532363

华胜3000工程工作站
系列资料

CCPixrect
程序员手册

机械电子工业部
第六研究所编

上海交通大学出版社

华胜3000工程工作站系列资料

CCPixrect程序员手册

中国机械电子工业部
第六研究所
朱湧 编译

上海交通大学出版社

内容简介

本书介绍CCPixrect软件包的功能与使用操作方法。内容包括：CCPxrect的基本概念，基本像矩操作，文本功能，存储器像矩，文件的I/O功能，驱动程序的编写，函数和宏指令，数据结构以及曲线绘制功能等。

读者对象，华胜3000工作站，SUN 3/160C及SUN 3系列用户，计算机工作者。

CCPixrect程序员手册
上海交通大学出版社出版
(淮海中路1984弄19号)
新华书店上海发行所发行印装

开本787×1092毫米 1/32 印张2,125 字数105,000
1988年11月第1版 1988年11月第1次印刷
印数：1–2000
ISBN7-313-00398-6/793

工本费4.00元

IS

目　　录

1　绪言　1
1.1　综述　1
1.2　重要的概念　1
1.3　实例　1
1.4　CCPixrect的Lisp函数库　2
1.5　参考资料　2

2　基本像矩操作　4
2.1　pixrectops结构　4
2.2　像矩操作参数的命名惯例　5
2.3　CCPixrect的错误　5
2.4　像矩的生成与破坏　5
2.4.1　生成基本显示像矩　5
2.4.2　生成辅助像矩　6
2.4.3　释放像矩资源　6
2.5　单像素操作　6
2.5.1　读取像素值　7
2.5.2　设置像素值　7
2.6　构造op参数　7
2.6.1　指定光栅操作　8
2.6.2　指定不变的源像素值　9
2.6.3　光栅操作中的裁剪控制　9
2.6.4　op参数构造实例　9
2.7　多像素操作　9
2.7.1　从源到目的光栅操作　9
2.7.2　经过屏蔽的光栅操作　10
2.7.3　复制源像矩　11
2.7.4　从多源到单目的光栅操作　11
2.7.5　描绘矢量　12
2.7.6　描绘结构化多边形　12
2.8　访问颜色表　14
2.8.1　读取颜色表项　14
2.8.2　设置颜色表项　14
2.8.3　像矩的视频反转　15
2.9　位帧控制属性　15
2.9.1　读取位帧屏蔽值　15
2.9.2　设置位帧屏蔽值　16

3.10 效果验证 16

3 CCPixrect的文本功能 1
3.1 p_xchar和pixrect的结构 17
3.2 打开和关闭字库 18
3.3 汉字库的选择 18
3.4 字符输出函数 19
3.5 辅助函数 20
3.6 实例 21

4 存储器像矩 22
4.1 mpr_data结构 22
4.2 生成存储器像矩 22
4.2.1 生成存储器像矩 22
4.2.2 根据映像生成存储器像矩 23
4.2.3 实例 23
4.3 静态存储器像矩 26
4.4 存储器像矩中的像素层次 28
4.5 使用存储器像矩 34

5 CCPixrect的文件I/O功能 35
5.1 读写光栅文件 35
5.1.1 写光栅文件 35
5.1.2 读光栅文件 37
5.2 光栅文件格式的细节 28
5.3 写光栅文件的部件 29
5.3.1 写光栅文件首部 29
5.3.2 初始化光栅文件首部 29
5.3.3 向光栅文件写映像数据 29
5.4 读光栅文件的部件 30
5.4.1 读光栅文件的首部 30
5.4.2 读光栅文件的颜色表 30
5.4.3 读光栅文件的映像 30
5.4.4 读标准光栅文件 31

附录A 编写Pixrect驱动程序 32
A.1 你需要的资料和工具 32
A.2 实现步骤 32
A.3 生成的文件 33
A.3.1 存储器映像的设备 33

此为试读,需要完整PDF请访问: www.ertongbook.com

A.4 像矩的专用数据 33
A.5 生成和破坏 34
A.5.1 生成基本像矩 34
A.5.2 生成辅助像矩 36
A.5.3 破坏像矩 37
A.5.4 pr_makeun操作向量 38
A.6 像矩核心设备驱动程序 39
A.6.1 支持的可配置设备 39
A.6.3 打开 45
A.6.3 存储器映像 46
A.6.4 I/O控制 46
A.6.5 关闭 48
A.6.6 把你的驱动程序插入CMIX 49
A.7 访问实用程序 49
A.8 rop 50
A.9 batchrop 50
A.10 矢量 50
A.11 颜色表 50
A.12 属性 50
A.13 像素 51
A.14 模板 51
A.15 曲线 51
A.16 多边形 51

