

- +
- • Programming for 8-bit Commodore systems using a modern IDE

Byron Stout
byron.stout@gmail.com



While you can still program directly on the original hardware, it sure is nice to have the conveniences of a development environment running on a modern PC. Let's use CBM PRG Studio to write BASIC and ML programs for our favorite 8-bit Commodore machines.



Healdton Oklahoma

(population 2,861)

Halfway between everything

That's not me in this picture,
But it happened just like this...



https://external-preview.redd.it/RhfX_IN3oP7orhnRfmOp5XlggMxKjhLGkjnnxhEFMCw.jpg?auto=webp&s=0a86b190e649b0cd2ab130b1c3d36f3b916f0ab6



I was PAID to USE

- DBase IV
- C++
- Borland Pascal
- Borland Delphi
- Visual Basic
- Microsoft Access VBA
- RPG
- COBOL
- Java
- Microsoft ASP
- JavaScript
- VB.NET / ASP.NET
- TSQL
- Microsoft C#
- Allaire ColdFusion
- PHP on LAMP

I learned for FREE

- BASIC !
- HyperTalk
- Pilot
- Turbo Pascal
- 8086 Assembly
- 6502 Assembly
- Fortran
- LISP
- Ada
- Clarion
- Perl
- Python
- Ruby

It's hard to visualize code inside
40 columns and 25 rows

You have to keep a lot of notes...

```
60 PRINT"***** *****"
65 PRINT"7 OR 11 WINS":PRINT"3, 6, OR 12
LOSES"
67 FORTD=1TO1000:NEXT
70 DT=D1+D2:IFDT=7ORDT=11THEN90
75 IFDT=3ORDT=6ORDT=12THEN110
80 PRINT:PRINT"NO WINNING COMBINATION, T
RY AGAIN."
85 FORTD=1TO1500:NEXTTD:PRINTCHR$(147):G
OTO30
90 FORTD=1TO100:NEXT:PRINTCHR$(147)
95 FORP=1TO10:PRINT" YOU WON !!!",:NEX
T
100 PRINT:PRINT"YOU WON $"BT:MN=MN+BT
105 FORTD=1TO1000:NEXT:PRINTCHR$(147):GO
TO20
110 PRINTCHR$(147);"TOO BAD,"
115 PRINT"YOU LOST $"BT
120 IFMN-BT<=0THEN130
125 MN=MN-BT:FORTD=1TO1000:NEXT:PRINTCHR
$(147):GOTO20
130 PRINT"YOU RAN OUT OF MONEY !!!!!!!"
135 END
READY.
```

Number System

We know that all operations in a computer are done using binary number system. There are 2 states in the binary number system: One and zero. One being a HIGH signal and zero being a LOW signal.

Let's take a look at a table with decimal, binary, octal and hexadecimal equivalents.

Decimal	Binary	Octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Conversions to Decimal

Binary to Decimal

Each binary digit has 2^0 2^1 2^2 2^3

- Convert (101101)₂ to 7 digits, reserve the highest digit
- $(1 \times 2^6) + (0 \times 2^5) + (1 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$
- Convert (100)₁₀ to binary
- 2^3 2^2 2^1 2^0
- 1 0 0 1
- 0 1 1
- 2^3 2^2 2^1 2^0

Reflect on the design? Is it a win? Revisit the

⑥ Generativity & fun & purpose in this essay

Use coding & data mining, coding (5 books)

① ✓ 1. (some existing practices)

2. ✓ We writing' does serve for writing

Analogy between writing & coding

② ✓ 2. (we) all write 2 code - as live coding academics

- don't write because function

3. * structuring

4. * Generativity

5. * performance (document?)

Experiments in a live writing

③ 1. - writing (writing)

