TECHNOLOGY, ENTREPRENEURSHIP AND THE REALITY OF BUILDING SYSTEMS

Five Decades of Innovation, Adaptation and Learning in Technology and Business

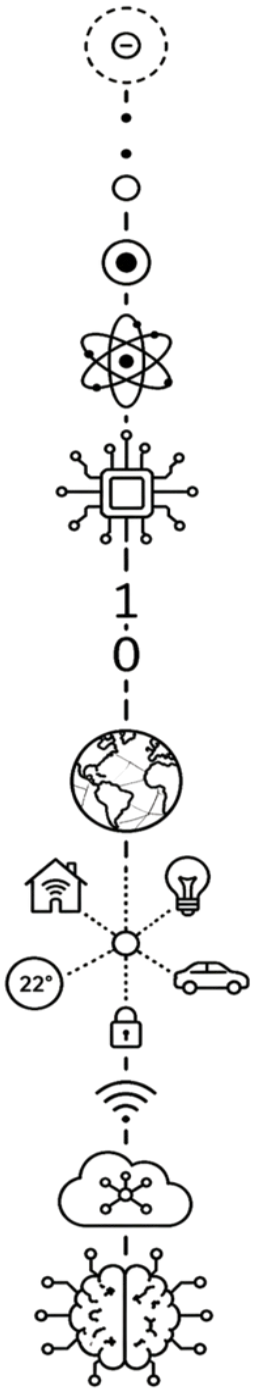
INNOVATION | ADAPTATION
RESILIENCE | IMPACT

PREVIEW
full edition available
for purchase at
www.ardiri.com

AARON ARDIRI

AN AUTOBIOGRAPHY

Technology, Entrepreneurship and the Reality of Building Systems



Aaron Ardiri

- an autobiography

© 2026 Aaron Ardiri
All rights reserved.

First Edition - 2026

ISBN 978-91-531-9116-2

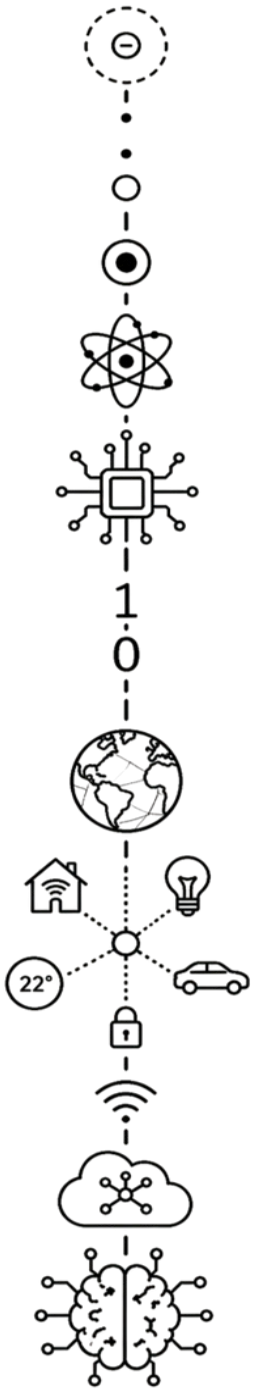
For Michael Leishman, whose early guidance sparked a curiosity that became a lifelong pursuit — without his mentorship, the path that followed would likely never have begun, and much of what this book represents would not exist.

And for Birgit, my wife, who has been a constant source of balance and calm — even when the work was difficult to explain, her support and presence brought clarity and stability, and connection to her daughter and grandchildren — reminding me of what matters in life beyond the systems I build.

Table of Contents

ACT 1:

Foundations: Curiosity, Chaos and Pattern Recognition



Dawn of the Digital Age

I was born on 7 June 1975, in Perth, Western Australia — at a time when something subtle but profound was already taking shape, something that would change the future of humanity.

There were no headlines announcing it, no sense that the world was about to change. In the background, quietly and steadily, technology was beginning its migration — from laboratories and institutions into homes, schools, and eventually into the hands of everyday people. I didn't know it then, but I was born at exactly the right time to grow up alongside it.

Aaron Ardiri — 5 years old

I was the third of five children, the first of what would become a new family unit. My father, Sicilian by birth, had migrated to Australia as a child and worked as a chef in various Italian restaurants. My mother, of Polish heritage, was born in Australia and focused on raising the household. My two older siblings — a brother and a sister — came from a different father, and later came my younger sisters.

We grew up in state housing. There was no excess, no safety net, and no room for waste. Meals happened at fixed times — breakfast, lunch, dinner — and if you missed it, you missed it. There was no open-ended schedule, no opening the fridge to see what was available whenever you were hungry. Other children seemed to have that freedom — we didn't. It wasn't discipline in the traditional sense, it was just the rules, purely about survival.

That structure shaped me in ways I didn't recognize at the time. It created an expectation that things happened when they were supposed to happen, leaving little room for delay, hesitation, or "I'll do it later." You learned quickly that timing mattered — and that missing your opportunity had consequences.

My parents eventually separated after the birth of my youngest sister, which added another layer of complexity with the introduction of a stepfather. I spent time not only within our immediate household, but also with my father's parents — Roman Catholic, traditional, and operating under a completely different set of values.

Moving between those worlds meant constantly adapting — different rules, different expectations, different ways of growing up. At the time, it felt normal. It taught me how to observe systems quickly, understand how they worked, and adjust my behavior accordingly. That ability — to step into an environment and figure it out — would become important later, though I didn't realize it then.

As a child, I was active — track and field, swimming, cricket, and Australian rules football — most of my time was spent outdoors, running around, competing, and burning energy. Alongside that, there was something else pulling at my attention. I was drawn to electronics, not in a structured way and certainly not in a way anyone encouraged. It was instinctive — if something had screws, I wanted to undo them; if something stopped working, I wanted to know why. I dismantled things I probably shouldn't have, driven by a need to see what was inside.

One moment in particular stands out — my father had bought me a watch. It wasn't anything particularly special, but to me it was something that was mine. I remember turning it over in my hands, looking at it, wondering not just what time it showed, but how it kept time. So I took it apart — carefully at first, then less carefully as curiosity took over and I wanted to figure out how it

worked. Screws removed, casing opened, tiny components exposed — gears, springs, and pieces that clearly had a purpose, even if I didn't yet understand what that purpose was.

Putting it back together was less successful. By the time I was done, the watch no longer worked — it wasn't even close. My father, understandably, was not impressed. It was something else entirely for me. It was the beginning of a pattern: taking things apart to understand them, even if it meant breaking them in the process. That curiosity never went away.

At school, things were different again — I did well, very well. I was a straight-A student, the kind teachers relied on to contribute, the kind who followed the rules. Some might say I was the "teacher's pet." Within the family, I was seen as the one who would always get special treatment — the "chosen one." I never saw it that way. I wasn't chosen — I just took control of my own destiny.

Doing well at school wasn't about approval — it was about direction. It was something I could influence, something I could control in an environment where many other things were not. It came with a cost. Being good at school didn't always make you popular — I was ridiculed for it, singled out for being different. If anything, that had the opposite effect of what people may have intended. It didn't push me away from learning — it pushed me deeper into it.

At the age of nine, that effort translated into something unexpected. I was given the opportunity to skip a complete year of school and travel to Sicily with my grandparents to see their homeland. It wasn't a break from education — I was expected to continue my studies independently — but it opened a completely different world: a new culture, a new language, and a new environment. It was the first time I experienced something beyond the boundaries of what I knew. Once again, the same instinct applied — observe, adapt, and understand.

My grandfather played a significant role during those years. He was a practical man — a railway worker by trade, a handyman by nature. Weekends were spent working on small projects: gardening, building a go-kart, making chairs, and fixing things that needed fixing. The radio was on constantly, playing 1980s music in the background as we focused on whatever we were doing. There was no formal teaching. You learned by doing — you watched, you helped, you tried and got it wrong, then instinctively tried again.

The closest thing I had to technology in that environment was a rotary phone and an aging black-and-white television from the late 1960s. There were no computers in the house, no digital devices, and nothing that hinted at what was coming. The curiosity was already there — if anything, the absence of technology made it stronger.

Atari 2600 Console (Wood Finish)

In the late 1970s and early 1980s, computers were beginning to appear, but they weren't common or cheap — in fact, they were quite expensive. If you didn't own one, you appreciated any opportunity to even encounter one. A neighbor might have an Atari 2600 connected to a television. A friend might have access to a VIC-20 or a Commodore 64. We didn't have those systems, which made every moment with them more valuable.

There was something different about them. Unlike the physical world, where outcomes were often unpredictable, these machines operated differently and predictably — they just followed rules. You pressed a button and something happened. You repeated an action and the result was the same. There was structure, logic, consistency — it made sense.

I was fortunate enough to receive a few handheld electronic games — Nintendo Game & Watch devices — as birthday or Christmas presents when I was young; they were the things kids just had to have. They were simple, self-contained systems, but they introduced something important: interaction with a system that could be learned and mastered. You didn't just play the game — you learned its behavior.

Commodore 64 Console

On the rare occasions I had access to the Commodore 64 — a staple of gameplay at the time — there was another part of the experience that stood out just as much as the games themselves: the loading process. Games didn't load instantly — they came on cassette tapes. You would press play on the tape deck, type a command, and then wait. The screen would change — blocks of color flashing, patterns shifting, and sometimes accompanied by distorted sounds coming from the television. It didn't look like anything meaningful, just noise. However, you knew something was happening.

You simply waited. There was no progress bar, no indication of how long it would take — you just sat there, watching the screen, listening to the sounds, in suspense, waiting for something to happen. Sometimes nothing did. The tape might fail to load — you'd rewind, try again, and wait all over again.

When it did succeed, the program finally appeared, often with its own intro screen or music. It felt like something had been created just for you, rewarding you for waiting.

It wasn't instant, it wasn't expected — it was the result of patience. Nothing happened instantly — because of that, everything felt more deliberate. Later, the 1541 disk drive (5.25" floppy) would replace the cassette tape, providing faster, more reliable loading.

At some point, I came across computer magazines that contained program listings. There was no internet, no places to download things — if you wanted to run a program, you had to type it in yourself, line by line, character by character. Sometimes it was a program listing in BASIC, sometimes it was raw hexadecimal.

Either way, it required focus — hours of it. You would sit there, carefully entering each line, knowing that a single mistake could stop the entire program from working — it often did. You'd reach the end, type the final command, and nothing would happen. Or worse, it would fail immediately, leaving no clear indication of where the problem was — which meant starting again, not from the point of failure but from the beginning.

You checked everything, comparing each character against what was printed, slowly working through the listing until you found the mistake. It wasn't frustrating in the way people might expect — it was just part of the process. You learned to pay more attention, to slow down, to be precise. Each failure wasn't a setback — it was a lesson learned.

Eventually, it worked. When it did, the reward wasn't just that the program ran — it was that you understood why it ran, especially if it was written in the BASIC programming language, which was strange but familiar to English.

