中华学习机

紫金Ⅱ系列

APPLE Ⅱ

朱国江
孙建隆 编著

图形技巧

高等教育出版社

中华学习机图形技巧

朱国江　孙建隆　编著

高等教育出版社

（京）新登字 046 号

内 容 简 介

本书深入浅出地介绍了中华学习机在图形处理方面的各种技巧，种类繁多，应有尽有。不仅包括常规的图形处理技术，还包括图象处理的特殊技巧，既有特色，又有创新，并给出一个通用性好，实用性强的多功能图形管理软件，附有全部程序清单、使用说明和图形资料。

本书与《中华学习机编程技巧》（气象出版社，1988 年 2 月出版），《中华学习机数据处理》（气象出版社，1989 年 12 月出版），《中华学习机汉字软件》（气象出版社，1990 年 2 月出版），《中华学习机图形管理》（气象出版社，1991 年 2 月出版）等成系列书。以国家教育委员会指定推广的优秀国产机型——CEC－I 中华学习机为选型机，并完全适用 APPLE－Ⅱ 兼容机、紫金－Ⅱ 系列机。

本书适用于从事微机应用、管理人员和广大青少年及微机爱好者自学，也可作为各种类型培训班的教学参考书。

中华学习机图形技巧

朱国江　孙建隆编著

责任编辑　黄丽荣

*　　*　　*

气象出版社出版

（北京西郊白石桥路 46 号）

国防科工委印刷厂印刷

新华书店总店科技发行所发行　全国各地新华书店经销

*　　*　　*

开本：787×1092　1/32　印张：14.75　字数：327 千字

1991 年 12 月第一版　1991 年 12 月第一次印刷

印数：1—5500　　定价：7.50 元

ISBN 7－5029－0673－8/TP・0027

前　言

计算机图形学以其强大的生命力，正在许多领域得到日益广泛的普及和应用。特别是对于变化万千、神奇莫测的动画和计算机游戏，更能吸引广大青少年，除了可以娱乐、激发灵感、增强思维、丰富想像外，一个更重要的原因是由于自身参加而得到精神上和艺术上的享受，但这并不能使他们感到满足，富有想像力和创造力的广大青少年朋友，更希望通过学习、思考、模仿和创造，亲自动手设计出更多更好的图形变换和游戏程序来。为了帮助大家了解和学习图形处理的技术，我们编写了这本小册子，试图引导大家提高对图形变换和计算机游戏的感性认识和兴趣，了解和学习计算机图形学的基本语言和有关算法，生成各种画面的技巧和方法，以及掌握设计图形和游戏的原理、过程和方法等。目的是抛砖引玉，以便把这方面的教育工作推向更高层次。

为使本书适用于更多读者，本书编写中力求照顾不同对象。对于只想玩游戏的读者来说，只要阅读本书第十一章内容，并把第十一章和附录的程序清单和图形资料送入计算机即可，因为这对于不熟悉程序设计和游戏原理制作的读者来说是适合的，把软件作为一个游戏程序来看，只要对照使用说明就可以了。对于已经掌握中华学习机或 APPLE－Ⅱ微机系统的基础知识，并初步掌握了 BASIC 语言和 6502 汇编语言的读者来说，本书是学习和借鉴的有用参考。而对于广大青少年计算机教师、课外辅导员以及从事计算机图形学研究和有志于游戏软件开发的设计人员来说，本书提供了一个实

用工具软件，希望给予扩展、充实和提高。

本书资料主要来自作者多年编程和教学实践，同时吸收了国内外的先进经验，也选用了一些书籍和杂志上发表的优秀文章，并经作者组织、加工、提炼和改造。全书着重介绍中华学习机在图形处理方面的各种技巧，种类繁多，应有尽有。不仅包括常规的图形处理技术，也包括图象处理的特殊技巧，既有特色，又有创新。巧妙地利用图象压缩存贮技术、16K RAM 卡，解决了处理多幅图形中华学习机内存不足的困难。运用多页联动、分页显示、图形剪辑、图象合成等技术，为图形处理提供了新路子。灵活修改指针，采用反复搬家的方法，解决了程序区和图形存贮、显示页面不相兼容的矛盾。选用 STC 汉字系统，用机器语言控制，解决了工具软件包可以同时在 APPLE－Ⅱ、紫金－Ⅱ系列机、中华学习机上通用和汉化问题。而书中提供的各种状态下的清屏技巧，使图形画面更加丰富多采。各种图形处理技巧，均可以直接引用或移植，从而为您的图形软件增辉添色。

