cost associated with its construction, multi-touch interaction is now common among general public. Indeed, from mobile devices to large multi-touch surfaces, a wide range of devices are available. These devices introduced new interaction paradigms, different than those provided by traditional interfaces based on windows, icons, menus and pointing devices, the WIMP interfaces [16]. While some of these paradigms already reached a maturity level, others are still in its infancy. For instance, the manipulation of two-dimensional objects in multi-touch surfaces has been clearly defined and a *de facto* standard is recognized for rotation, translation and scale [6]. On the other hand, several approaches to interaction with three-dimensional objects and 3D scene navigation have been proposed, but none were recognized as a definitive solution.

Due to this obstacle, the number of available multi-touch applications using 3D environments for the common user is limited. During the development a virtual LEGO modelling tool for multi-touch surfaces, we propose an application for manipulating 3D objects that we believe to be easy to understand and natural to use for a generic audience.

Plzen, Czech Republic.

Throughout childhood, many people played with building blocks that fit together to create what the imagination dictated. LEGO is the most famous manufacturer of such toys, globally known for its building blocks and beyond. In reality, building LEGO models is an activity shared by people of all ages. As a complement to traditional plastic blocks, there are now several applications that allows the construction of virtual models.

Although it was shown that building with virtual blocks using a mouse is considerably slower than building LEGO in the real world [1], there are also studies that show that the multi-touch interfaces based on gestures are more efficient than traditional ones. In [8] it was compared the performances of direct-touch, bi-manual and multi-finger for a task of multi target selection. It was concluded that "direct touch with a single finger provides a large performance benefit over over using a mouse and that bi-manual interaction provides a smaller additional benefit." In [2] was tested the selection, translation, rotation and scale of objects in a 3D scene. Results show that, using a multi touch interface, users can perform faster than using WIMP interfaces. We believe that using multi-touch surfaces for virtual LEGO modelling will provide an improved interaction experience for users through more natural and easy to use interfaces.

In the following sections we present a selection of existing virtual LEGO modelling tools, together with a comparative study that identifies advantages and disadvantages of each application. Then, we introduce the current status of multi-touch interfaces regarding ma-

nipulation of 3D objects. Next, we describe in detail our approach, with a special focus on LEGO blocks representation in memory and their manipulation in a multi-touch tabletop. Finally, we present some conclusions drawn from the work developed so far and identify the path for future work.

2 VIRTUAL LEGO MODELLING APPLICATIONS

Currently, several applications allow the creation of virtual LEGO models, each presenting peculiarities in relation to others: some want to give the feeling of building a LEGO model as it would be in reality, while others follow a more technical approach, oriented to three-dimensional modelling, some use open-source library parts, while others are proprietary and use their own parts system.

LEGO Digital Designer (LDD)[1], is a proprietary application of the LEGO Company. The LEGO modelling is done in a three-dimensional environment and has an efficient system of connecting parts, preventing their overlap.

This application displays all the pieces available using a list with their previews. It has always a visible grid, simulating a base plate, as can be seen in Figure 1 (a). The movement of parts is done exclusively in the grid plane, to which they adapt. Their rotation is done only along two axes, using intervals of 90 degrees. The camera movement is done by orbiting around a point and never tilting.

Mike's Lego CAD (MLCad)[2] is a CAD system applied to building LEGO models. It uses four viewports with different views of the model: three orthogonal (one top, one front and one side) and a perspective, as seen in Figure 1 (b). However, it only allows changes to the model made in the orthogonal views, and the perspective view is for viewing only. It uses the LDraw[3] open-source parts library.

The search for the parts is carried out using a textual list of names and another for their previews. The movement of the parts is carried out only in a plane parallel to the orthogonal views, which do not have any kind of auxiliary grid and there is no restriction on the parts position. Parts rotations are made at intervals of 90 degrees accordingly to three axes.

LeoCAD[4] uses the LDraw parts library, as the previous application. Its options provide a grid to which the pieces fit into each half unit, and various sets of viewports. Both the grid and the viewports are not active by default but can be activated if necessary. To switch between different functions, such as moving parts or rotating the camera, it is always necessary to use the buttons on the interface.