Speedy Mosquito

Speedy little mosquitoes are your prey in this fast-paced game for the C-64. Swat 'em quick to score big!

By Charles T. Kowal

RUN It Right

Commodore 64
Joystick

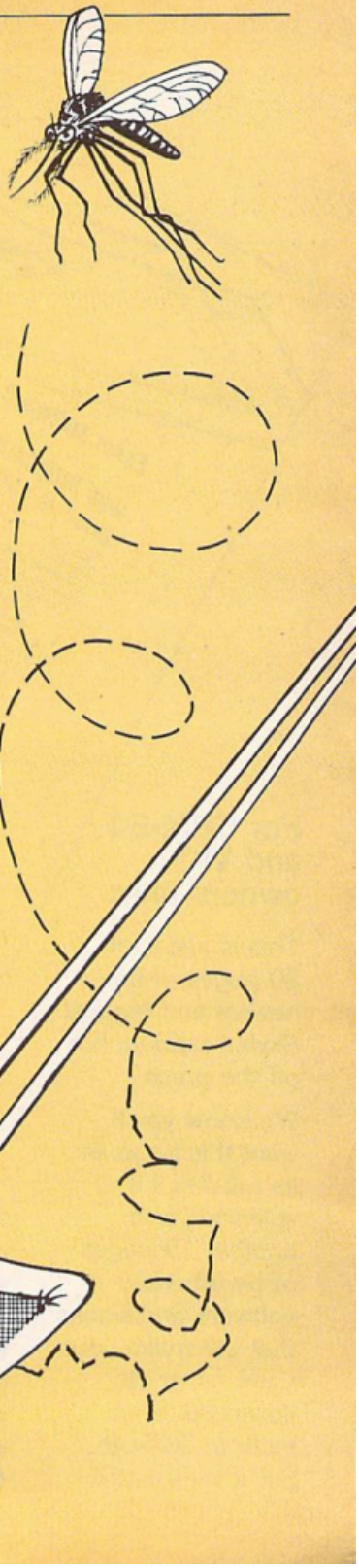
Address author correspondence to Charles T. Kowal, 3041 Alabama St., La Crescenta, CA 91214.

Mosquito is a simple but entertaining game, written completely in Basic for the C-64. The object is to swat the mosquito. You do this by moving your swatter with a joystick connected to Port 1. The mosquito moves around the screen at random (just like real life). When you swat at the mosquito, you have only one chance in nine of actually hitting it (just like real life)!

After you swat two mosquitoes, the other mosquitoes get angry, and big ones begin to appear in random places on the screen. If you or the little mosquito hit a big mosquito, you and your prey will get moved to different places on the screen. Sometimes you can use this to your advantage, if your swatter is far from the little mosquito.

But beware! One of the big mosquitoes is a killer. If you or the little mosquito touch this big one, the game is over. There is no way of knowing which mosquito is the killer until you hit it. Then it will turn red, and your score will be displayed.

After you swat ten little mosquitoes, the game ends, and your total time is displayed. Anything less than 100 seconds is a good score.



Commodore Magazine
May Not Reprint Without Permission

Run Magazine — Issue 2, 1984 (Feb) 'Speedy Mosquito' Game Listing (1)

Splats and Sprites

The program begins with a title screen, then a setting up message. During these preliminary displays, the computer is busily transferring the character set into RAM memory. This allows you to create your own characters.

Lines 130–160 create the little mosquito character and a "splat" character. Lines 1000–1060 create the sprites for the big mosquitoes. There are two sound effects in the program: the buzz of the mosquito and the sound of a swat. These sounds are created in lines 260–330 and 900–980, respectively. Lines 560–580 create the random mosquito movements.

After the computer finishes its initial setup, start the game by pressing the joystick fire button. You do *not* need to press this button to swat the mosquito. Just move the swatter to the little mosquito and keep it on top of him. Eventually the little insect will be swatted, but you must keep up with it and avoid the big mosquitoes.

Happy hunting! R

Listing 1. Mosquito program for the C-64.

```
10 PRINT"[SHFT CLR]":FORI=1TO5:PRINT:NEXT:PRINTTAB(16);
   "MOSQUITO"
15 PRINT:PRINT:PRINTTAB(19);"BY"
20 PRINT:PRINT:PRINTTAB(12);"CHARLES T. KOWAL":N=0:Q=16
   0:R=110:G=1
30 POKE53269,0:POKE53277,1:POKE53271,1:POKE53279,0:GOSU
   B1000
35 REM****   TRANSFER CHARACTER SET TO RAM   ****
40 POKE56334,PEEK(56334)AND254:POKE1,PEEK(1)AND251
50 FORI=0TO263:POKEI+12288,PEEK(I+53248):NEXT
60 PRINT"[SHFT CLR]SETTING UP CHARACTER TABLE"
70 FORI=384TO495:POKEI+12288,PEEK(I+53248):NEXT
80 FORI=816TO823:POKEI+12288,PEEK(I+53248):NEXT
90 FORI=1024TO1287:POKEI+12288,PEEK(I+53248):NEXT
100 FORI=1408TO1519:POKEI+12288,PEEK(I+53248):NEXT
110 POKE1,PEEK(1)OR4:POKE56334,PEEK(56334)OR1
120 POKE53272,(PEEK(53272)AND240)+12
125 REM****   MAKE MOSQUITO CHARACTER   ****
130 M=12288:POKEM,144:POKEM+1,80:POKEM+2,50:POKEM+3,252
    :POKEM+4,82:POKEM+5,73
140 POKEM+6,136:POKEM+7,132
145 REM****   MAKE 'SPLAT'   ****
150 M=M+1248:POKEM,137:POKEM+1,74:POKEM+2,36:POKEM+3,0:
    POKEM+4,0:POKEM+5,36
160 POKEM+6,74:POKEM+7,137
170 PRINT"[SHFT CLR]"
180 PRINT"PRESS FIRE BUTTON TO START"
190 IFPEEK(56321)<>239THEN190
200 POKE53269,0:POKE53277,PEEK(53277)AND254:POKE53271,P
    EEK(53271)AND254
210 POKE53279,0:YY=INT(8*RND(0)):ZZ=2[UP ARROW]YY
220 PRINT"[SHFT CLR]":FORN=56256TO56295:POKEN,14:NEXT
225 FORN=55296TO56255:POKEN,1:NEXT
230 FORN=1984TO2023:POKEN,160:NEXT:REM  CLEAR BOTTOM LI
    NE
240 SC=1524:DX=0:DY=0:SD=1024:CT=0:G=0:TM=TI/60
250 MD=MC:POKEMD,32:MC=1024+INT(960*RND(0)):POKEMC,0:PO
    KESC,102
255 REM****   MOSQUITO SOUND   ****
260 S=54272:FL=0
270 POKES+24,0
280 POKES+1,100
290 POKES+5,219
300 POKES+15,28
310 POKES+24,15
320 POKES+4,19
330 FORT=1TO200:NEXT:POKES+4,18
375 REM****   READ JOYSTICK   ****
380 JV=15-(PEEK(56321)AND15):DX=0:DY=0
400 IFJV=1THENDY=-40:POKESD,32:GOTO490
410 IFJV=2THENDY=40:POKESD,32:GOTO490
420 IFJV=4THENDX=-1:POKESD,32:GOTO490
430 IFJV=5THENDX=-1:DY=-40:POKESD,32:GOTO490
```

More

Circle 180 on Reader Service card.

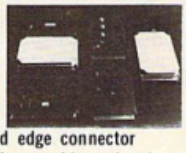
GLOUCESTER COMPUTER
Tools for learning and dedicated applications programming.

PROMQUEEN Write code for most common 8-bit microprocessors, test it in circuit, and burn it on EPROM with this all-in-one micro development system cartridge. Powerful machine code editor provides comprehensive ROMware development support. Ideal for robotics, process control, game development. Commodore VIC-20 host computer. Programs 2716, 2732, 2758 EPROMS and similar EPROMS. $199.00

PQ/64, all features of Promqueen less mimic mode. Software enhanced to include EPROM QC utilities, RS-232 communication, printouts. 28 pin ZIF socket. Reads, edits runs and programs all 5 volt 2500 and 2700 series EPROMS plus variety of EEPROMS all without personality modules. Commodore C-64 host computer. $299.00

PQ/64, RS pack performs RS-232 voltage conversions for PQ/64 system. $49.00

Plug your applications software into Commodore's computers on Gloucester Computer ROM Packs. Our top quality ROM cartridges include



- bypassing on all chips
- low noise layout with ground plane
- solder mask and gold plated edge connector
- wave soldered assembly and solvent cleaning
- DIP switch for address selection

VIC-20 versions include model V-8 for two 2732 EPROMS and model V-16 for 4 2732 EPROMS. C-64 versions include model C-16 for four 2732 EPROMS and model B-16 for two 2764 EPROMS.

All products shipped with comprehensive documentation. Call our user hotline 617-283-7719 or write for information: Gloucester Computer, 1 Blackburn Center, Gloucester, MA 01930.

www.Commodore.ca
May Not Reprint Without Permission

Run Magazine — Issue 2, 1984 (Feb) 'Speedy Mosquito' Game Listing (2)

Circle 66 on Reader Service card.

CBM 64 / VIC 20

Available for the 64 and VIC

PORTFOLIO MANAGER
(VIC version req. 8k)
Tape.....$44.95 Disc.....$49.95

SIGMA STAT
(VIC version req. 8K)
Tape.....$44.95 Disc.....$49.95

HOME ACCOUNTANT
Tape.....$24.95 Disc.....$29.95

INVESTOR
Tape.....$19.95 Disc.....$24.95

LOAN/MORT. ANALYZER
Tape.....$19.95 Disc.....24.95

Send check, cash or M.O. plus $1.50 p/h
NY Res. add 8.25% Sales Tax

EMBASSY COMPUTER PRODUCTS
P.O. Box 88, Little Neck, N.Y. 11363

DEALER INQUIRIES INVITED
PROGRAMMERS WANTED

VIC Trademark of Commodore

Circle 216 on Reader Service card.

NEW C-64

SAIL TO AMERICA

A totally new computer experience

- **Parents** Tell your kids Cadmean's *The Voyage of the Mayflower* has all the color, sound and excitement they love. Challenge the mighty Atlantic, defy its roaring storms and bring your passengers safely to the new world. There's never been an experience like it. Anywhere.
- **Kids** Tell your parents *The Voyage of the Mayflower* is a terrific learning adventure. Recreate the hazards and drama of the first Pilgrim voyage. Learn about sailing strategy, weather, navigation and history. The more you know the more fun it is. Every level is a unique experience whether you're 6 or 60. Unforgettable.
- **Families** Pit your imagination against the world as the Pilgrims knew it. Share the exciting journey to a new life in a new land. Risk the danger and feel the joy. Learn together how the Mayflower sailed into history on the courage of those few who dared.

All this and a **FREE** 11 x 16 Poster for only $29.