2. - recording / playback

3. - conversation as tool

4. - interview history / biology

④ Holding back procedural bank into its writing

- notation

1. - towards

2. - towards thinking, writing, reading

3. - systems

4. - knowledge

5. - knowledge

6. - knowledge

7. - knowledge

8. - knowledge

9. - knowledge

10. - knowledge

11. - knowledge

12. - knowledge

13. - knowledge

14. - knowledge

15. - knowledge

16. - knowledge

17. - knowledge

18. - knowledge

19. - knowledge

20. - knowledge

21. - knowledge

22. - knowledge

23. - knowledge

24. - knowledge

25. - knowledge

26. - knowledge

27. - knowledge

28. - knowledge

29. - knowledge

30. - knowledge

31. - knowledge

32. - knowledge

33. - knowledge

34. - knowledge

35. - knowledge

36. - knowledge

37. - knowledge

38. - knowledge

39. - knowledge

40. - knowledge

41. - knowledge

42. - knowledge

43. - knowledge

44. - knowledge

45. - knowledge

46. - knowledge

47. - knowledge

48. - knowledge

49. - knowledge

50. - knowledge

51. - knowledge

52. - knowledge

53. - knowledge

54. - knowledge

55. - knowledge

56. - knowledge

57. - knowledge

58. - knowledge

59. - knowledge

60. - knowledge

61. - knowledge

62. - knowledge

63. - knowledge

64. - knowledge

65. - knowledge

66. - knowledge

67. - knowledge

68. - knowledge

69. - knowledge

70. - knowledge

71. - knowledge

72. - knowledge

73. - knowledge

74. - knowledge

75. - knowledge

76. - knowledge

77. - knowledge

78. - knowledge

79. - knowledge

80. - knowledge

81. - knowledge

82. - knowledge

83. - knowledge

84. - knowledge

85. - knowledge

86. - knowledge

87. - knowledge

88. - knowledge

89. - knowledge

90. - knowledge

91. - knowledge

92. - knowledge

93. - knowledge

94. - knowledge

95. - knowledge

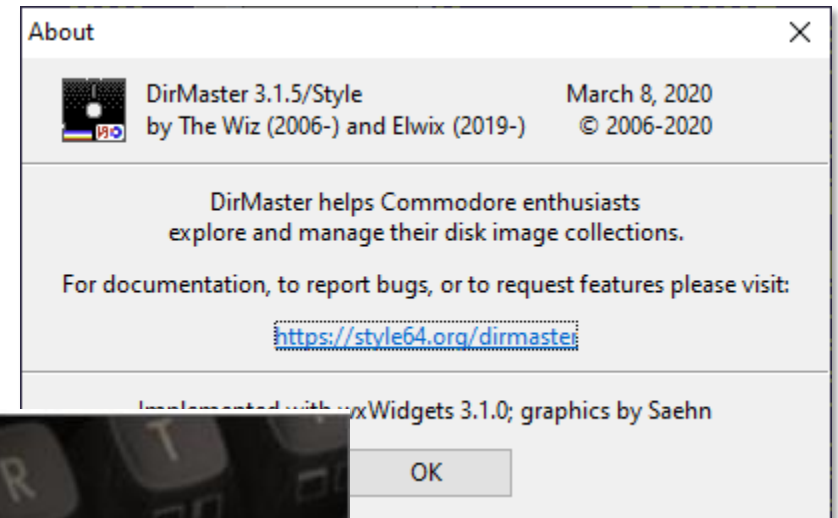
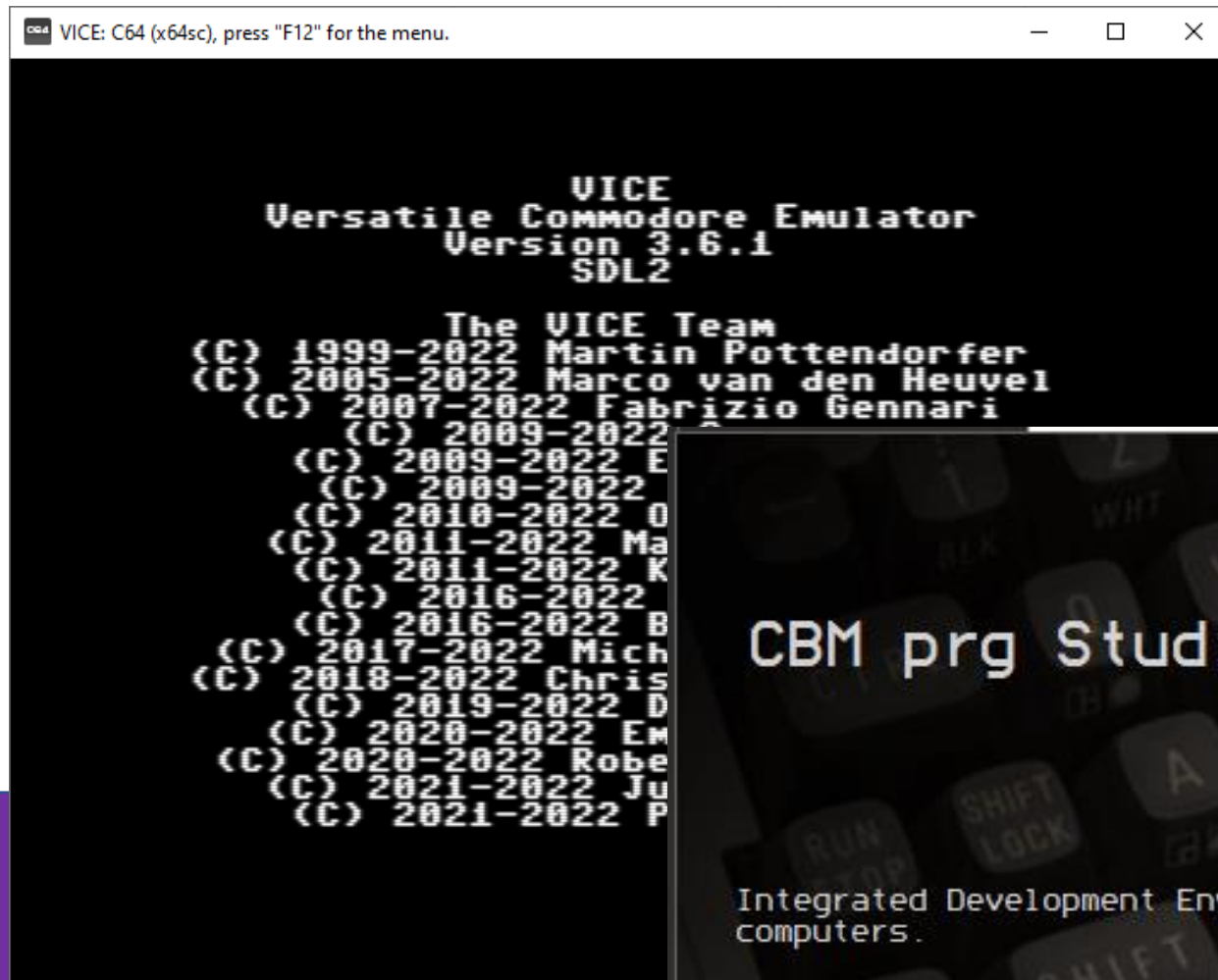
96. - knowledge