The pieces are presented through a list of groups whose identification, both as parts and their groups, is done just by text, although, after selecting a part in the list, there is a window with a its preview. The pieces translation is done in a horizontal plane, and to move in a vertical plane is needed a special action. For the rotation of the parts it followed the same approach as proposed in [14] to specify the orientation using a traditional mouse, visible in Figure 1 (c). Performing the rotation by clicking on the circumference causes the rotation to be done exclusively on an axis, at intervals of 30 degrees. This application offers many ways to rotate the camera: rotate around her own axis, orbiting around a point and tilt.

There are more applications to create virtual LEGO models, but these three are the most popular between those that have a WIMP interface. LSketchIt [12], built upon LeoCAD, has the particularity to enable the construction of LEGO models using sketches. The search and selection of parts is made by drawing sketches of the pieces to be obtained in the place that the user wants to put it. Given this outline, the program presents a list of suggestions and allows the user to make modifications to the piece, giving new suggestions according to that modification.

Currently there is no application for the creation of virtual LEGO models developed for a multi-touch based interaction.

3 COMPARATIVE STUDY

With the aim of trying to find out the best approach in interacting with objects in multi-touch surfaces, including virtual LEGO building blocks, we started by analysing existing solutions, which use the usual WIMP interfaces, described in the previous section.

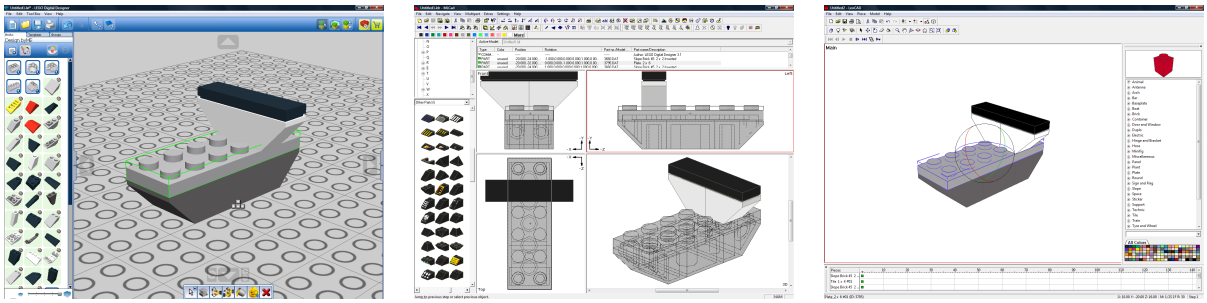
3.1 Users Testing

We developed a set of tests, which had the participation of twenty one users, to evaluate the three most common virtual LEGO modelling applications: LEGO Digital Designer, MLCad and LeoCAD. In order to determine the main positive and negative aspects of the various applications we conducted a series of tests with users, that followed the methodology suggested by Preece et al. in [11].

At the beginning of the experiment, the main features of each application were presented to the user, and it were followed by a training phase, consisting of a period of three minutes to interact freely with each application.

After the training phase, each user was asked to complete a task on each application. The task was to con-

[1] LEGO Digital Designer: http://ldd.lego.com last visited on October 20, 2010.
[2] MLCad: http://www.lm-software.com/mlcad, last visited on October 20, 2010.
[3] LDraw: http://www.ldraw.org, last visited on October 20, 2010.
[4] LeoCAD: http://www.leocad.org, last visited on October 20, 2010.



(a) (b) (c)

Figure 1: Most common virtual LEGO applications: (a) LEGO Digital Designer; (b) MLCad; (c) LeoCAD.

