M2R（蒙赛尔色彩与RGB色彩转换）Ver:alpha 0.2 开发文档

其实我也不知道这个算不算开发文档，算是做完这个东西的一个阶段性总结吧。把构思，制作的过程还有有哪些还需要改进记录一下。

一、前言

M2R，姑且说这个软件吧，主要是用来进行蒙赛尔色彩与RGB色彩之间的转换和以一种较为直观的方式进行查询。

之所以会产生做这么个东西的想法，是在写城市美学论文的时候，因为要写关于色彩的东西，就像找找关于蒙赛尔的资料。结果找了半天也没有，最后还是翻墙，在国外网站上找到一份表格，但是表格只有数字，很难直观的进行比较和查询。所以就想用这份表格做一个可视化查询工具，于是在寒假就有了这个东西。

二、开发工具

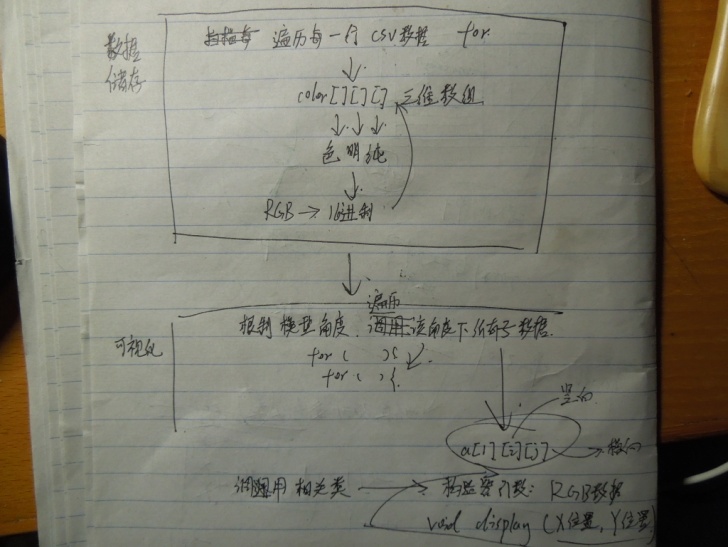
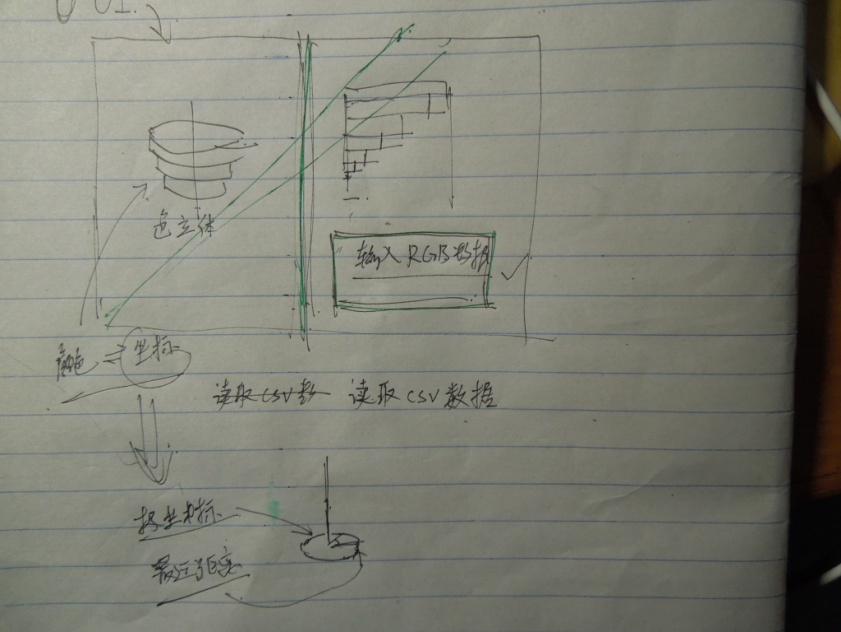
开发平台Processing

