

CLOCK 大作业报告

计算机 2 班

2252941 杨瑞灵

完成日期：2023 年 9 月 26 日

目录

1	设计思路与功能描述	3
1.1	设计思路	3
1.2	功能描述	3
2	问题及解决方法	5
2.1	问题一：实现时钟转动效果	5
2.2	问题二：抗锯齿	5
2.3	问题三：抗锯齿运算较慢使得画面闪烁	6
2.4	问题四：透明色和发光效果	6
2.5	问题五：渐变色	7
3	心得体会	8
3.1	9 月 24 日	8
3.2	9 月 26 日	8
4	源代码	10
4.1	clock.h	10
4.2	main.app	11
4.3	clock_face.app	15
4.4	anti_aliasing.app	18

1 设计思路与功能描述

1.1 设计思路

第一步：新建保存图像的 IMAGE 对象指针 background，将背景图片导入 background，并显示在图形窗口

第二步：画钟表面，两个同心圆，刻度线和数字。getimage 从当前设备中获取图像，将钟表和背景一起存在 background 指针里

第三步：无键盘操作时进入循环，不断获取时间信息，并新建 IMAGE 对象指针 partical，进行离屏渲染，画上 background，以及钟表指针，最后一起显示在屏幕上。

优化 1：对圆和直线进行 SDF+alpha 抗锯齿操作

优化 2：渐变色发光和透明效果

优化 3：发光和透明效果,alpha 修改实现，与抗锯齿整合为一个函数

1.2 功能描述

1. 钟嘛显示时间的，好看一点还能让人心情愉悦？下面是我 24 号和后面修改后的时钟对比



图 1: clock 原图 VS 修改

2 问题及解决方法

2.1 问题一：实现时钟转动效果

问题描述：在循环中上一秒和下一秒需要重新渲染时钟指针，然而暴力重画整个画面会出现图像闪动现象。

问题解答：可以建立一个 IMAGE 对象指针 background 来存放整个钟表表面，每次渲染只需要显示 background 图片，然后再画上钟表指针。

2.2 问题二：抗锯齿

问题描述：easyx 画出来的图像边界有明显齿轮状

问题解答：运用 SDF+alpha 抗锯齿操作解决圆和直线的抗锯齿,下面以圆为例,搜索半径-SDF_WIDTH_CIRCLE 到半径 +SDF_WIDTH_CIRCLE 的正方形范围，如果点到圆的距离 d 满足 (在圆内为负，圆外为正)

$$0 \leq d \leq 0.5$$

那么就按照距离比例 mix_color, 将背景色和前景色进行混合，达到模糊边缘的效果.

```
void SDF_circle(int center_x, int center_y, COLORREF color, int radius)
{
    for (int x = center_x - radius - SDF_WIDTH_CIRCLE; x < center_x + radius + SDF_WIDTH_CIRCLE; x++) {
        for (int y = center_y - radius - SDF_WIDTH_CIRCLE; y < center_y + radius + SDF_WIDTH_CIRCLE; y++) {
            double d;
            d = fabs(sqrt((pow(x - center_x, 2) + pow(y - center_y, 2))) - radius); // 点到圆边的距离

            double alpha = 0.5 - d / 2; // ? ? ? 0.5-d
            if (d < 0)
                continue;
            if (alpha >= 0 && alpha <= 1) {
                COLORREF bg = getpixel(x, y);
                COLORREF result = mix_color(bg, color, alpha);
                putpixel(x, y, result);
            }
        }
    }

    setfillcolor(color);
    setlinecolor(color);
    solidcircle(center_x, center_y, radius);
}
```

图 2: SDF_CIRCLE

2.3 问题三：抗锯齿运算较慢使得画面闪烁

问题描述：因为抗锯齿是对一片像素点进行计算和判断，所以钟表指针的画图速度较慢，指针转动时会伴随着画面闪动现象

问题解答：最开始，我试着减小 SDF 中搜索的像素点个数，速度确实变快了但是每隔几秒依然会有闪动现象（我也不知道为什么会是隔几秒一次）。所以我查看了学长的文档，发现了离屏渲染这个好方法，其实这和我问题一里面的解决方法是一样的，只是我在屏幕上操作然后保存为 IMAGE 指针对象，而学长直接在 IMAGE 对象上操作。

```
//离屏渲染，储存为particle
IMAGE* particle = new IMAGE(WINDOW_X, WINDOW_Y); //对particle_buffer进行处理
SetWorkingImage(particle);
putimage(0, 0, &background);
hand h(t.tm_hour, t.tm_min, t.tm_sec);
h.draw();
//将图片particle渲染至屏幕
SetWorkingImage(NULL);
putimage(0, 0, particle);
```

