Smart Playing Cards: A Ubiquitous Computing Game

Kay Römer, Svetlana Domnitcheva

Department of Computer Science
ETH Zurich
8092 Zurich, Switzerland
{roemer,domnitch}@inf.ethz.ch

Abstract. We present the "Smart Playing Cards" application, a ubiquitous computing game that augments a classical card game with information–technological functionality by attaching RFID tags to the cards. We also mention the requirements such an application makes on a supporting software infrastructure for ubiquitous computing.

Keywords: games, ubiquitous computing, RFID, radio tags

1 Introduction

Recent technological advances allow for turning parts of our everyday environment into so–called smart environments, which augment the physical environment with useful IT functionality in an unobtrusive way, without destroying the usual "look and feel". The main challenge of ubiquitous computing [11] is to envision such unobtrusive smart environments that provide a reasonable advantage for people using it, without violating social and legal rules of our society and life.

The area of gaming looks promising with respect to ubiquitous computing, since due to the entertaining nature of the social interactions, users are willing to explore innovative metaphors, modalities, and hardware even when they are not as apparent or fluid as the designers might have hoped [8].

In contrast to developing new games around the abilities of available technology, we took the opposite approach by augmenting a classical game with IT functionality. According to our vision, users play a classical card game with the usual "look and feel" and corresponding social interactions. Additionally, they are equipped with a small information appliance (ideally of the same size as a playing card) that displays game related information (score, winner) and gives hints (cheat alarm, playing hints).

Besides exploring possible applications of ubiquitous computing, design and implementation of the Smart Playing Cards application gave us some insight into the requirements on software infrastructures that would be useful for building ubiquitous computing applications.

The remainder of this paper will present a first prototype of the Smart Playing Cards application, followed by requirements this application makes on a supporting software infrastructure.

2 Smart Playing Cards: Whist

In this section we want to give an overview of the Smart Playing Cards prototype we developed for the game of Whist. Before going into detail about the prototype, we will present the rules of the game of Whist and motivate why we chose Whist.

2.1 The Game of Whist

The classic game of Whist [14] is a plain–trick game without bidding for four players in two fixed partnerships ("teams") using a standard 52 card pack.

At first, all the cards are dealt out so that each player has 13 cards, the last card indicates the trump color. The game then starts with the player to the right of the dealer laying down any card. The game continues clockwise with each of the players playing a card to the trick. They have to follow the suit if possible, otherwise any card is allowed. The trick is won by the highest trump or by the highest card of the suit led if there is no trump. The winner of a trick leads to the next trick. The team with the most tricks won, wins the game.

We chose Whist for our prototype implementation for two reasons. First of all, the RFID system we use to detect the cards on the table can only reliably detect a limited number of tags at the same time (about 12). In Whist there are no more than four cards on the table at any time. Additionally, Whist does not depend on spoken announcements like Skat. Furthermore, Whist allows us to implement a rich set of features in our information appliance, such as score counting, determining the winner, cheat alarm, and hints for beginners.

2.2 Prototype Description

The hardware setup of the prototype is depicted in Figure 1. It consists of a Philips I–Code Radio Frequency Identification (RFID) [15] system connected to a desktop PC, a set of PDAs, and a standard 52 card deck, where each card is equipped with an RFID tag (in form of an adhesive sticker). Each tag holds a unique ID, which identifies the card it is attached to.

The RFID system consists of a reader device and an antenna, which is mounted underneath a table. The reader device powers the antenna in order to generate an electromagnetic field that provides the tags with power via electromagnetic induction. Furthermore, the reader device implements the transceiver for communication with the tags. A PC running the RFID driver and application software is connected to the reader by a serial connection.

A large flat panel display connected to the PC displays game information common to all players, for example the current score. Each of the players can additionally use a PDA (in our case a Compaq iPAQ equipped with Cisco Aironet WLAN) to obtain private information such as a rating of the current move. The PDAs talk to the Smart Playing Cards application executing on the PC using a wireless link.