由于编者水平有限，缺点、错误在所难免，敬请读者批评指正。

编者　1989 年 11 月

目　　录

一、中华学习机显示页结构

本章详细介绍了中华学习机文本状态、低分辨率图形状态和高分辨率图形状态的显示页结构。它们是各种图形技巧的基础。弄清楚各状态的显示页结构，就可以设计出各种图形处理的程序来。

1. 文　本

打开中华学习机电源，屏幕上立即显示出“ZHONG HUA XUE XI JI”等字样，如图1.0所示。这就是中华学习机的西文文本状态。当你从键盘上敲入程序 P1 时可以看到，敲入的字符和数字在屏幕上一一显示出来，同时有一个

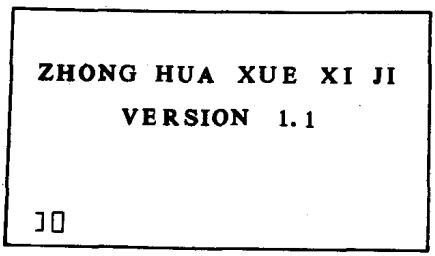

图 1.0　中华学习机西文文本状态

称为游标的闪烁的方块指示着下一个显示位置，游标依次向右移，满一行则自动向下行最左端移动，如果敲入回车（RE-

TURN）键，则跳过一行指在最左端。

P1：

```
10  REM   ***  POKE THE HORIZONTAL
      POSITION  ***
20  HOME
30  FOR I = 0 TO 255
40  POKE 1024 + I,I
50  NEXT I
60  END
```

从屏幕上看，字符位置是依次排列的。那么放置字符代码的地址是否也依次排列呢？运行一下 P1 程序，仔细观察屏幕上的变化，可以发现屏幕水平 24 行不是按地址顺序依次排列的，而是分成第 1 行至第 8 行、第 9 行至第 16 行、第 17 行至第 24 行三部分。第一部分是依次排列。这就是说第 1 行显示完后接着显示第 9 行，再接着显示第 17 行，然后回到第 2 行，第 10 行，第 18 行……。这种方式叫折叠显示数据在存贮器中的存储方式。中华学习机显示方式的基址分配都是采用这种方法，见图 1.1。

文本显示区的存贮地址是 $400—$7FF（第 1 页）和 $800—$BFF（第 2 页，通常不用）。仔细地计算一下，可以发现文本区是 24×40＝960 个地址，而 $400—$7FF 共有 1024 个地址，多出了 64 个地址，这 64 个地址显示区没有用，它分配给了 I/O 接口插座用。

2. 低分辨率图形

同文本显示页结构一样。$400—$7FF 这段地址区除了作文本显示缓冲区外，它也作为低分辨率图形的显示缓冲区地址。不同的是，低分辨率图形方式显示 48×40 个色块阵

