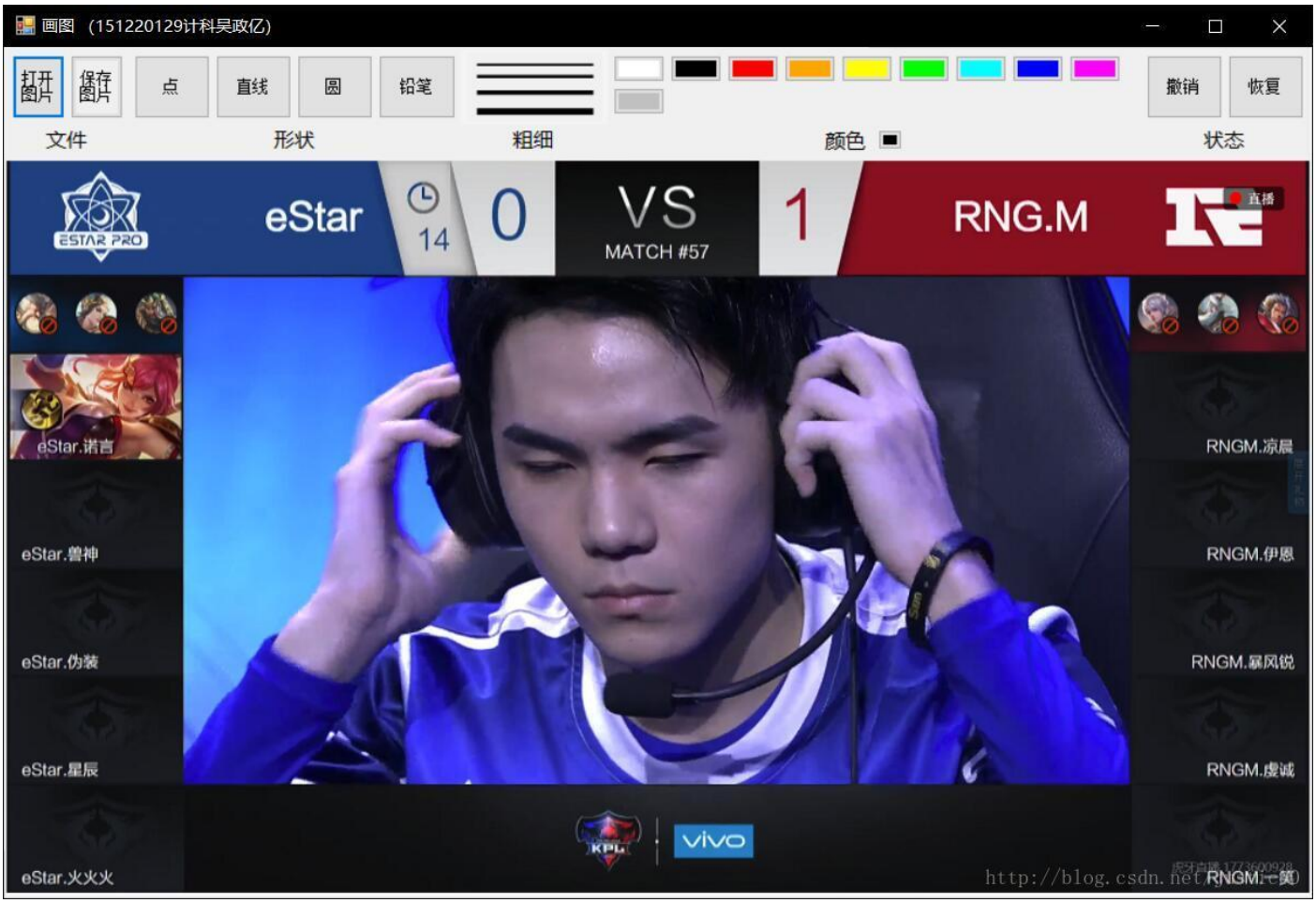
《计算机图形学》系统设计10月进展报告

151220129 计科 吴政亿

（南京大学 计算机科学与技术系，南京 210093）

摘要：用C#向windows的画图看齐，实现了文件的打开与保存，直线/圆/曲线/点等的形状绘制，可以自主选择粗细与颜色，并且应用派生的ArrayList类Step保存了每一步的操作，可以进行撤销与恢复操作。



实验环境

报告采用markdown编写，并另存为了html与pdf格式，置于压缩包内report文件夹，code文件夹中包含了本次阶段性代码，Paint文件则是project的打包发布版本，便于助教测试。

