```
//Visual Accompaniment for Suon Laulu
//Written by Brian Givens for Anne Yoncha, 2023
//Written with significant help from references and tutorials at processing.org
//I figured out the ffmpeg command with help from https://hamelot.io/visualization/using-ffmpeg-to-convert-a-set-of-images-into-a-video/

PImage imgOrig;
PImage source;
PImage img;
PGraphics pgImg;
int loc, locB;
color colorC, colorB;
color colorCompC, colorCompB;
int numFrames, numDir, effect, blurStart;
int xOffset, yOffset;
int compC, compB;
int[] xOffsets = {0, -1, -1, -1,  0,  1, 1, 1};
int[] yOffsets = {1,  1,  0, -1, -1, -1, 0, 1};
float dispH, dispRatio;
boolean saveOutputImages;


void setup () {
  size(800,800);
  //Set output image height in pixels Original image is 8222 pixels wide
  //Must be even for video creation
  dispH = 720;

  //Set output image ration (1.33 for 4:3, 1.78 for 16:9)
  dispRatio = 16.0/9.0;

  //Creates a lower res display window for screen
  windowResize(800,int(800/dispRatio));

  //Creates img file where work will happen, with specific size
  img = createImage(int(dispH*dispRatio),int(dispH),RGB);


  //Loads file used for sorting (Source) and file that will be modified and viewed (imgOrig)
  source = loadImage("Peatland extraction soil core sample rotated.jpg");
  imgOrig = loadImage("Sphagnum stem rotated.jpg");

  //If you want to save frames as PNG files, set to True
  saveOutputImages = true;

  //Resizes loaded image files to match specified resolution
  //Displayed image will just match width, so it will not be distorted and will have
  //black bars on top and bottom. This assumes image will always be wider and shorter than
```

```
  //display screen (which is the case with the orgiinal imager here).
  imgOrig.resize(img.width,0);
  source.resize(img.width,img.height);


  //Create PGraphics large canvas and load original image in center of space.
  //Area outside of the image is black
  pgImg = createGraphics(int(dispH*dispRatio),int(dispH));
  pgImg.beginDraw();
  pgImg.background(color(0,0,0));
  pgImg.imageMode(CENTER);
  pgImg.image(imgOrig,int(pgImg.width/2),int(pgImg.height/2));
  pgImg.imageMode(CORNER);
  pgImg.endDraw();

  source.loadPixels();

  //Initialize variable to record number of frames we have generated (maybe could be replaced by system
variable frameCount?)
  numFrames = 0;
  //Initialize direction that the blurring algorithm looks in when sorting pixels
  numDir = 0;
  //Initilize the type of sorting used when sorting pixels
  effect = 4;


  //Load initial state of the PGraphics canvas to img
  img = pgImg.get();

  //blurStart controls the portion of the image that is being blurred currenty. The program starts 5 pixels
from the right edge
  blurStart = img.width-5;

  //Write img back into pGraphics canvas to create single combined image
  pgImg.beginDraw();
  pgImg.image(img,0,0);
  pgImg.endDraw();
}

void draw() {


 //Uses numDir variable and offsets arrays to specify distance in x and y that will be used to seelect pixel
to be compared
 //to the current pixel for sorting. There are 8 possible directions, up, down, left, right and the diagonals
in between each.
    xOffset = xOffsets[numDir];
    yOffset = yOffsets[numDir];
```

```
//First loop starts at blurStart position and loops x until it reachs the right edge of the image
  for(int x=blurStart; x < img.width; x++){
//Load the pixels of img for manipulation
    img.loadPixels();
//Loop through all y values, top to bottom
    for(int y=0; y < (img.height-1); y++){
//All pixels are numbered sequentially, so calculate the pixel index we are working on based on x and y coordinate
     loc = x + y*img.width;

        //The location in the source image to use for comparison with the display image is offset from the current pixel by the
        //amount of xOffset and yOffset. The constrain command makes sure the selected point of comparision doesn't fall outside
        //of the source image.
        locB = constrain((x - xOffset),blurStart,img.width-1) + constrain((y - yOffset),0,img.height-1)*img.width;


        //Grab the color values of the img from the orginal location and the comparison location determined by the offset above
        colorC = img.pixels[loc];
        colorB = img.pixels[locB];

        //Either compare the colors from the second image, source, or the original image.
