期中Lab

概述

要求实现一个根据绘图指令在内存中绘图的程序，程序读入写在脚本文件中的指令，打印后给出输出的结果。

例如，`script.txt`中的内容为指令序列，运行

```
draw script.txt
```

后会根据script.txt中的指令绘制出图形并输出。

几个假设：

1. 假设脚本文件中指令的格式和顺序都是正确的，即无需做错误处理。
2. 假设虚拟屏幕的大小为(50X50)，任何超出的内容都做截断处理。

指令

基本指令(line, text)

```
//line(x1, y1, x2, y2)
line(0, 0, 19, 19)
//text(x,y,string)
text(11, 0, "!")
```

内存中的结果应该是

```
...
...
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
```

```
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ......
```

为了显示起来比较清晰，输出可以表示为:

```
                                  1
                                1
                              1
                            1
                          1
                        1
                      1
                    1
                  1
                1
              1
            1
          1                   1
        1                     1
      1                       1
    1                         1
  1                           1
1
```

目前，基本的图形指令只支持line与text，相关算法见附录。

宏指令(#,!)

如果我们希望在表示一个时间的同时，带上一个下划线，可以用下面的方式。

```
//下划线
line(0,0,45,0)
//(0,3)为显示字符串的左下角，显示结果为 10:15
text(0,3,"10:15")
```

如果我们需要定义一个新的专门用于显示时间的指令，时间可以显示在任何位置，但总需要像上面的示例那样的一个下划线，那可以这样来定义：

```
//定义一个新的名为timeview的宏指令
#timeview{
    line(0,0,45,0)
    text(0,3,"10:15")
}

//以(50,50)为坐标原点显示timeview
!timeview(50,50)
//以(100,100)为坐标原点显示timeview
!timeview(100,100)
```

即

- #定义了一个宏指令
- !调用了一个宏指令
- 对于任何一个定义好的宏指令m，调用方式为!m(x,y)，表示以(x,y)为坐标原点显示该宏指令。
- 宏指令中可以嵌套调用已经定义好的宏指令。

灰度控制指令(color)

对于二值的黑白图像，只需要一个比特位表示图像上的一点，而256(0~255)级的灰度图像则需要一个字节表示一个点。系统在启动的时候，通过命令行指定灰度级别(2或者256)，程序需要根据情况使用不同的较有效率的存储方式。

```
draw -g 256 script.txt
```

我们可以用下面的指令来指定接下来所有输出的灰度为50:

```
color(50)
line(0,0,45,0)
```

color(n)表示灰度的方式为：

- 对于二值的黑白图像，参数n为 0 和 非0 分别表示内存中相应比特位的 0 和 1
- 对于256级灰度图像，内存中存储实际的color指令中的灰度值n，输出显示的数值为(int)(grey/25.6)，即变换为0~9之间的数值。
- 系统初始状态color为0

undo/redo 指令(undo,redo)

除了以#开头的宏指令定义和undo/redo本身，其它的指令都可以undo和redo。

自动测试

每条指令都需要有对应的一个或者多个自动化测试的代码。

手工测试用例

2周后提供

评分标准

1. 总体权重

内容	权重	说明
自动测试用例	60%	
手工测试用例	30%	
代码质量	10%	评价内容为是否恰当使用了设计模式

2. 指令权重

指令	权重
line	15%
text	15%
#/!	15%
color	15%
undo	20%
redo	20%

提交内容

1. 源代码
2. 运行自动测试用例和手工测试用例的步骤（README.md)
3. 简单说明使用了什么面向对象的设计模式（Design.md,500字以内）

以上所有内容打包在一个zip文件中，文件名为： ${学号}_${姓名}.zip

附录

画线

画线有成熟的算法，可以到网上自行查找，在理解的基础上可以直接参考相关的代码，建议使用**bresenham**算法。Lab中对具体算法的选择和性能没有要求。

字符的显示

text指令为在指定的位置绘制字符、字符串。无需换行，超出部分截断即可。字符间距固定为1个像素。请使用提供的字体库，库中的字体的格式见附件`asciis.h`。如果需要使用不同的程序设计语言，可以将`asciis.h`中的数据进行转换使用。

`asciis.h`中存放了从空格(space)开始的ascii码的点阵结构,每一行数据代表了一个8*8像素的ascii字符，包括了对应字符在8行中有几列位置是0，几列是1。例如代表`!`的八个字节分别为:

```
0x10,0x10,0x10,0x10,0x10,0x00,0x10,0x00
```

表示为二进制为:

```
0 0 0 1 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0
```

显示为`!`的形状