Foreword by Steve Russel 9
Chapter 0: Introduction and a Little History about Game Development 11

PART I: THE HYDRA HARDWARE 25

Chapter 1: HYDRA System Overview and Quick Start 27
Chapter 2: 5V & 3.3V Power Supplies 77
Chapter 3: Reset Circuit 81
Chapter 4: USB-Serial Programming Port 83
Chapter 5: Debug Indicator Hardware 91
Chapter 6: Game Controller Hardware 95
Chapter 7: Composite NTSC / PAL Video Hardware 103
Chapter 8: VGA Hardware 115
Chapter 9: Audio Hardware 125
Chapter 10: Keyboard & Mouse Hardware 141
Chapter 11: Game Cartridge, EEPROM & Expansion Port Hardware 159
Chapter 12: HYDRA-NET Network Interface Port 167

PART II: PROPELLER CHIP ARCHITECTURE AND PROGRAMMING ... 175

Chapter 13: Propeller Chip Architecture and Programming 177
Chapter 14: Cog Video Hardware 233
Chapter 15: The Spin Language 245
Chapter 16: Programming Examples on the Propeller Chip / HYDRA 319

PART III: GAME PROGRAMMING ON THE HYDRA 425

Chapter 17: Introduction to Game Development 427
Chapter 18: Basic Graphics and 2D Animation 461
Chapter 19: Tile Engines and Sprites 509
Chapter 20: Getting Input from the “User” World 559
Chapter 21: Sound Design for Games 593
Chapter 22: Advanced Graphics and Animation 619
Chapter 23: AI, Physics Modeling, and Collision Detection - A Crash Course! 673
Chapter 24: Graphics Engine Development on the HYDRA 759
Chapter 25: HYDRA Demo Showcase 779

BACK MATTER 799

Epilog 800
Index 801

Foreword by Steve Russel

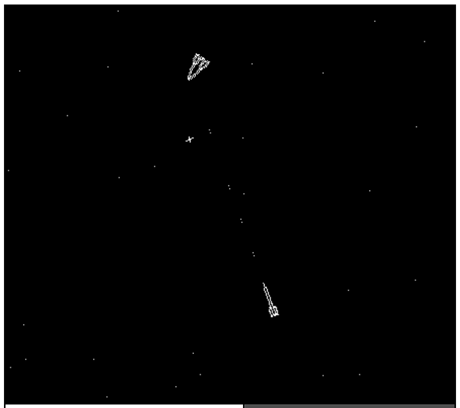
Over 45 years ago, a new PDP-1 computer arrived near my office with a display system that could do more than anything I had seen before. There was just one demonstration program, but it didn't use all the power of the display.

▲**The PDP Computer and Display** Figure F:1

I thought that I could make a better demonstration program, and after discussion with my friends, I started writing code. That program developed into "Spacewar!" – one of the first computer games to use a display and the distant ancestor of Atari Asteroids. I learned that playing computer games was fun, but writing them was even more fun!

One of the great things about writing a game is finding solutions to the puzzle of trying to get the most fun into the game while getting it to work well. Unlike most computer programming assignments, with a game you can adjust the problem to fit the available solution.

The development of SpaceWar! was a collaboration, for example Dan Edwards looked at my version of Spacewar! and decided it needed a sun and gravity to better show the problems of getting a spaceship into orbit. Even after he developed a run-time code generator that wrote a custom program to drive the display at its maximum speed, there still was only time to compute the gravity effect on 2 spaceships!


▲**Screen Shot of SpaceWar!** Figure F:2

We left the torpedoes untouched by gravity, and decided that they were "photon torpedoes" that were unaffected by gravity since they are pure energy (a game developer's perogative). The game still played fast enough, and the spaceship orbits added a great deal to the fun.

At one point, I decided that the torpedoes would be more "realistic" if they had a little random error, just like real torpedoes. I added this but everyone else complained loudly, so I took it out in the next version.

A few years later I had a different, much better display system arrive. I was able to write a very primitive flight simulator for it, but the pace was so slow that it was no fun. I learned that just having 3 dimensions doesn't necessarily make a game better.