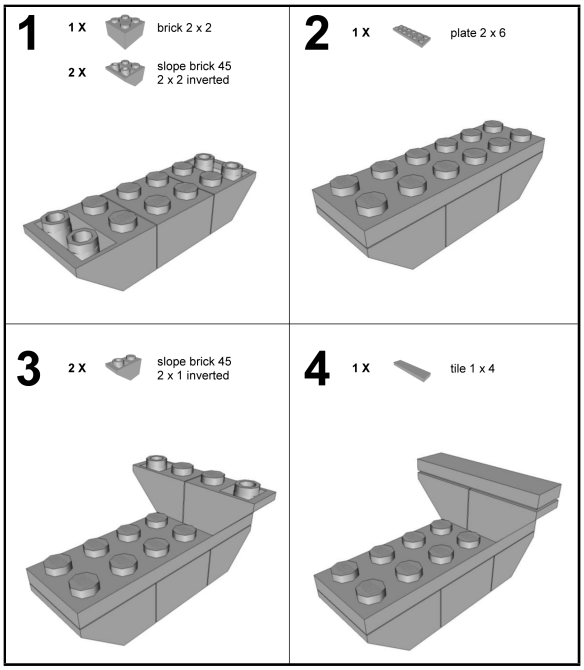


Figure 2: Requested task for users to complete.

struct a simple model, illustrated in Figure 2, which involved searching, selecting, manipulating, rotating and placing various parts. The order in which the users used the applications to carry out the task was randomly selected for each one. In this step, users were asked to think aloud as they performed the task, thus we can better understand what were the main difficulties that users encountered during the task. Since our main goal was to discover what most appeals to users, we did not use any performance measure because it is incompatible with the think aloud technique.

Having accomplished the task in the three applications, each user completed a short questionnaire that focused on the experience with each one. It was hoped thus to identify the strengths and weaknesses in each one of them, with particular emphasis on manipulation of the parts and the camera. The questionnaire consisted on a first set of multiple choice questions to identify the user profile, like age, qualifications and computer experience with image editing, 3D modelling and virtual LEGO tools. It was followed by a second set with five questions to classify the various aspects of each application using a Likert scale with four values and an open-response question so that users could leave comments for each application. These five questions were related to translation and rotation of parts, camera manipulation, searching and selecting parts and overall satisfaction.

3.2 Results

Through the analysis of the qualitative evaluation performed by the users, extracted from their feedback, both written in the questionnaire and said during the task execution, we identified important factors to consider in developing an application for visualization and manipulation of LEGO building blocks. Several of these factors may also be valid in another type of three-dimensional modelling applications.

The results obtained followed a normal distribution, demonstrated by application of the Shapiro-Wilk test, and the average classification was different for each application, proved with an One-Way ANOVA. Results showed that LEGO Digital Designer was the most appealing to users, in all aspects, followed by MLCad and, finally, LeoCAD, with negative ratings for everything except for searching parts. In [10] there are detailed results and analysis thereof. The main advantages and drawbacks we found in each application are summarized in Table 1.

The presented study revealed that users expect a system that tries to simulate the behaviour of the pieces in the real world, contributing to a more familiar interaction with the application. For example, it is desirable to prevent two parts from overlapping each other and pin their positions to a grid. On the other hand, we realize that too much freedom can lead to user frustration, as the case of an application that provided rotation of the pieces according to the three axes simultaneously.

From the conclusions of this study we were able to develop our solution taking advantage of strengths found in the existing applications, as well as sugges-

LDD	+ Search parts by image
	+ Effective part positioning system
	+ 90 degrees interval for parts rotation
	+ Simple orbiting camera
	- No part size information
	- Only two axes for the rotation of parts
	- No information about which rotation axis is being used
MLCad	+ Viewports good for experienced users
	+ Bricks rotations clearly identified by axis and direction
	+ Allows the rotation of parts around three axes
	+ 90 degrees interval for parts rotation
	- Groups of parts identified by name
	- Does not prevent parts overlapping
	- Viewports bad for novice users
	- Need to use interface buttons to rotate parts
LeoCAD	+ Allows rotation of parts around three axes
	- Search of parts only by name
	- Does not prevent parts overlapping
	- Allows rotation of parts around more than one axis simultaneously
	- 30 degrees interval for parts rotation
	- Camera controls, other than orbiting, are frustrating
	- Constant need to use interface buttons

Table 1: Main advantages(+) and drawbacks(-) of tested applications.

tions made by the users. Our solution combines this strengths and suggestions with a multi-touch interface.

