MultiMice Air Hockey. A Game with a Low-Cost Non-Conventional Interface

Damián Flores, Silvia Castro and Sergio Martig,

Laboratorio de Investigación y Desarrollo en Visualización y Computación Gráfica (VyGLab),
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
8000 Bahía Blanca, Argentina
{df, smc, srm}@cs.uns.edu.ar

Abstract. Interaction between user and computer is gradually going beyond the traditional mouse and keyboard. New ways of interaction are emerging, which use new and different means for data acquisition. Virtual or augmented reality, haptic, tangible or camera-based user interfaces are all examples of this. Computer games take advantage of these new facilities, as they obtain the benefit of providing new user (player) experience with every new interface developed. In this article, we present a non-conventional user interface for the popular Air Hockey game. This interface is based on a table-like surface with support for multiple users. Additionally, it does not require special nor costly hardware.

Keywords: Non-conventional Interfaces, Multiple-User Interaction, Table-like display.

1 Introduction

User interfaces are always in continuous development and progress. Since the command line interface, a large number of different interfaces have been developed, including mouse, touchpads and multitouch screens, among others. Nowadays, there is a tendency for user interfaces to go beyond the traditional desktop [1].

The use of compact accelerometers and gyroscopes allows to track the movement of a device, making it possible to control a game or application by using the physical movement of the device, as in the case of the Wii controller. Camera-based interfaces also provide another way to get gesture sensing or recognition. Augmented reality, haptic interfaces, voice recognition, etc, provide new ways for interaction with the computer. All these examples are not only non-traditional ways of providing input to the computer applications, but also are outside the view of an everyday desktop.

Every new interface presents its own new ways of interaction, and the user may sometimes need to get used or require some training. Also, some systems require specific hardware or buildings with special working conditions.

Computer games are one of the driving forces of software industry. New games are closely associated with computer technology development, as new games demand new technology and, at the same time, technology development allows new game experiences.

In this context, we developed an application, the MultiMice Air Hockey, that allows multiple user interaction, emphasizing not only the fact of transcending the desktop, but also the use of low cost hardware.

2 Background

Games with non-conventional interfaces have begun to emerge. Augmented reality is a source of new game ideas; some classical board games, such as Mah-Jonng, Casino, or a Maze, were developed by Szalavari [2] in which the emphasis was specially in collaboration. The Goblin XNA [3] is a framework for 3D user interfaces, focused in gaming and augmented reality.

Tangible interfaces are also used for game and learning, NikVision [4] is a console for children learning that makes use of tangible interfaces; force feedback also extends the interaction experiences by manipulating our sense of touch.

Other examples of new interfaces are multitouch tables, being probably the Microsoft Surface one of the most known. However, its price ascends to some thousand dollars. Cheaper alternatives exist, built by many other manufacturers, but the cost is still high for a home game application. These tables allow multiple users to interact at the same time. This allows stunning games and/or applications to be developed, such as the coming-soon RUSE [5] game. There also exist some approaches to simulate the behavior of such tables by using the Wiimote [6]. In our case, we have multiple user support, not by means of touching but by providing each user with a mouse and enhancing the output with a large projecting surface.

Taking into account these approaches, and with the design condition of an inexpensive development, we created a game application having multiple user support on a table-based interface.

3 The Game

Most computer games are played in a conventional PC configuration, based on traditional interaction styles. This configuration consists typically on a PC, a CRT or LCD display as an output device, different conventional input devices like mice, keyboard and joysticks; and WIMP interfaces.

Additionally, there is a large class of games played on surfaces representing boards, courts, terrains, etc. and those surfaces are represented on the conventional general purpose displays before mentioned. Display devices greatly influence the way in which a player interact with the game. Interaction enables the user to express himself in a human-like fashion and this interaction could be more effective and satisfying [7].

In order to achieve a more natural interface as an alternative to the quite expensive table-based games for multiple users, we have developed a hardware/software interface transcending the desktop barrier and providing intuitive and more human-like interaction techniques. Besides, another key characteristic is that it does not require costly hardware nor particular buildings, unlike some of the applications mentioned previously. The hardware used in this interface is easily available. An horizontal rectangular frame, wrapped with a special transparent plastic fabric, makes the natural surface display; a multimedia projector provides the output image on the surface and 2-to-4 mice allows interaction with the application. This arrangement allows a tabletop game to be played in a natural way and without special training.

Fig. 1. Arrangement of the table surface. Each user is provided with a mouse and the surface image is displayed with the help of a projector.

3.1 The Real Game

Air Hockey is a famous and popular game in which two competing teams (typically one player per team) try to score points in the goal of the opposing team. The game requires an air-hockey table, a mallet per player and a puck.

The air-hockey table consists of a large and smooth surface with a surrounding rail to prevent the puck and mallets from leaving the table (Fig. 2). The table surface has reduced friction to speed up the game. It is played typically by two players, and the winner is the first who scores seven goals.