You have much better hardware, software and examples to start with, so I hope you will learn how much fun game programming is with less pain and more fun than I did nearly half a century ago!

It turned out that I never got a new version of Spacewar! working well until some time between midnight and 6 AM.

Steve Russell

Co-creator of SpaceWar!
San Jose, California
July 2006

Chapter 0: Introduction and a Little History about Game Development

Welcome to ***Game Programming for the Propeller Powered HYDRA***. This is a no-holds-barred development manual about creating basic games and graphics applications on the Propeller powered HYDRA game console. As you might know, game development is the most complex field of computer science in the world and takes years to master. A video game is unlike any other program you can write for a computer; games must be fast, fun, graphically intensive, real-time, support multiple players, run on minimal hardware, and perform complex and/or seemingly impossible mathematical calculations at a rate fast enough to update the screen at 30-60 frames per second or more!

Additionally, games pull from advanced research in artificial intelligence, optimization theory, multiprocessing, compiler design, memory management, data structures, physics modeling, networking, compression, search algorithms, and much more. And if that wasn't enough, there are all the graphic, audio, and artistic media assets needed for a game. Some games literally are built upon tens to hundreds of terabytes of data and take hundreds of man-years to develop! Thus, video games are the ultimate fusion of science and art, together creating a real-time experience that billions of people have enjoyed since the late 50's.

▲**Halo II Running on the XBOX** Figure 0:1

Today games such as Halo II shown in Figure 0:1 amaze and delight millions. With the new next-generation systems available such as the XBOX 360 and the Playstation III (shown in Figure 0:2) the future is almost frightening to think of what will come next. The sheer computational power of these systems is staggering – each system is capable of an excess of 1.5 trillion floating-point operations per second! Both with multiple computational elements, especially the PS3 which contains the most advanced processor in the world – the ***"Cell"*** processor, a multibillion-dollar joint venture among Sony, IBM, and Toshiba.

▲**The XBOX 360 (left) and Playstation III (right)** Figure 0:2

Everyone knows that game development is serious business. With a gross revenue in excess of $30B, the game industry is larger that the movie industry, so getting into game development is one of the most desired job positions now and in the future for many engineers, programmers, and artists. Never has there been more freedom technically and artistically than there is today for game development. For fun, let's take a stroll down memory lane of some of the highlights in the game development and computer industry. This list is by no means complete. In fact, I highly recommend that you read some good books on the history of the video game industry and the computer industry, it's fascinating stuff.

✓ TIP

I highly recommend the following texts if you're interested in learning more about the foundations of the video game and computer industries and the amazing personality and technical challenges therein:

- ***Hackers: Heroes of the Computer Revolution*** **by Steven Levy**
- ***The Ultimate History of Video Games*** **by Steven Kent**
- ***Supercade: A Visual History of the Video Game Age*** **by Van Burnham**
- ***Masters of DOOM: How Two Guys Created an Empire and Transformed Pop Culture*** **by David Kushner**
- ***Opening the XBOX: Inside Microsoft's Plan to Unleash an Entertainment Revolution***

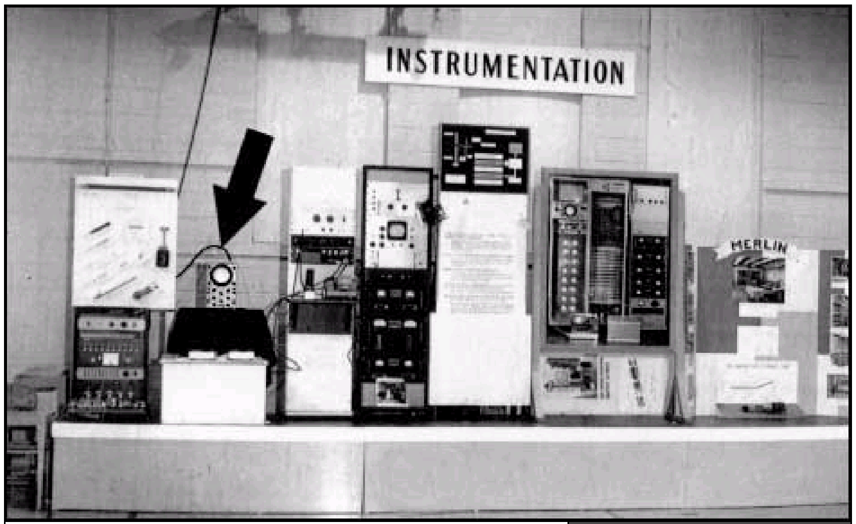
Finally, watch the DVDs *Pirates of Silicon Valley* and *Nerds 1.0,* both fascinating introspections into the genius and innovation the early years of computing generated.