The software setup consists of the main application executing on the PC and an additional application executing on the PDAs. Both are implemented in Java. The PC

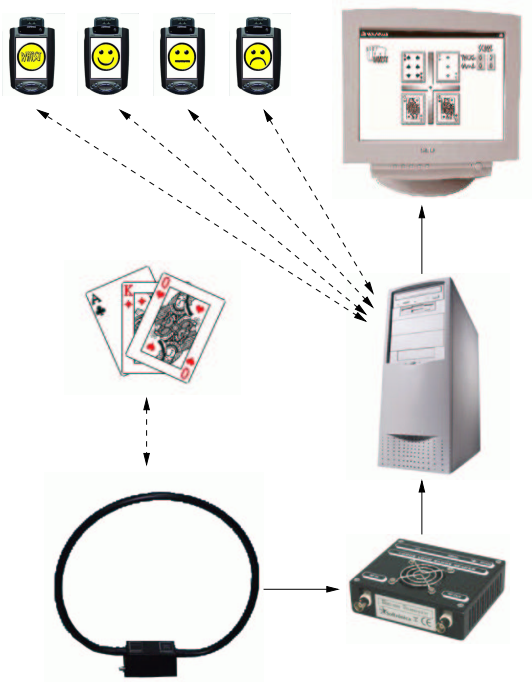

Fig. 1. Smart Playing Cards prototype: system architecture

application maintains the current game state using the RFID system. The PDA application first asks the player for an identification (numbers 1 to 4 according to the players position) and logs into the main application. Afterward a rating of each of the players moves is displayed.

Figure 2 shows two screen shots of the display in two different game situations. The game starts with dealing out the cards on the table four at a time (one for each player). The players have to take up their card before dealing the next round of cards. Upon dealing the last card, which indicates trump color, the application displays the trump color in the middle of the cross (2 in figure 2). Note that the Smart Playing Cards application now knows the cards each player got.

Now the Smart Playing Cards application indicates which player has to lay down the first card (3 in figure 2). Then each player plays a card to the trick, which is automatically displayed by the user interface. If a player does not follow the suit although he could, a cheat alarm is displayed (4 in figure 2), asking the player to correct the mistake. Upon completion of one round, the winner of the trick is determined and the trick count is increased in the upper right corner of the user interface (1 in figure 2). The winner of the trick is then indicated and waited upon for playing out the next card. Upon completion of the game, the game count of the winning team is increased in the upper right corner of the user interface (1 in figure 2).

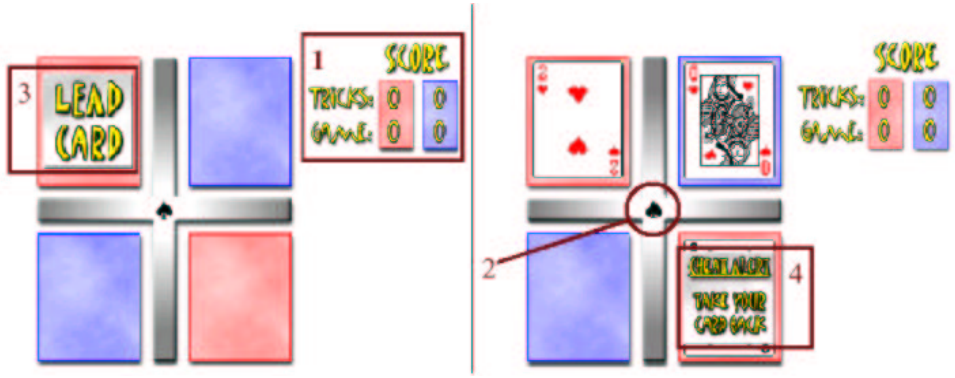

Fig. 2. Smart Playing Cards prototype: two game situations

As indicated in figure 1, the display of the PDA shows a happy, sad, or indifferent smiley as an indication of the quality of the player's move. The rating is implemented by a feedback module using a set of simple heuristics based on our game experience. This module takes into account the position of the player in the current trick (first through fourth), the actual cards already on the table, and the cards remaining in the game. Using this information, a subset of the players cards is computed containing cards which are considered good to play. Depending on whether or not the played card is contained in this subset, an according smiley is displayed.

