ART: Augmented Reality Table for Interactive Trading Card Game

Albert H. T. Lam, Kevin C. H. Chow, Edward H. H. Yau, and Michael R. Lyu *
Department of Computer Science and Engineering
The Chinese University of Hong Kong

Abstract

Real world games and computer games have their own distinct strengths. Augmented reality allows us to combine both strengths, improve existing game styles, and produce new games. In this paper, we present the *Augmented Reality Table (ART)*, a prototype platform employing augmented reality technology to provide a virtual table for playing trading card games. ART consists of an overhead head camera to perceive card inputs and player commands, and a plasma TV placing horizontally to act as the game table, display 3D models, and generate sound for the game play. The idea is to provide an interactive environment and input method for players, much the same as the original game, but with 3D graphics and sound enhancement to visualize and realize the game while maintaining complicated game rules at the same time. We illustrate ART as a system engaging augmented reality techniques to enable card games with more interactive and attractive user experience.

CR Categories: H.5.2 [Information Interfaces and Presentation]: User Interfaces—Input devices and strategies, Interaction styles K.8.0 [Personal Computing]: General—Game

Keywords: Augmented Reality, Computer Entertainment, Card Game Environment

1 Introduction

Mixed Reality combines the content of the real world with virtual imagination. In 1994, Milgram and Kishino characterized *Mixed Reality* interfaces on his Reality-Virtuality Continuum as shown in Figure 1 [Milgram and Kishino 1994].

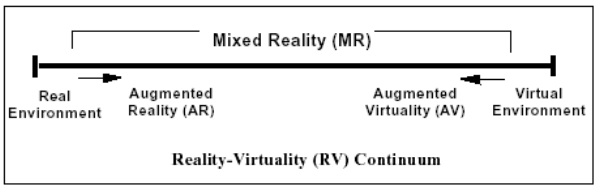


Figure 1: Reality-Virtuality Continuum.

Augmented Reality (AR) is a subset of Mixed Reality, where virtual contents are overlaid into real objects of the world. Extending the concept of AR, it includes virtual graphics and audio. An Augmented Reality system supplements the real world with virtual objects, such that computer-generated contents are added to the real world interactively, and in real time. These properties provide fascinating improvements to real world games, making them more enjoyable and attractive. One of the suitable games are *trading card games*, which are a kind of card games, such as *"Magic the Gathering"* and *"YU-GI-OH"*. Traditional trading card games are played on a table. Players draw cards alternately and put cards on the table to summon monsters, cast spells and fight against each other. They compete for a single basic resource, which may be the magic power or the hit point of players.

Nevertheless, *all these scenarios and actions are based on players' own imagination.* Players cannot really summon a monster or cast a spell. In addition, the increasing complexity of trading card games makes them more strategic and attractive, but *requires more complicated calculations.* Even an experienced player may be confused by the complicated rules. This greatly reduces the joy of the game. Therefore, we need to develop a system where players can play trading card games with concentration. On top of it, this system should also provide sound and audio enhancement in the game play. Moreover, complicated rule calculations can also be carried out by the system.

This paper is organized as follows. In section 2, we review some recent applications of AR techniques to computer games. Section 3 shows our hardware setup and system architecture of ART. Sections 4-6 discuss our implementation and technique use in input perception, rule-based game engine, and visual and audio enhancement, respectively. We conclude the paper in section 7.

2 Related Work

Augmented reality is becoming popular in digital entertainment area [Lyu et al. 2005]. It gives users immersed feeling into the combined virtual and real scenes, which is a very attractive technique for game development. Many recent papers describe some applications of AR in computer games.

In *Augmented Reality Checkers* [Balcisoy et al. 2000], a computer generated virtual players playing checkers on a real game board, acting the same way as real players. In *Augmented Reality Mah-Jongg* [Szalavhri et al. 1998], players play and grab virtual tiles from their private panel with a real pen. *ARQuake* [Thomas et al. 2000] is an outdoor/indoor augmented reality first-person shooting game modified from the desktop game Quake, where players can see and shoot monsters from their personal view through head-mounted devices. Another system, the *MIND-WARPING System* [Starner et al. 2000], allows two groups of player to fight each others, where the "magician" group work at a workbench to set monsters to beat the "fighter" group, while "fighter" group see these virtual monsters overlaid with real world and use gesture to destroy the monsters.

