```
//
// Code by Alex Bullock & Ryan Swart for NM Supercomputing 2013
//

import controlP5.*;

ControlP5 controlP5;

int sliderValue = 100;
int ssize = 200;
int xsize = 91;
int ysize = 144;
int rectwidth = 40;
int rectheight = 20;
int co2minSlider = 300;
int co2maxSlider = 385;
int year1slider;
int year2slider;
int year1 = 2012;
int year2 = 2004;
int month1 = 02;
int month2 = 02;
int choice = 2; // 1 = our first heatmap, 2 = AIRS approximate heat map
int onoffslider = 1; //
```

```
float sqsize = 8.0;

float newval;

float co2[][] = new float [xsize][ysize];

float co2count[][] = new float [xsize][ysize];

float lat[][] = new float [xsize][ysize];

float longa[][] = new float [xsize][ysize];

float co22[][] = new float [xsize][ysize];

float co2count2[][] = new float [xsize][ysize];

float lat2[][] = new float [xsize][ysize];

float longa2[][] = new float [xsize][ysize];

float co2diff[][] = new float [xsize][ysize];

float co2countdiff[][] = new float [xsize][ysize];

float latdiff[][] = new float [xsize][ysize];

float longadiff[][] = new float [xsize][ysize];

float latval, latmin, latmax, longaval, longamin, longamax;

float co2min, co2max, co2countmin, co2countmax;

float latval2, latmin2, latmax2, longaval2, longamin2, longamax2;

float co2min2, co2max2, co2countmin2, co2countmax2;

float latvaldiff, latmindiff, latmaxdiff, longavaldiff, longamindiff, longamaxdiff;

float co2mindiff, co2maxdiff, co2countmindiff, co2countmaxdiff;

float NaN=-9999.; // indicates bad data

float co2mindraw = 360e-6;

float co2maxdraw = 400e-6;

float co2mindraw2 = 360e-6;
```

```
float co2maxdraw2 = 400e-6;

float co2mindrawdiff = -100e-6;

float co2maxdrawdiff = +100e-6;

float mapGeoLeft   = -180.0;          // Longitude 180 degrees west

float mapGeoRight  =  180.0;          // Longitude 180 degrees east

float mapGeoTop    =   90.0;          // Latitude 90 degrees north.

float mapGeoBottom =  -90.0;          // Latitude 90 degrees south.

float mapScreenWidth, mapScreenHeight;  // Dimension of map in pixels.

float val;                     //  hold co2[i][j] for speed-up inside draw loop


//||||||||XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXX

boolean calcdiff = true;//takes the difference of two datasets if true

//||||||||XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXX

boolean ignoreMaxMinSlider = false;



PVector pos[][]=new PVector [xsize][ysize];

PFont font;

PImage backgroundMap;




Gradient g;



void setup()
```

```
{
 size(1200, 700);
 smooth();
 font = createFont("Arial", 16, true);
 textFont(font, 18);
 controlP5 = new ControlP5(this);
 if (! calcdiff) {
  controlP5.addSlider("co2minSlider", 300, 385, 365, 500, 615, 200, 25);
  controlP5.addSlider("co2maxSlider", 370, 450, 390, 500, 650, 200, 25);
  //from left to right(name of slider, minimum value, maximum value
  //number slider starts at, X position, Y position, width, length)
 }
 else {
  controlP5.addSlider("co2minSlider", -20, 0, -10, 500, 615, 200, 25);
  controlP5.addSlider("co2maxSlider", 0, 20, 10, 500, 650, 200, 25);
  //from left to right(name of slider, minimum value, maximum value
  //number slider starts at, X position, Y position, width, length)
 }
 // controlP5.addSlider("year1slider", 2003, 2012, 800, 615, 180, 25);
 //  Slider s1 = (Slider)controlP5.controller("year1slider");