More formally, the operation of the feedback module can be described by a decision tree. Figure 3 shows part of thistree. It consists of filters (squares) and branches (diamonds). Evaluation starts with a set G of "good" cards equal to the remaining cards of the player. Each filter modifies this set in some way. Branches evaluate some condition. The following list explains the filters and branches in Figure 3 in more detail:

- *last card?* Does only one card remain in G?
- *pos 1?* Is the player in the lead position?
- *any cards?* Is G non-empty?
- *any trumps?* Are there any trumps cards left in the game?
- *remove highest cards.* Remove the highest cards from G.
- *reset to lowest non-trumps.* Since G is empty at this point, set G to the lowest non-trump cards of the player.
- *remove bad suit.* Remove the suit from G the player played last time the trick was lost due to an opponent playing a trump.

The pairs of smileys in Figure 3 indicate what should be displayed on the PDA. The left hand side smiley is displayed if the played card is in G. Otherwise the right hand side smiley is displayed. Figure 4 shows the prototype system in action.

2.3 Further Ideas

On the hardware side we intend to replace the desktop PC with a small embedded computer that runs the RFID driver software and main Smart Playing Cards application.

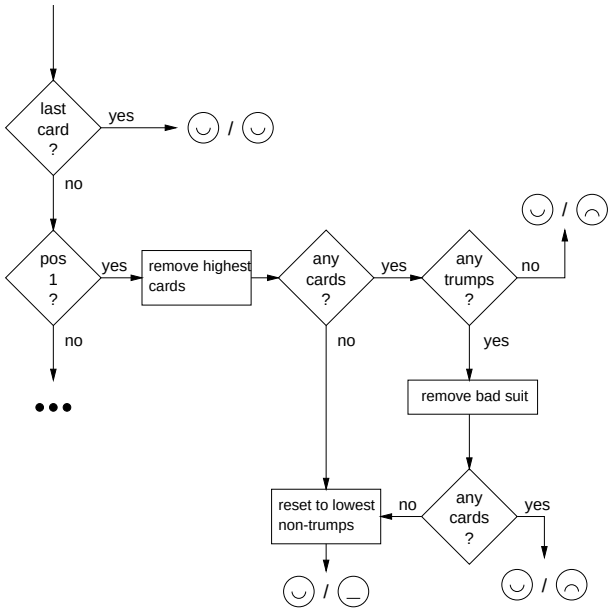

Fig. 3. Cutout of the Feedback Decision Tree: if the card just played is in the calculated set then the first smiley is displayed, otherwise the second one.

In the Smart-Its project [13] we develop such small-scale devices capable of wireless communication via Bluetooth. Note that the used iPAQ PDAs also support Bluetooth.

Since compared to playing cards, current PDAs are somewhat clumsy due to their form factor (see figure 4), we decided for a separate common large display, thus making the use of the PDA optional. However, if the PDA can be replaced by a dedicated information appliance with a more appropriate form factor, all information could be displayed on this appliance and a separate display is no longer needed.

On the software side we intend to improve the playing hints. The current version of the prototype implements a simple rating of the players moves. For learning the game it would be helpful if the system would additionally suggest a set of good moves. Note that the decision tree based approach sketched above does already provide the information (i.e., the set of “good” cards to play) necessary to implement this feature. On the other hand, the decision tree based approach is clearly only a proof-of-concept, which was chosen for its simplicity. Future versions need improvements in this respect.