Nevertheless, much of these applications use markers to identify objects or the environment. One problem of using marker is that tracking usually fails when occlusion of markers occurs, resulting in recognition errors and even exposing markers to players. *TARBoard* [Lee and Woo 2005] suggests placing markers at the back of

*e-mail:{htlam, chchow2, edyau, lyu}@cse.cuhk.edu.hk

VRCIA 2006, Hong Kong, 14–17 June 2006.
© 2006 ACM 1-59593-324-7/06/0006 $5.00

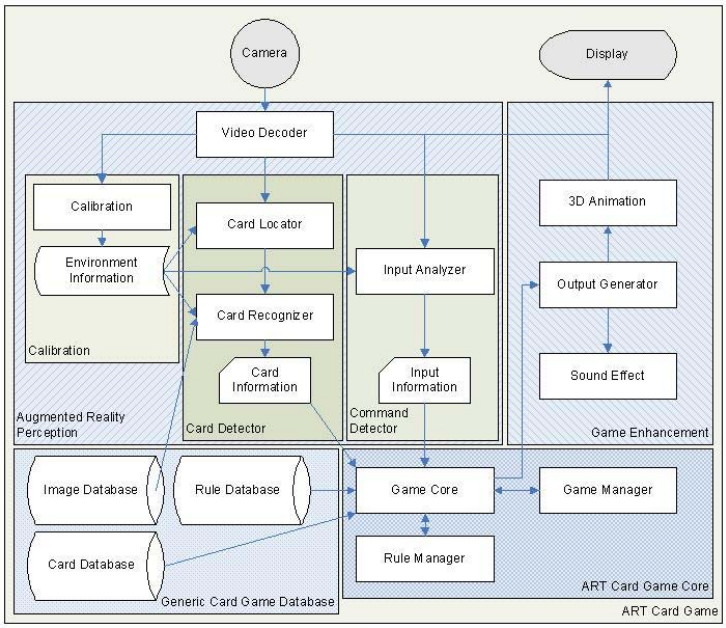


Figure 2: System Architecture.

the cards, so that they are invisible to players, but visible to camera under the transparent glass table. However then players can guess what the card is by watching the back of the card. In fact the back face is supposed to be the same for all the cards, so as to prevent players from knowing what cards they will draw in the face-down deck and what cards their opponents are holding.

3 System Design

3.1 Hardware Setup

Augmented Reality Table (ART) is a platform using augmented reality technology to provide a virtual table for players. It consists of a *computer*, a *plasma monitor* and a *camera*. The *Overhead Camera* captures events in the arena, acting as the *only* input device of the system. The *Plasma Monitor* is placed horizontally to act as a table for players to play trading card games naturally and traditionally on the virtual mat. It also displays the virtual environment to players, including visual and sound enhancement. The *Computer* processes captured image stream, recognizes valid inputs, and produces corresponding outputs according to the game rules. Figure 3 shows the hardware setup of ART.

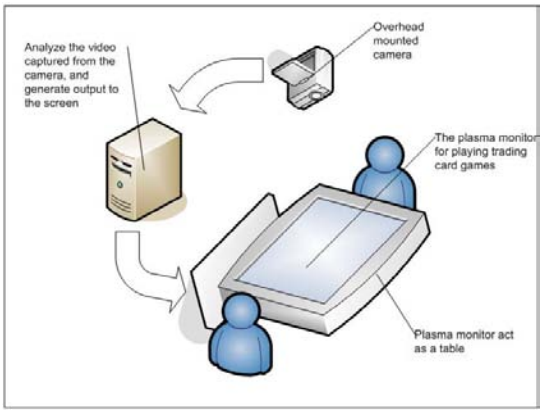


Figure 3: Hardware setup.

3.2 System Architecture

The main purpose of the ART system is to read video from the camera and output virtual scenes to the display. The system is divided into four main modules as shown in Figure 2. They are *Perception Module*, *ART Card Game Core*, *Game Enhancement Module*, and *Generic Card Game Database*.

Perception Module reads the raw video from the camera. *Calibration* calculates several coefficients of the environment and updates Environment Information, which provides essential parameters for *Card Detector* and *Command Detector*. *Card Detector* locates and recognizes the unique input card. *Input Detector* detects player inputs such as pressing the command buttons. The extracted card information and input information are then sent to *ART Card Game Core*.

ART Card Game Core reads the card information and input information from *Perception Module*. The information provides the current game status. It then analyzes this information, and checks the game rule database. Finally it outputs the game information to *Game Enhancement Module* for output.

Game Enhancement Module reads the game information from *ART Card Game Core*, and generates display and sound effects.

Generic Card Game Database allows user to plug-in different kind of trading card games. *Image Database* provides the basic card patterns and features for card recognition in Perception module. *Card Database* contains all card information that appears on the card, such as names and effects. *Rule Database* contains basic game rules and relationships between cards.

