TANGICAD: TANGIBLE INTERFACE FOR MANIPULATING ARCHITECTURAL 3D MODELS

SHERIF M. ABDELMOHSEN & ELLEN YI-LUEN DO
College of Architecture, Georgia Institute of Technology,
247 Fourth Street, Atlanta, GA 30332-0155, USA
sherif.morad@gatech.edu, ellenodo@gatech.edu

Abstract. This paper presents an interface for interacting with tangible objects to produce and edit architectural 3D models, called TangiCAD. TangiCAD is a tangible/virtual construction kit which allows architects to manipulate virtual models using easy hand control of tangible cubes, as an alternative interface for 3D modeling. It consists of a set of tangible cubes representing architectural elements, such as walls, columns, slabs,…etc., in addition to some editing operations. With more developed versions, the paper argues that architects could use tangible interfaces to carry out 3D modeling in an intuitive way, using their "flip-the-box" hands-on movement.

1. Motivation

Tangible blocks are significant tools for manipulation, especially with architects and designers, who prefer hand manipulation and physical interaction with tangible models rather than graphical user interfaces (GUIs). Evidence has shown that tangible user interfaces (TUIs) have more positive effects on designers’ spatial cognition, perception and reasoning for visuo-spatial information (Kim and Maher, 2006; Sharlin et al., 2002).

Kim et al. (2005) argue that TUIs are usually passive elements, hardly interpreted by computers once arranged in a tangible setting. The manipulation and visualization of virtual building models, however, is limited by the use of graphical display constraints such as the visual screen, keyboard and mouse. Confusion and delay can occur along the shifted focus between computer controls and represented data, thus decreasing focus on the model configuration instead of the tools used to build it.

TangiCAD, shown in Figure 1, has attempted to address the issue of achieving best results out of both GUIs and TUIs. TangiCAD was designed for the purpose of addressing the virtue of manipulating blocks physically and the virtue of enclosing design alternatives virtually for future modification, comparison of alternatives, and digitally representing the built model. It takes advantage of the human ability to easily manipulate tangible objects which represent a corresponding "wall", "column", or "slab" in the virtual world. It also provides effortless and easy hand tactile interaction with tangible objects to generate diverse design alternatives for tangible design attempts, allowing future digital manipulation and more exchangeable formats and media.
2. Relevant Work

Several TUI applications are evolving as more natural HCI alternatives to the standard mouse and keyboard interface. Sutphen et al. (2000) uses tangible planes on a sensor bed to generate a 3D representation of the planes in a virtual environment setting. Fjeld et al. (1998) BUILD-IT project supports complex planning and composition tasks by allowing users, grouped around a table, to interact in a virtual scene, using physical bricks to select and manipulate virtual models. Gorbet et al. (1998) “Triangles” provides tactile feedback through physical connection of triangles to trigger certain events and manipulate complex digital information. Watanabe et al. (2004) “Active Cube” allows users to interact with 3D environments by using physical cubes through a real-time and bi-directional user interface.

Fitzmaurice et al. (1995) “Bricks” introduces the concept of “graspable user interfaces” that allow direct control of virtual objects through physical handles, using physical artifacts as input devices that can be tightly coupled to virtual objects for manipulation. Dias et al. (2002) developed “MIXDesign”; a mixed reality system which allows architects to interact with a physical model by using tangible interfaces with AR Toolkit patterns and paddle gestures. The “Navigational Blocks” by Camarata et al. (2002) demonstrates a tangible interface that facilitates retrieval of historical stories in a tourist spot. The system implements tactile manipulation and haptic feedback, where the orientation, movement, and relative positions of physical blocks support visitor navigation in a virtual gallery.

3. System Overview

TangiCAD consists of a set of tangible cubes, using RF transceivers to communicate wirelessly to a PC. Each side of the cube holds a unique ID. TangiCAD adopts similar command execution hierarchy to conventional CAD packages, with drop-down menus consisting of main operations, such as drawing and modifying, and sub-commands for each operation, which hold details about orientation, dimensions,…etc. The system, however, uses tangible cubes to represent such a hierarchy in an easy to use manner.

The implemented prototype of TangiCAD is composed of two cubes: a controller cube, and a sub-controller cube. The controller cube holds unique ID for the following CAD operations: create, edit, move, rotate, view and preset views. The sub-controller cube holds detailed sub-commands, such as which 3D objects to create, which to edit, where to move them, in which direction they should be rotated, how to view and get preset views for them.
As the user holds one side of the controller cube facing up, the embedded function is then executed. It then waits for a signal from the sub-controller cube to implement a full command. For example, the “create” command waits for a signal to know ‘what to create’; wall, slab, column,…etc. The system is capable of generating multiple instances of the 3D objects. As the user retains the position of the cubes, other instances of the same object are generated.