0.1 A Brief History of Games 1958 - 1993

Ironically, game development is a very complex field. Most would think games are toys and simple, but many game developers have been programming 10-25+ years and are experts in numerous fields of computer science; moreover, the field is extremely competitive and changes on a day-to-day basis. Nonetheless, developing games and graphics applications are some of the most rewarding things to do with a computer; there is nothing like playing your own games, or watching others have fun with what you have made! As an artist it's the ultimate form of what I call "liquid art." Additionally, learning to develop games makes you a much better programmer; you will no longer be limited by memory, processor speeds, or the need for high-level languages; a game developer can literally make impossible things happen with a computer.

0.1.1 Table Tennis for Two (1958)

History is replete with examples that literally changed the world. With that in mind let's take a look at few key events in the development of the video game industry.



▲Table Tennis for Two Hardware Figure 0:3

Let's begin by setting the record straight. Many people think that ***Nolan Bushnell*** created the first video game with ***"PONG,"*** then others think that technically it was ***Ralph Baer*** with the ***"Brown Box"*** and the Magnavox ***Odyssey*** game console, still others think it's ***"Space War!"*** developed by ***Stephen "Slug" Russell***, but they are all wrong – in fact, it was a physicist – ***William Higginbotham*** in 1958 at the Brookhaven National Laboratory developed a game called ***"Table Tennis for Two"*** for an open house to show their new analog computer. Figure 0:3 shows a picture of the hardware that "Table Tennis for Two" ran on with an arrow pointing to the output device. Also, check out the link below to see the game in action (Real Player format):

http://real.bnl.gov/ramgen/bnl/PONG.rm

The game was developed completely in hardware by means of an analog computer. The lab wanted to show off something interesting other than weapons design research, so Willy took the manual that came with the analog computer and read about examples of drawing trajectories and curves on the oscilloscope. He took this information, and with the addition of some hardware he and a colleague cobbled together the VERY first video game in history.

0.1.2 Space War! (1962)

Next up was the creation of ***"Space War!"*** by Stephen "Slug" Russell at MIT. Other major contributors include Peter Samson, Martin Graetz, Wayne Witanen, Alan Kotok and Dan Edwards. The game was written on a DEC PDP-1 in pure assembly language in 1962. Steve "Slug" was nick-named "Slug" since like all software engineers, he took forever to finish anything! Figure 0.4 shows a screen shot of the original Space War! hardware, quite a difference from your laptop. You can actually play a remake of Space War! by following this URL:

http://lcs.www.media.mit.edu/groups/el/projects/spacewar/

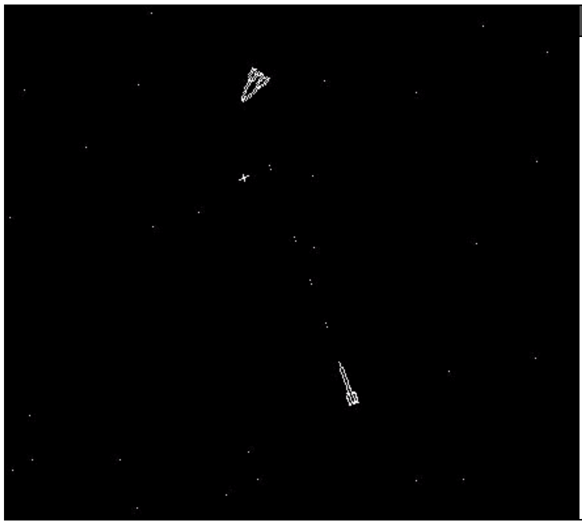


Figure 0:4

◂Space War! Running on a DEC PDP-1

0.1.3 Ralph Baer, the Brown Box, and the Maganvox Odyessy (1966)

Figure 0:5

◀**Ralph Baer and his Magnavox Odyssey**

Ralph H. Baer was a TV engineer who had an interest in interactive TV. He was the first person to ever have the notion of moving objects around on a TV screen, and quite frankly his associates and boss at Sander and Associates told him to forget about it and focus on making better TV sets. Nonetheless, Ralph kept working away on his ***"Brown Box"*** and in 1968 had a working prototype of a hard-wired game system capable of moving simple dots around on the screen.