 //  s1.setNumberOfTickMarks(10);
 //  controlP5.addSlider("year2slider", 2003, 2012, 800, 650, 180, 25);
 //  Slider s2 = (Slider)controlP5.controller("year2slider");
 //  s2.setNumberOfTickMarks(10);
 // create colors for heat map
```

```
g = new Gradient();
if (choice == 1) {
  g.addColor(color(0, 0, 0));
  // g.addColor(color(102, 0, 102));
  g.addColor(color(0, 144, 255));
  g.addColor(color(0, 255, 207));
  g.addColor(color(51, 204, 102));
  g.addColor(color(111, 255, 0));
  g.addColor(color(191, 255, 0));
  g.addColor(color(255, 240, 0));
  g.addColor(color(255, 153, 102));
  g.addColor(color(204, 51, 0));
  g.addColor(color(153, 0, 0));
}
else if (choice == 2) { // our approx of heat map used by NASA for AIRS
  g.addColor(color(109, 214, 236));
  g.addColor(color(114, 249, 232 ));
  g.addColor(color(123, 255, 216 ));
  g.addColor(color(135, 253, 159 ));
  g.addColor(color(157, 255, 118 ));
  g.addColor(color(195, 255, 71 ));
  g.addColor(color(229, 255, 37 ));
  g.addColor(color(248, 229, 28 ));
  g.addColor(color(226, 140, 21 ));
  g.addColor(color(210, 64, 17 ));
```

```
  g.addColor(color(209, 28, 17 ));
}

//backgroundMap   = loadImage("world.jpg");
  backgroundMap   = loadImage("earth3.jpg");
//backgroundMap   = loadImage("earth4.png");
//backgroundMap   = loadImage("earth6.png");
// backgroundMap   = loadImage("earth7.jpg");**DONT USE
mapScreenWidth  = width;
mapScreenHeight = height;

//set up data path and file names

String foldername="";
String latname=foldername + str(year1)+"." + nf(month1, 2) + "/lat.csv";
String longaname=foldername + str(year1)+"." + nf(month1, 2) + "/longa.csv";
String co2countname=foldername + str(year1)+"." + nf(month1, 2) + "/co2count.csv";
String co2name=foldername + str(year1)+"." + nf(month1, 2) + "/co2.csv";

String foldername2="";
String latname2=foldername2 + str(year2)+"." + nf(month2, 2) + "/lat.csv";
String longaname2=foldername2 + str(year2)+"." + nf(month2, 2) + "/longa.csv";
String co2countname2=foldername2 + str(year2)+"." + nf(month2, 2) + "/co2count.csv";
String co2name2=foldername2 + str(year2)+"." + nf(month2, 2) + "/co2.csv";
```

```
//read latitude data and calc max and min of lat
println("read lat file " +latname);
String[] data = loadStrings (latname);

//latmin= 9999
//latmax=-9999
for (int i = 0; i< data.length; ++i) {
  String[] pieces = split(data[i], ',');
  for (int j = 0; j< pieces.length; ++j) {
    //println("i,j= "+ str(i) +"," + str(j));
    lat[i][j] = float(pieces[j]);
  }
}
dataMaxMin(lat, latmax, latmin, 0);

//read longitude data and calc max and min of longa
println("reading longa file " +longaname);
data = loadStrings(longaname);
for (int i = 0; i< data.length; i++) {
  String[] pieces = split(data[i], ',');
  for (int j = 0; j< pieces.length; j++) {
    longa[i][j]  = float(pieces[j]);
  }
}
```

```
dataMaxMin(longa, longamax, longamin, 0);



//store lat/long to PVector:pos
// convert to x,y screen location using geoToPixel function
for (int i = 0; i< xsize; i++) {
  for (int j = 0; j< ysize; j++) {
    pos[i][j] = geoToPixel(new PVector(longa[i][j], lat[i][j]));
  }
}



//read co2 data and calc max and min of co2
println("read co2 file " +co2name);
data = loadStrings (co2name);
for (int i = 0; i< data.length; i++) {
  String[] pieces = split(data[i], ',');