建模工具Grasshopper

三、开发过程

前期构思

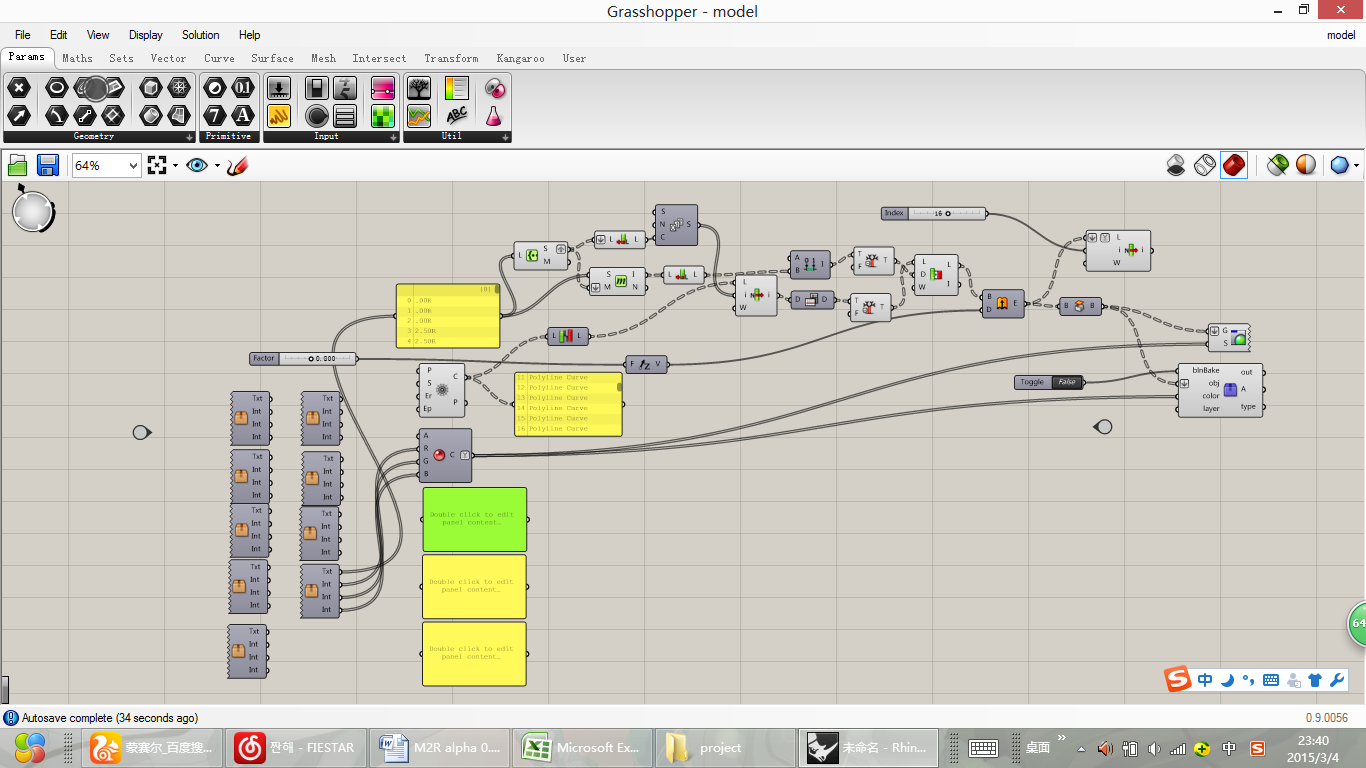
确定了总体功能：交互和查询



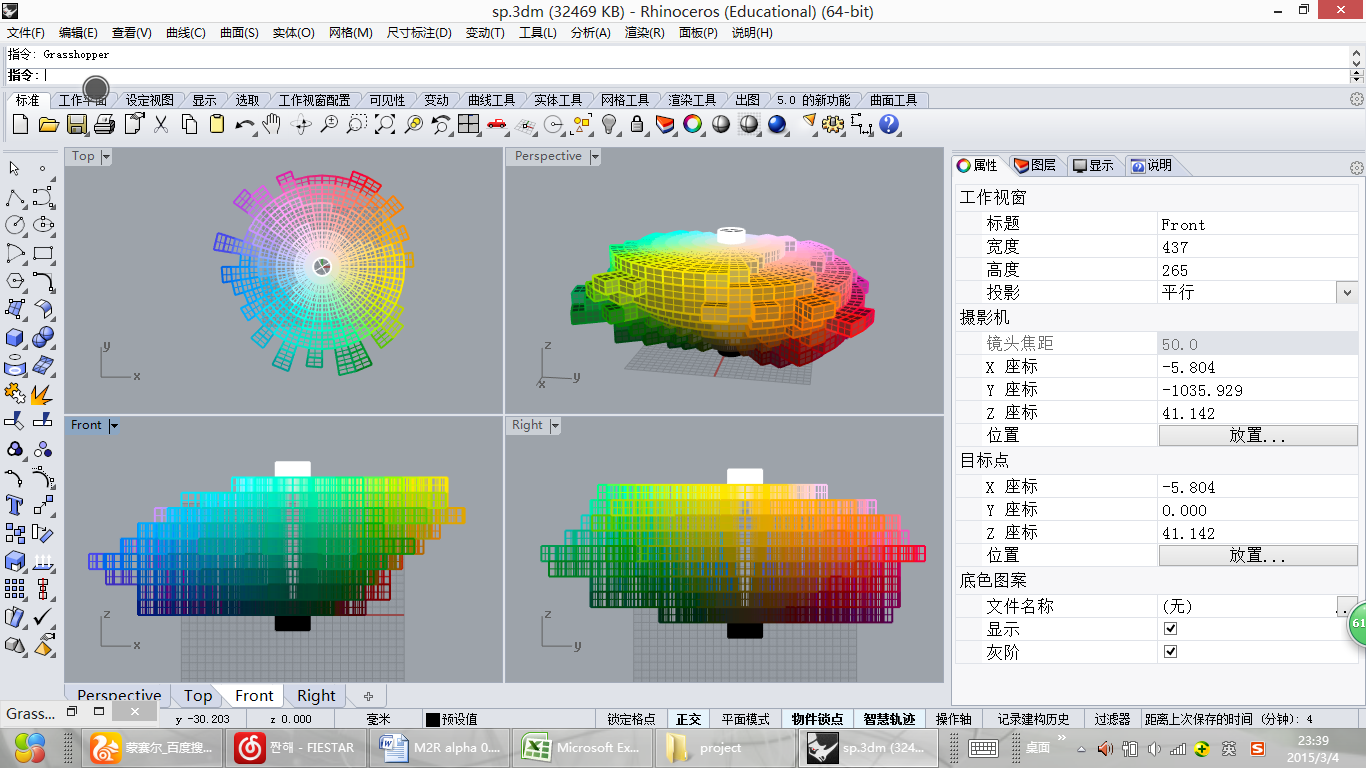
模型构建

考虑到蒙赛尔色立体并不是规则几何体，如果根据表格数据，纯手工建模工作量大而且容易出错。因此采用Grasshopper进行参数化建模。将数据按明度分为9个部分，分别导入构建好的Grasshopper电池图中，最后将9个部分手工组合即可。