图 3: 离屏渲染

2.4 问题四：透明色和发光效果

问题描述：一直想设置透明色但是好像 easyx 没有透明的参数，所以修改了 alpha 的参数来混合背景色和前景色，以实现混色效果

问题解答：其实还挺戏剧的，最开始是因为抗锯齿效果不好，我仔细观察了一下学长 PPT 的函数

$$\alpha = 0.5 - d$$

发现其实 d 只取了 -0.5-0.5 的范围, 我只要增加范围, 就能增加模糊效果从而得到更模糊的边缘。于是我给 d 除了个 2

$$\alpha = 0.5 - d/2$$

确实效果不错，然后我又想，如果我除以 20, 40 甚至更大呢，那混色范围就会更大是不是就会出现透明边缘？事实是我的想法是正确的。最终，我用 SDF+alpha 抗锯齿实现了透明效果



图 4: 透明和发光

2.5 问题五：渐变色

问题解答：不想写了直接放图！

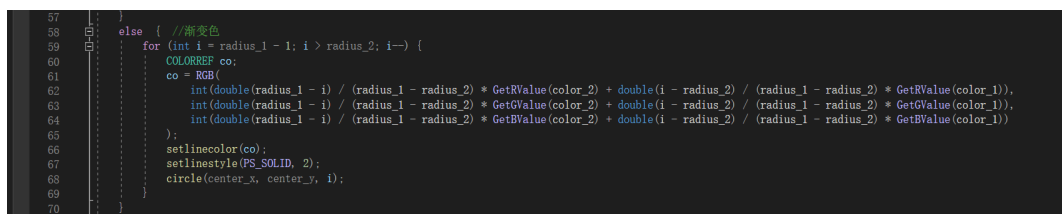


图 5: 渐变色

3 心得体会

3.1 9 月 24 日

以下是我 9 月 24 日的心得体会，由于我现在只是简单设计+抗锯齿，所以我不确定下周会不会再卷卷画面和阴影。但是鼠鼠我是真不想再卷了就现在这个也做了我一个晚上加一个上午。哎但是要是大家太能卷鼠鼠我啊还是得跟着做捏。真的是每天都在卷死别人和卷死自己的恐惧中阴暗的爬行（

现在是正文：怎么说呢还是很有意思的让我想起小时候用小虫子画圆的经历（？忘记是用什么画的了），当时就觉得这个圆边缘很丑（如果可以的话想试试高斯模糊，之前在 3blue1brown 上面看过用这种方法压缩图像，也许下周会试试？感觉计算机要用到一些数学的知识，高数还是不能丢。

这次应该算是锻炼我们自己查资料和学习的能力，我觉得面对这种不会的工具还是应该在最开始做一个系统的了解，先查查资料看看 B 站学习一下基础，然后再开始设计构思，最后才是写代码。这因该是我这次较快完成作业的原因，可能有些同学会以来就开始上手代码，个人觉得还是不太建议。

and STRUCT 封装真的很好用，就结构很清晰吧，把钟表指针封装为一个结构体，只需要在主函数里面调用结构体函数就好了，不过，结构体必须要和主函数写在一起或者放在头文件里面，为什么不能设置一个放在其他 cpp 里面的机制？

看到之前学长学姐做的很好看呢，喜欢他们的设计。

感谢助教的指导让我这么快写出来一个能看的表。

3.2 9 月 26 日

又花了一天时间来修改，首先是抗锯齿公式的调整，增加了模糊的范围。其次加上了透明感和发光的动态，以及渐变色，主打一个约花哨约好看。写完还是很有成就感，特别是在抗锯齿的基础上修改，实现透明感。其实整体还是非常简单，我也看到很多同学做的钟表，各有特色，都非常好看。