Another important area for improvements is supporting players in learning and remembering the rules of the game. For Whist this might not be an issue, since the rules of the game are reasonably simple. However, there are many games with a huge set of rather complicated rules. The supporting application could for example indicate the actions still possible in the current game situation.

Fig. 4. Smart Playing Cards in action.

2.4 User Experiences

Due to the early version of the prototype we did not conduct a broad study to gain experience from users. However, we already demonstrated the prototype to some technical and non-technical people. During those demonstrations, we just started to play the game without explaining the technical setting at first. The first reaction was always a great surprise on the part of the spectators, since it is not obvious how the actions on the display are technically linked to the physical game play.

Some of the spectators also played with the system. Although many of them did hardly or not at all know the rules of Whist, they quickly learned how to play the game by exploiting the cheat alarm, which turned out to be helpful in teaching players the rules in a trial and error fashion.

Players did not like to be forced by the system to play the game in a way they are not used to. For example, forcing the players to take off the cards from the table before dealing out the next round of cards (see section 2.1) already is an annoyance to some people. A similar problem was caused by a delay (due to “flickering”, see section 4) of about one second between removing cards from the table and the display reflecting the change, which confused players a lot.

Our observations led us to the conclusion that people seem to basically like the idea of augmented everyday objects in this special setting. However, when making artifacts smart without visibly changing them, people expect to find the exact behavior of the “dumb” artifact also in the smart version. Already very subtle changes in the behavior known from the “dumb” artifact can cause a lot of confusion. On the other hand it seems

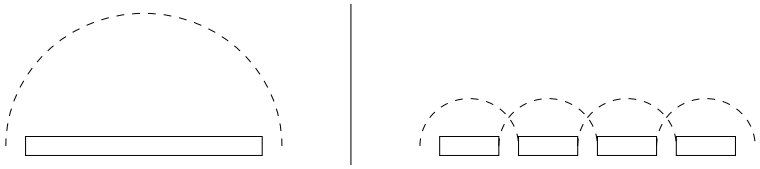

Fig. 5. Detection range of one large antenna vs. an array of small antennas

possible to introduce new functionality in the smart artifact without confusing people, as long as this does not conflict with the classical behavior of the artifact.

A further lesson we learned during demonstrations is that people are very creative in using the functionality of a system in unforeseen ways. Although the cheat alarm was not intended for this purpose, players used it as a help for learning the rules of the game.

Note that the statements in this section are rather preliminary, since we did not perform a true user survey. For this we want to wait until the more elaborate features (playing hints, rule teacher) are fully implemented, which will provide a real advantage of Smart Playing Cards over the classical game.

3 Technical Issues

As mentioned above, we use the Philips I–Code RFID system with an antenna of a size of about 70x50cm. The detection range of such an antenna is about a sphere with a diameter of the length of the antenna as depicted in the left hand side of figure 5. This gives us a reasonable area on the table where cards are detected, but players have to take care to keep the cards in their hands out of the detection range. Therefore we would prefer a large but flat detection area, which can be achieved with an array of smaller antennas as shown in the right hand side of figure 5. Experiments showed that overlapping electro-magnetic fields of multiple readers do not cause serious problems. Tags located in overlapping regions are detected by both readers, which doesn't harm.

In order to further reduce ambiguities with "almost played" cards, a proximity sensor could be used. Quantum research [12] has developed capacitive proximity sensors which can detect the presence of a hand at a distance of up to 5cm.

A different problem arises when placing two or more tags exactly on top of each other. The RFID system we use is then no longer able to detect any of the tags. Experiments with our prototype system showed that this happens sometimes if one does not take care when placing the cards on the table. A possible solution to this problem is to place two or more tags randomly on each card as depicted in figure 6, so that it is very unlikely if not impossible to place cards on the table so that all the tags on one card are "shadowed" by other tags.