基于Visual Studio 2015下的 C# 编写。

代码结构浅析

代码主要分为三个部分：

Name	Information
Program.cs	应用程序的主入口点。
Form1.cs	画图程序的主窗口与实现的部分代码。
StepPaint.cs	ArrayList的派生类，用于存储每一步的操作。

Form1.cs

Variable Name	Information
CASE now_case	定义了当前的操作形状，例如画线与画圆
BREATH bh	定义了画笔的粗细
Color color	当前画笔颜色
int x1,y1	定义了直线，圆等的起点坐标，在鼠标摁下时更新
bool mouse_down	用于判断鼠标时候松开
bool is_back	用于判断是否处于撤销状态
StepPaint Step	存储每一步的操作动作

Function Name	Information
private void InitForm1()	初始化成员变量
private void drawPixel(x,y)	在(x,y)处画点(粗细取决于bh)
private void DDALine(x1,y1,x2,y2)	画线函数DDA算法
private void BresenhamLine(x1, y1, x2, y2)	画线函数Bresenham算法

Function Name	Information
private void MidpointLine(x0, y0, x1, y1)	画线函数中心点生成算法
private void BresenhamCircle(R, xc, yc)	画圆函数Bresenham算法
private void button_*Color*_Click(sender, e)	不同颜色对应了不同的按钮来改变当前画笔颜色
private void openfile_Click(sender, e)	打开文件响应函数
private void savefile_Click(sender, e)	保存文件响应函数
private void button_*Function*_Click(sender, e)	相应功能的相应函数，其中有撤销、恢复以及各种形状选择
private void pictureBox1_MouseUp(sender, e)	鼠标左键松开时响应函数
private void pictureBox1_MouseMove(sender, e)	鼠标移动时响应函数
private void pictureBox1_MouseDown(sender, e)	鼠标左键摁下时响应函数
private void *粗细* ToolStripMenuItem_Click(sender, e)	设置画笔粗细的响应函数

StepPoint.cs

Function Name	Information
ArrayList	存储所有步骤的Image

Function Name	Information
int StepImage_now	记录当前显示的图像在ArrayList的下标
void ClearStep()	清空ArrayList与StepImage_now
void InitStep(Image)	用一张Image初始化并作为保卫者
void AddStep(obj)	添加一个Image Step
bool StepIsNull()	判断StepImage_now是否位于最底部
bool StepIsFull()	判断StepImage_now是否位于最顶部
Image PopStep()	撤销当前步骤
Image PushStep()	恢复上一个步骤
Image RefreshStep()	将最后一个步骤永久撤销
void RemoveNullStep()	当撤销状态下执行一个步骤，将清除后面的步骤

核心代码实现

这部分简述略过，因为都是按照上课讲的实现的算法

DDA直线算法

```
private void DDALine(int x1, int y1, int x2, int y2){
    double dx, dy, e, x, y;
    dx = x2 - x1;
    dy = y2 - y1;
    e = (Math.Abs(dx) > Math.Abs(dy)) ? Math.Abs(dx) : Math.Abs(dy);
    dx /= e; dy /= e;
    x = x1;
    y = y1;
    for (int i = 1; i <= e; i++)
    {
        drawPixel((int)(x + 0.5), (int)(y + 0.5));
        x += dx;
        y += dy;
    }
```

```
}
```

Bresenham画圆算法

```
private void BresenhamCircle(int R, int xc, int yc){
    int x, y, p;
    x = 0; y = R;
    p = 3 - 2 * R;
    for (; x <= y; x++)
    {
        /* 下面八个分别对称 八个部分*/
        drawPixel(x + xc, y + yc);
        drawPixel(x + xc, -y + yc);
        drawPixel(y + xc, x + yc);
        drawPixel(y + xc, -x + yc);
        drawPixel(-x + xc, y + yc);
        drawPixel(-x + xc, -y + yc);
        drawPixel(-y + xc, x + yc);
        drawPixel(-y + xc, -x + yc);
        if (p >= 0)
        {
            p += 4 * (x - y) + 10;
            y--;
        }
        else
        {
            p += 4 * x + 6;
        }
    }
}
```

实验感悟

对直线，圆等的生成算法有了更深刻的了解，并将ppt中的所有算法都简要测试了一遍并选择了最为简洁的方法，另外对像素与图像生成也有了实践的经验，下一次的目标是对于线的粗细有着更好的优化方案，现在的线粗细采用的是画圆的方案，当选项为粗时略有卡顿。