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

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

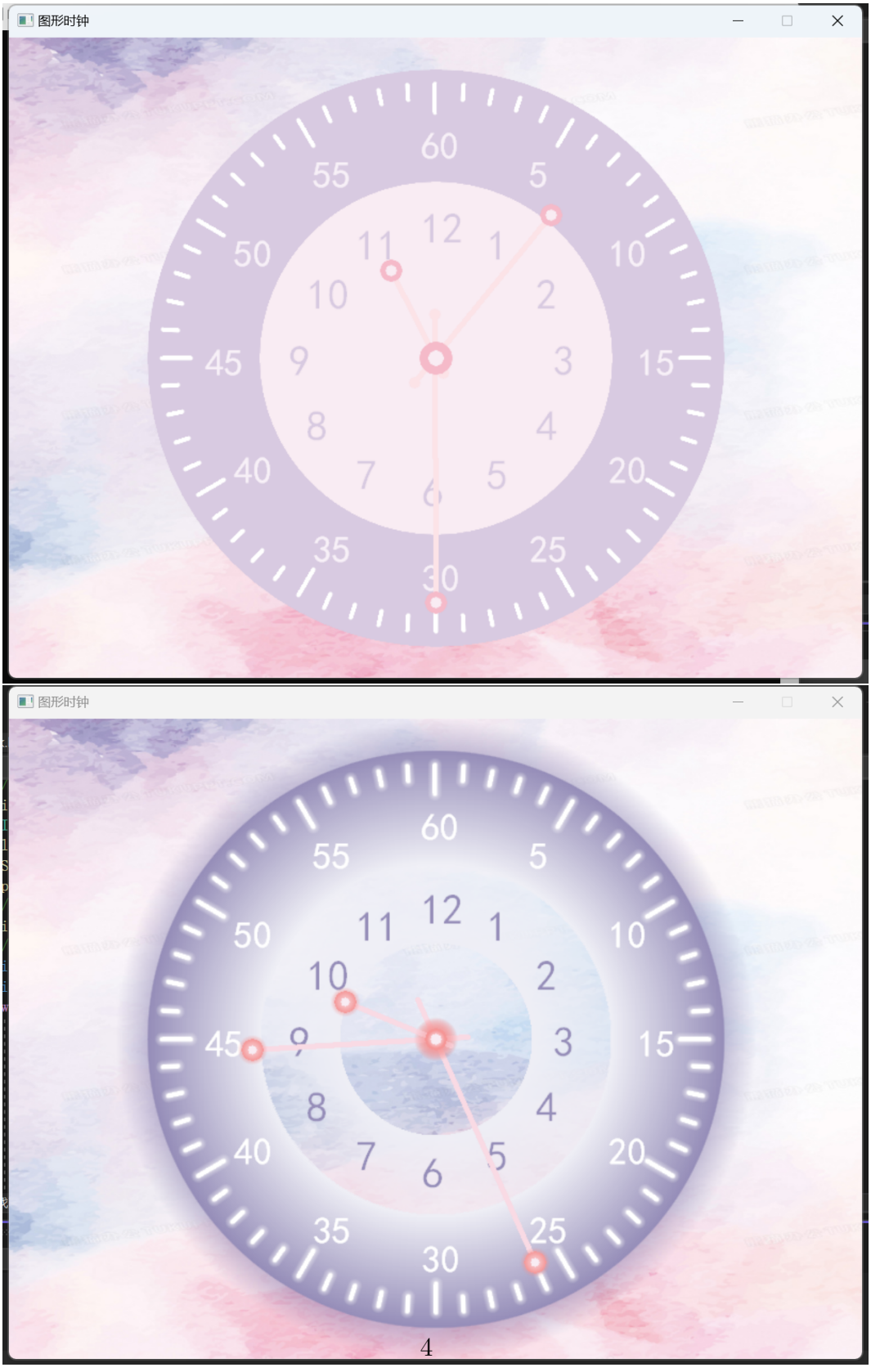

图 1: clock 原图 VS 修改

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

```
/*********************************
函数功能：画抗锯齿/渐变色/透明圆
形参：int center_x：圆心横坐标
      int center_y：圆心纵坐标
      COLORREF color_1：颜色1
      COLORREF color_2：颜色2，为NULL时表示画的是纯色圆solidcircle
      int  radius_1：半径1
      int  radius_2：半径2，两个半径相等表示画的是circle
      int SDF_degree：抗锯齿的程度，默认值是2，如果设置10及以上可以达到半透明的效果
返回值：无
*********************************/
void SDF_circle(int center_x, int center_y, COLORREF color_1, COLORREF color_2, int  radius_1, int radius_2, int SDF_degree)
{
    for (int x = center_x - radius_1 - SDF_WIDTH_CIRCLE; x < center_x + radius_1 + SDF_WIDTH_CIRCLE; x++) {
        for (int y = center_y - radius_1 - SDF_WIDTH_CIRCLE; y < center_y + radius_1 + SDF_WIDTH_CIRCLE; y++) {
            double d;
            d = sqrt((pow(x - center_x, 2) + pow(y - center_y, 2))) - radius_1;//点到圆边的距离
            double alpha = 0.5 - d / SDF_degree;
            if (alpha >= 0 && alpha <= 1) {
                COLORREF bg = getpixel(x, y);
                COLORREF result = mix_color(bg, color_1, alpha);
                putpixel(x, y, result);
            }
        }
```

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！

```
    }
    else {  //渐变色
        for (int i = radius_1 - 1; i > radius_2; i--) {
            COLORREF co;
            co = RGB(
                int(double(radius_1 - i) / (radius_1 - radius_2) * GetRValue(color_2) + double(i - radius_2) / (radius_1 - radius_2) * GetRValue(color_1)),
                int(double(radius_1 - i) / (radius_1 - radius_2) * GetGValue(color_2) + double(i - radius_2) / (radius_1 - radius_2) * GetGValue(color_1)),
                int(double(radius_1 - i) / (radius_1 - radius_2) * GetBValue(color_2) + double(i - radius_2) / (radius_1 - radius_2) * GetBValue(color_1))
            );
            setlinecolor(co);
            setlinestyle(PS_SOLID, 2);
            circle(center_x, center_y, i);
        }
    }
```

图 5: 渐变色

3 心得体会

3.1 9 月 24 日

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

```
#pragma once
#define PI 3.1415926
//画图窗口大小
#define WINDOW_X 800
#define WINDOW_Y 600
//圆心坐标
#define CENTER_X WINDOW_X / 2
#define CENTER_Y WINDOW_Y / 2
//两个圆的半径
#define RADIUS_1 int(min(WINDOW_X,WINDOW_Y)/2*0.9)
#define RADIUS_2 int(min(WINDOW_X,WINDOW_Y)/2*0.55)
// 抗锯齿
#define SDF_WIDTH_CIRCLE 30//抗锯齿处理像素宽度 circle
#define SDF_WIDTH_LINE 10//抗锯齿处理像素宽度 line
//各种color
#define MY_PURPLE RGB(150,142,185)
#define MY_WHITE RGB(248, 241, 248)
#define MY_RED RGB(245,150,150)
#define MY_LIGHT_RED RGB(255, 186, 201)
#define MY_PINK RGB(249,218,228)