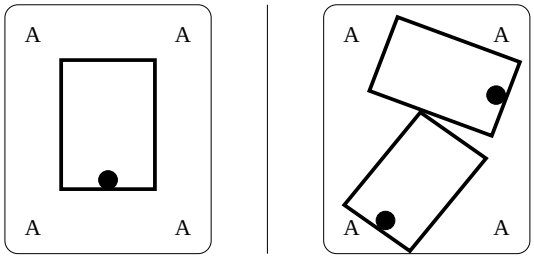

Fig. 6. Single tag vs. randomly placed multiple tags on one playing card

4 Infrastructure Support

Although we implemented the game prototype from scratch without using a software infrastructure, it quickly became clear during development that there are several reasonably complex tasks that will likely show up in other applications as well. Therefore the developer of RFID-based "smart applications" should be supported by a software infrastructure to handle these tasks. In this section we want to point out some of these tasks.

First of all, *event based programming* is an adequate approach to realize the Smart Playing Cards and other ubiquitous computing applications that are based on detecting real–world events in the physical environment (e.g., a playing card has been put on the table / has been removed from the table) as pointed out in [4].

In order to support distributed applications like the Smart Playing Cards application, where a PDA is connected to the RFID reader via wireless short range radio communication, the infrastructure should support *ad hoc networking* and *distributed delivery of events* from event generating entities (e.g., the RFID reader) to event consuming entities (e.g., the Smart Playing Cards application running on a PDA).

Since players may want to leave the table for a short time while taking the PDA along without stopping the game, the infrastructure should provide support for *intermittent disconnects*, so that the PDA shows what happened in between, upon return to the table.

A natural way to handle the presence and absence of playing cards in an event–based programming model is to generate *entry* and *exit* events for each card. However, the RFID reader can only periodically scan for tags and return a list of IDs of detected tags. Therefore, scan lists have to be converted to corresponding entry and exit events. While this seems a simple task at first, it is complicated by the fact that typically the RFID reader does not detect all present tags in each scan, an effect that is at least partially due to the anti-collision algorithm, which enables the reader to distinguish and detect multiple tags at a time [10]. Even without changing the physical setting, the list of detected tags is typically changing with each scan. To handle this problem, the infrastructure should provide some means of *event filtering*, enabling the programmer to remove all leave events followed by an entry event for the same card within a certain small amount of time, thus avoiding "flickering". Generation of entry and exit events becomes even more complicated with the introduction of an array of antennas as pointed

out in section 3. In this case, duplicate detections of the same card by different RFID readers have to be removed and causal ordering of detections by different readers has to be ensured.

On the application level, we need infrastructure support for *composite event detection* in order to detect certain game situations, e.g., the completion of one round of the game when four cards are on the table. This task is complicated by the fact that a player is allowed to change his mind and take back a card he almost put on the table, replacing it by a different card (Note that "almost played" cards might already be detected slightly *above* the table).

Although there are many event distribution services such as [2] and [9], none of them provides appropriate support for ad hoc networks, and intermittent disconnects. Likewise, there are systems for composite event detection [6], but the detection languages they provide are too simple to accomplish the tasks pointed out in this section. We are therefore working on an infrastructure for ubiquitous computing applications that supports, among other things, all the requirements pointed out above [7].

5 Related Work

Many of the documented ubiquitous computing gaming projects have developed new games ideas based on the possibilities of available technology. Examples include Pirates! [1], which uses location and proximity as new game elements, and the MIND–WARPING game system [8], which explores aspects of wearable computing and augmented reality.

On the other hand, our approach is to augment or transform existing games according to the abilities of available technology, as exemplified by Smart Playing Cards. Another example of this genre is PingPongPlus [3], a ping pong table equipped with ball location and overhead projection systems.