电池图：



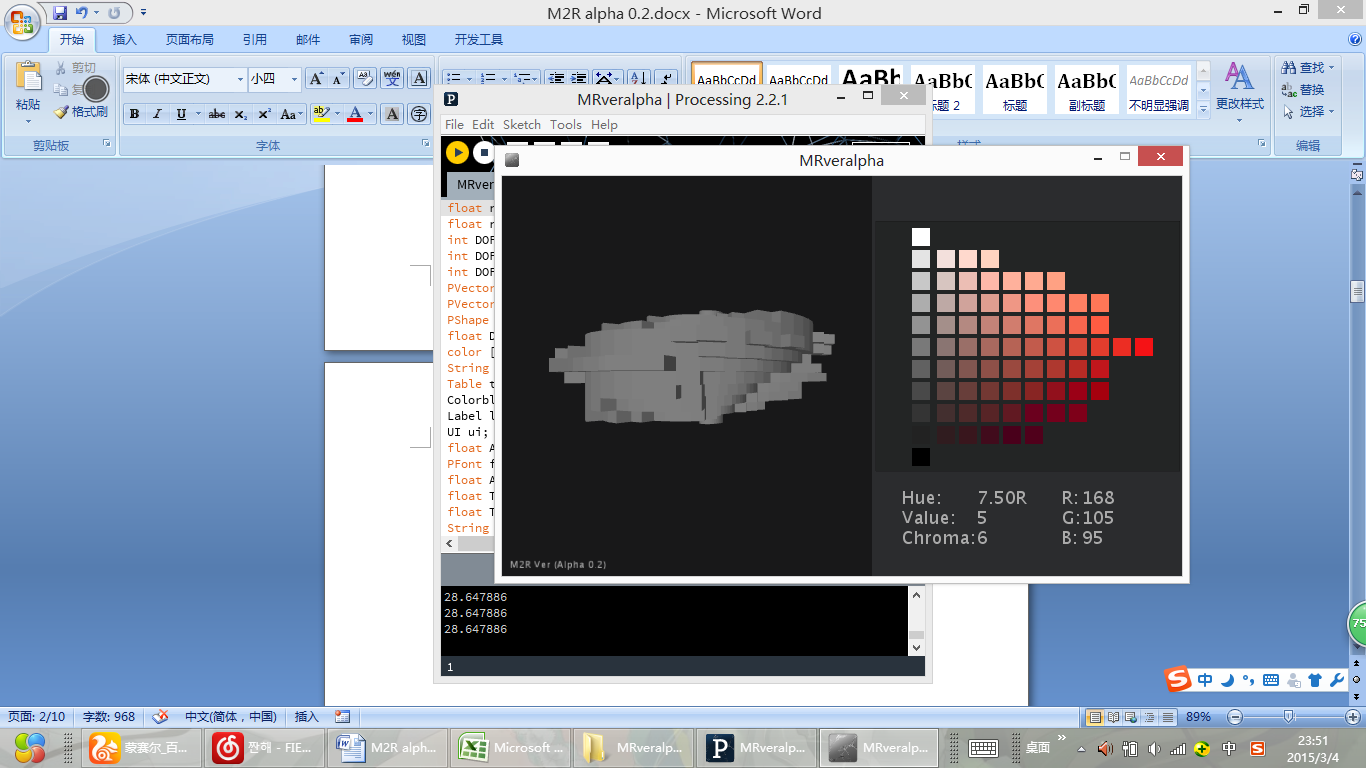
成品：

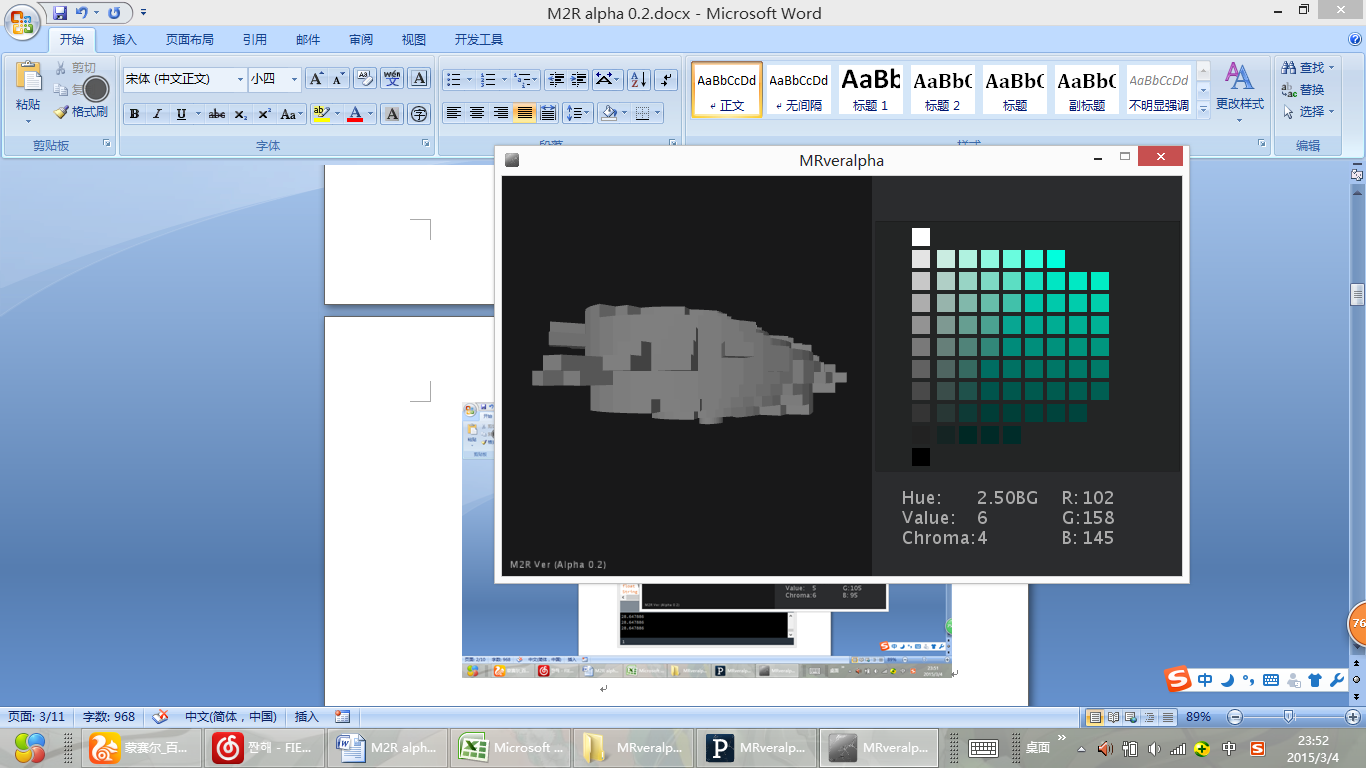


代码实现

在Processing中进行主要功能编写，代码部分在后面。

最初版本——M2R Ver:alpha 0.2





四、问题与优化

问题

1、可以看到这个版本的模型没有上色，是个白模。在最初的版本中确实是想做成彩色的，但是在建模过程中发现，grasshopper中的上色电池只能给网格上色，并不是上材质。因此最终导出的obj文件也没有材质。

2、数据过少，表格中除了RGB色彩还有别的数据，没有充分利用。

3、UI设计过于单调。

4、没有完全实现查询功能，只能由蒙赛尔到RGB，反向则还需要人工比较。但是由于从RGB与蒙赛尔色彩之间并不是线性关系，因此还需要查阅相关资料。

优化

1、增加模型的材质，修复模型视角畸变问题。

2、增加更多数据类型。

3、增加物理阻尼效果。

五、附录：

表格版权

Munsell Conversion - Version 6.5.9 Copyright ? 2006 WallkillColor Table can be used for personal and educational but not for commercial purpose without license from WallkillColor

代码部分

float rotx=0;

float roty=0;

int DOF=15;

int DOFmax=270;

int DOFmin=-50;

PVector origenXaxis;

PVector changeXaxis;

PShape munsell;

float Degree;

color [][][]iro = new color[40][9][12];

String []Hue =new String[40];

Table table;

Colorblock blocks;

Label label;

UI ui;

float Angleshifting=0;

PFont f;

float Angle=9;

float TranslateX=435;

float TranslateY=250;

String temphue;

void setup() {

size(680, 400, P3D);

f=loadFont("BankGothicBT-Light-48.vlw");

munsell = loadShape("sp.obj");

loadDate();

rectMode(CENTER);

label=new Label();

ui=new UI();

}

void draw() {

stroke(0);

background(24, 24, 25);

ui.display();

cy();

label.display();

lights();

pushMatrix();

translate(215, height/2+35, DOF);