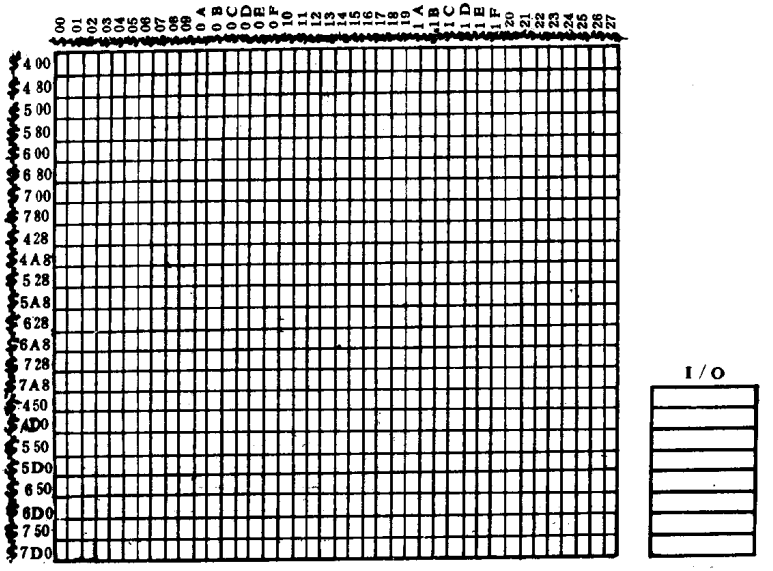

图 1.1　文本显示页面结构图

列，每个色块占一个字符宽度和半个字符高度的位置。因此，每个字节表示两个色块，一个在上，一个在下。结构图见图 1.2。

在 BASIC 状态下，键入程序 P2，运行后屏幕上出现由低分辨率色块组成的汉字“欢迎”二字。画面可以有 16 种颜色。低分辨率色块的颜色是由其对应地址的内容决定的，每个字节中，低 4 位设置上面色块的颜色，高 4 位设置下面色块的颜色，4 位二进制数值可示 16 种不同的颜色，见表 1。

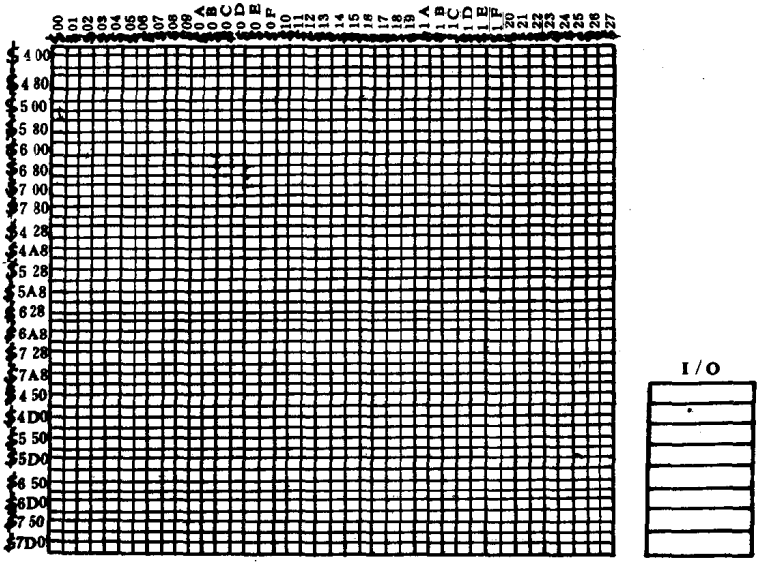

图 1.2　低分辨率图形页面结构图

表 1　低分辨率图形彩色码

十进制	十六进制	颜色
0	$0	黑
1	$1	深红
2	$2	深蓝
3	$3	紫
4	$4	灰 1
5	$5	蓝
6	$6	浅蓝
7	$7	深橙
8	$8	棕
9	$9	橙
10	$A	灰 2
11	$B	深红
12	$C	绿
13	$D	浅橙
14	$E	浅绿
15	$F	白

P2：

```
10  GR : COLOR= 15
20  FOR I = 1 TO 21
30  READ A,B,C
40  VLIN A,B AT C
50  NEXT I
60  FOR J = 1 TO 17
70  READ A,B,C
80  HLIN A,B AT C
90  NEXT J
100  FOR K = 1 TO 13
110  READ A,B
120  PLOT A,B
130  NEXT K
140  DATA 11,15,8,11,15,10,10,13,1
     2,11,13,16,13,15,14,12,16,24,
     12,15,26,11,17,28,12,15,30,22
     ,27,2,22,27,4,22,27,6,22,27,8
     ,22,27,13,22,27,18
150  DATA 22,27,23,22,27,26,22,27,
     28,22,27,30,22,27,32,22,27,34
     ,13,16,11,26,30,10,26,30,12
160  DATA 23,31,17,3,5,27,9,11,27,
     14,16,27,19,21,27,24,26,27,34
     ,37,27,34,37,24,34,37,22
170  DATA 29,31,22,24,25,22,19,21,
     22,9,11,22,9,11,24,9,11,8,17,
     9,16,10,17,12,17,13,16,15,16,
     16,17,24,10,23,12,16,26,21,26
     ,21,23
```