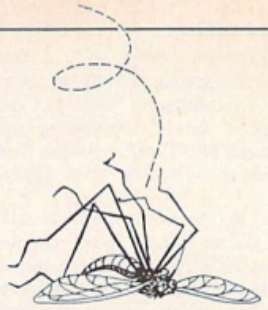
School and dealer inquiries welcomed

DISK ONLY

ORDER NOW. **FREE** shipping for MasterCard and Visa orders. Call (313) 994-0845 Day or Night. C.O.D., checks, money orders add $3.00 shipping.

CADMEAN CORP.,
309 Koch, Ann Arbor, MI 48103

Listing continued.



```
44Ø IFJV=6THENDX=-1:DY=4Ø:POKESD,32:GOTO49Ø
45Ø IFJV=8THENDX=1:POKESD,32:GOTO49Ø
46Ø IFJV=9THENDX=1:DY=-4Ø:POKESD,32:GOTO49Ø
47Ø IFJV=1ØTHENDX=1:DY=4Ø:POKESD,32
49Ø SC=SC+DX+DY:IFSC>1983THENSC=SC-96Ø
5ØØ IFSC<1Ø24THENSC=SC+96Ø
51Ø POKESC,1Ø2:SD=SC
52Ø XX=PEEK(53279):IFXX=ZZTHENPOKE53287+YY,2:XX=Ø:POKE5
     3269,ZZ:GOTO8ØØ
53Ø IFXX<>OTHENPOKESD,32:SC=SC+INT(1Ø*RND(Ø)):SD=SC:POK
     ESC,1Ø2:XX=Ø:GOTO25Ø
54Ø POKES+4,19:FORT=1TO1Ø:NEXT:POKES+4,18
55Ø DX=Ø:DY=Ø
56Ø MX=INT(RND(Ø)*3):MX=MX-1
57Ø MY=INT(RND(Ø)*3):MY=(MY-1)*40
58Ø MD=MC:MC=MC+MX+MY
59Ø IFMC>1983THENMC=MC-96Ø
6Ø¢ IFMC<1Ø24THENMC=MC+96Ø
61¢ IFMC=SCTHENTN=TI/6¢:GOTO9¢¢
62Ø POKES+4,19:FORT=1TO1Ø:NEXT:POKES+4,18
63Ø POKEMD,32:POKEMC,¢:GOTO38Ø
7ØØ CT=CT+1
71Ø IFCT>9THENPOKE53269,¢:GOTO8¢¢
72Ø IFCT<2THEN74Ø
73Ø N=CT-2:RESTORE:Q=155*RND(¢):R=175*RND(¢):G=2*G+1:GO
     SUB1¢4Ø
74Ø POKE1984,147:POKE1985,131:POKE1986,143:POKE1987,146
     :POKE1988,133
75Ø POKE1989,189:POKE199Ø,(CT+176)
76Ø POKESC+54272,1:GOTO25Ø
8ØØ POKEMD,32:POKEMC,32
81Ø PRINT"[SHFT CLR]","[CTRL 9]GAME OVER[CTRL Ø]":PRINT
     "TIME=";INT(TN-TM+.5);"SECONDS"
82Ø PRINT"SCORE=";CT:POKE53279,Ø:POKES+24,Ø
83Ø PRINT"[CRSR DN][CRSR DN][CRSR DN][CRSR DN][CRSR DN]
     PRESS FIRE BUTTON TO START AGAIN":FL=1:GOTO19Ø
895 REM****   SWATTING SOUND   ****
9ØØ FORL=ØTO24:POKES+L,Ø:NEXT:POKESC,156:POKEMD,32:POKE
     SC+54272,7
91Ø POKES,24Ø:POKES+1,33
92Ø POKES+5,8
93Ø POKES+22,1Ø4
94Ø POKES+23,1
95Ø POKES+24,79
96Ø POKES+4,129
97Ø FORT=1TO25Ø:NEXT:POKES+4,128
98Ø FORL=ØTO24:POKES+L,Ø:NEXT
99Ø GOTO7ØØ
1ØØØ DATAØ,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,97,Ø,Ø,99,Ø,Ø,54,Ø,Ø
     ,54,Ø,Ø,28,Ø,1,255,64
1Ø1Ø DATA3,255,224,Ø,218,32,1,85,16,2,84,136,2,84,128,4
     ,146,128,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
1Ø2Ø DATAØ,Ø,Ø,Ø,Ø,Ø,Ø
1Ø3Ø FORI=2Ø4ØTO2Ø47:POKEI,13:NEXT
1Ø4Ø FORI=ØTO62:READZ:POKE832+1,Z:NEXT
1Ø5Ø V=53248:POKEV+2*N,R+5Ø:POKEV+1+2*N,Q+4Ø
1Ø6Ø POKEV+16,Ø:POKEV+21,G:POKEV+39+N,1
1Ø7Ø IFPEEK(53279)<>OTHENMD=MC:POKEMD,32:MC=MC+5:SD=SC:
     POKESD,32:SC=SC+5
1Ø8Ø RETURN
```

www.Commodore.ca
May Not Reprint Without Permission

Run Magazine — Issue 2, 1984 (Feb) 'Speedy Mosquito' Game Listing (3)

You had read every part of it, entered every instruction, and corrected every mistake along the way. That process — slow, deliberate, and unforgiving — left a mark. In some cases, the listings were accompanied by explanations of what was going on, which helped deepen that understanding.

Beyond my own experiences, the broader world of computing was beginning to evolve. Systems like the BBC Micro in the UK and the Microbee in Australia were appearing in schools. These were Z80-based machines — 8-bit systems with extremely limited resources. Those limitations were part of what made them powerful — developers had to think carefully, every byte mattered, and every instruction had a cost.

At the same time, home consoles were gaining traction. The Atari 2600 had already established itself, and more systems were beginning to appear. Technology was no longer confined to governments and institutions — it was becoming something everyday people could interact with directly, even if you didn't own it.

I didn't know it then, but everything I was exposed to — the structure at home, the discipline of school, the hands-on work with my grandfather, the process of taking things apart, the careful repetition of typing code, and the waiting for machines to respond — was shaping the way I saw the world.

Not as a collection of isolated experiences, but as a series of patterns. Once you see patterns, you start understanding how things work — and that changes everything.

Coin-op: Pixels and Patterns

While computers slowly found their way into homes and classrooms, a different kind of machine appeared in public spaces — the Arcade machine. Arcades didn't introduce themselves quietly. You became aware of them before you ever saw them. The sound carried through the streets — layered noise from machines competing for attention. Electronic tones, explosions, fragments of music, overlapping in a way that felt chaotic but deliberate. It was designed to be noticed.

By the age of thirteen, my world had started to expand beyond the boundaries of home. Alongside a group of friends, I began working as a paperboy in the city. It was simple work, structured in the same way much of my early life had been. We were assigned a location, given a stack of newspapers and paid a commission for every copy we sold. The more we sold, the more we earned.

But the real value wasn't the money itself — it was what the money represented. For the first time, we had a degree of independence. We were catching buses into the city, navigating streets on our own, making decisions without immediate supervision. It was a shift — from being part of a structured household to operating, in small ways, on our own terms.

After the work was done, there was a routine. We would find something to eat, usually something cheap, and then decide what to do with whatever we had earned. That was when the sound would find us. It was not something that could be ignored.

The Arcade — Timezone circa 1992 (State Library of Western Australia)

You would hear it from a distance and, almost without discussion, begin moving toward it. Around a corner, down a side street, or inside a shopping complex, there it was — the Arcade.

The first impression was always the same. Rows of stand-up machines, each one alive with color and motion. Screens flickering, attract modes cycling through gameplay demonstrations, high-score tables displayed as a challenge. The machines were arranged to maximize exposure — each one competing for attention, each one inviting interaction.

Every machine had the same requirement — a 20-cent coin.

You would approach, coin in hand, already making a decision. Which game? How long would it last? Could you do better than last time? You inserted the coin, pressed start, and the machine took over your full attention.

At first, it felt simple. The early moments of most games were designed that way — they gave you just enough control, just enough success, to create a sense of understanding. It felt manageable, predictable even. But that changed quickly. As the game progressed, the difficulty increased.

Movement became faster, patterns became tighter, and mistakes became more costly. What initially felt like control shifted into something else — a test of how well you had actually understood what was happening.

Eventually, you would fail. Then came the decision — another coin, or stop? Most of the time, you chose to continue, because you were in the moment, you knew you could beat it next time and you still had coins in your pocket.

These machines were designed around that choice. They balanced progression with failure in a way that encouraged persistence. You were not just playing — you were investing: time, attention, and increasingly, money. The game itself did not end — you did, when you ran out of money.

On the surface, these were just games. For most people, that was enough. They were a way to pass time, to compete, to enjoy something immediate and engaging. For me, they were something else — they were structured systems, governed by rules that could be learned.

A few games are vivid in my memory from that time

- Frogger,
- Galaga,
- 1942, and
- Bubble Bobble.

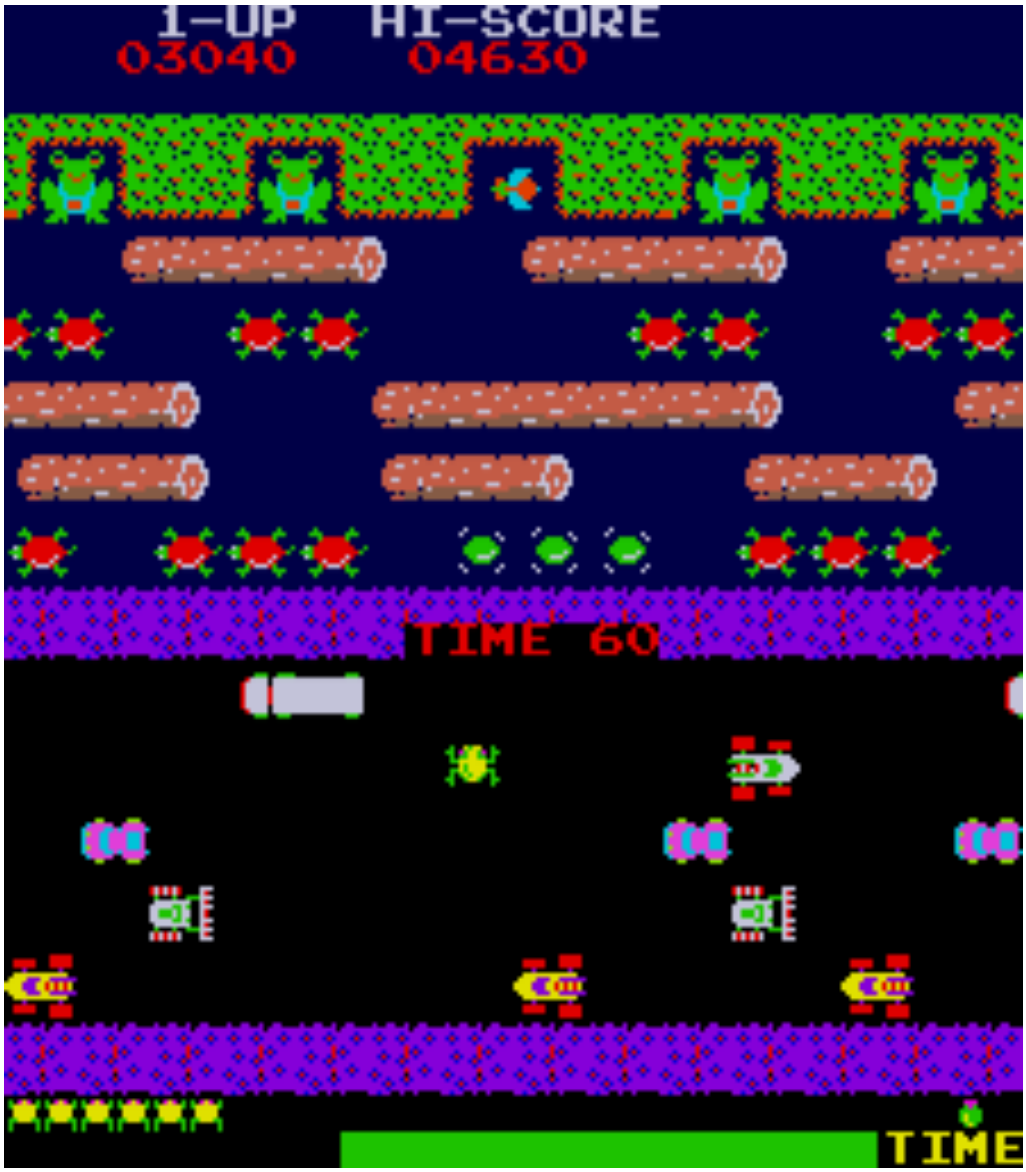
Frogger: Timing, Movement and the Illusion of Chaos

