Game Controllers

By

Achal R Chaurasia

Department of Mathematics and Computer Science

Submitted in fulfillment

Of the requirements for the degree of

Bachelor of Science (Computer Science)

Algoma University College

Sault Ste Marie, Ontario

March 2007

Abstract

I was always fascinated with games and every aspect of game development including the hardware and software. I have worked with software before and therefore wanted to do explore the world of gaming hardware.

Therefore the objective of this thesis is to study and research the history of various gaming devices, the various types and technologies of gaming devices, see the various facets of the gaming industry and observe how it rose from its humble beginnings to form such a global market worldwide, the future of gaming devices and to study a particular technology for the purpose of developing a gaming device and develop a working model to demonstrate.

Keywords: console, buffer, FIFO(first in first out), MFC(Microsoft Foundation Class), operation, application, cartridge, magnetic media, security, optical media, circuit, devices, portable, handheld, microprocessor, flash memory, internet distribution, Bluetooth, RAM, coprocessor, volatile storage, USB, COM.

Acknowledgements

I would like to thank the entire Computer Science faculty at Algoma University College for their encouragement and support. Special thanks to Dr George Townsend for providing constant guidance and for holding valuable meetings. Gratefully thanking my Supervisor Dr Yi Feng for her time and support towards helping me write and present this paper. This would certainly not have been possible without her guidance. Thanks to Dr George Townsend again for offering to be my thesis reader and for rendering support through the presentations. The Computer Science TA's provided the necessary lab facilities promptly and I thank them. I would also like to thank all my classmates, friends and family that gave me vital input and feedback throughout, as well offered their help and encouragement in every way they could.

Table of Contents

List of Figures

List of Tables

Introduction

Games and gaming devices have been around for more than 100 years. They have increased in complexity over the years from their humble beginnings as strictly mechanical devices to sophisticated, state-of-the-art, microprocessor-driven machines [5].

The earliest gaming hardware made was in 1899, when Charles Fey invented the Liberty Bell machine [1]. Since then many changes have taken place in the development of gaming devices in terms of size, technology, implementation and ease of use. The very first gaming devices were mechanical as stated earlier. They had very limited use and high maintenance. Electro- mechanical devices were developed soon after. They had a combination of both mechanical parts and electronic circuitry. This was a quantum leap from the earlier generation of devices. This was the time when the gaming revolution really caught on and video games became commonplace. People started considering gaming as a serious business industry in this era [8].

The electronic age followed soon afterwards. It saw the beginning of video games power-driven by solid-state circuitry. These machines gave players a challenging new way to play – offering decision-making options like never before. This technology extended in the following decade creating a hybrid series of gaming consoles and devices. The size of these game consoles also started reducing. From being huge stationary machines, they developed into portable gaming consoles which could be easily be connected to the T V and be played almost anywhere. The latest generation of gaming consoles and devices include the use of computer technology and microprocessors. It offers flexibility in game, making two-way communication possible between

the microprocessor and internal components. Over a period of time, games became faster, more intelligent and the ultimate effect it had on the users was unreal. The graphics in these games also keep getting better and better and because of the use of computer technology, more and more games can be created fairly easily. The latest generation of game consoles have a variety of features and playing games is just one of them. A user can surf the internet, play music, videos and DVDs on it. Even while gaming, a user can play the same game with multiple people online with the use of the internet [9].

1.1 Industry Standard for Game Controllers

Unfortunately, there is no one standard for game controllers, as every company tries to out do each other by adding more functionality in making their games consoles and gaming controllers better than their competition.

Keyboards and mice are still considered as the gold standard thanks to its unprecedented precision and flexibility and the fact that they remain standard even in multiple platforms.

Chapter 2

History of Gaming Devices

Although the first gaming machine was developed in 1899, it wasn't until the early 1970's until these games came on the market. I have categorized them in seven generations depending upon the changes and improvements that were made in every generation [18].

2.1 First Generation:

In 1952, A.S. Douglas wrote the first version of Tic-Tac-Toe (a k a Noughts and Crosses) for the EDSAC as part of his doctoral dissertation at the University of Cambridge. He was interested in exploring the interaction between humans and computers. In 1962, Steve Russell created "Spacewar!" on a DEC PDP-1 while working on his graduate degree in engineering at MIT.

In 1966, Ralph Baer began exploring ways to use the television as a display device for interactive amusement. While working at Sanders Associates, he began development of a trial product of a video game system. In 1969, Ralph Baer filed one of his many video game patents on August 21st, 1969. This one was specifically for creating a "Television Gaming Apparatus and Method." This was just one of many pieces that later evolved into the Magnavox Odyssey. In the 1970's, Ralph Baer and Nolan Bushnell

were amongst the pioneers in starting the video game industry , in terms of not only developing it, but also filing patents, selling their games to the manufacturers and in short were responsible in building the business.

In 1971, Nolan Bushnell sold his developed game Spacewar to Nutting Associates who produced 1500 units to sell. But the people found the game very complex and it intimidated them, so it made no money. So the first ever commercial arcade video game was a financial failure. In 1972, Nolan Bushnell left Nutting Associates and formed his own company Atari. He actually wanted to name his company as Syzygy, but the name was already taken. Atari's first game, Pong, was a worldwide success unlike Spacewar. They actually got sued by Ralph Baer's company Magnavox over the game and ultimately lost the case and had to pay a royalty on every game sold[1].

So this is how the gaming industry started and evolved in the 1970's.

2.2 Second Generation:

In 1976, Fairchild Video Entertainment System (VES) released its first game. While there had been previous game consoles that used cartridges, either the cartridges had no information and or just flipped switches or the console itself did not have any gaming knowledge and the cartridge contained all of the game components. The VES, however, developed a programmable microprocessor therefore cartridges only needed a

single ROM chip to store microprocessor instructions. RCA and Atari followed its path soon by releasing their own cartridge-based consoles [12].

2.3 Third Generation:

In 1983, Nintendo released its post popular game at that time, the Famicom in Japan. It supported higher resolution than any other game at that time, had more colours, tiled backgrounds, and high-resolution sprites. This in turn allowed Famicom to be longer and have more detailed graphics. Seeing Famicom's global success, Nintendo brought this product over to the US in the form of the Nintendo Entertainment System (NES) in 1985. In the US, video games were seen as a trend that had already passed. So to sell its products, Nintendo started to make small but innovative changes in its products to distinguish it from older video game consoles, Nintendo used a front-ended cartridge port similar to a VCR on the NES.

They also built a lockout chip into the Nintendo Amusement System games. This kept other companies from producing their own cartridges using the Nintendo cartridge technology and forced all developers to go through Nintendo to get NES games published. This was an extremely smart move from Nintendo which allowed it to do things like prevent developers from releasing low-quality games and limit developers to five titles a year. In short, it really helped them to control the pace of gaming in this era, although they implemented many monopolistic moves like these.

Soon after, Nintendo found its breakout hit game in Super Mario Brothers which is arguably one of the most successful games of all times. Nintendo's success revived the video game industry and more and more companies started to take interest in the video game industry. New gaming devices were soon introduced by these companies in the following years to compete with the NES [15].

2.4 Fourth Generation:

Sega was the only company in the last 1980's and early 1990's that competed with Nintendo but it never made any significant in roads in the popular US market. But Sega soon regained its market share when it released their next console, the Sega Mega Drive, which first came out in Japan in 1988, in the US in 1989 and in Europe in 1990, thus getting a two years advantage before Nintendo finally released the Super Nintendo Entertainment System in reply to Sega's Sega Mega Drive [3].

2.5 Fifth Generation:

Atari was the first company to come out with a 64-bit gaming console. It was called the Jaguar and it was released to compete with Nintendo and Sega which together dominated the video game market at that time. Although Atari was a late entrant in video game market when the competition was already stiff, their first few products were technologically superior than any other console at that time and therefore gained instant popularity and recognition in the market and amongst users.

The Jaguar could not hold its own because of a number of reasons. It was difficult to write games for the Jaguar, therefore not many third parties wrote games for it. Users therefore started getting bored of the game. Also, the competition at the time was very stiff, and customers had a wide variety to choose from. In 1995, Sony released its PlayStation and the Sega announced its Saturn bringing the end for the Jaguar.

Sony finally released its Play station in Japan which after the US is the biggest market for video games in the world. The Play station was actually created after plans to create games on CD along with Nintendo failed. Nintendo backed out at the very end and by then Sony had almost completed the design and research, so it went ahead with the development of the console and eventually released it under the name of Play station. Play station went on to become Sony's most successful gaming console and it forced all the other companies to make optical media based gaming consoles. This was also the first time; gamers started taking Sony as a serious video game company. The Play station went on to sell more than a 100 million consoles making it one of the most successful gaming consoles ever. Boosted by the idea of the play station, Sony released a more compact version of play station calling it the PSone. But PSone could not replicate the play station's success.

The Nintendo 64 was Nintendo's answer to the growing dominance of the PlayStation. It was a 64-bit console, but the only problem was the Nintendo 64 was cartridge based and not CD based as the play station. Nintendo thought that

cartridges were better than CD's as they could not be easily destroyed and this wrong move cost them dearly, as the cost for producing and distributing CD's is much less than that of cartridges. Also the capacity of the optical media kept growing making it possible to store more data on them than that in the cartridges [2].

2.6 Sixth Generation:

In this generation, the PC was used more and more for development of games in gaming consoles. Gaming companies also started using DVDs instead of CDs for storage of game media. Games now kept getting visually better and better, the sound quality improved drastically from the previous generations. Some companies like Microsoft also tried putting a hard drive for storage for game data.

Sony released the PlayStation 2 after the highly successful PlayStation; it had many improvements over the play station like the ability to play DVDs. The play station 2 also became hugely popular and went on to sell more than 100 million consoles worldwide.

Microsoft entered the video game market with their first product called the Xbox. It was also the first video game console to have a hard drive built in for storage. Initially, the game controllers for the Xbox were considered to be extremely bulky and not very easy to handle in terms of comfort, but Microsoft changed all that over a period of time. Xbox really started gaining popularity of the release of its game, Halo[13].

