

**创意编程**

**大作业开发计划书**

**汤凯鹏 520432910016**

**2020~2021秋季学期2021年11月19日**

**目 录**

[1 设计工具简介 3](#_Toc87970178)

[1.1 选题背景及意义 3](#_Toc87970179)

[1.2 目标要求 3](#_Toc87970180)

[1.3 术语说明 3](#_Toc87970181)

[2 功能需求分析（人机交互方式，输入输出属性） 4](#_Toc87970182)

[3 程序开发设计 5](#_Toc87970183)

[3.1 总体架构设计 5](#_Toc87970184)

[3.2 功能模块设计 5](#_Toc87970185)

[3.3 数据结构设计 5](#_Toc87970186)

[3.4 源代码文件组织设计 11](#_Toc87970187)

[3.5 滤镜设计描述 13](#_Toc87970188)

[4 部署运行和使用说明 15](#_Toc87970189)

[4.1 编译安装 15](#_Toc87970190)

[4.2 使用操作 15](#_Toc87970191)

[5 开发过程 18](#_Toc87970192)

[5.1 开发计划 18](#_Toc87970193)

[6 参考文献资料 18](#_Toc87970194)

多功能风格化滤镜工具设计

# 设计工具简介

## 选题背景及意义

在信息爆炸的时代，图片与视频需要给人一种更加新颖的刺激。传统计算机绘画与视频编辑工具行为相对固定，操作空间较少，变化控制不够深入，这就迫切需要以更加数字化的手断控制图片。Processing即能完成这一任务，网络上已经有不少Processing制作滤镜的方法，但不幸的是，一个程序往往只能进行一类的操作，同时这些操作依然缺少可变性，操作可视化没有很好地完成，控制者只能去自行修改代码。作为一个完整独立的设计工具，我们需要将各种操作整合起来，并赋予相当多的直接的控制手段，让从来没有接触过Processing的用户能很好地利用这个设计工具，达到数字化修改图片的目的。

## 目标要求

1、总体要求：

基于processing实现一个简易的多功能风格化滤镜工具，支持图片输入、编辑、文件保存功能，并设计实现尽量种不同风格的滤镜，并配有多种数据修改。

1. 功能要求：

多功能风格化滤镜工具软件主要涵盖了绘制功能、修改功能、存取功能、模板功能。

3、设计要求：

界面的用户注重友好性，要求画面美观，菜单、按钮等图形控件要合理使用，支持鼠标、 键盘等交互操作，操作容易。

## 术语说明

**滤镜（filter）**，主要是用来实现图像的各种特殊效果。它在[Photoshop](https://baike.baidu.com/item/Photoshop)中具有非常神奇的作用。所有的滤镜在Photoshop中都按分类放置在菜单中，使用时只需要从该菜单中执行这命令即可。滤镜的操作是非常简单的，但是真正用起来却很难恰到好处。滤镜通常需要同[通道](https://baike.baidu.com/item/%E9%80%9A%E9%81%93/12004680)、[图层](https://baike.baidu.com/item/%E5%9B%BE%E5%B1%82/1494948)等联合使用，才能取得最佳艺术效果。如果想在最适当的时候应用滤镜到最适当的位置，除了平常的美术功底之外，还需要用户对滤镜的熟悉和操控能力，甚至需要具有很丰富的想象力。这样，才能有的放矢的应用滤镜，发挥出艺术才华。

# 功能需求分析（人机交互方式，输入输出属性）

1、业务逻辑要求：

（1）本软件是为了满足用户滤镜制作的需求，尤其是当用户想通过绘制滤镜生成风格化图片。

（2）本软件基于对象实时绘制，因此用户想修改某个对象的属性时，就必须点选操作组件。

（3）一次只能修改一张图片，替换图片将会覆盖原图片。

2、需求功能要求：

1. 绘制功能：

内置多种笔刷并能进行修改，能够在面板上用鼠标拖拽的方式绘制。界面有实时显示的鼠标坐标的功能，以便用户绘制。

（2） 修改功能：

能够修改对象的样式，每个滤镜相应的操作都会修改文件的绘制样式如像素大小、像素排列、文本内容、边框颜色、填充颜色、文本颜色、边框粗细、对象大小，并创作出多种多样的风格化效果。