Figure 0:5 shows Ralph and the Magnavox ***Odyssey*** system. The rendering ability (if you can call it that) of the Odyssey was non-existent, so in a brilliant stroke of "engineering ingenuity" Ralph thought "Why not add transparent backgrounds as overlays on the TV set itself?" So, that's what they did; the games that ran on the Odyssey all were nothing more than dots moving around, but when you put a nice background on the TV set screen itself of a tennis court, baseball diamond, or haunted house, it was like nothing anyone had seen. The Odyssey sold about 100-150,000 units depending on where you get your information. Interestingly though, it came out in 1972 officially, which was the same time that Atari PONG came out.

0.1.4 Atari PONG (1972)

Next, the most important commercial game was ***"PONG"*** developed by ***Nolan Bushnell*** and ***Al Alcorn*** of newly formed ***Atari*** in 1972. This game was responsible for putting games on the map and was the genesis of the entire video game industry as we know it. It all happened at Andy Capp's Tavern in Sunnyvale, CA. Nolan Bushnell, with his newly founded company Atari, decided to test a prototype of new game that his new engineer Al Alcorn developed called PONG in local neighborhood Andy Capp's Tavern as an experiment.

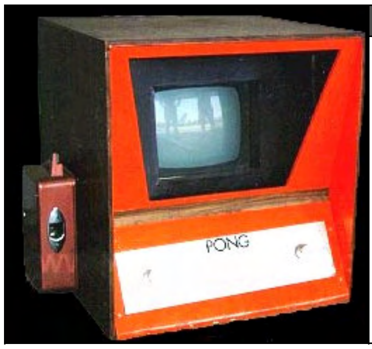


Figure 0:6

◂Original Atari PONG machine developed by Nolan Bushnell and Al Alcorn

Figure 0:6 shows one of the hand-made early prototypes. To their surprise, one week after the game was deployed there was a line around the corner to play, and the coin mech (a coffee can) was jammed since the machine was completely full of quarters! This moment in time launched the $30B video game industry, and Atari, one of the icons of American business and innovation, was created.

Atari was the fastest growing company in history at the time! And Bushnell, when he sold the company for $24M+ and change, was the "rock star" of Silicon Valley. Atari PONG more or less put the Odyssey out of business when Atari came out with a home version of PONG – remember it? The Atari version of PONG and the system it ran on (PONG on a chip) was light-years ahead of the Odyssey. The reason why is that the Odyssey technology was really early 60's technology and it took Ralph Baer such a long time to get the suits to listen, by the time they did, Nolan Bushnell was able to create the 2nd generation of games with PONG and capture the consumer market. But, we should acknowledge that technically the first game console was the brain-child of Ralph Baer, thus the designation of ***"Father of the Video Games"*** goes to Nolan Bushnell, while the ***"Grandfather of Video Games"*** goes to Ralph Baer! Interestingly, the first big patent infringement goes to Nolan Bushnell and PONG: Magnavox, on their knees, financially sued Atari as a last-ditch effort saying that PONG was a copy of games on the Odyssey. Atari did settle, but patenting dots running around on the screen – you've got to be kidding!

0.1.5 The Apple Computer (1977)

The personal computer industry was also a result of video games. ***Steve Jobs*** (co-founder of Apple) worked at Atari, and he and ***Steve "The WOZ" Wozniak*** were both interested in developing their own computer and game system to play games on and hack. Steve Jobs actually worked at Atari – Nolan Bushnell requested him to create a prototype of a new game called ***"Breakout"*** and Jobs accepted the challenge, enlisting electronics guru Steve Wozniak to do the design.