希望这就是最后的成品了。

4 源代码

4.1 clock.h

```
1      #pragma once
2      #define PI 3.1415926
3      //画图窗口大小
4      #define WINDOW_X 800
5      #define WINDOW_Y 600
6      //圆心坐标
7      #define CENTER_X WINDOW_X / 2
8      #define CENTER_Y WINDOW_Y / 2
9      //两个圆的半径
10     #define RADIUS_1 int(min(WINDOW_X,WINDOW_Y)/2*0.9)
11     #define RADIUS_2 int(min(WINDOW_X,WINDOW_Y)/2*0.55)
12     // 抗锯齿
13     #define SDF_WIDTH_CIRCLE 30//抗锯齿处理像素宽度 circle
14     #define SDF_WIDTH_LINE 10//抗锯齿处理像素宽度 line
15     //各种color
16     #define MY_PURPLE RGB(150,142,185)
17     #define MY_WHITE RGB(248, 241, 248)
18     #define MY_RED RGB(245,150,150)
19     #define MY_LIGHT_RED RGB(255, 186, 201)
20     #define MY_PINK RGB(249,218,228)
21     #define MY_WHITE_2 RGB(249, 236, 243)
22     #define MY_BLUE RGB(239,239,246)
23     //初始界面
24     void init();
25     //画抗锯齿 / 渐变色 / 透明圆 / 发光色
```

```

26 void SDF_circle(int center_x, int center_y, COLORREF
    color_1, COLORREF color_2, int radius_1, int
    radius_2, int SDF_degree = 2);
27
28 //line抗锯齿 / 放光色
29 void SDF_line(int x1, int y1, int x2, int y2, COLORREF
    color, int thickness, int SDF_degree = 2);

```

4.2 main.app

```

1  #include <iostream>
2  #include <iomanip>
3  #include <graphics.h>
4  #include <math.h>
5  #include <conio.h>
6  #include <time.h>
7  #include <stdio.h>
8  #include "clock.h"
9  using namespace std;
10 //表的指针的结构体
11 struct hand {
12 private:
13     double hour_hand_angle;
14     double minite_hand_angle;
15     double second_hand_angle;
16     int hour;
17     int minite;
18     int second;
19     int light;

```

```

20     friend void SDF_circle(int center_x, int center_y,
        COLORREF color, int radius);
21     friend void SDF_line(int x1, int y1, int x2, int
        y2, COLORREF color, int thickness);
22     void draw_hms(double len, double angle);
23 public:
24     hand(int h, int m, int s, int l);
25     void draw();
26 };
27 /*****
28 函数功能：初始化结构体，角度和时间
29 *****/
30 hand::hand(int h, int m, int s, int l)
31 {
32     light = l;
33     hour = h;
34     hour_hand_angle = -PI / 2 + h * 2 * PI / 12 + m *
        2 * PI / 12 / 60 + s * 2 * PI / 12 / 60 / 60;
35     minite = m;
36     minite_hand_angle = -PI / 2 + m * 2 * PI / 60 + s
        * 2 * PI / 60 / 60;
37     second = s;
38     second_hand_angle = -PI / 2 + s * 2 * PI / 60;
39 }
40 /*****
41 函数功能：画每一根指针
42 形参：double len：指针长度
43         double angle：指针角度
44 返回值：无

```

```

45  *****/
46  void hand::draw_hms(double len, double angle)
47  {
48      int x1 = int(CENTER_X + len * 0.85 * cos(angle));
49      int y1 = int(CENTER_Y + len * 0.85 * sin(angle));
50      int x2 = int(CENTER_X - len * 0.15 * cos(angle));
51      int y2 = int(CENTER_Y - len * 0.15 * sin(angle));
52
53      SDF_line(x1, y1, x2, y2, MY_PINK,
54              5); // 指针的本体线条
55      // SDF_circle(x2, y2, MY_PINK, NULL, 5,
56      NULL); // 短头的圆圈
57      SDF_circle(CENTER_X, CENTER_Y, MY_RED, NULL, 10,
58              0, (20 - light / 2) + 5); // 中心的圆圈 40
59      SDF_circle(CENTER_X, CENTER_Y, MY_WHITE_2, NULL,
60              5, 0, light / 4 + 5); // 20
61      SDF_circle(x1, y1, MY_RED, NULL, 10, 0,
62              light); // 长头的圆圈 20
63      SDF_circle(x1, y1, MY_WHITE_2, NULL, 4, 0, (10 -
64              light / 4) + 5); // 10
65
66  }
67
68  /*****
69  函数功能：调用draw_hms画三根指针
70  形参：无
71  返回值：无
72  *****/
73  void hand::draw()
74  {