（3） 存取功能：

能够实现新建、另存为、读取文件，并支持processing支持的数据文件格式。

1. 模板功能：

软件内每个滤镜内置了默认风格的模板（即参数调整至适合范围），用户能够随时打开并修改使用。

3、数据结构要求：

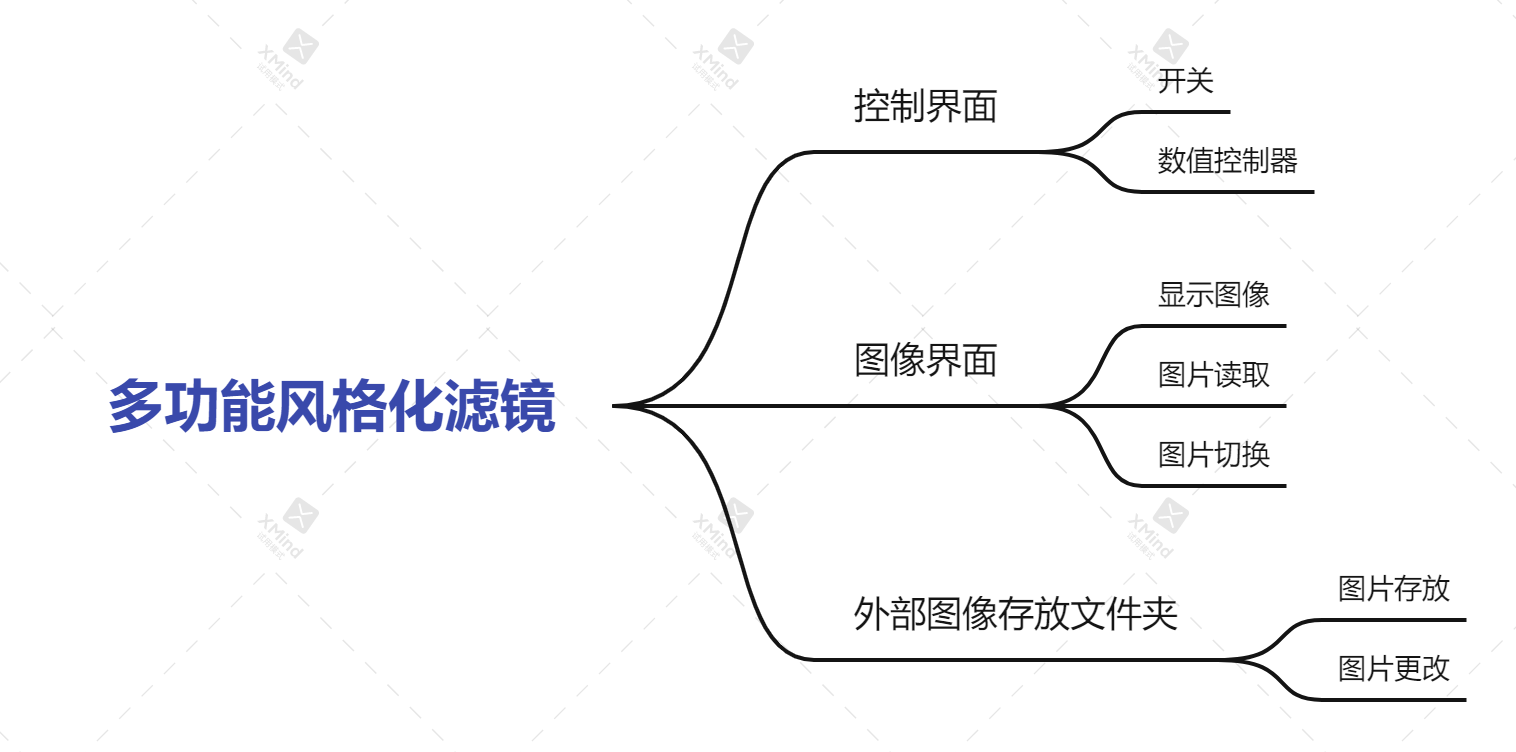
在本软件用到的数据类型有int、float、void等，因此用到了线形数据结构（例如数组）。由于一个对象可能引出许多下一级分支，因此本软件也用到了图形数据结构（“多对多”的结点），并分类、分文件编写，并且多出运用函数嵌套。

