TetraTetris: A Study of Multi-User Touch-Based Interaction Using DiamondTouch

Christian Collberg **Stephen Kobourov** **Steven Kobes** **Ben Smith**
Stephen Trush **Gary Yee**

Department of Computer Science, University of Arizona, Tucson, AZ 85721.
{collberg,kobourov,kobes,bsmith,strush,gyee}@cs.arizona.edu

Abstract:
The DiamondTouch table is a touch-sensitive input device that allows several users to interact with a program at the same time and do so using their hands. We describe our exploration of the capabilities and limitations of the DiamondTouch by focusing on TetraTetris, one of the first nontrivial applications written specifically for the DiamondTouch hardware. TetraTetris is a multi-player game, loosely based on the popular game Tetris, in which players use their hands to manipulate objects moving around on the table; it illustrates some of the table's key advantages and drawbacks by comparison with conventional input hardware. In this paper we describe the DiamondTouch hardware, the rules of TetraTetris, the problems we encountered implementing it, and what TetraTetris can show us about the future of input devices like the table.

Keywords: touch table, game, collaboration

1 Introduction

The DiamondTouch table (Dietz & Leigh, 2001) from Mitsubishi Electric Research Laboratories (MERL) is a desktop device that allows up to four users to simultaneously manipulate virtual objects. Users can move objects around on the table by touching and dragging them with their fingers. The purpose of the table is to allow several people to interact with a program at the same time and to do so using their hands rather than more common input devices such as mice.

Many applications could benefit from this novel interface technology. In any situation where a group of people wishes to collaboratively explore and manipulate a plan or design the DiamondTouch table would potentially allow a richer experience than current technology. For example, architects could gather around the table to explore the design of a new housing development. By simply touching the table surface, houses, trees, and roads could be added, deleted, rotated, or moved. Similarly, musicians could gather around the table to collaboratively compose music on virtual instruments. Or, during a natural disaster relief operation, the table could show maps and pictures of the affected area. Those in charge of the rescue operation (including personnel from a variety of backgrounds, such as the police, fire brigade, military, and city administrators) could explore alternative evacuation routes, rescue scenarios, etc.

However, every interface technology comes with its own usability problems and limitations. To explore the usefulness of the DiamondTouch table we have implemented one of the first nontrivial applications written specifically for the DiamondTouch hardware. TetraTetris is a multi-player game, loosely based on Tetris (Burgiel, 1996; Demaine *et al.* , 2002) in which players use their hands to manipulate objects moving around on the table; it illustrates some of the table's key advantages and drawbacks by comparison with conventional input hardware.

In some ways the table is quirky and more difficult to control than a mouse. In other ways it is superior even to the futuristic computers seen in science fiction movies such as *Minority Report* or *Star Trek*. There characters don special gloves to interact with a fast-moving game; to play TetraTetris, however, one need only sit down on a special pad that connects one to the table. The user is completely unencumbered and only has to touch the table with their fingers to interact with the application. More importantly, however, is the table's ability to give the immediate colleagues the ability to work together simultaneously on the same intuitive interface that is also able to differentiate one user from the next.

In the remainder of this paper, we describe the DiamondTouch hardware, the rules of TetraTetris, the problems we encountered implementing it, and what TetraTetris can show us about the future of input devices like the table.

The remainder of this paper is organized as follows. In Section 2 we describe the DiamondTouch hardware. In Section 3 we describe the rules of TetraTetris, a game we designed and implemented to explore the usefulness of the DiamondTouch table. In Section 4 we discuss the problems encountered during implementation. In Section 5 we discuss our experiences with the table, and what TetraTetris can show us about the future of these sorts of input devices. In Section 6 we discuss related work and in Section 7, finally, we summarize our work.