Air Hockey was chosen given the similarity of the interface proposed and the actual game, with the horizontal display representing the table and each player controlling one mallet through the use of its mouse. User interaction in this way results natural for the player, since mouse movements map directly into mallet movements.

Fig. 2. A physical air hockey table (*left*), and the mallets (*right*).

3.2 Our Proposal

Novel display designs present new opportunities for games; in the presented table-like display, the surface enables users to interact with the game in an intuitive and natural way; for this application, an intuitive physical interaction modality is also presented.

In the following sections we describe how the game can be played in this context, and then, a detail of the hardware and software developed is presented.

3.3 How to Play

The user is presented on the table surface with the air hockey table and is able to control his mallet by moving the mouse over the surface. In a classic configuration, two players compete and the interaction between them is merely competitive. There exists the possibility of four players, two per team, to play the game; in this case, team-mates can cooperate to build strategic moves, such as a zigzag puck movement.

Fig. 3. The game in two-player mode. Each mouse controls a game mallet.

The table area is divided, half the area for each team, and each team must remain inside its area. In the game this is accomplished by restricting the mallets movement inside their area. In the four player case, the team area is further divided, half for each player. This is due, in part to avoid interference between players of the same team, and in part to simplify the collision management.

3.4 Hardware

From the hardware point of view, we must consider the input and output elements of the interface. For output, it is used a horizontal table display, with the output image being projected onto the surface from below of the table. In order to make the image larger, a mirror is placed below the table; thus allowing the projector to be placed farther, enlarging the final image size (Fig. 4 shows the mirror).

For input, each user is provided with a traditional mouse, with the ability of using them all at the same time. Wireless mice may be used to make the user to feel more comfortable. Additionally, the interaction could be enhanced, in a more realistic way, with an adequate prop based on cheap mouse technology.

This combination of horizontal display and multiple mice allows more than a single user to stand at the table and to interact intuitively in a collaborative or competitive way within the game.

Fig. 4. The mirror below the table makes possible to enlarge the image.

3.5 Software

The application developed is a version of the popular Air Hockey game, taking advantage of the high similarity with the real game. Every physical element such as

the table, puck and mallets has its corresponding game component, which is responsible of the correct update and drawing of the models. The models of the objects were built using the 3D sketching software SketchUp.

There are three main activities to perform in the game logic: Positioning the mallets, moving the puck and checking for goals. The mallets are moved according to user movements of the mouse; the puck follows physics laws of movement and collisions with the mallets and with the table borders; and the goals are detected monitoring the position of the puck on the table.

The physics behavior is encapsulated in a single class in the project, the *CollisionManager* class. This class provides its clients with methods to detect collisions and to handle the behavior of the colliding objects.

4 Implementation

The game was written in C#, using the XNA framework, and runs on Windows XP. XNA is a game development framework developed by Microsoft, with the particularity of running in a managed runtime environment which is available for Windows XP, Vista and the Xbox 360 game console. As a framework, it provides all the base infrastructure for game development, such as the game loop, time management, interface with the graphics device, etc.

In the next sections, a description of how the problem of multiple mice arose and how was it addressed, is presented.

4.1 Getting Multiple Mice Support

In order to being able to use the proposed hardware, it is needed to support the interaction of multiple mice at the same time; thus, the first task was the addressing of this issue. This topic is not new, it was addressed for different platforms, MID [8] (Multiple Input Devices) is a java package that supports multiple mice, CPNMouse [9] is a driver for Windows XP and Tsee and Greenberg [10] made a toolkit for Single Display Groupware applications.

However, difficulties arose when interacting with some of them, for example, MID does not work on Windows XP, the others are event-driven or callback-based, not having a good behavior in our particular domain, considering a game implemented with a game loop polling the state of the input device on every loop. In the event-driven case, an event meaning “MouseMove” interrupts the game loop at an irregular frequency, depending on the user movements, a similar effect happens with the use of callbacks. Based on this, we used a facility provided by Windows XP to manage the state of the mice.

Windows supports multiple mice, but is not able to recognize or differentiate which one the data is coming from. [11] Having more than a single mouse connected is possible and easy by simply plugging them into free USB ports. However, in such a situation, we cannot identify the source pointing device. Even though each mouse provides its own input stream, the resulting stream is made by the sum of all mice. This behavior is due to Windows architecture, in which the implicit assumption that

nobody would use multiple mice, led to the fact that the ID of the different devices were not taken into account by the Win32 API. This API receives packets with mouse information sent from the drivers. Information includes button state, delta values and a unique ID. When the Win32 API system processes the mouse packets to generate the mouse events, the OS just eliminates the ID identifier, making the programmer unable to distinguish between the multiple mice sending events.

4.2 The Raw Input API

The raw input API [12] is an interface that allows programmers to gather user raw input from a wide range of devices. Provided by Microsoft, this API is available in its operating systems since Windows XP. User input can come from a joystick, touch screen, remote control, or other devices, thus, allowing great flexibility in user input. These devices are sometimes known as Human Interface Devices (HIDs) and the raw input API provides us with a robust method of accepting input from any combination of HIDs, including the mouse and keyboard.