4、性能要求：

众所周知，由于各种底层问题，processing相较其他工具性能较差，建议运行软件的电脑打开电池性能最佳模式以获得良好的使用体验。屏幕分辨率调至合适。软件的使用要求流畅不卡顿。

# 程序开发设计

## 总体架构设计



## 功能模块设计

把文件拆分成两大模块：UI界面模块、图片处理模块。

UI界面模块：

设定自定义的滤镜工具，各个控件的位置、颜色、说明等。

图片处理模块：

展示图片效果，读取、切换和保存图片。

## 数据结构设计

1、全局变量

ImageLoader loader;

ImageList list;

Image img; //库

PFont myFont; //文字

=

int index=0; //保存文件用

color col;

ControlFrame cf;

//一下均为控件UI

float speed1,speed2,speed3,speed4;

float cl;

float pos;

float c0, c1, c2, c3,c4,c5,c6,c7,c8;

boolean auto1;

boolean auto2\_1;

boolean auto2\_2;

boolean auto2\_3;

boolean auto2\_4;

boolean auto2\_5;

boolean auto2\_6;

boolean auto2\_7;

boolean auto3;

boolean auto4;

boolean auto5;

boolean auto6;

boolean auto7;

2、类

/\*UI的类\*/

class ControlFrame extends PApplet {

  int w, h;

  PApplet parent;

  ControlP5 cp5;

  public ControlFrame(PApplet \_parent, int \_w, int \_h, String \_name) {

    super();

    parent = \_parent;

    w=\_w;

    h=\_h;

    PApplet.runSketch(new String[]{this.getClass().getName()}, this);

  }

  public void settings() {

    size(w, h);

  }

  public void setup() {

    surface.setLocation(10, 10);

    cp5 = new ControlP5(this);

    cp5.addToggle("on/off\_1")

       .plugTo(parent, "auto1")

       .setPosition(10, 10)

       .setSize(20, 20)

       .setValue(false);

    cp5.addToggle("on/off\_2")

       .plugTo(parent, "auto2\_1")

       .setPosition(10, 70)

       .setSize(20, 20)

       .setValue(false);

    cp5.addToggle("GRAY")

       .plugTo(parent, "auto2\_2")

       .setPosition(340, 50)

       .setSize(20, 20)

       .setValue(false);

    cp5.addToggle("INVERT")

       .plugTo(parent, "auto2\_3")

       .setPosition(340, 90)

       .setSize(20, 20)

       .setValue(false);

           cp5.addToggle("ERODE")

       .plugTo(parent, "auto2\_4")

       .setPosition(340, 130)

       .setSize(20, 20)

       .setValue(false);

           cp5.addToggle("DILATE")

       .plugTo(parent, "auto2\_5")

       .setPosition(340, 170)

       .setSize(20, 20)

       .setValue(false);

           cp5.addToggle("on/off\_3")

       .plugTo(parent, "auto2\_6")

       .setPosition(10, 130)

       .setSize(20, 20)

       .setValue(false);

           cp5.addToggle("on/off\_4")

       .plugTo(parent, "auto2\_7")

       .setPosition(10, 190)

       .setSize(20, 20)

       .setValue(false);

       cp5.addToggle("on/off\_5")

       .plugTo(parent, "auto3")

       .setPosition(10, 250)

       .setSize(20, 20)

       .setValue(false);

       cp5.addToggle("on/off\_6")

