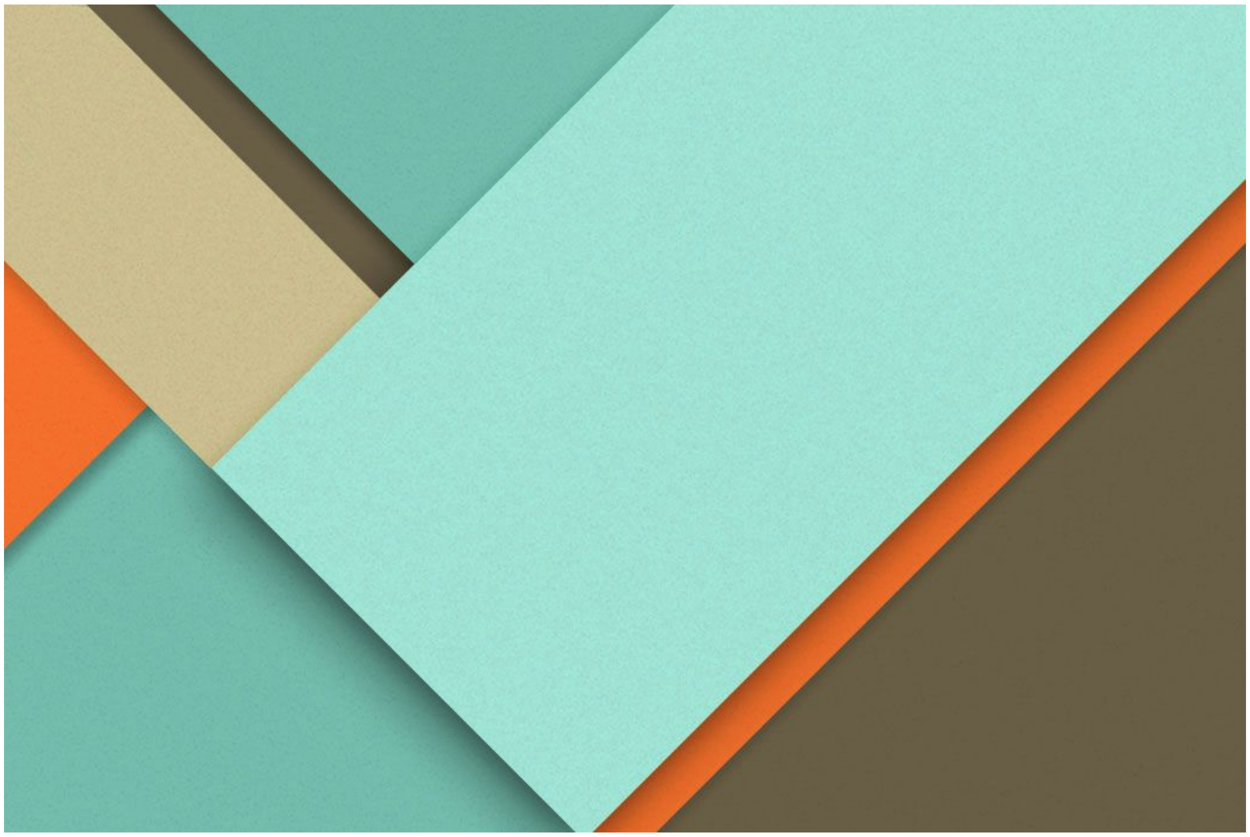

CS 837 - Assignment 2

15.10.2015

Trevor Tomesh
University of Regina - Computer Science
tmtomesh@gmail.com
Student No. 200343884

Overview

This submission contains an information visualization program written in Processing 3.0. The program:

- Reads an external CSV file and stores it within a custom class in processing.
- Allows users to render single-variable data in different methods including:
    - Position X
    - Position Y
    - Size
    - Brightness
    - Color
- Renders x-y data read as points on a scatter-plot.
- Provides the user with interactivity tools to explore the data including:
    - X and Y zoom
    - Point zoom
    - Panning
    - Ability to toggle through points:
        - “a” and “d” keys to move between points
        - “r” key to return to the first point

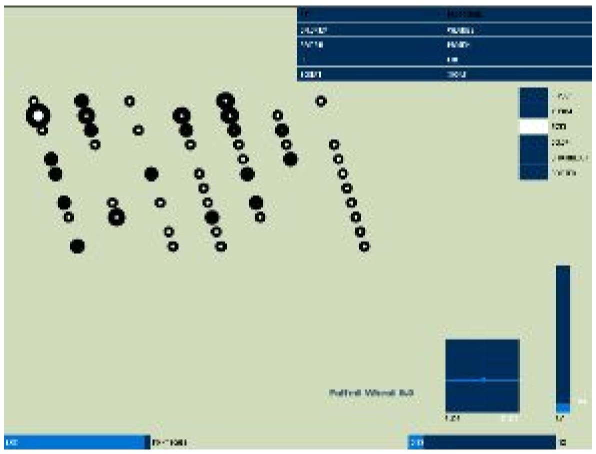
Fig. 1 - Working Prototype of the processing information viz. software.