Together after a 4 day straight engineering/programming tribute to sleep deprivation, the result was a completed game in a ridiculously low number of chips with NO microprocessor! In fact, the design was so clever, so optimized, that Atari engineers couldn't understand it! However, the knowledge that Steve Wozniak learned and experimented with over those 4 days helped him develop both the Apple I and II computers, and the beginning of the personal computer era begun in 1977.

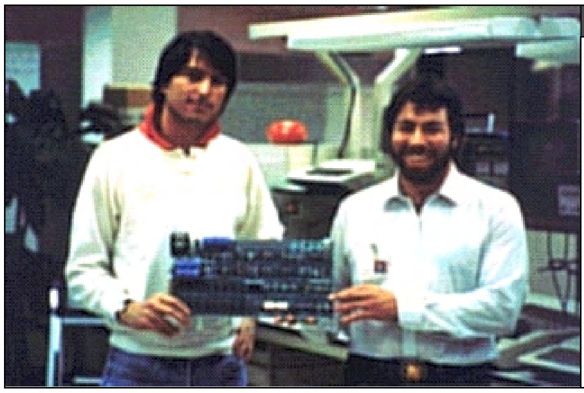


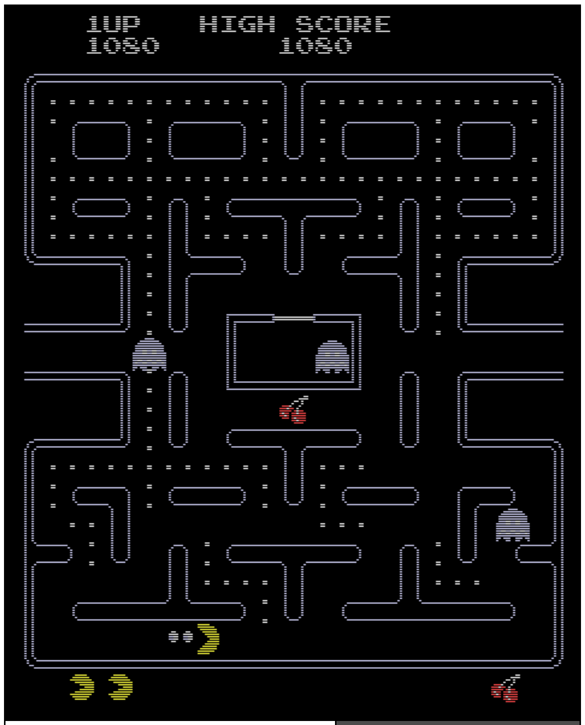
Figure 0:7

◀The two Steves (Jobs left, Woz right) holding their creation – the Apple II

Figure 0:7 shows the two Steves working on the original Apple I personal computer; this was of course followed by the Apple II which made Apple computer the fastest growing company in American history and the largest IPO (initial public offering) in history – I still am mad at my dad for not believing me in the late 70's when I told him to buy Apple stock!

0.1.6 Pac-Man (1980)

So, now we have the creation of the video game industry and the personal computer industry, the 80's are upon us, and things are getting serious and competitive. With the USA taking the lead position in the industry, the Japanese weren't far behind with their own blockbuster game and their contribution to changing the world of games. ***Toru Iwatani***, a 24 year old programmer, was in Tokyo and decided to sit down with some friends and have some pizza at the American franchise Shakey's Pizza. While ordering pizza, someone took a single piece of the cheese pizza and that image of a yellow circle with a piece removed was the inspiration for "Pac-Man," quite arguably one of the most successful games in history. Toru and his colleagues worked for 18 months on the game with a team of hardware and software engineers to develop Pac-Man. It was the largest game ever developed and the largest team ever to develop a game, but it paid off.