Frogger: Arcade Flyer

One of the earliest arcade games that left a lasting impression was Frogger.

On the surface, it appeared simple. The objective was clear: guide your character — a frog — from the bottom of the screen to designated positions at the top, crossing both a busy road and a river along the way, maybe picking up your girlfriend for a bonus.

The design was minimal. There were no complex controls — it was up, down, left, and right — no narrative context, and no explicit instruction beyond immediate interaction. Yet beneath that simplicity were structured rules that rewarded observation more than reaction.



Frogger: Game Screen

The first phase involved crossing a road filled with moving vehicles. At a glance, the traffic appeared unpredictable — cars and trucks moving at different speeds across multiple lanes.

However, extended play revealed that the movement was not random. Each lane operated on a fixed cycle. Vehicles followed consistent timing patterns, and once those patterns were recognized, movement across the road became a matter of alignment rather than avoidance.

You made it to the middle — a safe zone to recover before moving on.

The second phase introduced a different form of movement. Instead of avoiding obstacles, the player relied on them. Logs and turtles moved horizontally across the screen, providing temporary platforms to cross a river.

By that point, what had initially felt random no longer was. The movement hadn't changed — only the understanding of it had. What looked unpredictable at first revealed itself as consistent, once observed long enough.

The game required a shift in thinking — from evasion to utilization.

Movement was no longer about avoiding objects, but about synchronizing with them. Each element operated independently, yet within a predictable framework. Logs followed repeating paths, some moving faster than others. Turtles submerged and resurfaced in timed intervals. The space between platforms was consistent, even when it appeared otherwise.

The final objective — reaching one of several predefined slots at the top of the screen — introduced positional constraints. Not all targets were equal. The leftmost position, in particular, required precise timing due to the alignment of incoming platforms. While other positions allowed for patience and observation, this one demanded commitment at the correct moment.

Completion of a level did not conclude the game — it expanded it.

Subsequent stages introduced increased speed, reduced margins for error, and additional hazards. Elements that had previously represented safety were altered — a crocodile might appear within a target slot, transforming a known endpoint into a threat. Later, snakes would occupy areas that had once been considered safe resting points.

These changes were not arbitrary — they were incremental modifications to an existing behavior. The underlying patterns remained intact, but the assumptions built upon them were challenged.

This was the first clear instance where the game revealed its structure through repetition. Success was not determined by reflex alone, but by the ability to recognize patterns — movement became deliberate rather than reactive.

At the time, it was simply a game. In retrospect, it was an introduction to structured behavior — an early lesson in how complexity can emerge from simple, repeatable rules.

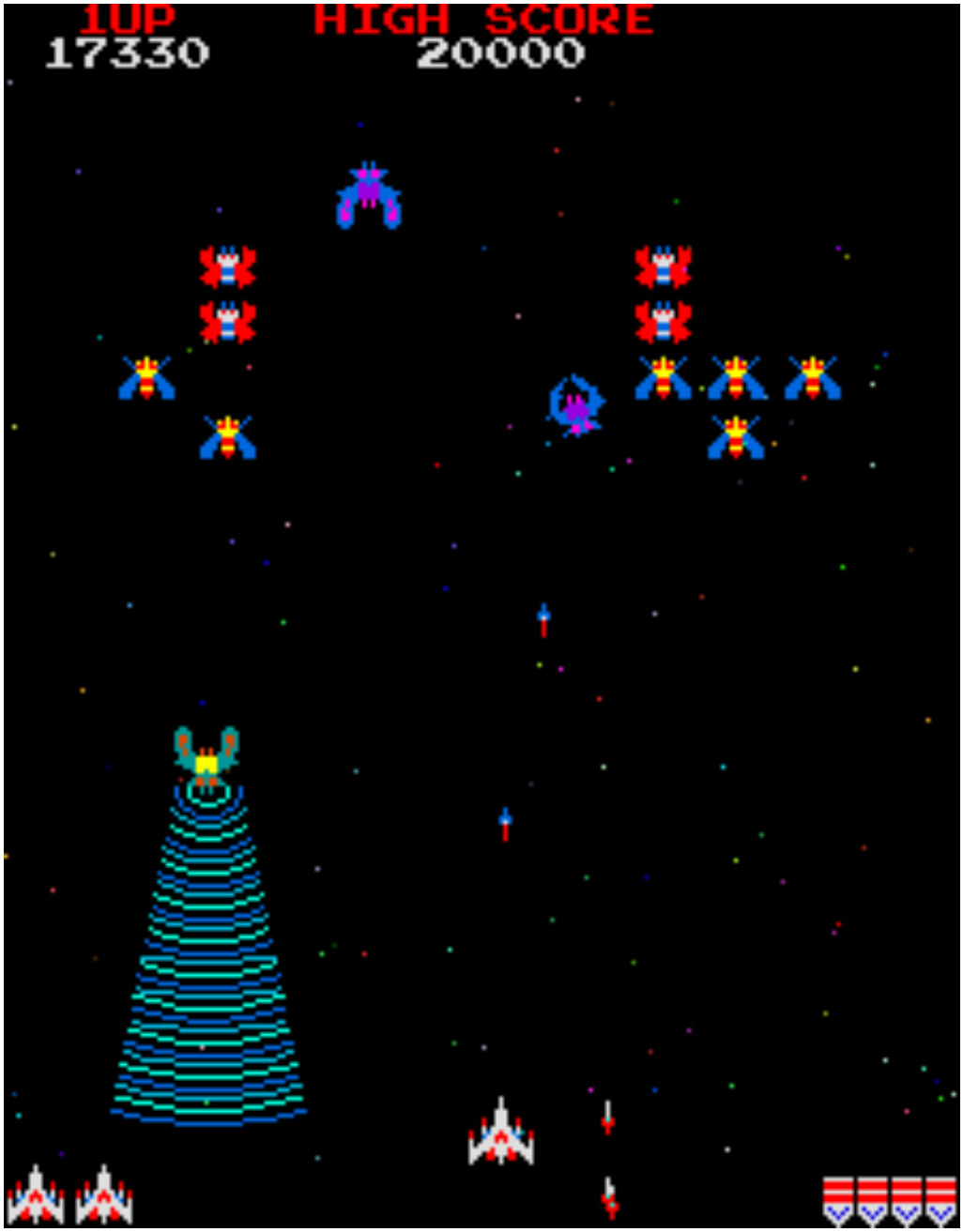
Galaga: Formation, State and Risk

Galaga: Arcade Flyer

Galaga introduced a different kind of game — a system based on formation, state, and controlled risk.

At its core, it was a fixed-screen shooter with a parallax background scrolling to simulate stars in the distance. The player controlled a single ship positioned at the bottom of the display, tasked with eliminating waves of

incoming enemies. The objective was straightforward: survive, score points, wipe out the enemies, and progress through increasingly difficult stages.



Galaga: Game Screen

However, the behavior was more complex than it initially appeared. Enemy ships did not move randomly — each wave followed a structured sequence. At the beginning of a stage, enemies would enter the screen in formation, tracing predefined paths before settling into a grid at the top.

These entry patterns were consistent. Over time, they became recognizable. Once the entry wave was over, the game shifted into a new phase. Individual

enemies would break formation and initiate attack patterns. The movements were not arbitrary — they followed repeatable arcs, loops, and dive sequences. What initially appeared chaotic was, in fact, predictable.

At first, the speed of the movement masked that structure. Multiple enemies diving, crossing paths, and firing created the impression of randomness. But the patterns didn't change — only the ability to recognize them did. What seemed reactive became something that could be anticipated.

The distinction between reacting and anticipating became critical.

An inexperienced player would respond to movement as it occurred. A more experienced player would position themselves based on where enemies were going to be. The game rewarded foresight rather than reflex.

Within this structure, enemy types introduced variation — they behaved differently, moved at different speeds, and carried different point values. Some were more valuable if destroyed during movement rather than while stationary, introducing a layer of decision-making beyond simple survival.

The most significant shift in the game, however, came through the capture mechanic. At certain points, larger enemies would descend, pause, and emit a beam toward the player's ship. The instinctive response was avoidance. The beam appeared dangerous — and it was. If caught, the player's ship was taken. The game labelled this state clearly: FIGHTER CAPTURED.

At first, this was interpreted as a penalty — a loss of control and a reduction in capability. However, continued observation revealed a second layer to the mechanic. The captured ship did not disappear. It remained attached to the enemy that had taken it.

This created a new objective. If the player survived long enough and destroyed the capturing enemy at the correct moment, the lost ship could be recovered.

Upon recovery, the game entered a new state — the player now had two ships, aligned side by side, firing simultaneously. This altered the entire dynamic of the game. Firepower was effectively doubled, allowing for more efficient clearing of enemies and higher scoring potential.

However, this increased capability introduced additional risk. The combined ship occupied a larger horizontal space, making it more vulnerable to incoming attacks.

The game presented a deliberate trade-off.

A player could choose to avoid capture entirely, maintaining a smaller, more agile presence. Alternatively, the player could intentionally allow capture, with the goal of recovering the ship and gaining increased offensive capability.

This was one of the first clear examples in the Arcade of rewarding controlled risk. The mechanic was not explained — it had to be discovered. Once understood, it became a strategy.