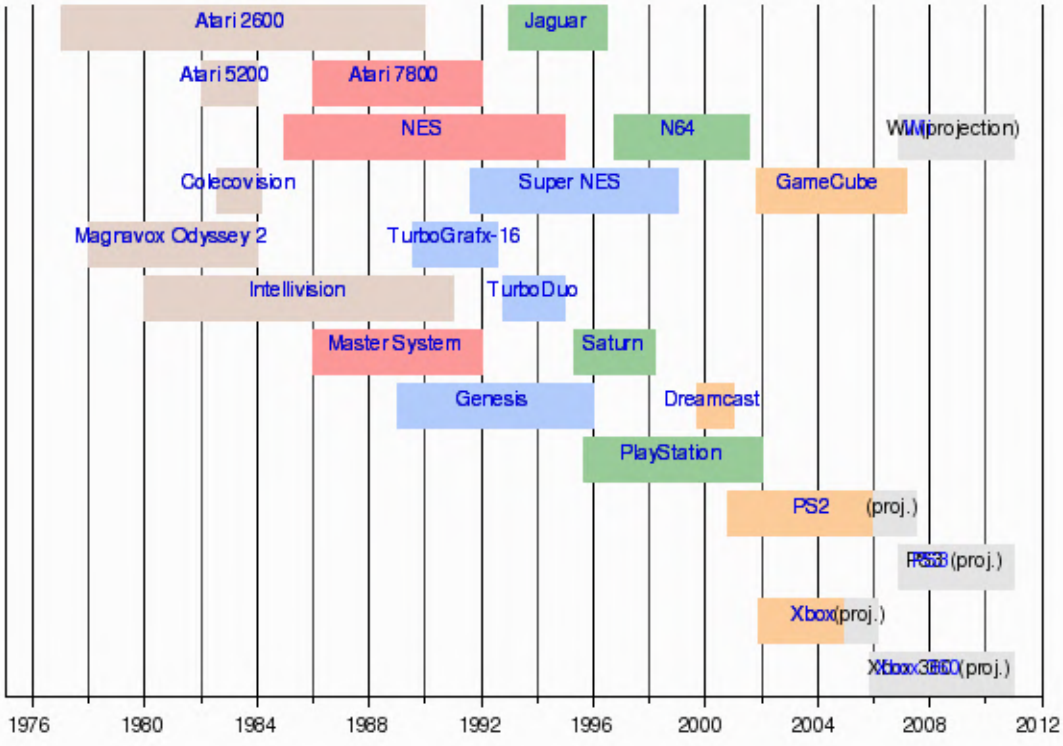
2.7 Seventh Generation:

This is the latest generation in the development of the gaming industry. This will be the first time when Blu- Ray disks will be used in gaming and gaming consoles will be almost looked as a mini computer and an entertainment set rather than just a gaming console. It would be able to do almost anything that can be done on a computer like, surfing, playing videos and movies, connect wirelessly, play DVDs, play games and music, etc

Nintendo is the only company which is not trying to compete in this hi tech, expensive state of the art gaming console market. They have now chosen a different approach by making different types of gaming consoles like the Wii which are much cheaper than the competition and offer a different type of gaming experience.

Microsoft's Xbox 360 and Play station 3 and Play station portable are the latest gaming consoles available. The Xbox has an option of having an HD –DVD; it also has wireless connectivity capabilities. Sony's PlayStation 3 was released in 2006. Play stations come with a built in hard drives with the option of reading and playing Blu-Ray Controllers. It also has blue tooth connectivity capability and tilt sensing ability [14].

Fig 2.7 Abridged Timeline of Game and Gaming Consoles.



Chapter 3

Different Types of Gaming Devices

3.1 Cartridge based gaming console:

Popular gaming consoles like the Nintendo DS and Nintendo 64 still use cartridges. Game cartridges usually involve a circuit board or a PCB kept inside of a plastic casing with a connector at the bottom allowing the device to interface with the console. Some cartridges carried components that boosted the original console's power, such as extra RAM or a coprocessor [19].

Cartridges were the first external media to be used with home consoles. They constantly grew in popularity as more and more companies started adopting this technology and remained the most common mode of distributing games until the mid nineties due to continued improvements in capacity. The size of the cartridges also kept getting smaller and smaller. But, the relatively high manufacturing costs saw them completely replaced by optical media for home consoles by the early 21st century. They are still in use in some handheld video game consoles [18].

3.2 Card based gaming console:

Due to the improvements in flash memory technology, many companies like Sega started producing games stored on smaller cards. The primary advantage that these cards had over the traditionally used cartridge was of smaller size and cheaper cost price. The Sega Master System was the first console to have used these cards. TurboGrafx – 16 also used it soon afterwards. The Nintendo DS still uses these cards. Over a period of time, significant improvements have been made in this technology. Even today, there is an option of saving an Xbox or Play station 2 games on the flash card and then playing it again [20].

3.3 Magnetic media based gaming console:

Magnetic media was first used in computer systems, but was rarely thought as a serious option for the games as the very first magnetic tapes were very unreliable and did not offer such a huge difference in terms of storage capacity. However, they did have a cost advantage over the cartridges. As the magnetic media technology grew, the magnetic disks and tapes became more and more reliable and popular. As a result, the price of the magnetic tapes dropped even further [22].

Apart from the obvious cost advantage, the magnetic tapes enjoyed the advantage of larger capacity, faster reading speed, greater robustness and the growing popularity and awareness by the computer users around the world. But they were also rendered obsolete by the optical media.

3.4 Optical media based gaming console:

By the mid 1990s, optical media enjoyed being the most popular mode of storing data. CDs were the first optical media to come out and CD-ROMS became a house hold name. The optical media was even cheaper and more reliable than the magnetic media. Also, the development of this technology grew at a mind boggling pace [24].

After the CDs, came the DVDs which had nearly 7 times more storage capacity than CDs. Bigger and better games started being developed because storage capacity was not an issue anymore for game development. By the early 21st century, all of the major home consoles used optical media, usually DVD-ROM, which are widely replacing CD-ROM for data storage. The PlayStation 3 has the option of using even higher-capacity Blu-ray optical discs [23].

3.5 Internet distribution:

All three seventh generation or "next-gen" consoles (the PlayStation 3, Wii, and Xbox 360) offer some kind of Internet games distribution service starting a whole new concept of distributing, marketing and playing games over the internet. This concept allows users to download games for a fee onto some form of non-volatile storage, typically a hard disk or flash memory. But buying and playing games still requires considerable bandwidth and speed. Right now, this concept is only viable in North American countries and market like Europe. But many believe that this method of distribution and gaming will only get more popular in times to come [26].

Sony and Microsoft are one of the pioneers in this internet distribution paradigm. Sony's next two games Gran Turismo HD and the vastly popular Tekken 5: Dark Resurrection will only be released and distributed via the internet. They also plan to start a video service in the near future. Microsoft's Xbox Live service still includes the Xbox Live Arcade, featuring digital distribution of classic and original titles. These include arcade classics, original titles, and games originally released on other consoles [28].

3.6 Handheld devices:

Although the first handheld device was produced in 1992, it wasn't up until late 1998, when the colour Game Boy was released, that the true potential of this market was known. It was then that all the gaming companies actually started taking the handheld gaming devices market seriously [29].

These days gaming companies are including many additional features such as playing music, videos and movies, storing contacts, schedules, data on their handheld devices to tap into other markets as well. The latest Sony Play Station Portable can do all of these things and much more giving to the user additional facility of an music player, video player and storage device at the same time.

There are many companies like Sony, Nintendo, Sega and Atari which have released several handheld devices in the market. Some of the popular ones are the Sony Play station Portable (PSP), Nintendo Game Boy Advance, Nintendo DS and Tiger Telematics Gizmondo. Although Nintendo has always been the market leader in this segment, Sony with its latest offering of Play Station Portable is making good in roads in this market and eating into Nintendo's profits. It is believed that Nintendo will come up with a rival product by the end of 2007[30].

3.7 Games on Mobile Phones:

Mobile companies too are looking into taping into the gaming market with offering game options on the cell phones. This trend is still on the early stages and still has not gained as much popularity as many believed it might.

Nokia was the first mobile company to tap into the gaming market when it released the N-Gage in 2003. It had all the features to be qualified as a full fledged mp3 player and a PDA. One could also listen to radio on it. It could also be used as a gaming device. It was a truly amazing concept and when the cell phone came out, it was the only one at that time to boast of such features [31].

Unfortunately, the phone had many technical problems such as the need to turn off the phone and remove the battery while switching between games. It also had many bugs in the software and many operating defects. The most well known of these was "side talking", or the act of placing the phone speaker and receiver on an edge of the device instead of one of the flat sides, causing the user to appear as if they are speaking into a taco[32]. Due to all these problems, the phone was a failure in the US but it did fairly reasonably in Asia and other markets where Nokia had dominance. Nokia later released the new and improved Nokia N-Gage QD which rectified all the flaws of its predecessor but by that time all the hype surrounding mobile gaming devices had gone. Critics believe that Nokia hurried

into launching the N-Gage in order to get a first movers advantage in this segment. They would have been much better in launching it after a year after taking care of all the technological problems.

Chapter 4

Gaming Device – Implementation

4.1 Introduction to the demonstration project

For the purpose of demonstration, I chose to build and develop my own hardware controller right from scratch. The initial idea was to take an existing game controller and make some changes to it, but as I said earlier, the reason why I picked this thesis topic was to research and learn more about the hardware than the software. So I decided to make my own hardware game controller.

After some research, I had the option of either making it USB port based or LPT port based. I chose the latter as there were plenty of resources available on the internet as the LPT port has been around for a very long time. Also some people had already tried to make some type of LPT port based devices. They had put their entire experience online which gave me some idea of what I was getting into. They also offered me their help and guidance as and when I would need it. On the other hand, a USB based device would require being plug and play, which would require lot more coding effort. Not only would that consume more time and effort but would go beyond the scope of my research as I primarily wanted to concentrate on the hardware rather than the software [37].

So after discussing all my plans with my supervisor Dr Yi Feng, I decided on making my gaming device on LPT port. So my plan was to initially develop a hardware

controller, connect all the controls with wire configuration mentioned in the Standard Parallel Port protocol. The next step would be to make the gaming hardware controller to be able to communicate with the computer via the parallel port. This required a lot of patience as even a single loose connection on any of the controls but make the device behave erratically and generate unexpected results. Once the communication was established between the device and the computer, the only thing left was to make a game communicate and respond to the game controller [34].

My original plan in terms of choice of game was to take a full fledged first shooter game like Quake 3, Unreal Tournament, etc and to make the device work with that game. But because of the time constraints, that would not be a possibility. So after further discussions with my supervisor, Dr Yi Feng, we came to the conclusion that it would be better if I would take a simple free game with open source code from the internet and to make it work with the hardware device.

If time permits, there were also plans to make a device driver for the game controller hardware so that other games could also be played using the same hardware controller. Also if anyone wants to continue my research forward, then he/she would be able to do it without having to go through any unnecessary hassle of going through tons of code from my game. Again due to time constraints, I wasn't able to make a true device driver for the game but I did make a .dll file which contains all the major functions to access and use the gaming device. Any programmer can include that .dll file in his/her project and then call the necessary functions to access the gaming hardware. This would

save them the time and effort to make sure the device works and they can concentrate on adding more features to it and connecting them to better games.

So I did manage to get a simple free puzzle game from the internet which was programmed in VC++. It was a Microsoft Foundation Classes (MFC) application. Its programming was fairly simple and did not take a very long to understand. I was able to make changes in the game to make it accommodate the extra functionality of the gaming controller. I would be explaining the working and development of the game and game controller further in this chapter.

4.2 Introduction to Parallel (LPT) port

The Parallel Port is the most commonly used port for interfacing home made projects. This port will allow the input of up to 9 bits or the output of 12 bits at any one given time, thus requiring minimal external circuitry to implement many simpler tasks. The port is composed of 4 control lines, 5 status lines and 8 data lines. It's found commonly on the back of your PC as a D-Type 25 Pin female connector. There may also be a D-Type 25 pin or D-Type 9 pin male connector. This will be a serial RS-232 (COM) port and thus, is a totally incompatible port [37].