附录B CCPixrect的函数和宏指令表 52

附录C CCPixrect的数据结构 57

附录D 曲线绘制功能 59

1 绪言

本手册描述CCPixrect图形支撑软件包，该软件包是一个低级的光栅操作子程序库，用户可以用它来编写与设备无关的应用程序。

CCPixrect是一个位于硬件设备驱动程序之上的低层的软件包。对于大多数应用程序来说，使用其他较高级的图形支撑软件包更为有利。

CCPixrect只能用来按照与设备无关的方式访问和处理显示设备的矩形区域。CCPixrect不支持重叠的窗口，应用程序可以利用存储器像矩来实现这种功能。CCPixrect不包含任何输入功能，应用程序可以使用窗口系统的输入功能来接收输入数据。

虽然本手册中给出了一些简单的程序实例，但它并不是一本讲述如何运用CCPixrect编写应用程序的教材，它只是向读者提供一些使用该软件包所必需的参考资料。读者应熟悉位映像图形原理以及C程序设计语言。

1.1 综 述

本手册分为若干章节，它们分别描述CCPixrect软件包的主要特性。第2 章介绍生成和处理像矩的一些基本操作。第3 章描述CCPixrect软件包的文本处理功能。第4 章讨论存储器像矩，它是与显示像矩有类似特性的虚存区域。第5 章解释CCPixrect软件包的I/O功能，用户可以利用这些功能在像矩和磁盘文件之间传递数据。附录A是像矩驱动程序的实现指南。附录B列出了CCPixrect软件包的所有函数和宏指令。附录C 列出了CCPixrect软件包所使用的所有类型和结构定义。附录D描述CCPixrect软件包的曲线绘制功能。

1.2 重要的概念

本节描述有关CCPixrect软件包的一些重要概念。通过这些概念来解释CCPixrect软件包区别于其他图形支撑软件包的特点。

屏幕坐标：屏幕坐标系统是一个原点在左上角的二维整数坐标系统，其X和Y坐标分别向右和向下增大。

像素：像素是指具有屏幕坐标或相对于屏幕上某个矩形子域的地址的单一图像元素。

位映像：位映像是指屏幕空间中的一个矩形区域。例如整个屏幕或一个窗口。

光栅操作：光栅操作是一种作用于两个或三个位映像的操作。它利用目像素的原有值、对应的源像素值以及可能的屏蔽像素的值，按照布尔运算的规律计算每个目像素的值。

像矩：像矩是由位映像的数据和一组可作用于这些数据的操作组合而成的数据结构。像矩可以存在于包括存储器和打印机在内的多种设备上。

1.3 实例

下列程序在显示界面上画一条直线

```
#include <pixrect/pixrect_hs.h>

main()
{
        struct  pixrect  *screen;

        screen=pr_open("/dev/fb");
        pr_vector(screen,10,20,70,80,PIX_SET,1);
        pr_close(screen);
}
```

图1-1 简单的程序实例

这个程序可以用下列命令进行编译:

```
cc  line.c  -o  line  -lccpixrect
```

编译得到的可执行文件是line。这个程序将在屏幕左上角画一条直线。

1.4 CCPixrect的lint函数库

CCPixrect提供了一个lint函数库,可以对超出C编译程序能力的类型检查提供帮助。例如,你可以使用CCPixrect的lint函数库通过下列命令检查程序glass.c:

```
lint  glass.c  -lccpixrect
```

注意:lint产生的大部分错误信息都是报警信息,可能不会影响程序的正常操作。

1.5 参 考 资 料

[1] J.D.Foley and A.van Dam. Fundamentals of Interactive Computer Graphics. Addison-Wesley,1982.

[2] Smalltalk Graphics Kernel. D.Ingalls. Byte,August 1981.