The game itself was structured in cycles. Standard attack waves were interspersed with challenge stages — phases where enemies followed fixed patterns without firing, allowing the player to maximize score through precision shooting. These stages reinforced the idea that the game was not purely reactive. It contained predictable, repeatable sequences that could be mastered.

Across repeated play, the structure became increasingly clear. Enemy entry patterns were consistent, attack sequences followed defined rules, and the capture mechanic introduced state changes that could be exploited. Scoring systems rewarded precision and timing.

What appeared at first to be a fast-paced, reflex-driven game revealed itself to be a layered system governed by rules. Progress was not determined by speed alone — it was determined by understanding, as new waves of enemies appeared and the patterns became familiar.

Over time, I found that I was no longer reacting to the game — I was anticipating it, positioning myself based on expected behavior rather than visible movement. The underlying rules had not changed. My perception of them had.

1942: Flow, Scaling and Controlled Complexity

1942: Arcade Flyer

1942 expanded the experience into continuous motion. The perspective shifted from defending a position to navigating through an environment. The screen scrolled vertically, creating the impression of forward progression — as though the player was moving through a space rather than reacting to it.

1942: Game Screen

This introduced a different type of game. Instead of discrete stages with clear boundaries, 1942 presented a continuous flow of encounters.

Enemy aircraft appeared in waves, each with distinct formations and behaviors. At first glance, these waves appeared unpredictable — planes entering from different angles, diverging, regrouping, and attacking in rapid succession.

However, as with previous games, the behavior was structured.

Each wave followed a defined pattern. Some formations approached in straight lines before breaking apart. Others curved across the screen in arcs,

looping before converging on the player's position. What initially appeared chaotic revealed itself, over time, to be repeatable.

Recognition of these patterns changed the way the game was played. Instead of reacting to individual enemies, the player began to anticipate entire sequences. Positioning became proactive rather than defensive. Movement was no longer about avoiding immediate threats, but about aligning with the expected flow of incoming waves.

The game introduced another layer through power-ups. Certain enemy aircraft — often distinguished by color — would release enhancements when destroyed under specific conditions. These power-ups varied in effect. Some increased firepower, extending the player's ability to engage multiple targets simultaneously. Others cleared the screen, removing immediate threats. Additional configurations introduced wingmen, expanding offensive coverage.

These enhancements altered the player's state within the game. More power provided greater control over the screen, allowing for more aggressive positioning and faster elimination of threats. However, this increased capability came at a cost. A larger visual presence meant reduced maneuverability and a greater likelihood of collision or incoming fire.

The game did not present power-ups as purely beneficial — they introduced trade-offs. A player could choose to maximize firepower, accepting increased risk, or maintain a smaller profile, prioritizing control and survivability. The optimal choice depended on the player's understanding of the game and their ability to manage its constraints.

Scoring introduced another structured element. When destroying larger enemy planes — that appeared twice in each level and sprayed bullets until taken out — point values increased in a doubling sequence: 500, 1000, 2000, 4000, 8000, and so on. This reflected an underlying pattern that rewarded sustained performance. Longer survival, combined with consistent elimination of these planes, led to exponential growth in score.

At the time, this appeared to be a simple reward mechanism. In retrospect, it was an early exposure to exponential scaling. The more enemies you prevent from escaping, the greater the bonus at the end of the level.

The longer you remain without failure, the greater the return. This principle extended beyond scoring. Enemy density increased as progression continued. Waves became more complex. Patterns overlapped, requiring simultaneous recognition of multiple behaviors. The game did not change its rules — it increased the complexity of their interaction.

At regular intervals, the flow was interrupted by a distinct phase — a boss encounter. Unlike the smaller aircraft, these entities occupied a significant portion of the screen. Their behavior was more concentrated, often involving repeated firing patterns that required precise navigation to avoid.

These encounters acted as checkpoints — tests of the player's ability to interpret and respond to structured patterns under pressure. Again, the behavior was not random. Bullet patterns followed repeatable sequences, and movement cycles could be learned. Success depended on recognizing these patterns quickly enough to respond effectively.

Across repeated play, the structure of the gameplay became clearer. Enemy waves followed defined sequences. Power-ups appeared under predictable conditions. Scoring rewarded sustained performance. Difficulty increased through the layering of existing rules rather than the introduction of entirely new ones.

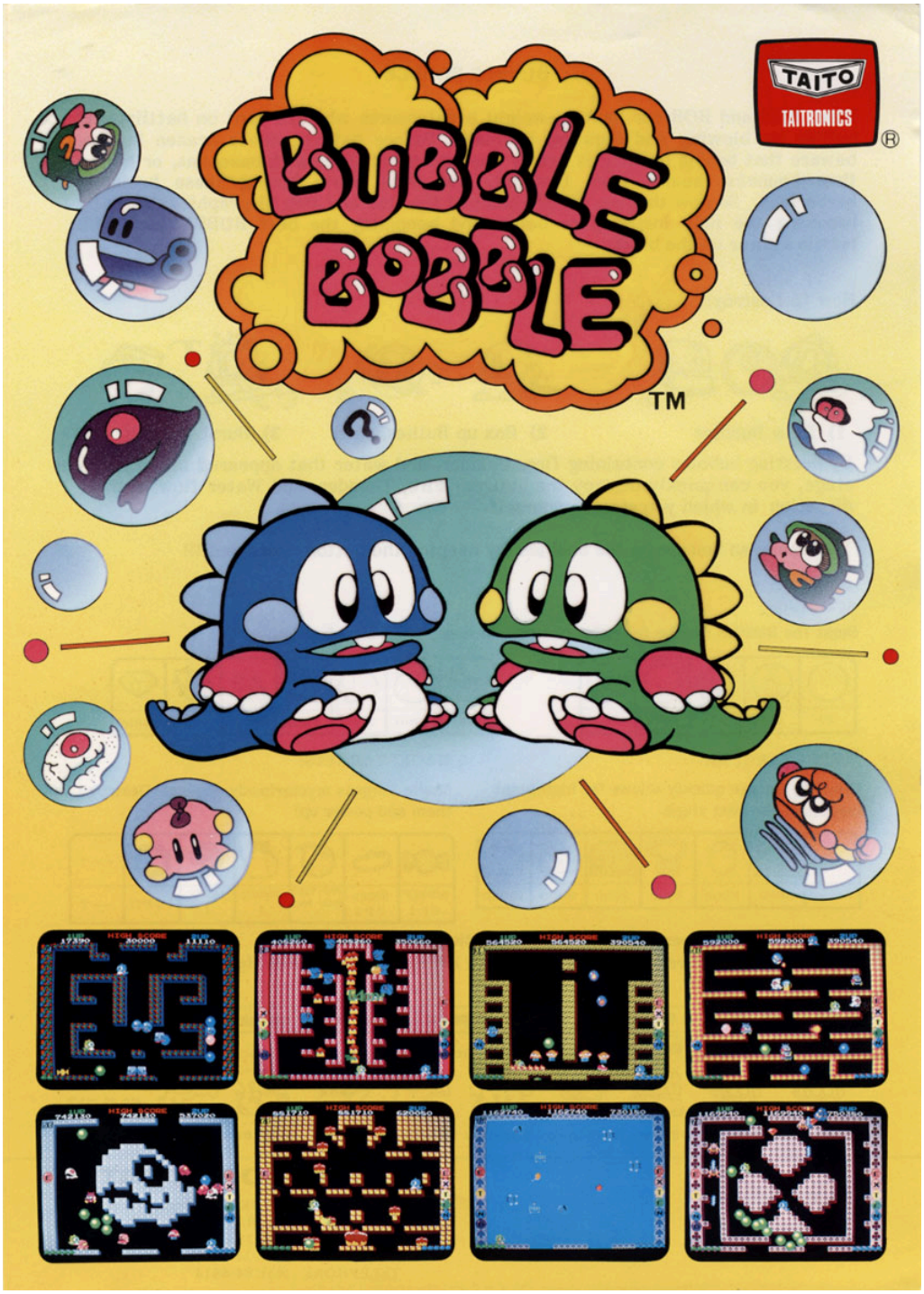
The rules scaled — they did not fundamentally change, and that mattered.

It demonstrated that complexity could emerge from repetition and interaction rather than constant reinvention. The rules remained consistent. The challenge came from how those rules combined over time. As with earlier games, the shift in experience came from perception.

What began as reactive play evolved into structured anticipation. Movement became deliberate, aligned with expected patterns rather than immediate threats. Decisions around power-ups and positioning became strategic rather than instinctive. The game itself remained unchanged — understanding it did.

Over time, it became less about surviving individual waves and more about managing the game as a whole.

Bubble Bobble: Hidden Systems and Cooperation



Bubble Bobble: Arcade Flyer

If earlier games revealed structured systems through repetition, Bubble Bobble extended that concept into something far deeper. It was no longer just about recognizing patterns — it was about uncovering underlying rules that were not immediately visible.

At a surface level, the game appeared simple. Each level was contained within a single screen, either enclosed or with openings at the top and bottom that allowed you to fall through and reappear above. Two players — Bub and Bob — navigated platforms, trapping enemies in bubbles before bursting them to clear the stage. The objective was consistent: eliminate all enemies and progress.

Bubble Bobble: Game Screen

However, this apparent simplicity concealed a layered system of interactions, conditions, and hidden rules. The fundamental mechanic — trapping enemies in bubbles — introduced a temporal element. Enemies were not removed immediately. They existed in an intermediate state, suspended within bubbles that could be interacted with. These bubbles could be used as platforms, altering movement across the level, or left unattended, allowing enemies to escape and re-enter the gameplay in a more aggressive state.

Even at this level, the game was already more complex than it appeared. Enemies were not static entities. If left too long, they became “angry” — moving faster and behaving more aggressively, increasing the pressure on the player. The game dynamically adjusted difficulty based on player behavior, not

just progression. Wait too long, and an additional element would appear — something that could move freely and actively hunt you down.

Scoring introduced another layer of structure. Defeating multiple enemies simultaneously resulted in exponentially increasing rewards. The sequence — 1000, 2000, 4000, 8000, 16000, 32000, 64000 — reinforced the same doubling pattern seen in earlier games, what technical people would identify as base-2.

What mattered was not simply eliminating enemies, but how they were eliminated. Grouping enemies together before bursting them produced disproportionately higher rewards. The game incentivized control, planning, and timing over immediate action. This principle extended further through item drops.

Each defeated enemy transformed into a collectible item — food, gems, or chests. These items were not arbitrary. Their value and type were influenced by how enemies were defeated. Larger group eliminations produced more valuable rewards, reinforcing the same exponential structure.

There was also the candy cane — it gave a significant bonus and turned every bubble into a bonus as well. Over time, it became clear that the game was not rewarding survival alone — it was rewarding strategy, specifically the ability to optimize bonus generation.

Beyond direct player actions, the game also introduced environmental effects through special bubbles that altered the behavior of an entire level. These bubbles appeared under specific conditions and, when activated, affected all entities within the gameplay.