Goals

1. Read data-file once, store data in internal structure.
2. Render data as different visualizations:
    a. Position-x
    b. Position-y
    c. Mark
    d. Size
    e. Brightness
    f. Color
3. Provide interface for user to select vizualization method.

4. Simple “focus” operation to allow selection of individual points.

Critique

This program was written in only two days, however, it manages to address the user requirements (with the exception of the “mark” rendering) and includes an x-y scatter plot tool with independent variable selection.

It is clear and easy from this software to tell what sort of visualizations work and what sort of methods should be avoided given the data set.

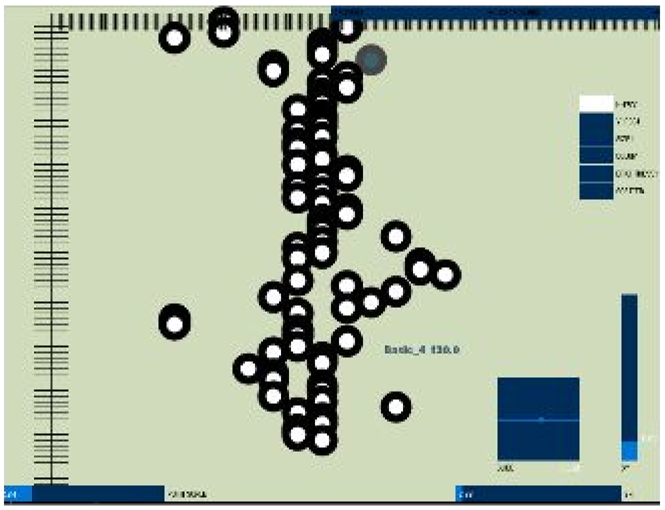

Figure 2 - A good visualization of caloric data. The x-value corresponds with the amount of calories in the individual cereal and. The y-value is just iterative.

Good and Bad Visualization

This software is good to illustrate what sorts of vizualizations are useful and what sorts are not good for the data set. For example, positional data (X or Y) is very good to illustrate numeric data like caloric information. To find relationships between variables, the x-y scatter-plotting capability is excellent. However, when dealing with value like caloric information, using brightness is fairly useless.

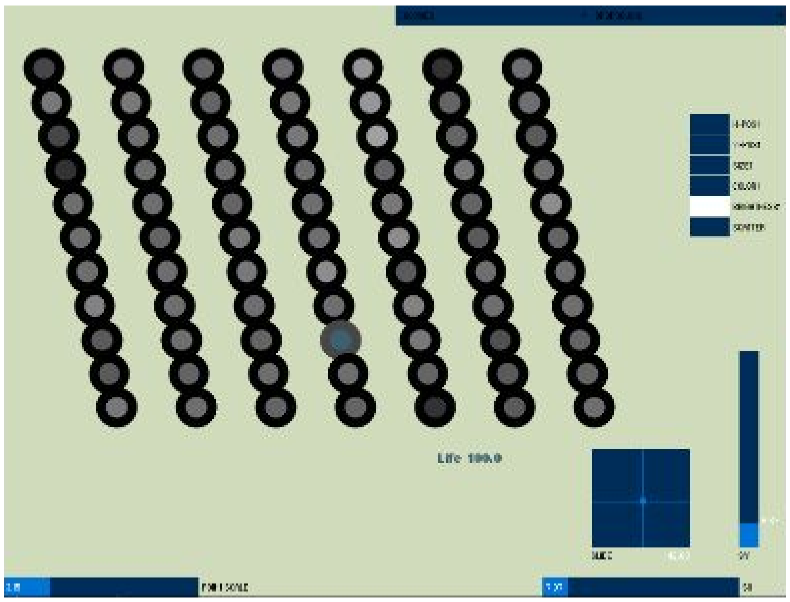

Figure 3 - A bad vizualization of caloric information. The positional data is meaningless and the whiter the point the higher the calories per serving. It’s very difficult to pick which is highest and lowest calorically speaking.