  for (int j = 0; j< pieces.length; j++) {
    co2[i][j] = float(pieces[j]);
  }
}
dataMaxMin(co2, co2max, co2min, 1);

//read co2count data and calc max and min of co2count
println("read co2count file " +co2countname);
```

```
data = loadStrings (co2countname);
for (int i = 0; i< data.length; i++) {
  String[] pieces = split(data[i], ',');
  for (int j = 0; j< pieces.length; j++) {
    co2count[i][j] = float(pieces[j]);
  }
}
dataMaxMin(co2count, co2countmax, co2countmin, 0);
println("done with data file 1");



//read second lat datafile if calcdiff is true
if (calcdiff) { // start of calcdiff = true
  println("reached calcdiff = true in data read part");
  //read latitude data and calc max and min of lat
  //println("read lat file " +latname2);
  // String[] data = loadStrings (latname2);

  //latmin= 9999
  //latmax=-9999
  for (int i = 0; i< data.length; ++i) {
    String[] pieces = split(data[i], ',');
    for (int j = 0; j< pieces.length; ++j) {
      //println("i,j= "+ str(i) +"," + str(j));
```

```
    lat2[i][j] = float(pieces[j]);
  }
}



dataMaxMin(lat2, latmax2, latmin2, 0);



//  println("min lat = "+str(latmin));
//  println("max lat = "+str(latmax));




println("read longa file 2" +longaname2);
data = loadStrings (longaname2);
for (int i = 0; i< data.length; i++) {
  String[] pieces = split(data[i], ',');
  for (int j = 0; j< pieces.length; j++) {
    longa2[i][j]  = float(pieces[j]);
  }
}


dataMaxMin(longa2, longamax2, longamin2, 0);
```

```
println("read co2 file 2 " +co2name2);
data = loadStrings (co2name2);
for (int i = 0; i< data.length; i++) {
  String[] pieces = split(data[i], ',');
  for (int j = 0; j< pieces.length; j++) {
    co22[i][j] = float(pieces[j]);
  }
}
dataMaxMin(co22, co2max2, co2min2, 0);



println("read co2count file " +co2countname2);
data = loadStrings (co2countname2);
for (int i = 0; i< data.length; i++) {
  String[] pieces = split(data[i], ',');
  for (int j = 0; j< pieces.length; j++) {
    co2count2[i][j] = float(pieces[j]);
  }
}
dataMaxMin(co2count2, co2countmax2, co2countmin2, 0);

for (int i = 0; i < xsize; ++i) {
  for (int j = 0; j < ysize; ++j) {
```

```
      if ((co22[i][j] == NaN) || (co2[i][j] == NaN)) {
        co2diff[i][j] = NaN;
      }
      else {
        co2diff[i][j] = co22[i][j] - co2[i][j];
      }
      if ((co2count2[i][j] == NaN) || (co2count[i][j] == NaN)) {
        co2countdiff[i][j] = NaN;
      }
      else {
        co2countdiff[i][j] = co2count2[i][j] - co2count[i][j];
      }
    }
   }
  } // endif
}

void draw()
{

  image(backgroundMap, 0, 0, mapScreenWidth, mapScreenHeight);


  PVector p;
  if (! calcdiff) {
```

```
 if( ! ignoreMaxMinSlider ){
 co2mindraw = co2minSlider*1.0e-6;//Use slider value
 co2maxdraw = co2maxSlider*1.0e-6;//Use slider value
 // println("co2mindraw = "+str(co2mindraw));
 // println("co2maxdraw = "+str(co2maxdraw));
 }
 //    println("reached not calcdiff in draw()");
 for (int i = 0; i < xsize; ++i) {
  for (int j = 0; j < ysize; ++j) {
    val = co2[i][j];
    if ( isGoodData(val)) {
    newval=toColor(co2[i][j], co2mindraw, co2maxdraw);
    fill(g.getGradient(newval));
          }
          else {

    fill(0,0,0);//black
         }
          rect(pos[i][j].x, pos[i][j].y, sqsize, sqsize);

}
 }

 drawColors(10, 600, g, co2mindraw, co2maxdraw);
}
```

```
