CSC 480 Object-Oriented Multimedia Programming, Spring 2020

Dr. Dale E. Parson, Assignment 3, manipulating pixel-based "photos" of the canvas.
This assignment is due via **D2L Dropbox** **Assignment 3** due by **11:59 PM on Wednesday April 8**.
Updated from April 1 because of the one-week KU shutdown. 10% penalty for each day it is late.

See earlier assignment handouts for downloading & setting up Processing 3.x.
Make sure to set your Sketchbook directory to a persistent location at the start of each session to ensure that your work is not erased when you log out.

The handout starting code is at this link. Copy & paste into Processing and play with existing functionalities before you start coding.
https://faculty.kutztown.edu/parson/spring2020/CSC480PixelPhotosAssn3.txt

There are STUDENT comments in upper case every place that you have to write code. I have numbered them in approximate order of difficulty. You will need to load a small PImage file (typically .png or .jpeg) into your sketch for a paintbrush, and either create or load a PShape vector paintbrush, so you will have to ZIP & turn in your entire sketch directory. Here are the STUDENT requirements and points.

0. ENTER STUDENT NAME HERE: I will deduct points if your name is missing.

1. **(15%)** Add the 't' and 'l' (letter ell) keyboard commands for triangle and line per the following comments.

```
   't' triangle (STUDENT must add this:
      startx,starty, endx,endy, (startx+endx)/2, Y_BASED_ON_THESE_4_VARS
   'l' line (STUDENT add from startx,starty to endx,endy, thick strokeWeight)
```

```
boolean isBrushKey(char key) {
  // ellipse, rectangle, canvas, Tinted canvas
  // STUDENT must add some brush types here and in class brush.display()
```

This requirement also requires changes to method Note.display() for the brush type 't' and 'l'.

2. **(15%)** Complete the code in keyPressed() for the 'SNN.nn' canvas-scaling command.

'SNN.nn' for scaling canvas, may be -, STUDENT MUST COMPLETE

```
      // STUDENT MUST IMPLEMENT SCALE BASED ON value
      value = Float.parseFloat(cmdbuffer.substring(1));
```

Look at the code for UP and DOWN arrows to figure out what variable(s) to change in your keyPressed().

3. **(15%)** Complete the code for the 'M' command to set paintbrushes in motion.

```
   'Mxx.yy' Set all current Brushes in motion, where xx is x speed,
     yy is y speed, and either may be 0 or negative.
     Just running 'M\n' sets to random x & y speeds. STUDENT MUST COMPLETE.
```

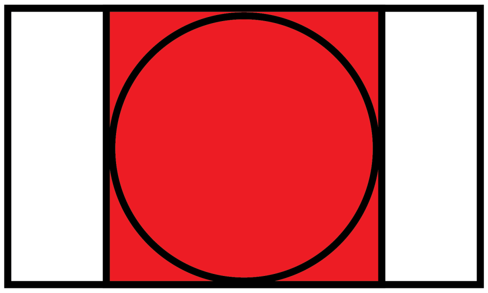
// STUDENT: Call setSpeed() on every Brush object in brushes.

4. **(20%)** Update method Cropper.run() that runs in a worker thread per this comment, required:

```
      // STUDENT IS REQUIRED TO IMPLEMENT THIS PORTION OF THIS METHOD.
      // Copy pixels in the square centered in the incoming into outgoing.
      // Leave the outside margins transparent - they already are.
      // Use nested for loops *similar* (not identical) to above block of code.
```

This is required to implement the ‘q’ (fixed from ‘Q’) command to set the recursive paintbrush image to a square instead of a circle.

'q' Make the recursive paintbrush for 'c' and 'T' a square ~~'Q' makes clipping mask a centered square~~



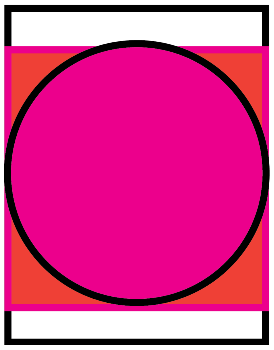


Illustration of Requirement 4 discussion from March 25 Zoom session

5. (15%) In setup() call loadImage on your image file, storing the PImage reference in this global variable:

PImage STUDENTIMAGE = null ;

This is part of the implementation of the ‘i’ command:

'i' image (PImage), STUDENT add, shape takes same args as ellipse()

This also requires enhancements to methods isBrushKey(char key) and Note.display() similar to ‘t’ above. Once completed, typing the ‘i’ command will display your PImage as a Brush object.

6. (20%) In setup call either a function that you write to return a custom 2D vector PShape, or use loadShape() to load a 2D .svg file into your sketch, storing your PShape here:

PShape STUDENTSHAPE = null ;

This is part of the implementation of the ‘v’ command:

'v' vector (PShape), STUDENT add, shape takes same args as ellipse()

This also requires enhancements to methods isBrushKey(char key) and Note.display() similar to ‘t’ above. Once completed, typing the ‘v’ command will display your PShape as a Brush object.

Note – You may **NOT** just do something like this using built-in PSHAPEs corresponding to simple shapes:

STUDENTSHAPE = createShape(BUILTIN)

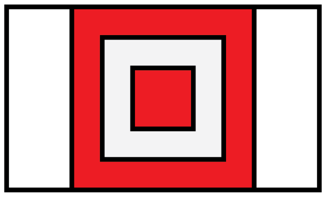
where BUILTIN is one of ELLIPSE, RECT, ARC, TRIANGLE, SPHERE, BOX, QUAD, or LINE.

You may stack several of these to a GROUP PShape and store that reference in STUDENTSHAPE. You may also create a custom 2D PShape using vertices, see function makeCustomPShape() in sketch
https://faculty.kutztown.edu/parson/fall2019/CSC220F19Demo3Da.txt
for an example of building a 3D GROUP PShape using nested 2D PShapes. DO NOT TEXTURE YOUR PShape.

A final requirement is that the 0,0 reference point for your 2D PShape must be at the center of your displayed PShape when using a command like shape(STUDENTSHAPE, 0, 0, w, h) in Note.display(), so that shapeMode(CENTER) appears as expected. I will deduct 10 points if it is off center from the mouse-swept region that defines its extents. See the 3D statement custom.translate(100,100,0) in makeCustomPShape() linked above if you need to re-center your PShape at creation time. This step may also be necessary for a loaded SVG file; some SVG files are offset from their centers. If you need to translate the shape itself, so it before its first use, not repeatedly in Brush.display(). You can test whether your PShape is centered correctly by watching your mouse sweep closely while in the 'v' command.

7. You can earn 10 bonus points by correctly completing this functionality in Cropper.run() that we will discuss in class. This is not a requirement. Make sure that everything else works, and save a copy of that, before you tackle this option. If you tackle this, make sure to add comments about it right after your name at the top of the sketch.

// STUDENT 10 bonus points to implement E.NN.nn.etc as documented.



E.33.66.100
(white is transparent)
bonus points from 3/25 Zoom session

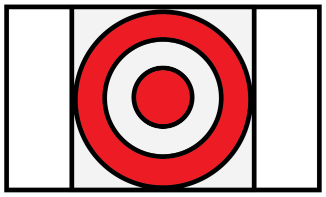


Illustration of bonus point discussion from March 25 Zoom session

The nested white regions in this illustration would actually be transparent in the cropped paintbrush