Known Issues / Planned Features

As of now, there are a number of issues both major and minor with the software:

-
- No method for displaying “mark data”.
- The origin is in the upper left-hand corner rather than the lower left-hand corner. This is because of the way that processing renders its coordinate system. A simple transform could fix this.
- The user should eventually be able to select the data by clicking and dragging.
- Some scaling issues.

Running the software

I. From Source

- The source code for this project can be found at: https://github.com/trevortomesh/cs837/tree/master/assignment2
- Download the latest version of processing: https://processing.org/download/
- Either copy and paste the code from the .pde file or open it through the file-menu in processing.
- Take care the the csv file is in the same directory as the sketch.
- Import the ControlP5 library by clicking Sketch > Import Library> Add Library
- Press “run”.

II. From executable

- Executables are provided for both windows and linux versions of the software (os x pending)
- Within the “assignment2” folder, open “build” and navigate to the folder for your operating system.
- Execute your binary.

Note: You MUST have the CSV file in the same directory as your executable! Also, the CSV file has been re-formatted from the original file to conform with accepted CSV protocol.

Source Code

```
import controlP5.*;
import java.util.*;
ControlP5 cp5;
Slider2D s;
ControlP5 control1, control2, slidex, slidey, scalex, scaley, pointSize;
String ctrl1, ctrl2;
int c1, c2;
float v1, v2, sx, sy, ps;
RadioButton r1, r2;

int selectIndex = 0;

Table rawData;

float drawPointSize = 10;
int windowWidth = 800;
int windowHeight = 600;
Cereal[] cereals;
int index;

int offsetx = 10;
int offsety = 10;

int[] scale = {50, 50};

void settings(){
  size(windowWidth,windowHeight);

}

void setup(){
  rawData = loadTable("cereal.csv", "header");
  println(rawData.getRowCount() + " total rows in table");
```

```
cereals = new Cereal[rawData.getRowCount()];
int index = 0;

for (TableRow row : rawData.rows()) {
  cereals[index] = new Cereal(
  row.getString("name"),
  row.getString("mfr"),
  row.getString("type"),
  float(row.getString("calories")),
  float(row.getString("protein")),
  float(row.getString("fat")),
  float(row.getString("sodium")),
  float(row.getString("fiber")),
  float(row.getString("carbo")),
  float(row.getString("sugars")),
  float(row.getString("shelf")),
  float(row.getString("potass")),
  float(row.getString("vitamins")),
  float(row.getString("weight")),
  float(row.getString("cups")));
  index++;
}


cp5 = new ControlP5(this);
r1 = cp5.addRadioButton("radioButton1")
       .setPosition(windowWidth-100,110)
       .setSize(40,20)
       .setColorForeground(color(120))
       .setColorActive(color(255))
       .setColorLabel(color(0))
       .setItemsPerRow(1)
       .setSpacingColumn(50)
       .addItem("x-pos1",0)
       .addItem("y-pos1",1)
       .addItem("size1",2)
       .addItem("color1",3)
       .addItem("brightness1",4)
       .addItem("scatter",5);
       ;
 /*
 cp5 = new ControlP5(this);
 r2 = cp5.addRadioButton("radioButton2")

       .setPosition(windowWidth-200,250)
       .setSize(40,20)
       .setColorForeground(color(120))
       .setColorActive(color(255))
       .setColorLabel(color(255))
       .setItemsPerRow(1)
       .setSpacingColumn(50)
       .addItem("x-pos2",1)
       .addItem("y-pos2",2)
       .addItem("mark2",3)
       .addItem("size2",4)
       .addItem("color2",5)
       .addItem("brightness2",6);*/


   pointSize = new ControlP5(this);
   pointSize.addSlider("ps")
       .setPosition(0, windowHeight-20)
       .setSize(200, 20)
       .setRange(1, 10)
       .setValue(5)
       .setColorCaptionLabel(color(0))
       .setCaptionLabel("Point Scale");

scaley = new ControlP5(this);
   scaley.addSlider("sy")
       .setPosition(windowWidth - 50, windowHeight - 250)
       .setSize(20, 200)
```