Newer Parallel Port's are standardized under the IEEE 1284 standard first released in 1994. This standard defines 5 modes of operation which are as follows:

1. Compatibility Mode.

2. Nibble Mode.

3. Byte Mode.

4. EPP Mode.

5. ECP Mode.

4.2.1 Compatibility Mode:

Compatibility mode or "Centronics Mode" as it is commonly known can only send data in the forward direction at a typical speed of 50 Kbytes per second but can be as high as 150+ Kbytes a second. It is the most basic mode used to send data in only one direction. It is used in very simple devices.

To output a byte to a printer (or anything in that matter) using compatibility mode, the software must.

1. Write the byte to the Data Port.

2. Check to see is the printer is busy. If the printer is busy, it will not accept any data, thus any data which is written will be lost.

3. Take the Strobe low. This tells the printer that there is the correct data on the data lines.

4. Put the strobe high again after waiting approximately 5 microseconds after putting the strobe low.

4.2.2 Nibble Mode:

Nibble mode can input a nibble (4 bits) in the reverse direction. Say from device to computer. Byte mode uses the Parallel's bi-directional feature (found only on some devices) to input a byte (8 bits) of data in the reverse direction.

4.2.3 Byte Mode:

Byte mode can input 8 bits or a byte at a time. Byte modes use just the standard hardware available on the original Parallel Port cards. Byte mode uses the Parallel's bi-directional feature (found only on some devices) to input a byte (8 bits) of data in the reverse direction.

4.2.4 EPP Mode (Enhanced Parallel Port):

Enhanced Parallel Ports use additional hardware to generate and manage handshaking. The EPP port gets around this by letting the hardware check to see if the printer is busy and generate a strobe and appropriate handshaking. This means only one I/O instruction need to be performed, thus increasing the speed. These ports can output at around 1-2 megabytes per second.

4.2.5 ECP Mode (Extended Capabilities Port):

Extended Capabilities Port also use additional hardware to generate and manage handshaking. The ECP port also has the advantage of using DMA channels and FIFO buffers, thus data can be shifted around without using I/O instructions.

4.3 Hardware Properties

The following is a table of the "Pin Outs" of the D-Type 25 Pin connector and the Centronics 34 Pin connector. The D-Type 25 pin connector is the most common connector found on the Parallel Port of the computer, while the Centronics Connector is commonly found on printers. The IEEE 1284 standard however specifies 3 different connectors for use with the Parallel Port. The first one, 1284 Type A is the D-Type 25 connector found on the back of most computers. The 2nd is the 1284 Type B which is the 36 pin Centronics Connector found on most printers.

IEEE 1284 Type C however, is a 36 conductor connector like the Centronics, but smaller. This connector is claimed to have a better clip latch, better electrical properties and is easier to assemble. It also contains two more pins for signals which can be used to see whether the other device connected,

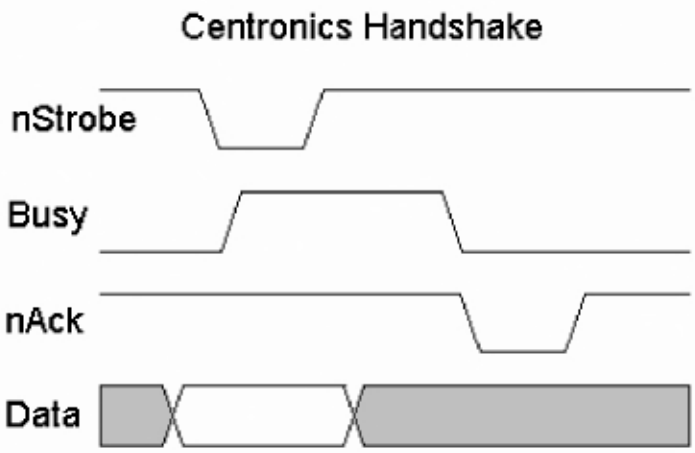
Table 4.3.1 Pin Assignment of D Type 25 Pin Parallel Port Controller

Pin No (D-Type 25)	Pin No (Centronics)	SPP Signal	Direction In/out	Register	Hardware Inverted
1	1	nStrobe	In/Out	Control	Yes
2	2	Data 0	Out	Data	
3	3	Data 1	Out	Data	
4	4	Data 2	Out	Data	
5	5	Data 3	Out	Data	
6	6	Data 4	Out	Data	
7	7	Data 5	Out	Data	
8	8	Data 6	Out	Data	
9	9	Data 7	Out	Data	
10	10	nAck	In	Status	
11	11	Busy	In	Status	Yes
12	12	Paper-Out PaperEnd	In	Status	
13	13	Select	In	Status	
14	14	nAuto-Linefeed	In/Out	Control	Yes
15	32	nError / nFault	In	Status	
16	31	nInitialize	In/Out	Control	
17	36	nSelect-Printer nSelect-In	In/Out	Control	Yes
18 - 25	19-30	Ground	Gnd		

4.4 Centronics

Centronics is an early standard for transferring data from a host to the printer. The majority of printers use this handshake. This handshake is normally implemented using a Standard Parallel Port under software control. Below is a simplified diagram of the 'Centronics' Protocol.

Fig 4.4 Centronics Handshake



Data is first applied on the Parallel Port pins 2 to 7. The host then checks to see if the printer is busy. I.e. the busy line should be low. The program then asserts the strobe, waits a minimum of 1 microsecond, and then de-asserts the strobe. Data is normally read by the printer/peripheral on the rising edge of the strobe. The printer will indicate that it is busy processing data via the busy line. Once the printer has accepted data, it will acknowledge the byte by a negative pulse about 5 microseconds on the nAck line.

4.5 Interfacing (Connecting) the Parallel Port

When the computer is first turned on, BIOS (Basic Input/Output System) will determine the number of ports you have and assign device labels LPT1, LPT2 & LPT3 to them. BIOS first looks at address 3BCh. If a Parallel Port is found here, it is assigned as LPT1, and then it searches at location 378h. If a Parallel card is found there, it is assigned the next free device label. This would be LPT1 if a card wasn't found at 3BCh or LPT2 if a card was found at 3BCh. The last port of call is 278h and follows the same procedure than the other two ports. Therefore it is possible to have a LPT2 which is at 378h
And not at the expected address 278h [36].

The following sample program in C, shows how you can read these locations to obtain the addresses of your printer ports.

```
#include <stdio.h>
#include <dos.h>
void main(void)
{
unsigned int far *ptraddr; /* Pointer to location of Port Addresses */
unsigned int address; /* Address of Port */
int a;
ptraddr=(unsigned int far *)0x00000408;
```

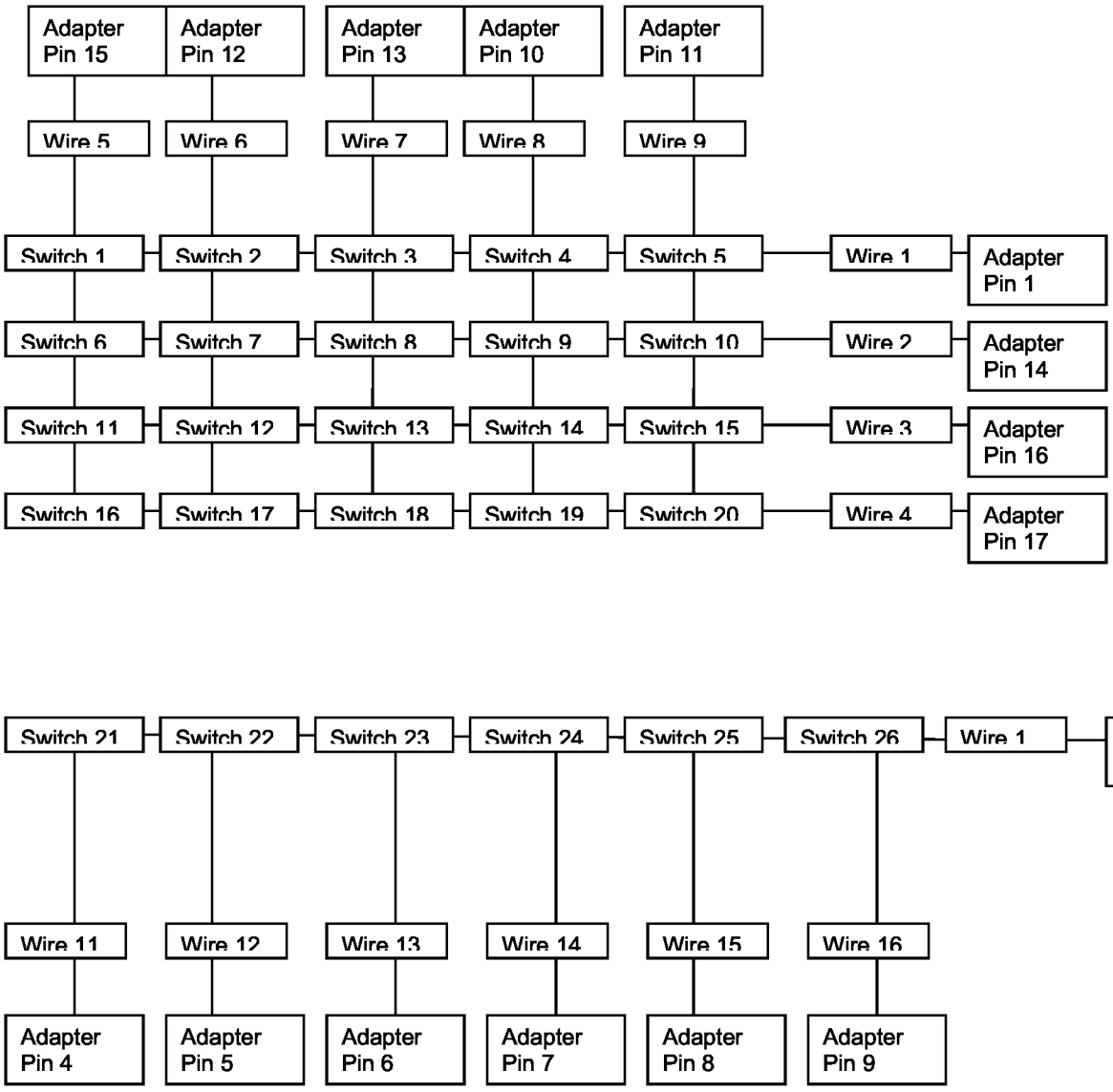
```
for (a = 0; a < 3; a++)
{
address = *ptraddr;
if (address == 0)
printf("No port found for LPT%d \n",a+1);
else
printf("Address assigned to LPT%d is %Xh\n",a+1,address);
*ptraddr++;
}
}
```

4.6 Connecting the Device to the game

After making sure that the computer can connect to the device through the parallel port, I proceeded to connect the device to the game. The name of the game is Puzzle Game. The objective of the game is to arrange all the numbers in a specific order. The game was developer in VC++ and was an MFC application. The game can be played by using arrow keys on the keyboard. So I reprogrammed all the 4 keys to 4 buttons on the gaming controller. Just to show that all the other buttons work as well, I have put a message box for all the buttons in the code to demonstrate all other buttons work.