97. - knowledge

98. - knowledge

99. - knowledge

100. - knowledge



Tools for using a Windows PC for 8-bit Work

<https://www.ajordison.co.uk>

Arthur Jordison



- First Released 2011
- Freeware
- Not Open Source
- Built on Microsoft.NET Framework
- Support for all Commodore 8-bit Machines

Why Use CBM PRG Studio?

- Git integration
- Sprite Editor
- Character Editor
- Screen Designer
- SID tool
- Screen Code Builder
- Memory Viewer
- Assembler/Disassembler

- Program Import/Export
- Code Formatting
- Renumbering Tool
- Auto Code Formatting
- BASIC Constants
- Renumbering
- Multiple Source Files
- MDI Interface

Code Editor Features

- Auto Numbering
- Auto Complete
- Closing Quotes and Parenthesis
- White space
- Block Comment / Indent
- Listing Comments / Comment Blocks
- Regions
- Screen Code Builder
- Renumbering tool
- Multiple Files

```
5 rem BASIC test program
10 print chr$(147)
15 print "{white}"
20 print "hello vcf east"
30 print

#region !-waiting to press a key
50     print "press <any key> to continue"
60     get a$: if a$ = "" then 60
70     print "{clear}"
#endregion

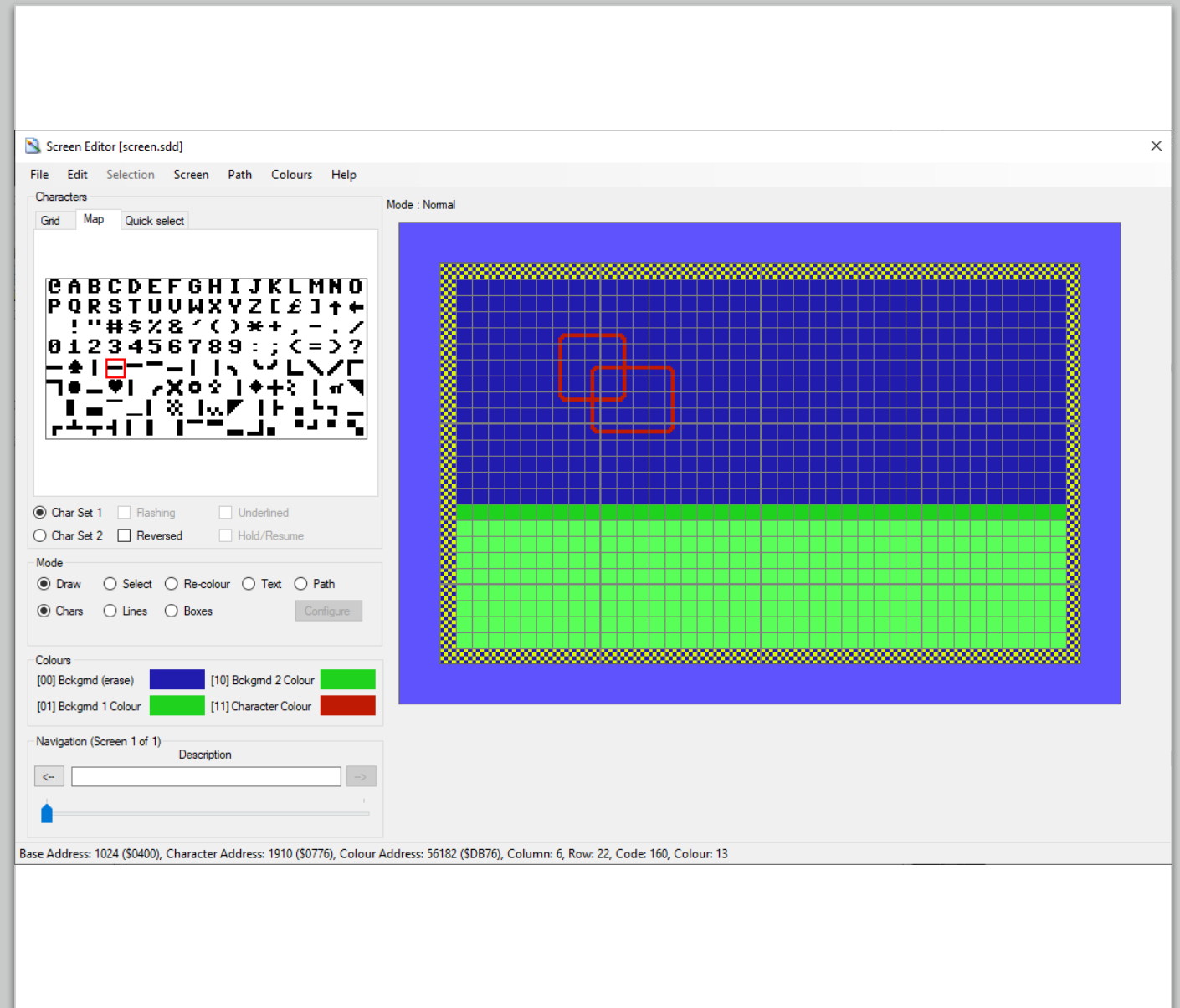
!-=====
!-Print some characters
!-=====

100 for i = 65 to 80
105     print tab(-64 + i);
110     print chr$(i)
120 next i

130 end
```