4 MULTI-TOUCH INTERACTION

Following the evolution of computers, interaction methods also change. According to van Dam [16], unlike the predictable evolution of computer components described by Moore's law, interfaces suffer from long periods of stability followed by an abrupt transition. Currently the most popular interfaces are based on windows, icons, menus and pointing devices, such as the mouse, the so-called WIMP interfaces. Since these interfaces are the most popular for over twenty-five years and the technologies capable of providing an interaction more natural and powerful, closer to human interaction, being more common, there is now a need to develop the next generation of interfaces, which the author calls the Post-WIMP.

In [13] it is said that multi-touch based interfaces offer more exciting and efficient ways to interact with information. These interfaces have demonstrated natural metaphors, allowing a coordinated manipulation to replace what would have been multiple actions controlling a cursor.

With the recent work of Jeff Han [3], the interest in touch sensitive surfaces has increased and has accelerated the development of more accessible technologies to create such surfaces. Today there are surfaces that can detect multiple points of contact simultaneously, using different technologies.

4.1 Multi-Touch Surfaces

With the emergence of multi-touch surfaces, the first step was, naturally, to create new forms of interaction to existing applications. In [7] and [15] an interaction with a multi-touch surface to control Google Earth and Warcraft 3 was implemented. In that context, a set of gestures was developed, both with one and two hands, allowing a more natural way to control the navigation within the applications.

In [18] the authors explored both visualization and interaction techniques to support shared environments and privacy. They created a prototype for organizing a furniture plant, RoomPlanner, which is based on a diverse set of gestures using one finger, two fingers, one hand and two hands.

These are just some examples showing the capabilities of multi-touch surfaces and interaction possibilities. A plethora of applications for this type of surfaces exists, though none was made for virtual LEGO modelling.

4.2 3D Objects Manipulation

Although there are a *de facto* standard to manipulate two-dimensional objects using multi-touch surfaces, the manipulation of three-dimensional objects does not have a definitive solution yet. Regarding this challenge, there are already several approaches.

Hancock et al. [4, 5] presented a set of techniques for manipulating 3D objects. These techniques consist in maintaining the finger always in touch with the initial point of contact in the object. Using one, two and three touches, the user can simultaneously control up to six degrees of freedom. Test results show that an interaction with more contact points tends to be faster, and with just a touch users end up spending more time to carry out cognitive processing.

Two different approaches for moving objects in a three dimensional space are suggested by Martinet et al. in [9]. The first uses multiple viewports, each one responsible for moving objects in its corresponding viewing plane. The second, called Z-technique, consists of a single perspective view in which, with one touch, the user can move the object in a plane parallel to the view plane and, with a second touch, can adjust the depth of the object.

Andrew Wilson [17] follows a different approach, in which there is no need to program a specific interaction through gesture recognition. His solution is to create virtual proxies in the scene with the shape of the user's contact zone with the surface and then use a physical simulation to control these proxies and, consequently, the manipulation of other objects in the scene. However, given the constraints inherent to LEGO building

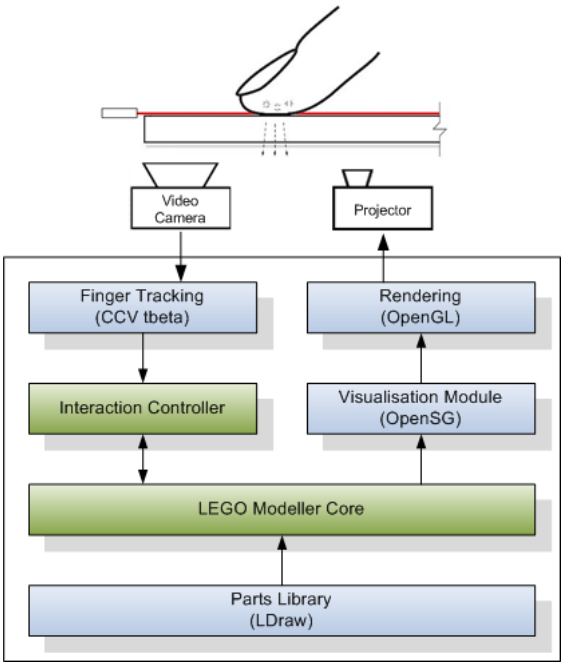


Figure 3: LTouchIt architecture.

blocks, such as the model visualization with free perspectives and the need to move objects in depth, this latter approach does not seem to be most appropriate.