```

```

68         draw_hms(RADIUS_1 * 1.00, second_hand_angle); //秒针
69         draw_hms(RADIUS_1 * 0.75, minite_hand_angle); //分针
70         draw_hms(RADIUS_1 * 0.40, hour_hand_angle); //时针
71     }
72     /*****
73     函数功能：判断是否需要重新渲染
74     形参：struct tm t：现在的时间
75           struct tm t_copy：上一次的时间
76     返回值：bool类型 1：不需要渲染
77           0：需要重新渲染
78     *****/
79     //bool cmp(struct tm t, struct tm t_copy)
80     //{
81     //    return t.tm_sec == t_copy.tm_sec;
82     //}
83     int main()
84     {
85         // 获取当地时间
86         struct tm t,t_copy;
87         time_t now;
88         time(&now);
89         localtime_s(&t, &now);
90
91         // 初始化绘图窗口
92         initgraph(WINDOW_X, WINDOW_Y);
93         IMAGE* background = new IMAGE(WINDOW_X,
94         WINDOW_Y); //生成image指针对象
95         loadimage(background, L"./bk.jpg", WINDOW_X,
96         WINDOW_Y);

```

```

95     SetWorkingImage(background);
96     putimage(0, 0, background);
97     // 自定义图形初始化函数，用于绘制时钟界面
98     init();
99     // 无键盘操作时进入循环
100    int light = 0;
101    int dir = 2;
102    while (!_kbhit()) {
103        t_copy = t;
104        time(&now);
105        localtime_s(&t, &now);
106        light += dir;
107        if (light >= 40)
108            dir = -2;
109        else if (light <= 2)
110            dir = 2;
111        //if (cmp(t, t_copy))
112        //    continue;
113        //离屏渲染，储存为particle
114        IMAGE* particle = new IMAGE(WINDOW_X,
115                                     WINDOW_Y); //对particle_buffer进行处理
116        SetWorkingImage(particle);
117        putimage(0, 0, background);
118        struct hand h(t.tm_hour, t.tm_min, t.tm_sec,
119                     light);
120        h.draw();
121        //将图片particle渲染至屏幕
122        SetWorkingImage(NULL);
123        putimage(0, 0, particle);

```

```

122     }
123     _getch();          // 按任意键准备退出时钟程序
124     closegraph();      // 退出图形界面
125     return 0;
126 }

```

4.3 clock_face.app

```

1  #define _CRT_SECURE_NO_WARNINGS
2  #include <iostream>
3  #include <iomanip>
4  #include <graphics.h>
5  #include <math.h>
6  #include <conio.h>
7  #include <time.h>
8  #include <stdio.h>
9  #include "clock.h"
10 using namespace std;
11 /*****
12  函数功能：文字输出抗锯齿以及设计格式
13  形参：int x：文字输出横坐标
14         int y：文字输出纵坐标
15         TCHAR s[]：字符串
16         COLORREF color：颜色
17         int Height：文字高度，宽度自适应
18  返回值：无
19  *****/
20 void text(int x,int y, TCHAR s[],COLORREF color, int
    Height)

```



```

21 {
22     // 设置输出效果为抗锯齿
23     LOGFONT f;
24     gettextstyle(&f); //
        获取当前字体设置
25     f.lfHeight = Height; //
        设置字体高度
26     setbkmode(TRANSPARENT); // 背景透明字体
27     settextrcolor(color);
28     _tcscpy_s(f.lfFaceName, _T("黑体")); //
        设置字体为“黑体”（高版本 VC 推荐使用 _tcscpy_s
        函数）
29     f.lfQuality = ANTIALIASED_QUALITY; //
        设置输出效果为抗锯齿
30     settextrstyle(&f); //
        设置字体样式
31     outtextxy(x, y, s);
32 }
33 /*****
34 函数功能：初始化界面
35 形参：无
36 返回值：无
37 *****/
38 void init()
39 {
40     SDF_circle(CENTER_X, CENTER_Y, MY_PURPLE, NULL,
41         RADIUS_1, RADIUS_1, 60); // 透明
42     // SDF_circle(CENTER_X, CENTER_Y, MY_PURPLE, NULL,
43         RADIUS_1 + 5, RADIUS_1 + 5, 10);