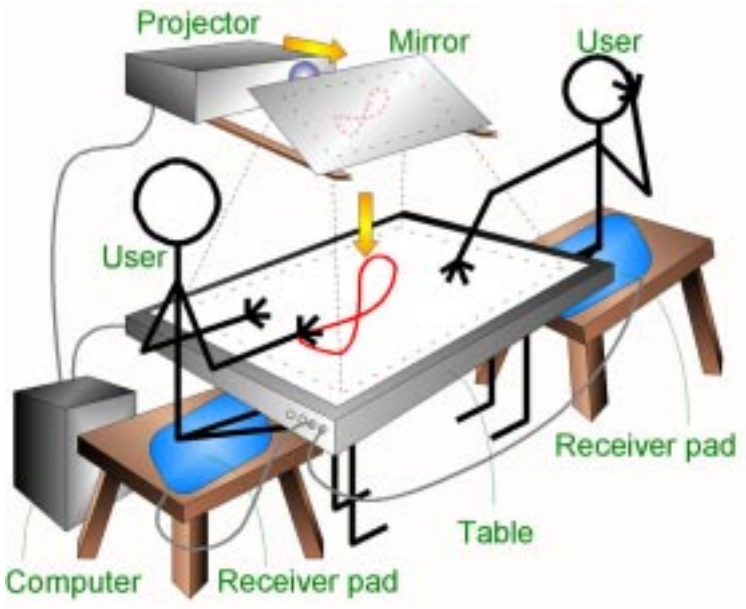


Figure 1: The DiamondTouch table setup.

2 The DiamondTouch Table

Figure 1 illustrates the setup of the table. The surface is approximately $36'' \times 24''$ and contains an array of conductive antennas. Each antenna transmits a signal to the computer that corresponds to the strength of the capacitance between itself and the user. This capacitance is greatest when the user is in direct contact with a particular antenna: a circuit is completed from the antenna, through the user's body, through the receiver pad on the chair in which he/she is sitting, and back into the table.

The table is designed to be used with an ordinary desktop PC or laptop. It sends the data from the antennas to the DiamondTouch SDK drivers through the USB port, allowing the software to examine the data and to determine where on the table the user's fingers are located. The table is not a touch-screen: it has no ability to display output. Instead all images which would normally appear on the display monitor are routed to a video projector which projects them onto the surface of the table with the aid of a mirror and some painstaking calibration.

The complexity of this setup makes transportation of the table difficult. Indeed, Figure 1 is misleading in its slickness because it omits various essential structural supports and the copious quantities of duct tape that were needed to attach them.

Although the table is designed to support up to four simultaneous players, our installation contains only three controller boards and only two receiver pads. Consequently, TetraTetris has only been tested with two players.

3 The Rules of TetraTetris

To allow us to easily study the interactive features of the DiamondTouch table, we decided to build a simple multi-player game, *TetraTetris*. An alternative strategy would have been to retrofit a "real" application (such as a CAD/CAM system) to use multi-user touch-table technology, but this was considered too much work for this initial study.

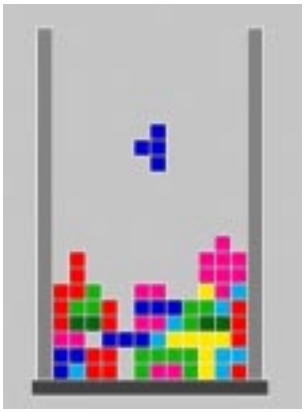


Figure 2: A traditional Tetris game board.

TetraTetris is loosely based on the game of Tetris (Burgiel, 1996), in which the player must direct falling blocks into the correct locations and orientations so as to completely fill horizontal rows (see Figure 2). TetraTetris borrows Tetris's four-square blocks, shown