- Water-filled bubbles, when burst, would release a cascade that flowed through the level — but only where the structure allowed it, moving through gaps and descending across platforms. Enemies caught in this flow were swept away and eliminated, demonstrating that the environment itself could become an active component of the gameplay. Other bubbles introduced directional effects.

- Lightning bubbles discharge energy across the screen based on the opposite direction the player was facing at the moment of popping, reinforcing the importance of orientation and timing. Additional power-ups extended these interactions further. Fire-based abilities

allowed for direct area attacks, while specific items — such as crosses — triggered global effects that removed enemies entirely.

- Potions introduced a different form of interaction — temporarily altering the level into a high-reward state, where bonus collection became the primary objective.

- Umbrellas enabled progression control, allowing players to skip ahead multiple levels and bypass sections of the game entirely.

Each of these mechanics operated within defined rules, yet introduced variability in how those rules could be applied, reinforcing that the behavior was not static but responsive to both player input and conditional events.

The cooperative nature of the game introduced another dimension. Unlike previous games, Bubble Bobble was designed around two players operating within the gameplay at the same time. Each player shared space, resources, and outcomes. Movement, positioning, and timing were no longer individual decisions — they were interdependent. This created subtle asymmetries.

Player positioning affected item collection. In situations where both players occupied the same space, the scoring system appeared to prioritize one player over the other when allocating rewards. While not explicitly documented, these behaviors were observable through repeated play. It became possible to influence outcomes by understanding these biases. This was not cooperation in the traditional sense — it was coordinated interaction within a shared gameplay.

Beyond the visible mechanics, Bubble Bobble contained hidden layers — systems that were not explained and, in many cases, not intended to be immediately understood. Special conditions triggered secret, undocumented behaviors.

- Collecting letter bubbles to spell “EXTEND” awarded additional lives and advanced progression — these bubbles appeared based on how many enemies you burst at the same time.

- Reaching specific levels (20, 30, 40 and 50) without losing a life unlocked secret rooms filled with high-value items (10,000 points) — they also contained encoded messages for the player to figure out

that would help unlock more secrets of the game.

- Certain scoring conditions triggered unexpected outcomes. Through experimentation, we discovered that if specific digits — the hundreds and tens — in the score aligned when the final enemy was defeated, every bubble on the screen would transform into collectible bonuses.

Players didn't put much thought into it — but we knew how to make it happen.

There were also secret tweaks embedded even deeper — hidden before the game had even begun. Before even inserting a coin, it was possible to enter sequences of inputs at the title screen. These were not documented. They were discovered, shared, and refined within the Arcade community.

Entering specific combinations on the title screen before inserting a credit would activate alternate modes.

- "POWER UP" altered the player's starting state — providing immediate access to important power-ups, such as increased speed and enhanced bubble-blowing capabilities.

- "ORIGINAL GAME" modified how the gameplay handled bonus levels. Secret doors — normally only available after playing without losing a life — would always appear, providing access to hidden levels and shortcuts.

These modes effectively changed the conditions of gameplay before play even began. They were not cheats in the traditional sense — they were alternate configurations of the game itself (though, to be honest, they were cheats).

The structure of progression reinforced the same layered design.

The game contained one hundred levels, each introducing variations on existing mechanics rather than entirely new ones. Difficulty emerged through combination — more enemies, tighter spaces, and more complex interactions between movement, positioning, and timing.

Bubble Bobble: Final Level (100)

At the final level, the game presented its most significant deviation — a large boss that couldn't be captured in a single bubble. Instead, you needed to obtain the lightning potion, creating a wave of lightning bubbles that, when popped, would traverse across the screen — hopefully hitting the boss.

With enough hits, the boss would eventually be encapsulated in a bubble, but only for a short time. You had to react quickly and pop it before the opportunity passed. The music intensified during this phase, reinforcing the urgency — you had to be fast, or you were done.

The game could be completed in single-player mode. However, doing so revealed that the experience was incomplete. The message was explicit: the true ending required two players. The game had been designed with that requirement from the beginning. It was not optional — it was fundamental. Over time, the nature of play changed — the objective was no longer simply to progress.

The game rewarded those who explored beyond its surface. It rewarded those who observed, tested, and refined their understanding. What appeared to be a simple platform game revealed itself to be a layered system of rules, conditions, and hidden interactions. These effects were not just observed, but anticipated — used deliberately to control outcomes rather than react to them. It was not just about playing the game. It was about understanding it.

Patterns Recognized

What began as a way to waste coins became something far more influential.

The arcade was not simply a place of entertainment. It was an environment built on structured systems — systems designed to reward attention, repetition, and understanding. Each game operated within its own set of rules, yet shared common principles. Movement followed patterns. Difficulty increased through layering rather than randomness. Rewards were tied not just to success, but to how that success was achieved.

With repeated exposure, those principles became increasingly visible. What initially appeared chaotic began to resolve into structure. Enemy behavior could be anticipated. Timing could be controlled. Outcomes could be influenced through positioning, sequencing, and deliberate action. The games themselves did not change — understanding them did.

There was a clear distinction between playing and learning. Playing relied on reaction — responding to what appeared on the screen in the moment. Learning required observation — identifying patterns, testing assumptions, and refining behavior over time. The longer one remained within the gameplay, the more predictable it became. This applied across all the games.

In Frogger, movement patterns defined success. In Galaga, state changes introduced controlled risk. In 1942, progression emerged through scaling complexity. In Bubble Bobble, hidden systems and conditional behavior rewarded deeper exploration. Each game revealed a different aspect of structured design.

Taken together, they formed a consistent pattern. Systems, regardless of their apparent complexity, were governed by rules. Those rules could be observed. Once observed, they could be understood. This realization did not arrive suddenly — it emerged gradually, shaped by repetition and reinforced through experience. Each failure provided information. Each success confirmed an understanding. Progress was not the result of chance — it was the result of refinement. The arcade environment encouraged this process.

There was a cost to participation — each attempt required a coin, and that constraint reinforced focus. There was no incentive to waste time or act without intent. Every decision carried weight, every action had consequences

— the games demanded attention. In return, they offered clarity. At the same time, something else was changing — the experience was no longer limited to the screen.

The arcade was a social environment as much as it was a technical one. Techniques evolved through interaction, not isolation. Patterns and knowledge were not just recognized — they were shared. Strategies were discussed, refined, and passed between players.

Observations became collective knowledge. This period marked a shift — the focus moved from participation to learning and understanding.

The games were no longer just challenges to be completed. They became systems to be explored — understood in terms of how they could be influenced, optimized, and, in some cases, manipulated.

Once patterns became visible in one game, they became easier to recognize in others. The process of observation, testing, and refinement was transferable. It was not limited to games — it was a way of thinking.

The transition from player to observer was subtle, but significant. It marked the point where interaction with games became intentional, and once that shift occurred, a new question emerged — not how to play the game, but how it had been designed to maximize your time in front of it.

The Mentorship of Michael Leishman

The transition from observing systems to understanding them did not happen in isolation. It required an environment where curiosity could be explored without restriction, and more importantly, someone willing to create that environment. It was not about instruction in the traditional sense — it was about access, encouragement, and the freedom to explore without constraint.

At Balga Senior High School, that role was filled by Michael Leishman, the head of computing, whose approach to teaching would prove to be quietly transformative. Before formal classes began, he opened the computer laboratory to a small group of students. It was not structured, and it was not presented as formal learning. For most of us, it was simply an opportunity to use computers — something that was otherwise limited or inaccessible.

Microbee Computer with Monitor and Disk Drive

The attraction was straightforward. The lab contained a suite of Microbee systems, commonly used in Australian schools at the time, and they could run games. That alone was enough to draw us in.

Most of the students who attended were there for that reason alone. Disks were shared, machines were occupied quickly, and the focus was on playing. The sound of drives spinning, the delay as systems loaded, and the anticipation of what would appear on screen were all part of the experience.

The environment was informal, but not careless. We were taught to handle the equipment with precision. The 5.25-inch floppy disks were fragile, and access

to them was limited. Michael Leishman ensured that everyone understood their value, guiding students through loading programs and managing the machines. At a surface level, it was a room full of students playing games. In reality, something else was beginning to take shape.

Michael recognized early that a few students were interested in more than just the outcome of the games. The question was no longer limited to how to play, but extended to how the machines operated. His response was subtle. He did not instruct directly — instead, he introduced opportunity. He would later say that he never really taught us anything — that we simply worked it out ourselves.

A small collection of books began to appear on a nearby table. Titles such as Programming in Z80 and the Microbee Technical Reference were placed within reach, without explanation or expectation. For most students, these were ignored. The draw of immediate interaction — the games — was stronger than the effort required to understand the underlying mechanics.

For some of us, however, those books represented something different. They were not typical reading material for students of that age. The content was dense, technical, and largely unexplained within the context of our formal education. There was no structured guidance — only examples, fragments, and concepts that had to be pieced together through experimentation. Yet they provided a path — one that extended beyond interaction into creation.

The laboratory environment reflected this progression. Most machines were equipped with green monochrome displays — functional, consistent, and widely available. A smaller number featured amber displays, which carried a subtle sense of distinction.

Beyond these, there was a single system that stood apart — Michael's personal BBC Micro, connected to a color display. Access to it was not restricted, but it was not freely given. It became associated with curiosity. Those who moved beyond simply playing — those who explored — were given the opportunity to use it.

Within a relatively short period of time, the shift from playing to creating began to take hold. Using examples from the available material, I started to understand the fundamentals of BASIC programming. Simple programs emerged first — interactive routines that accepted input and returned output,

behaving in ways similar to basic tools such as calculators or question-and-response systems.

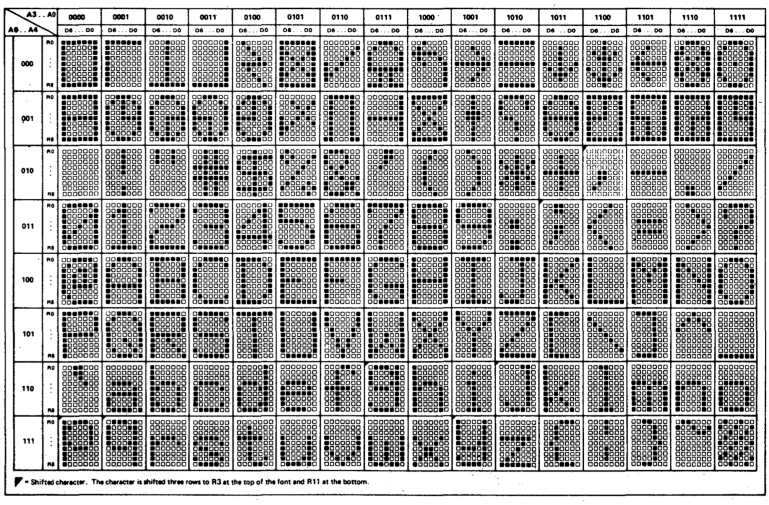
From there, the exploration extended into Z80 assembly language. At that stage, the level of understanding was incomplete. The instruction set was complex, and the relationship between code and behavior was not immediately obvious. However, the process of experimentation — modifying examples, observing results, and refining understanding — began to establish a foundation.

