Processing Cheatsheet

color

www.navax.net.br/blog
www.processing.org/reference

Setting

background

```
(gray)
(gray, alpha)
(value1,value2,valu3)
(value1,value2,valu3, alpha)
(color)
(color,alpha)
(hex)
(hex,alpha)

gray   int or float: specifies a value between white and black
alpha  int or float: opacity of the background
value1       int or float: red or hue value (depending on the current color mode)
value2       int or float: green or saturation value (depending on the current color mode)
value3       int or float: blue or brightness value (depending on the current color mode)
color  color: any value of the color datatype
hex    int: color value in hexadecimal notation (i.e. #FFCC00 or 0xFFFFCC00)
```

colorMode

```
(mode)
(mode,range)
(mode,range1,range2,range3)
(mode,range1,range2,range3,range4)

mode   Either RGB or HSB, corresponding to Red/Green/Blue and Hue/Saturation/Brightness
range  int or float: range for all color elements
range1 int or float: range for the red or hue depending on the current color mode
range2 int or float: range for the green or saturation depending on the current color mode
range3 int or float: range for the blue or brightness depending on the current color mode
range4 int or float: range for the alpha

colorMode (rgb,255);
colorMode (HSB,100);
```

noFill()

fill

```
(gray)
(gray, alpha)
(value1,value2,valu3)
(value1,value2,valu3, alpha)
(color)
(color,alpha)
(hex)
(hex,alpha)

fill(153);
fill (#CCFFAA)
fill (0XFFCCFFAA) -
fill(204,102,0)
```

noStroke()

stroke

```
(gray)
(gray, alpha)
(value1,value2,valu3)
(value1,value2,valu3, alpha)
(color)
(color,alpha)
(hex)
(hex,alpha)

fill(153);
fill (#CCFFAA)
fill (0XFFCCFFAA) -
fill(204,102,0)
```

Creating & Reading

red()
green()
blue()

```
color c=color(0,126,255); // rgb
float value= red (c); //Sets “value” to 0
float value=color >> 16 & 0xFF; // Faster!
```

color

Creates colors for storing in variables of the color datatype.

```
color c1= color(102,102,0);
color c2= color (93,255,130,0) // alpha
color c3= color(0xFFFFCC00);
```

hue()
brightness()
saturation()

```
colorMode (HSB,255);
color c=color(0,126,255);
float value=brightness(c); // Sets “value” to “255”
```

blendColor

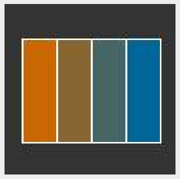
```
(c1,c2, MODE);

C1     color: the first color to blend
c2     color: the second color to blend
MODE   Either BLEND, ADD, SUBTRACT, DARKEST, LIGHTEST, DIFFERENCE, EXCLUSION, MULTIPLY,
SCREEN, OVERLAY, HARD_LIGHT, SOFT_LIGHT, DODGE, or BURN
```

lerpColor

```
(c1,c2, amt);
```

Calculates a color or colors between two color at a specific increment.

```
c1     color: interpolate from this color
c2     color: interpolate to this color
amt    float: between 0.0 and 1.0
```



```
stroke(255);
background(51);
color from = color(204, 102, 0);
color to = color(0, 102, 153);
color interA = lerpColor(from, to, .33);
color interB = lerpColor(from, to, .66);
fill(from);
rect(10, 20, 20, 60);
fill(interA);
rect(30, 20, 20, 60);
fill(interB);
rect(50, 20, 20, 60);
fill(to);
rect(70, 20, 20, 60);
```

shapes

2D Primitives

point
(x,y)

point
(x,y,z)

line
(x1,y1,x2,y2)

line
(x1,y1,z1,x2,y2,z2)

quad
(x1,y1,x2,y2,x3,y3,x4,y4)

rect
(x,y,width,height)

triangle
(x1,y1,x2,y2,x3,y3)

ellipse
(x,y,width,height)

arc
(x,y,width,height,start,stop)

Vertex

beginShape
(MODE)

MODE =Either POINTS, LINES, TRIAN-GLES, TRIANGLE_FAN, TRIANGLE_STRIP, QUADS, QUAD_STRIP

endShape()

vertex
(x,y)
(x,y,z)
(x,y,u,v)
(x,y,z,u,v)

bezierVertex
(cx1,cy1,cx2,cy2,x,y)
(cx1,cy1,cz1,cx2,cy2,cz2,x,y,z)

curveVertex
(x,y)
(x,y,z)

texture
(img)

textureMode
(MODE) IMAGE or NORMALIZED

Texture applied to vertex points:

```
noStroke();
PImage a = loadImage(“arch.jpg”);
textureMode(IMAGE);
 beginShape();
       texture(a);
       vertex(10, 20, 0, 0);
       vertex(80, 5, 100, 0);
       vertex(95, 90, 100, 100);
       vertex(40, 95, 0, 100);
endShape();
```

3D Primitives

box
(size)

box
(width,height,depth)

sphere
(radius)

sphereDetail
(res)
res   int: number of segments (minimum of 3) used per full circle revolution

sphereDetail
(ures, vres)

Curves

bezier
(x1,y1,cx1,cy1,cx2,cy2,x2,y2)

bezier
(x1,y1,z1,cx1,cy1,cz1,cx2,cy2,cz2,x2,y2,z2)

curve
(x1,y1,x2,y2,x3,y3,x4,y4)

curve
(x1,y1,z1,x2,y2,z2,x3,y3,z3,x4,y4,z4)

The first and second parameters specify the beginning control point and the last two parameters specify the ending control point.The middle parameters specify the start and stop of the curve.

Load & Displaying

Loads a vector shapes into a variable of type PShape.To load correctly, the file (SVG) must be located in the data directory of the current sketch.

```
PShape s;
s = loadShape(“bot.svg”);
smooth();
shape(s, 10, 10, 80, 80);
```

shape
(sh)
(sh,x,y)
(sh,x,y,width,height)

Processing Cheatsheet

Attributes

smooth()

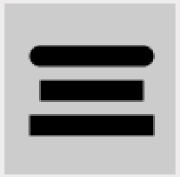
noSmooth()

ellipseMode
(MODE)
MODE   Either CENTER, RADIUS, CORNER, or CORNERS

rectMode
(MODE)
MODE   Either CORNER, CORNERS, CENTER or RADIUS

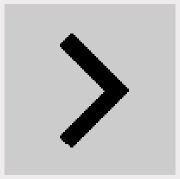
strokeWeight
(width)

strokeCap
(MODE)
MODE   Either SQUARE, PROJECT, or ROUND



```
smooth();
strokeWeight(12.0);
strokeCap(ROUND);
line(20, 30, 80, 30);
strokeCap(SQUARE);
line(20, 50, 80, 50);
strokeCap(PROJECT);
line(20, 70, 80, 70);
```

strokeJoint
(MODE)
MODE   Either MITER, BEVEL, or ROUND



```
noFill();
smooth();
strokeWeight(10.0);
strokeJoin(MITER);
beginShape();
vertex(35, 20);
vertex(65, 50);
vertex(35, 80);
endShape();
```