The Design and Implementation of Multi-player Card Games on Multi-user Interactive Tabletop Surfaces

Shwetak N. Patel, John A. Bunch, Kyle D. Forkner, Logan W. Johnson, Tiffany M. Johnson, Michael N. Rosack, and Gregory D. Abowd

College of Computing & GVU Center, Georgia Institute of Technology, 801 Atlantic Drive, Atlanta, GA 30332-0280, USA
{shwetak, bunch, dasbrute, logan, tif, michael.rosack, abowd}@cc.gatech.edu

Abstract. We present the design and implementation of a card game architecture for mulit-user interactive tabletop surfaces. Our system is built on the DiamondTouch, a touch-sensitive input surface that allows several users to interact with a program at the same time. We describe the software architecture and present Blackjack as a sample implementation using this framework.

1 Introduction and Motivation

Card games have been around for generations. From classic games like Poker and Blackjack to relatively recent ones like Magic the Gathering, card games have entertained and educated us in countless ways. Almost every card game has been digitized in some form or another; Blackjack is a good example of this. Most interactive systems for card games today are limited to the desktop where interaction is usually achieved indirectly through a mouse or keyboard. Systems that allow a more direct interaction have also been available for some time now like touch screen monitor overlays or touch screen laptops and PDAs. In addition, the latest Tablet PCs allow direct pen-based interaction. Smart technology also offers an interactive whiteboard called Smart Board [5], which employs a touch system similar to the Tablet PCs. However, these systems only allow for single user interaction. These systems could be extended so that they detect multiple inputs, but they would still lack the ability to differentiate between users and resolve multiple interaction points for a particular person. With these systems multi-user applications are limited to only sequential user interactions, making is unsuitable for many multi-player card games.

We describe our exploration of the DiamondTouch as a viable card game platform. The DiamondTouch is a touch-sensitive input surface that allows several users to interact with a program at the same time. Although current research has focused on collaborative workspace applications [1, 0, 4], we particularly focus on card gaming and show how we can create a rich gaming experience despite the loss of some physical affordances. We also present a software architecture written in Java that allows users to quickly implement a wide variety of card games on the DimondTouch, and we show Blackjack as an example.

2 System Implementation

2.1 Hardware

The DiamondTouch, developed by Mitsubishi Electric Research Laboratory (MERL), is a multi-user touch sensitive input device that allows simultaneous interaction and can identify which user is touching where [0]. The DiamondTouch accomplishes this by using an array of antennas embedded in the touch surface. Each one of these antennas transmits a unique signal. In addition, each user is touching some receiver, a 3M electrostatic mat (Figure 1b) that is placed on a chair. When a user touches the surface of the DiamondTouch, the antennas near the touch point send a small amount of current through the user's body to the receiver mat that the user is sitting on. The current system supports up to 4 users. The unique signal determines where exactly the person is touching the surface. The effective resolution with signal interpolation is about 2550 by 1550, which is about .1 mm of accuracy. However, the projected top-down image has a 1025 X 768 pixel resolution.

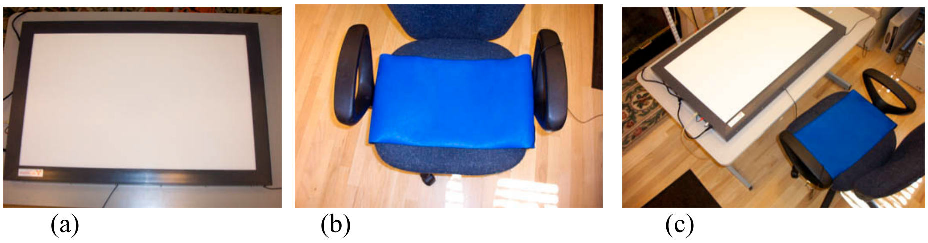

Fig. 1. a) MERL DiamondTouch. b) Conductive mat on chair c) DiamondTouch surface and mat

2.2 Software Framework

Figure 2 shows the overall high-level architecture for our tabletop card gaming system. The input layer provides an extensible programming interface for a variety of gestures common in card games such as hit, stand, or drag. All of the low-level Diamond Touch events are abstracted out into the input layer. The logical table provides hooks for implementing custom card games. Similar to the input layer, developers can create custom card games using generic archetypes. Finally, we provide a display layer for specifying how artifacts like chips and cards render or are graphically represented.

2.2.1 Generic Card Game Platform

We created *DTCards* as a multi-cardgame system, as well as an SDK for developing other card games. Pieces used for construction, interaction, and representation of a card game are largely reusable and thus are abstracted for game programmers. However, there is a great deal of variety in card games, and some aspects of any given game may not be reusable in another. Therefore our framework provides an interface to give developers the ability to extend commons archetypes and provide specialized, game-specific objects and interactions whenever necessary.