Following is the schematics diagrams of the device.

Fig 4.6.1 Schematic diagram of the device

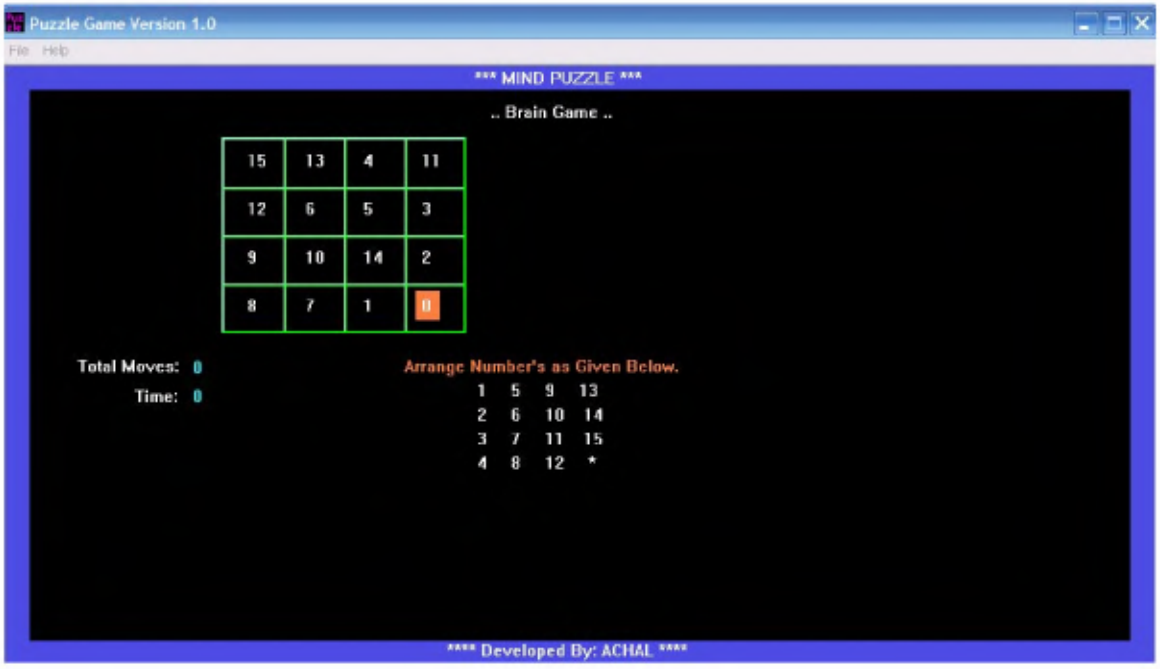


Following is a snapshot of the game before it starts.

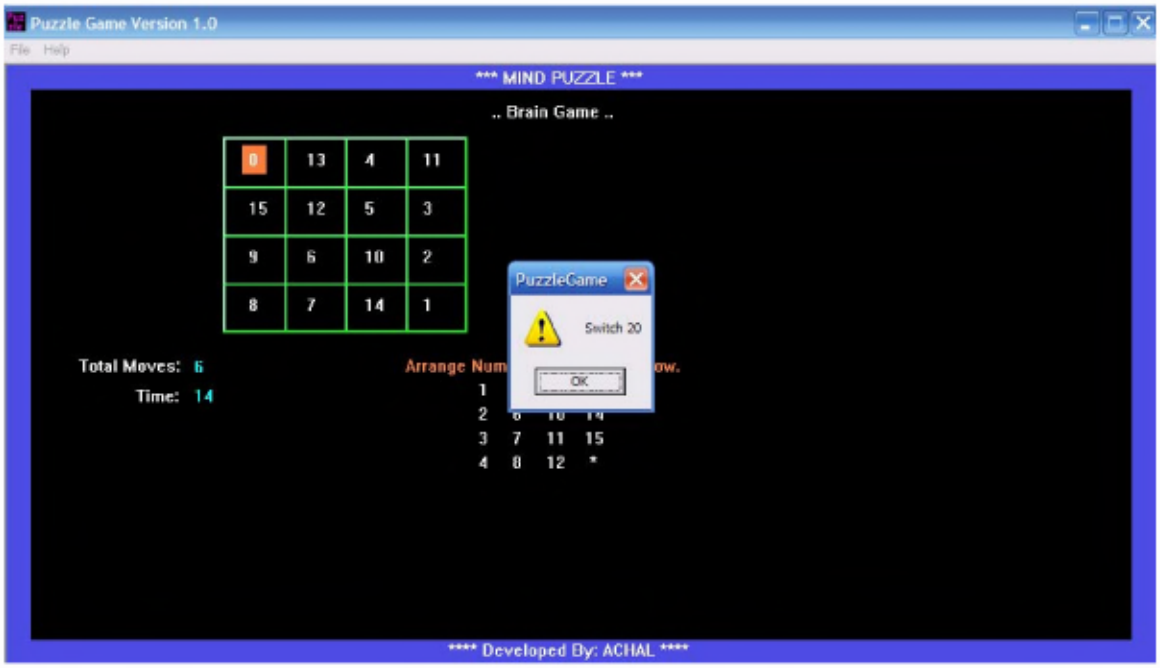
Fig 4.6.2 Snapshot of the game before it starts

This is after I click to start the game.

Fig 4.6.3 Snapshot after the game is started



This is when I click on a button on the game controller.



I put the parallel port communication code in a dll and lib file INPOUT32.dll and INPOUT32.lib. I used LoadLibrary() function to load the dll

CPuzzleGameView::CPuzzleGameView()

{

HINSTANCE hLib;

hLib = LoadLibrary("inpout32.dll");

}

I exported two functions of this dll which are

```
short _stdcall Inp32(short portaddr);

void _stdcall Out32(short portaddr, short datum);
```

I used Inp32() for reading data from port and out32() for writing data from port.The game pad is communicating on Printer port 0x378. I used a function call in OnTimer() in which I check input signal on port in every 100 milisecond. The time period can be changed depending upon how fast or slow one wants the the response of the device to be. I also called a SetTimer() function in:

```
void CPuzzleGameView::OnNewGame()

{

    SetTimer(START_GAME,100,NULL);

}
```

I have tested all 22 buttons and most of the code is written in OnTimer() function.

Following is the snap shot of the construction of gaming controller which my room mate Al helped built:

Fig 4.6.4 Snapshot of Building the device

Fig 4.6.5 Snapshot of constructing the device

Fig 4.6.6 Snapshot of making the device

Fig 4.6.7 The final product

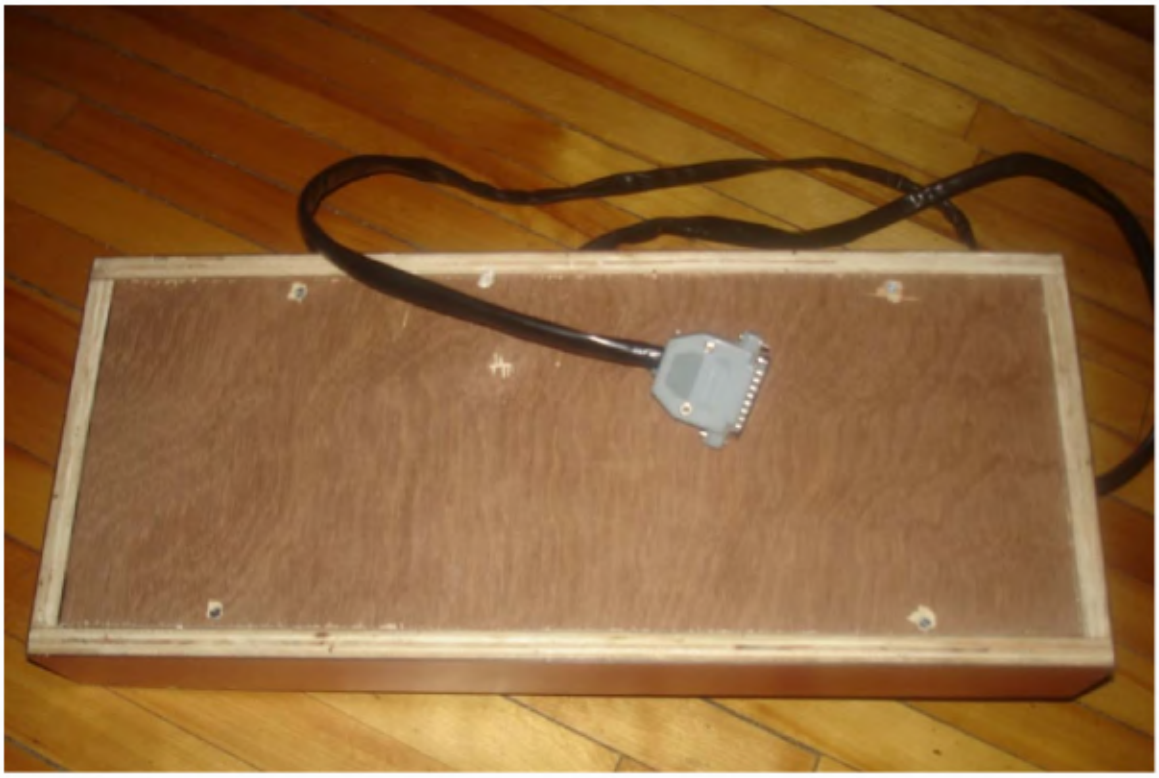
The wires finally coming from the game controller got connected in the parallel port as shown below:

Fig 4.6.8 The parallel Port

Fig 4.6.9 The finished Game Controller:

Fig 4.6.10 Final snapshot of the finished controller

Fig 4.6.11 Final snapshot of the controller



Chapter 5

Future Work

The gaming controller developed by me using the parallel port interface is my first attempt to make a personal game controller. Although it is functional, there are many things that can be done to make it more useful.

Some of them are:

The parallel port supports bi directional flow of data and signals. So it is possible to send signals back to make the gaming controller as well. So features such as having force feed back (starts shaking and vibrating) could be included when ever something major happens in the game.

Although there are 22 buttons and a joystick on the gaming controller, only 4 buttons are used in the game. So in future projects, someone can make use of more controls.

I have connected the gaming controller to a simple game. Anyone wanting to carry the research ahead can use the device to connect it to more sophisticated games. There are many games that give support for external devices. Someone can try to connect it with them.

A device driver for the device would be really helpful. That way the device would be independent from the coding aspect of the game. Also, it would be able to run on any platform. It would also facilitate further development of the device.

Chapter 6

Conclusion

This chapter is the conclusion of the research carried out so far.

Games and gaming devices have truly come from a long way from their humble beginnings. It was fascinating to know about how the video game business started and how it evolved to be a multi billion dollar industry it is today. After working on the parallel port device, I have come to the conclusion that although a lot of research and development has taken place in the software game development, there is still a lot of scope for improvement in the hardware segment.