3.1 CREATING AN OBJECT

When the “create” side is facing up in the controller cube, it issues a “create” operation. It still waits for a signal from the sub-controller cube, which holds a unique ID for each of its six sides. For “create”, each side would create any of these six 3D objects according to its ID (as shown in figure 2): wall, slab, column, sphere, cube, and cylinder. To create a wall, for example, an instance of a wall appears on the screen in a 3D grid window, and is located at the origin. If the user retains the same position for a certain interval of time, another instance of the same 3D object is generated. This interval is set such that commands do not interfere in a way too difficult to perceive by the user. The system then appends the created object to a list of all existing similar 3D objects, e.g. walls, and another list containing all objects in the scene.

<table>
<thead>
<tr>
<th>CONTROLER CUBE</th>
<th>SUB-CONTROLLER CUBE</th>
</tr>
</thead>
<tbody>
<tr>
<td>WALL</td>
<td>SLAB</td>
</tr>
<tr>
<td>COLUMN</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2. “Create” has 6 commands: Wall, Slab, Column, Sphere, Cube and Cylinder

3.2 EDITING AN OBJECT

When the “edit” side is facing up in the controller cube, it issues an “edit” operation. It waits for the sub-controller cube signal, which holds an ID for these commands (as shown in figure 3): select, select all, deselect all, save, color, and exit. The selection method depends on the idea of a selection counter. This counter goes through a selection list consisting of all the objects in the scene, and assigns the selected object a red color to denote selection. When the selected object(s) are selected, they are ready for editing operations such as “move” or “rotate”. When the user flips the sub-controller cube to the “select all” side, all the objects in the scene are added to the selection list and assigned a red color, while they are removed from the selection list and assigned their original color when the user flips it to the “deselect all” side.
3.3 MOVING AN OBJECT

When the “move” side is facing up in the controller cube, it issues a “move” operation, which is not activated unless an object in the scene is selected. The “move” operation only works on objects in the selection list. Depending on which side of the sub-controller cube is flipped, one of the six Cartesian motion modes exists for the selected object(s) (as shown in figure 4): move in +X, −X, +Y, −Y, +Z and −Z directions. The selected 3D object(s) move according to a translation method with a steady increment to the assigned direction, until the user flips the controller cube to another position.

3.4 ROTATING AN OBJECT

When the “rotate” side is facing up in the controller cube, it issues a “rotate” operation. This operation is not activated unless an object in the scene is selected. The “rotate” operation only works on objects in the active selection list. Depending on which side of the sub-controller cube is flipped, one of the six angular motions is activated for the selected object(s) (as shown in figure 5): rotate clockwise around X, counter-clockwise around X, clockwise around Y, counter-clockwise around Y, clockwise around Z, and counter-clockwise around Z. The selected object(s) start to rotate with a steady angular increment until the user flips the controller cube to another position.
3.5 VIEWING AN OBJECT

When the “view” side is facing up in the controller cube, it issues a “view” operation. Depending on which side of the sub-controller cube is flipped, one of six different viewing modes is activated (as shown in figure 6): camera spin, bird's eye view, human's eye view, zoom in, zoom out, and default view. The user can move freely with the assigned camera view around the objects, experience bird's and human's eye view, zoom in and zoom out, and return to the default perspective view, with steady increments, allowing the user to stop upon desire at any given time.

3.6 PRESET VIEWS

When the “preset views” side is facing up in the controller cube, it issues a “preset views” operation. Depending on which side of the sub-controller cube is flipped, one of six preset views is activated (as shown in figure 7): plan view, right view, left view, front view, back view and default view.

TangiCAD can produce an infinite number of the 3D objects and editing operations, thus producing a diversity of compositions. Through multi-user collaboration, multiple users can possibly use these tangible cubes to generate different models in early design phases. A lot of collaborative work could be achieved through the exchange of design views, enhanced with the experience of tangible manipulation and virtual archiving in digital format.
4. Implemented Tools

4.1 HARDWARE

TangiCAD hardware includes a circuit board, containing a microprocessor, SureLink RF transceiver, and 9V battery, placed inside the tangible cube to achieve wireless connection, as shown in figure 8. Three mercury orientation sensors are placed to induce an analog signal in every new position of the cube. The signal from each mercury tube is off if it does not change position, and on if it changes position, thus detecting a new signal for that side. Three outgoing signals and three incoming signals are connected to the transceiver. A QuickLink transceiver is connected directly to PC via a serial to USB cable. It contains a SureLink RF transceiver on top to achieve wireless connection with the two cubes. When one side of the controller cube is facing on top, the SureLink RF transceiver sends a wireless signal through the QuickLink/SureLink connection to the PC. This signal waits for one of the six signals on the sub-controller cube to be activated, which then activates the operations in the implemented software for each of 36 possible annotations defined by the ID on each side of the controller and sub-controller cubes.