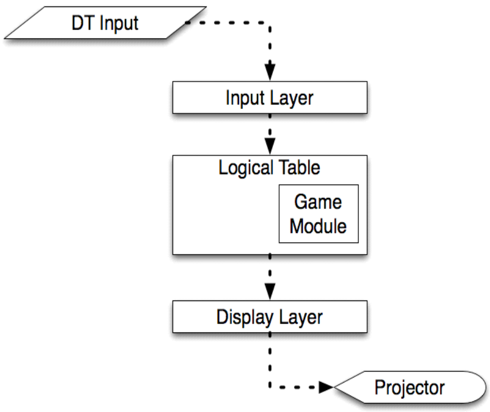

Fig. 2. High-level software architecture

2.2.2 Input and Gestures

The modularity of the *Input Layer* eases the burden of handling raw table events by the game developer. *Input Layer* reads events from the DiamondTouch and feeds them to *Gesture* objects. Each user has a copy of each *Gesture*, analyzes the event streams fed to them, and tries to recognize himself. Each user's *Gestures* compete among each other for recognition. When a *Gesture* matches a user's actions, it notifies the subscribed events. Our current implementation supports the following gestures, which were partly inspired by [6]:

1. **Hit** – a vertically motion on the tabletop
2. **Stand** – a horizontal motion
3. **Drag** – touch object and slide to another location
4. **Tap** – single quick touch
5. **Double tap** – two quick taps
6. **Private hand view** – a vertical hand is placed on the table and a message is projected on the palm of the hand so that only that player can view it

These are just a subset of potential gestures. Information flow through the system is essentially one-way. A user's gesture is picked up by the *Input Layer's* recognizer, which then sends it to the *LogicalTable*.

2.2.3 Game Module

The *LogicalTable* represents the binding (or "system") module. It ties the *Input Layer, Game module, and Display Layer* together. The *LogicalTable* contains a set of regions, which are areas on the game table that represent game objects, text strings, or special input areas. These regions are defined on the *LogicalTable* by the *Game* *module*, and when they are updated the *LogicalTable* notifies the *Display Layer* so that screen updates can be performed.

The *Display Layer* is represented to the rest of the system through the *CardTable* class. The *CardTable's* job is to receive updates from the *LogicalTable* and manage the on-screen game that the users see.

The rules for a game are supplied by a module that is represented to the rest of the system through a subclass of *Game*. This class -- in the case of our first demo, Blackjack (see Figure 3) -- manages players, game state, and game table contents. It receives players' *Gestures* from the *LogicalTable* and decides what, if anything, to do with them based on the rules of the game. This module is what a game author supplies to the *CardTable* system.

In constructing a game module, programmers must use or extend a variety of classes which comprise the *Game Kit*. This includes simple *Player objects, Cards, Suits*, and even specialized *Gestures* if need be. In the case of Blackjack, we needed a subclass of *Hand* that knew how to calculate the point value of a hand of cards in Blackjack. This in turn necessitated a subclass of *Player* which contained a *BlackjackHand* rather than a generic *Hand*. The *Game Kit* even provides an abstract *Gamepiece* class which game modules can subclass to introduce tokens, chips, or other game pieces not explicitly conceived and provided by the framework.

3 Blackjack Example

Figure 4 shows our example implementation of simple Blackjack (4a) and a more sophisticated casino version that involves betting (4b) built using the framework and toolkit described previously.

In the simple Blackjack version, the computer automatically deals two cards face up to each individual player. An indicator icon shows the player's turn. Each player can either hit by sliding a finger from the left to right on player's space. A player can stand by sliding a finger vertically, in which case the turn goes to the next user. On a bust, a message pops up and the player's hand clears. Figure 4b shows a version that includes betting. Players can drag chips from their stack and move them into the betting region very similar to casino style Blackjack.

4 Reflections and Conclusions

Despite the loss of the physical affordances of real cards, there are other advantages of an electronic tabletop card game system. Our implementation on the Diamond Touch allows gestures such as dragging and tapping for a richer gaming experience than traditional computer interfaces. In addition to the tabletop metaphor, users can directly manipulate digital artifacts by simple natural gestures on the table.

