面向对象程序设计课程大作业

BitMagic 程序文档

Bit Magic image Processor...
More than BMP...

指导老师：陈奇

开发者：郑博阳

吴粤

王益

李骁

2008

浙 江 大 学

BitMagic 程序文档

姓名	学号	专业
郑博阳	3060601370	应用心理学
吴粤	3060601314	数学与应用数学
王益	3071102635	计算机科学与技术
李骁	3080030116	竺可桢学院工科平台

在生活中，您可能会听到这样的声音……

是的，Photoshop 作为一个强大而又简易的图像设计/处理工具，在该类软件中的地位让其他难以望其项背。然而强大的软件总伴随着在有些程度上的笨拙，而当前的 BitMagic 程序则试图从灵巧性出发，基于图层的概念简易实现 Photoshop 的功能。

谨以这个十分简单的程序向 Photoshop 的最早作者 Thomas Knoll 致敬！

目录

1 系统的功能设计

1.1 基本特征

基于图层处理，可同时支持 GUI 和命令行操作。可通过程序参数直接处理图片。

注：版本 v0.1 未实现 GUI。

1.2 基本功能

1.2.1 程序基本功能

显示、保存合成的图像，给出当前各层信息，更改画布大小，帮助。

运行已经编辑好的命令文件。

1.2.2 图层管理功能

新建、复制、删除图层，选中图层，调整图层的 Z 轴（上下层）顺序。

1.2.3 图层调整功能

调整图层可视性、透明度、混合模式、名称，移动图层中图像在画布中的位置

注：版本 v0.1 的混合模式仅支持类似 PS 中的正常、变亮、变暗效果。

1.2.4 图像基本功能

载入，保存图像文件，生成图像直方图。

注：版本 v0.1 仅支持 24 位位图文件。

1.2.5 图像调整功能

缩放、旋转、设置透明色。

1.3 扩展功能

1.3.1 “水印大师”

加入更多功能的打水印功能，除了传统的在指定图像的指定位置打上指定 LOGO 的功能外，更整合了前面的基本功能，加入了指定 LOGO 可以旋转、缩放和以不同透明度以及混合模式覆盖在指定图像上。

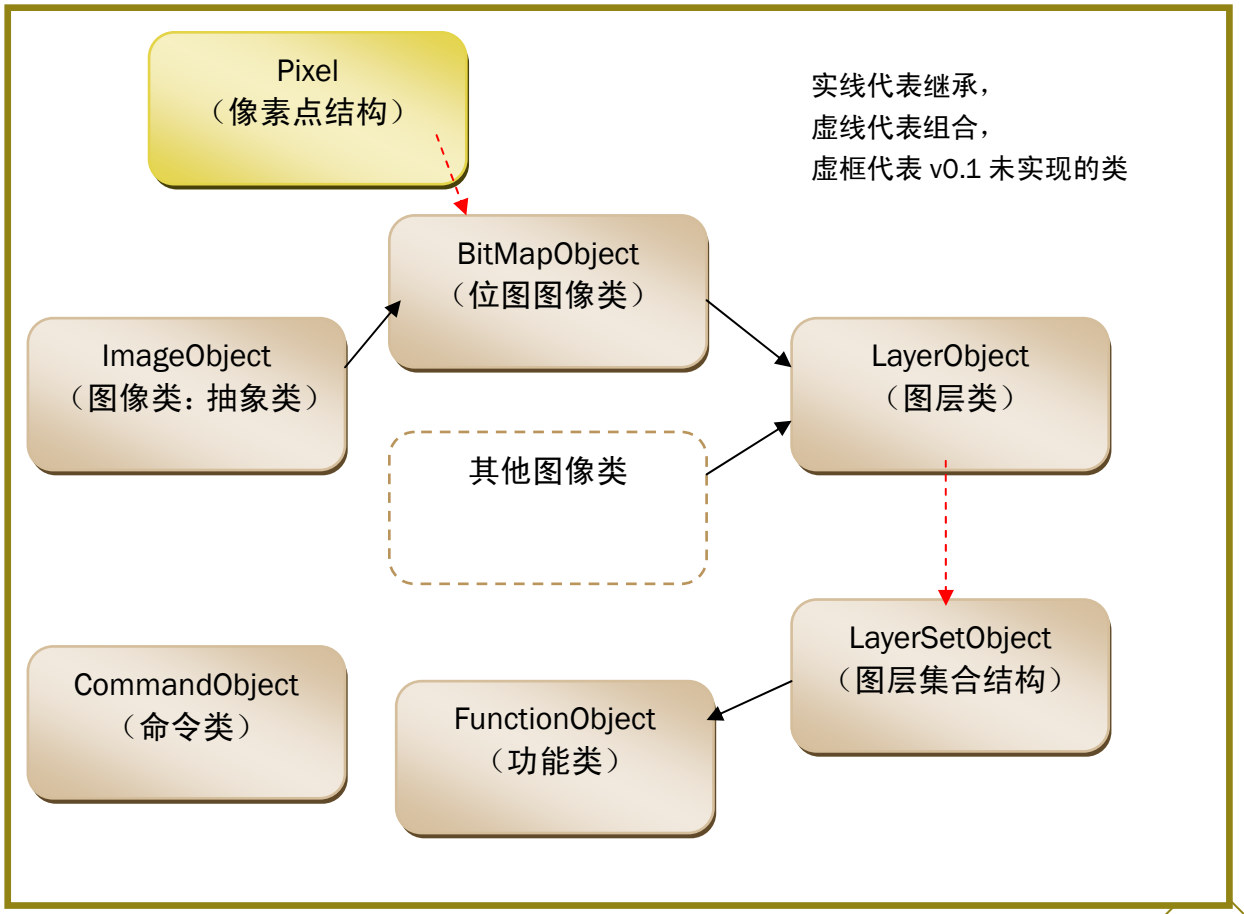
1.3.2 “爱相框”