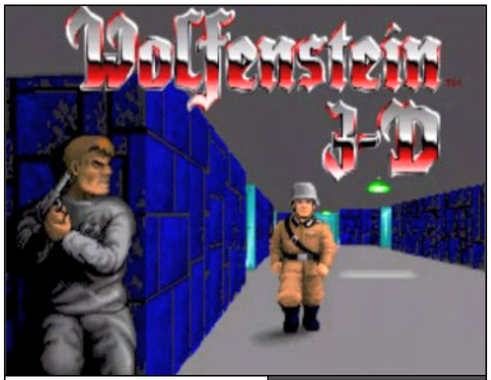


▲Pac-Man – the First System Engineered Game Figure 0:8

Pac-Man as shown in Figure 0:8 was an instant hit in America and all over the world where the machines were sent. The characters of Pac-Man also become overnight stars and everything from sequels to cartoons to breakfast cereals had a Pac-Man logo on it. The age of engineered games and product marketing was born. People realized this was serious business, and there were billions to be made…

INFO **Pac-Man was originally called "PUC-MAN", but when shipped to America, kids used to erase part of the "P" and the resulting name was less than desired by Namco. Thus, they change it to "Pac-Man" so at worst the game would read "Fac-Man"!**

0.1.7 Wolfenstein 3D and the Era of First Person Shooters (1992)



▲**Wolfenstein 3D by id Software** Figure 0:9

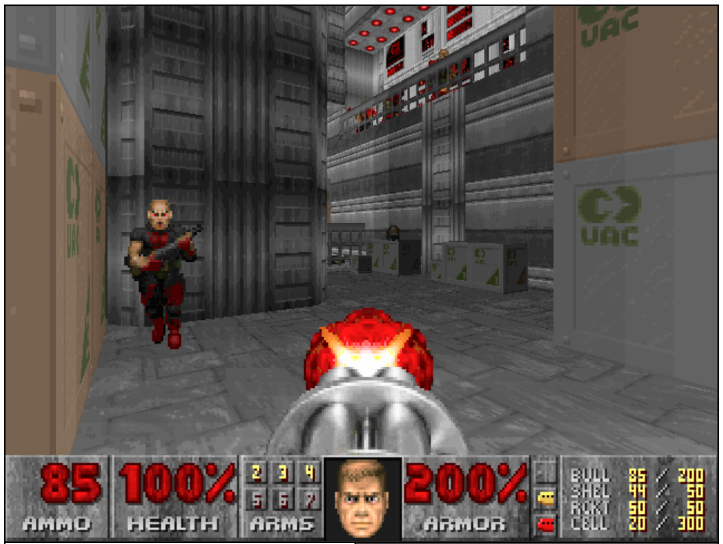
Certainly, there are dozens of games worth mentioning that were eventful in the industry, but we don't have time to really cover them in the depth that they deserve. Games like Space Invaders, Asteroids, Computer Space, and more all made a difference in the early 60's, 70's and 80's, but it wasn't until the 90's that games got scary – enter ***id Software*** the creators of ***Wolfenstein 3D*** as shown in Figure 0:9. Another Cinderella story, ***John Carmack*** and ***John Romero*** both were Apple II fanatics, both loners, and both interested in making games and world domination. John Romero, a little older than Carmack, had been bouncing around working at various places on game projects; at some point he met up with John Carmack and the results were similar to Bill Gates and Paul Allen getting together. John and John literally changed the world with their games. Soon after their initial meeting they were working at a company called ***Softdisk Publishing***, and to make a long and interesting story short, they were making a game a month for Softdisk to place on a floppy with a magazine! This is a feat to say the least, but during this time they got really good at making games, and did what most game programmers take years to do in months. Thus they honed their skills to a white-hot blaze ready to cut the fabric of space-time.

Ready to take on the world and report to no one, they started id Software. Their first game of note was ***"Commander Keen"*** (1990), a side scrolling tour de force thought to be impossible to achieve on the IBM PC, but they were just warming up. Carmack, turning into the technical guru of the group, had been experimenting with "ray casting" technology, a simplified version of "ray tracing" used to create photo real imagery in CG movies. However, ray casting allows 3D rendering to be achieved at blazing speeds due to simplified geometrical assumptions and a lot of tricks. The results of this ray casting technology was "Wolfenstein 3D" released in 1992, a 3D remake of the popular Apple II game "Castle Wolfenstein", but Wolfenstein 3D was 3D, and immersed the users in a fluid world running at blistering speeds. Figure 0:9 shows a screen shot.