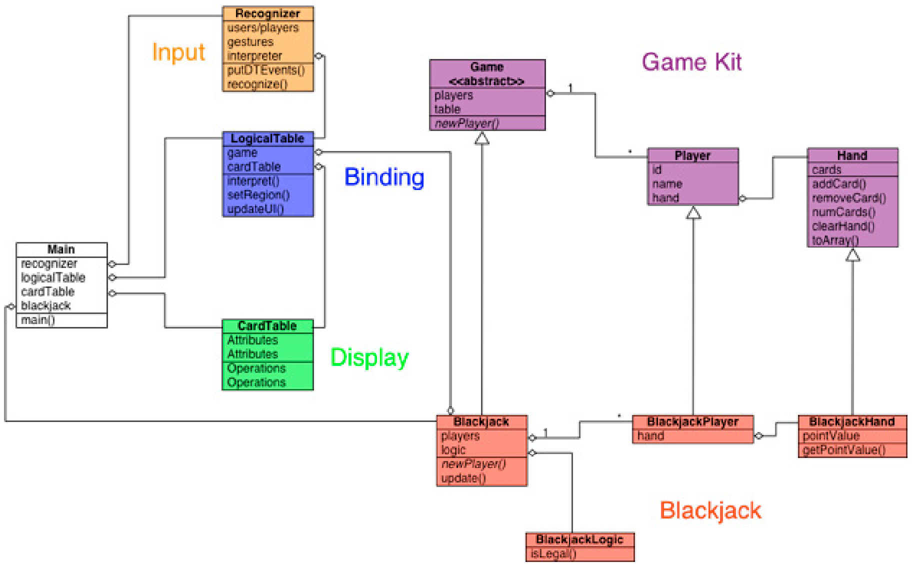

Fig. 3. Class diagram of Blackjack implementation

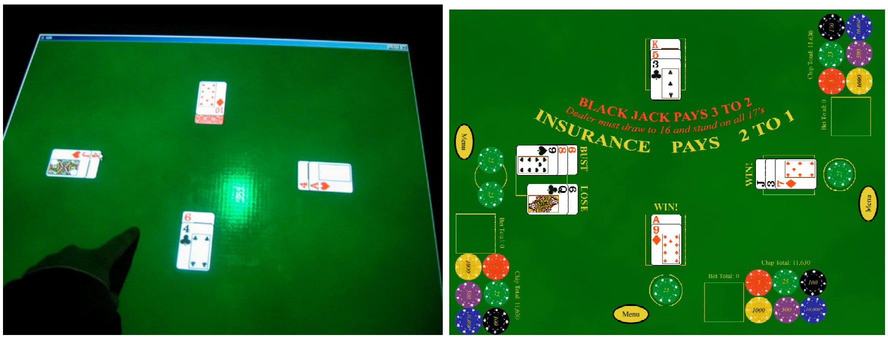

Fig. 4. Photographs of (a) simple Blackjack implementation and (b) Casino Blackjack that includes betting

The digital table can also provide novices with hints and rules as they play the game, making it an ideal learning system. Another interesting extension is the ability to remotely play others with a similar tabletop setup. For instance, a son could play with grandparents at a distance, while still retaining the tabletop metaphor. Now communities can emerge to resurrect long lost card games with this network capability. We are currently developing other multi-player games that require simultaneous interaction with the table such as Poker and Rummy.

Acknowledgements. The authors thank Mitsubishi Electronics Research Laboratory (MERL), and in particular Joe Marks, for the donation of the DiamondTouch for this research. We acknowledge that this work was done as part of an undergraduate senior design project with supervision by Bob Waters and Santosh Pande at Georgia Tech. This work is sponsored in part by National Science Foundation (ITR grant 0121661) and the industrial sponsors of the Aware Home Research Initiative at Georgia Tech.

References

1. Brodie, J. & Perry, M. *Mobile Collaboration at the Tabletop in Public Spaces*. Presented at Co-Located Tabletop Collaboration Workshop, CSCW 2002, New Orleans, November 2002.
2. Dietz, P. and Leigh, D. *DiamondTouch: a multi-user touch technology*. In proceedings of UIST 2001, pp. 219-226.
3. Rekimoto, J. and Saitoh, M. *Augmented surfaces: A spatially continuous workspace for hybrid computing environments*. In Proc. ACM CHI '99, Pittsburgh, PA, May 15--20 1999. ACM Press.
4. Shen, C., Everitt K., Ryall, K,*: UbiTable: Impromptu Face-to-Face Collaboration on Horizontal Interactive Surfaces*. In Ubicomp 2003, pages 281-288, Seattle, WA. USA, October 2003.
5. Smart Technologies Inc., *Smart Board*. http://www.smarttech.com
6. Wu, M., Ravin Balakrishnan. *Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays*. In proceeding of ACM UIST 2003 Symposium on User Interface Software & Technology. Vancouver, Canada. November 2003.