简单化处理在照片上打相框的操作。

注：版本 v0.1 未加入“爱相框”的设置界面，需要用户自行编辑命令。

2 类体系与主要数据结构

本程序将图像、图层、功能、命令以不同的类来表示。



2.1 ImageObject

```
class ImageObject
{
protected:
    int width;
    int height;
public:
    ImageObject(){};
    ~ImageObject(){};
    virtual char* get_type()=0;//给出图像类型
};
```

2.2 Struct Pixel

```
typedef unsigned char BYTE;
struct Pixel
{
public:
    Pixel(int x=255, int y=255, int z=255);
    Pixel(const Pixel& other);
    Pixel& operator=(const Pixel& other);
    Pixel operator*(double other);
    Pixel operator+(const Pixel& other);
    BYTE r, g, b;//像素点红色、绿色、蓝色的值
};
```

2.3 BitMapObject

```
class BitMapObject: virtual public ImageObject
{
private:
    Pixel **data;//像素点值
    bool *blank; //像素点是否透明
    int width;
    int height;
    long data_start; // The number Indicates the start position of the image
public:
    BitMapObject();
     BitMapObject::BitMapObject(long width1,long height1);
    BitMapObject(const BitMapObject& other);
    ~BitMapObject();
```

```
    // Get Property
    Pixel** get_data() const;
    bool* get_blank() const;
    int get_width() const;
    int get_height() const;
     char* get_type();
    // Set Property
    int set_data(Pixel **px, long w, long h);
    int set_blank(bool *blk);
    int set_size(long width, long height);
    // ImageObject:Load and save
    int BitMapLoad(const char* filename);
    int BitMapSave(const char* filename);
};
```

2.4 LayerObject

```
class LayerObject:public BitMapObject
{
private:
     string name;//图层名称
     string source;//图层中图像的来源
     long loc_x;//相对画布原点（位于左上角）的坐标x
     long loc_y;//相对画布原点（位于左上角）的坐标y
     int z_index;//在文档中的Z轴值（上下层位置）
     int visible;//图层是否可见
     int transparency;//图层透明度
     int mix_mode;//图层混合模式
public:
     static int id;
     LayerObject();
     LayerObject(const LayerObject& other);
     ~LayerObject();
     // Get Property
     long get_x();
     long get_y();
     int get_z();
     int get_visible();
     int get_transparency();
     int get_mix_mode();
     string get_name();
     string get_source();
```

```
    // Set Property
    int set_location(long new_loc_x,long new_loc_y);
    int set_z(int new_z);
    int set_visible(bool vis);
    int set_transparency(int trp);
    int set_mix_mode(int mix);
    int set_name(string& nname);
    int set_source(string& sname);
};
```