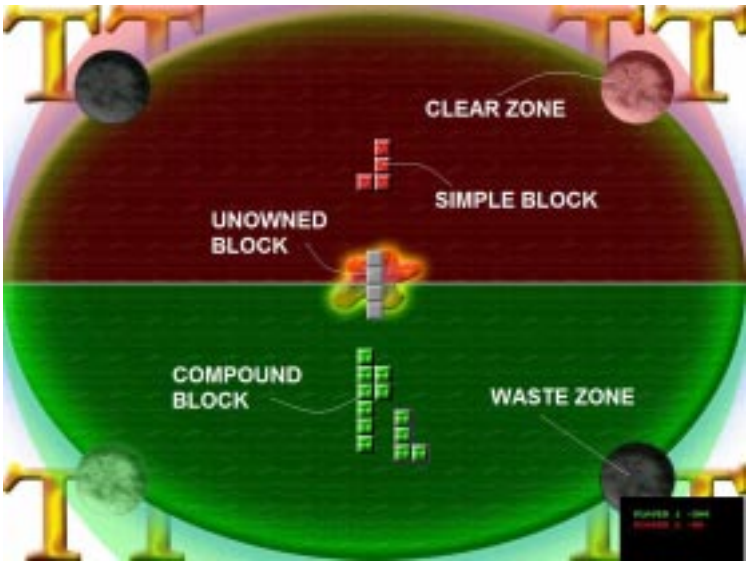


Figure 3: The TetraTetris game board.

in Figure 4—these (and variations formed by rotation) constitute all the shapes that can be produced by arranging four squares together.

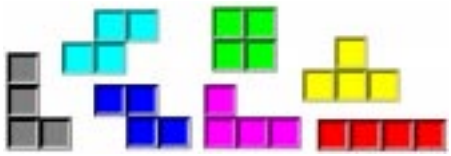


Figure 4: The seven basic Tetris blocks.

TetraTetris derives its name from the fact that it was originally envisioned for four players. However, nothing intrinsic in the design requires that there be four—more or fewer may play, and as noted above, we have only tested with two due to hardware limitations.

The TetraTetris game board, shown in Figure 3, consists of a central source from which blocks emanate periodically. The board area around the source is divided into "player spaces", one for each player. These blocks have an initial momentum upon leaving the center, but there is no notion of rigid gravity as there is in Tetris, and players are free to push blocks around in all directions within the player space.

Every player is designated by a unique color, and if a block is "owned" by a particular player, it is drawn in that player's color and naturally gravitates toward the center of that player's player space. For example, the board in Figure 3 depicts two players, Red and Green, each of whom currently owns one block.

When a block first emerges from the central source, it is unowned and drawn gray. An unowned block becomes owned as soon as:

1. A player grabs it by physically touching or dragging the block with his finger or hand; the player then owns the block. (See Figure 8 (b).)

2. It comes into contact with an owned block; in this case the blocks merge into a "compound block" and ownership is maintained.

The goal of TetraTetris is to form compound blocks out of simple blocks. When two blocks come into contact, they will merge into a compound block, unless they are owned by two different players.

When a player forms a compound block, he/she may "clear" it by moving it to one of the colored, circular "clear zones" located around the perimeter of the game board. When a block is cleared, it disappears from the board and rewards the player with points.

If points could be gained for clearing *any* compound block, the game would be too easy. Consequently TetraTetris rewards symmetry. When a player clears a block, four types of symmetry are checked. These are shown in Figure 5. Figure 8 (c) shows a symmetric block being dragged towards a clear zone.

Figure 5: Four different types of symmetry.

The number of points added to a player's score when a block is cleared is equal to the number of squares inside the block raised to the power of the number of types of symmetry it displays. For example, a block that contains 12 squares and exhibits two types of symmetry is worth 144 points.

If a block exhibits no symmetry at all and is moved to a "waste zone" (any of the gray spheres), and the number of squares inside it is deducted from the player's score.

Several screenshots showing a game in progress can be seen in Figure 9. In Figure 9 (a) two new blocks have emanated from the central source. In Figure 9 (b) the two players have each grabbed a block by touching it with their fingers. The blocks are now owned and have changed color. In Figure 9 (c) the two players have collected and merged several simple blocks into two compound blocks.

4 Implementing TetraTetris

TetraTetris is written in C using Microsoft Visual C++. It uses Simple DirectMedia Layer (SDL) and the Windows API for output. For input, it uses the DiamondTouch library (dtlib) supplied with the SDK. The code is available for download at `http://www.u.arizona.edu/~slk/tt/`.

The two major problems that arose during the development of TetraTetris are summarized below.

4.1 Finicky Electronics

Because the table is triggered by electric current flowing through the body of the user and not by pressure, it is extremely sensitive to the presence of other electronic devices. The table cannot be operated reliably if any part of the user's body is touching a computer or another user. Firm contact with the receiver pad must be made at all times. Even when these rules are followed, input is prone to sporadic fluctuations.