4 Perception

4.1 Calibration

To simplify our implementation providing that the environment is stable, we omit color calibration. Owing to the fixed environmental setting, we assume that the position of the overhead camera and table are fixed, the camera is approximately placing right above the table, and cards can only be placed in predefined card zones. Some of the implementations in the following session take advantages of these static environment conditions in the calibration stage to increase their efficiency. Calibration starts from showing a calibration mat on the screen. It checks for any error in the environmental setup and extracts the rectangular search windows.

4.2 Card Locator

We use *search window* as a heuristics to search only the regions corresponding to player input. They are predefined as card zones identified during calibration. To find player input, we first compute the image difference between the current frame and the previous frame, and apply a threshold to get a binary image. Then we select those with significant number of changed pixels as candidates to ignore noises. The system maintains the state of change of each predefined search window, and *activates* only when its state transits from "changing" to "unchanged", which occurs after the player has put a card and withdrawn his/her hand completely. This approach searches only the regions where the player inputs are most likely taking place, thus reducing both frequency and size for searching.

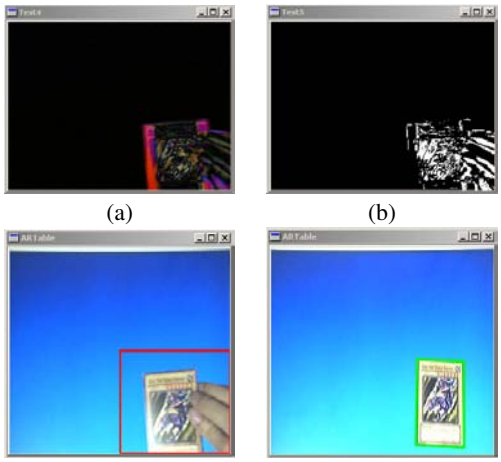
To locate the position of the card, we use Canny edge detection algorithm to find all the edges. Then we search all possible contours for: 1) The contour is composed of exactly four corner points; 2) The four angles between joint edges are nearly right angle; 3) The area bounded by the contour lies within certain thresholds determined in the calibration procedure. Because the camera is orthogonal to the table surface, we can assume that the card approximately remains its rectangular shape and area.

Figure 4 shows the procedure to locate the card. The search window at the lower right corner contains changes (Figure 4a). The image difference (Figure 4b) and the binary image after applying the threshold (Figure 4c) are computed. The search window is activated when its content just stops changing, and a card is located by performing a search on this search window (Figure 4d).

4.3 Card Detector

Finally, we have to obtain the card input, including its *location* (already indicated by its corresponding search window), *orientation*, and the *card ID*. Orientation, either horizontal or vertical, is easily determined by comparing the length of the card's edges. To identify the card with the database is much difficult. First an undistorted version of the card is computed using geometric transformation followed by bilinear interpolation. This eliminates geometric distortion occurring when an image is captured. The undistorted image is then applied to query the image database.

The query process can be subdivided into two main stages. The first stage can be regarded as an *image retrieval stage*, from which relevant images are retrieved as candidates using an image query. Our image database belongs to color-based retrieval system for 2D still images, since higher level features are lost in low resolution query images (about 40x40 pixels). As a prototype system, we only use



(a) (b) (c) (d)

Figure 4: Procedure to locate search window.

a set of 100 cards in the database, thus indexing with card types is enough, as indicated by their background colors. The second stage is the *image matching stage*, during which the ART system selects one card out of those candidates and confirms that this card matches the querying image. As many features are lost due to extremely low resolution of the card image, performing a pixel-by-pixel matching would be the most accurate way. We use *Block Matching Algorithm* to compare the inner parts of the images since they are distinct among cards. The overall pixel difference of the query image and the database image is calculated as follows:

$$D_P(I_Q, I_D) = \sum_{i=1}^{w} \sum_{j=1}^{h} (I_Q(i,j) - I_D(i,j))^2$$

$I_D(i,j)$ and $I_Q(i,j)$ are the pixels of query image I_Q and database image I_D respectively. The candidate with the smallest value of $D_P(I_Q, I_D)$ below the threshold is selected as the best match. To further improve the accuracy, we propose the *Improved Block Matching Algorithm*, in which the matching is performed on nine subdivided squares separately. We accept a match only if all the nine pixel differences are below the threshold. This avoids some errors in a practical game play, since this method would reject large local errors, which is often ignored in the global sense.

4.4 Command Detector

Other than the card inputs, players can also use buttons on the screen as input, allowing player to make more advance actions, like pressing button to trigger monster attacks and spells. With the overhead camera only, approaches described in *EnhancedDesk* [Koike et al. 2001] and *DigitalDesk Calculator* are not appropriate. Another method to locate the pressed button, which is less expensive but still accurate enough for our system requirement, is to keep track of the predefined search windows of the button for changes, using similar techniques as the card detection (Section 4.2)

5 Rule-based Game Engine

The *Game Core Module* co-operates Perception Module for input, Game Enhancement Module for output, and Database Module for

game data. It also maintains game states and game flows as well. The *Game Core* is designed as a *rule-based game engine* which receives and raises appropriate events according to the game rules. It is *extensible* such that new cards can be extended by adding new rules and new card info to the database alone. It is *flexible* in the sense that the game logic is maintained in term of rules, which is more flexible comparing to brute-force programming. And it is *generic* in the sense that a new game can be implemented by modeling it into a new set of game rules, since the game logic and game flow are solely determined by the game rules.