2.5 LayerSetObject

```
class LayerSetObject
{
private:
    long width;//画布宽
    long height;//画布高
    vector<LayerObject> vlso;//图层集合
    vector<LayerObject>::iterator cur_l;//当前被选中图层
    LayerObject cur_image;//当前图层合成后的图像
public:
    LayerSetObject();
    ~LayerSetObject();
    int Compound();//合成所有图像
    int SaveImage(const char* filename);//保存合成图像
    int Show();//execute "windows 图片与传真查看器" to display cur_image
    int ShowDetail();//给出当前文档信息，包括画布大小，当前选中图层，每层信息
    //get Property
    vector<LayerObject>::iterator get_cur_l();
    long get_width() const;
    long get_height() const;
    LayerObject& get_cur_image();
    //Set Property
    int set_size(long width1,long height1);//设定画布大小
    //operate the layers
    int SelectLayer(string lname);//选中名称为lname的图层
    int CreateLayer(char* filename);//新建图层，并载入文件名为filename的图像，执行后，新图层被选中
    int DeleteLayer(string lname);//删除名称为layername的图层
    int Duplicate();//复制当前图层
    int ChangeLayerZindex(int id);//复制当前图层，执行后，新图层位于最顶上，且被选中
    int ChangeLayerName(string lname);//将选中图层向上或向下移动
};
```

2.6 FunctionObject

```
class FunctionObject:public LayerSetObject
{
public:
    virtual int Zoom(BitMapObject& bmp_obj,int neww,int newh);//对bmp_obj进行缩放，neww，newh为缩放后的大小
    virtual int Rotate(BitMapObject& bmp_obj,double angle);//bmp_obj逆时针旋转angle度
    virtual int SetBlank(BitMapObject& bmp_obj,Pixel pcolor,int tolerance);//对bmp_obj进行透明色设置，pcolor为颜色，tolerance为容差
    virtual int Histogram(BitMapObject& bmp_obj,int mode);//给出bmp_obj的直方图，里面的模式参见函数定义
};
```

2.7 CommandObject

```
class CommandObject
{
private:
    char parameter[MAX_PARA][MAX_PARA_LENGTH];//命令参数，最多支持MAX_PARA个命令
    int parameter_num;//命令参数个数
    int cmd_split(char* scmd);//将命令字符串scmd以空格为分隔符拆开
    int IsType(string s1,const string& s2);//类型检查，s2为类型扩展名

public:
    CommandObject();
    ~CommandObject();

    int ParseCommand(LayerSetObject& lso,char* scmd);//分析命令，并执行
    int help();//帮助
    int Watermark();//水印设置向导
    int Welcome();//欢迎画面
    int Welcome(char* ImageFile);//欢迎画面，用于运行程序时带参数*.bmp
    int Welcome(char* ImageFile,char* ScriptFile);//欢迎画面，用于运行程序时带参数*.bmp *.bmc
    int Goodbye();//结束画面
};
```

3 关键的技术问题研究

3.1 多个图层如何组织？

本项目采用 vector 容器储存多个图层。

3.2 图像类的设计和算法

处理图像已经有 GDI+这一强大工具，但考虑到本次项目的学习性质，除了有用 GDI+实现的版本外 ,最后提交的版本为自己写的图像类和图像处理函数。图像处理算法见下一节。

3.3 可视化界面的尝试

尝试使用 MFC 和 QT 开发。

注：版本 v0.1 提交版本未实现可视化界面。

4 开发过程中的问题和解决

4.1 BMP图像格式

BMP 是一种与硬件设备无关的图像文件格式，也是我们最常在 PC 机上的 Windows 系统下见到的标准位图格式，使用范围很广泛。它采用位映射存储格式，除了图像深度可选以外，不采用其他任何压缩，因此，BMP 文件所占用的空间很大。它最大的好处就是能被大多数软件“接受”，可称为通用格式。

BMP 文件由文件头、位图信息头、颜色信息和图形数据四部分组成。

1、 BMP 文件头：BMP 文件头数据结构含有 BMP 文件的类型、文件大小和位图起始位置等信息。

```
typedef struct tagBITMAPFILEHEADER{
WORD bfType; // 位图文件的类型，必须为 BM
DWORD bfSize; // 位图文件的大小，以字节为单位
WORD bfReserved1; // 位图文件保留字，必须为 0
WORD bfReserved2; // 位图文件保留字，必须为 0
DWORD bfOffBits; // 位图数据的起始位置，以相对于位图文件头的偏移量表示，以字节为单位
} BITMAPFILEHEADER;
```