[3] B.W.Kernighan and D.M.Ritchie The C Programming Language. Prentice-Hall,1978

[4] W.M.Newman and R.F.Sproull. Principles of Interactive Computer Graphics. McGraw Hill,1979.

[5] R.Pike,Leo Guibas,Dan Ingalls. Bitmap Graphics. ACM/SIGGRAPH 1984 Conference Course Notes.
[6] V.R.Pratt. Standards and Performance Issues in the Workstation Market. IEEE Computer Graphics and Applications, April 1984.
[7] SunCore Reference Manual.
[8] SunCGI Reference Manual.
[9] SunView Programmer's Guide.
[10] SunView System Programmer's Guide.

2 基本像矩操作

CCPixrect软件包提供的基本操作包括：

- 生成和破坏像矩(open,region和destroy)
- 读写单个像素的值(get和put)
- 处理多个像素的值：

pr_rop	从源像矩写到目像矩，
pr_stencil	在一个屏蔽的控制下从源像矩写到目像矩，
pr_replrop	把不变的源像矩模式复制到目像矩，
pr_batchrop	把一批源像矩写到单个目像矩中的不同位置，
pr_vector	在像矩中画一条直线。

- 读写颜色表(getcolormap,putcolormap)
- 选择彩色像矩中的特定位帧(getattributes,putattributes)

其中有些操作与像矩的类型无关，他们是由单个程序实现的。这些与设备无关的操作可以由CCPixrect的用户直接调用。另外一些操作是与像矩类型有关的。每个像矩中都有一个指向pixrectops结构的指针，这个结构中包含所有与设备有关的操作程序的入口地址。这就允许用户通过指针而不是直接使用程序名来访问这些程序。为了简化这种访问，CCPixrect软件包提供了一组宏指令，使用户可以像访问其他程序一样，以函数调用的形式访问这些与设备有关的程序。

2.1 pixrectops结构

```
struct  pixrectops  {
        int  (*pro_rop)();
        int  (*pro_stencil)();
        int  (*pro_batchrop)();
        int  (*pro_nop)();
        int  (*pro_destroy)();
        int  (*pro_get)();
        int  (*pro_put)();
        int  (*pro_vector)();
        struct  pixrect  *(*pro_region)();
        int  (*pro_putcolormap)();
        int  (*pro_getcolormap)();
        int  (*pro_putattributes)();
        int  (*pro_getattributes)();
};
```

pixrectops结构是一个指向与设备有关的程序入口的指针集合。所有其他操作是用与设备无

关的程序实现的。

2.2 像矩操作参数的命名惯例

表2-1中列出了在命名光栅操作参数时惯用的符号。

表2-1 参数命名惯例

参 数	定 义
d	目
s	源
x 和 y	左上角的坐标
w 和 h	宽度和高度

像矩操作函数所用的x和y值限定在像矩的边界之内。

2.3 CCPixrect的错误

那些应该(正常情况下)返回一个结构指针的像矩操作程序如果失败,则将返回一个空指针NULL。除此之外,返回值PIX_ERR(-1)表示失败,零值表示成功。个别函数的返回值与此不同,将在相应的章节中予以说明。

2.4 像矩的生成与破坏

用于生成像矩的程序有pr_open和mem_create,宏指令有pr_region和mpr_static。用于破坏像矩的是宏指令pr_destroy。mem_create和mpr_static将在第四章中讨论;下面讨论其他几个程序和宏指令。

2.4.1 生成基本显示像矩

```
struct  pixrect  *pr_open(devicename)
char  *devicename;
```

非存储器像矩的特性取决于对应的UNIX设备。在为某个设备生成第一个像矩时,需要调用程序pr_open来打开这个设备。隐含的显示设备名是"/dev/bwone0"、"/dev/cgone0"或"/dev/cgtwo0"。

此为试读,需要完整PDF请访问: www.ertongbook.com

pr_open不能用于生成存储器像矩。pr_open将返回一个指向pixrect结构的指针，与之对应的基本显示像矩将覆盖整个设备表面。如果该程序失败，他将返回NULL，并在标准错误输出设备上打印错误信息。

2.3.2 生成辅助像矩

```
#define  struct  pixrect  *pr_region(pr,x,y,w,h)
struct  pixrect  *pr;
int  x,y,w,r;

#define  struct  pixrect  *prs_region(subreg)
struct  pr_subregion  subreg;
```