On the software side, this means that the game is very sensitive to the "threshold" setting which determines how strong of a signal an antenna must detect before the application should conclude that a user's finger is upon it. Too high, and the user must press down on the table with great force for it to detect his/her presence. Too low, and the table will go wild with false positives.

4.2 The Bounding Box

This problem is a limitation of the table's design. The table can detect a user touching it in multiple places, because multiple antennas will be activated. However, it was considered infeasible to form a matrix of individually driven antennas corresponding to every pixel under the surface; instead antennas are arranged in a row/column pattern, with a set of row antennas running across the rows and a set of column antennas running down the columns.

This arrangement makes detection of multiple touch points impossible in the most general case, because an antenna has no way to know *where* along its length it is being touched. If only one point is touched, its coordinates can be deduced based on which row antenna is active and which column antenna is active. But if two points are touched simultaneously, we can know only the bounding rectangle that encloses them, because two row antennas and two column antennas are active, but we cannot determine which ones correspond. This is illustrated in Figure 6.

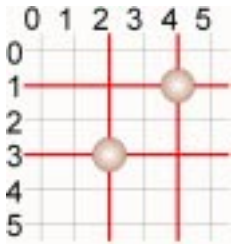


Figure 6: Touching at (4, 1) and (2, 3) is indistinguishable from touching at (2, 1) and (4, 3).

This issue nearly sounds the death knell for intuitive rotation of blocks. We hoped to rotate a block by describing a circular motion around it with the fingers, but if we do not know where the fingers lie within their bounding rectangle, we cannot distinguish a clockwise rotation from a counter-clockwise one. Thus, although the user is able to rotate pieces, the gesture is unreliable and the direction of rotation cannot be controlled. Other mechanisms for indicating rotation might be devised, but that defeats one of the main goals of the table in the first place: making user input *more* intuitive, not less.

4.3 User Setup and Interface

Figure 7 shows the physical setup of the table and the way users interact with it. Figure 9 shows a sequence of screenshots from the TetraTetris game and illustrates how compound blocks are created.

5 Discussion

5.1 Advantages of the Table

TetraTetris highlights a number of advantages of the DiamondTouch input mechanism over the traditional keyboard-and-mouse interface:

No physical separation of input from output. Anyone who has used a laptop trackball knows that an input device can be a cumbersome intermediary between the human hand and the objects on the screen that must be manipulated. In (Cohen *et al.* , 1993), Cohen conclude that when pointing and dragging, a touchscreen interface shows an overall advantage over other input devices when looking at speed and second best rating for accuracy.

A billion years of evolution in a world without computers have trained our brains to be most comfortable with physical objects that we can reach out and grab, right where we see them. That is why someone with no computer experience at all must be

(a) (b)

Figure 7: Figure (a) shows the bottom view of the DiamondTouch Table. Figure (b) shows a top view of the table with two players.

trained to use a mouse before he/she can be taught to use any particular application. But such a person could master interaction with TetraTetris far more easily, because it requires the same sort of coordination that is used in picking up a pencil. A user can see a block, touch the block, and move the block using nothing more than their fingers. These are intuitive skills that any user can easily master.

Fast manipulation of large regions of the screen. Rotations notwithstanding, TetraTetris does find one good use for the "bounding box" concept that arises from multiple touch points. The box described by the user's hands becomes a "lasso" that holds all blocks that fall within it just as if the player had touched each individually. This means the player can move large numbers of blocks across the board in one sweeping motion of the hand.

This sort of control would be nearly impossible to achieve with a mouse. A click-and-drag motion would be necessary to describe the rectangle containing the blocks that are to be moved. Then another click-and-drag could move the blocks to their new location. Finally another click, outside any block, would be necessary to "deselect" the grouping. That's three operations for the user, each one more complex, more time-consuming, and less intuitive than the single hand-sweep of TetraTetris.

Support for multiple simultaneous users. There are three ways for a game to achieve multiplayer functionality with conventional PC input hardware:

- The game can be turn-based—at any given moment only one player has command of the interface (Bjork *et al.* , 2002; Mannien, 2002). This works well for games like chess and checkers, but not so well for action games such as first person shooters.