2、 位图信息头：BMP 位图信息头数据用于说明位图的尺寸等信息。

```
typedef struct tagBITMAPINFOHEADER{
DWORD biSize; // 本结构所占用字节数
LONGbiWidth; // 位图的宽度，以像素为单位
LONGbiHeight; // 位图的高度，以像素为单位
WORD biPlanes; // 目标设备的级别，必须为 1
WORD biBitCount// 每个像素所需的位数，必须是 1(双色),4(16 色)，8(256 色)或 24(真彩色)之一
DWORD biCompression; // 位图压缩类型，必须是 0(不压缩),1(BI_RLE8 压缩类型)或 2(BI_RLE4 压缩类型)之一
DWORD biSizeImage; // 位图的大小，以字节为单位
LONG biXPelsPerMeter; // 位图水平分辨率，每米像素数
LONG biYPelsPerMeter; // 位图垂直分辨率，每米像素数
DWORD biClrUsed;// 位图实际使用的颜色表中的颜色数
DWORD biClrImportant;// 位图显示过程中重要的颜色数
} BITMAPINFOHEADER;
```

3、颜色表 :颜色表用于说明位图中的颜色 ,它有若干个表项 ,每一个表项是一个 RGBQUAD 类型的结构，定义一种颜色。

```
typedef struct tagRGBQUAD {
BYTE rgbBlue;// 蓝色的亮度(值范围为 0-255)
BYTE rgbGreen; // 绿色的亮度(值范围为 0-255)
BYTE rgbRed; // 红色的亮度(值范围为 0-255)
BYTE rgbReserved;// 保留，必须为 0
} RGBQUAD;
```

颜色表中 RGBQUAD 结构数据的个数有 biBitCount 来确定: 当 biBitCount=1,4,8 时，分别有 2,16,256 个表项; 当 biBitCount=24 时，没有颜色表项。 位图信息头和颜色表组成位图信息，BITMAPINFO 结构定义如下:

```
typedef struct tagBITMAPINFO {
BITMAPINFOHEADER bmiHeader; // 位图信息头
RGBQUAD bmiColors[1]; // 颜色表
} BITMAPINFO;
```

4、 位图数据：位图数据记录了位图的每一个像素值，记录顺序是在扫描行内是从左到右,

扫描行之间是从下到上。位图的一个像素值所占的字节数:

当 biBitCount=1 时，8 个像素占 1 个字节;

当 biBitCount=4 时，2 个像素占 1 个字节;

当 biBitCount=8 时，1 个像素占 1 个字节;

当 biBitCount=24 时,1 个像素占 3 个字节;

Windows 规定一个扫描行所占的字节数必须是 4 的倍数(即以 long 为单位),不足的以 0 填充。

示例分析：

下图是一张图 16 进制数据，以此为例进行分析。在分析中为了简化叙述，以一个字（两个字节为单位，如 424D 就是一个字）为序号单位进行，"h"表示是 16 进制数。

424D 4690 0000 0000 0000 4600 0000 2800

0000 8000 0000 9000 0000 0100 1000 0300

0000 0090 0000 A00F 0000 A00F 0000 0000

0000 0000 0000 00F8 0000 E007 0000 1F00

0000 0000 0000 02F1 84F1 04F1 84F1 84F1

06F2 84F1 06F2 04F2 86F2 06F2 86F2 86F2

1：图像文件头。424Dh='BM'，表示是 Windows 支持的 BMP 格式。

2-3：整个文件大小。4690 0000，为 00009046h=36934。

4-5：保留，必须设置为 0。

6-7：从文件开始到位图数据之间的偏移量。4600 0000，为 00000046h=70，上面的文件头就是 35 字=70 字节。

8-9：位图图信息头长度。

10-11：位图宽度，以像素为单位。8000 0000，为 00000080h=128。

12-13：位图高度，以像素为单位。9000 0000，为 00000090h=144。

14：位图的位面数，该值总是 1。0100，为 0001h=1。

15：每个像素的位数。有 1（单色），4（16 色），8（256 色），16（64K 色，高彩色），24（16M 色，真彩色），32（4096M 色，增强型真彩色）。T408 支持的是 16 位格式。1000 为 0010h=16。