```

```

42     SDF_circle(CENTER_X, CENTER_Y, MY_BLUE, NULL,
                RADIUS_2, RADIUS_2, 150);
43     SDF_circle(CENTER_X, CENTER_Y, MY_PURPLE, MY_BLUE,
                RADIUS_1, RADIUS_2); // 画圆
44     SDF_circle(CENTER_X, CENTER_Y, MY_PURPLE, NULL,
                RADIUS_1, RADIUS_1); // 抗锯齿
45     SDF_circle(CENTER_X, CENTER_Y, MY_BLUE, NULL,
                RADIUS_2, RADIUS_2);
46
47     // 画线和数字
48     for (int i = 1; i <= 60; i++) {
49         double sita = 2 * PI / 60 * i - PI / 2;
50         int x1 = int(CENTER_X + RADIUS_1 * 0.95 *
                    cos(sita)); // 线的起点
51         int y1 = int(CENTER_Y + RADIUS_1 * 0.95 *
                    sin(sita));
52         int x2; // 线的终点
53         int y2;
54         int x3; // 外圈数字的位置
55         int y3;
56         int x4; // 内圈数字位置
57         int y4;
58         if (i % 5) {
59             x2 = int(CENTER_X + RADIUS_1 * 0.9 *
                    cos(sita));
60             y2 = int(CENTER_Y + RADIUS_1 * 0.9 *
                    sin(sita));
61         }
62         else {

```

```

63         x2 = int(CENTER_X + RADIUS_1 * 0.85 *
64                 cos(sita));
65         y2 = int(CENTER_Y + RADIUS_1 * 0.85 *
66                 sin(sita));
67         x3 = int(CENTER_X + RADIUS_1 * 0.75 *
68                 cos(sita) - 13);
69         y3 = int(CENTER_Y + RADIUS_1 * 0.75 *
70                 sin(sita) - 13);
71         x4 = int(CENTER_X + RADIUS_2 * 0.75 *
72                 cos(sita) - 13);
73         y4 = int(CENTER_Y + RADIUS_2 * 0.75 *
74                 sin(sita) - 17);
75         TCHAR s[5];
76         _stprintf(s, _T("%d"), i);
77         text(x3, y3, s, WHITE, 35); // 外圈数字
78         _stprintf(s, _T("%d"), i/5);
79         text(x4, y4, s, MY_PURPLE, 40); // 内圈数字
80     }
81     SDF_line(x1, y1, x2, y2, WHITE, 4, 10);
82     SDF_line(x1, y1, x2, y2, WHITE, 4);
83 }
84 }

```

4.4 anti_aliasing.app

```

1     #define _CRT_SECURE_NO_WARNINGS
2     #include <iostream>
3     #include <iomanip>
4     #include <graphics.h>

```

```

5      #include <math.h>
6      #include <conio.h>
7      #include <time.h>
8      #include <stdio.h>
9      #include "clock.h"
10     using namespace std;
11     //SDF+alpha blending
12     /*****
13     函数功能：混合颜色
14     形参：COLORREF bg：背景色
15           COLORREF color：前景色
16           double alpha：比例
17     返回值：无
18     *****/
19     COLORREF mix_color(COLORREF bg, COLORREF color, double
20         alpha)//背景色 前景色 alpha=0.5-d 返回RGB( , , )
21     {
22         COLORREF result;
23         result = RGB(GetRValue(bg) * (1 - alpha) +
24             GetRValue(color) * alpha, GetGValue(bg) * (1 -
25             alpha) + GetGValue(color) * alpha,
26             GetBValue(bg) * (1 - alpha) + GetBValue(color)
27             * alpha);
28         return result;
29     }
30
31     /*****
32     函数功能：画抗锯齿/渐变色/透明圆
33     形参：int center_x：圆心横坐标