5 LTOUCHIT

Based on the study existing applications, described previously, and on the state of the art regarding the handling of multi-touch surfaces, the development of our solution has been initiated. Next we describe the architecture used, the representation of LEGO pieces and how the interaction can be done.

5.1 Architecture Overview

Our solution is to develop the modules represented in green in the architecture illustrated in Figure 3. The first and most relevant to the job is aimed at the interaction. This is responsible for analysing the movements of the user's fingers from the CCV, in order to determine the gestures made and what action he wants to do. These actions are then passed to the second component, the LEGO modeller. This is the core of the application and it's where all of its logic is programmed. It also contains information about LEGO pieces, the model that is being built and the camera.

All the data related to the scene, including the representation of the used pieces, their position, orientation and details of the visualization, is stored using OpenSG, which undertakes the conversion to OpenGL primitives and the rendering process. The information regarding the pieces to be used in the modeller is gathered from the LDraw parts library. LDraw is an open standard for applications designed to create virtual LEGO models offering a vast library of parts.

Using LEGO modelling as the context, the purpose of this application is the study and development of natural interaction techniques for manipulating three dimensional objects in virtual environments, using multi-touch surfaces.

5.2 LEGO blocks in OpenSG

For the manipulation and visualization of the different LEGO pieces, these have to be converted from its original LDraw format to one that offers features such as transformations and rendering. For this purpose, we have chosen to work with OpenSG[5], an open source scene graph system to create real-time graphics programs. In this conversion we chose to follow an approach as close as possible to the specification of LDraw. Therefore, we shall briefly explain its structure.

The files of this format are formed by six types of data: triangles, quadrilaterals, lines, optional lines, inclusion of sub-files and META commands. The latter are intended to specify the particularities of the current file or the command that will follow. Due to the large amount of these commands, they will not be explained here, except those referring to back-face culling. The inclusion of sub-files is intended to include parts or sub-parts defined in another file within the geometry of the current part. Each included sub-part is associated with a transformation matrix relative to the including part.

The remaining types consist of the representation of the part itself. The optional lines are lines that should only be drawn as some points are visible or not and serve, for example, to draw the contours of cylinders. This type of line, given the complexity of its implementation and its limited relevance, was disposed of in this implementation for now.

Thus, in our application, each piece will have a structure like the one visible in Figure 4. Basically each part is a scene graph, being the part itself the primary node which contains the transformations, as translations and rotations, to be applied to the part. Thus, by applying a transformation to this node, this transformation is automatically applied to all of its children. This node is also the parent node of the geometry nodes of the part and all of it sub-parts. There are two OpenSG nodes for information on the geometry of the piece. The first one has the part geometry itself - triangles and quadrilaterals, and the second has the geometry of the lines. To disable the drawing of lines we simply have to disable this second node.

In each piece there is also a reference to the structure corresponding to each of its sub-parts, in order to be able to change the color of a piece. To do so, we have to change the color of the geometry of the part and the geometry of each of its sub-parts recursively.

[5] OpenSG: http://www.opensg.org, last visited on October 20, 2010.

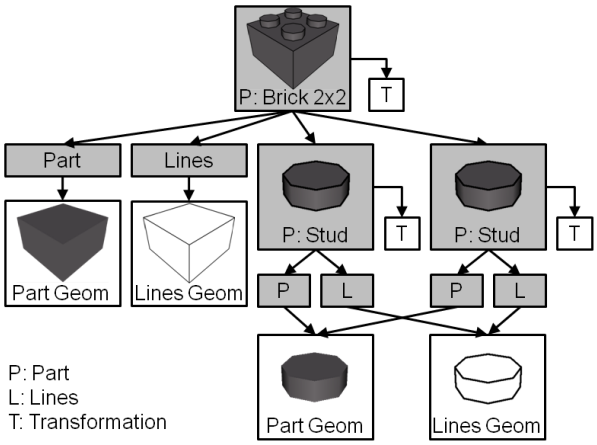


Figure 4: Our LEGO part structure for a simplified 2x2 brick. The grey filled boxes represent OpenSG nodes, while the white ones are the nodes' core.