给出一个已存在的像矩，可以生成一个包含他的部分或全部像素的辅助像矩。这种辅助像矩是由宏指令pr_region和prs_region所调用的程序生成的。

pr是指向已存在的像矩结构的指针，这个像矩可以是基本像矩也可以是辅助像矩。x和y表示新像矩的原点在原有像矩中的坐标；w和h分别表示新像矩的宽度和高度。prs_region的用途与pr_region相同，只不过他把所有的参数都收集在单一的subreg结构中。如果他们调用成功，将返回一个指向新像矩的指针；如果失败，则将返回NULL，并在标准错误输出设备上打印出错误信息。

如果在调用时指定了一个已存在的辅助像矩，则生成的将是对应的基本像矩的另一个辅助像矩，在这两个辅助像矩之间不存在进一步的关系。一般来说，基本像矩和辅助像矩之间的区别并不重要，但在基本像矩被破坏以后，就不能再使用对应的辅助像矩。

2.3.3 释放像矩资源

```
#define  pr_close(pr)
struct  pixrect  *pr;

#define  pr_destroy(pr)
struct  pixrect  *pr;

#define  prs_destroy(pr)
struct  pixrect  *pr;
```

宏指令pr_close、pr_destroy和prs_destroy将调用与设备有关的程序来破坏像矩并释放他所占用的资源。这些程序如果成功则返回0，如果失败则将返回PIX_ERR。这些宏指令可以用于基本像矩也可以用于辅助像矩。这三个宏指令是相同的，同时定义这些宏指令是由于历史的原因以及为了保持风格上的一致。

2.5 单像素操作

本节描述用于处理单个像素的两个操作。

7.5.1 读取像素值

```
#define  pr_get(pr,x,y)
struct  pixrect  *pr;
int  x, y;

#define  prs_get(srcprpos)
struct  pr_prpos  srcprpos;
```

宏指令pr_get和prs_get调用与设备有关的程序读取单个像素的值。pr指出像素所在的像矩;x和y是像素的坐标。对于prs_get来说,全部参数都在单一的srcprpos结构中提供。像素的值将作为一个32位整数返回;如果失败,则返回PIX_ERR。

7.5.2 设置像素值

```
#define  pr_put(pr,x,y,value)
struct  pixrect  *pr;
int  x,y,value;

#define  prs_put(dstprpos,value)
struct  pr_prpos  dstprpos;
int  value;
```

宏指令pr_put和prs_put调用与设备有关的程序设置单个像素的值。pr指出要设置的像素所在的像矩;x和y是像素的坐标。对于prs_put来说,全部参数都在单一的dstprpos结构中提供。value的值经适当地截取后写入指定的像素。如果失败,将返回PIX_ERR。

7.6 构造op参数

下一节中将要介绍的多像素操作都使用相同的方法指定产生目像素的操作方式。这种操作在op参数中给出,他包括三个部分:

- op参数的第5—31位是一个颜色值,他可作为单个源像素值使用。
- op参数的第1—4位用来指定光栅操作的类型。
- op参数的第0位是一个裁剪标志,他可以控制光栅操作中是否进行裁剪。

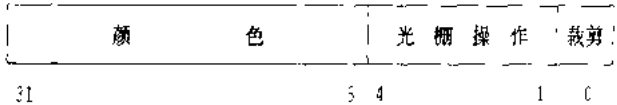

图2-2 op参数的结构

2.6.1 指定光栅操作

op参数中的第5—4位用来指定16种不同的逻辑功能之一，这些功能把单色源像素和单色目像素组合成为单色的结果。其中有些功能比其他功能更为常用，表2-2中列了最有用的几种功能。

下列定义为表示光栅操作功能提供了方便易懂的形式：

```
#define  PIX_SRC  0x16
#define  PIX_DST  0x14
#define  PIX_NOT(op)  (0x1E & ~(op))
```

其中PIX_SRC和PIX_DST被定义为常数，PIX_NOT被定义为宏指令。利用这些定义可以组合成各种期望的逻辑功能。注意，在所有光栅操作中，必须使用PIX_NOT而不能使用求反(~)运算符。下面是两个特殊的例子：

```
#define  PIX_SET  (PIX_SRC | PIX_NOT(PIX_SRC))
#define  PIX_CLR  (PIX_SRC & PIX_NOT(PIX_SRC))
```