The machines themselves reinforced this learning. The Microbee systems, typically configured with 32 KB of memory and running CP/M, imposed clear constraints. Programs had to be efficient. Memory usage was visible. Execution time was noticeable. Every instruction carried a cost. These limitations did not restrict learning — they defined it.

One of the most compelling challenges was understanding how these machines produced graphics. The systems were primarily text-based, yet the games we interacted with demonstrated movement and visual representation that extended beyond simple character output. The question was not whether it was possible — but how it was achieved.

The games we had access to — such as Hoards of the Deep Realm, a Lode Runner clone written by Vaughan Clarkson and published by Honeysoft — demonstrated movement and visual representation that extended beyond simple text output..

FIGURE 12 – MCM66740 PATTERN



Microbee Character ROM (Motorola MCM66740)

The answer lay in the programmable character generator (PCG). The system allowed individual characters to be redefined at a pixel level, in parallel with the ROM-based font provided by the Motorola MCM66740. Each character consisted of a grid — typically an 8 by 16 grid — where each position could be either active or inactive. Designing a character required translating a visual idea into a binary representation.

This process often began on graph paper. Each filled square represented a bit set to one. Each empty square represented zero. Rows of these values were then converted into decimal form, allowing them to be stored and rendered by the system. This was not simply drawing — it was encoding. Each pixel corresponded to a binary value, with active pixels represented as 1 and inactive as 0. These binary sequences were then converted into decimal values, allowing the system to store and render them as custom characters.

The limitations of the hardware became immediately apparent. A standard Microbee with 32 KB of memory supported only a limited number of programmable characters — providing 2 KB (128 unique characters), sufficient for small graphical elements, but not for full-screen representations.

However, access to a 128 KB system expanded those boundaries, increasing this to 16 KB (1024 characters), allowing for a greater number of custom characters and more complex visual output. Capability was not fixed — it depended on the resources available and how they were used. Options to upgrade were available, expanding to 32 KB (2048 characters) on Premium Plus+ models.

From these foundations, further experimentation followed. Drawing lines became an early challenge. Without knowledge of efficient algorithms, the approach relied on mathematical expressions we learned in class, such as $y = mx + b$. On a 3.375 MHz Z80 processor, this was computationally expensive. Execution was slow, and visual output required patience. At the time, this was not seen as a limitation — waiting was expected. More complex shapes, such as circles, introduced additional challenges.

The mathematics became more involved and the inefficiencies more apparent. Yet the objective remained consistent: translate an idea into a sequence of instructions that the machine could execute. The result was secondary — the process was primary.

The environment extended beyond the machines themselves.

Each morning, a small group of us would arrive at the laboratory well before the start of the school day. Waiting outside at 6:30 am, we anticipated the moment the lab would open, at 7:00 am. For two hours, until classes began, the focus was uninterrupted. We played, we experimented, we learned. This routine was consistent — it was deliberate.

Michael Leishman observed this progression. Without formal instruction, he began to introduce challenges.

One such task involved creating a system to display daily notices for the school. The objective was practical. A Microbee would be placed in the administration building, cycling through messages loaded from disk. The system needed to read data, manage timing, and present information clearly.

I accepted the challenge. The resulting program operated as intended, cycling through notices and updating as required. It served a real purpose within the school environment — it was no longer an exercise, it was an application.

Formal education followed. Enrollment in computer science classes introduced structure to what had previously been exploratory learning. However, the foundation had already been established.

The material presented in class was familiar, and the approach encouraged by Michael remained consistent. There was no emphasis on simply following instructions — creativity was encouraged, exploration was expected. I was already committed — what began as curiosity had become something more clearly defined.

This period led to the development of increasingly complex programs. Simple text-based games emerged, building on earlier experiments and expanding in structure and interaction. The work did not go unnoticed.

At one point, a representative from the education department visited the school to evaluate what had been created. The programs themselves were modest by later standards, but at the time, they represented a level of understanding that extended beyond the expectations of the curriculum.

I was not alone in this progression. Among my peers, there was a strong sense of competition, particularly in mathematics. Performance was closely

measured, and small differences in results carried significance. A mark of 97% might be followed by another achieving 98%, always pushing slightly ahead. That dynamic shifted within computing.

This was a domain where I found a clear advantage. Results were consistent, often achieving full marks, and the gap between myself and others became more defined. For the first time, there was a sense of ownership. This was no longer simply an interest — it was an area of strength.

By the end of Year 10, the trajectory was established. Time spent in the laboratory extended into other environments, including the school library, where access to machines allowed continued exploration beyond structured hours. The limitation was no longer interest — it was access.

Recognizing this, Michael provided a solution. He understood that access outside the school environment was limited and arranged for one of the Microbee systems to be made available for use at home.

This changed everything — the environment was no longer bound by time. Exploration could continue without interruption. The constraints of shared access were removed, replaced by the ability to experiment, fail, and refine without limitation — the process accelerated.

Computer science was not offered beyond Year 10 within Balga Senior High School. However, this did not represent an endpoint. The direction had already been established, and Michael ensured that pathways remained available, facilitating enrollment in a school where computing continued as part of the curriculum. That transition led to Morley Senior High School.

Michael introduced me to the computer science teacher responsible for continuing that journey — Mike Robey. The association was immediate. The name carried a sense of familiarity that was difficult to ignore, echoing the Microbee systems that had defined those early years. It did not take long before I began referring to him, privately at first, as “Microbee.” It was not a name that was particularly appreciated when used more openly.

Years later, the connection remained.

Contact was re-established, and the relationship extended beyond its original context. The influence of that early environment — and the role Michael Leishman played in creating it — remained clear. He had not imposed

direction; he had created the conditions for it. The outcome was not the result of instruction alone, but of opportunity, environment, and the presence of someone who understood when to guide — and when to step back.

Michael would often reflect that over the years he had seen thousands of students pass through the classroom, and only a small number he described as “diamonds in the rough.”

AI Rendering of Michael Leishman

Today, Michael is retired, spending his time pursuing photography and enjoying good wine. The same qualities remain evident — patience, attention to detail, and a quiet appreciation for craft — that once shaped a generation of students without ever being imposed upon them.

Course Correction: Finding Direction Through Failure

While I had been a strong academic performer throughout school, I was far from disciplined in the traditional sense. Good grades created a perception of consistency, but they did not reflect the full picture — I got bored easily.

Structure, when it lacked challenge, became something to push against rather than follow. Skipping school, ignoring rules, and testing boundaries were not isolated incidents — they were part of a pattern. I was never alone in this. There was always a small group of us operating just outside the expected path, navigating consequences as they came.

The system had mechanisms to control this. Daily roll call at 9:00 am was one of them. Absence required justification — a note from your parents, an explanation that could be verified. It was designed to enforce accountability. In practice, the outcomes were not always consistent.

Academic performance carried weight — students who struggled often faced stricter consequences, while those who performed well were given more latitude. That difference was never explicitly stated, but it was understood.

It did not remove consequences entirely — I still spent time in the principal's office — but it changed how they were applied. It created a quiet contradiction. Expectation and behavior did not always align, and I learned early how to operate within that space.

Changing schools was intended to be a reset. The move to Morley Senior High School represented an opportunity to focus, particularly as it offered subjects that were not available locally. Gaining entry required an exception to zoning rules — something that was discussed directly with the headmaster in a formal meeting with my parents. It was a clear moment of accountability, one that suggested a shift in direction was expected.

That assumption did not last long. The same headmaster from my previous school had also moved to the new one. Familiarity followed me — what was intended as a fresh start quickly became a continuation, with history intact. Walking through the school during recess and lunch, surrounded by friends, it was not unusual for him to pass by and acknowledge me by name.

For my peers, this carried a different weight. We were at an age where identity mattered, and being publicly associated with academic recognition was not always desirable.

Success came with visibility. Mathematics competitions, distinctions, and academic performance were occasionally brought to the attention of the wider school. These moments were intended as recognition, but they did not always align with the social expectations of the time. There was a balance to manage — performance and perception did not naturally coexist. The broader system reinforced that tension.

In Australia, formal schooling begins at the age of six and continues through ten mandatory years, followed by an optional two-year period focused on tertiary entrance. These final years are structured around ranking — students are assessed relative to others across the state, and those rankings determine access to university placements.

The process is competitive and limited. There are only so many positions available, and entry is determined by performance. For students approaching the end of secondary school, the expectation is clear: pursue higher education, enter a trade, or move directly into the workforce.

The decision is made early — for many, it is made without full clarity. At that stage, direction was less about certainty and more about momentum. Choices were influenced by peers, availability, and assumptions rather than deliberate planning. I followed the expected path, applied, and was accepted into university. On paper, it was the next logical step — in practice, it would become the first significant misstep. The decision itself was not deeply considered.

Curtin University Logo

University offered multiple pathways, and while computing was an obvious strength, the choice I made did not align with it. I enrolled in a degree titled Information Technology at Curtin University of Technology. On the surface, it appeared relevant. In reality, it was a commerce-focused program with only a minor emphasis on computing.

The distinction became clear almost immediately. The core subjects were not technical. Law, Accounting, Economics, and Marketing formed the foundation of the program. These were disciplines that required a different way of thinking — structured, abstract, and often disconnected from the logical frameworks that I was comfortable with.

My strengths had always been grounded in mathematics, computing, and systems-oriented thinking. This environment was different. It did not suit me. The contrast was stark — the computing components of the course required little effort, while the surrounding subjects demanded significantly more.

Despite that effort, the results did not follow. Within six months, performance had declined to the point where continuation was uncertain. Courses were barely passed, some were failed, and the trajectory was clear.

I was close to leaving university entirely — forcing a critical decision. Continuing along the same path was not viable. The mismatch was too significant, and the consequences were immediate.

The only option was to correct the course, but the system did not easily allow for it. Entry into the engineering faculty, and specifically into computer science, was competitive. Places were limited, and each year new applicants competed for them — transferring was not guaranteed.

Meeting with the dean of the engineering faculty became the next step. It was not a casual discussion — the expectation was clear: a mistake had been made, and justification was required to correct it. The question was not simply whether I wanted to change direction, but whether I could demonstrate the capability and commitment to succeed if given the opportunity.

The responsibility was mine — this was the first time the outcome was not assumed. Up until that point, academic success had come with relatively little resistance. This was different — there was no established position, no expectation of success, and no guarantee of acceptance. The situation required focus in a way that had not been necessary before.

The response was immediate. Attention shifted entirely to the areas that mattered. Programming, engineering fundamentals, logic, and mathematical reasoning became the focus. The difference was not just in the subject matter, but in engagement. The material aligned with how I thought, and the results

followed accordingly. Performance changed rapidly. From a position of near failure, results moved into the top tier of the cohort. What had previously required significant effort now came naturally. The contrast was not subtle — it was clear that the issue had never been capability, it had been alignment.

