Smart Playing Cards
A Ubiquitous Computing Game

Kay Römer

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

Abstract. Recent technological advances allow for turning parts of our everyday environment into so–called smart environments. In this paper we present the "Smart Playing Cards" application, a ubiquitous computing game that augments a classical card game with information–technological functionality, in contrast to developing new games around the abilities of available technology. Furthermore, we present 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 information–technological functionality in an unobtrusive way, without destroying the usual "look and feel". The main challenge of ubiquitous computing [13] 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 [10].

In contrast to developing new games around the abilities of available technology, we took the opposite approach by augmenting a classical game with information–technological 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, designing and implementing 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 [1] 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. 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 consists of a Philips I–Code Radio Frequency Identification (RFID) [2] system connected to a desktop PC and a standard 52 card deck, where each card is equipped with a unique RFID tag (in form of an adhesive sticker). The RFID system is used to bridge the physical and virtual worlds [12]. Each tag holds a unique ID, which is used to identify the card to which it is attached.

The RFID system consists of an antenna, which is mounted underneath a table, and a reader device. The antenna is connected to the reader device that 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 is connected to the reader by a serial connection and runs the RFID driver software.

In our early prototype, the PC also runs the Smart Playing Cards application and displays the user interface. In a later version, the PC will be replaced by a small embedded PC that runs the RFID driver software, and a PDA or even a dedicated information

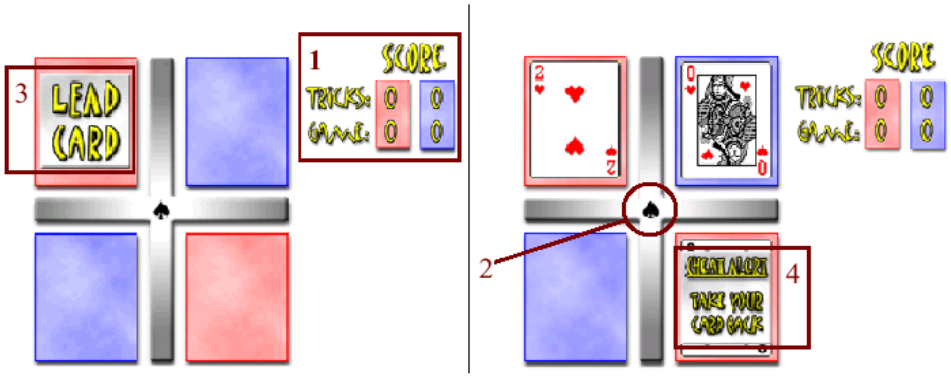

Fig. 1. Smart Playing Cards prototype: two game situations

appliance that runs the Smart Playing Cards application and communicates with the embedded PC via wireless short range radio.

Figure 1 shows two screen shot of the Smart Playing Cards prototype in two different game situations. The prototype has been implemented using Java. 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 1). 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 1). 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 1), asking the player to correct the mistake. Upon completion of one round, the winner of the trick is determined and the according trick count is increased in the upper right corner of the user interface (1 in figure 1). 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 1).

2.3 Further Ideas

Future versions of the prototype will additionally provide playing hints for beginners. The simplest version is to assess the players last move by displaying a happy or sad smiley. A more elaborate version would point out certain game situations to the player and make suggestions. An interesting way of doing this would be to augment each playing card with a so-called virtual counterpart as pointed out in [6]. A virtual counter part essentially is a virtual representation of a playing card, giving each card a kind of personality. Over a series of games each counter part would remember the tricks the according card was involved in, together with an assessment of the trick. Thus, when playing a card to a trick, the according counterpart is able to “speak” to the player like

this: “Oh no, I don’t want to join ugly spade ace and vain diamonds queen!”, or “Yes, I like that charming clubs king!” using its past experience. One might even think of support for enabling players to link persons or things and characteristics to the cards in some way. This way playing hints are mapped to relationships between personalities. We think this might help people improve their playing skills, since thinking in relationships between people and things is very common to human beings anyway.

A second 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.

2.4 User Experiences

Due to the early version of the prototype we did not conduct a broad study to gain experiences 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 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 certain unusual (to them) way. 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 ubiquitous computing 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 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 very preliminary, since we did not perform a real user survey. For this we will wait until the more elaborate features (playing hints, rule teacher) are implemented, which will provide a real advantage of Smart Playing Cards over the classical game.

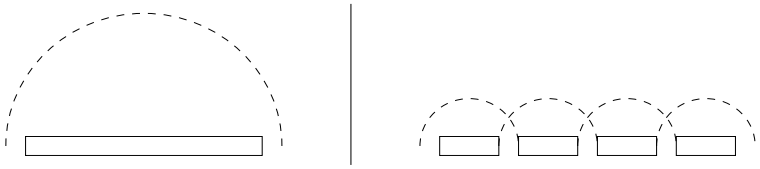

Fig. 2. Detection range of one large antenna vs. an array of small antennas

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 2. 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 2. 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.