```
        .setRange(1, 50)
        .setValue(30)
        .setColorCaptionLabel(color(0));


    scalex = new ControlP5(this);
   scalex.addSlider("sx")
        .setPosition(windowWidth - 250, windowHeight - 20)
        .setSize(200, 20)
        .setRange(1, 50)
        .setValue(30)
        .setColorCaptionLabel(color(0));

cp5 = new ControlP5(this);
s = cp5.addSlider2D("slide")
        .setPosition(windowWidth - 200, windowHeight - 150)
        .setSize(100,100)
        .setMinMax(-1000,-1000,1000,1000)
        .setValue(0,0)
        .setColorCaptionLabel(color(0));

        //.disableCrosshair()
        ;

 smooth();

  List l1 = Arrays.asList("calories","protein","fat","sodium","fiber","carbohydrates",
 "sugars","shelf","potassium","vitamins","weight","cups");

 control1 = new ControlP5(this);
 control1.addScrollableList("dropdown1")
    .setPosition(windowWidth-400,0)
    .setSize(200, 100)
    .setBarHeight(20)
    .setItemHeight(20)
    .setColorCaptionLabel(color(1))
    .addItems(l1);



  control2 = new ControlP5(this);
  control2.addScrollableList("dropdown2")
    .setPosition(windowWidth-200, 0)
    .setSize(200, 100)
    .setBarHeight(20)
    .setItemHeight(20)
    .setColorCaptionLabel(color(1))
    .addItems(l1);





}

void draw(){
  clear();
  background(#D1DBBD);
  update();
  //scatterPlot();
  }



void update(){
  if(r1.getArrayValue()[0] == 1){
    xPlot();
    drawAxies();

  }

  if(r1.getArrayValue()[1] ==1){
    yPlot();
      drawAxies();
```

```
 }

 if(r1.getArrayValue()[2] ==1){

  sizePlot();

 }

 if(r1.getArrayValue()[3] == 1){
  colorPlot();

 }

 if(r1.getArrayValue()[4] == 1){
  brightPlot();
 }

   if(r1.getArrayValue()[5] == 1){
   scatterPlot();
   drawAxies();


}
}



void radioButton(int a) {
 println("a radio Button event: "+a);
}



void drawAxies(){
 stroke(0);
 fill(#000000);
 strokeWeight(1);
 pushMatrix();
```

```
translate(s.getArrayValue()[0],s.getArrayValue()[1]);
line(0,0,2000, 0);
line(0, 0, 0, 2000);

strokeWeight(1);




for(float line = 0; line < 1000; line+= 1){
line(float(-20), line*sy, 20, line*sy);
}

for(float line = 0; line < 1000; line+=1){
  line(float(-10), line*sy, 10, line*sy);
}

for(float line = 0; line < 1000; line+=1){
  line(line*sx, -10, line*sx, 10);
}
popMatrix();

 }

void keyPressed(){

if(key == 'a'){
  selectIndex--;
}

if(key == 'd'){
 selectIndex++;
}

if(key == 'r'){
 selectIndex = 0;
}
```

```
}

void renderFont(String name, float value){
PFont drawFont = loadFont("Arial-Black-12.vlw");
String toDraw = name + ' ' + value;
    fill(#3E606F);
    textFont(drawFont);
    text(toDraw, 400, 400);
}
//---------------------------------------------------------------------
void scatterPlot(){
  //float maxx = 0;
  //float maxy = 0;
 // println(r1.getArrayValue()[6] == 1);

  for(int i = 0; i < cereals.length; i++){
    stroke(0);
    strokeWeight(10);
    fill(#FFFFFF);
    pushMatrix();
    translate(s.getArrayValue()[0],s.getArrayValue()[1]);
    ellipse(cereals[i].properties[c1]*sx,cereals[i].properties[c2]*sy, drawPointSize*ps,
drawPointSize*ps);
    popMatrix();

 }
   }

void xPlot(){
  stroke(0);
   strokeWeight(10);
   for(int i = 0; i < cereals.length; i++){
    fill(#FFFFFF);
```

```
      pushMatrix();
      translate(s.getArrayValue()[0],s.getArrayValue()[1]);
     if(selectIndex == i){
        stroke(70);
        renderFont(cereals[i].name,cereals[i].properties[c1]);
      }
      else{
       stroke(0);
      }
      //scale(sx,sy);
      ellipse(cereals[i].properties[c1]*sx,i*sy, drawPointSize*ps, drawPointSize*ps);
      popMatrix();
    }

}

void yPlot(){
  stroke(0);
  strokeWeight(10);
    for(int i = 0; i < cereals.length; i++){
      fill(#FFFFFF);
      pushMatrix();
      translate(s.getArrayValue()[0],s.getArrayValue()[1]);
            if(selectIndex == i){
            renderFont(cereals[i].name,cereals[i].properties[c1]);

        stroke(70);
      }
      else{
       stroke(0);
      }