       .plugTo(parent, "auto4")

       .setPosition(10, 310)

       .setSize(20, 20)

       .setValue(false);

       cp5.addToggle("on/off\_7")

       .plugTo(parent, "auto5")

       .setPosition(10, 360)

       .setSize(20, 20)

       .setValue(false);

       cp5.addToggle("on/off\_8")

       .plugTo(parent, "auto6")

       .setPosition(10, 420)

       .setSize(20, 20)

       .setValue(false);

       cp5.addToggle("on/off\_9")

       .plugTo(parent, "auto7")

       .setPosition(10, 480)

       .setSize(20, 20)

       .setValue(false);

    cp5.addNumberbox("black lines")

       .plugTo(parent, "c0")

       .setRange(0, 300)

       .setValue(0)

       .setPosition(100, 10)

       .setSize(100, 20);

    cp5.addNumberbox("GRAY THRESHOLD")

       .plugTo(parent, "c1")

       .setRange(0, 100)

       .setValue(0)

       .setPosition(100, 70)

       .setSize(100, 20);

    cp5.addNumberbox("POSTERIZE")

       .plugTo(parent, "c2")

       .setRange(20, 250)

       .setValue(0)

       .setPosition(100, 130)

       .setSize(100, 20);

       cp5.addNumberbox("BLUR")

       .plugTo(parent, "c4")

       .setRange(0, 300)

       .setValue(0)

       .setPosition(100, 190)

       .setSize(100, 20);

       cp5.addNumberbox("angle lines")

       .plugTo(parent, "c5")

       .setRange(0, 300)

       .setValue(0)

       .setPosition(100, 250)

       .setSize(100, 20);

    cp5.addSlider("thread")

       .plugTo(parent, "speed1")

       .setRange(0, 3000)

       .setValue(0.0)

       .setPosition(100, 310)

       .setSize(200, 20);

           cp5.addSlider("height")

       .plugTo(parent, "speed2")

       .setRange(0, height)

       .setValue(0.0)

       .setPosition(100, 360)

       .setSize(200, 20);

       cp5.addSlider("pixel size")

       .plugTo(parent, "speed3")

       .setRange(1, 200)

       .setValue(12)

       .setPosition(100, 420)

       .setSize(200, 20);

         cp5.addSlider("textSize")

       .plugTo(parent, "speed4")

       .setRange(1, 30)

       .setValue(10)

       .setPosition(100, 480)

       .setSize(200, 20);

       cp5.addKnob("THRESHOLD")

       .plugTo(parent, "c3")

       .setPosition(340, 290)

       .setSize(50, 50)

       .setRange(0, 150)

       .setValue(200);

       cp5.addKnob("strong")

       .plugTo(parent, "c6")

       .setPosition(340, 350)

       .setSize(50, 50)

       .setRange(0, 500)

       .setValue(150);

       cp5.addKnob("color height")

       .plugTo(parent, "c7")

       .setPosition(340, 410)

       .setSize(50, 50)

       .setRange(0,height)

       .setValue(150);

       cp5.addKnob("distance")

       .plugTo(parent, "c8")

       .setPosition(340, 470)

       .setSize(50, 50)

       .setRange(0,30)

       .setValue(10);

    cp5.addColorWheel("col" , 0 , 600 , 200 ).setRGB(color(128,0,255))

       .plugTo(parent, "col");

       //text(cp5.get(ColorWheel.class,"col").getRGB(),340, 550);

  textFont(myFont);

  textSize(12);

  textAlign(CENTER, TOP);

  fill(0, 0, 0);

  //上面四条设定了文字的样式，要在text()渲染文字前设定好。

  text("将图片放入data文件夹，点击图片窗口会随机切换图片", 160, 540);

  text("在选中图片窗口的情况下按‘s’保存图片",120,560);

  text("（多图片只能随机切换，不用修改图片名）",120,580);

  }

3、各个函数（详见下个部分）

4、保存文件的数据结构

int index=0; //保存文件用

void keyPressed() {

  if(key=='s'||key=='S'){

    save("mywork"+index+".png");

    index++;