- The game can allocate a portion of the keyboard to one player and a portion to the other. Racing games sometimes do this: one player uses the arrow keys while the other uses 'A', 'S', 'D', and 'W'. This works well until we want to indicate a location on the screen (e.g. with a mouse).

- The game can be played over a network, with each player at a separate computer. This is perhaps the most common method used in games today. But it destroys the social aspect of playing a game with another human being (Cheok *et al.* , 2002). Instead of having your opponent right next to you to see and talk to, he/she may be in a different room, or even a different country. Great efforts have been expended to get around this: many games let the players communicate with "chat" messaging or even a microphone. But nothing so far has been able to truly substitute for sitting side by side around the same physical machine.

TetraTetris supports multi-user functionality without any of these methods' drawbacks. Users can all play the same game on the same board at the same time. This constitutes a major improvement in intuitive interaction.

5.2 Disadvantages of the Table

The table has some flaws, described below. However, it is to be stressed that none of these problems are limitations in the concept of touch-sensitive user interaction upon which the table is based—they are merely problems in the design or construction of the DiamondTouch table prototype.

Consequently these criticisms indicate areas where improvements would be welcome, not reasons to abandon touch-sensitive input hardware. Indeed, it is quite conceivable that these flaws will be alleviated as the technology improves. Conversely, it is not conceivable that the next-generation mouse will let multiple people move it in different directions at the same time.

We summarize the main problems that we encountered below:

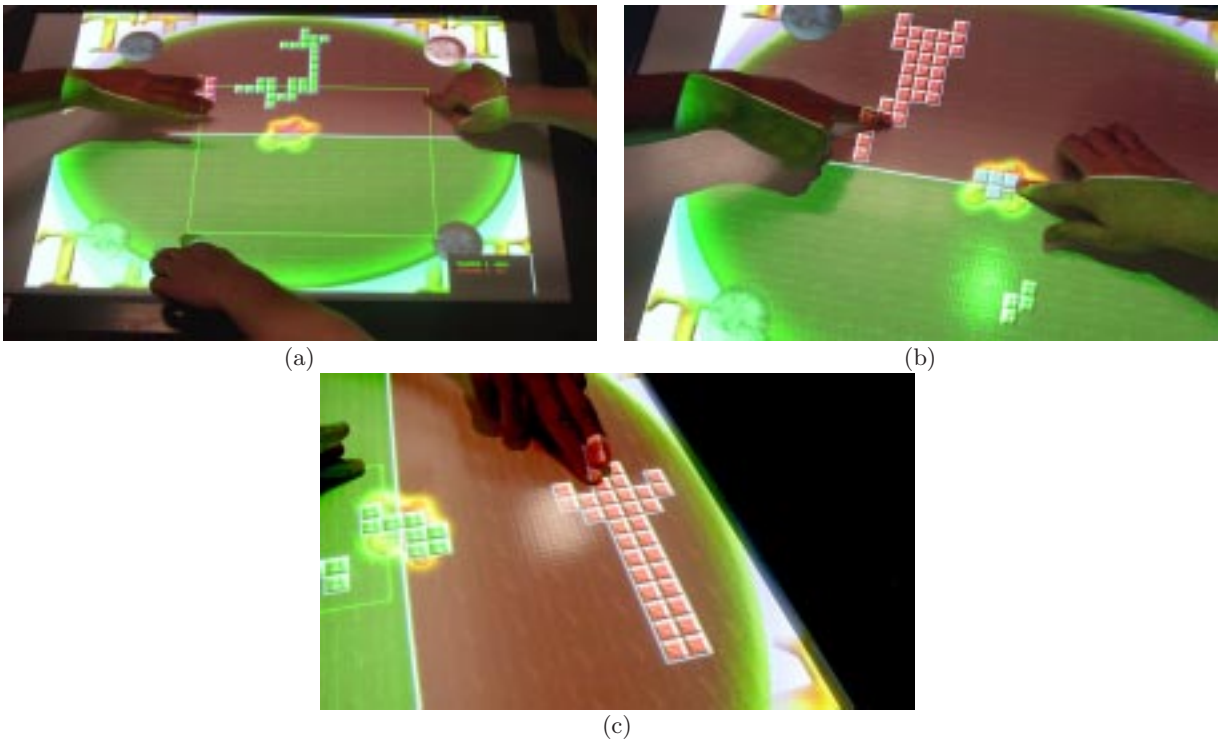


Figure 8: Figure (a) shows a game in progress; the player on the right has defined a bounding box. Figure (b) shows players grabbing TetraTetris blocks. Figure (c) shows a player dragging a symmetric block toward its destination.