16-17：压缩说明：有 0（不压缩），1（RLE 8，8 位 RLE 压缩），2（RLE 4，4 位 RLE 压缩），3（Bitfields，位域存放）。RLE 简单地说是采用像素数+像素值的方式进行压缩。T408 采用的是位域存放方式，用两个字节表示一个像素，位域分配为 r5b6g5。图中 0300 0000 为 00000003h=3。

18-19：用字节数表示的位图数据的大小，该数必须是 4 的倍数，数值上等于位图宽度×位图高度×每个像素位数。0090 0000 为 00009000h=80×90×2h=36864。

20-21：用象素/米表示的水平分辨率。A00F 0000 为 0000 0FA0h=4000。

22-23：用象素/米表示的垂直分辨率。A00F 0000 为 0000 0FA0h=4000。

2：位图使用的颜色索引数。设为 0 的话，则说明使用所有调色板项。

26-27：对图象显示有重要影响的颜色索引的数目。如果是 0，表示都重要。

28-35：彩色板规范。

4.2 最邻近法与双线性插值法

图像变形后输出像素通常被映射到输入图像中的非整数位置，即位于四个输入像素之间。因此，为了决定与该位置相对应的灰度值，必须进行插值运算。

最简单的插值方法是最邻近插值，即令输出像素的灰度值等于它所映射到的位置最近的

输入像素的灰度值。

双线性插值法与最邻近法相比具有更好的效果。令 $f(x,y)$为两个变量的函数，其在单位正方形顶点的值已知。假设我们希望通过插值得到正方形内任意点的 $f(x,y)$值。我们可令由双线性方程

$$F(x,y)=ax+by+cxy+d$$

来定义的一个双曲抛物面与四个已知点拟合。

从 a 到 d 这四个系数须由已知的四个顶点的 $f(x,y)$值来选定。有一个简单的算法可产生一个双线性插值函数，并使之与四个顶点的 $f(x,y)$值拟合。首先，我们对上端的两个顶点进行线性插值可得

$$F(x,0)=f(0,0)+x*[f(1,0)-f(0,0)]$$

类似地，对于底端两个顶点进行线性插值有

$$F(x,1)=f(0,0)+x*[f(1,0)-f(0,0)]$$

最后，做垂直方向的线性插值，以确定

$$F(x,y)=f(x,0)+y*[f(x,1)-f(x,0)]$$

若将上述式子合并，则有：

$$F(x,y)=[f(1,0)-f(0,0)]*x+[f(0,1)-f(0,0)]*y+[(f(1,1)+f(0,0)-f(0,1)-f(1,0))*x*y+f(0,0)$$

4.3 缩放算法

对于缩放后图像的任意像素点（x,y）进行逆向变换，得到（x/zoomx, y/zoomy）。对该实数坐标点进行双线性插值，最后得到（x,y）的颜色。

4.4 旋转算法

对于缩放后图像的任意像素点(x,y),先将中心点移到原点 ,再进行逆向变换 ,得到(x0, y0) ,最后再将变换后图像左上角点移到原点 ,对每一实数坐标点使用最邻近法确定颜色。其中：

$$X0=(cosa*x-sina*y)$$
$$Y0=(sina*x+cosa*y)$$

对该实数坐标点进行双线性插值，最后得到（x,y）的颜色。

4.5 透明色设置算法

源图像像素点颜色 src 和目标色 dest 的 RGB 值每个分量的差都小于容差 tolerance 的颜色被判定为同一种颜色，这些同种颜色的像素点被设定为透明属性。公式：

$$|src.r - dest.r| <= tolerance$$
$$\&\& |src.g - dest.g| <= tolerance$$
$$\&\& |src.b - dest.b| <= tolerance$$

4.6 直方图算法

分别统计各种颜色值的点的个数，最后经过一定的比例调整，输出到某一位图图像上。

所有颜色直方图计算方法：

$$count[i]=red[i]+green[i]+blue[i]（i，灰度值，0~255）$$

RGB通道颜色计算方法：

$$count[i]=red[i]*0.11+green[i]*0.59+blue[i]*0.3（i，灰度值，0~255）$$

统计出原始值后，有一些灰度值上有极值，需要对直方图进行调整。取所有灰度平均值+2个标准差作为最大显示的灰度值，大于此数字的不予显示，小于此数字的按（该灰度像素点个数*直方图图像高/最大显示的灰度值）计算，最后输出直方图图像。