//camera(-200, 0, (height/2.0) / tan(PI\*30.0 / 180.0),

//0, 0, 0, 0, 1, 0);

rotateX(PI/2);

//rotateX(constrain(rotx,-0.15,0.12));

rotateZ(roty);

stroke(0);

fill(255);

//box(50);

shape(munsell);

//println(DOF);

//coords();

popMatrix();

Degree=degree(roty, Angleshifting);

for (int j=0; j<iro[0].length; j++) {

for (int k=0; k<iro[0][0].length; k++) {

if (iro[int(Degree/Angle)][j][k]!=0) {

blocks = new Colorblock(iro[int(Degree/Angle)][j][k], Hue[int(Degree/Angle)], k, j);

blocks.display();

blocks.mousePositionCheck();

//println(iro[int(Degree/Angle)][j][k]);

}

}

}

//println(degrees(rotx), "+", degrees(roty));//propmt

println(Degree);

}

void mouseDragged() {

float rate=-0.02;

rotx+=(pmouseY-mouseY)\*-rate;

roty+=(pmouseX-mouseX)\*-rate;

}

float degree(float cumulateY, float angel) {

float n = degrees(cumulateY)+angel;

cumulateY=n-int(n/360)\*360;

if (n<0) {

cumulateY = n+360\*int(abs(n/360)+1);

}

return cumulateY;

}

/\*void keyPressed() {

if (DOF<=DOFmax) {

if (key=='z') {

DOF+=10;

}

}

if (DOF>=DOFmin) {

if (key=='x') {

DOF-=10;

}

}

}\*/

/\*void coords() {

stroke(0);

stroke(255, 0, 0);

line(0, 0, 0, 500, 0, 0);

stroke(0, 255, 0);

line(0, 0, 0, 0, 200, 0);

stroke(0, 0, 255);

line(0, 0, 0, 0, 0, 500);

}\*/

void cy() {

pushMatrix();

translate(TranslateX, TranslateY);

noStroke();

fill(0);

rect(-25, 22, 18, 18);

.