#define MY_WHITE_2 RGB(249, 236, 243)
#define MY_BLUE RGB(239,239,246)
//初始界面
void init();
//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,
                5);//指针的本体线条
54          //SDF_circle(x2, y2, MY_PINK, NULL, 5,
                NULL);//短头的圆圈
55          SDF_circle(CENTER_X, CENTER_Y, MY_RED, NULL, 10,
                0, (20 - light / 2) + 5);//中心的圆圈 40
56          SDF_circle(CENTER_X, CENTER_Y, MY_WHITE_2, NULL,
                5, 0, light / 4 + 5);//20
57          SDF_circle(x1, y1, MY_RED, NULL, 10, 0,
                light);//长头的圆圈 20
58          SDF_circle(x1, y1, MY_WHITE_2, NULL, 4, 0, (10 -
                light / 4) + 5);//10
59
60      }
61      /********************************
62      函数功能: 调用draw_hms画三根指针
63      形参: 无
64      返回值: 无
65      ********************************/
66      void hand::draw()
67      {
```

```
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,
               WINDOW_Y);//生成image指针对象
94          loadimage(background, L"./bk.jpg", WINDOW_X,
               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,
                    WINDOW_Y);//对particle_buffer进行处理
115             SetWorkingImage(particle);
116             putimage(0, 0, background);
117             struct hand h(t.tm_hour, t.tm_min, t.tm_sec,
                    light);
118             h.draw();
119             //将图片particle渲染至屏幕
120             SetWorkingImage(NULL);
121             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          settextcolor(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          settextstyle(&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,
                RADIUS_1, RADIUS_1, 60);//透明
41          //SDF_circle(CENTER_X, CENTER_Y, MY_PURPLE, NULL,
                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 *
                        cos(sita));
64                  y2 = int(CENTER_Y + RADIUS_1 * 0.85 *
                        sin(sita));
65                  x3 = int(CENTER_X + RADIUS_1 * 0.75 *
                        cos(sita) - 13);
66                  y3 = int(CENTER_Y + RADIUS_1 * 0.75 *
                        sin(sita) - 13);
67                  x4 = int(CENTER_X + RADIUS_2 * 0.75 *
                        cos(sita) - 13);
68                  y4 = int(CENTER_Y + RADIUS_2 * 0.75 *
                        sin(sita) - 17);
69                  TCHAR s[5];
70                  _stprintf(s, _T("%d"), i);
71                  text(x3, y3, s, WHITE, 35);//外圈数字
72                  _stprintf(s, _T("%d"), i/5);
73                  text(x4, y4, s, MY_PURPLE, 40);//内圈数字
74              }
75              SDF_line(x1, y1, x2, y2, WHITE, 4, 10);
76              SDF_line(x1, y1, x2, y2, WHITE, 4);
77          }
78      }
```