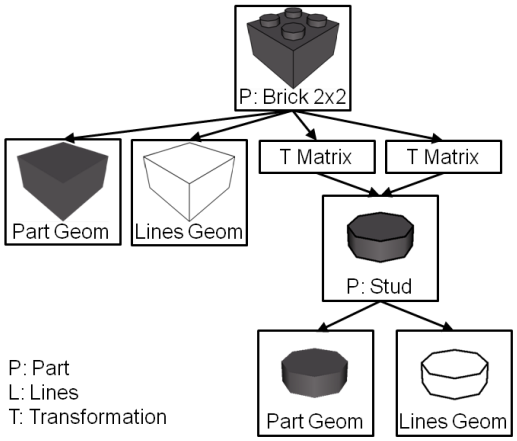


Figure 5: Our LEGO part cache structure for the same 2x2 brick.

To achieve more efficiency in terms of memory and since many parts have sub-parts in common, we built a cache of parts, illustrated in Figure 5, which is fully loaded at the beginning of the application execution. Contrarily to the part structure described above, this cache is not a scene graph. It has, for each part, the geometries required for each color and the transformation matrices to be applied to each sub-parts, also in cache. With this approach, it is possible that only one geometry is created for each part, being shared by all parts that include it, requiring only the application of the transformation matrix for each case.

As mentioned, the only LDraw META commands that will be explained here will be the ones regarding back-face culling. Other ones have been disregarded. The specification of the LDraw file lets one decide using META commands which culling should be made, according to the vertices of the polygon are specified clockwise or counter-clockwise. In addition, there is a META command that allows to reverse the test for the clipping of polygons in the middle of the file itself. Thus it was necessary that the insertion order of the polygon vertices in the OpenSG geometry was different depending on the specification of each particular situation, in order to be able to support this diversity.

Regarding the conversion of parts of the LDraw format for OpenSG remains only the fact of both having different coordinate systems. While the former uses a right-handed coordinate system, the latter uses a left-handed one. To circumvent this problem it is necessary for the main node of each piece, and not in any of its sub-parts, to be applied a scale transformation with the values of 1, -1, -1 for x, y and z respectively, i.e. an inversion of the part relatively to the y and z axes.

5.3 Current set of LEGO blocks

Although the LDraw parts library contains more than three and a half thousand pieces, not all of them were considered in our application. This measure was undertaken with the aim of reducing the complexity of the implemented solution, since the primary focus of this work is the manipulation of three dimensional objects in multi-touch surfaces, and not to achieve the total diversity of the supported parts.

We discarded all the pieces that have geometries that contain incorrectly specified polygons. These are polygons that contain points that are not co-planar, non convex angles and/or vertices that are not defined in the correct order. We also accept only pieces that support some sort of back-face culling and that use only the default LDraw color, so that the piece is all coloured alike. Whenever any piece includes some sub-part that does not meet the requirements cited above, it is also discarded.

With these restrictions, we currently support more than eight hundred pieces and more than eighty colours, including colours with alpha. It is expected, however, that the number of used parts will decrease, given the complexity of some of them regarding the bricks adaptation and collision detection, something that will be further developed.

5.4 LEGO Modelling on a Tabletop

A first approach to interact with LEGO bricks, described next, is already implemented. Our solution is being developed in a multi-touch table that uses the technology Laser Light Plane (LLP). This table has dimensions of 180x120 cm and uses a projector with 720p resolution.

LLP technology allows the detection of multiple points of interaction using the dispersion caused by the user fingers on the laser beams placed over the surface. This technology has a lower construction cost and is more accurate than other optical technologies like FTIR, DI and DSI.

Our application, illustrated in Figure 6, is designed for Microsoft Windows environment, using C++ programming language and OpenSG.