Companies like Nintendo have come out with innovative products like the Nintendo Wii which has some interesting applications for the gaming hardware. Companies like Microsoft and Sony are also constantly improving their gaming controllers by making them wireless. The newer gaming controllers have also become USB based so that they can connect to both the gaming consoles and the computer at the same time.

Today's games yield more realism than ever. Many have written about the future of gaming predicting which current hardware platform will prevail in the near term or perhaps theorizing how gamers will frolic in a virtual playground in the far out future.

6.1 Future of Gaming and Gaming Devices

If Moore's Law of processor power doubling every 18 months is still in effect, we will be gaming on systems with speeds well over 1,000 Ghz. No matter how far technology advances, certain aspects of gaming will remain constant. The plots and characters of today's role playing games will likely remain. Such elements as the marksmanship of shooters, the thrill of speed in the racing genre, the classic dogfight of flight sims, and the strategizing of the oldest game in the world, chess will continue until our swords are beaten into plowshares. We would expect the basic elements that charm the human heart to remain intact[21].

We're probably a long way away from having a coin-op Holodeck in every truck stop, yet we can all see the day coming when a game character appears as a real human. Ultimately, we will end up with a gaming platform that simulates reality in a very convincing way.

Bibliography

1. First Generation Games
http://en.wikipedia.org/wiki/History_of_video_game_consoles_%28first_generation%29

2. Fifth Generation Games
http://en.wikipedia.org/wiki/History_of_video_game_consoles_%28fifth_generation%29

3. Fourth Generation Games
http://en.wikipedia.org/wiki/History_of_video_game_consoles_%28fourth_generation%29

4. Future of Gaming
http://www.clickfire.com/viewpoints/articles/gaming/gaming_in_the_future.php

5. History of Video Games
http://www.thocp.net/software/games/early_years.htm

6. Gaming Controllers in various platforms
http://www.immersion.com/developer/technology/devices/index.php

7. History of Computer and Video Games
http://en.wikipedia.org/wiki/History_of_computer_and_video_games

8. International Game Technology (IGT), "Introduction to Gaming"
http://media.igt.com/Marketing/PromotionalLiterature/IntroductionToGaming.pdf

9. International Game Technology
http://www.igt.com/Content/base.asp?pid=25.115.273&bhcp=1

10. Michael Dolan, Future of Video Game
http://www.pbs.org/kcts/videogamerevolution/impact/future.html

11. Neurophilosophy, Types of Gaming
http://neurophilosophy.wordpress.com/2007/03/07/mind-control-the-future-of-gaming/

12. Second Generation Games

http://en.wikipedia.org/wiki/History_of_video_game_consoles_%28second_generation%29

13. Sixth Generation Games
http://en.wikipedia.org/wiki/History_of_video_game_consoles_%28sixth_generation%29

14. Seventh Generation Games
http://en.wikipedia.org/wiki/History_of_video_game_consoles_%28seventh_generation%29

15. Third Generation Games
http://en.wikipedia.org/wiki/History_of_video_game_consoles_%28third_generation%29

16. Types of Gaming Devices
http://www.geek.com/news/geeknews/2001feb/gam20010221004445.htm

17. Tayfun King, Challenges in building games and gaming devices
http://www.geek.com/news/geeknews/2001feb/gam20010221004445.htm

18. Video Game Consoles
http://en.wikipedia.org/wiki/Video_game_console

19. Video Game: Cartridge based games
http://www.answers.com/topic/video-game-console

20. Video Game: Card Based
http://spritesmods.com/?art=chip8console&page=2

21. Video Game: Future
http://www.gamepro.com/gamepro/international/games/features/83408.shtml

22. Video Game: Introduction
http://www.electronics-manufacturers.com/info/video-equipment/video-game-console.html

23. Video Game: About Blu – Ray
http://www.blu-ray.com/

24. Video Game : Evolution
http://help.com/wiki/Video_game_console/Media

25. Video Game: Different types
http://www.gamingdump.com/consoleevolution.html

26. Video Games : Internet Distribution
http://www.prnewswire.com/cgi-bin/stories.pl?ACCT=104&STORY=/www/story/05-17-2001/0001496169&EDATE=

27. Video Games: Xbox connected to the internet
http://www.prnewswire.com/cgi-bin/stories.pl?ACCT=104&STORY=/www/story/05-17-2001/0001496169&EDATE=

28. Video Games: Features of Xbox and PS2
http://www.galaxine.com/Xbox-Game-Consoles.htm

29. Video Games: Handheld Devices
http://en.wikipedia.org/wiki/Handheld_device

30. Video Games: Handheld Devices 2
http://portables.about.com/od/gamingdevices/Handheld_Gaming_Devices.htm

31. Video Games: Nokia N Gage
http://en.wikipedia.org/wiki/N-Gage

32. Video Games: Nokia N Gage 2
http://www.n-gage.com/

33. Video Games: COM Port Devices
http://tldp.org/HOWTO/Modem-HOWTO.html#toc9

34. Video Games: Parallel Port
http://www.neurobs.com/pres_docs/html/08_hardware_interfacing/01_ports/01_port_devices.htm

35. Video Game: Parallel Port Pin Diagram
http://pinouts.ru/connector/25_pin_D-SUB_male_connector.shtml

36. Video Game: Construction of Parallel Port
http://www.tkk.fi/Misc/Electronics/circuits/lptpower.html

37. Video Game: Information of Parallel Port
http://books.google.com/books?hl=en&lr=&id=hjEAE9BMaYQC&oi=fnd&pg=PR9&sig=jTjH5koA6l3sLd7IximqN2o3Fa0&dq=PARALLEL+port+devices+pin+diagram#PPR5,M1

Appendix A

Source Code of Puzzle Game in Microsoft Visual C++

```
//coded by achal
//MainDlg.cpp : implementation file
//

#include "stdafx.h"
#include "PuzzleGame.h"
#include "MainDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// MainDlg dialog


MainDlg::MainDlg(CWnd* pParent /*=NULL*/)
```

```
    : CDialog(MainDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(MainDlg)
            // NOTE: the ClassWizard will add member initialization here
    //}}AFX_DATA_INIT
}


void MainDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(MainDlg)
            // NOTE: the ClassWizard will add DDX and DDV calls here
    //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(MainDlg, CDialog)
    //{{AFX_MSG_MAP(MainDlg)
    ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

```
////////////////////////////////////////////////////////////////////////////

// MainDlg message handlers


void MainDlg::OnButton1()

{

    OnOK();


}


//coded by achal

//MainDlg.h  header file

#if !defined(AFX_MAINDLG_H__24E70ED4_F618_11D8_8351_B19D0C604C2D

__INCLUDED_)

#define

AFX_MAINDLG_H__24E70ED4_F618_11D8_8351_B19D0C604C2D__INCLUDE

D_


#if _MSC_VER > 1000

#pragma once

#endif // _MSC_VER > 1000

// MainDlg.h : header file

//
```

```
/////////////////////////////////////////////////////////////////////////////
// MainDlg dialog

class MainDlg : public CDialog
{
// Construction
public:
    MainDlg(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
    //{{AFX_DATA(MainDlg)
    enum { IDD = IDD_MAINDIALOG };
            // NOTE: the ClassWizard will add data members here
    //}}AFX_DATA


// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(MainDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL
```

```
// Implementation
protected:

    // Generated message map functions
    //{{AFX_MSG(MainDlg)
    afx_msg void OnButton1();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif
// !defined(AFX_MAINDLG_H__24E70ED4_F618_11D8_8351_B19D0C604C2D_
_INCLUDED_)

//coded by achal
// MainFrm.cpp : implementation of the CMainFrame class
//

#include "stdafx.h"
```

```
#include "PuzzleGame.h"
#include "MainFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CMainFrame

IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
    //{{AFX_MSG_MAP(CMainFrame)
    ON_WM_CREATE()
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction
```

```
CMainFrame::CMainFrame()
{
    // TODO: add member initialization code here

}

CMainFrame::~CMainFrame()
{
}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    if( !CFrameWnd::PreCreateWindow(cs) )
            return FALSE;
    // TODO: Modify the Window class or styles here by modifying
    //  the CREATESTRUCT cs

    cs.style = WS_OVERLAPPED | WS_CAPTION | FWS_ADDTOTITLE
            | WS_SYSMENU | WS_MINIMIZEBOX;

    cs.style &= ~(LONG) FWS_ADDTOTITLE;

    return TRUE;
```

```
}

/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
    CFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
    CFrameWnd::Dump(dc);
}

#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CMainFrame message handlers


int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
```

```
{
    if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
            return -1;


    SetWindowText("Puzzle Game Version 1.0");

    HICON h_icon;
    h_icon = AfxGetApp()->LoadIcon(IDI_ICON1);
    CMainFrame::SendMessage(WM_SETICON,TRUE,(LPARAM)h_icon);

    return 0;
}

//coded by achal
// MainFrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_MAINFRM_H__24E70EC9_F618_11D8_8351_B19D0C604C2D
__INCLUDED_)
```

```
#define
AFX_MAINFRM_H__24E70EC9_F618_11D8_8351_B19D0C604C2D__INCLUD
ED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

class CMainFrame : public CFrameWnd
{

protected: // create from serialization only
    CMainFrame();
    DECLARE_DYNCREATE(CMainFrame)

// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
```

```
    //{{AFX_VIRTUAL(CMainFrame)
    public:
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CMainFrame();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

// Generated message map functions
protected:
    //{{AFX_MSG(CMainFrame)
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////
```

```
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif
// !defined(AFX_MAINFRM_H__24E70EC9_F618_11D8_8351_B19D0C604C2D_
_INCLUDED_)



//coded by achal
// PuzzleGame.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "PuzzleGame.h"

#include "MainFrm.h"
#include "PuzzleGameDoc.h"
#include "PuzzleGameView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
```

```
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CPuzzleGameApp

BEGIN_MESSAGE_MAP(CPuzzleGameApp, CWinApp)
    //{{AFX_MSG_MAP(CPuzzleGameApp)
    ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
            // NOTE - the ClassWizard will add and remove mapping macros here.
            //    DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG_MAP
    // Standard file based document commands
    ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
    ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CPuzzleGameApp construction

CPuzzleGameApp::CPuzzleGameApp()
{
    // TODO: add construction code here,
```

```
    // Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CPuzzleGameApp object

CPuzzleGameApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CPuzzleGameApp initialization

BOOL CPuzzleGameApp::InitInstance()
{
    AfxEnableControlContainer();

    // Standard initialization
    // If you are not using these features and wish to reduce the size
    //  of your final executable, you should remove from the following
    //  the specific initialization routines you do not need.

#ifdef _AFXDLL
    Enable3dControls();                 // Call this when using MFC in a shared
DLL
```

```
#else
    Enable3dControlsStatic();      // Call this when linking to MFC statically
#endif

    // Change the registry key under which our settings are stored.
    // TODO: You should modify this string to be something appropriate
    // such as the name of your company or organization.
    SetRegistryKey(_T("Local AppWizard-Generated Applications"));

    LoadStdProfileSettings();  // Load standard INI file options (including MRU)

    // Register the application's document templates.  Document templates
    //  serve as the connection between documents, frame windows and views.

