 master ▼


...

[PEmbroider](#) / PEmbroider_Cheat_Sheet.md



golanlevin Update to cheats

 History

 1 contributor

Raw Blame



188 lines (119 sloc) 7.03 KB

PEmbroider Cheat Sheet

Quick Reference Links

- Here is the detailed [API for PEmbroider](#), in progress.
- This document assumes that you're familiar with the [Processing drawing API](#).
- All of our examples can be browsed [here](#).

Overview of Startup, Drawing and Export

```
// Example PEmbroider program
import processing.embroider.*;
PEmbroiderGraphics E;

void setup() {

    // Starting up:
    noLoop();
    size(800, 600);
    E = new PEmbroiderGraphics(this, width, height);
    String outputPath = sketchPath("filename.dst");
    E.setPath(outputFilePath);
    E.beginDraw();
    E.clear();

    //-----
    // Content goes here:
    E.fill(0, 0, 0);
    E.circle(200, 200, 200);

    //-----
    // Visualization and export:
    // NOTE: Leave optimize() and endDraw() commented out
    // until you are ready to export!
    //
    // E.optimize(); // VERY SLOW, but essential for file output!
    E.visualize();
}
```

```
// E.endDraw(); // Write out the embroidery file
}
```

The `optimize()` function is **slow**, as it takes time to calculate a more effective order for the machine to stitch the embroidery. It does not change the appearance of your onscreen embroidery preview, but it will slow you down, so generally it does not need to be called when you're still designing your sketch.

Calling `visualize()` is the same as calling `visualize(false, true, false)`. The first variable, `false` by default, shows you a full color preview. The second variable, `true` by default, shows stitch ends. The third variable, `false` by default, shows you a preview of connecting stitches, which you will have to manually remove after your design is embroidered.

Shapes

PEmbroider tries to mirror the Processing drawing API. It includes commands for elements including lines, circles, ellipses, rects, arcs, triangles, quads, and arbitrary polygons with both straight and curved sides.

See Examples: [shapes](#), [shape_hatching_1](#), [shape_hatching_2](#), [shape_hatching_3](#), [shape_hatching_4](#), [shape_culling](#)

```
E.circle(x, y, radius)
E.rect(x, y, width, height)
E.triangle(x1, y1, x2, y2, x3, y3);
E.line(x1, y1, x2, y2);
```

Composite Shapes

See Examples: [stroke_outlines](#), [stroke_outlines_2](#)

```
E.beginComposite();
  E.composite.circle(320, 250, 200);
  E.composite.circle(420, 250, 200);
  // add more, etc. ...
E.endComposite();
```

Fills & Hatching

*See Examples: [shape_hatching_1](#), [shape_hatching_2](#), [shape_hatching_3](#), [shape_hatching_4](#), [satin_hatching_1](#)

```
E.noFill();
```

Hatch Modes:

```
E.hatchMode(E.CONCENTRIC);
E.hatchMode(E.PARALLEL);
E.hatchMode(E.SATIN);
E.hatchMode(E.SPIRAL);
E.hatchMode(E.PERLIN);
E.hatchMode(E.CROSS);
```

Hatch Settings:

```
E.hatchSpacing(spacing); // sets the density of adjacent runs
E.hatchAngleDeg(angle); // sets the orientation for SATIN & PARALLEL
```

```
E.fill(R, G, B);          // sets your thread color
```

Strokes

See Examples: [lines_1](#), [lines_2](#), [stroke_outlines](#), [stroke_outlines_2](#), [interactive_demo_2](#) (drawing)

Stroke Modes:

```
E.noStroke();  
E.strokeMode( E.PERPENDICULAR );  
E.strokeMode( E.TANGENT);
```

Stroke Locations

```
E.strokeLocation(E.CENTER);  
E.strokeLocation(E.INSIDE);  
E.strokeLocation(E.OUTSIDE);
```

Stroke Settings

```
E.strokeWeight(width);  
E.strokeSpacing(spacing);  
E.stroke(R, G, B);
```

Text

See Examples: [Hello_PEmbroider](#), [text_1](#), [text_2](#), [text_3](#)

```
E.textSize(size);  
E.textAlign(CENTER); // LEFT, RIGHT, etc., just like Processing  
E.textFont(PEmbroiderFont.DUPLEX);  
E.text(string, x, y);
```

Images

PEmbroider can work with both bitmap and vector images.

See Examples: [bitmap_image_1](#), [bitmap_image_2](#), [bitmap_animation](#), [png_image_multicolor](#), [svg_image](#)

Rastr (bitmap) images must be black-and-white only, where white shapes indicate the graphics to be embroidered. Processing will expect your image assets to be located in the "data" folder of your sketch; for more information on this, see [here](#).

```
PImage myImage = loadImage("filename.png");  
E.fill(0,0,0);  
E.image(myImage, x, y);
```

Vector shapes:

```
PShape mySvgImage = loadShape("filename.svg");  
E.fill(0,0,0);  
E.shape(mySvgImage, 50, 50, 350, 350);
```

Experimentally, we have also begun using the TSP code to render photographs. See [this](#) and [this](#)

Other Really Useful Things!

Set hatch spacing with `hatchSpacing()`

As mentioned above, the `hatchSpacing()` function has a major impact on your design. It sets the distance between adjacent runs in hatch modes like SATIN, PARALLEL, and CONCENTRIC. The units are machine units (for our machine, this is 0.1mm).

```
E.hatchSpacing(3);
```

Set stitch properties with `setStitch()`

The `setStitch()` function allows you to set the following:

- the minimum stitch length (in machine units; for our machine, this is 0.1mm)
- the desired stitch length (in machine units)
- the amount of noise (0...1) affecting stitch length. (This can be helpful for dithering the stitches in fills, so that they don't all line up.)

See Examples: [shape_hatching_3](#)

```
E.setStitch(minLength, desiredLength, noise);
```

For SPIRAL hatching, `setStitch` behaves differently;

```
E.setStitch(desiredLength, minLength, noise);
```

Protect your lines with `repeatEnd()`

The `repeatEnd()` function is very helpful for detailed linework; it essentially ties a knot so your single-line designs don't fray.

See Examples: [ruler](#)

```
beginRepeatEnd(3);  
// draw some lines...  
endRepeatEnd();
```

Render Order

You can render a shape's stroke before or after its fill:

See Examples: [text_1](#)

```
E.SetRenderOrder(STROKE_OVER_FILL);  
E.SetRenderOrder(FILL_OVER_STROKE);
```