대화면 FPS 게임을 위한 레이저센서기반의 대형스크린과 레이저광원 권총의 설계와 구현

한녹손, 김성환
서울시립대학교 컴퓨터과학부
e-mail : han.ngoc.son@gmail.com, swkim7@uos.ac.kr

Laser-recognizable Screen and Gun with Laser Source for Realistic Big Screen First Person Shooters Games

Ngoc Son Han and Seong-Whan Kim
Dept. of Computer Science, University of Seoul

Abstract

In this paper, we present a new game interface design for First Person Shooters (FPS). Previously, FPS on computer is commonly played using keyboard/mouse or joystick along with PC display. We improve the communication environment between player and game world by means of new control system including large screen and laser gun, which create a real life-like space for players. Because traditional display for FPS uses CRT, it cannot support large screen display due to limitation of CRT technology. We designed and implemented a new input device using laser recognizable display. Results suggest that the combined interface creates a method which helps beginners to enjoy playing a FPS immediately and gives experienced players a new gaming experience.

1. Introduction

A first-person shooter (FPS) as shown in Figure 1 is a video game that renders the game world from the visual perspective of the player character and tests the player's skill in aiming guns or other projectile weapons. All FPS feature the core game-play elements of movement and shooting, but many variations exist, with different titles emphasizing certain aspects of the game-play.

Fig. 1. Example of First Person Shooter.

Because traditional display for FPS uses CRT, it cannot support large screen display due to limitation of CRT technology. We designed and implemented a new input device using laser recognizable display. We improve the gaming interface between player and game world by means of new control system: laser-recognizable screen and laser gun. Our system creates a real life-like space for players. Results suggest that the combined interface creates a method which helps beginners to enjoy playing a FPS immediately and gives experienced players a new gaming experience.

We review related technologies for FPS in section 2, and propose and show experimental results of our new input device in section 3, section 4. We conclude in section 5.

2. Related works

A first-person shooter (FPS) is a video game that renders the game world from the visual perspective of the player character and tests the player's skill in aiming guns or other projectile weapons. Most modern first-person shooters on the PC utilize a combination of the WASD keys of the keyboard and mouse as a means of controlling the game (commonly referred to as "WASD/Mouse") as shown in Figure 2. One hand uses the mouse, which is used for free look (also known as mouse look), aiming and turning the player's axis. On the keyboard, the arrow keys (or other keys arranged in the same manner, such as WASD, ESDF or IJKL) provide digital movement forwards, backwards, and sidestepping (often known as "strafing" among players) left and right. Usually these buttons make the player run, and a nearby button must be pressed in order to walk. WASD/Mouse interface is not natural way of FPS game interface because WASD/Mouse is initially designed for entering texts.

Fig. 2. WASD/Mouse interface for PC FPS games.

Besides FPS game on PC, we can experience FPS using CRT television as shown in Figure 3. However, CRT display cannot be larger than 30 inches due to the limitation of CRT technology. The primary advantage of CRT monitors is their color rendering. The contrast ratios and depths of colors displayed were much greater with CRT monitors than other type of monitors. The other advantage that CRT monitors is the ability to easily scale to various resolutions. This is referred to as multi-sync by the industry. By adjusting the electron beam in the tube, the screen can easily be adjusted downward to lower resolutions while keeping the picture clarity intact. While these two items may play an important role for CRT monitors, there are disadvantages. The biggest of these are the size and weight of the tubes. An equivalent sized LCD monitor for example is upwards of 80% smaller in size and weight compared to a CRT tube. The other major drawback deals with the power consumption. The energy needed for the electron beam means that the monitors consumer and generate a lot heat. Cons of large CRT display includes (1) Very heavy and big, (2) large amounts of energy consumption, and (3) Generation of excess heat. Arcade FPS uses CRT for its main display and gets gun shot position (x, y) information using cathode ray position tracking.

Fig 3. FPS on CRT televison.

We can also experience arcade FPS as shown in Figure 4. As shown in Figure 4, arcade FPS game uses CRT display, and the display cannot be larger than 30 inches due to the limitation of CRT technology.

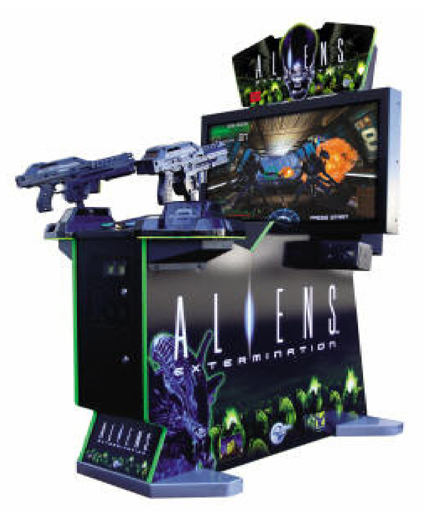

Fig. 4. CRT ray tracking for gun shot positioning.

Another notable gaming interface is PistolMouse as shown in Figure 5. It has a joystick and gun interface. It is basically same as InterCube2 technology.

Fig. 5. PistolMouse gaming interface.

Another gaming interface is WIImote as shown in Figure 6. It has two components: sensor bar and WIImote which is a remote controller. Sensor bar can compute the location of WIImote, and thereby game console can track user action. It is based on 3 axis acceleration sensor and 10 Infrared LED sensing.

Fig. 6. WIImote: gaming interface for WII console.

3. Laser-Recognizable Screen for FPS

Because traditional display for FPS uses CRT, it cannot support large screen display due to limitation of CRT technology. We designed and implemented a new input device using laser recognizable display.