A further advantage of an array of antennas is the potential ability to detect more tags at the same time compared to a single antenna, in case the tags are equally distributed over the detection area of the antennas. While such an array is an easy solution from a hardware point of view, it provides us with a challenge on the software side, because now we have to combine detection results from multiple antennas. We will further discuss this issue in section 4.

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 3, such that it is very unlikely if not impossible to place cards on the table such that all the tags on one card are “shadowed” by other tags.

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 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 [6].

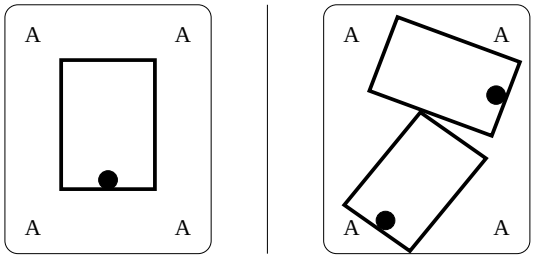

Fig. 3. Single tag vs. randomly placed multiple tags on one playing card

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*, such 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 probably due to the anti-collision algorithm, which enables the reader to distinguish and detect multiple tags at a time. Even without changing the physical setting, the list of detected tags is constantly changing with each scan. To handle this problem, the infrastructure should provide some means of *composite event filtering*, enabling the programmer to remove all leave events followed by an enter 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. First of all, we have to remove duplicate detections of the same card by different RFID readers, again using composite event filtering. A more serious problem arises when we have to determine whether a leave event generated by one reader happened before an enter event for the same card generated by a different reader. For solving this problem, the infrastructure has to provide a means of *physical time synchronization* as discussed in [9]. But time synchronization is still not sufficient to solve this problem, since due to the distributedness of the antennas, events might not arrive in temporal order. Consider the case where an enter event is received from one reader. How can the application decide whether there is an earlier leave event for the same card generated by a different reader, in which case both the enter and leave event have to be deleted to avoid flickering as described above? Just waiting long enough might not be sufficient due to the unbounded delays resulting

from temporary disconnections (e.g., caused by the PDA going offline). One solution would be the support for *temporal delivery order of events* by the infrastructure (i.e., if there is an earlier enter event, it has to be delivered before the later leave event).

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 *above* the table).

Although there are many event distribution services such as [4] and [11], none of them provides appropriate support for ad hoc networks, intermittent disconnects, and temporal delivery order. Likewise, there are systems for composite event detection [8], 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 all the requirements pointed out above. We are especially focusing on developing a composite event handling system that is capable of handling the complex tasks discussed above. The system will be covered by a later publication.

5 Related Work

Many of the documented ubiquitous computing gaming projects take a *technology driven approach*: new games are designed around the abilities of available technology. Examples include Pirates! [3], which uses location and proximity as new game elements, and the MIND–WARPING game system [10], which explores aspects of wearable computing and augmented reality.

On the other hand, the *game driven approach* augments or transforms existing games according to the abilities of available technology, which is the approach we took for Smart Playing Cards. Another example of this genre is PingPongPlus [5], a ping pong table equipped with ball location and overhead projection systems.

Little focus has been put 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 [10] provides an infrastructure for developing personal, intelligent, networked games, but does not focus on distributed computing issues. MEX [7] 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 the ubiquitous computing game "Smart Playing Cards", 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 games in particular and ubiquitous computing applications in general.

7 Acknowledgments

We would like to thank Vlad Coroama and Svetlana Domnitcheva for pointing out Whist as a suitable game for Smart Playing Cards, and Philip Graf for figure 1 and help with implementing first prototypes of the system.

References

1. The Game of Whist. www.pagat.com/whist/whist.html.
2. The Philips I–Code System. www-us2.semiconductors.philips.com/identification/-products/icode.
3. 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.
4. 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.
5. 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.
6. 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.
7. 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.
8. 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.
9. K. Römer. Time Synchronization in Ad Hoc Networks. In *ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 01)*, Long Beach, CA, October 2001.
10. 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.
11. 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.
12. R. Want, K. Fishkin, A. Gujar, and B. Harrison. Bridiging Physical and Virtual Worlds with Electronic Tags. In *ACM Conference on Human Factors in Computing Systems (CHI 99)*, Pittsburgh, PA, May 1999.
13. M. D. Weiser. The Computer for the 21st Century. *Scientific American*, pages 94–104, September 1991.