4.2 SOFTWARE

TangiCAD is implemented using Python 2.4, with OpenGL. The python file consists of two integrated sets of commands; one involves the connection to the implemented hardware via the SureLink and QuickLink modules, and the other is concerned with defining the operations that should be triggered on each different signal produced by the hardware. The program imports SureLink and QuickLink modules, where the SureLink programmable ID is matched to its corresponding number in the program. A method is defined in the program for the signal sent by the controller cube. This method contains
the programmable SureLink ID number, and receives and sends back the signal from and to the RF transceiver through six pins in the circuit board embedded in each tangible cube. The method then calls a threadtimer class, which defines which operation to call upon each different signal, which corresponds to each side of the cube. The signals in the controller cube wait for signals from the sub-controller cube through another defined method, so that each main operation has its own submenus.

5. Discussion and Future Work

In order to develop TangiCAD into an effective and everyday tool for architects and designers, a few issues should be explored. Object generation in TangiCAD is restricted to fixed modular units, with no user intervention in size alteration, resulting in less design flexibility. Editing is restricted to modular increments, leaving some distances and angles difficult to achieve. More editing operations, such as Boolean operations and scaling, should be integrated. Cubes, being six-sided objects, limit the number of possible functions. It becomes harder to use multiple cubes to operate on simple processes. Future versions would hope for less perplexing and more intelligent translation of operations. When the user, for example, moves the cube while in “move” mode, the 3D object would actually move, when he rotates the cube in “rotate” mode, it would rotate, and when he moves out outward and inward in “view” mode, he could zoom in and out.

Compared to commercially available input and manipulation devices, e.g. 3DConnexion, TangiCAD may lack the smoothness of navigation and control, but can be more promising regarding cognition and perception. As users manipulate “meaningful” tangible objects (tangible objects could represent the nature of the featured virtual command; a “wall” object to represent controls of a virtual wall, a rotating ball for rotation controls, a slider for selection controls), the cognitive load in navigation and editing is reduced. Through embedding design knowledge in the tangible-virtual system, higher levels of cognition and perception could be achieved.

Teamwork collaboration is a significant issue. Optimistic images of TangiCAD hope for multi-user sessions, where the architect and his co-workers use tangible domain-specific “cubes” to express designs ideas. Concerning output formats, models produced by TangiCAD could be exported to other modeling software for further refinement. Users could then be able to print their models out directly on 3D printers or laser cutters.

TangiCAD has been experimented by a group of users from several disciplines at the Open House at Carnegie Mellon University. Most feedback encouraged using a tangible interface as an alternative for virtual modeling due to the reduced cognitive load, but outlined issues concerning perplexity of dealing with multiple cubes and sides, the relatively large size of the current cubes, the need for intuitive representations on the cubes relating to the nature of the command in hand for more ease of use rather than symbolic representations of menus and submenus, and the need for an extension of the current limited 3D commands without being overwhelmed by a large number of tangible controls. Users expressed satisfaction in the first steps of modeling a 3D object, but with more complex models, they could not quickly switch to the right commands and expressed confusion in the selection method, the time lag between the cube flipping and the command execution, and in identifying the required commands without knowing their exact location beforehand.
6. Conclusions

This paper presented TangiCAD, a tangible interface for manipulating architectural 3D models through a set of tangible cubes. TangiCAD is still in its very beginning. It is now capable of generating, selecting and editing 3D objects, but its potentials are more promising. Future improvement should tackle issues related to applicability, usability, embedding design knowledge, multi-user collaboration, design flexibility, and issues that contribute to both better perception of editing commands for the corresponding virtual controls and reduced cognitive load in manipulating the required artifact.

Acknowledgements

This work was carried out at the CoDe Lab, School of Architecture, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA. Thanks go to Mark Gross, Michael Weller, Sheng Kai Tang, and Sora Key at the CoDeLab, for their continuous support and advice throughout implementing this project.

References