The decision to transfer was approved. With that, the direction was corrected. The program shifted to computer science, supported by minors in computer graphics and geographic information systems.

The earlier exposure to commerce subjects was not lost. While initially difficult, those courses had introduced concepts that would only make sense much later. At the time, they felt irrelevant — in hindsight, they were foundational.

From that point forward, the trajectory stabilized. The environment was no longer something to navigate around, but something to engage with directly. The concepts were familiar, the challenges were meaningful, and the outcomes were consistent.

Academic performance reflected that shift, but more importantly, so did confidence. The uncertainty had passed, and direction had been established.

Although the initial choice of degree had proven to be a misstep, it was not without value. The commerce-focused subjects that had once felt disconnected — economics, accounting, and marketing — began to take on a different significance over time. At the point of studying them, the relevance was unclear. The concepts did not align with how I approached problems, and the results reflected that disconnect.

Understanding came later — those early exposures introduced ideas that would remain dormant for years. Concepts around value, trade-offs, market dynamics, and financial structure did not immediately integrate into my thinking, but they were retained.

Over time, as experiences expanded beyond purely technical domains, those foundations began to surface. What initially appeared to be a mistake had introduced a different kind of perspective. The learning had not been wasted — it had simply been delayed.

At the same time, the structure of university life introduced a new set of practical realities. Unlike school, where the environment was largely contained,

university required a level of independence that extended beyond academic performance. I had moved out of home relatively early, and while support systems existed, they were limited. Government assistance provided some stability, but it was not sufficient to live independently — the gap had to be filled.

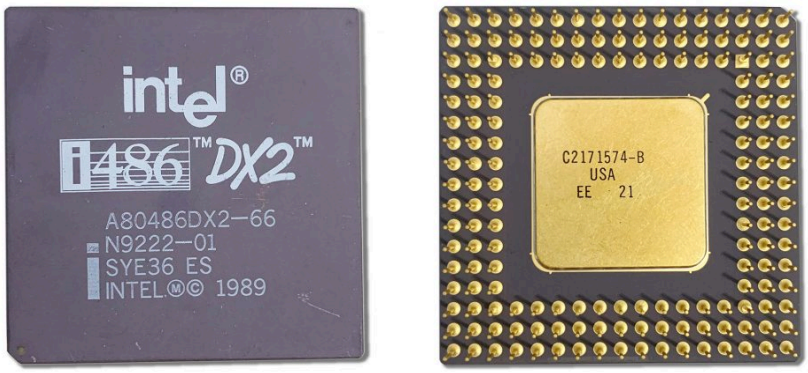
Work became a necessity rather than an option. Consulting opportunities, short-term jobs, and whatever else was available were taken on to cover the basics — rent, food, and transport. Time became a constrained resource, and priorities had to be managed accordingly.

Attendance was no longer guaranteed — lectures were missed, lab sessions were skipped. The structure of formal education became secondary to maintaining the conditions required to continue it. Despite that, the outcomes remained consistent.

The ability to engage with the material independently allowed progress to continue, even without full participation in scheduled classes. The model had shifted. Success was no longer defined by adherence to structure, but by the ability to operate effectively without it.

University also marked the point where computing shifted from shared access to personal ownership. Up until then, interaction with computers had largely been confined to schools and laboratories — access was scheduled, environments were shared, and time was limited.

That changed when I purchased my first computer.



Intel DX2-66 CPU

The system — a 486DX2 66 MHz, 4 MB RAM, 20 MB HDD — represented a significant financial commitment. At the time, it cost over $2500 AUD, a substantial amount given the circumstances.

Securing it required a loan, with my father acting as guarantor. It was, in many ways, the equivalent of taking on a small mortgage.

The intention was practical. It was meant to support university work, provide a consistent environment for assignments, and remove reliance on shared facilities. In practice, it became much more than that.

The machine introduced a new level of immersion. For the first time, access was unrestricted. There were no lab hours, no queues, and no imposed limitations. Exploration could occur at any time — and often did. While the original purpose was academic, it did not take long for other uses to emerge.

Gaming was one of them — this period coincided with rapid advancements in PC gaming. Titles from id Software, including Commander Keen, Wolfenstein 3D, and Doom, demonstrated what was possible within the constraints of the hardware. Other developers, such as Psygnosis, contributed equally compelling experiences, with games like Lemmings offering a different kind of challenge.

The machine was no longer just a tool — it was a platform. Alongside this, access to networked systems introduced another dimension.

Through university resources, it was possible to connect to Sun Microsystems machines using VT100 terminal emulators. The environment was entirely text-based. There were no graphical interfaces, no web browsers, and no search engines as they would later exist. Despite that, it was connected.

Using protocols such as Telnet, it was possible to access remote systems. Internet Relay Chat (IRC) enabled real-time communication with individuals across the world. Bulletin board systems provided a mechanism for sharing software and information.

Primitive by today's standards — however it was global — connectivity extended beyond the constraints of the university campus.

USRobotics 28.8K Faxmodem

At home, access to the university modem pool allowed persistent connections to be established using a 28.8K modem — painfully slow by today's standards.

Scripts were written to automatically reconnect when disconnected, maintaining continuous access. This removed the need to rely on physical presence in university labs. The environment shifted again — local access became global access.

At the same time, the exploration of how software was constructed continued. A small group of us remained interested in understanding how the games we were playing were built. This led to experimentation with graphics programming under MS-DOS, specifically using Mode 13h, which provided a 320×200 resolution with 256 colors. It was one of the simplest entry points into pixel-based graphics within the MS-DOS environment..

Michael Abrash introduced Mode X in 1991 through his Ramblings in Realtime column in Dr. Dobb's Journal, detailing a 320×240 resolution with 256 colors and square pixels that significantly outperformed standard Mode 13h.

His book, Graphics Programming Black Book, became a gold mine — if you could get access to it. It showed how to enable page flipping for smooth animation and how to utilize the VGA's plane-oriented hardware to process pixels in parallel, improving performance by up to four times over standard modes.

Using tools such as Turbo Pascal, I began experimenting with drawing routines, sprite handling, and basic rendering techniques. The objective was straightforward — to recreate elements of the games that had inspired me. — the results were limited.

While individual functionality could be implemented — drawing shapes, moving objects, managing simple interactions — bringing everything together into a complete game proved significantly more challenging.

The gap between understanding individual techniques and building cohesive systems was clear — the process continued. Beyond personal experimentation, there was a broader movement taking place. The demo scene was emerging as a distinct culture within computing.

Groups such as Future Crew demonstrated what could be achieved through deep technical understanding and creative application. Their demos, including productions like Second Reality, pushed the boundaries of what was possible on standard hardware.

These were not games — they were demonstrations of capability, built using pure CPU without hardware acceleration. Alongside visual output, audio played an important role.

File formats such as MOD and S3M files, rooted in earlier systems like the Commodore 64, introduced structured approaches to music composition using sampled sound.

These formats became widely distributed through bulletin boards and networked systems, further expanding the range of what could be experienced and studied. The environment was evolving rapidly.

It was during this period that computing shifted once again. Up until that point, the machines I interacted with — whether for study, experimentation, or entertainment — had largely been defined by their interfaces.

Operating systems such as Windows 3.1 provided structure, but they also imposed constraints. Access to the underlying system was limited, and interaction was shaped by what the software allowed.

That boundary began to change with the introduction of Linux. At the time, Linux was not widely accessible in the way modern systems are today.

There were no simple installers, no automated configuration, and very little abstraction from the underlying hardware — gaining access required effort.

The process began with obtaining a distribution, typically available as a series of floppy disk images.

Linux Slackware — Distribution in Floppy Disk Format

Slackware was one of the more common options, but acquiring it was not immediate. It required downloading large volumes of data over slow connections, often in segments, with the constant risk of interruption.

Each step demanded patience, and failure was part of the process — completion was not guaranteed, and corruption of disk images was common.

Installation was only the beginning. Once the system was in place, it rarely functioned as expected without further work. Hardware compatibility was not assumed. Support for components such as CD-ROM drives, graphics cards, and sound devices required manual configuration.

In many cases, this extended to compiling the kernel itself — selecting the appropriate drivers and resolving conflicts that were not always well documented. It was a process of trial, error, and persistence.

This was fundamentally different from anything that had come before. The system was no longer something to be used — it was something to be understood.

Configuration required awareness of how the machine operated at a lower level. Decisions had consequences. Errors were visible, and resolving them required investigation rather than repetition.

There was no abstraction to rely on — for those willing to invest the time, the reward was control. Every component of the system could be examined, modified, and optimized. The boundaries that had previously existed between user and machine were reduced.

Understanding was no longer limited to behavior at the surface level — it extended into the structure of the system itself.

The computer had become more than a tool — it had become something to work with and tweak. The experience demanded a different mindset — patience was essential — progress was incremental. Success was rarely immediate, but when it was achieved, it was understood.

Each step forward represented not just completion of a task, but an expansion of capability. This was not passive interaction — it was active engagement.

By the end of this period, the path had become clear — the questions had changed. It was no longer about how things worked, but what could be done with that understanding.

The foundations were in place — the next phase would test them.

PREVIEW EDITION

to purchase a copy of the book
visit the following website

www.ardiri.com

the book is available in both
digital and physical form.

Technology, Entrepreneurship and the Reality of Building Systems is the story of what actually happens behind the scenes of innovation — beyond the headlines, beyond the hype, and far from the clean narratives often associated with success in technology.

Aaron Ardiri's journey spans from the earliest days of home computing to the rise of artificial intelligence, tracing a path defined not just by what was built, but by how it was built. From typing hexadecimal code into Z80-based systems, to mastering assembly programming in an era before documentation was readily available, his early experiences were shaped by curiosity, experimentation, and a need to understand systems at their most fundamental level.

That mindset carried forward into the arcade era, where pattern recognition, timing, and optimization became second nature — forming the foundation for a career that would repeatedly place him ahead of the technology curve, often years before the market was ready.

As a pioneer in mobile computing, Aaron played a role in shaping the early Palm OS ecosystem, working across game development, emulation, and digital distribution, while navigating the less visible side of the industry — piracy, protection, and the blurred lines between innovation and enforcement. His work evolved into enterprise systems, mobile platforms, and large-scale architectures, culminating in acquisition, corporate integration, and eventual early retirement.

But stepping away was not the end.

Returning through the world of IoT, Aaron re-engaged with the same constraints that defined his early career — limited resources, complex systems, and the need for efficient design — this time focusing on security, platform architecture, and real-world deployment at scale. His work ultimately extends into artificial intelligence, exploring how structured systems, data, and distributed architectures converge to shape the next generation of technology.

This is not just a story of technology or entrepreneurship.

It is a story about timing, about being early, about building in environments that are often unpredictable, and about understanding that success is rarely defined by what you know alone, or even who you know, but by how you approach the problem.

A candid, technical, and deeply personal account of a career spent not just building systems — but understanding them.

ISBN 978-91-531-9116-2