  else {
    // code for visualizing
    // co2diff = co2 - co22

    co2mindraw = co2minSlider*1.0e-6;//Use slider value
    co2maxdraw = co2maxSlider*1.0e-6;//Use slider value
    // println("co2mindraw = "+str(co2mindraw));
    // println("co2maxdraw = "+str(co2maxdraw));

    //    println("reached not calcdiff in draw()");
    for (int i = 0; i < xsize; ++i) {
      for (int j = 0; j < ysize; ++j) {
        newval=toColor(co2diff[i][j], co2mindraw, co2maxdraw);
        fill(g.getGradient(newval));
        rect(pos[i][j].x, pos[i][j].y, sqsize, sqsize);
      }
    }

    drawColors(10, 600, g, co2mindraw, co2maxdraw);
  }
}



class Gradient
{  // Class for heatmap
```

```
ArrayList colors;


// Constructor
Gradient()
{
  colors = new ArrayList();
}


void addColor(color c)
{
  colors.add(c);
}


color getGradient(float value)
{
  // make sure there are colors to use
  if (colors.size() == 0)
    return #000000;


  // if its too low, use the lowest value
  if (value <= 0.0)
    return (color)(Integer) colors.get(0);


  // if its too high, use the highest value
  if (value >= colors.size() - 1)
```

```
    return (color)(Integer) colors.get(colors.size() - 1);

   // lerp between the two needed colors
   int color_index = (int)value;
   color c1 = (color)(Integer) colors.get(color_index);
   color c2 = (color)(Integer) colors.get(color_index + 1);
   return lerpColor(c1, c2, value - color_index);
  }
}

float toColor(float val, float min, float max) {
  // return adjusted value between 0 and 10
  // given value val between min and max
  return 9.0*(val-min)/(max-min);
}

boolean isGoodData(float x) {
  // returns true if not very close to -9999.0
  if ( x > -9998.999 || x < -9999.0001 ) return true;
  else return false;
}

void dataMaxMin(float matrix[][], float maxval, float minval, int print) {
  // find max & min values, ignore bad data
  maxval= -9999;
```

```
  minval= 9999;
  for (int i = 0; i< matrix.length; ++i) {
    for (int j = 0; j< matrix[i].length; ++j) {
      if ( print>0 ) {
        // println("i,j= "+ str(i) +"," + str(j));
        // println( matrix[i][j]);
      }
      if ( isGoodData(matrix[i][j]) ) {
        if ( print>0 ) println("     good data!");
        if ( minval > matrix[i][j] )
          minval = matrix[i][j];
        if ( print>0 ) println("   changed min val = "+str(minval));
        if ( maxval < matrix[i][j] ) {
          maxval = matrix[i][j];
          if ( print>0 ) println("   changed max val = "+str(maxval));
        }
      }
    }
  }
  if ( print>0 ) println("min val = "+str(minval));
  if ( print>0 ) println("max val = "+str(maxval));
}

float dataMax(float matrix[][]) {
  // calc max of matrix while ignoring bad data
```

```
  float maxval= -9999;
  for (int i = 0; i< matrix.length; ++i) {
    for (int j = 0; j< matrix[i].length; ++j) {
      // println("i,j= "+ str(i) +"," + str(j));
      // println( matrix[i][j]);
      if ( isGoodData(matrix[i][j]) ) {
        //   println("     good data!");
        if ( maxval < matrix[i][j] ) {
          maxval = matrix[i][j];
          //println("   changed max val = "+str(maxval));
        }
      }
    }
  }
  println("max lat = "+str(maxval));
  return maxval;
}




// Converts screen coordinates into geographical coordinates.
// Useful for interpreting mouse position.