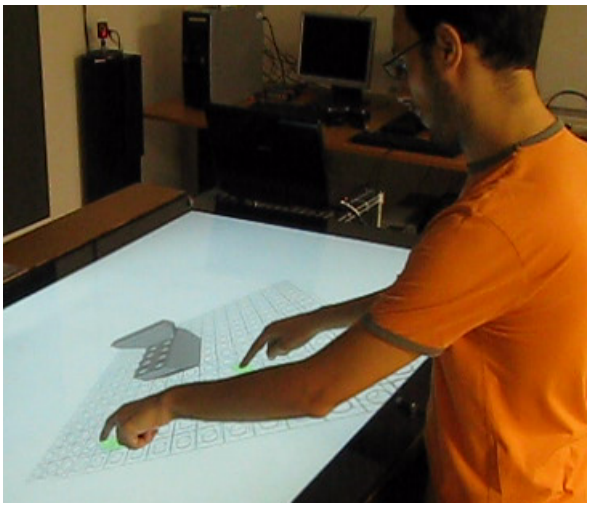


Figure 6: User interacting with LTouchIt.

Inside the table, the camera receives the rays reflected by the fingers of the user and the obtained image is sent to the CCV tbeta. This program processes this image, recognizes the touches and sends their positions for the Interaction Controller, using the TUIO protocol. Whenever the controller receives a message from CCV tbeta, it originates an event. This event is then processed and the position of the fingers over time is analysed to determine the gestures of the user, transmitting to the corresponding actions to the Core, such as moving a piece or rotate the camera.

For the generation of the image to be projected on the table, OpenSG goes through the scene graph of the model being worked on the LEGO Core, originating all the OpenGL commands needed to render the scene, such as the geometries of all the objects in the scene and the position of the camera and the light sources. In order to have the geometry information of all the LEGO bricks, these are all loaded and stored in the Core, according to a procedure described in the previous sections.

It is already implemented the internal representation of LEGO pieces and the interaction techniques next described.

5.5 Manipulation of virtual LEGO blocks

To interact with three dimensional objects on multi-touch surfaces, which in our case are LEGO bricks, we followed some of the techniques already presented.

Concerning the translation of the pieces, we turned to two approaches. The first is used to move them in an horizontal plane. This approach consists of, after selecting a piece, dragging it by moving a finger in contact with it, making the piece follow the touch, while remaining in this plane. This is the technique that can be found in several applications currently available for creating LEGO models. We chose this approach instead of moving the object in a plane parallel to the camera plane [9][5], since we believe that this movement can become confusing when the camera plane is not perpendicular to one of the scene axis, due to the translation being made in the three axis simultaneously.

For the movement of the pieces according to the third dimension is used a technique similar to the Z-technique [9]. Keeping a first finger in contact with the piece, it is used the vertical motion of a second finger to regulate its depth in relation to the camera, as can be seen in Figure 7.

The application also has a grid that underlies the construction in which the parts fit after every movement.

Regarding the control of the camera while viewing the model, we followed an approach used in some existing applications of virtual LEGO. While keeping a point of the model centred, the user can orbit around that point by dragging a finger, whose touch was started out of a selected piece. The movement is made like dragging a spherical surface, moving sideways, up and down, with the particularity that it never tilts. To zoom in and out the scene, we again use the vertical motion of a second touch, as shown in Figure 8. This gesture is identical to the Z-technique, thus diminishing the vocabulary needed to interact with the application.

6 CONCLUSIONS AND FUTURE WORK

In this paper we presented an innovative application for for constructing virtual LEGO models. This application takes advantage of multi-touch interfaces on a distinct approach to LEGO modelling. However, the current prototype is not a final version. Indeed it stands more as a proof-of-concept prototype with which we seek to find the most natural way to interact with three dimensional objects - the LEGO blocks - in multi-touch surfaces.

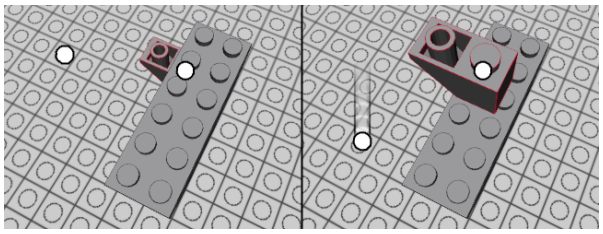


Figure 7: Depth manipulation using two touches.