    CSingleDocTemplate* pDocTemplate;
    pDocTemplate = new CSingleDocTemplate(
            IDR_MAINFRAME,
            RUNTIME_CLASS(CPuzzleGameDoc),
            RUNTIME_CLASS(CMainFrame),      // main SDI frame window
            RUNTIME_CLASS(CPuzzleGameView));
    AddDocTemplate(pDocTemplate);

    // Parse command line for standard shell commands, DDE, file open
```

```
    CCommandLineInfo cmdInfo;
    ParseCommandLine(cmdInfo);

    // Dispatch commands specified on the command line
    if (!ProcessShellCommand(cmdInfo))
        return FALSE;

    // The one and only window has been initialized, so show and update it.
    m_pMainWnd->ShowWindow(SW_SHOW);
    m_pMainWnd->UpdateWindow();

    return TRUE;
}


//coded by achal
// PuzzleGame.h : main header file for the PUZZLEGAME application
//

#if !defined(AFX_PUZZLEGAME_H__24E70EC5_F618_11D8_8351_B19D0C604
C2D__INCLUDED_)
```

```
#define
AFX_PUZZLEGAME_H__24E70EC5_F618_11D8_8351_B19D0C604C2D__INCL
UDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#ifndef __AFXWIN_H__
    #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"       // main symbols

/////////////////////////////////////////////////////////////////////////////
// CPuzzleGameApp:
// See PuzzleGame.cpp for the implementation of this class
//

class CPuzzleGameApp : public CWinApp
{
public:
    CPuzzleGameApp();
```

```
// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CPuzzleGameApp)
    public:
    virtual BOOL InitInstance();
    //}}AFX_VIRTUAL

// Implementation
    //{{AFX_MSG(CPuzzleGameApp)
    afx_msg void OnAppAbout();
            // NOTE - the ClassWizard will add and remove member functions here.
            //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.
```

```
#endif
// !defined(AFX_PUZZLEGAME_H__24E70EC5_F618_11D8_8351_B19D0C604C
2D__INCLUDED_)


//coded by achal
// PuzzleGameDoc.cpp : implementation of the CPuzzleGameDoc class
//

#include "stdafx.h"
#include "PuzzleGame.h"

#include "PuzzleGameDoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CPuzzleGameDoc
```

```
IMPLEMENT_DYNCREATE(CPuzzleGameDoc, CDocument)


BEGIN_MESSAGE_MAP(CPuzzleGameDoc, CDocument)
    //{{AFX_MSG_MAP(CPuzzleGameDoc)
            // NOTE - the ClassWizard will add and remove mapping macros here.
            //    DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////////////
// CPuzzleGameDoc construction/destruction


CPuzzleGameDoc::CPuzzleGameDoc()
{
    // TODO: add one-time construction code here


}


CPuzzleGameDoc::~CPuzzleGameDoc()
{
}
```

```
BOOL CPuzzleGameDoc::OnNewDocument()
{
    if (!CDocument::OnNewDocument())
            return FALSE;

    // TODO: add reinitialization code here
    // (SDI documents will reuse this document)

    return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CPuzzleGameDoc serialization

void CPuzzleGameDoc::Serialize(CArchive& ar)
{
    if (ar.IsStoring())
    {
            // TODO: add storing code here
    }
    else
```

```
    {
        // TODO: add loading code here
    }
}

/////////////////////////////////////////////////////////////////////////////
// CPuzzleGameDoc diagnostics

#ifdef _DEBUG
void CPuzzleGameDoc::AssertValid() const
{
    CDocument::AssertValid();
}

void CPuzzleGameDoc::Dump(CDumpContext& dc) const
{
    CDocument::Dump(dc);
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CPuzzleGameDoc commands
```

```
//coded by achal
// PuzzleGameDoc.h : interface of the CPuzzleGameDoc class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_PUZZLEGAMEDOC_H__24E70ECB_F618_11D8_8351_B19D0
C604C2D__INCLUDED_)
#define
AFX_PUZZLEGAMEDOC_H__24E70ECB_F618_11D8_8351_B19D0C604C2D__
INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000


class CPuzzleGameDoc : public CDocument
{
protected: // create from serialization only
    CPuzzleGameDoc();
    DECLARE_DYNCREATE(CPuzzleGameDoc)
```

```
// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CPuzzleGameDoc)
    public:
    virtual BOOL OnNewDocument();
    virtual void Serialize(CArchive& ar);
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CPuzzleGameDoc();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:
```

```
// Generated message map functions
protected:
    //{{AFX_MSG(CPuzzleGameDoc)
            // NOTE - the ClassWizard will add and remove member functions here.
            //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif
// !defined(AFX_PUZZLEGAMEDOC_H__24E70ECB_F618_11D8_8351_B19D0C
604C2D__INCLUDED_)


//coded by achal
// PuzzleGameView.cpp : implementation of the CPuzzleGameView class
```

```
//

#include "stdafx.h"
#include "PuzzleGame.h"

#include "PuzzleGameDoc.h"
#include "PuzzleGameView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

#define START_GAME 10

#define DATA 0x378
#define STATUS 0x379
#define CONTROL 0x37a

short _stdcall Inp32(short portaddr);
void _stdcall Out32(short portaddr, short datum);
```

```
/////////////////////////////////////////////////////////////////////////////

// CPuzzleGameView

IMPLEMENT_DYNCREATE(CPuzzleGameView, CView)

BEGIN_MESSAGE_MAP(CPuzzleGameView, CView)
    //{{AFX_MSG_MAP(CPuzzleGameView)
    ON_WM_KEYDOWN()
    ON_WM_CREATE()
    ON_WM_TIMER()
    ON_COMMAND(ID_NEW_GAME, OnNewGame)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////

// CPuzzleGameView construction/destruction

CPuzzleGameView::CPuzzleGameView()
{
    count = 0;
    moves = 0;
    cur_x = 3;
    cur_y = 3;
```

```
    test = FALSE;
    GameStart = FALSE;
    Initialize();

    HINSTANCE hLib;
    hLib = LoadLibrary("inpout32.dll");
}

CPuzzleGameView::~CPuzzleGameView()
{
}

BOOL CPuzzleGameView::PreCreateWindow(CREATESTRUCT& cs)
{

    return CView::PreCreateWindow(cs);
}

/////////////////////////////////////////////////////////////////////////////
// CPuzzleGameView drawing

void CPuzzleGameView::OnDraw(CDC* pDC)
{
```

```
    CPuzzleGameDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    if(GameStart == TRUE)
    {
            DrawScreen(pDC);
            Time_Count();
            Move_Count();
            Draw_Box();
            Draw_Numbers();
    }
    else
            DrawAbout(pDC);
}

/////////////////////////////////////////////////////////////////////////////
// CPuzzleGameView diagnostics

#ifdef _DEBUG
void CPuzzleGameView::AssertValid() const
{
    CView::AssertValid();
}
```

```
void CPuzzleGameView::Dump(CDumpContext& dc) const
{
    CView::Dump(dc);
}

CPuzzleGameDoc* CPuzzleGameView::GetDocument() // non-debug version is
inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CPuzzleGameDoc)));
    return (CPuzzleGameDoc*)m_pDocument;
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CPuzzleGameView message handlers

void CPuzzleGameView::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{

  if(GameStart == TRUE)
  {
    if( nChar == 37 )
```

```
{
        if(cur_x!=0)
        {
                dum = arr[cur_y][cur_x-1];
                arr[cur_y][cur_x-1] = arr[cur_y][cur_x];
                arr[cur_y][cur_x] = dum;
                cur_x--;
                InvalidateRect(CRect(180,60,380,220),FALSE);
                moves++;
        }
}

if( nChar == 39 )
{
        if(cur_x<3)
        {
                dum = arr[cur_y][cur_x+1];
                arr[cur_y][cur_x+1] = arr[cur_y][cur_x];
                arr[cur_y][cur_x] = dum;
                cur_x++;
                InvalidateRect(CRect(180,60,380,220),FALSE);
                moves++;
        }
```

```
}

if( nChar == 38 )
{
        if(cur_y!=0)
        {
                dum = arr[cur_y-1][cur_x];
                arr[cur_y-1][cur_x] = arr[cur_y][cur_x];
                arr[cur_y][cur_x] = dum;
                cur_y--;
                InvalidateRect(CRect(180,60,380,220),FALSE);
                moves++;
        }
}


if( nChar == 40 )
{

        if(cur_y<3)
        {
                dum = arr[cur_y+1][cur_x];
                arr[cur_y+1][cur_x] = arr[cur_y][cur_x];
```

```
                arr[cur_y][cur_x] = dum;
                cur_y++;
                InvalidateRect(CRect(180,60,380,220),FALSE);
                moves++;
        }
}

InvalidateRect(CRect(154,240,190,255),FALSE);

for(int i=0;i<=3;i++)
{
        for(int j=0;j<=3;j++)
        {
                if(arr[i][j] != chk_arr[i][j])
                {
                        test = FALSE;
                        break;
                }
        }
}

if( test == TRUE )
{
```

```
            KillTimer(1);
            MessageBox("   YOU WIN !!!!!   ","Puzzle
Game",MB_ICONINFORMATION);
            GameStart = FALSE;
            Invalidate(TRUE);
    }

    if(test == FALSE)
            test = TRUE;
  }
  else if(GameStart == FALSE)
  {
      OnNewGame();
  }

    CView::OnKeyDown(nChar, nRepCnt, nFlags);
}

int CPuzzleGameView::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CView::OnCreate(lpCreateStruct) == -1)
            return -1;
```

```
    return 0;
}

void CPuzzleGameView::OnTimer(UINT nIDEvent)
{

    if(nIDEvent==START_GAME)
    {

            int statusreg,controlreg;
            Out32(CONTROL,0x0FC);
            controlreg=Inp32(CONTROL);
            statusreg=Inp32(STATUS);
            int datareg;
            datareg=Inp32(DATA);


            if(((datareg & 0x02) == 0) || ((datareg & 0x80) == 0)) // For Switch no.26
            {
                    ::AfxMessageBox("Switch26");
                    //Right movement
                    if(cur_x<3)
                    {
```

```
                dum = arr[cur_y][cur_x+1];
                arr[cur_y][cur_x+1] = arr[cur_y][cur_x];
                arr[cur_y][cur_x] = dum;
                cur_x++;
                InvalidateRect(CRect(180,60,380,220),FALSE);
                moves++;
            }

        }

        else if(((datareg & 0x02) == 0) || ((datareg & 0x40) == 0)) // For Switch
no.25
        {
            ::AfxMessageBox("Switch25");
            // Left movement
            if(cur_x!=0)
            {
                dum = arr[cur_y][cur_x-1];
                arr[cur_y][cur_x-1] = arr[cur_y][cur_x];
                arr[cur_y][cur_x] = dum;
                cur_x--;
                InvalidateRect(CRect(180,60,380,220),FALSE);
                moves++;
```