The particularity of this API is that it allows simple access to all HIDs, but only to the raw input, meaning that the data offered is low-level and device-specific, being up to the application to interpret it properly. The application must register the HIDs it wants to receive input from, and the information is received via the WM_INPUT message.

4.3 The RawInputLib

Based on the raw input API, we developed a new library to interface with the low level information offered by the API. Written in C#, this library offers the following features: It handles and encapsulates the complexities of having to manage the low level data, allows the management of multiple independent mice and offers a simple and clear interface to application programs.

This library was designed to be capable of managing the raw input of multiple devices at the same time. But, at the present, it just supports multiple mice only.

It manages two aspects of each device, the status and the device information. The status is all the information that varies very frequently, such as the mouse position or state of the buttons, whereas the device information is practically static. Retrieved mainly from the Windows registry, device information include: Device ID, Name given by Windows and a description (Such as “Generic PS/2 mouse”).

As this library was thought with the game in mind, it maintains the state of the device and its information is accessed by polling, instead of through events.

Another aspect taken into account in the library is the orientation of the mouse, that is, when two people stand in front of each other, moving the mouse up, down, left or right is not the same in the sense that, for example, left for the first means right for the other. This issue arose because of the change in the display orientation, when it is vertical there is a general agreement in what “left” means; in contrast, when horizontal, it depends on user location. So, coordinates and movements need to be transformed according to mouse orientation.

The interface with the programmer is very simple, carried out by having an instance of the MouseManager class. This class, as its name suggests, is the responsible of managing every mouse and it is the façade of the library. Through this class, it is possible to access the number of devices detected, and hence query information for each one.

Once created, and before start querying data, the mouse manager needs to register the devices by providing the handle of the window corresponding to the application.

5 Conclusions and Future Work

We have presented the MultiMice Air Hockey, a game provided with a low cost non conventional interface allowing players to interact in an intuitive and simple way. This is due to the similarity in the disposition of the display, enabling the user to play the game as if he were at a real table. Also, the use of mice as input devices makes the user capable of using the system without requiring special training, except for the game itself. Until now, only several informal tests have been undertaken, mainly for study of how intuitive the game interactions are. We plan to design tests for evaluating this framework as well as for designing new ones.

A trend towards new interaction schemes could be identified when developing new applications in this hardware/software context. We plan to develop other interaction styles which are beyond the mouse, to obtain intuitive interaction techniques to work with users' hands as they would do in the physical world. Many questions remain to be answered in this growing field.

Moreover, we have developed the multiple mice library and the table as basic elements that can be used to create new multiple collaborative games and applications.

6 Acknowledgements

This work was partially supported by the PGI 24/N020 and 24/ZN19, Secretaría General de Ciencia y Tecnología, Universidad Nacional del Sur, Bahía Blanca, Argentina.

References

1. Martig, S., Castro, S., Larrea, M., Escarza, S.: Interfaces no Convencionales. Su impacto en las interacciones. WICC'09 (2009)
2. Szalabari, Z., Eckstein, E., Gervautz, M.: Collaborative Gaming in Augmented Reality. Proceedings of VRST'98, pp. 195-204, Taipei, Taiwan, 1998.
3. The Goblin XNA Framework, http://graphics.cs.columbia.edu/projects/goblin
4. Marco, J., Cerezo, E., Baldasarri, S.: NikVision: Desarrollo de Videojuegos Basados en Interfaces Naturales. CEIG'08 Barcelona, 2008.
5. The RUSE game, http://www.therusegame.com

6. Johnny Chung Lee Projects, http://johnnylee.net/projects/wii
7. Kerren, A., Ebert, A., Meyer, J., Editors. Human-Centered Visualization Environments, GI-Dagstuhl Research Seminar, LNCS 4417 Springer-Verlag, 2007
8. MID: Multiple Input Devices, http://www.cs.umd.edu/hcil/mid
9. Westergaard, M.: Suporting Multiple Input Devices in Microsoft Windows. Proceedings of Microsoft Summer Workshop for Faculty and Ph Ds. Cambridge, England, 2002.
10. Tsee, E., Greenberg, S.: Rapidly Prototyping Single Display Groupware through the SDGToolkit. Proc Fifth Australasian User Interface Conference. Australian Computer Society Inc., pp 101-110. (2004)
11. Pawar, U.S., Pal, J., Toyama, K.: Multiple Mice for Computers in Education in Developing Countries.
12. Microsoft Raw Input API, http://msdn.microsoft.com/en-us/library/ms645536(VS.85).aspx
13. Castro, S., Larrea, M., Martig, S.: Interfaces No Convencionales para Juegos. Primeras Jornadas de Educación en Informática y TICS de Argentina (JEITICS 2005)