      //scale(sx,sy);
      ellipse(i*sx,cereals[i].properties[c1]*sy, drawPointSize*ps, drawPointSize*ps);
      popMatrix();
    }
```

```
}




void sizePlot(){
      stroke(0);
      strokeWeight(10);
      int j = 0;
      for(int i = 0; i < cereals.length; i++){
      //color c = color(map(cereals[i].properties[c1],0,255,0,255),
map(cereals[i].properties[c1],0,255,0,255), map(cereals[i].properties[c1],0,255,0,255));
      fill(255);
      pushMatrix();
      translate(s.getArrayValue()[0],s.getArrayValue()[1]);
            if(selectIndex == i){
        stroke(70);
                renderFont(cereals[i].name,cereals[i].properties[c1]);

      }
      else{
       stroke(0);
      }
      //scale(sx,sy);
      ellipse(i*sx,j*sy, ps*cereals[i].properties[c1]/2,ps*cereals[i].properties[c1]/2);
      popMatrix();
      if(j < 50){
        j = j+5;}
       else{ j = 0;}

    }




}
```

```
void brightPlot(){

      stroke(0);
      strokeWeight(10);
      int j = 0;
      for(int i = 0; i < cereals.length; i++){
      color c = color(map(cereals[i].properties[c1],0,255,0,255), map(cereals[i].properties[c1],0,255,0,255),
map(cereals[i].properties[c1],0,255,0,255));
      fill(c);
      pushMatrix();
      translate(s.getArrayValue()[0],s.getArrayValue()[1]);

      if(selectIndex == i){
        stroke(70);
               renderFont(cereals[i].name,cereals[i].properties[c1]);

      }
      else{
       stroke(0);

      }
      //scale(sx,sy);
      ellipse(i*sx,j*sy, ps*drawPointSize, ps*drawPointSize);
      popMatrix();

      if(j < 50){
        j = j+5;}
       else{ j = 0;}

    }

}

void colorPlot(){
  stroke(0);
  strokeWeight(10);
  int j = 0;
      for(int i = 0; i < cereals.length; i++){
      color c = color(map(cereals[i].properties[c1],0,255,0,255), 0,
255-map(cereals[i].properties[c1],0,255,0,255));
      fill(c);

      pushMatrix();
      translate(s.getArrayValue()[0],s.getArrayValue()[1]);

      if(selectIndex == i){
        stroke(70);
               renderFont(cereals[i].name,cereals[i].properties[c1]);

      }
      else{
       stroke(0);
      }

      ellipse(i*sx,j*sy, ps*drawPointSize, ps*drawPointSize);
      popMatrix();

      if(j < 50){
        j = j+5;}
       else{ j = 0;}

    }


}


//------------------------------------------------------------------
void dropdown1(int n) {
  control1.get(ScrollableList.class, "dropdown1").getItem(n);
   c1 = (int) control1.get(ScrollableList.class, "dropdown1").getItem(n).get("value");
//   println(c1,c2);
```

```
}

void dropdown2(int n){
control2.get(ScrollableList.class, "dropdown2").getItem(n);
c2 = (int) control2.get(ScrollableList.class, "dropdown2").getItem(n).get("value");
}



//-----------------------------------------------------------------------
class Cereal {
  String name, mfr,type;
  float calories,protein,fat,sodium,fiber,carbohydrates,sugars,shelf,potassium,vitamins,weight,cups;
  float[] properties = {0,0,0,0,0,0,0,0,0,0,0,0};

  // Here's a rediculously large constructor to accept all the properties of a cereal object.
  Cereal(String tmpname, String tmpmfr, String tmptype,
  float tmpcalories, float tmpprotein,float tmpfat,
  float tmpsodium,float tmpfiber,float tmpcarbo,float tmpsugars,
  float tmpshelf, float tmppotass, float tmpvitamins, float tmpweight, float tmpcups) {

  name = tmpname;
  mfr = tmpmfr;
  type = tmptype;
  calories = tmpcalories;
  protein = tmpprotein;
  fat = tmpfat;
  sodium = tmpsodium;
  fiber = tmpfiber;
  carbohydrates = tmpcarbo;
  sugars = tmpsugars;
  shelf = tmpshelf;
  potassium = tmppotass;
  vitamins = tmpvitamins;
  weight = tmpweight;
  cups = tmpcups;
```

```
 properties[0] = calories;
 properties[1] = protein;
 properties[2] = fat;
 properties[3] = sodium;
 properties[4] = fiber;
 properties[5] = carbohydrates;
 properties[6] = sugars;
 properties[7] = shelf;
 properties[8] = potassium;
 properties[9] = vitamins;
 properties[10] = weight;
 properties[11] = cups;

 }
}
```