```
            }

        }

        else if(((datareg & 0x02) == 0) || ((datareg & 0x20) == 0)) // For Switch
no.24
        {
            ::AfxMessageBox("Switch24");
            // Down movement
            if(cur_y<3)
            {
                dum = arr[cur_y+1][cur_x];
                arr[cur_y+1][cur_x] = arr[cur_y][cur_x];
                arr[cur_y][cur_x] = dum;
                cur_y++;
                InvalidateRect(CRect(180,60,380,220),FALSE);
                moves++;
            }

        }

        else if(((datareg & 0x02) == 0) || ((datareg & 0x10) == 0)) // For Switch
no.23
```

```
{

        ::AfxMessageBox("Switch23");
        // Up movement
        if(cur_y!=0)
        {
                dum = arr[cur_y-1][cur_x];
                arr[cur_y-1][cur_x] = arr[cur_y][cur_x];
                arr[cur_y][cur_x] = dum;
                cur_y--;
                InvalidateRect(CRect(180,60,380,220),FALSE);
                moves++;
        }


}

        else if(((datareg & 0x02) == 0) || ((datareg & 0x08) == 0)) // For Switch
no.22
        {
                ::AfxMessageBox("Switch 22");
                // Button(Bet) Right movement
                if(cur_x<3)
```

```
            {
                dum = arr[cur_y][cur_x+1];
                arr[cur_y][cur_x+1] = arr[cur_y][cur_x];
                arr[cur_y][cur_x] = dum;
                cur_x++;
                InvalidateRect(CRect(180,60,380,220),FALSE);
                moves++;
            }

        }


        else if(((datareg & 0x01) == 0) || ((datareg & 0x04) == 0)) // For Switch
no.21
        {
            ::AfxMessageBox("Switch 21");
            // Button(Deal) Down movement
            if(cur_y<3)
            {
                dum = arr[cur_y+1][cur_x];
                arr[cur_y+1][cur_x] = arr[cur_y][cur_x];
                arr[cur_y][cur_x] = dum;
                cur_y++;
```

```
                InvalidateRect(CRect(180,60,380,220),FALSE);
                moves++;
        }

}

else if((statusreg & 0x80) == 0x80) //For Switch no.20
{
        ::AfxMessageBox("Switch 20");
        // Button(Cancel) Left movement
        if(cur_x!=0)
        {
                dum = arr[cur_y][cur_x-1];
                arr[cur_y][cur_x-1] = arr[cur_y][cur_x];
                arr[cur_y][cur_x] = dum;
                cur_x--;
                InvalidateRect(CRect(180,60,380,220),FALSE);
                moves++;
        }

}

else if((statusreg & 0x40) == 0) // For Switch no.19
```

```
{
        ::AfxMessageBox("Switch 19");
        // D-Up Button

}

else if((statusreg & 0x20) == 0) // For Switch no.17
{
        //::AfxMessageBox("Switch 17");
        // Bet Button
        // Down movement
        if(cur_y<3)
        {
                dum = arr[cur_y+1][cur_x];
                arr[cur_y+1][cur_x] = arr[cur_y][cur_x];
                arr[cur_y][cur_x] = dum;
                cur_y++;
                InvalidateRect(CRect(180,60,380,220),FALSE);
                moves++;
        }

}
```

```
else if((statusreg & 0x10) == 0) // For Switch no.18
{
        //::AfxMessageBox("Switch 18");
        // Small Button
        // Button(Bet) Right movement
        if(cur_x<3)
        {
                dum = arr[cur_y][cur_x+1];
                arr[cur_y][cur_x+1] = arr[cur_y][cur_x];
                arr[cur_y][cur_x] = dum;
                cur_x++;
                InvalidateRect(CRect(180,60,380,220),FALSE);
                moves++;
        }
}

else if((statusreg & 0x08) == 0) // For Switch no.16
{
        //::AfxMessageBox("Switch 16");
        // Take Button
        // Left movement
        if(cur_x!=0)
        {
```

```
                dum = arr[cur_y][cur_x-1];
                arr[cur_y][cur_x-1] = arr[cur_y][cur_x];
                arr[cur_y][cur_x] = dum;
                cur_x--;
                InvalidateRect(CRect(180,60,380,220),FALSE);
                moves++;
        }

}

//-----Switch 11-15------//
//      int statusregp,controlregp;

Out32(CONTROL, 0x0E0);
controlreg=Inp32(CONTROL);
statusreg=Inp32(STATUS);


if((statusreg & 0x08) == 0) // For Switch no.11
{
        //::AfxMessageBox("Switch 11");
        // Hold1  Button
        //up movement
```

```
        if(cur_y!=0)
        {
                dum = arr[cur_y-1][cur_x];
                arr[cur_y-1][cur_x] = arr[cur_y][cur_x];
                arr[cur_y][cur_x] = dum;
                cur_y--;
                InvalidateRect(CRect(180,60,380,220),FALSE);
                moves++;
        }

}

else if((statusreg & 0x20) == 0) // For Switch no.12
{
        ::AfxMessageBox("Switch 12");
        // Hold2  Button

}

else if((statusreg & 0x10) == 0) // For Switch no.13
{
        ::AfxMessageBox("Switch 13");
        // Hold3  Button
```

```
}

else if((statusreg & 0x40) == 0) // For Switch no.14
{
        ::AfxMessageBox("Switch 14");
        // Hold4  Button

}

else if((statusreg & 0x80) == 0x80) // For Switch no.15
{
        ::AfxMessageBox("Switch 15");
        // Button(Hold5) Up movement
        if(cur_y!=0)
        {
                dum = arr[cur_y-1][cur_x];
                arr[cur_y-1][cur_x] = arr[cur_y][cur_x];
                arr[cur_y][cur_x] = dum;
                cur_y--;
                InvalidateRect(CRect(180,60,380,220),FALSE);
                moves++;
        }
```

```
}

//-----Switch 1-5------//
//        int statusregp,controlregp;

Out32(CONTROL, 0x0F5);
controlreg=Inp32(CONTROL);
statusreg=Inp32(STATUS);

if((statusreg & 0x08) == 0) // For Switch no.1
{
        ::AfxMessageBox("Switch 1");
        // 1- Button

}
else if((statusreg & 0x20) == 0) // For Switch no.2
{
        ::AfxMessageBox("Switch 2");
        // 2- Button

}
else if((statusreg & 0x10) == 0) // For Switch no.3
```

```
{
        ::AfxMessageBox("Switch 3");
        // 3- Button

}
else if((statusreg & 0x40) == 0) // For Switch no.4
{
        ::AfxMessageBox("Switch 4");
        // 4- Button

}
else if((statusreg & 0x80) == 0x80) // For Switch no.5
{
        ::AfxMessageBox("Switch 5");
        // 5- Button

}

//-----Switch 6-10------//
//      int statusregp,controlregp;

Out32(CONTROL, 0x0E6);
controlreg=Inp32(CONTROL);
```

```
statusreg=Inp32(STATUS);


if((statusreg & 0x08) == 0) // For Switch no.6
{
        ::AfxMessageBox("Switch 6");
        // 6- Button

}
else if((statusreg & 0x20) == 0) // For Switch no.7
{
        ::AfxMessageBox("Switch 7");
        // 7- Button

}
else if((statusreg & 0x10) == 0) // For Switch no.8
{
        ::AfxMessageBox("Switch 8");
        // 8- Button

}
else if((statusreg & 0x40) == 0) // For Switch no.9
{
        ::AfxMessageBox("Switch 9");
```

```
        // 9- Button

}
else if((statusreg & 0x80) == 0x80) // For Switch no.10
{
        ::AfxMessageBox("Switch 10");
        // 10- Button

}

InvalidateRect(CRect(154,240,190,255),FALSE);

for(int i=0;i<=3;i++)
{
        for(int j=0;j<=3;j++)
        {
                if(arr[i][j] != chk_arr[i][j])
                {
                        test = FALSE;
                        break;
                }
        }
```

```
            }

            if( test == TRUE )
            {
                    KillTimer(1);
                    MessageBox("   YOU WIN !!!!!   ","Puzzle
Game",MB_ICONINFORMATION);
                    GameStart = FALSE;
                    Invalidate(TRUE);
            }

            if(test == FALSE)
                    test = TRUE;

    }
    else if(GameStart == FALSE)
   {
      OnNewGame();
   }

    if( nIDEvent == 1 )
    {
```

```
            Time_Count();
            count++;
            InvalidateRect(CRect(154,264,190,280),FALSE);
    }

    CView::OnTimer(nIDEvent);
}


void CPuzzleGameView::Initialize()
{
    int i,j,n,equal,p[16];
    int dup2[4][4]={{1,5,9,13},{2,6,10,14},{3,7,11,15},{4,8,12,0}};

    for(i=0;i<=3;i++)
    {
            for(j=0;j<=3;j++)
            {
                    chk_arr[i][j] = dup2[i][j];
            }
    }

    for(i=0;i<=15;i++)
```

```
        p[i] = 0;


for(i=0;i<=14;)
{
        n = rand( ) % 16 ;


        if ( n == 0 )
                continue ;


        equal = 0 ;
        for (j=0;j<i;j++)
        {
                if( p[j] == n )
                {
                        equal = 1 ;
                        break ;
                }
        }


        if( equal != 1 )
        {
                p[i] = n ;
                i++ ;
```

```
            }
        }

        for(i=0;i<=3;i++)
        {
            for(j=0;j<=3;j++)
            {
                arr[i][j] = p[i*4+j] ;
            }
        }
}

void CPuzzleGameView::DrawScreen(CDC *pDC)
{
    CRect r;
    CBrush pb;
    GetClientRect(r);
    pb.CreateSolidBrush(RGB(0,0,0));

    pDC->FillRect(r,&pb);

    pb.DeleteObject();
```

```
pb.CreateSolidBrush(RGB(77,77,230));

pDC->FillRect(CRect(0,0,r.right ,20),&pb);

pDC->FillRect(CRect(0,0,20 ,r.bottom),&pb);

pDC->FillRect(CRect(0,r.bottom - 20,r.right ,r.bottom),&pb);

pDC->FillRect(CRect(r.right - 20,0,r.right ,r.bottom),&pb);

pDC->SetBkMode(TRANSPARENT);
pDC->SetTextColor(RGB(255,255,255));
pDC->TextOut((r.right /2) - 87,2,"*** MIND PUZZLE ***");
pDC->SetTextColor(RGB(255,255,255));
pDC->TextOut((r.right /2) - 78,30," .. Brain Game .. ");
pDC->SetTextColor(RGB(255,255,255));
pDC->TextOut((r.right /2) - 110,(r.bottom -  20),"**** Developed By: ACHAL
****");

pDC->SetTextColor(RGB(255,128,64));
pDC->TextOut(330,240,"Arrange Number's as Given Below.");
pDC->SetTextColor(RGB(255,255,255));
pDC->TextOut(390,260,"1    5    9    13");
```