5 程序存在的问题与改进展望

因为程序编写时间仓促，一些细节并没有很好的处理，而且受水平所限，类的编写难免存在一定的问题 ,这些或多或少造成了这次的遗憾 ,现将存在的问题与对此的展望总结如下：

5.1 Vector容器的利与弊

STL 提供了许多功能强大的容器与算法，给开发者带来极大的便利。本程序中，采用 vector 容器来保存多个图层对象，构成图层集合。然而，因为 BitMapObject 中，表示像素点颜色的变量 (data)和是否为透明的变量(blank)类型分别为 Pixel**和 bool*（指针），在调用 vector 对象的 erase 函数会出现严重错误——函数调用后 ,vlso 中其他元素的 data 和 blank 所指的地址保持不变 ,然而所指地址的内容却可能发生了改变(一般 erase vector 非队末元素会发生这一问题)。

为解决这一问题，目前采取了创建原图层集合的副本，再把副本里未被“删除”的图层拷贝回原图层集合的方法。这一方法的问题是导致运行效率降低。

而为了完美解决这一问题，可能需要修改图像类的 data、blank 的属性或者采用指针链表保存各个图层的信息。如果要提高开发效率，需要采用前者；如果要提高运行效率，则需要采用后者。

5.2 像素点结构的设计

本程序仅实现了极少的一部分功能，许多与颜色有关的功能未去实现。除去开发周期较短的缘故，另外的原因在于像素点颜色系统的设计。目前的颜色系统采用 RGB 颜色系统，但是许多颜色变换采用 HSB 系统（H 色调，S 饱和度，B 明度）更加方便。在后面的版本如果需要加强这方面的功能，可以在 Pixel 结构存储 HSB 颜色系统的变量。

5.3 MORE…

5.3.1 更加友好的界面

支持鼠标直接进行拖动，变形操作的 GUI 界面。

5.3.2 更多的图像支持

加入对 JPG，GIF，PNG 等图像格式的支持。

5.3.3 更多种类的图层

除去图像图层，也加入工作路径图层（形状图层）、调整图层。

5.3.4 更加强大的功能

加入通道的概念。

加入更多的颜色变化，如亮度/对比度，色调/饱和度，黑白。

加入更多的图层混合方式。

6 项目合作总结

我们的小组来自 3 个年级，4 个专业，但是巨大的差异没有对程序的完成造成影响，反而提供了更多元化的思考。我们在实际开发过程中所做的远远比最后提交的程序更多，只是时间所限，最后提交相对更加成熟的功能和体系。在这里，需要赞扬每个成员所付出的努力，虽然结果并没有让努力得到很好的体现，但是更重要的在于开发的过程。经过这次的开发，对于 C++的内涵我们有了更多的了解，对于更加复杂的程序开发的合作，我们也有了一定的体会。

这次合作也存在问题，主要在于类的设计，限于我们的经验所限，存在了多次返回修改类的设计的情况，而合理设计的类是实现高效程序开发的基础。深切体会了这一点之后，相信对于将来更复杂体系的开发，我们能更快找到高效的方法。