}

## 源代码文件组织设计

<文件目录结构>

[1]test1主函数头文件

[2]button1 UI文件

[3]filter1 第一个点阵滤镜，测试原型用

[4]filter2 filter函数中的所有滤镜

[5]filter3 线条滤镜

[6]filter4 filter三角函数飞线滤镜

[7]filter5 filter像素塌陷滤镜

[8]filter6 filter马赛克滤镜

[9]filter7 filter文字滤镜

[10]color色彩UI（已注释）

1. 文件函数结构

[1]主要的声明

import controlP5.\*;

import at.mukprojects.imageloader.\*;

import at.mukprojects.imageloader.file.\*;

import at.mukprojects.imageloader.image.\*;

String path = "yourPath";

ImageLoader loader;

ImageList list;

Image img;

PFont myFont;

[2]setup函数

void setup() {

  size(1000, 800);

  path = dataPath("");

  loader = new FileImageLoader(this);

  list = loader.start(path);

UIsetup();

//Colorsetup();

myFont=createFont("PTMono.ttc", 72);

}

[3]draw函数

void draw() {

  UIdraw();

  if (img == null) {

    img = list.getRandom();

  } else {

    image(img.getImg(), 0, 0, width, height);

  }

if (auto1) filter1();

if (auto2\_1) filter2\_1();

if (auto2\_2) filter2\_2();

if (auto2\_3) filter2\_3();

if (auto2\_4) filter2\_4();

if (auto2\_5) filter2\_5();

if (auto2\_6) filter2\_6();

if (auto2\_7) filter2\_7();

if (auto3) filter3();

if (auto4) filter4();

if (auto5) filter5();

if (auto6) filter6();

if (auto7) filter7();

}

[4]图片切换函数

void mousePressed() {

  img = list.getRandom();

}

[5]保存文件函数

int index=0; //保存文件用

void keyPressed() {

  if(key=='s'||key=='S'){

    save("mywork"+index+".png");

    index++;

}

[6]UI函数（见上文中的类）

[7]各个滤镜函数（略）

## 滤镜设计描述

**（1）**只有单一开关的滤镜

GRAY：图像变灰

INVERT：图片颜色反转

ERODE：腐蚀效果

DILATE：膨胀效果

**（2）**有开关和一个参数控件的滤镜

BLACK LINES：风格化点阵，在图片亮度低的地方绘制黑色点迹

参数控件描述：参数控制识别黑色的阈值

GRAY THRESHOLD：将图片变为黑白并只显示黑色部分

参数控件描述：参数控制识别黑色的阈值

POSTERIZE： 色调分离

参数控件描述： 控制色调分离的承程度，数值越低程度越高

BLUR： 模糊图片

参数控件描述： 控制模糊承程度

ANGLE LINES： 线条风格化图片，图片的亮度对应线条的角度

参数控件描述： 控制线条角度的变化程度

**（3）**有开关和两个参数控件的滤镜

三角函数滤镜：风格化三角函数线条，在图片亮度低的地方绘制三角函数线条

参数控件描述：THREAD：参数控制线的总数

THRESHOLD：参数控制图片黑色部分的阈值

塌陷滤镜：将图片变为塌陷粒子

参数控件描述：HEIGHT：参数控制塌陷的高度

STRONG：参数控制塌陷的强度

马赛克滤镜： 马赛克画图片

参数控件描述： PIXEL SIZE：参数马赛克的大小

COLOR HEIGHT：参数控制黑色马赛克的高度阈值

文字滤镜： 文字滤镜，将暗部的文字放大，亮部的文字缩小

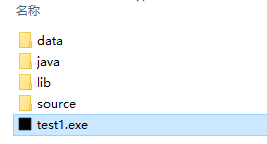
参数控件描述： TEXT SIZE：参数控制文字的大小

THRESHOLD：参数控制图片暗部的阈值

# 部署运行和使用说明

## 安装

1. 将多功能风格化滤镜工具压缩包解压，windows打开application.windows，Linux打开application. Linux（不需要安装processing）