.

fill(255);

rect(-25, -198, 18, 18);

popMatrix();

}

void loadDate() {//input the data

table=loadTable("M2R.csv", "header");

int rowCount=0;

for (TableRow row : table.rows ()) {

int chroma=int(row.getFloat("Chroma"));

int value=int(row.getFloat("Value"));

int angle=int(row.getFloat("Angle"));

int r=int(row.getFloat("R"));

int g=int(row.getFloat("G"));

int b=int(row.getFloat("B"));

iro[int(angle/Angle)][value-1][(chroma-2)/2]=color(r, g, b);

String hue = row.getString("Hue");

if (hue!=temphue) {

Hue[int(angle/Angle)]=hue;

}

String temphue = hue;

}

}

class Colorblock {

color Bcolor;

float positionX;

float positionY;

float rectlength=18;

float FontY=78;

float FontX=85;

float scaleX=22;

float scaleY=-22;

float TranslateX=435;

float TranslateY=250;

String Hue;

float Value;

float Chroma;

Colorblock(color tempC, String i, float x, float y) {

Bcolor=tempC;

positionX=x\*scaleX;

positionY=y\*scaleY;

Hue=i;

Value=y+1;

Chroma=2\*(x+1);

}

void display() {

noLights();

pushMatrix();

translate(TranslateX, TranslateY);

noStroke();

//fill(2\*red(Bcolor), 2\*green(Bcolor), 2\*blue(Bcolor));

fill(red(Bcolor), green(Bcolor), blue(Bcolor));

//println(red(Bcolor),green(Bcolor),blue(Bcolor));

rect(positionX, positionY, rectlength, rectlength);

popMatrix();

}

void mousePositionCheck() {

pushMatrix();

translate(TranslateX, TranslateY);

if ((mouseX-TranslateX)<positionX+rectlength/2&&

(mouseX-TranslateX)>positionX-rectlength/2&&

(mouseY-TranslateY)<positionY+rectlength/2&&

(mouseY-TranslateY)>positionY-rectlength/2) {

//println(red(Bcolor), green(Bcolor), blue(Bcolor));

textSize(18);

fill(170, 170, 170);

text(int(red(Bcolor)), FontX+60, FontY+0);

text(int(green(Bcolor)), FontX+60, FontY+20);

text(int(blue(Bcolor)), FontX+60, FontY+40);

text(Hue, FontX-45, FontY+0);

text(int(Value), FontX-45, FontY+20);

text(int(Chroma), FontX-45, FontY+40);

}

popMatrix();

}

}

class Label {

float TranslateX=435;

float TranslateY=250;

float FontY=78;

float FontX=125;

Label() {

}

void display() {

noLights();

pushMatrix();

translate(TranslateX, TranslateY);

noStroke();

textSize(18);

fill(170, 170, 170);

text("R: ", FontX, FontY+0);

text("G: ", FontX, FontY+20);

text("B: ", FontX, FontY+40);

text("Hue: ", FontX-160, FontY+0);

text("Value: ", FontX-160, FontY+20);

text("Chroma: ", FontX-160, FontY+40);

popMatrix();

noStroke();

textSize(10);

fill(170, 170, 170);

text("M2R Ver (Alpha 0.2)",8,392);

}

}

class UI {

float LeftX;

float LeftY;

float LeftLength;

float LeftHeight;

float DiagramX;

float DiagramY;

float DiagramLength;

float DiagramHeight;

UI() {

LeftX=width/2+30;

LeftY=0;

LeftLength=width/2-30;

LeftHeight=height;

DiagramX=width/2+34;

DiagramY=45;

DiagramLength=width/2-36;

DiagramHeight=250;

}

void display() {

rectMode(CORNER);

noStroke();

fill(43, 44, 46);

rect(LeftX, LeftY, LeftLength, LeftHeight);

strokeWeight(0.25);

stroke(10);

fill(35,37,37);

rect(DiagramX,DiagramY,DiagramLength,DiagramHeight);

}

}