The *Rule Manager* is responsible for storing all the game rules and inferring the rules to trigger appropriate events. It contains a pool that loads and stores the rules dynamically during the game. A rule consists of a rule body in a *premise-conclusion form* and an action list. The inference mechanism we engaged is *forward-chaining*. Once a rule is fired, its actions are carried out accordingly with one of the two types. For actions that *update game states*, such as updating the score or summoning a monster, they are executed in the *Game Manager* which stores and manipulates game states. For actions that *update rules* in the pool, they are executed in the *Rule Manager*. There are six fundamental actions for updating rules as shown in Table 1. The first four actions are essential to keep the game logic, while the last two actions add and remove rules from the pool when cards are put or destroyed.

Action	Effect
Set Premise	Set a premise to `TRUE`.
Unset Premise	Set a premise to `FALSE`.
Reset Premise	Reset a premise to `UNKNOWN`.
Reset Rule	Set a rule to `UNKNOWN`.
Load Rule	Load a rule to the pool.
Remove Rule	Remove a rule from the pool.

Table 1: Six fundamental actions.

6 Game Enhancement

The *Game Enhancement* module receives commands from the *ART Card Game Core*, and generates corresponding display and sound effect. It displays the 3D game mat and animations to players. During the game, the instructions and command buttons change according to game status, and respond to player inputs. Monsters are shown in the field with their attack points. They fight each other if players give commands to them with the input buttons. This provides an interactive interface for players to play the game. At the same time, it produces sound effect to make the game more enjoyable and entertaining. For example, monsters attacks or buttons pressed raise appropriate sound to enhance the game and notice the players. Figure 5a shows the configuration of the game interface, and Figure 5b is a screen shot of the animation when a monster destroys another monster.

7 Conclusion

We developed a prototype of the Augmented Reality Table for card games, which makes use of recent augmented reality techniques to enhance existing trading card games, while remaining much of original playing style. Using visual input as the only input, the system identifies players' input commands and cards without the use of markers. It provides an interactive interface for players, performs

(a) (b)

Figure 5: Visual enhancement.

player's effort to maintain game rules and calculations, and creates much joy to players by fancy visual and audio enhancements.

Acknowledgement

The work described in this paper was fully supported by the Shun Hing Institute of Advanced Engineering (SHIAE) of CUHK, and by the Innovation and Technology Fund of HKSAR (Project No. ITS/105/03).

References

BALCISOY, S., TORRE, S., PONDER, J., FUA, P., AND THALMANN, D. 2000. Augmented reality for real and virtual humans. In *Computer Graphics International*, 303–307.

KOIKE, H., SATO, Y., AND KOBAYASHI, Y. 2001. Integrating paper and digital information on enhanceddesk: a method for realtime finger tracking on an augmented desk system. In *ACM Trans. Comput.-Hum. Interact*, vol. 8, 307–322.

LEE, W., AND WOO, W. 2005. Tarboard: Tangible augmented reality system for table-top game environment. In *2nd International Workshop on Pervasive Gaming Applications*.

LYU, M. R., KING, I., WONG, T. T., YAU, E., AND CHAN, P. W. 2005. Arcade: Augmented reality computing arena for digital entertainment. In *2005 IEEE Aerospace Conference, Big Sky, Montana, U.S.A.*

MILGRAM, P., AND KISHINO, F. 1994. Augmented reality: A class of displays on the reality-virtuality continuum. In *SPIE, Telemanipulator and Telepresence Technologies*, vol. 2351, 42–48.

STARNER, T., LEIBE, B., SINGLETARY, B., AND PAIR, J. 2000. Mindwarping: towards creating a compelling collaborative augmented reality game. In *5th international conference on Intelligent user interfaces*.

SZALAVHRI, Z., ECKSTEIN, E., AND GERVAUTZ, M. 1998. Collaborative gaming in augmented reality. In *ACM Symposium on Virtual reality software and technology*.

THOMAS, B., CLOSE, B., DONOGHUE, J., SQUIRES, J., DE BONDI, P., MORRIS, M., AND PIEKARSKI, W. 2000. Arquake: An outdoor/indoor augmented reality first person application. In *4th International Symposium on Wearable Computers*, 139–146.