中断程序 P2 的运行，返回文本状态，可以看到画面上出现许多反白ⓐ字形，这是怎么一回事呢？查看一下反白ⓐ的 ASCII 代码，发现，原来低分辨率时的空白（黑色，$00）与文本显示反白ⓐ字形编码相同。也就是说，同一数据存入同样的显示缓冲区地址，在不同的显示状态下所见到的屏幕显示是不同的。

3. 高分辨率图形

高分辨率图形显示页结构也与文本区类似，见图 1.3。但是文本区状态每个字形占用一个地址单元字节，而高分辨率状态每个字形却要占用 8 个地址单元字节。也就是说，图 1.3 中每一个方格包含了 8 条水平扫描线，而每条水平扫描线由 7 个点组成，所以整个屏幕就有（8×24）×（7×40）=192×280 个点。高分辨率图形显示是以屏幕上某点对应于缓冲

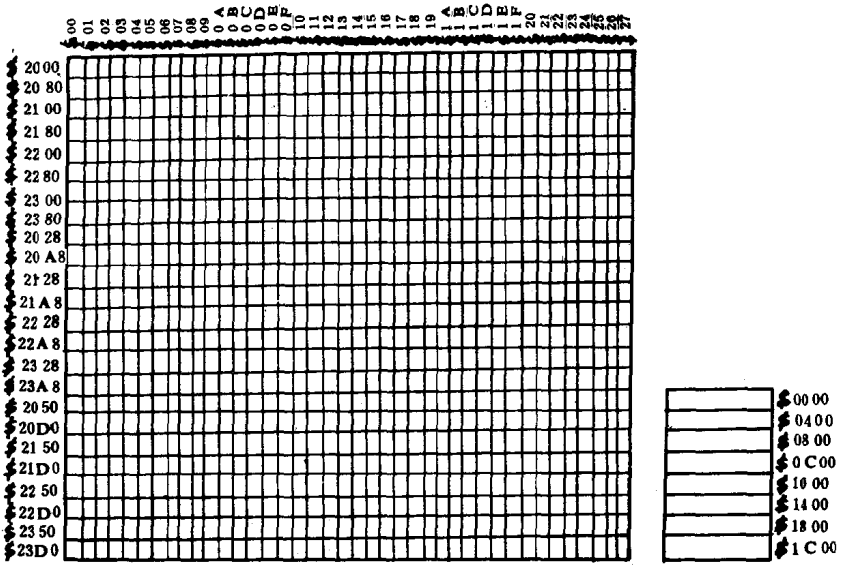

图 1.3　高分辨率图形显示页结构图

地址单元字节中的某一位来作图的，见图 1.4。地址单元中的数据有 8 位，分别对应每条线的一个点后还多一位，这一位

就用来作颜色控制用。数据位的最低位决定相邻 7 点中最左边一点的显示，而最高位选择颜色。高分辨率图形的颜色结构可用图 1.5 表示。

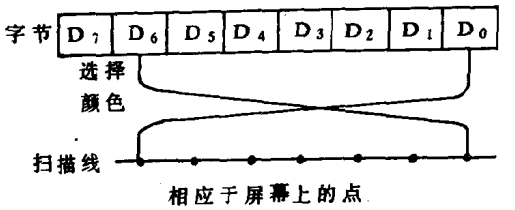

图 1.4　数据位与显示点的对应关系

最高位	6～0位							颜色	十六进制值
0	0	0	0	0	0	0	0	黑	$00
1	0	0	0	0	0	0	0	黑	$80
0	1	1	1	1	1	1	1	白	$7F
1	1	1	1	1	1	1	1	白	$FF
0	1	0	1	0	1	0	1	紫色	$55
1	1	0	1	0	1	0	1	蓝色	$D5
0	0	1	0	1	0	1	0	绿色	$2A
1	0	1	0	1	0	1	0	橙色	$AA.