        //If the comparison is numFrames%1, then it will use source every time. If it's numFrames%2 it will use source every other time, etc

        //Get the colors from the source image at the current pixel location and the comparision location determined above.
        colorCompC = source.pixels[loc];
        colorCompB = source.pixels[locB];

        //Use the colorEffect subroutine to get the values to compare for sorting
        compC = colorEffect(colorCompC, effect);
        compB = colorEffect(colorCompB, effect);

        //Compare the values from above. If the calue from the current location is less than the comparison location
        //switch the pixels in the display image from the current location and the comparison location.
        //Since we are basing this comparison on pixels from the cource image, but applying the result to the display image
        //this will create a blurring effect that appears random but it guided by the source image
        if (compC < compB) {
          img.pixels[loc] = colorB;
          img.pixels[locB] = colorC;
        }
```

```
      }
//Once changes are made to the img, update the pixels
       img.updatePixels();

    //Update the offscreen canvas with the current state of img
      pgImg.beginDraw();
      pgImg.image(img,0,0);

  }

  //Once all pixels in the current region have been updated save an image file of the current state
  //The file name will include the frame number, so we can make a movie of the images in order.
  if (saveOutputImages) {
    pgImg.save("sl-" + nf(frameCount) + ".png");
    //ffmpeg command for forward movie: ffmpeg -r 24 -f image2 -s 1280x720 -i sl-%d.png -vcodec
libx264 -crf 25 -pix_fmt yuv420p test500f.mp4
    //ffmpeg command for reverse movie: ffmpeg -r 24 -f image2 -s 1280x720 -start_number -1253 -i
sl%d.png -vcodec libx264 -crf 25 -pix_fmt yuv420p test_tenth_0826.mp4
    //The value after -r changes the framerate.
    //The value after -s is the resolution, it should match the image resolution
    //For the reverse movie, the value after -start_number should match the number of the last image
file.
    //The very last part is the output file name. Set it to whatever you want it to be called.
    //Run this command in the terminal, in the folder where the image files are located.
    //Tested on Mac, should work on Windows or Linux if ffmpeg is installed.
 }
    pgImg.endDraw();
  //Increment frame number by 1
  numFrames++;

  numDir = int(random(8));
  effect = int(random(7));

//   if (numFrames%10 == 0) {
//     image(img,0,0,width,height);
//   }

  //Determines number of times the blurred region will blur before expanding. %5 = 5 times, %1 = 1
time, etc
  if (numFrames%10 == 0) {
  //As long as blurStart is positive, decrease it by 1 (expanding region closer to left edge)
    if (blurStart > 0) {
      blurStart--;
      println("Blurred ", nf(float(img.width-blurStart)/float(img.width)*100,0,2), "% of screen");
  //Once the left edge is reached, stop the program
  } else {
      exit();
    }
```

```
  }


//   if (numFrames == 500) {
//     exit();
//   }
}

//Select how the blurring comparison will be made, based on the variable "effect"
//There are 7 possible numerical values that can be extracted from a pixel:
//alpha, brightness, hue and saturation and the red, green and blue components
//Once the type of comparison is selected, the appropriate value for the pixel is returned.

int colorEffect(color incColor, int effNum) {
  int eff=0;

  switch(effNum) {
  case 0:
    eff = int(alpha(incColor));
  case 1:
    eff = int(blue(incColor));
  case 2:
    eff = int(brightness(incColor));
  case 3:
    eff = int(green(incColor));
  case 4:
    eff = int(hue(incColor));
  case 5:
    eff = int(red(incColor));
  case 6:
    eff = int(saturation(incColor));
  }
  return eff;
}
```