其中PIX_SET总是设置结果，而PIX_CLR总是清除结果。

表2-2 有用的光栅操作组合

op 中 的 操 作	结 果
PIX_SRC	写(d=s)
PIX_DST	空操作(d=d)
PIX_SRC | PIX_DST	涂抹(d=s | d)
PIX_SRC & PIX_DST	屏蔽(d=s & d)
PIX_NOT(PIX_SRC) & PIX_DST	抹除(d=~s & d)
PIX_NOT(PIX_DST)	反转(d=~d)
PIX_SRC ^ PIX_DST	反转涂抹(d=s^d)

2.6.2 指定不变的源像素值

在某些情况下，用户可能希望指定一个无限的具有相同值的像素来源。要做到这一点，可以用NULL作为源像矩指针，并把op参数的第5～31位作为一个颜色值来处理。下面的两个宏指令可用于这种情况：

```
#define PIX_COLOR(color) (color<<5)
#define PIX_GETCOLOR(op) ((op)>>5)
```

如果op中没有指定颜色值，则隐含为0。当源像矩指针为NULL时，就使用在op中指定的颜色值。

注意，颜色值不是功能代码的一部分；他不能作为PIX_NOT的参数使用。

当源像矩是一个单色像矩，目像矩是一个彩色像矩时，op参数的颜色部分用来确定目像素的颜色。在这种情况下，如果源像素的值为0，则对应目像素的值将是背景色；如果源像素的值为1，则对应目像素的颜色将是op中的颜色部分所指定的颜色。

如果op的颜色部分为0，则其颜色将隐含为前景色(-1)。

2.6.3 光栅操作中的裁剪控制

像矩操作通常限定在被操作像矩的边界之内进行。有些时候，像矩操作的裁剪可以由用户在较高的层次更有效地进行。如果用户能够确保他所执行的像矩操作不会超出像矩的边界，就可以指示像矩操作不进行裁剪检查，从而加速程序的运行。op参数的第0位是一个裁剪标志位，当他为1时，表示不进行裁剪。

```
#define  PIX_DONTCLIP  0x1
```

如果设置了这个标志，当像矩操作越界时，其结果是不确定的，并可能引起存储器故障。

注意，PIX_DONTCLIP标志不是op参数中功能代码的一部分；他不能作为PIX_NOT的参数来使用。

2.6.4 op参数构造实例

第一个例子非常简单，他指出要把源像素写到目像素，并进行裁剪：

```
op=PIX_SRC;
```

第二个例子比较复杂，他的定义在语法上讲是正确的，但他究竟有多少用处，尚不清楚：

```
op=(PIX_SRC ^ PIX_NOT(PIX_SRC | PIX_DST))
     | PIX_COLOR(red) | PIX_DONTCLIP
```

2.7 多像素操作

本节将介绍同时处理多个像素的操作。这些操作都使用op参数来指定目像素的生成方式。在附录D中将讨论另一个多像素操作pr_traprop。

2.7.1 从源到目的光栅操作

```
#define  pr_rop(dpr,dx,dy,dw,dh,op,spr,sx,sy)
struct  pixrect  *dpr,*spr;
int  dx,dy,dw,dh,op,sx,sy;

#define  prs_rop(dstregion,op,srcprpos)
struct  pr_subregion  dstregion;
int  op;
struct  pr_prpos  srcprpos;
```

宏指令pr_rop和prs_rop调用与设备有关的程序从源像矩向目像矩执行指定的光栅操作。dpr是目像矩的指针;(dx,dy)是将被影响的矩形区域的原点(左上角);dw和dh分别表示这个矩形区域的宽度和高度。spr是源像矩的指针;(sx,sy)是源像矩内的一点,op指定所要执行的光栅操作。