Lack of precise control: Some of the reasons for this problem were described in Section 4 (finicky electronics and ambiguous input). Another is the resolution of the table: input is detected at a granularity of 160×96, which is far below the graphical output resolution of TetraTetris (1024×768). This makes it difficult to position blocks precisely—a problem which will manifest itself in just about any application for the DiamondTouch table. We expect that the table resolution will improve in future versions.

Size: The table measures about $36'' \times 24''$, and we found it too small for four people to sit comfortably around. (It works nicely for two people, and it can even be used as a coffee table, since unlike a keyboard or mouse it can take an occasional coffee spill.)

Complexity of setup: In order to be able to use our DiamondTouch table, it was necessary to set up the following additional items:

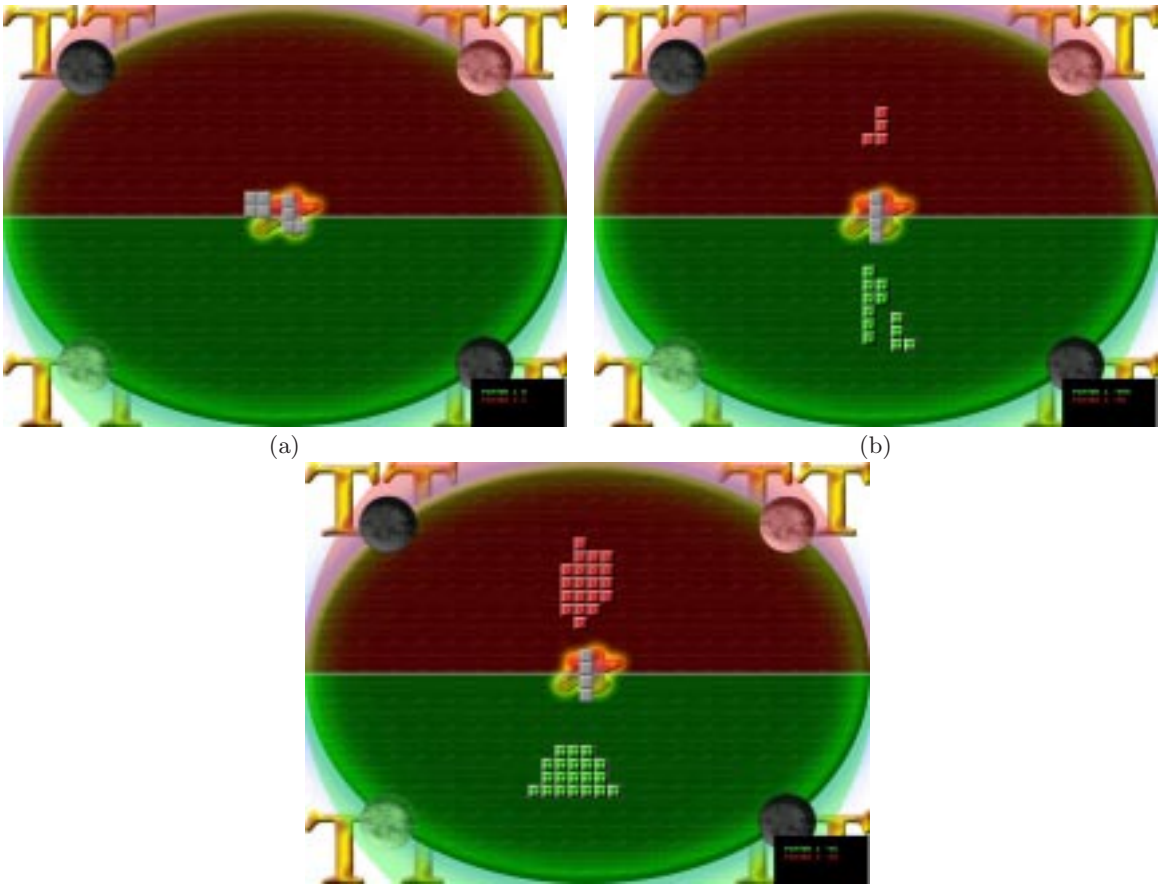
- Video projector
- Mirror
- Supports for mirror (we used rulers)
- Abundant amounts of duct tape