Wolfenstein 3D was not only a technical marvel and for the billionth time made all the doubters realize that game developers are sorcerers and capable of magic, but Wolfenstein was highly controversial – its depiction of Nazis', blood and gore got the whole world up in a roar, but it was the first real-time cinematic experience on a personal computer. And like it or not, the world wanted more...and more they got...

0.1.8 DOOM (late 1993)

DOOM shown in Figure 0:10 speaks for itself; there are few people that do not know what DOOM is or who haven't played it.



▲Enter DOOM Figure 0:10

DOOM by far was the most impressive technical achievement on a PC the world had ever seen. Released in 1993, DOOM was based on a technology called *"Binary Space Partition"* or BSP trees, a technique discovered in the 60's to bisect space into half spaces for easier computation in a recursive algorithm.

Technical details aside, the results of the algorithm coupled with a game developer's clever programming was the most incredible experience ever on a PC: DOOM. Millions of people were stunned by the technology, and numerous industries including military, medical, and architectural, were affected. Not to mention the game spawned (no pun intended) the entire 3D accelerator market.

TIP

If you are interested in DOOM technology and how BSP trees work and how to implement them, you will be pleased to know that *The Black Art of 3D Game Programming* by yours truly is in electronic form included with the CD of this book. It came out in 1994/1995, and within it I showed the world how DOOM worked among other things.

0.2 Origins of the HYDRA

A few other hits have come out since including Quake, Half Life and of course Halo, but none with the impact of awe of these early games. The technology of game development is now being disseminated at an exponential rate; books, courses, and entire degrees in game development technology are now available. Alas, we won't be changing the world here, but I can't think of a more engaging way to have fun with the new Propeller chip than to make games on it! The Propeller chip has something near and dear to my heart and that's ***multiprocessing***. I simply love multiprocessing; if I could I would multiprocess in my sleep – I would! Game developers for years, including myself, have had to fake multiprocessing and/or use pseudo-multiprocessing with Pentium or PowerPC chips via ***"multiple execution units"*** which isn't the same. The Propeller chip is a true multiprocessing processor and definitely a very interesting chip to develop games on. Therefore, I thought "What better application than a game console around it, and to make some games on to get people interested in the processor and of course interested in games!"

When I developed the HYDRA, I wanted to keep the system open, simple, and more or less just a Propeller chip without adding a lot of ancillary hardware, thus the HYDRA has no extra computing augmentation and is more or less completely powered for the most part by the Propeller chip itself. The HYDRA is a good example of what you can do with just a Propeller chip; if you were to add extra SRAM or other hardware then the Propeller can be used to create all kinds of embedded applications. Additionally, the HYDRA was developed to simply experiment with the Propeller chip; the HYDRA has an expansion port, mouse and keyboard ports, game ports, dual 3.3 V / 5.0 V supplies, VGA and NTSC/PAL out, networking (RJ-11 based) and much more – I had a lot of fun designing it, and hopefully you have a lot of fun learning the Propeller chip and game development with it!

0.3 What to Expect

There is so much to cover in game development, a complete treatise on the subject usually takes about 1000-2000 pages to even scratch the surface. Alas rather than go nuts like I usually do, I decided to take a more beginners' approach with this book since unlike my other game development books where I assume we are all programming on a PC with DirectX, this is not the case. In this case, we have new hardware, a new chip, a new language, and you might be learning game development for the first time, not to mention being only a beginning programmer as well. Thus, I decided rather than engaging the transwarp drive like I usually do, let's keep this at impulse speed for most of the time with a romp here and there to warp speed!

With that in mind, I assume that you are a programmer; this book will not teach you programming. However, I don't assume you have done any game or graphics programming, so that part we will explore together, but you should be familiar with one or more of the following languages: BASIC, C/C++, JAVA, ASM, PASCAL, DELPHI, etc. I will discuss the language constructs of the Propeller chip's native language "Spin", but I will not teach programming concepts. Additionally, there is a large part of the book on the Propeller chip itself and a lot of Assembly language material; if you are new to Assembly language, I suggest you read a good book on 6502, or ARM, or even 8086 and write some programs to get the hang of the language. Specifics aside, I will always try my best to teach where possible, so those of you that get bored, simply skip past anything that is old news to you. Now, let's take a look at the three main sections that make up the book:

- **The HYDRA Hardware** - This is a fast and furious circuit description of the HYDRA Game Console's implementation around the Propeller chip. Not meant to be complete, it simply gives you a frame of reference as programmers, so you know what hardware does what along with some technical detail here and there. Each chapter tends to focus on a specific aspect of the HYDRA, thus some chapters are short; others are longer.
- **Propeller Chip Architecture and Programming** – This is the nitty-gritty of the Propeller chip and has examples of programming graphics, sound, joysticks, I/O, networking, and explains both the ASM and high-level language (Spin) supported by the Propeller chip as well as the technical description of the Propeller chip itself. This part of the book is hands-on and you will get to run a number of demos and see what they do. Also, we will focus on using Parallax general-purpose objects rather than high performance gaming code, so we can keep a black box approach.
- **Game Programming on the HYDRA** – This is the fun part. Once we have all the fundamentals down and you know what the HYDRA does and how the Propeller chip works and is programmed, then we can sit down and start learning about game development and graphics.

0.4 Target Audience

Typically game development is all about software; however, if you have purchased a HYDRA then you probably are interested in embedded systems, hardware, and may even be a full-fledged Electrical Engineer. On the other hand, you might be a programmer that is interested in getting into embedded systems, and what better way than with games? Trying to cater to everyone is nearly impossible, so this book is going to be more of a software guide rather than a hardware guide in as much as we are going to spend 90% of our time programming, rather than doing circuit analysis. That is, when I show a circuit to you, I am going to assume that you understand electronics, rather than explain the nitty-gritty. If you don't know anything about electronics, the explanation will be more than enough for programming purposes. So this book is about writing games, graphics, and media applications on the HYDRA and learning the Propeller chip, it's not about designing game consoles or the hardware therein. Considering that, we are still going to cover every single piece of hardware in the HYDRA in the first part of this book before getting into software. This way, even software guys will have some idea of what does what, and hardware guys will have a good reference for each sub-system to know what's doing what, or can make changes if they wish.

0.5 Conventions Used in this Book

The book's text is more or less straightforward: what you see is what you get. Typically, I will highlight important terms in the text the first time I introduce them, secondly, code listings will always be set off in a fixed point font and in a slightly smaller font pitch that the general text so more code can fit per page. Also, from time to time you will see special sidebars, like Notes, Warnings, etc. Lastly, when discussing key presses and menu item selection sequences I will always place angled brackets around the key or menu selection sequence, for example, if I wanted to tell you to press the control key and J at the same time, you will see "<CTRL + J>", similarly if I want you to go to the main menu, then select the sub-menu tools, then from there select configuration, I will write it something like this ***<Main Menu → Tools → Configuration>.*** And I may italicize the sequence and or highlight it to bring your attention to it and separate it from the text. Also, in the text, to set off code variables, I will simply italicize them. For example, if I wanted to talk about a "for loop", I would say something like, "referring to the *FOR* statement on line 10…", as you can see the "*FOR*" element is italicized.

0.6 Requirements

The main part of working with the HYDRA or the Propeller chip is using the Propeller Tool IDE. Currently, it only supports Windows XP, 2000, 2003. There are no Windows 95/98/ME tools or Linux. In the near future, I suspect there will be, so stayed tuned. But, most everyone with a PC has a copy of Windows XP/200X on it, so you should be fine. Other than that you should have at least one USB port free, and a standard multimedia type PC. Since the only thing you will use the PC for is compiling programs, you don't need a lot of horsepower, so a Pentium II or greater (or AMD equivalent) is more than enough.

Additionally, you will need a NTSC/PAL compatible TV to connect to the output of the HYDRA since it generates standard composite video. Also, if you want to experiment with the HYDRA's VGA output abilities you will need a VGA monitor or a simple KVM to switch your PC with the HYDRA. The HYDRA kit comes with everything else you need, so turn the page and let's start experimenting!

WARNING

Last, but not least, skim entire book *BEFORE* doing anything! There are a few items that are embedded in the middle or end that will help you understand things, so best to read the whole thing first *THEN* go ahead and start playing with the hardware and programming.