Little focus has been put so far on the distributed computing software infrastructure needed for development and deployment of ubiquitous computing games, although ubiquitous computing games present some interesting challenges for such infrastructure as pointed out in section 4. The WARPING system [8] provides an infrastructure for developing personal, intelligent, networked games, but does not focus on distributed computing issues. MEX [5] aims at providing an infrastructure for wearable computing, but does not address many of the requirements pointed out in section 4.

6 Conclusion and Outlook

We presented "Smart Playing Cards", a ubiquitous computing game which augments the classical card game Whist with an unobtrusive smart environment providing functionality like score counting, winner determination, cheating alarm, and playing hints while retaining the look and feel and social interactions of the classical game. We also discussed some technical issues related to the detection of cards using an RFID system, and pointed out requirements on supporting software infrastructure.

Further research will focus on new functionality of the smart gaming environment and on studying whether our approach can be generalized to other card games and classical games. However, the major topic of our research is the development of a software infrastructure for ubiquitous computing applications in general.

7 Acknowledgments

We would like to thank Friedemann Mattern for the initial idea, Vlad Coroama for initial discussions about Smart Playing Cards, as well as Philip Graf and Martin Hinz for help with implementing first prototypes of the system.

References

1. S. Björk, J. Falk, R. Hansson, and P. Ljungstrand. Pirates! Using the Physical World as a Game Board. In *IFIP Conference on Human–Computer Interaction (Interact 01)*, Tokyo, Japan, July 2001.
2. A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Achieving Scalability and Expressiveness in an Internet-Scale Event Notification Service. In *Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, Portland, OR, July 2000.
3. H. Ishii, C. Wisneski, J. Orbanes, B. Chun, and J. Paradiso. PingPongPlus: Design of an Athletic-Tangible Interface for Computer-Supported Cooperative Play. In *ACM Conference on Human Factors in Computing Systems (CHI 99)*, Pittsburgh, PA, May 1999.
4. M. Langheinrich, F. Mattern, K. Römer, and H. Vogt. First Steps Towards an Event–Based Infrastructure for Smart Things. In *Ubiquitous Computing Workshop, PACT 2000*, Philadelphia, PA, October 2000. www.inf.ethz.ch/vs/publ/papers/firststeps.pdf.
5. J. Lehikoinen, J. Holopainen, M. Salimaa, and A. Aldro-Vandi. MEX: A Distributed Software Architecture for Wearable Computers. In *Third International Symposium on Wearable Computers (ISWC 99)*, Victoria, Canada, June 1999.
6. M. Mansouri-Samani and M. Sloman. GEM – A Generalised Event Monitoring Language for Distributed Systems. *IEE/IOP/BCS Distributed Systems Engineering Journal*, 4(25), February 1997.
7. K. Römer and T. Schoch. Infrastructure Concepts for Tag-Based Ubiquitous Computing Applications. In *Workshop on Concepts and Models for Ubiquitous Computing, Ubicomp 2002*, Goteborg, Sweden, September 2002.
8. T. Starner, B. Leibe, B. Singletary, and J. Pair. MIND–WARPING: Towards Creating a Compelling Collaborative Augmented Reality Game. In *Intelligent User Interfaces (IUI 2000)*, New Orleans, LA, January 2000.
9. P. Sutton, R. Arkins, and B. Segall. Supporting Disconnectedness – Transparent Information Delivery for Mobile and Invisible Computing. In *IEEE International Symposium on Cluster Computing and the Grid (CCGrid 01)*, Brisbane, Australia, May 2001.
10. H. Vogt. Efficient Object Identification with Passive RFID Tags. In *Pervasive 2002*, pages 98–113, Zurich, Switzerland, August 2002.
11. M. D. Weiser. The Computer for the 21st Century. *Scientific American*, pages 94–104, September 1991.
12. Quantum Research. www.qprox.com.
13. Smart-Its Project. www.smart-its.org.
14. The Game of Whist. www.pagat.com/whist/whist.html.
15. The Philips I–Code System. www-us2.semiconductors.philips.com/identification/-products/icode.