As a result, our table is now fairly immobile in our research lab. Transporting it to another location would require several hours of work in setting up and calibrating. By comparison, a mouse may be stuffed into a pocket and taken anywhere.

6 Related Work

In 1991, Mark Weiser published an article on his vision of "Ubiquitous Computing" (Weiser, 1991), illustrating a different paradigm of computing and HCI which pushes computers into the background and attempts to make them invisible. For example, the personal computer in its current form is notoriously poor when it comes to bodily interaction (Ishii & Ullmer, 1997). In the area of ubiquitous games there has been recent interest in ubiquitous, touch-based, and interactive games. Touch-Space (Cheok *et al.* , 2002) has a mixed-reality games space with tangible and ubiquitous aspects. Contextual virtual interaction as a part of game design is studied by Manninen (Mannien, 2002), and a special issue of *Personal and Ubiquitous Computing* is dedicated to Ubiquitous games (Bjork *et al.* , 2002).

Dietz and Leigh (Dietz & Leigh, 2001) describe the DiamondTouch table, focusing on the design of the system and its application. Brown, Buxton, and Murtagh (Brown *et al.* , 1990) show how the sur-



(a) (b) (c)

Figure 9: A sequence of screenshots from a game in progress.

face of a single input device, a tablet, can be partitioned into a number of virtual input devices. They also demonstrate properties of tablets that distinguish them from other input devices such as mice, and show how the technique can be particularly effective when implemented using a touch sensitive tablet. Sears, Kochavy, and Shneiderman (Sears *et al.* , 1990) consider touchscreen interfaces and database queries. They study three interfaces for constructing queries on alphabetic field values, including a QWERTY keyboard, an Alphabetic keyboard, and a RIDE interface.

Computer Supported Collaborative Work systems (Dewan, 1993) share some of the features of the DiamondTouch table. Similarly, Olson *et al.* (Olson *et al.* , 1990) review aspects of systems built for group work that allow real-time, concurrent editing of a single work object. LeeTiernan and Grudin (LeeTiernan & Grudin, 2001) present an experimental test of a prototype system for asynchronous distance learning. It allows viewers of audio and video to create and share text annotations that are synchronized with the multimedia. Abowd and Mynatt (Abowd & Mynatt, 2000) consider everyday computing, focused on scaling interaction with respect to time.

7 Summary

In this paper we have described one of the first non-trivial applications of the MERL DiamondTouch multi-user touch-sensitive table. The table allows up to four simultaneous users to collaborate by interacting with virtual objects. The objects can be touched, dragged, and rotated, allowing intuitive interaction with a virtual world.

The table is a particularly useful interface in situations where novice users from a variety of backgrounds need to collaborate, explore, and brain-storm around a common problem. In particular:

- The table makes it very easy for several users to interact with the same application. In comparison, it is difficult for several users to gather around a computer screen and simultaneously manipulate the mouse, keyboard, etc.
- Touching and dragging objects on the table is highly intuitive.

- Using the "bounding-box" feature, it is easy to manipulate several objects at once.

We have, however, also discovered several drawbacks with the technology:

- Moving, setting up, and calibrating the unit is tedious and time-consuming.
- Depending on the needs of the application, the table's resolution can be too low for accurate placement of objects.
- The table is extremely sensitive to the presence of other electronic devices.
- The table cannot be operated reliably if any part of the user's body is touching a computer or another user.
- The table is unable to detect multiple simultaneous touch points unambiguously. This makes rotating objects difficult.
- A complete setup is costly, requiring a computer and an LCD projector in addition to the table itself.