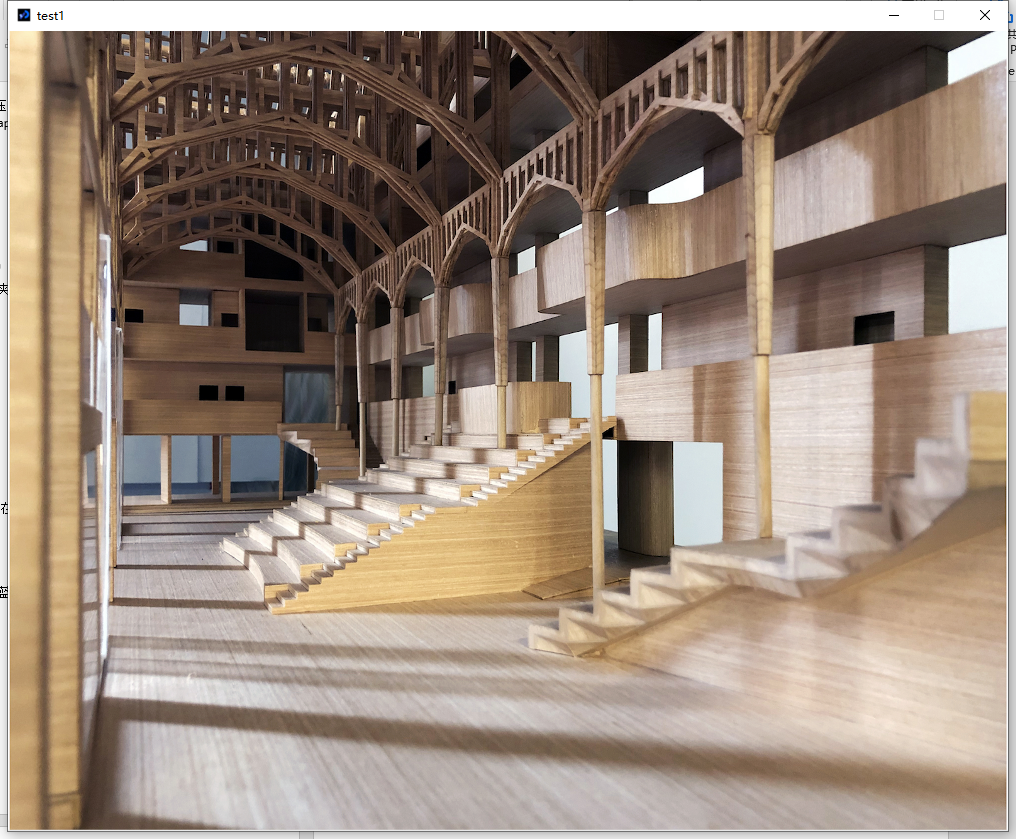


1. 将想要修改的图片直接放入data文件夹。
2. 运行test1.exe程序。
3. 确定可以打开后即可开始使用，出现两个窗口。

## 使用操作

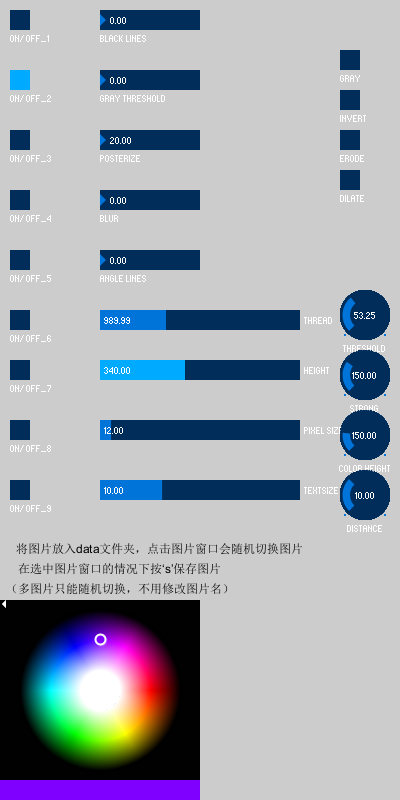
1. 确定图片

文件自动打开data文件夹中的图片，如果data文件夹中有多个图片，鼠标单击图片会进行随机切换。（不需要改图片名字）



1. 进行操作

可以进行以下操作：



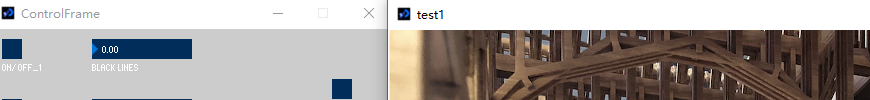
（1）打开开关：所有的正方形按钮是滤镜开关

（2）调节：有调节控件的滤镜可以调节， 按住鼠标上下移动调整左上方五个滤镜控制器，按住鼠标左右移动调整下方四个滤镜控制器。

（支持同时打开并调节多个滤镜，滤镜可以叠加）

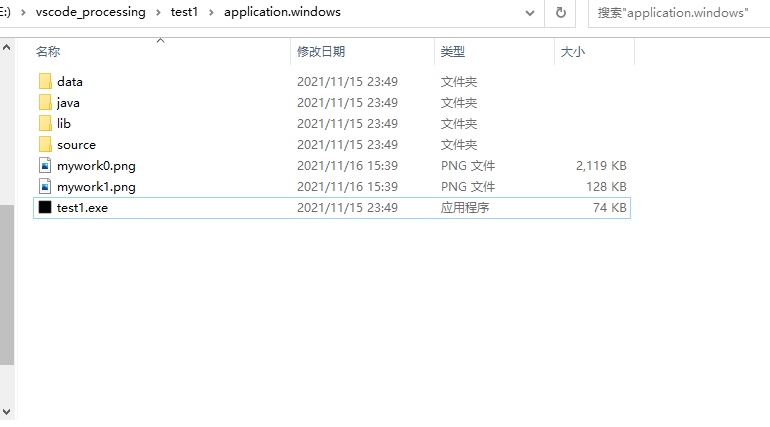
1. 保存图片

鼠标选择图片窗口栏，高亮图片窗口，按“s”或“S”保存。（可多次保存）