图 1.5　高分辨率图形颜色结构图

从图 1.5 可知，如果连续两值为 1，则为白色；如果是非连续的奇数位置的点，当最高位为 1 时点为橙色，为 0 时点为绿色；如果是偶数位置的点（包含 0），当最高位为 1 时点为蓝色，为 0 时点为紫红色。例如十六进制数 $D3 (211)，为 11010011，颜色为蓝和白色。运行程序 P3，可以看到屏幕左上角是白蓝方块。

高分辨率图形显示缓冲区地址也有 2 页，第一页为 $2000—$3FFF，第二页为 $4000—$5FFF。图 1.3 中每一小方格包含 8 条扫描线，表格最左边的数字是代表每一方格内最上面一条扫描线的地址，如果要求出其它扫描线所对应的地址则可用表格最右边的 8 个数字来相加即可。例如，要求出第三行方格最下边一条线的起始地址，则为 $2100＋$1C00＝$3D00。所以当你键入 HGR：POKE 15616，255 便可在第 3 行方格最低线（第 24 条扫描线）的左端看到一段白线（7 个连续的白点）。

P3：

```
10  HGR
20  FOR I = 0 TO 7
30  POKE 8192 + I * 1024,211
40  NEXT I
50  END
```

4. 屏幕位置与缓冲区地址对应关系计算

通过上面介绍，我们知道屏幕上某点的位置与缓冲区地址顺序不是一一对应的。那么如何来计算它们的对应关系呢？我们用 A 来表示缓冲区地址单元，用直角坐标（X，Y）表示屏幕上某点的位置，将坐标原点（0，0）表示屏幕左上角的位置，则从图 1.3 可知原点的地址为 8192。由于水平方向每个点的位置是连续的，因此水平方向的地址只要用 8192 加上 X 坐标即可。至于垂直方向的关系就很复杂了。我们已经知道屏幕分成三部分：第 1—8 行；第 9—16 行；第 17—24 行；而高分辨每一行又包含 8 条扫描线。因此显示的顺序是

这样的：显示第一条扫描线后接下来显示第二部分第一条扫描线，再接着显示第三部分第一条扫描线，然后回到第一部分显示第九条扫描线……依次下去，直至显完。请键入程序P4，运行一下看看，是不是这样。

P4：

```
10  HGR : POKE  - 16302,0
20  FOR I = 0 TO 8191
30  POKE 8192 + I,255
40  NEXT I
50  END
```

按照图 1.3，每一部分内每行之间的地址相差 128，而每行内每条扫描线的地址相差 1024；第二部分各行又比第一部分各行多 40；第三部分各行又比第二部分各行多 40。仔细分析一下，找出其中的规律，我们发现可以用取模的方法来列出屏幕位置与缓冲区地址的关系式：

$$\begin{aligned} A = {} & 8192 + \mathrm{INT}(Y/64) * 40 \\ & + (\mathrm{INT}(Y/8) - \mathrm{INT}(Y/64) \\ & * 8) * 128 + (Y - \mathrm{INT}(Y/8) \\ & \times 8) * 1024 + X \end{aligned} \qquad (1.1)$$

因此，只要知道屏幕位置的（X，Y）坐标值，即可用式（1.1）求出其对应的地址单元。应当指出，在高分辨率作图时，用式（1.1）计算地址速度很慢，常改用机器语言编程来计算基地址。为了便于说明原理，我们将式（1.1）改写成 INTEGER BASIC 形式：

$$\begin{aligned} A = {} & 8192 + (Y/64) * 40 \\ & + [(Y/8)\mathrm{MOD}8)] * 128 \\ & + (Y \ \mathrm{MOD} \ 8) * 1024 \\ & + X \end{aligned} \qquad (1.2)$$