As shown in Figure 7, there are three components in our new control system, laser gun and larges screen which recognizes position of the laser shot. The Laser gun is installed away from the large screen, it can fire laser beam that will reach the screen and locate the position of target for the player. The Large screen is a special one made of optical sensors with capabilities of rendering image and receiving signals from laser gun. Our approach comes from interplay between the large screen and a specially designed laser gun. The Laser beam coming from the gun draws a marker on the screen and its position is recognized and is inputted to the computer with other trigger information.

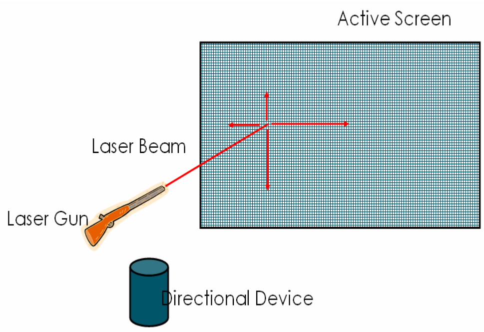

Fig. 7. Active laser recognizable screen and laser gun

As shown in Figure 8, drivers for new devices are written under the scope of Window Driver Foundation (WDF), Windows Display Driver Model for large screen and Kernel-Mode Driver Framework (KMDF) for the laser gun and the screen. On top of that new control system, we develop a FPS called QuakeS based on Quake III whose source code was released under GPL license. A new module using DirectInput Windows API for collecting input from user via game controllers for input control is added to interact with new devices.

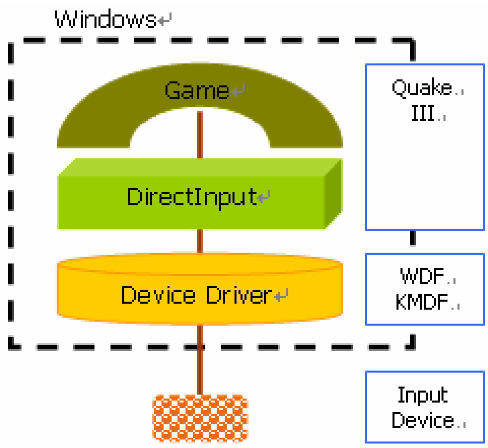

Fig. 8. FPS with New Input Devices Architecture.

4. Experimental Results

We implemented laser gun and laser-recognizable screen as shown in Figure 9. The screen can be more than 100 inches with thin panel structure. In this experiment, we implemented 50 inch screen, which recognizes the position of laser shot in the screen. This show that we can develop new FPS game for large screen support (e.g. 40 inches screen).

Fig. 9. Laser gun and laser-recognizable screen

New device driver uses the WDF defaults includes the following functions: (1) DriverEntry routine, which creates the driver object, (2) Evt

Driver Device Add event callback, which creates the device object, a device interface, and a default I/O queue, and (3) I/O call back functions for read, write, and device I/O control requests. The Driver Entry routine is the first driver function called when the driver is loaded. It has functionality of (1) creating a driver object (WDFDRIVER), which represents the loaded instance of the driver in memory. In effect, creating this object "registers" the driver with KMDF, (2) registering the driver's EvtDriverDeviceAdd callback. KMDF calls this function during device enumeration, (3) Optionally initializing event tracing for the driver, (4) Optionally allocating resources that are required on a driver-wide (rather than per-device) basis.

Our driver supporting Plug and Play has an EvtDriverDeviceAdd callback function, which is called each time the system enumerates a device that belongs to the driver. This callback performs actions required at device enumeration, such as (1) Creating and initializing a device object (WDFDEVICE) and corresponding context area, (2) Setting entry points for the driver's Plug and Play and power management callbacks, (3) Ceating a device interface, (4) Cnfiguring and creating one or more I/O queues, (5) Creating an interrupt object, if the device controls physical hardware that generates interrupts.

Figure 10 shows the laser gun, which has laser source. It can shot laser beams using reddish beams.

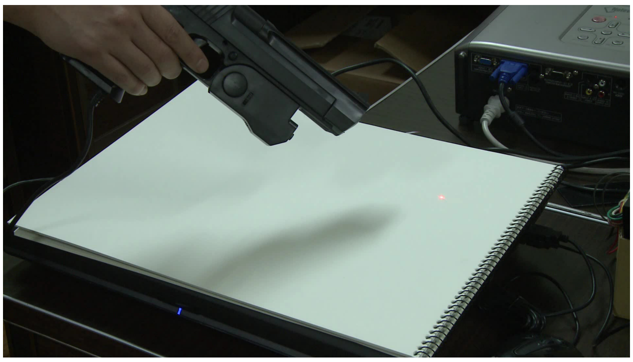

Fig. 10. Laser gun with laser source.

5. Conclusion

In this paper, we have presented new design for control system of FPSs. We covered several related issues ranging from hardware design to software design that help building and deploying new devices for a practical game. New control system including Laser gun and laser-recognizable screen were designed and manufactured with Windows drivers support that compatible with Windows Application. Device drivers were developed with Windows Driver Kit and using WDF model. On the game side, input control module used DirectInput library to deal with input signals. Besides that the new control system creates fun and natural environment for players. And despite of the absence of some activities within the game, but almost skilled players find new control system worth experiencing.

References

[1] S. Beckhaus, K. J. Blom, and M. Haringer, A new gaming device and interaction method for a First-Person-Shooter, In Computer Science and Magic 2005, GC Developer Science Track, Leipzig, Germany, 2005.

[rhdrr2] Microsoft Windows Hardware Developer Central, http://www.microsoft.com/whdc.

[3] Microsoft, Architecture of the Kernel-Mode Driver Framework, Microsoft Corporation, 2006.