// following code written by Jo Wood
// http://forum.processing.org/user/jo-wood
public PVector pixelToGeo(PVector screenLocation)
```

```
{
  return new PVector(mapGeoLeft + (mapGeoRight-mapGeoLeft)*(screenLocation.x)/mapScreenWidth,
  mapGeoTop - (mapGeoTop-mapGeoBottom)*(screenLocation.y)/mapScreenHeight);
}

// Converts geographical coordinates into screen coordinates.
// Useful for drawing geographically referenced items on screen.
public PVector geoToPixel(PVector geoLocation)
{
  return new PVector(mapScreenWidth*(geoLocation.x-mapGeoLeft)/(mapGeoRight-mapGeoLeft),
  mapScreenHeight - mapScreenHeight*(geoLocation.y-mapGeoBottom)/(mapGeoTop-
mapGeoBottom));
}

void drawColors(int x, int y, Gradient g, float minval, float maxval) {
  // draw color bar representing values in color gradient
  pushMatrix();
  translate(x, y);
  fill(g.getGradient(0));
  rect(rectwidth-40, 0, rectwidth, rectheight);
  fill(g.getGradient(1));
  rect(rectwidth, 0, rectwidth, rectheight);
  fill(g.getGradient(2));
  rect(rectwidth+40, 0, rectwidth, rectheight);
  fill(g.getGradient(3));
  rect(rectwidth+80, 0, rectwidth, rectheight);
```

```
 fill(g.getGradient(4));
 rect(rectwidth+120, 0, rectwidth, rectheight);
 fill(g.getGradient(5));
 rect(rectwidth+160, 0, rectwidth, rectheight);
 fill(g.getGradient(6));
 rect(rectwidth+200, 0, rectwidth, rectheight);
 fill(g.getGradient(7));
 rect(rectwidth+240, 0, rectwidth, rectheight);
 fill(g.getGradient(8));
 rect(rectwidth+280, 0, rectwidth, rectheight);
 fill(g.getGradient(9));
 rect(rectwidth+320, 0, rectwidth, rectheight);
 //fill(g.getGradient(10));
 //rect(rectwidth+360, 0, rectwidth, rectheight);
 popMatrix();
 //writes text headings for color bar
 fill(0, 0, 0);
 text(nfc(1000000*co2mindraw, 0), 15, 640);
 text(nfc(1000000*co2maxdraw, 0), 375, 640);
 text(("CO2 ppm"), 200, 660);
 fill(180, 120, 120);
 strokeWeight(0.5);
}
```

```
class AIRSdata {
  // experimental, use merely as data package
  float co2[][] = new float [xsize][ysize];
  float co2count[][] = new float [xsize][ysize];
  float lat[][] = new float [xsize][ysize];
  float longa[][] = new float [xsize][ysize];
  //float co22[][] = new float [xsize][ysize];
  //float co2count2[][] = new float [xsize][ysize];
  //float lat2[][] = new float [xsize][ysize];
  //float longa2[][] = new float [xsize][ysize];
  //float co2diff[][] = new float [xsize][ysize];
  //float co2countdiff[][] = new float [xsize][ysize];
  //float latdiff[][] = new float [xsize][ysize];
  //float longadiff[][] = new float [xsize][ysize];
  float latval, latmin, latmax, longaval, longamin, longamax;
  float co2min, co2max, co2countmin, co2countmax;
  //float latval2, latmin2, latmax2, longaval2, longamin2, longamax2;
  //float co2min2, co2max2, co2countmin2, co2countmax2;
  //float latvaldiff, latmindiff, latmaxdiff, longavaldiff, longamindiff, longamaxdiff;
  //float co2mindiff, co2maxdiff, co2countmindiff, co2countmaxdiff;
  //float NaN=-9999.;
  //float co2mindraw = 360e-6;
  //float co2maxdraw = 400e-6;
  //float co2mindraw2 = 360e-6;
  //float co2maxdraw2 = 400e-6;
```

```
  //float co2mindrawdiff = -100e-6;

  //float co2maxdrawdiff = +100e-6;

}
```