组内分工如下

郑博阳

总体设计

图层类实现

部分功能实现

程序调试与文档

吴粤

图像类实现

图像变形功能实现

程序调试与关键问题解决

王益

用户交互设计

MFC 开发版 BitMagic 的设计

李骁

总体设计

GDI+开发版本实现

附录 1 程序使用说明

BitMagic命令行版本全部命令

==========命令详解：图层管理命令==========

create filename

用途：新建图层，并载入文件名为 filename 的图像，执行后，新图层被选中

参数：char* filename；仅限当前目录的 bmp 图像文件；

示例：create temp.bmp

delete

用途：删除当前选中图层

delete layername

用途：删除名称为 layername 的图层

参数：layername 图层名称，可通过 info 命令查看

示例：delete L0

duplicate

用途：复制当前图层，执行后，新图层位于最顶上，且被选中

info

用途：给出当前文档信息，包括画布大小，当前选中图层，每层信息

select layername

用途：选中名称为 layername 的图层

参数：layername 图层名称，可通过 info 命令查看

示例：select L0

setsize width height

用途：将画布调整为宽 width 像素，高 height 像素。调整后，原有图像都在新画布左上角

参数：int width,height；建议两者值均小于 1000

示例：setsize 800 600

……按任意键继续……

==========命令详解：图层调整命令==========

moveto x y

用途：将选中图层移到相对画布（x，y）位置

参数：int x，y;坐标原点（0，0）位于左上角，x>0 代表在原点右方，y>0 代表在原点下方

示例：moveto 20 15

mixmode mode
用途：设置选中图层的混合模式
参数：int mode 的值可以为
0 覆盖
1 变暗（以上下层中较暗的颜色为混合色）
2 变亮（以上下层中较亮的颜色为混合色）
注意，当 mode 不为 0 时，不透明度属性失效
示例：mixmode 1

setname name
用途：将选中图层的名称改为 name
参数：char* name
示例：setname myL

setz mode
用途：将选中图层向上或向下移动
参数：char* mode 的值可以为
top 当前层置顶
bottom 当前层置底
up 当前层向上移一层
down 当前层向下移一层
示例：setz top

transparency trp
用途：设置选中图层的透明度
参数：int trp；trp 范围为 0~100；注意，当 mode 不为 0 时，不透明度属性失效
示例：transparency 10

……按任意键继续……

==========命令详解：图像调整命令==========
rotate angle
用途：逆时针旋转 angle 度当前选中图层的图像
参数：double angle；
示例：rotate 45

zoom width height
用途：将选中图层的图像调整至宽为 width 像素，高为 height 像素的图像
参数：int width;int height;建议 width 和 height 值不大于 1000；
示例：zoom 800 600

zoom zoomx zoomy
用途：将选中图层的图像的宽放大（缩小）至画布宽的 zoomx%，高放大（缩小）至画布

长的

zoomy%
参数：zoomx，zoomy 为字符串形，表示法为 0%至 100%
示例：zoom 200% 200%

setblank red green blue tolerance
用途：对于选中图层的图像，将颜色为以（red，green，blue）为基准的，容差为 tolerance

的颜色设为透明
参数：red，green，blue 范围为 0~255；tolerance 建议设为 10，一般小于 32；
示例：setblank 255 255 255 10 （将白色设为透明色）

histogram [mode]
用途：给出当前图像的直方图
参数：mode 的值可以为
1 R 通道
2 G 通道
3 B 通道
4 所有颜色通道
5 RGB 通道
如果 mode 未给出，则默认为 5
示例：histogram 5

……按任意键继续……

==========命令详解：程序命令==========
exit
用途：退出程序

help
用途：获得帮助

save
用途：保存当前修改结果至 temp.bmp

save filename
用途：保存当前修改结果至名称为 filename 的文件
参数：char* filename；bmp 图像文件名；
示例：save out.bmp

show
用途：调用外部程序，显示当前修改后的图像

附录 2 系统开发日志

日期	日志
12.2	组队完成
12.3	一部分头文件设计完成 修改 BitMapLoad、BitMapSave 函数（感谢 BIP 已经提供的函数）
12.5	完善 类 BitMapObject 的设计，实现类中的成员函数 结构 Pixel 加入成员函数
12.6	学习与数字图像处理的有关内容 完成缩放函数
12.7	完成旋转函数 完成类 LayerObject 的成员函数 完成直方图函数
12.8	完成将透明色设置函数
12.10	小组第二次讨论，修正方案 以 GDI+开发的另一方案也开始同时执行
12.14	完成类 CommandObject 的部分成员函数
12.20	开始编写有关类 LayerSetObject 的成员函数 同时修改类 CommandObject 的 GetCommand 函数
12.24	以 GDI+开发的程序完成，含有除图层外其他基本功能 类 BitMapObject 的再次完善 第一方案中类 LayerSetObject 的部分函数完成

12.26	所有图层操作函数完成 Compound 函数（合成图层）支持透明度的合成 GetCommand 函数基本完成
12.27	Compound 函数支持混合属性的合成
12.29	程序再次测试，代码美化
1.1	修正程序的其他问题
1.2	文档撰写完成