1. 查看图片

图片直接生成在application文件夹下：



1. 替换图片

打开data文件夹删除并替换。（不需要改图片名字）

（其中字体文件和txt文件是文字滤镜的组件，也可以自定义替换）



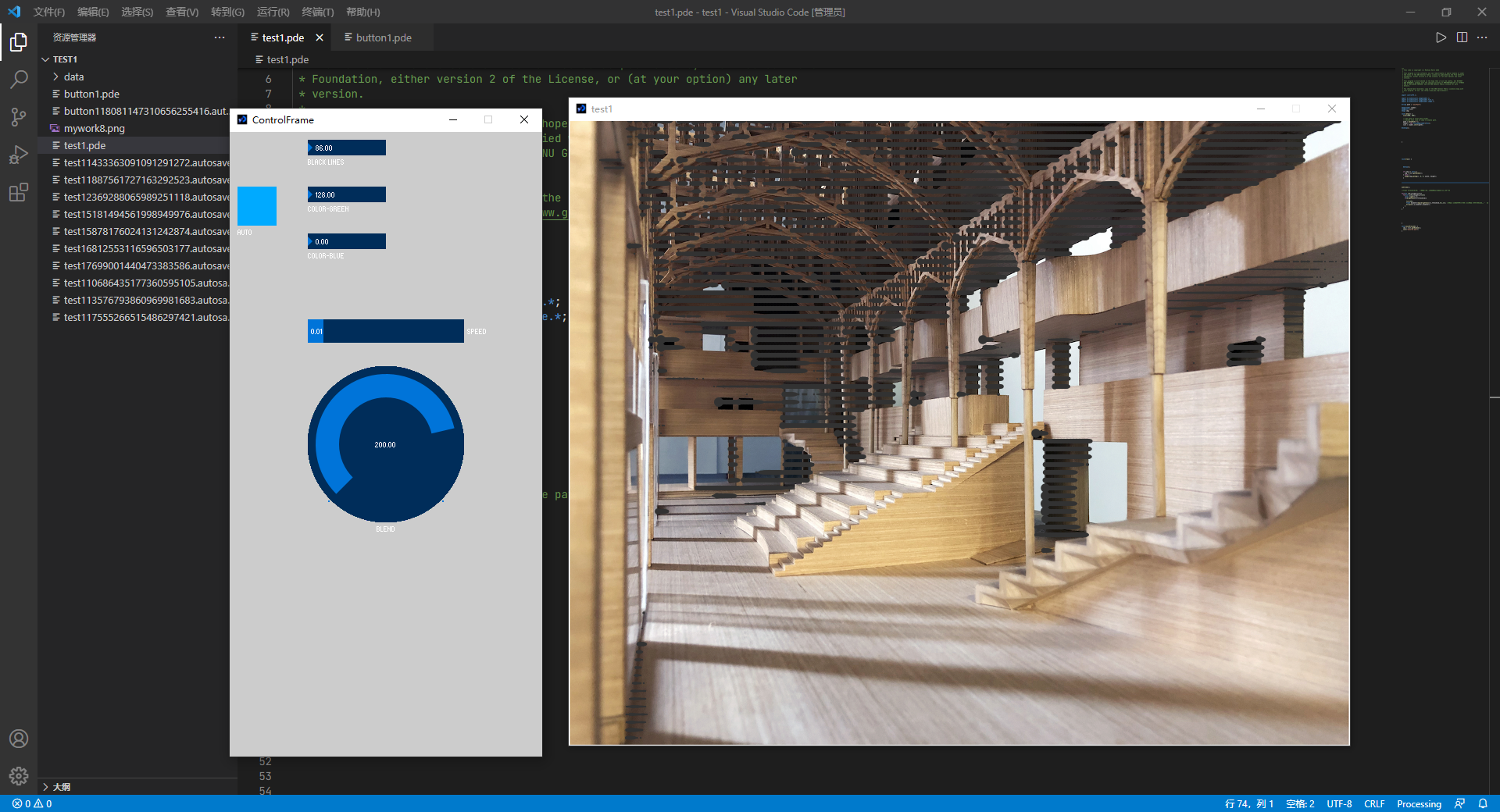
# 开发过程

## 开发计划

第一天：立下目标，写下整个计划书框架，整合已有的资源，找到能够利用的点，思考着之后怎么才能将已有的作品利用上去。看了不少他人的作品。

第二天：琢磨了好几个小时，最后发现一个叫controlp5的GUI库，非常好用。

想着需要撰写整个程序框架，从最简单的一个功能做起，搭建一个功能的UI，并用库中的函数将UI与图片窗口分开，方便调节。于是把之前的一次作业与controlp5库结合，尝试用第一个拖拽控件控制生成点阵的阈值，成功！



第三天：将功能按钮优化，每一个滑动条都配有相应的开启开关。至此，大的框架已经搭建完毕，之后开始创建各种滤镜功能。

第四天：完成滤镜的整合和UI的指定和更改。

第五天：至此，任务基本完成。完成作业报告。

# 参考文献资料

1. 图片库：ImageLoader  
   [Javadocs: ImageLoader](file:///E:\Processing\libraries\ImageLoader\reference\index.html)  
   UI库：controlP5  
   [Javadocs: controlP5](file:///E:\Processing\libraries\controlP5\reference\index.html)
2. 文件相关代码编写：

[Welcome to Processing! / Processing.org](https://processing.org/)

[The Nature of Code](https://natureofcode.com/book/)

（3）部分滤镜参考：  
[Reference / Processing.org](https://processing.org/reference/filter_.html)