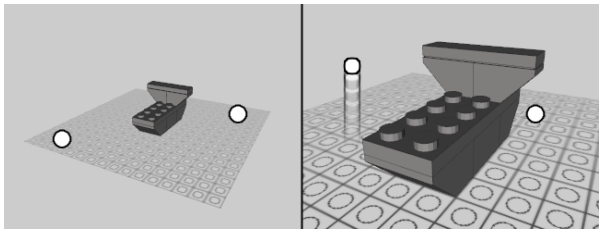


Figure 8: Camera zoom using two touches.

The development of this prototype was based on the results of a comparative study of existing virtual LEGO modelling applications. This procedure guided us into developing a solution easy-to-use for all users. However, as stated before, we are aware that such application still in its infancy and a considerable effort must be dedicated to augmented it with additional features.

Indeed, the next step of our work will focus on extensive user evaluation of our LTouchIt prototype. With this evaluation we intend not only to validate the proposed approach for virtual LEGO modelling, but also to identify further enhancements in the prototype. We believe that we are on the correct path for introducing a simple and easy-to-use 3D manipulation on multi-touch surfaces that will allow users to build LEGO models in a virtual environment as fast as they do in real world.

7 ACKNOWLEDGEMENTS

The work described in this paper was partially supported by the Portuguese Foundation for Science and Technology (FCT) through the project 3DORuS, reference PTDC/EIA-EIA/102930/2008.

REFERENCES

[1] H. Baradaran and W. Stuerzlinger. A Comparison of Real and Virtual 3D Construction Tools with Novice Users. *CGVR'06*, 2006.

[2] D. Fiorella, A. Sanna, and F. Lamberti. Multi-touch user interface evaluation for 3d object manipulation on mobile devices. *Journal on Multimodal User Interfaces*, 2009.

[3] J. Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. *Proceedings of the 18th annual ACM symposium on User interface software and technology*, 2005.

[4] M. Hancock, S. Carpendale, and A. Cockburn. Shallow-depth 3d interaction: Design and evaluation of one-, two- and three-touch techniques. *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2007.

[5] M. Hancock, T. ten Cate, and S. Carpendale. Sticky tools: Full 6dof force-based interaction for multi-touch tables. *ITS '09: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, 2009.

[6] M. S. Hancock, S. Carpendale, F. D. Vernier, D. Wigdor, and C. Shen. Rotation and translation mechanisms for tabletop interaction. In *TABLETOP '06: Proceedings of the First IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, 2006.

[7] S.-G. Kim, J.-W. Kim, K.-T. Bae, and C.-W. Lee. Multi-touch interaction for table-top display. *Advances in Artificial Reality and Tele-Existence*, 2006.

[8] K. Kin, M. Agrawala, and T. DeRose. Determining the benefits of direct-touch, bimanual, and multifinger input on a multitouch workstation. In *GI '09: Proceedings of Graphics Interface 2009*. Canadian Information Processing Society, 2009.

[9] A. Martinet, G. Casiez, and L. Grisoni. The design and evaluation of 3d positioning techniques for multi-touch displays. *IEEE Symposium on 3D User Interfaces 2010*, 2010.

[10] D. Mendes and A. Ferreira. Estudo comparativo de aplicações para a construção de modelos lego. In *Actas da 4ª Conferência Nacional em Interacção Humano-Computador*, 2010.

[11] J. Preece, Y. Rogers, and H. Sharp. *Interaction Design: Beyond Human-Computer Interaction*, chapter 14. John Wiley and Sons Ltd, 2002.

[12] 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.

[13] T. Selker. Touching the future. *Communications of the ACM*, 2008.

[14] K. Shoemake. Arcball: a user interface for specifying three-dimensional orientation using a mouse. In *Proceedings of the conference on Graphics interface '92*, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[15] E. Tse, C. Shen, S. Greenberg, and C. Forlines. Enabling interaction with single user applications through speech and gestures on a multi-user tabletop. *AVI '06: Proceedings of the working conference on Advanced visual interfaces*, 2006.

[16] A. van Dam. Post-wimp user interfaces. *Communications of the ACM*, 1997.

[17] A. Wilson. Simulating grasping behavior on an imaging interactive surface. *ITS '09: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, 2009.

[18] M. Wu and R. Balakrishnan. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. *Proceedings of the 16th annual ACM symposium on User interface software and technology*, 2003.