```

```

29         int center_y; 圆心纵坐标
30         COLORREF color_1: 颜色1
31         COLORREF
            color_2: 颜色2, 为NULL时表示画的是纯色圆solidcircle
32         int radius_1: 半径1
33         int
            radius_2: 半径2, 两个半径相等表示画的是circle
34         int
            SDF_degree: 抗锯齿的程度, 默认值是2, 如果设置10及以上可以达到半
35         返回值: 无
36         *****/
37     void SDF_circle(int center_x, int center_y, COLORREF
            color_1, COLORREF color_2, int radius_1, int
            radius_2, int SDF_degree)
38     {
39         for (int x = center_x - radius_1 -
            SDF_WIDTH_CIRCLE; x < center_x + radius_1 +
            SDF_WIDTH_CIRCLE; x++) {
40             for (int y = center_y - radius_1 -
                SDF_WIDTH_CIRCLE; y < center_y + radius_1 +
                SDF_WIDTH_CIRCLE; y++) {
41                 double d;
42                 d = sqrt((pow(x - center_x, 2) + pow(y -
                    center_y, 2))) -
                    radius_1; //点到圆边的距离
43                 double alpha = 0.5 - d / SDF_degree;
44                 if (alpha >= 0 && alpha <= 1) {
45                     COLORREF bg = getpixel(x, y);
46                     COLORREF result = mix_color(bg,

```

```

        color_1, alpha);
47         putpixel(x, y, result);
48     }
49 }
50
51 if (radius_1 == radius_2) //只进行抗锯齿
52     ;
53 else if (color_2 == NULL || radius_2 == 0) {
54     //没有渐变
55     setfillcolor(color_1);
56     setlinecolor(color_1);
57     solidcircle(center_x, center_y, radius_1);
58 }
59 else { //渐变色
60     for (int i = radius_1 - 1; i > radius_2; i--) {
61         COLORREF co;
62         co = RGB(
63             int(double(radius_1 - i) / (radius_1 -
64                 radius_2) * GetRValue(color_2) +
65                 double(i - radius_2) / (radius_1 -
66                     radius_2) * GetRValue(color_1)),
67             int(double(radius_1 - i) / (radius_1 -
68                 radius_2) * GetGValue(color_2) +
69                 double(i - radius_2) / (radius_1 -
70                     radius_2) * GetGValue(color_1)),
71             int(double(radius_1 - i) / (radius_1 -
72                 radius_2) * GetBValue(color_2) +
73                 double(i - radius_2) / (radius_1 -
74                     radius_2) * GetBValue(color_1))

```

```

65         );
66         setlinecolor(co);
67         setlinestyle(PS_SOLID, 2);
68         circle(center_x, center_y, i);
69     }
70 }
71 }
72
73 /*****
74 函数功能：画抗锯齿直线
75 形参：int x1: 起点横坐标
76       int y1: 起点纵坐标
77       int x2: 终点横坐标
78       int y2: 终点纵坐标
79       COLORREF color: 颜色
80       int thickness: 线宽
81 返回值：无
82 *****/
83 void SDF_line(int x1, int y1, int x2, int y2, COLORREF
84              color, int thickness, int SDF_degree)
85 {
86     for (int x = min(x1, x2) - SDF_WIDTH_LINE; x <
87          max(x1, x2) + SDF_WIDTH_LINE; x++) {
88         for (int y = min(y1, y2) - SDF_WIDTH_LINE; y <
89              max(y1, y2) + SDF_WIDTH_LINE; y++) {
90             double d;
91             bool b1 = ((x - x1) * (x2 - x1) + (y - y1)
92                      * (y2 - y1)) >= 0;
93             bool b2 = ((x - x2) * (x1 - x2) + (y - y2)

```

```

    * (y1 - y2)) >= 0;
90     if (b1 && b2)
91         d = fabs(((x2 - x1) * (y - y1) - (y2 -
            y1) * (x - x1)) / sqrt(pow(x1 - x2,
            2) + pow(y1 - y2, 2))) - thickness
            / 2;
92     else if (!b1)
93         d = sqrt(pow(x - x1, 2) + pow(y - y1,
            2))-thickness / 2;
94     else
95         d = sqrt(pow(x - x2, 2) + pow(y - y2,
            2))- thickness / 2;
96     double alpha = 0.5 - d / SDF_degree;
97     if (alpha >= 0 && alpha <= 1) {
98         COLORREF bg = getpixel(x, y);
99         COLORREF result = mix_color(bg, color,
            alpha);
100         putpixel(x, y, result);
101     }
102 }
103 }
104 setlinestyle(PS_SOLID | PS_ENDCAP_ROUND,
    thickness);
105 setlinecolor(color);
106 line(x1, y1, x2, y2);
107 }

```