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
        alpha)//背景色 前景色 alpha=0.5-d 返回RGB( , , )
20  {
21      COLORREF result;
22      result = RGB(GetRValue(bg) * (1 - alpha) +
            GetRValue(color) * alpha, GetGValue(bg) * (1 -
            alpha) + GetGValue(color) * alpha,
            GetBValue(bg) * (1 - alpha) + GetBValue(color)
            * alpha);
23      return result;
24  }
25
26  /*********************************
27  函数功能: 画抗锯齿/渐变色/透明圆
28  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) {
        //没有渐变
54          setfillcolor(color_1);
55          setlinecolor(color_1);
56          solidcircle(center_x, center_y, radius_1);
57      }
58      else  {  //渐变色
59          for (int i = radius_1 - 1; i > radius_2; i--) {
60              COLORREF co;
61              co = RGB(
62                  int(double(radius_1 - i) / (radius_1 -
                        radius_2) * GetRValue(color_2) +
                        double(i - radius_2) / (radius_1 -
                        radius_2) * GetRValue(color_1)),
63                  int(double(radius_1 - i) / (radius_1 -
                        radius_2) * GetGValue(color_2) +
                        double(i - radius_2) / (radius_1 -
                        radius_2) * GetGValue(color_1)),
64                  int(double(radius_1 - i) / (radius_1 -
                        radius_2) * GetBValue(color_2) +
                        double(i - radius_2) / (radius_1 -
                        radius_2) * GetBValue(color_1))
```

```
                );
                setlinecolor(co);
                setlinestyle(PS_SOLID, 2);
                circle(center_x, center_y, i);
            }
        }
    }

    /*********************************
    函数功能: 画抗锯齿直线
    形参: int x1: 起点横坐标
          int y1: 起点纵坐标
          int x2: 终点横坐标
          int y2: 终点纵坐标
          COLORREF color: 颜色
          int thickness: 线宽
    返回值: 无
    *********************************/
    void SDF_line(int x1, int y1, int x2, int y2, COLORREF
       color,int thickness, int SDF_degree)
    {
        for (int x = min(x1, x2) - SDF_WIDTH_LINE; x <
           max(x1, x2) + SDF_WIDTH_LINE; x++) {
            for (int y = min(y1, y2) - SDF_WIDTH_LINE; y <
               max(y1, y2) + SDF_WIDTH_LINE; y++) {
                double d;
                bool b1 = ((x - x1) * (x2 - x1) + (y - y1)
                   * (y2 - y1)) >= 0;
                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     }
```