对于prs_rop来说,dpr、dx、dy、dw和dh都收集在单一的pr_subregion结构中。

光栅操作的裁剪按源像矩中的范围进行,如果这个范围小于指定的目区域的话。如果失败,他们将返回PIX_ERR,如果成功则返回0。

源像矩和目像矩通常必须具有相同的深度。但如果源像矩是单色的(深度为1),则目像矩可以是任意深度的像矩。在这种情况下,如果源像素的值为0,则对应的目像素也写为0;如果源像素的值为1,则对应的目像素被写为op中的颜色值。如果op中的颜色值为0并且源像素的值为1,则对应的目像素中将被写入他所能存放的最大值。

pr_rop的用法可参考图4-1中的程序。

2.7.2 经过屏蔽的光栅操作

```
#define  pr_stencil(dpr,dx,dy,dw,dh,op,stpr,stx,sty,spr,sx,sy)
struct  pixrect  *dpr,*stpr,*spr ;
int  dx,dy,dw,dh,op,stx,sty,sx,sy;

#define prs_stencil(dstregion,op,stencilprpos,srcprpos)
struct pr_subregion dstregion;
int op;
struct pr_prpos stencilprpos,srcprpos;
```

宏指令pr_stencil和prs_stencil调用与设备有关的程序从源像矩向目像矩中由模板像矩限定的区域内执行指定的光栅操作。除了带有一个起屏蔽作用的模板像矩之外,pr_stencil与pr_rop是相同的。模板像矩必须是一个单色存储器像矩。只有那些与模板像矩中的非零像素对应的目像素才有可能被修改,其他目像素保持不变。源像矩中从(sx,sy)开始的矩形与模板像矩中从(stx,sty)开始的矩形对齐,写到目像矩中的从(dx,dy)开始、宽为dw高为dh的矩形区域之中。源像矩指针spr可以是NULL,这时将使用op参数中的颜色值。裁剪操作将限定在源、目和模板矩形相交的部分之内。如果失败,将返回一个PIX_ERR,如果成功则返回0。

2.7.3 复制源像矩

```
pr_replrop(dpr,dx,dy,dw,dh,op,spr,sx,sy
struct pixrect *dpr,*spr;
int dx,dy,dw,dh,op,sx,sy;

#define prs_replrop(dsubreg,op,sprpos) \
struct pr_subregion dsubreg;
int op;
struct pr_prpos sprpos;
```

光栅操作中的源像矩常常是一种用来覆盖某个区域的模式。如果要把单一的值写入目区域中的所有像素，则最好的办法是在pr_rop操作的op参数中指定这个值。但当这种模式大于一个像素时，就需要某种机构来指定基本模式以及这种模式在目区域中的排列方式。

pr_replrop程序把源像矩中的指定模式重复地拷贝到整个目区域。目区域的位置和大小由dx、dy、dw和dh指定。spr是源像矩的指针，(sx,sy)是源模式的起点。对应的宏指令prs_replrop 将把dsubreg扩充成5个目参数，把sprpos扩充成3个源参数调用pr_replrop。op用来指定所要执行的操作。

pr_replrop如果失败，将返回PIX_ERR，如果成功则返回0。

2.7.4 从多源到单目的光栅操作

```
#define pr_batchrop(dpr,dx,dy,op,items,n)
struct pixrect *dpr;
int dx,dy,op,n;
struct pr_prpos items[];

#define prs_batchrop(dstpos,op,items,n)
struct pr_prpos dstpos;
int op,n;
struct pr_prpos items[];
```

宏指令pr_batchrop和prs_batchrop调用与设备有关的程序从一系列源像矩向同一目像矩中的连续位置执行指定的光栅操作。items是一个pr_prpos结构数组，pr_batchrop程序把他作为一个源像矩序列使用。数组中的每个元素指定一个源像矩以及一对在x和y方向上的增量。这个增量用来修改目的位置，而不是作为源像矩中的源点坐标使用。

dpr是目像矩的指针，(dx,dy)是目像矩内的一点，op指定所要执行的操作，items是一个结构数组，其长度由参数n给出。对于prs_batchrop来说，参数dstpos指定了目的位置。

该操作的目位置首先初始化为由(dx,dy)所给出的位置。然后，对于结构数组中的每一项来说，当前目位置是以前的目位置加上该项中的增量所得到的位置。这种调节是累加的。

pr_batchrop的最常见的应用是描绘文本。支持这种应用的附加功能将在第三章中描述。注意，pr_batchrop的定义支持可变间距的和旋转的字体，以及非罗马字书写系统和简单的

此为试读，需要完整PDF请访问：www.ertongbook.com