It is our intention to continue the study of the DiamondTouch technology by developing applications outside the game sphere. We are currently in the process of investigating the use of the table as a front-end to a graph visualization and manipulation tool. This tool is used to explore very large graphs, such as those generated from telephone call traffic, compilers, citation databases, etc.

The software is available for download from `http://www.u.arizona.edu/~slk/tt`.

Acknowledgments

The authors wish to thank Joe Marks and Kathy Ryall of MERL for supplying us with the DiamondTouch table and for their great help with all the information we needed about it. We would also like to thank John Luiten of the University of Arizona for helping us with the resources we needed to make this project a reality.

Christian Collberg was partially supported by the NSF under grant CCR-0073483. Stephen Kobourov was partially supported by the NSF under grant ACR-0222920.

Steven Kobes, Ben Smith, Stephen Trush, and Gary Yee are undergraduate students in the Computer Science program at the University of Arizona.

References

Abowd, Gregory D., & Mynatt, Elizabeth D. 2000. Charting past, present, and future research in ubiquitous computing. *ACM Transactions on Computer-Human Interaction*, **7**(1), 29–58.

Bjork, Staffan, Holopanien, Jussi, Ljunngstrand, Peter, & Mandryk, Regan. 2002. Special Issue on Ubiquitous Games. *Personal and Ubiquitous Computing*, **6**, 358–361.

Brown, Ed, Buxton, William A.S., & Murtagh, Kevin. 1990. Windows on Tablets as a Means of Achieving Virtual Input Devices. *In: INTERACT'90*. North-Holland.

Burgiel, Heide. 1996. *How to lose at TETRIS*. Tech. rept. `citeseer.nj.nec.com/34848.html`.

Cheok, Adrian D., Yang, Xubo, Ying, Zhou Z., Billinghurst, Mark, & Kato, Horokazu. 2002. Touch-Space: Mixed Reality Game Space Based on Ubiquitous, Tangible, and Social Computing. *Personal and Ubiquitous Computing*, **6**, 430–442.

Cohen, Oryx, Meyer, Shawna, & Nilsen, Erik. 1993. Studying the Movement of High-Tech. Rodentia: Pointing and Dragging. *Pages 135–136 of: Proceedings of ACM INTERCHI'93 Conference on Human Factors in Computing Systems – Adjunct Proceedings*.

Demaine, Erik D., Hohenberger, Susan, & Liben-Nowell, David. 2002 (October). *Tetris is Hard, Even to Approximate*. Tech. rept. http://arxiv.org/abs/cs.CC/0210020.

Dewan, Prasun. 1993. *A Survey of Applications of CSCW Including Some in Educational Settings*.

Dietz, P., & Leigh, D. 2001. DiamondTouch: A Multi-User Touch Technology. *In: Proceedings of the 14th annual ACM symposium on User interface software and technology*. `http://ltm.cs.uec.ac.jp/~mcashen/classes/thesis/diamondtouch.pdf`.

Ishii, Hiroshi, & Ullmer, Brygg. 1997. Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms. *Pages 234–241 of: CHI*.

LeeTiernan, S., & Grudin, J. 2001. *Fostering Engagement in Asynchronous Learning Through Collaborative Multimedia Annotation*.

Mannien, Tony. 2002. Contextual Virtual Interaction as Part of Ubiquitous Game Design and Development. *Personal and Ubiquitous Computing*, **6**, 390–406.

Olson, Judith S., Olson, Gary M., Mack, Lisbeth A., & Wellner, Pierre. 1990. Concurrent editing: the group's interface. *Pages 835–840 of: INTERACT*. North-Holland.

Sears, Andrew, Kochavy, Yoram, & Shneiderman, Ben. 1990. Touchscreen field specification for public access database queries: let your fingers do the walking. *Pages 1–7 of: Proceedings of the 1990 ACM annual conference on Cooperation*. ACM Press.

Weiser, Mark. 1991. The Computer for the 21st Century. *Scientific American*, **265**(3), 94–104 (Intl. ed. 66–75).