```
    pDC->TextOut(390,280,"2    6    10    14");
    pDC->TextOut(390,300,"3    7    11    15");
    pDC->TextOut(390,320,"4    8    12     *");

    pDC->TextOut(60,240,"Total Moves:");
    pDC->TextOut(107,265,"Time:");
}

void CPuzzleGameView::DrawAbout(CDC *pDC)
{
    CRect r;
    CBrush pb;
    GetClientRect(r);
    pb.CreateSolidBrush(RGB(0,0,0));

    pDC->FillRect(r,&pb);

    pb.DeleteObject();

    pb.CreateSolidBrush(RGB(77,77,230));

    pDC->FillRect(CRect(0,0,r.right ,20),&pb);
```

```
    pDC->FillRect(CRect(0,0,20 ,r.bottom),&pb);

    pDC->FillRect(CRect(0,r.bottom - 20,r.right ,r.bottom),&pb);

    pDC->FillRect(CRect(r.right - 20,0,r.right ,r.bottom),&pb);

    pDC->SetBkMode(TRANSPARENT);
    pDC->SetTextColor(RGB(255,255,255));
    pDC->TextOut((r.right /2) - 87,2,"*** MIND PUZZLE ***");
    pDC->SetTextColor(RGB(255,255,255));
    pDC->TextOut((r.right /2) - 78,30," .. Brain Game .. ");
    pDC->SetTextColor(RGB(255,255,255));
    pDC->TextOut((r.right /2) - 110,(r.bottom -  20),"**** Developed By: ACHAL
****");
    pDC->SetTextColor(RGB(10,245,10));
    pDC->TextOut((r.right/2) - 100,110,"Puzzle  Game  Version  1.0");
    pDC->TextOut((r.right/2) - 110,135,"Copyright (C) 2007 By ACHAL.");
    pDC->TextOut((r.right/2) - 80,160,"All Rights Reserved.");
    pDC->SetTextColor(RGB(255,255,55));
    pDC->TextOut((r.right/2) - 105,290,"Press any Key to Start Game....");
}

void CPuzzleGameView::Draw_Box()
```

```
{
    CClientDC dc(this);

    dc.Draw3dRect(CRect(180,60,380,220),RGB(150,255,190),RGB(0,255,0));
    dc.Draw3dRect(CRect(179,59,381,221),RGB(150,255,190),RGB(0,255,0));

    dc.Draw3dRect(CRect(180,100,380,102),RGB(150,255,190),RGB(0,255,0));
    dc.Draw3dRect(CRect(180,140,380,142),RGB(150,255,190),RGB(0,255,0));
    dc.Draw3dRect(CRect(180,180,380,182),RGB(150,255,190),RGB(0,255,0));

    dc.Draw3dRect(CRect(230,60,232,220),RGB(150,255,190),RGB(0,255,0));
    dc.Draw3dRect(CRect(280,60,282,220),RGB(150,255,190),RGB(0,255,0));
    dc.Draw3dRect(CRect(330,60,332,220),RGB(150,255,190),RGB(0,255,0));
}

void CPuzzleGameView::Draw_Numbers()
{
    int i,j,x,y;
    CClientDC dc(this);
    CBrush pb;
    CString str;

    x = 200; y = 70;
```

```
    pb.CreateSolidBrush(RGB(255,128,64));

    dc.SetBkMode(TRANSPARENT);
    dc.SetTextColor(RGB(255,255,255));

    for(i=0;i<=3;i++)
    {
        for(j=0;j<=3;j++)
        {
         str.Format("%d",arr[i][j]);
         if(arr[i][j] == 0)
         {
            dc.FillRect(CRect(x-5,y-4,x+15,y+20),&pb);
            dc.TextOut(x,y,str);
         }
         dc.TextOut(x,y,str);
         x+=48;
        }
        y+=40;
        x=200;
    }
}
```

```
void CPuzzleGameView::Time_Count()
{
    CClientDC dc(this);
    dc.SetBkMode(TRANSPARENT);
    dc.SetTextColor(RGB(23,255,255));
    dup.Format("%d",count);
    dc.TextOut(155,265,dup);
}

void CPuzzleGameView::Move_Count()
{
    CClientDC dc(this);
    dc.SetBkMode(TRANSPARENT);
    dc.SetTextColor(RGB(23,255,255));
    dup.Format("%d",moves);
    dc.TextOut(155,241,dup);
}

void CPuzzleGameView::OnNewGame()
{
    KillTimer(1);
```

```
    KillTimer(START_GAME);


    count=0;
    moves=0;
    cur_x=3;cur_y=3;
    Initialize();
    GameStart = TRUE;
    Draw_Box();
    Draw_Numbers();
    Invalidate(TRUE);
    SetTimer(1,1000,NULL);
    SetTimer(START_GAME,100,NULL);
}



//coded by achal
// PuzzleGameView.h : interface of the CPuzzleGameView class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_PUZZLEGAMEVIEW_H__24E70ECD_F618_11D8_8351_B19D
0C604C2D__INCLUDED_)
```

```
#define
AFX_PUZZLEGAMEVIEW_H__24E70ECD_F618_11D8_8351_B19D0C604C2D_
_INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000


class CPuzzleGameView : public CView
{
protected: // create from serialization only
    CPuzzleGameView();
    DECLARE_DYNCREATE(CPuzzleGameView)

// Attributes
public:
    CPuzzleGameDoc* GetDocument();

// Operations
public:

// Overrides
```

```
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CPuzzleGameView)
    public:
    virtual void OnDraw(CDC* pDC);  // overridden to draw this view
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    protected:
    //}}AFX_VIRTUAL

// Implementation
public:
    void DrawAbout(CDC *pDC);
    void DrawScreen(CDC *pDC);
    void Move_Count();
    void Draw_Numbers();
    void Draw_Box();
    int chk_arr[4][4];
    int bak[4][4];
    int cur_x,cur_y,dum;
    void Initialize();
    int arr[4][4];
    void Time_Count();
    unsigned long int count,moves;
    CString dup;
```

```
    BOOL test,GameStart;
    virtual ~CPuzzleGameView();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
    //{{AFX_MSG(CPuzzleGameView)
    afx_msg void OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags);
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    afx_msg void OnTimer(UINT nIDEvent);
    afx_msg void OnNewGame();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG  // debug version in PuzzleGameView.cpp
inline CPuzzleGameDoc* CPuzzleGameView::GetDocument()
   { return (CPuzzleGameDoc*)m_pDocument; }
```

```
#endif

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif
// !defined(AFX_PUZZLEGAMEVIEW_H__24E70ECD_F618_11D8_8351_B19D0
C604C2D__INCLUDED_)


//coded by achal
// resource.h
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by PuzzleGame.rc
//
#define IDD_ABOUTBOX                    100
#define IDR_MAINFRAME                   128
#define IDR_PUZZLETYPE                  129
#define IDI_ICON1                       131
```

```
#define IDC_BUTTON1                1001
#define ID_NEW_GAME                32771

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_3D_CONTROLS           1
#define _APS_NEXT_RESOURCE_VALUE   134
#define _APS_NEXT_COMMAND_VALUE    32772
#define _APS_NEXT_CONTROL_VALUE    1003
#define _APS_NEXT_SYMED_VALUE      101
#endif
#endif


//coded by achal
// stdafx.cpp : source file that includes just the standard includes
//    PuzzleGame.pch will be the pre-compiled header
//    stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"
```

```
//coded by achal
// stdafx.h : include file for standard system include files,
//  or project specific include files that are used frequently, but
//      are changed infrequently
//

#if !defined(AFX_STDAFX_H__24E70EC7_F618_11D8_8351_B19D0C604C2D__
INCLUDED_)
#define
AFX_STDAFX_H__24E70EC7_F618_11D8_8351_B19D0C604C2D__INCLUDED
_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define VC_EXTRALEAN            // Exclude rarely-used stuff from Windows
headers

#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions
#include <afxdisp.h>        // MFC Automation classes
```

```
#include <afxdtctl.h>                // MFC support for Internet Explorer 4 Common
Controls
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>                        // MFC support for Windows Common
Controls
#endif // _AFX_NO_AFXCMN_SUPPORT



//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif
// !defined(AFX_STDAFX_H__24E70EC7_F618_11D8_8351_B19D0C604C2D__I
NCLUDED_)

//coded by achal
// main parts of the MFC application
    MICROSOFT FOUNDATION CLASS LIBRARY : PuzzleGame
================================================================
==========
```

AppWizard has created this PuzzleGame application for you. This application not only demonstrates the basics of using the Microsoft Foundation classes but is also a starting point for writing your application.

This file contains a summary of what you will find in each of the files that make up your PuzzleGame application.

PuzzleGame.dsp

This file (the project file) contains information at the project level and is used to build a single project or subproject. Other users can share the project (.dsp) file, but they should export the makefiles locally.

PuzzleGame.h

This is the main header file for the application. It includes other project specific headers (including Resource.h) and declares the CPuzzleGameApp application class.

PuzzleGame.cpp

This is the main application source file that contains the application class CPuzzleGameApp.

PuzzleGame.rc

This is a listing of all of the Microsoft Windows resources that the

program uses. It includes the icons, bitmaps, and cursors that are stored in the RES subdirectory. This file can be directly edited in Microsoft Visual C++.

PuzzleGame.clw

This file contains information used by ClassWizard to edit existing classes or add new classes. ClassWizard also uses this file to store information needed to create and edit message maps and dialog data maps and to create prototype member functions.

res\PuzzleGame.ico

This is an icon file, which is used as the application's icon. This icon is included by the main resource file PuzzleGame.rc.

res\PuzzleGame.rc2

This file contains resources that are not edited by Microsoft Visual C++. You should place all resources not editable by the resource editor in this file.

//

For the main frame window:

MainFrm.h, MainFrm.cpp

These files contain the frame class CMainFrame, which is derived from CFrameWnd and controls all SDI frame features.

///

AppWizard creates one document type and one view:

PuzzleGameDoc.h, PuzzleGameDoc.cpp - the document

These files contain your CPuzzleGameDoc class. Edit these files to add your special document data and to implement file saving and loading (via CPuzzleGameDoc::Serialize).

PuzzleGameView.h, PuzzleGameView.cpp - the view of the document

These files contain your CPuzzleGameView class.
CPuzzleGameView objects are used to view CPuzzleGameDoc objects.

///

Other standard files:

StdAfx.h, StdAfx.cpp

These files are used to build a precompiled header (PCH) file

named PuzzleGame.pch and a precompiled types file named StdAfx.obj.

Resource.h

This is the standard header file, which defines new resource IDs.

Microsoft Visual C++ reads and updates this file.

//

Other notes:

AppWizard uses "TODO:" to indicate parts of the source code you

should add to or customize.

If your application uses MFC in a shared DLL, and your application is

in a language other than the operating system's current language, you

will need to copy the corresponding localized resources MFC42XXX.DLL

from the Microsoft Visual C++ CD-ROM onto the system or system32 directory,

and rename it to be MFCLOC.DLL. ("XXX" stands for the language abbreviation.

For example, MFC42DEU.DLL contains resources translated to German.) If you

don't do this, some of the UI elements of your application will remain in the

language of the operating system.

//