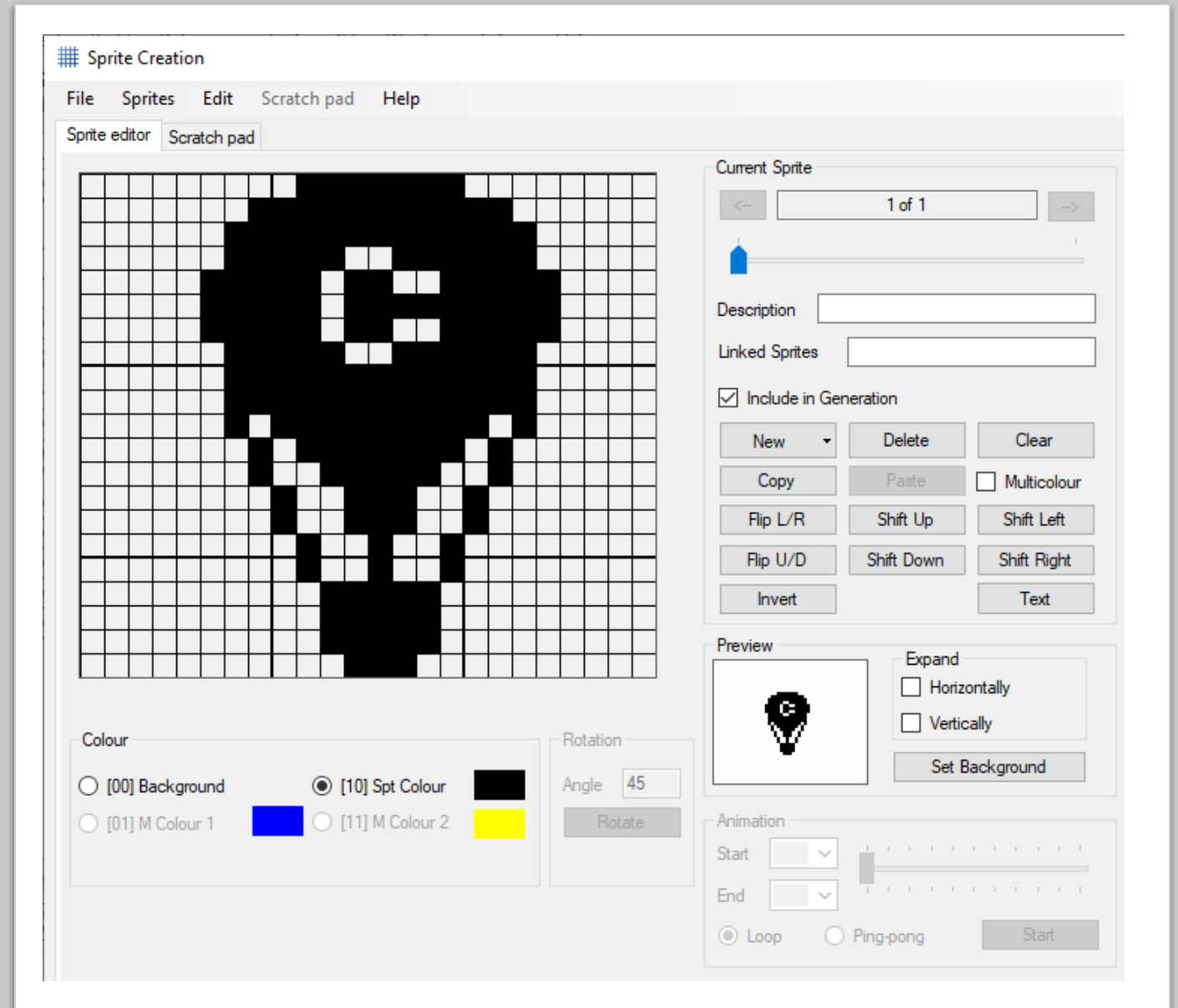
Screen Editor Features

- Draw Mode
- Background / Character Colors
- Line and Box Mode
- Text Mode
- Grid / Map / Quick Select
- Export to Code



Sprite Editor Features

- Import From Source
- Flip Left / Right
- Flip Up / Down
- Shift Left / Right
- Shift Up / Down
- Scratch Pad
- Export To Source



Assembly Features

- 3-pass Assembler
- Labels
- Variables
- Conditionals
- Macros
- Generate a BASIC Loader
- Generate SYS Call

```
1
2 *= $0810
3
4 clear    JSR    $E544
5
6 start    LDA    #02
7          STA    $d020
8          STA    53281
9
10
11         LDX    #65      ; ASCII A
12
13 LOOP     TXA
14         JSR    $E716
15         INX
16         CPX    #91
17         BNE    LOOP
18
19 exit     RTS
```


What did we See? What's Left?

- Git integration
- Sprite Editor
- Character Editor
- Screen Designer
- SID tool
- Screen Code Builder
- Memory Viewer
- Assembler/Disassembler

- Program Import/Export
- Code Formatting
- Renumbering Tool
- Auto Code Formatting
- BASIC Constants
- Renumbering
- Multiple Source Files
- MDI Interface



Next Steps on Your Journey

DOWNLOAD THE TOOLS

CHECK OUT THE HELP FILE

CHECK OUT THE VIDEOS FROM
OLDSKOOLCODER

EXPERIMENT AND
HAVE FUN

Thank You for Listening



Byron Stout
byron.stout@gmail.com